Е. П. Угрюмов

Цифровая схемотехника

3-е издание

- ИС всех уровней интеграции: от логических элементов до БИС/СБИС с программируемыми структурами, включая системы на кристалле
- Принципы построения цифровых узлов и устройств и их практическая реализация
- Современные методы проектирования цифровых средств обработки информации

bhv

Угрюмов Е. П.

Цифровая схемотехника

3-е издание

Рекомендовано Учебно-методическим объединением вузов РФ по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки 230100 «Информатика и вычислительная техника»

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.06
ББК 32.973.26-04я73
У27

Угрюмов Е. П.

У27 Цифровая схемотехника: учеб. пособие для вузов. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 816 с.: ил.

ISBN 978-5-9775-0162-0

Рассматриваются цифровые и аналоговые компоненты и структуры электронных систем обработки информации, являющиеся базой для создания разнообразной аппаратуры как в области вычислительной техники, так и в смежных областях: цифровой автоматике, измерительной технике, телекоммуникациях и т. д. Диапазон изучаемых вопросов — от уровня логических элементов до уровня простых микропроцессорных систем, в том числе систем на кристалле. Рассмотрены принципы и методика проектирования устройств обработки информации, в том числе с применением языка VHDL и его расширения VHDL-AMS, рассчитанного на разработку схем со смешанными сигналами.

В третьем, значительно обновленном, издании отражены новые достижения в области схемотехники, введены контрольные вопросы и задачи.

Для студентов технических вузов, аспирантов, инженеров и научных сотрудников, работающих в области создания цифровой аппаратуры

УДК 681.3.06
ББК 32.973.26-04я73

Рецензенты:

А. И. Водяхо, д. ф.-м. н., проф. кафедры вычислительной техники Санкт-Петербургского государственного электротехнического университета «ЛЭТИ»,
Кафедра автоматики и вычислительной техники Санкт-Петербургского государственного политехнического университета (завкафедрой д. т. н., проф. В. Ф. Мелехин),
С. Р. Иванов, к. т. н., доцент Московского технического университета им. Н. Э. Баумана

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Татьяна Лапина*
Зав. редакцией	*Григорий Добин*
Редактор	*Юрий Рожко*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Фото	*Кирилла Сергеева*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.04.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 65,79.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09
от 26.05.2009 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0162-0

© Угрюмов Е. П., 2010
© Оформление, издательство "БХВ-Петербург", 2010

Оглавление

Предисловие **1**

Введение **3**

Глава 1. Схемотехнические проблемы построения цифровых устройств **7**

§ 1.1. Модели и параметры логических элементов 7
Простейшая модель логического элемента 7
Сигналы, отображающие логические переменные 8
Учет задержек сигналов в логических схемах 9
Статические параметры логических элементов 10
Уровни напряжений и статическая помехоустойчивость логических элементов 10
Токовые параметры 11
Быстродействие цифровых элементов 12
Мощности потребления цифровых элементов 13
§ 1.2. Типы выходов цифровых элементов 15
Логический выход 16
Выходы с тремя состояниями 17
Открытые выходы 19
Программируемый выход 22
§ 1.3. Схемные особенности выводов КМОП-элементов 23
Pull-up- и Pull-down-резисторы 23
Выводы с запоминанием последнего значения сигнала 25
§ 1.4. Паразитные связи по цепям питания. Фильтрация питающих напряжений 26
Качество заземления 28
Фильтрация напряжений питания 29
§ 1.5. Передача сигналов. Помехи в сигнальных линиях. Сигнальные линии повышенного качества 30
Перекрестные помехи и электромагнитные наводки 30
Искажения сигналов в несогласованных линиях 31
Параллельное согласование волновых сопротивлений 33
Последовательное согласование волновых сопротивлений 37
Согласование волновых сопротивлений в конце и начале линии 38

Линии передачи сигналов 38
Линии связи с гальваническими развязками....41
Линии типа "токовая петля"....42
Стандарты сигналов ввода/вывода данных....42
Терминирование на кристалле....47
Банки ввода/вывода 47
Передача данных с двойной скоростью (технология DDR) 48
О разрядностях высокоскоростных шин 49
§ 1.7. Элементы задержки, формирования, обнаружения и генерации импульсов 51
Элементы задержки 51
Формирование импульсов по длительности 54
Разностные преобразователи и детекторы событий....54
Кольцевые генераторы 56
§ 1.8. Элементы визуальной индикации 57
Элементы индикации на светодиодах....57
Индикаторы на жидких кристаллах 59
§ 1.9. О некоторых типовых ситуациях 62
Режимы неиспользуемых входов 62
Согласование уровней сигналов при сопряжении разнотипных элементов....63
Режимы неиспользуемых элементов 65
Наращивание числа входов....65
Снижение нагрузок на выходах логических элементов....65
§ 1.10. Прошлое и настоящее малых и средних интегральных схем. Логические примитивы в системах автоматизированного проектирования 66
Контрольные вопросы и упражнения 69

Глава 2. Функциональные узлы комбинационного типа 73

§ 2.1. Проблематика проектирования комбинационных схем 73
Комбинационные цепи и автоматы с памятью 73
Риски сбоя....74
Сигналы синхронизации 75
Распространение сигналов в комбинационных цепях 76
Этапы разработки и средства реализации комбинационных цепей 77
Логические блоки табличного типа 78
Логические блоки с матрицами И и ИЛИ 79
Блоки на основе типовых логических элементов....79
§ 2.2. Двоичные дешифраторы 81
Схемотехническая реализация дешифраторов 83
Пример применения дешифратора....84
Воспроизведение логических функций 86

§ 2.3. Приоритетные и двоичные шифраторы. Указатели старшей единицы 87
§ 2.4. Мультиплексоры и демультиплексоры 91
Мультиплексоры 91
Мультиплексоры в КМОП-схемотехнике 92
Многоразрядные мультиплексоры 94
Наращивание размерности мультиплексоров 95
Демультиплексоры 95
Мультиплексоры и демультиплексоры в системах коммутации 97
§ 2.5. Универсальные логические модули на основе мультиплексоров 98
Первый способ настройки УЛМ 98
Второй способ настройки УЛМ 99
Структуры УЛМ, содержащие несколько мультиплексоров 101
§ 2.6. Компараторы 102
Сравнение на равенство 103
Сравнение на "больше" 104
Пример реализации компаратора 104
§ 2.7. Схемы контроля 105
Цели и задачи контроля 106
Мажоритарные элементы 106
Контроль по модулю 2 108
Схемы свертки 109
Передача данных с контролем по модулю 2 111
Контроль логического преобразователя 111
Контроль с использованием кодов Хемминга 112
Схемы кодера и декодера для кода Хемминга 115
§ 2.8. Сумматоры 116
Одноразрядный сумматор 116
Сумматор для последовательных операндов 119
Сумматор параллельных операндов с последовательным переносом 120
Сумматор с передачей сигнала переноса по цепочке замкнутых ключей 121
Сумматор параллельных операндов с параллельным переносом 123
Сумматоры групповой структуры 126
Сумматор с условным переносом 128
Микросхемы сумматоров 129
§ 2.9. Арифметико-логические устройства и блоки ускоренного переноса 130
§ 2.10. Матричные умножители 132
Множительно-суммирующие блоки 133
Наращивание размерности матричных умножителей 134
Схемы ускоренного умножения 136
Учет знаков сомножителей 139

§ 2.11. Быстрые сдвигатели 139
Сдвигатель, управляемый кодом "1 из N" 139
Сдвигатель, управляемый двоичным кодом 141
Контрольные вопросы и упражнения 142

Глава 3. Триггеры. Тактирование и синхронизация в цифровых устройствах 145

§ 3.1. Триггеры. Основные сведения. Внешнее поведение 145
Бистабильная ячейка 145
Простейший триггер 146
Классификация триггеров 147
Классификация триггеров по логическому функционированию 148
Классификация триггеров по способу приема информации 149
Тактирование уровнем. Режим прозрачности. Круговые гонки 152
Времена предустановки и выдержки 153
Метастабильные состояния триггеров 154
Способы описания триггеров 155
§ 3.2. Схемотехника триггерных устройств 158
Триггеры в биполярной схемотехнике 158
Простые RS-триггеры и защелки 158
Логические структуры триггеров T и JK 159
Двухступенчатые триггеры 161
Одноступенчатые триггеры, управляемые фронтом 163
Входы установки/сброса и разрешения тактирования 164
Триггеры в схемотехнике КМОП 165
Триггер-защелка 165
Двухступенчатый триггер 167
Примеры стандартных триггеров. Примитивы триггеров в системах автоматизированного проектирования цифровых устройств 167
§ 3.3. Примеры использования триггеров 170
Ввод логических сигналов от механических ключей 170
Синхронизаторы 172
Арбитры 173
§ 3.4. Тактирование и синхронизация. Общие сведения 175
Тактирование процессов 175
Системы с передачей в приемник тактовых сигналов 176
Выработка тактовых сигналов в приемнике данных 177
Синхронизация сигналов 177
§ 3.5. Тактирование сигналами, выработанными генератором 178
Общие сведения. Возможные решения 178
Концепции тактирования 178
Фазность тактирования 179

Разомкнутые и замкнутые системы тактирования 179
Медленные и быстрые сдвиги фаз ТИ 180
Обобщенный тракт обработки данных 180
Параметры тактовых импульсов 181
Длительности импульса и паузы 181
Стабильность частоты 182
Крутизна фронтов 182
§ 3.6. Структура и элементы систем тактирования 183
Структура системы тактирования 183
Кварцевые генераторы 184
Вторичные тактовые сигналы 187
Размножение тактовых импульсов 189
§ 3.7. Однофазное и двухфазное тактирование 190
Однофазное тактирование 190
Двухфазное тактирование 194
Многофазное тактирование 197
§ 3.8. Блоки PLL, DLL и DCM 197
Блоки PLL 198
Блоки DLL 200
Блоки DCM 201
§ 3.9. Тактирование сигналами, выработанными в приемниках информации 202
Выработка тактовых сигналов без передачи эталонов 202
Выработка тактовых сигналов с передачей эталона 204
О самосинхронизирующихся схемах 204
§ 3.10. Ввод внешних сигналов в синхронные устройства. Синхронизаторы 205
Ввод асинхронных сигналов 205
Синхронные, асинхронные и "полусинхронные" сигналы 206
Синхронизаторы мезохронных сигналов 207
Синхронизаторы с элементами задержек 207
Синхронизаторы с двумя регистрами 209
Синхронизатор с круговым буфером 210
Синхронизаторы плезиохронных сигналов 212
Контрольные вопросы и упражнения 212

Глава 4. Функциональные узлы последовательностного типа (автоматы с памятью) 217

§ 4.1. Введение в проблематику проектирования автоматов с памятью 217
О проектировании автоматов 219
Примеры проектирования 222
Вариант 1 223
Автомат, построенный на триггерах D и элементах И-НЕ 223

Вариант 2....224
Автомат, построенный на JK-триггерах и элементах И-НЕ....224
Вариант 3....225
Автомат, реализованный на D-триггерах и мультиплексорах....225
Вариант 4....227
Автомат с состояниями, кодируемыми в коде "1 из N"....227
§ 4.2. Регистры и регистровые файлы....230
Регистровые файлы....232
Сдвигающие регистры....233
Универсальные регистры....234
§ 4.3. Основные сведения о счетчиках. Двоичные счетчики....237
Классификация и режимы работы счетчиков....237
Двоичные счетчики....238
Асинхронные счетчики....238
Синхронные счетчики....240
Счетчики с групповой структурой....242
§ 4.4. Двоично-кодированные счетчики с произвольным модулем....244
Счетчики с модифицированными межразрядными связями....245
Счетчики с управляемым сбросом....247
§ 4.5. Счетчики с недвоичным кодированием....248
Счетчики в коде Грея....249
Счетчики в коде "1 из N"....251
Счетчики в коде "1 из N" на кольцевых регистрах....252
Счетчики в коде "1 из N" на основе счетчиков Джонсона....255
§ 4.6. Полиномиальные счетчики. Делители полиномов....258
Схемы генераторов псевдослучайных последовательностей....260
Кодеры и декодеры циклических кодов....262
Контрольные вопросы и упражнения....263

Глава 5. Запоминающие устройства....267

§ 5.1. Основные сведения. Параметры. Классификация....267
Важнейшие параметры ЗУ....268
Классификация ЗУ....270
ЗУ типа ROM....272
ЗУ типа RAM....273
Классификация статических ЗУ....273
Классификация динамических ОЗУ....274
Классификация перспективных ЗУ....275
Модули памяти....276
§ 5.2. Основные структуры запоминающих устройств....277
Структура 2D....277
Структура 3D....277
Структура 2DM....280

Блочные структуры 281
Видеопамять 283
Буферы FIFO, LIFO, круговой 284
Кэш-память 287
Модели основной памяти и кэша 288
Полностью ассоциативная кэш-память 290
Кэш-память с прямым размещением 291
Кэш-память с частично-ассоциативным отображением 291
§ 5.3. Структурные методы повышения быстродействия запоминающих устройств 294
Быстрый страничный доступ 294
Пакетная передача данных и команд 295
Технологии DDR и QDR 295
Многобанковые структуры 296
Конвейеризация трактов передачи данных 296
§ 5.4. Запоминающие устройства ROM, PROM, EPROM, EEPROM 297
ROM 297
Масочные ROM 298
Лазерные ROM 299
PROM и EPROM-OTP 299
EPROM и EEPROM 301
МНОП-транзисторы 301
Транзисторы с плавающим затвором 303
Транзисторы с двумя затворами 303
EPROM 304
EPROM-OTP 304
EEPROM 304
Внешняя организация рабочих режимов для микросхем постоянной памяти 305
Пример схемы ЗУ типа EPROM 306
§ 5.5. Флэш-память 307
Основные разновидности 307
Накопители с ячейками ИЛИ-НЕ и И-НЕ 308
Накопители на ячейках ИЛИ-НЕ 308
Накопители на ячейках И-НЕ 309
Средства улучшения характеристик 310
Команды управления 311
Память с несимметричными блоками 312
Память с симметричными блоками (файловая) 314
Память с многоуровневым хранением заряда 316
Память с зеркальным битом 317
Флэш-память с MLC-ячейками И-НЕ 318

§ 5.6. Последовательные репрограммируемые ЗУ 320
§ 5.7. Импульсное питание ROM 321
§ 5.8. Использование программируемых ЗУ для решения задач обработки информации 322
Реализация логических функций 322
Реализация конечных автоматов 323
Воспроизведение числовых функций 323
§ 5.9. Статические оперативные ЗУ 325
Структура асинхронного (стандартного) ЗУ 325
Запоминающие элементы 327
Запоминающий элемент в схемотехнике КМОП 327
Запоминающий элемент в схемотехнике n-МОП 327
Требования к усилителям считывания 328
Внешняя организация и временные диаграммы 329
Пример асинхронного ЗУ 331
Синхронные ЗУ 331
Структура синхронных ЗУ 333
§ 5.10. Искусственная энергонезависимость статических ОЗУ 334
Варианты с резервным источником питания 334
Память NV-SRAM 336
§ 5.11. Статические ЗУ типа БиКМОП 337
§ 5.12. Динамические запоминающие устройства — базовая структура 338
Запоминающие элементы 338
Усилители-регенераторы 341
Мультиплексирование шины адреса 342
Внешняя организация и временные диаграммы 342
Схема динамического ЗУ 343
§ 5.13. Динамические запоминающие устройства повышенного быстродействия 346
FPM, EDORAM, BEDORAM 346
FPM 346
EDORAM 347
BEDORAM 348
SDRAM и DDR SDRAM 348
RDRAM 352
Состав микросхем памяти RDRAM 353
Структура канала 353
Связь канала с микросхемами и их совместная работа 354
CDRAM 356
Ускорение произвольного доступа 357
ЗУ с блочной структурой 357
RLDRAM 358
FCRAM 359

§ 5.14. Регенерация данных в динамических ЗУ 359
Рабочий режим 360
Переход к режиму регенерации 361
Режим регенерации 361
Квазистатические ЗУ 361
§ 5.15. Перспективные запоминающие устройства 362
FRAM (ферроэлектрические ЗУ) 362
PFRAM (полимерно-ферроэлектрические ЗУ) 364
MRAM (магниторезистивные ЗУ) 365
ЗУ типа OUM (с фазовыми переходами вещества) 366
§ 5.16. Заключительные замечания 366
Контрольные вопросы и упражнения 368

Глава 6. Простые микропроцессоры и микропроцессорные системы. Микроконтроллеры 371

§ 6.1. Общие сведения. Структура и функционирование микропроцессорной системы 371
Структура простой МПС 373
Мультиплексирование шины адресов/данных 376
Принстонская и Гарвардская архитектуры процессоров 376
§ 6.2. Структура микропроцессора 378
Операционный блок 379
Блок регистров 380
Дешифрация команд 381
Блок синхронизации и управления 382
Исключения и прерывания 383
Блок управления прерываниями 384
§ 6.3. Функционирование микропроцессора 386
Синхронизация и последовательность действий МП 386
Адресные пространства, способы адресации, форматы команд 391
О системе команд 393
Пример выполнения команды и фрагмента программы 396
§ 6.4. О развитии микропроцессорной техники 399
CISC-процессоры 400
RISC-процессоры 400
VLIW-процессоры 400
Направления развития МП 400
§ 6.5. Управление памятью и внешними устройствами 402
Абсолютная и неабсолютная адресации 402
Интерфейсы с общей и раздельной шиной 402
Построение модуля памяти 403
Схемы подключения памяти к шинам МПС 404

Пример 1. Абсолютная адресация 404
Пример 2. Неабсолютная адресация 405
Пример 3. Декодирование адресов при совмещенном вводе/выводе 407
Выработка сигналов управления 408
Анализ нагрузочных условий 412
Согласование временных диаграмм МП и ЗУ 412
Разновидности операций ввода/вывода 415
Обмен по инициативе программы 415
Обмен по прерываниям 416
Прямой доступ к памяти 416
Безусловный программный ввод/вывод 416
Условный программный ввод/вывод 418
§ 6.6. Микроконтроллеры. Основные сведения 421
§ 6.7. Структура микроконтроллера 423
§ 6.8. Организация памяти и функционирование МК 427
Распределение памяти в МК AVR 427
Способы адресации 429
Выполнение команд 429
Режимы потребления мощности 430
Система прерываний 431
Программирование МК 431
Контрольные вопросы и упражнения 432

Глава 7. Интерфейсные схемы, адаптеры, контроллеры 435

§ 7.1. Общие сведения 435
Интерфейсы микропроцессорных систем 435
§ 7.2. Шинные формирователи и буферные регистры 438
Шинные формирователи 438
Буферные регистры 440
§ 7.3. Параллельные порты 441
§ 7.4. Параллельные адаптеры 443
Структура адаптера 444
Режимы работы портов 444
Режим 0 445
Режим 1 445
Режим 2 446
Работа адаптера в режиме 1 446
Работа адаптера в режиме 2 448
§ 7.5. Передачи последовательных данных 449
Тракты передачи последовательных данных 449
Характер передаваемой информации 450
Асинхронные и синхронные передачи 451
Структура кадра при последовательной асинхронной передаче 451

Работа приемника при асинхронных передачах 452
Фиксируемые ошибки передачи 453
Синхронные передачи 453
§ 7.6. Связные адаптеры 454
Передатчик ПСА 456
Приемник ПСА 458
§ 7.7. Интерфейсы SPI и I^2C 462
Интерфейс SPI 462
Интерфейс I^2C 466
§ 7.8. Схемы обслуживания прерываний 467
Программный опрос 468
Аппаратный опрос источников прерываний 468
Контроллеры прерываний 469
Структура ПКП 471
§ 7.9. Контроллеры прямого доступа к памяти 475
Структура и функции КПД 476
Выводы и сигналы контроллера 480
Передачи "память-память" 481
Наращивание числа каналов ПДП 481
§ 7.10. Таймеры 482
Простые таймеры 482
Таймер 0 482
Таймер 1 484
Формирование ШИМ-сигналов 484
Сторожевой таймер 485
Программируемый интервальный таймер 486
Структура таймера 487
Режим 0 489
Режим 1 489
Режим 2 491
Режим 3 491
Режим 4 492
Режим 5 493
§ 7.11. Схемотехника интерфейса JTAG 493
Интерфейс JTAG и граничное сканирование 493
Ячейка BSC 495
Интерфейс JTAG 495
Транспортный механизм 496
Устройство управления граничным сканированием 496
Механизм граничного сканирования 497
Команды граничного сканирования 498
Расширения интерфейса JTAG 499
Контрольные вопросы и упражнения 501

Глава 8. SPLD и CPLD — простые и сложные программируемые логические устройства 505

§ 8.1. Микросхемы с программируемой структурой. Вводные замечания 505
§ 8.2. Программируемые логические матрицы и программируемая матричная логика (ПЛМ и ПМЛ) 508
Структура ПЛМ 508
Упрощенное изображение схем ПЛМ 510
Воспроизведение скобочных форм логических функций 511
Схемотехника ПЛМ 511
Подготовка задачи к решению на ПЛМ 514
Структура ПМЛ 515
Обогащение функциональных возможностей ПЛМ и ПМЛ 516
Программирование выходных буферов 516
Применение двунаправленных выводов 518
Введение элементов памяти 519
Использование разделяемых конъюнкторов в схемах ПМЛ 520
Примеры отечественных ПМЛ 521
ПМЛ без элементов памяти 522
ПМЛ с элементами памяти 522
Пример подготовки задачи к решению с помощью ПМЛ 523
ПМЛ типа PAL 22V10 525
§ 8.3. CPLD — сложные программируемые логические устройства 528
Структура CPLD 528
Функциональные блоки CPLD 529
Логические расширители 530
Макроячейки 531
Системы коммутации CPLD 533
Блоки ввода/вывода CPLD 535
Пример типичной CPLD 536
Контрольные вопросы и упражнения 539

Глава 9. FPGA — программируемые пользователями вентильные матрицы 541

§ 9.1. Общие сведения 541
Свойства и возможности FPGA 541
Программируемые элементы 543
§ 9.2. Архитектура и основные блоки FPGA 546
Базовая архитектура 546
Усложненные архитектуры 547
Логические блоки 548
Системы межсоединений 554
Блоки ввода/вывода 560

§ 9.3. Ресурсы памяти 563
Распределенная память 563
Встроенная память 566
Применение встроенных блоков памяти 570
§ 9.4. Умножители и блоки ЦОС 572
Умножители 572
Основные операции обработки сигналов 574
Структура ЦОС-блока 578
§ 9.5. Программируемые аналоговые и аналого-цифровые схемы 582
Два варианта интегральных аналоговых схем 582
Практические разработки 585
§ 9.6. Способы оценки параметров ПЛИС 590
Оценки логической сложности ПЛИС 590
Оценки быстродействия 592
Факторы, влияющие на стоимость 593
§ 9.7. Конфигурирование программируемых микросхем 594
Режимы конфигурирования 594
Этапы конфигурирования 595
§ 9.8. Засекреченность проектов 596
Клонирование и реконструкция проектов 597
§ 9.9. Примеры типичных FPGA средней сложности 599
FPGA с триггерной памятью конфигурации 599
FPGA с программируемыми перемычками 601
Контрольные вопросы 604

Глава 10. Программируемые системы на кристалле 607

§ 10.1. Основные сведения 607
IP-ядра. Блочное и платформенное проектирование 608
Типы программируемых "систем на кристалле" 610
Soft-ядра процессоров 614
Hard-ядра процессоров 618
Шинные системы 621
§ 10.2. FPGA класса "система на кристалле" 622
Серия Stratix 622
Серия Virtex 626
Микросхемы с флэш-памятью конфигурации 630
§ 10.3. Системы на кристалле микроконтроллерного типа 632
Серия PSoC 633
Контрольные вопросы 639

Глава 11. Микросхемы, программируемые с участием изготовителя 641

§ 11.1. Базовые матричные кристаллы (вентильные матрицы, программируемые изготовителем) 641
Основные сведения 641
Классификация БМК 643
Компонентный состав базовых ячеек 647
Основные понятия и определения 649
Параметры БМК 650
Этапы проектирования МАБИС 651
§ 11.2. Структурированные вентильные матрицы 653
Конвертация проектов 653
Практические разработки 654
Контрольные вопросы 657

Глава 12. Методика и средства автоматизированного проектирования цифровых устройств 659

§ 12.1. Общее описание процесса проектирования 659
§ 12.2. О выборе альтернативных средств реализации проекта 665
Традиционная реализация проектов 666
Реализация проектов на кристаллах с программируемыми структурами 667
Место программируемой логики в процессе создания современной аппаратуры 671
§ 12.3. Инструментарий проектировщика 673
Средства системного этапа проектирования 674
Разработка специфических фрагментов проекта 675
Средства разработки процессорной части проекта 676
Средства разработки цифровой части проекта 678
Средства разработки аналоговых и аналого-цифровых фрагментов 681
Работа и средства этапа комплексной отладки проекта 681
Специфика конструирования и отладки проектов на ПЛИС и СнПК 682
§ 12.4. Системный этап проектирования цифровых устройств на базе ПЛИС 683
Выбор САПР 684
Представление проекта на блочно-функциональном уровне 684
Средства описания проекта 686
Графическое представление проекта 686
Текстовое описание 687
Языки низкого уровня 687
Языки высокого уровня 688
Средства описания автоматов 688

§ 12.5. Маршрут проектирования ПЛИС и возможности типовых САПР 690
Этапы проектных процедур с использованием САПР 690
§ 12.6. Основные сведения о языке VHDL 694
Назначение и возможности языка 694
Основные понятия и синтаксические конструкции языка 695
Описание проекта на языке VHDL 698
Примеры описаний элементов на языке VHDL 698
Структурное и поведенческое описание проекта 701
Язык VHDL для моделирования и синтеза 701
О возможностях и средствах описания типовых узлов цифровой техники 702
Введение в язык VHDL-AMS 727
§ 12.7. Пример автоматизированного проектирования цифрового устройства с использованием языков описания аппаратуры 737
Первый этап. Рассмотрение ТЗ на разрабатываемое устройство 737
Второй этап. Разработка общей структуры операционного блока 738
Третий этап. Описание работы управляющего автомата 740
Пояснения к синтаксису VHDL программы устройства управления 742
Четвертый этап. Компиляция проекта и основные параметры устройства 746
Пятый этап. Тестирование проекта 746
Шестой этап. Автоматическое определение временных характеристик устройства 748
Седьмой этап. Практическое использование результатов проектирования 748
Контрольные вопросы и упражнения 748

Приложение. Основные логические операции и законы 755

Контрольные вопросы и упражнения 758

Словарь иностранных сокращений и терминов 761

Принятые сокращения 769

Литература и источники информации в Интернете 775

Краткая библиография 775
Интернет-ресурсы 779

Предметный указатель 781

Предисловие

Интересы России требуют преодоления сложившегося в последнее время однобокого развития экономики с акцентом на сырьевые отрасли производства. Неотъемлемой чертой развитых государств является способность к разработке и производству высокотехнологичных изделий, среди которых ведущее место занимает электронная аппаратура. Успехи электроники в значительной мере определяют прогресс промышленности и науки. Современная промышленная продукция, будь то автомобиль, самолет, корабль, станок для металлообработки, изделие бытовой техники и т. д. насыщена электроникой. Трудно переоценить роль компьютеров во всех сферах жизни.

По вопросам актуальности разработок в России электронной аппаратуры имеется ряд весьма определенных мнений (академик Е. П. Велихов, лауреат Нобелевской премии академик Ж. И. Алферов и др.). Приведем некоторые из них.

Академик Ж. И. Алферов пишет (газета "Известия", 15 марта 2001 г.):

> "Важно заниматься научными и технологическими исследованиями в области электроники, потому что именно она определяет технологический и даже социальный прогресс... Без собственных современных электронных технологий любые наши другие (те же космические) быстро перейдут во второсортные... Сейчас у нас два пути — либо становиться страной третьего мира, живущей за счет ресурсов, либо развивать наукоемкие отрасли".

Начальник Управления радиоэлектронной промышленности и систем управления Роспрома Ю. И. Борисов в статье "Отечественная электронная промышленность и компонентная база" (журнал "Электроника: Наука, Технология, Бизнес", № 2/2006) пишет:

> "Если мы хотим создавать передовую военную технику и обеспечить технологическую независимость и информационную безопасность всех наших электронных систем, в том числе и гражданских, ключевые изделия микроэлектроники необходимо проектировать и производить в России".

В том же номере журнала в статье главного редактора П. П. Мальцева сказано:

> "Способность российских предприятий разрабатывать, производить и обеспечивать современной электронной компонентной базой и радиоэлектронной аппаратурой отечественную продукцию радиоэлектронного комплекса является одним из критических условий, обеспечивающих национальную безопасность России в информационную эпоху и решающим фактором социально-экономического развития страны".

Отставание от мировой техники в области микроэлектроники возникло еще в СССР. Российский кризис 90-х годов сделал это отставание чрезвычайно сильным. Сегодняшний день, как можно надеяться, явится началом преодоления кризиса. Непременное условие выхода из кризиса — подготовка квалифицированных кадров, способных разрабатывать проекты с помощью современных методов и средств. В уже цитированном журнале ("Электроника: Наука, Технология, Бизнес", № 4/2006) председатель Федерального фонда развития электронной техники А. И. Сухопаров пишет:

> "Пока есть специалисты, умеющие проектировать, мы никогда не отстанем "навсегда"".

В связи с ростом интереса к проблемам разработки электронной аппаратуры увеличивается и потребность в соответствующей учебной литературе. Первое и второе издания предлагаемого учебного пособия (2000 и 2004 гг.) получили широкое признание преподавателей и студентов и используются во многих вузах при изучении соответствующих дисциплин. В третьем издании обновлены методика изложения и содержание многих разделов, особенно тех, которые посвящены новым направлениям развития компонентной базы цифровой техники (программируемая логика, особо быстродействующие устройства и др.). Как и ранее, рассмотрен широкий круг вопросов, связанных с изучением и применением современной компонентной базы цифровой техники. Кроме того, третье издание снабжено контрольными вопросами и упражнениями, которые отсутствовали в предыдущих изданиях.

Пособие предназначено для студентов вузов, обучающихся по специальности 220100 (Вычислительные машины, комплексы, системы и сети) и направлению 552800 (Информатика и вычислительная техника). Поскольку средства цифровой техники разного назначения (микропроцессорные системы, связь, сети и т. д.) во многом строятся на идентичной компонентной базе, пособие может быть полезно и для студентов других специальностей и направлений, связанных с использованием и разработкой цифровых средств обработки информации. Оно может быть полезно также для работников промышленности и научных учреждений.

Содержание пособия основано на материале курса лекций, читаемого на кафедре "Вычислительная техника" Санкт-Петербургского государственного электротехнического университета "ЛЭТИ".

В работе над книгой участвовал к. т. н., доц. Р. И. Грушвицкий, которым написана глава 12 (§ 12.2 совместно с автором) и § 7.10 (совместно с автором).

Авторы благодарят рецензентов за ряд ценных замечаний, способствовавших улучшению книги.

Введение

Компонентную базу цифровых устройств и систем составляют *микросхемы* (МС). Со времени их изобретения (США, 1959 г.) микросхемы постоянно совершенствуются и усложняются, подчиняясь так называемому *закону Мура*. Гордон Мур (один из основателей известной фирмы Intel) заметил, что уровень интеграции МС удваивается каждые 1,5 — 2 года (уровень интеграции является характеристикой сложности МС и оценивается числом базовых логических элементов или транзисторов, размещенных на кристалле). Закон Мура соблюдается уже более тридцати лет. Что же движет этот процесс? *С ростом уровня интеграции МС существенно улучшаются характеристики* устройств и систем, реализованных на их основе (снижается стоимость/вентиль, повышаются быстродействие и надежность, уменьшаются габариты и потребляемая мощность, упрощаются конструкции и т. д.).

Различия в уровне интеграции делят МС на несколько категорий: *МИС*, *СИС*, *БИС*, *СБИС* (соответственно малые, средние, большие и сверхбольшие интегральные схемы). МИС выполняют простейшие логические преобразования, в виде СИС выпускаются малоразрядные функциональные узлы (регистры, счетчики, дешифраторы, сумматоры и т. п.), в БИС и СБИС размещаются схемы с тысячами, миллионами и более логических элементов (сейчас уже можно разместить на кристалле несколько миллиардов транзисторов). Практическое использование в той или иной мере находят все категории, однако с течением времени все большую долю используемых микросхем составляют схемы высокого уровня интеграции.

Современные МС высокого уровня интеграции чрезвычайно сложны в проектировании и производстве. Разработка СБИС может занимать годы, требует изготовления около 40 фотошаблонов стоимостью приблизительно 50 или 250 тыс. долларов каждый (в технологиях с минимальными размерами 90 нм и 45 нм соответственно) и выполнения ряда других этапов с применением высокотехнологичного оборудования, причем общая стоимость всех работ превышает величины порядка десятков миллионов долларов. Поэтому *заказные* БИС/СБИС могут применяться лишь для продукции массового применения, когда затраты на проектирование раскладываются на большое число выпускаемых кристаллов и могут окупаться.

Снижения сроков и стоимости проектирования БИС/СБИС можно достичь, применяя так называемые *полузаказные* МС. В этом случае промышленность выпускает некоторые полуфабрикаты, пригодные для реализации на их основе различных устройств, а для получения законченного изделия *(специализации кристалла)* полуфабрикат дорабатывается по требованиям конкретного заказчика. Полуфабрикат выпускается массовыми тиражами, а потребитель платит только за его доработку

до конечного изделия. Длительность и стоимость проектирования полузаказных МС в 3 — 4 раза меньше, чем заказных, но все же остаются настолько большими, что для малотиражной аппаратуры могут быть недоступными.

Возможность применения БИС/СБИС для реализации проектов широким кругом разработчиков была найдена на пути *переноса процесса специализации микросхем в область программирования.* Появились микропроцессоры и, позднее, БИС/СБИС с программируемой структурой.

Микропроцессор способен выполнять определенный набор команд. Меняя последовательность и состав команд (программу), можно решать различные задачи на одном и том же микропроцессоре. Иначе говоря, в этом случае структура аппаратных средств не связана с характером решаемой задачи. Будучи стандартным для производителя, микропроцессор настраивается программой на решение разных задач. Это обеспечивает микропроцессорам массовое производство с соответствующим снижением стоимости и делает их доступными для широкого круга потребителей.

В виде БИС/СБИС *с программируемой структурой* потребителю предлагается кристалл, содержащий множество элементов, межсоединения для которых назначает сам системотехник. Промышленность получает возможность производить кристаллы массовым тиражом, не адресуясь к отдельным потребителям. Реализация конкретного варианта межсоединений элементов может возлагаться на изготовителя (для полузаказных кристаллов) или потребителя (для ПЛИС и *ПАИС — программируемых логических и аналоговых интегральных схем*). Работая с ПЛИС и ПАИС, системотехник сам программирует структуру МС соответственно своему проекту и не обращается для этого на предприятия электронной промышленности для выполнения сложных и дорогостоящих технологических процессов. В то же время дополнительные схемотехнические средства для программирования структуры кристалла усложняют и удорожают его. Тем не менее, *ПЛИС и ПАИС открывают доступ к применению БИС/СБИС тем многочисленным разработчикам, которые создают аппаратуру ограниченной тиражности, в первую очередь ту, от которой требуется высокая производительность.*

Микропроцессоры реализуют последовательную обработку информации, выполняя большое число отдельных действий, соответствующих командам, что может не обеспечить требуемого быстродействия. При аппаратных реализациях алгоритмов обработка информации может происходить без разбиения этого процесса на последовательно выполняемые элементарные действия. Задача может решаться "целиком", ее характер определяет структуру устройства. Преобразование данных происходит одновременно во многих частях устройства. Сложность устройства зависит от сложности решаемой задачи, чего нет в микропроцессорных системах, где сложность задачи влияет лишь на программу, а не на аппаратные средства ее выполнения.

Таким образом, БИС/СБИС с аппаратной реализацией алгоритмов могут быстрее решать задачи, сложность которых ограничена уровнем интеграции микросхем, а микропроцессорные средства — задачи неограниченной сложности, но с меньшим быстродействием. Оба направления открывают ценные перспективы дальнейшего

улучшения технико-экономических показателей создаваемой на них аппаратуры. В БИС/СБИС с программируемой структурой могут быть выполнены как системы с микропроцессором, так и аппаратные варианты решения задачи. Более того, современные кристаллы высшего уровня интеграции содержат *одновременно и микропроцессоры и большие массивы программируемой логики,* обладая в силу этого большими функциональными возможностями. Подобная структура свойственна микросхемам класса *"система на кристалле"*, важная роль которых в проектировании современной аппаратуры неоспорима.

С ростом уровня интеграции ИС в проектировании на их основе все больше усиливается аспект, который можно назвать интерфейсным проектированием. Задачей разработки становится составление блоков из все более сложных стандартных субблоков путем правильного их соединения. Успешное проектирование требует хорошего знания номенклатуры и параметров МС и привлечения систем автоматизированного проектирования (САПР). Типовые базовые функциональные компоненты, из которых составляются сложные устройства, в ходе развития микроэлектроники остаются почти неизменными, но формы их представления становятся более разнообразными. Если ранее такие компоненты были отдельными кристаллами МИС/СИС, то теперь они чаще представляются в виде фрагментов более сложных схем, не имеющих индивидуального конструктивного оформления и задаваемых файлами, содержащими информацию о том, как реализовать тот или иной компонент в некоторой области кристалла. Набор таких сведений образует *библиотеку функциональных компонентов системы проектирования.*

Подавляющее большинство современных МС изготовляются по схемотехнологии КМОП. Некоторое число микросхем или схемных фрагментов кристалла реализуются по биполярной технологии (ТТЛШ, ЭСЛ). Элементы КМОП при субмикронных топологических нормах обладают рядом уникальных параметров и доминируют в схемах внутренних областей БИС/СБИС. За биполярной технологией осталась пока область некоторых периферийных схем, где требуется быстрая передача сигналов по внешним цепям, испытывающим большую емкостную нагрузку.

При работе над учебным пособием автор стремился учесть современную ситуацию в высшей школе — наличие двухуровневой подготовки "бакалавр — магистр" и сквозной подготовки специалистов (инженеров). Структурирование текста пособия позволяет варьировать глубину изучения материалов согласно необходимому уровню. Изучение тем начинается с рассмотрения общей постановки вопроса, выяснения характера и содержания проблем, связанных с темой, и способов их решения, а затем продолжается на более подробном уровне с привлечением дополнительных сведений и примеров.

Структурно рукопись построена по следующему плану:

- Общие основы цифровой схемотехники. Логические элементы и сигналы, а также их параметры. Методы и средства решения задач взаимодействия компонентов и сигналов в составе системы, позволяющие понимать и предлагать грамотные схемотехнические решения.

- Функциональные узлы. Типовые функциональные узлы комбинационного типа (не содержащие элементов памяти). Элементы памяти (триггеры). Проблемы тактирования и синхронизации в цифровых устройствах, играющие ключевую роль при разработке устройств высокого быстродействия. Типовые функциональные узлы последовательностного типа (содержащие элементы памяти).
- Функциональные устройства и простые микропроцессорные системы. Запоминающие устройства (широкий набор возможных реализаций). Интерфейсные схемы, адаптеры и контроллеры. Простые процессоры, микропроцессорные системы, микроконтроллеры, т. е. типовые композиции устройств, обеспечивающие решение задач программными методами и иллюстрирующие взаимодействие модулей системы и требования к их функционированию. Этот раздел посвящен изучению цифровой техники на более высоком иерархическом уровне, чем уровень узлов.
- Программируемые средства создания цифровой техники. Простые и сложные программируемые логические устройства (SPLD, CPLD). Программируемые пользователями вентильные матрицы (FPGA). Системы на кристалле. Микросхемы, программируемые с участием изготовителя (базовые матричные кристаллы, структурированные вентильные матрицы, гибридные схемы ASIC + FPGA).
- Методы и средства автоматизированного проектирования цифровых устройств. Инструментарий проектировщика (средства разработки для выполнения различных этапов проекта). Основные сведения о языках описания аппаратуры. Язык VHDL и VHDL-AMS. Пример автоматизированного проектирования цифрового устройства с использованием языка VHDL.
- Заключение. Итоговый обзор возможных вариантов реализации цифровых устройств на основе различных средств компонентной базы и методов проектирования.

Глава 1

Схемотехнические проблемы построения цифровых устройств

При разработке цифровых устройств (ЦУ) приходится решать не только задачи логического синтеза структур, но и схемотехнические задачи общего характера, не зависящие от реализуемого алгоритма и логики функционирования схем. К таким задачам относятся, например, подавление помех, передача данных по линиям связей, обеспечение требуемых режимов входов и выходов элементов, генерация и распределение тактовых сигналов и т. п. Подобным задачам и посвящена эта глава, изучение которой призвано повысить общую схемотехническую грамотность читателя и подготовить его к пониманию схемных решений, характерных для цифровых устройств.

§ 1.1. Модели и параметры логических элементов

Простейшая модель логического элемента

Даже самые сложные преобразования цифровой информации, в конечном счете, сводятся к простейшим операциям над логическими переменными 0 и 1. Такие операции реализуются логическими элементами в соответствии с формулами алгебры логики. Предельно идеализированные логические элементы могут быть представлены моделями вида (рис. 1.1, *а*), т. е. условными графическими обозначениями — прямоугольниками, в которых ставится символ выполняемой операции (на рисунке операция обозначена символом "звездочка"), а на линиях входных и выходных переменных изображаются кружки (индикаторы инверсии), если данная переменная входит в реализуемую формулу в инверсном виде.

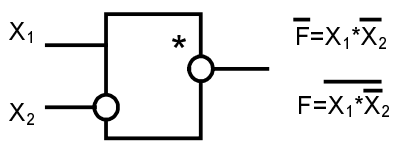


Рис. 1.1. Обозначение предельно идеализированного логического элемента

Логические функции могут быть выражены с помощью разных логических операций. Булевский базис (набор операций И, ИЛИ, НЕ) удобен своей наглядностью и легче всего воспринимается человеком, поэтому широко используется для исходного описания логических зависимостей. Операции И-НЕ (Шеффера) и ИЛИ-НЕ (Пирса) — основные "рабочие" базисы, поскольку их схемные реализации, как правило, оказываются наиболее выигрышными. К широко применяемым операциям относится и сложение по модулю 2, т. к. некоторые логические преобразования эффективнее всего выполняются именно в этом базисе.

> ***Примечание***
>
> Логические и арифметические основы цифровой техники предполагаются известными читателю. Материалы, кратко напоминающие о функциях и логических законах, чаще всего используемых при проектировании ЦУ, даны в *приложении 1*.

Сигналы, отображающие логические переменные

В цифровых устройствах логические переменные 0 и 1 обычно отображаются двумя различными уровнями напряжения: U_0 и U_1 соответственно (наряду с обозначениями U_1 и U_0 могут быть использованы и обозначения высокого и низкого уровней напряжения соответственно как *H (High)* и *L (Low)*).

Переход от логических переменных к электрическим сигналам ставит вопрос о *логических соглашениях*. Необходимо условиться, какой из двух уровней напряжения принять за U_0 и какой за U_1. Существуют соглашения положительной и отрицательной логики. В положительной логике $U_1 > U_0$, а в отрицательной $U_1 < U_0$. Один и тот же элемент, в зависимости от принятого логического соглашения, выполняет различные логические операции. В дальнейшем, если не оговорено иное, *будем пользоваться соглашением положительной логики*.

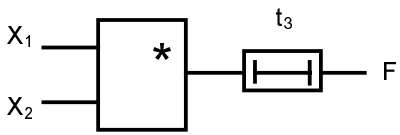
Пользуясь алфавитом всего из двух символов {0, 1}, нельзя точно описать форму сигнала, так как эти символы позволяют отображать только идеально прямоугольные импульсы. Реальные сигналы имеют форму, которая с приемлемой для большинства приложений точностью аппроксимируется трапециями. Для трапецеидальных сигналов на интервалах *переходных процессов* при переключениях в направлениях 0 → 1 и 1 → 0 логические значения сигналов не определены, и эти интервалы отмечаются *символами неопределенности* X, что дает трехсимвольный алфавит {0, 1, X}, на котором основано *троичное моделирование*, выявляющее критические временные состязания сигналов при допущении об их произвольных задержках. Для учета реального диапазона изменений задержек разработаны *модифицированные методы* троичного моделирования.

Добавление символа Z, отображающего присущий некоторым элементам *режим "отключено",* приводит к четырехсимвольному алфавиту {0, 1, X, Z}. Вводя понятие *силы сигнала* и др. его признаки, расширяют алфавит и далее (в частности, в ряде систем моделирования применяют девятисимвольный алфавит). Расширение алфавита при описании сигналов позволяет приближать модельное описание процессов в цифровых схемах к реальным, но существенно увеличивает время

моделирования и объем проводимых в его ходе вычислений. Поэтому в системах моделирования на разных этапах работы могут применяться модели сигналов разной сложности — простые для быстрого неточного первоначального моделирования и более сложные для более адекватного описания процессов в схемах.

Учет задержек сигналов в логических схемах

Быстродействие и работоспособность ЦУ зависят от задержек сигналов в элементах и связях между ними. По мере уменьшения топологических норм роль задержек в связях все более возрастает и в современных БИС/СБИС они составляют 70…80% общей задержки. Длительность переключения элемента и распространения сигнала по линии связи отображается введением в выходную цепь элемента задержки на время $t_з$ (рис. 1.2). В простейшем случае задержка принимается постоянной, но такая модель является очень грубой, поскольку реальная задержка существенно зависит от целого ряда факторов: емкостных нагрузок на выходах элементов, крутизны фронтов их входных напряжений, напряжения питания, температуры, технологического разброса задержек, их зависимости от направления переключения элемента и т. д.



1. t_3 = const,
2. t_3 = f (C_L, t_{BX}, U_{CC} и т. д.)

Рис. 1.2. Модели логического элемента с задержкой

Перечисленные факторы могут сильно изменять величину задержки. В частности, для элементов КМОП резко выражена зависимость задержки от емкостной нагрузки.

Уточненные отображения динамических свойств элементов учитывают также их *фильтрующие свойства*, благодаря которым короткие входные импульсы, обладающие малой энергией, не способны вызвать переключение элемента, даже если их амплитуда велика.

Точный расчет переходных процессов в реальных цепях весьма сложен, поэтому приходится прибегать к упрощениям, что вносит погрешности в вычисление задержек сигналов. В то же время правильная оценка задержек остро необходима, особенно при проектировании современных схем высокого быстродействия. Многие специалисты считают расчет и обеспечение надлежащих временных соотношений между сигналами *(Timing) важнейшей проблемой проектирования быстродействующих цифровых устройств* ("Timing is everything in SoC Design", SoC — System on Chip).

Задержки вычисляют разными способами: с помощью таблиц при учете влияющих на задержки факторов (емкостных нагрузок, крутизны фронтов входных сигналов и др.), путем описания задержек как функций от влияющих на них факторов или с помощью нелинейных моделей.

Применение таблиц дает хорошую точность, но требует больших затрат памяти. Как правило, применяют двухвходовые таблицы, в которых задержка элемента считывается из таблицы по вычисленным для данного элемента крутизне входного сигнала и емкостной нагрузке на его выходе.

Примером второго метода (используется в САПР фирмы Synopsis) может служить вычисление задержки по выражению:

$$t_з = \alpha\, \tau_{in} + \beta\, C_L + \gamma\, \tau_{in} C_L + \delta,$$

где C_L — емкостная нагрузка на выходе элемента, τ_{in} — длительность фронта входного сигнала, характеризующая его крутизну, а α, β, γ, δ — постоянные коэффициенты, зависящие от схемотехнологических параметров элемента. В сущности, такое формульное представление задержек аппроксимирует таблицу их значений и является более компактным, чем таблица. В некоторых системах проектирования применяют более простое выражение, в котором пренебрегают влиянием крутизны фронта входного сигнала на величину задержки и ограничиваются соотношением:

$$t_з = k_1\, C_L + k_2,$$

где k_1 и k_2 — постоянные коэффициенты.

Грубые модели с фиксированными значениями задержек полезны не столько для окончательных расчетов, сколько для обозначения проблем, возникающих в схемах из-за задержек сигналов.

Статические параметры логических элементов

Для правильного проектирования и эксплуатации цифровых устройств необходимо знать статические и динамические параметры логических элементов. В качестве важнейших статических параметров приводятся пять значений напряжений и пять значений токов.

Уровни напряжений и статическая помехоустойчивость логических элементов

В числе *напряжений* указывается напряжение питания U_{cc} (величина и поле допуска) и четыре значения напряжений задающих границы зон для переменных (0 и 1) на выходе и входе элемента. Для надежной работы элемента требуется, чтобы поступающие на его вход напряжения, соответствующие логической 1 и логическому 0, четко отличались друг от друга, т. е. чтобы напряжение U_1 было достаточно высоким, а напряжение U_0 достаточно низким. Эти требования задаются параметрами $U_{вх.1.min}$ и $U_{вх.0.max}$.

Входные напряжения данного элемента есть выходные напряжения предыдущего (источника сигналов). Уровни, гарантируемые на выходе элемента при соблюдении допустимых нагрузочных условий, задаются параметрами $U_{вых.1.min}$ и $U_{вых.0.max}$. Выходные уровни несколько "лучше", чем требуемые входные, что обеспечивает определенную помехоустойчивость элемента (рис. 1.3).

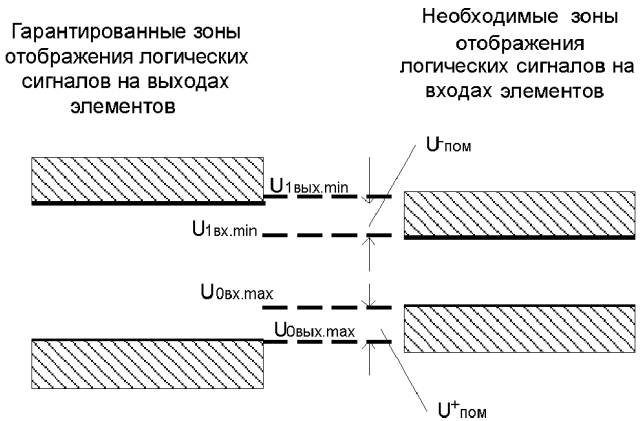


Рис. 1.3. Зоны отображения сигналов на входах и выходах логических элементов

Для уровня U_1 опасны отрицательные помехи, снижающие его, причем допустимая статическая помеха (т. е. помеха любой длительности)

$$U^{-}_{пом} = U_{1вых.min} - U_{1вх.min}.$$

Для уровня U_0 опасны положительные помехи, причем допустимая статическая помеха

$$U^{+}_{пом} = U_{0вх.max} - U_{0вых.max}.$$

Токовые параметры

Для *токов* в первую очередь указывается ток потребления, который нужен и для определения потребляемой элементом мощности, рассчитываемой как произведение напряжения питания элемента на потребляемый им ток.

Указываются и еще четыре значения — входные и выходные токи в обоих логических состояниях. При высоком уровне выходного напряжения из элемента-источника ток вытекает, цепи нагрузки ток поглощают. При низком уровне выходного напряжения элемента-источника ток нагрузки втекает в этот элемент, а из входных цепей элементов-приемников токи вытекают. Зная токи $I_{вых.1.max}$ и $I_{вых.0.max}$, характеризующие возможности элемента-источника сигнала, и токи $I_{вх.1.max}$ и $I_{вх.0.max}$, потребляемые элементами-приемниками, можно контролировать соблюдение нагрузочных ограничений, обязательное для всех элементов схемы ЦУ.

Для элементов ТТЛШ характерно большое различие между токами нулевых и единичных состояний. Ток $I_{вх.1}$ намного меньше тока $I_{вх.0}$ (приблизительно на порядок), а ток $I_{вых.1.max}$ меньше тока $I_{вых.0.max}$ приблизительно в 20 раз. Токи элементов КМОП такой несимметрии не имеют. Оба статических входных тока у них находятся в микроамперном диапазоне (или еще меньше), выходные токи могут составлять десятки миллиампер. Их роль состоит в перезаряде емкостей при переключениях элемента.

Быстродействие цифровых элементов

Быстродействие цифровых элементов определяется скоростями их перехода из одного состояния в другое. Быстродействие схем ограничивается задержками сигналов как в элементах, так и в цепях их межсоединений.

Переходные процессы для элемента с инвертирующим выходом (НЕ, И-НЕ, ИЛИ-НЕ) изображены на рис. 1.4. Их этапы отсчитываются по так называемым измерительным уровням. Моментом изменения логического значения сигнала считают момент достижения им *порогового уровня*. Обычно за пороговый уровень принимают середину логического перепада сигнала, т. е. $(U_0 + U_1)/2$. На диаграммах показаны задержки сигнала t^{10} и t^{01} при изменении выходного напряжения от U_1 до U_0 и обратно. Для упрощения расчетов пользуются *усредненным значением задержки* распространения сигнала

$$t_з = (t^{10} + t^{01})/2.$$

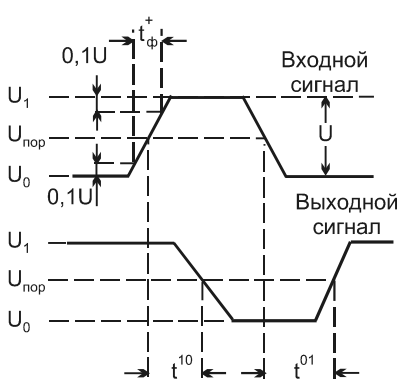


Рис. 1.4. Временные диаграммы процессов переключения инвертирующего логического элемента

Приемлемость такого усреднения усиливается тем, что в последовательных цепочках наиболее употребляемых инвертирующих элементов (И-НЕ, ИЛИ-НЕ) соседние элементы переключаются в противоположных направлениях, так что задержка пары элементов на самом деле будет равна $2t_з$.

Следует обратить внимание на то, что усреднение согласно приведенному соотношению не относится к технологическому разбросу задержек. Также следует за-

метить, что справочные данные о задержках соответствуют определенным условиям измерений, указанным в справочниках. Если условия работы элемента отличаются от условий измерения, то может потребоваться коррекция справочных данных.

На быстродействие ЦУ влияют как внутренние процессы в транзисторах, так и схемные емкости, на перезаряд которых требуется время. В справочных данных приводятся входные и выходные емкости элементов, учет этих емкостей и емкостей межсоединений позволяет подсчитать *нагрузочные емкости* в узлах схемы.

Для подключаемой к выходу элемента емкости приводятся две цифры: номинальная емкость C_L (L от Load) и предельно допустимая емкость C_{max}. Первая емкость соответствует условиям измерения задержек сигналов, так что именно для нее справедливы значения, приведенные в справочных данных. Если реальная нагрузочная емкость отличается от номинальной, то изменятся и значения задержек, что можно оценить с помощью соотношения $t_з = t_{з.н} + k\Delta C$, где $t_{з.н}$ — номинальное значение задержки; $\Delta C = C - C_L$; C — фактическое значение нагрузочной емкости; k — коэффициент, величина которого задается для серий элементов индивидуально.

Предельно допустимая емкость C_{max} указывает границу, которую нельзя нарушать, поскольку при этом работоспособность элемента не гарантируется.

Мощности потребления цифровых элементов

При разработке ЦУ требуется оценивать мощности их потребления, чтобы сформулировать требования к источникам питания, оценить температурный режим устройства и конструкцию теплоотвода. Для портативных устройств с автономным (батарейным или аккумуляторным) питанием особенно актуально проектирование схем с минимальной потребляемой мощностью. При оценке потребляемой мощности суммируются мощности, рассеиваемые всеми элементами схемы (в том числе и межсоединениями).

В справочных данных обычно указывается ток, потребляемый микросхемой. Произведение этого тока и напряжения питания определяет потребляемую микросхемой мощность.

Мощности, потребляемые элементами, делят на *статические и динамические*. Статическая мощность $P_{СТ}$ потребляется элементом, который не переключается. При переключении потребляется дополнительно динамическая мощность $P_Д$, которая практически пропорциональна частоте переключения элемента. Полная мощность является суммой статической и динамической мощностей ($P = P_{СТ} + P_Д$).

Элемент, который не переключается, может находиться в нулевом или единичном состоянии, соответственно чему может потреблять разную мощность (P_0 или P_1). Поскольку затруднительно оценить время нахождения элементов в том или ином состоянии, статическую мощность усредняют, полагая $P_{СТ} = (P_0 + P_1)/2$.

До появления схем с субмикронными топологическими нормами статическая мощность элементов КМОП была пренебрежимо малой, и полная мощность практически определялась ее динамической составляющей (например, для стандартного логического элемента серии КР1554 статическая мощность составляет 0,002% от мощности, потребляемой на частоте 50 МГц). Позднее положение изменилось. Схемы с субмикронными топологическими нормами имеют увеличенные токи утечек и потребляют статическую мощность, соизмеримую с динамической (в самых современных КМОП-технологиях находят способы уменьшения токов утечки). Характер зависимости полной мощности от частоты переключения элемента показан на рис. 1.5 для микросхемы стандартной КМОП-технологии (*а*) и субмикронной технологии (*б*). Для элементов типа ТТЛШ мощность потребления зависит от частоты по варианту рис. 1.5, *б*.

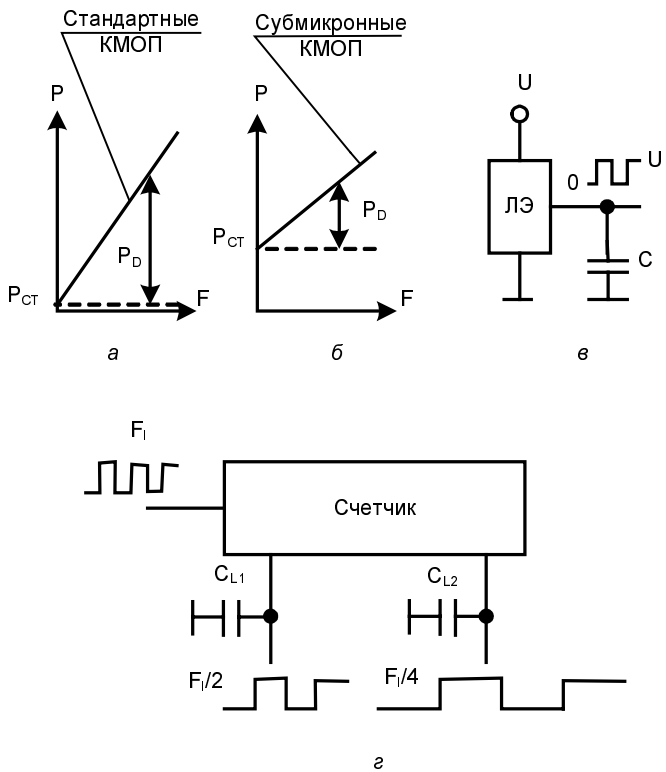


Рис. 1.5. Зависимость мощности, потребляемой элементом, от частоты его переключения (*а*, *б*), выходная цепь элемента с нагрузочной емкостью (*в*), пример схемы с несколькими выходными сигналами разной частоты (*г*)

При перезаряде емкости C затрачивается энергия $CU^2/2$, где U — перепад напряжения на емкости. Если цепь с емкостной нагрузкой переключается с частотой F (рис. 1.5, *в*), то при этом энергия, затрачиваемая за секунду, т. е. потребляемая мощность, составит величину CU^2F, поскольку в каждом периоде происходят два

перезаряда. На этом соотношении базируется формула оценки динамической мощности:

$$P_D = C_{PD}\, U^2 F_I + N C_L U^2 F_O = U^2 (C_{PD} F_I + N C_L F_O),$$

где P_D — динамическая мощность, F_I и F_O — частоты изменения сигналов на входе и выходе элемента, U — перепад сигнальных напряжений (для КМОП-элементов практически равный напряжению питания U_{CC}), C_L — емкость нагрузки и N — число переключаемых выходов. Особое место занимает параметр C_{PD} — так называемая *внутренняя емкость* элемента (схемы), интегрально учитывающая составляющие динамической мощности, обусловленные емкостными и сквозными токами во внутренних цепях элемента. Эта емкость указывается в паспортных данных микросхемы (см., например, [26]).

Приведенная формула предполагает наличие у схемы нескольких выходов, переключающихся с одинаковой частотой. Иногда у элемента или узла имеется несколько выходов, сигналы на которых изменяются с разной частотой. Легко учесть и такой случай, что иллюстрируется примером двухразрядного счетчика (рис. 1.5, *г*), у которого частота переключения старшего разряда вдвое ниже, чем у младшего разряда. Для этого счетчика

$$P_D = C_{PD}\, U^2 F_I + C_{L1} U^2 F_I / 2 + C_{L2} U^2 F_I / 4.$$

Частоту переключения элементов, работающих в составе сложной схемы, обычно определить трудно. Для синхронных ЦУ известна тактовая частота устройства в целом, однако входящие в устройство элементы переключаются нерегулярно или с различными индивидуальными частотами, подавляющее большинство которых много ниже тактовой частоты. Точно оценить подобную ситуацию практически невозможно. В связи с этим при расчете динамической мощности частоту переключений элементов или их групп или иных фрагментов схемы оценивают приближенно, умножая тактовую частоту устройства F на коэффициент, называемый *переключательной активностью*. Для внутренних элементов и их групп, входящих в большие системы, практикой проектирования переключательная активность оценивается ориентировочно цифрой 0,1 (10%).

§ 1.2. Типы выходов цифровых элементов

Цифровые элементы (логические, запоминающие, буферные) могут иметь выходы следующих типов:

- ❒ логические;
- ❒ с третьим состоянием;
- ❒ открытые (с открытым стоком или коллектором).

Применение трех типов выходов объясняется различными условиями работы элементов в логических цепях, магистрально-модульных системах и т. д.

Логический выход

Логический выход формирует два уровня напряжения (U_0 и U_1). Логический выход стремятся сделать низкоомным, способным развивать большие токи для перезаряда емкостных нагрузок и, следовательно, высокого быстродействия элемента. Такой тип выхода характерен для логических элементов, используемых в комбинационных цепях.

В схемах логических выходов элементов ТТЛ(Ш) и КМОП (рис. 1.6, *а*, *б*) с противофазным управлением выходными транзисторами Т1 и Т2 обеспечиваются малые выходные сопротивления как при любом направлении переключения выхода, так и в обоих статических состояниях.

Особенность логических выходов состоит в том, что их *нельзя соединять параллельно*. Во-первых, это создает логическую неопределенность, т. к. в точке соединения выхода, формирующего логическую единицу, и выхода, формирующего логический нуль, не будет определенного результата. Во-вторых, при соединении выходов, находящихся в различных логических состояниях, возникает низкоомная цепь от источника питания до "земли" (для схемы КМОП через открытые транзисторы Т1 элемента, находящегося в единичном состоянии, и Т2 элемента, находящегося в нулевом состоянии). Ток этой цепи может достичь большой величины и для некоторых типов элементов достаточной даже для вывода из строя соединяемых элементов.

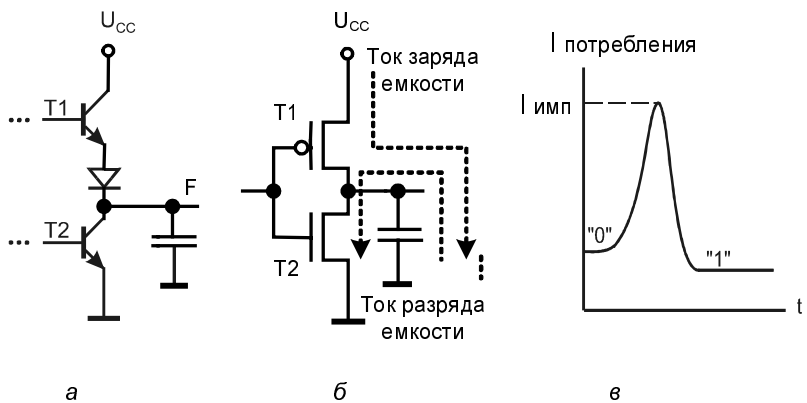


Рис. 1.6. Схемы логических выходов цифровых элементов (*а*, *б*) и график изменения тока, потребляемого ими в процессе переключения (*в*)

В выходной цепи с последовательным включением противофазно управляемых транзисторов Т1 и Т2 при переключениях из одного логического состояния в другое через оба транзистора протекают короткие импульсы тока. Эти токи протекают от источника питания на общую точку ("землю"). В статических состояниях таких токов быть не может, т. к. транзисторы Т1 и Т2 работают в противофазе, и один из них всегда заперт. Однако в переходном процессе из-за несинхронности переклю-

чения транзисторов возникает кратковременная ситуация, характерная для элементов ТТЛ(Ш), в которой проводят оба транзистора, что и порождает короткий *импульс сквозного тока* значительной величины (рис. 1.6, *в*). В современных КМОП-элементах путем подбора пороговых напряжений транзисторов сквозные токи устраняют. Однако импульсные токи возникают в цепях питания логических элементов не только из-за сквозного тока, но и *вследствие перезаряда емкостей* при их переключениях. При заряде емкости в линии питания возникает отрицательный импульс, а при разряде емкости в линии "земли" возникает положительный импульс (см. § 1.4). Это обстоятельство свидетельствует о неустранимости даже при строго синхронных переключениях транзисторов Т1 и Т2 такого неприятного факта, как *импульсные токи помех* в цепях питания цифровых элементов.

Выходы с тремя состояниями

Элементы с тремя состояниями выхода кроме состояний 0 и 1 имеют состояние "отключено", в котором ток выходной цепи пренебрежимо мал. В это (третье) состояние (ТС или Z) элемент переводится специальным управляющим сигналом, обеспечивающим запирание обоих транзисторов выходного каскада (Т1 и Т2 на рис. 1.6, *а*, *б*). Сигнал управления элементом типа ТС обычно обозначается как OE (Output Enable). При наличии разрешения (OE = 1) элемент работает как обычно, выполняя свою логическую операцию, а при отсутствии разрешения (OE = 0) переходит в состояние "отключено".

На рис. 1.7 показаны выходные каскады с третьим состоянием, используемые в схемотехнике КМОП. В схеме рис. 1.7, *а* высокий логический уровень сигнала OE означает разрешение работы, он открывает транзисторы Т3 и Т4 и, тем самым, позволяет нормально работать инвертору на транзисторах Т1 и Т2, через который данные DO передаются на выход. При низком уровне сигнала OE транзисторы Т3 и Т4 заперты и выход DO находится в состоянии "отключено". Для схемы рис. 1.7, *б* высокий логический уровень сигнала OE не мешает прохождению информационного сигнала на выход схемы, а низкий уровень запирает оба транзистора, переводя схему в третье состояние.

Буферные элементы типа ТС (с тремя состояниями) широко используются для управляемой передачи сигналов по тем или иным линиям. Буферы могут быть неинвертирующими или инвертирующими, а сигналы OE — H-активными или L-активными, что создает четыре типа буферных каскадов (рис. 1.8). Выходы типа ТС отмечаются в обозначениях элементов значком треугольника, как на рис. 1.8, или буквой Z (удобнее при выполнении документации с помощью компьютера).

Выходы типа ТС можно соединять параллельно при условии, что в любой момент времени активным может быть только один из них. В этом случае отключенные выходы не мешают активному формировать сигналы в точке соединения выходов. Эта возможность позволяет применять элементы типа ТС *в магистрально-модульных микропроцессорных и иных системах, где многие источники информации поочередно пользуются одной и той же линией связи.*

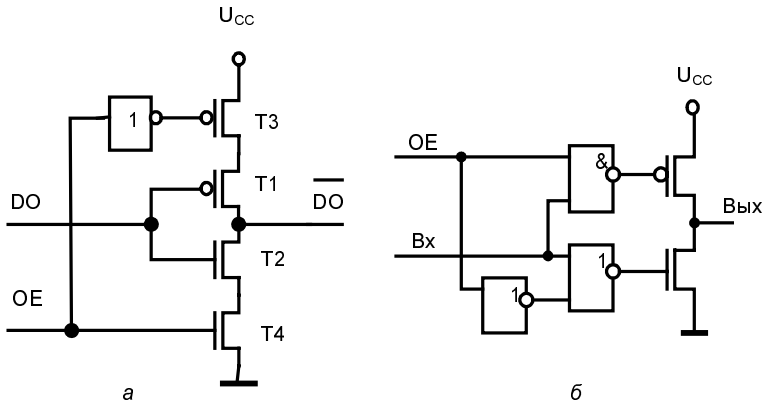


Рис. 1.7. КМОП-инверторы с тремя состояниями выхода

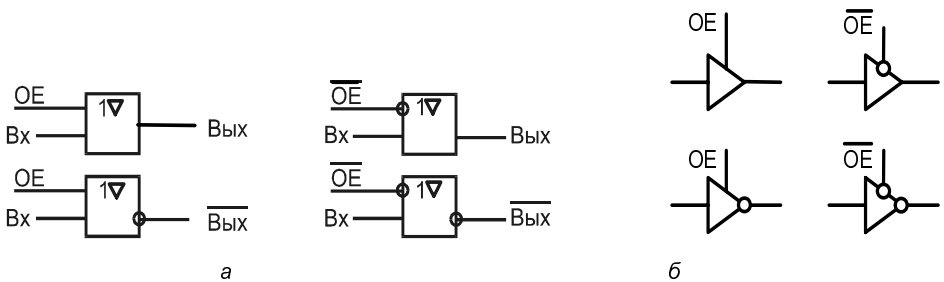


Рис. 1.8. Типы буферных каскадов с третьим состоянием (*а, б* — два варианта изображения)

Элементы ТС практически сохраняют такие достоинства элементов с логическим выходом, как быстродействие и высокая нагрузочная способность. В то же время они требуют обязательного соблюдения условия отключения всех выходов, соединенных параллельно, кроме одного, т. е. условия $OE_1 + OE_2 + ... + OE_n \le 1$. Кроме того, в отличие от выходов типа ОК (ОС) (с открытым коллектором или стоком), рассматриваемых далее, выходы ТС не дают возможности выполнять операции монтажной логики.

Элементы типа ТС приобрели широкое распространение, особенно в связи с массовым применением микропроцессорных систем с магистрально-модульными структурами. Они сохраняют свою популярность в проектах, реализованных на печатных платах. Однако для систем на одном кристалле ситуация иная. Это объясняется несколькими причинами. Так, в частности, недопустимость одновременной активности более чем одного источника сигнала в схемах очень высокого быстродействия делает их слишком критичными к задержкам и технологии изготовления элементов. Поэтому задача поочередного использования разными источниками одной и той же линии зачастую стала решаться иначе — с помощью мультиплексоров согласно принципу, показанному на рис. 1.9.

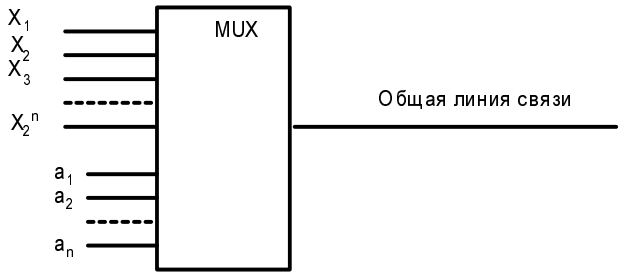


Рис. 1.9. Схема мультиплексируемой линии передачи, используемой несколькими источниками сигнала в режиме разделения времени

Здесь несколько входных информационных сигналов X_i присутствуют на соответствующих входах ($i = 1, 2, \ldots, 2^n$). На выход мультиплексора пропускается лишь один из этих сигналов в зависимости от адресующего кода $a_1 \ldots a_n$. Подробнее работа мультиплексора рассмотрена в *§ 2.4*.

Открытые выходы

В схемах на биполярных или МОП-транзисторах элементы с открытым коллектором (типа ОК, рис. 1.10, *а*) или стоком (типа ОС или OD, Open Drain, рис. 1.10, *б*) имеют выходную цепь, не соединенную с какими-либо цепями внутри микросхемы. В обоих случаях выходной транзистор управляется от предыдущей части схемы и может находиться в открытом или запертом состоянии. Открытое состояние трактуется как отображение логического нуля, запертое — единицы.

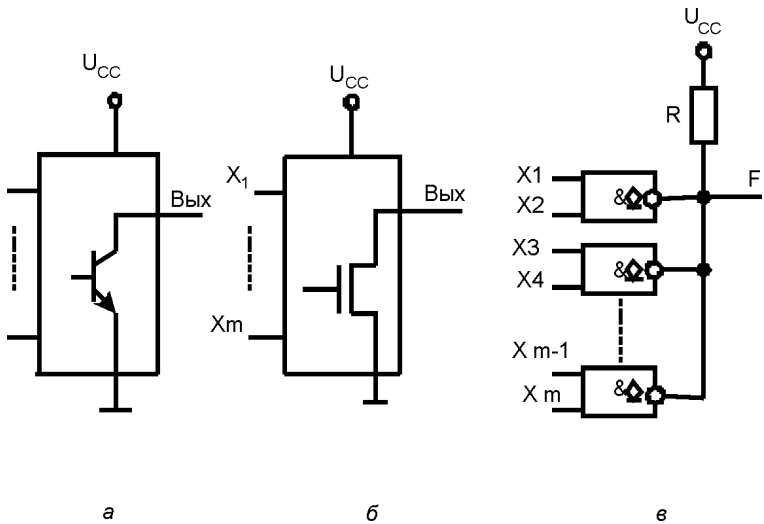
Открытый транзистор подключает выход к общей точке схемы и обеспечивает на выходе малое напряжение U_0. При запирании транзистора выход имеет неизвестный "плавающий" потенциал, т. к. не подключен к каким-либо цепям схемы. Поэтому для формирования высокого уровня напряжения при запирании транзистора к выходу требуется подключать внешние резисторы (или другие нагрузки), соединенные с источником питания.

Несколько открытых выходов (признаком открытого выхода служит наличие в обозначении элемента ромба с чертой внизу) можно соединять параллельно, подключая их к общей для всех выходов цепочке $U_{cc} - R$ (рис. 1.10, *в*). При этом можно получить режим поочередной работы элементов на общую линию, как и для элементов типа ТС, если активным будет лишь один элемент, а выходы всех остальных окажутся запертыми. Если же разрешить активную работу всех элементов, выходы которых соединены, то получим логическую операцию, называемую операцией *монтажной логики.*

В схеме монтажной логики высокое напряжение на выходе возникает только при запирании всех объединяемых транзисторов, т. к. отпирание хотя бы одного из них подключает выход к "земле" (точнее снижает выходное напряжение до уровня U_0). То есть для получения логической единицы на общем выходе требуется единичное

состояние всех выходов — для выходов выполняется монтажная операция И. Поскольку каждый элемент выполняет операцию Шеффера над своими входными переменными, общий результат окажется следующим

$$F = \overline{x_1x_2}\,\overline{x_3x_4} \ldots \overline{x_{m-1}x_m} = \overline{x_1x_2 \vee x_3x_4 \vee \ldots \vee x_{m-1}x_m}.$$



а *б* *в*

Рис. 1.10. Схема выходной цепи цифрового элемента с открытым коллектором (*а*), открытым стоком (*б*) и реализация монтажной логики (*в*)

Применяя элементы с ОК или ОС в магистрально-модульных структурах, нужно разрешать или запрещать их работу. Для элементов типа ТС это делается специальным сигналом ОЕ. Для элементов типа ОК или ОС в качестве входа ОЕ может быть использован один из обычных входов. Если, например, речь идет об элементе И-НЕ, то, подавая 0 на любой из входов, можно запретить работу элемента, поставив его выход в разомкнутое состояние независимо от состояния других входов. Уровень 1 на этом входе разрешит работу элемента под управлением остальных входов.

Положительная черта элементов ОС и ОК — защищенность от повреждений из-за ошибок управления, приводящих к параллельному включению нескольких выходов, а также возможность реализации операций монтажной логики. Недостаток — большая задержка переключения из 0 в 1, при котором выходная емкость заряжается через сопротивление резистора R. Сопротивление резистора нельзя сделать слишком малым, т. к. это привело бы к большим токам через открытый выходной транзистор и увеличило бы значение U_0. Поэтому положительный фронт выходного напряжения формируется относительно медленно (с постоянной времени RC). До порогового напряжения (середины полного перепада напряжения) экспоненциально изменяющийся сигнал изменится за время 0,7RC, что и составляет задержку $t_з^{01}$. Чтобы уменьшить задержки работы схем с ОК или ОС, стараются в качестве

рабочего использовать только переход из 1 в 0, для чего исходным состоянием выхода (состоянием до активизации схемы) делают высокий уровень (логическую единицу). Тогда при выполнении операции потребуется либо сохранить исходное состояние (если результат операции 1), либо перевести выход из единичного состояния в нулевое, что не сопровождается повышенной задержкой.

Для элементов ОС или ОК проектировщик должен задать сопротивление резистора R, определяемое конкретными условиями. Статические режимы задают ограничения R снизу и сверху. Значение R выбирается в этом диапазоне с учетом быстродействия схемы и потребляемой ею мощности.

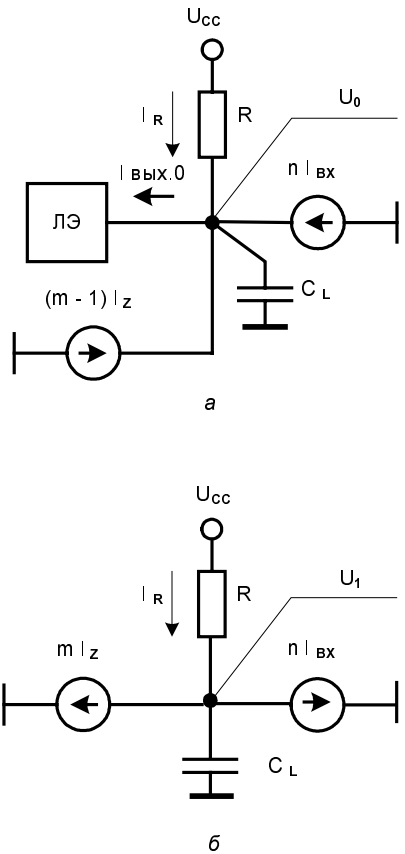


Рис. 1.11. Схема для расчета минимального и максимального значений сопротивления внешней цепи в каскадах с открытым выходом ОС или ОК

Ограничение R снизу связано с возможностью перегрузки открытого транзистора по току. На рис. 1.11, *а* показан режим, в котором нулевое состояние выхода обеспечивается одним логическим элементом ЛЭ (это наиболее неблагоприятный случай). Видно, что через выход ЛЭ протекает суммарный ток, складывающийся из токов резистора I_R, входных токов m элементов, подключенных к выходу ЛЭ, и то-

ков (m – 1) запертых транзисторов элементов, включенных параллельно с ЛЭ, каждый из которых равен I_Z, т. е.

$$I_{вых.0} = I_R + nI_{вх.0} + (m - 1)I_z ,$$

где $I_R = (U_{cc} - U_0)/R$; $I_{вх.0}$ — входные токи элементов-приемников сигнала при низком уровне входных напряжений; I_z — токи запертых выходов (обычно пренебрежимо малые). Чтобы ток выхода ЛЭ не превысил допустимого значения $I_{вых.0.max}$, следует соблюдать условие

$$R \geq (U_{cc} - U_0)/(I_{вых.0.max} - nI_{вх.0.max} - (m - 1)I_Z).$$

Состояние схемы при единичном выходе показано на рис. 1.11, *б*. Ограничение сопротивления R сверху связано с необходимостью гарантировать достаточно высокий уровень напряжения U_1, формируемого в схеме при запертом состоянии всех элементов с открытым выходом. Из схемы видно, что

$$I_R = mI_z + nI_{вх.1.max}$$

и

$$U_1 = U_{cc} - I_R R.$$

Чтобы соблюдалось условие $U_1 \geq U_{вых.1.min}$, необходимо иметь сопротивление

$$R \leq (U_{cc} - U_{вых.1.min})/(mI_z + nI_{вх.1.max}),$$

где $U_{вых.1.min}$ — паспортный параметр элемента.

Имея границы диапазона значений R, полученные, как показано выше, проектировщик должен выбрать его конкретное значение. Выбор вблизи нижней границы улучшает быстродействие схемы, а выбор вблизи верхней уменьшает потребляемую схемой мощность.

Выход с открытым эмиттером. Выход с открытым эмиттером характерен для элементов ЭСЛ (эмиттерно-связанной логики). Соединение друг с другом выходов с открытым эмиттером приводит к получению дополнительной операции монтажной логики ИЛИ.

Программируемый выход

В некоторых микросхемах используются выходные каскады с возможностью их программирования на два варианта — либо как каскада с открытым электродом, либо как каскада с третьим состоянием. На рис. 1.12 приведена схема такого каскада.

Легко убедиться, что показанная схема функционирует согласно табл. 1.1.

Таблица 1.1

OE	OD	Режим
0	X (произвольное значение)	Отключено (оба транзистора заперты)

Таблица 1.1 *(окончание)*

OE	OD	Режим
1	0	Обычный инвертирующий выходной каскад (нформационный сигнал управляет транзисторами T1 и T2)
1	1	Выход с открытым стоком (транзистор T1 заперт, транзистор T2 управляется входным информационным сигналом)

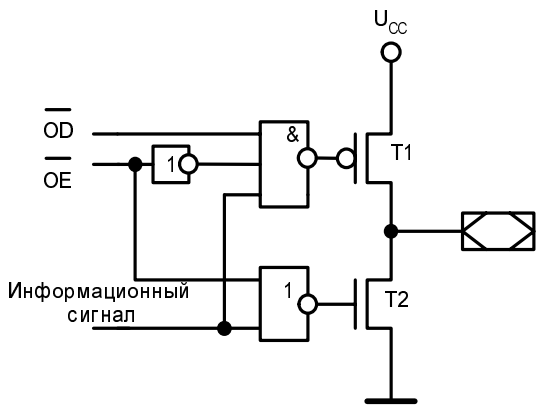


Рис. 1.12. Выходной каскад, программируемый на варианты OD (Open Drain) и Z

§ 1.3. Схемные особенности выводов КМОП-элементов

Pull-up- и Pull-down-резисторы

Рассмотрим ситуацию с выводами, к которым не подключены информационные сигналы. Оставлять высокоомные выводы КМОП-элементов разомкнутыми нельзя, поскольку на них могут наводиться произвольные потенциалы, что недопустимо, т. к. влияет на состояние элементов и придает схеме неконтролируемые, неопределенные состояния. Поэтому *потенциалы разомкнутых выводов фиксируют слабыми сигналами,* определяющими потенциалы выводов в отсутствие информационных сигналов и не играющими заметной роли при их присутствии. На рис. 1.13 показаны фрагменты типичных цепей, применяемых во входных схемах КМОП-элементов, в том числе цепи, иллюстрирующие взаимодействие слабого и сильного сигналов.

На рис. 1.13, *а* показаны цепи защиты входа транзистора от перенапряжений. Благодаря включению диодов диапазон изменения напряжений на затворе транзистора ограничен уровнями, близкими к напряжению питания U_{CC} и "земле".

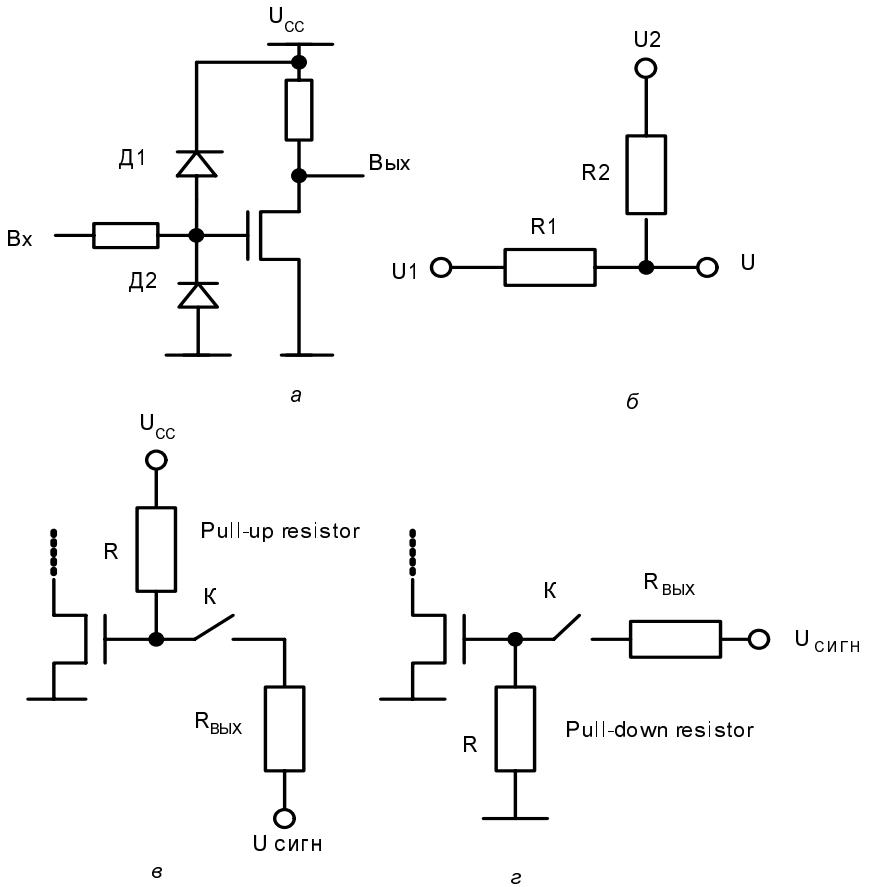


Рис. 1.13. Схемы защиты входа транзистора от перегрузки (*а*), взаимодействия сильного и слабого сигналов (б) и выводов с подтягивающим и заземляющим резисторами (*в*, *г*)

Рисунок 1.13, *б* иллюстрирует взаимодействие слабого и сильного сигналов. Слабым является сигнал, подаваемый в точку U через высокоомный резистор R1 (например, 100 кОм). Сильный сигнал подан в точку U через низкоомную цепь R2 (например, 100 Ом). Напряжение на выходе цепи согласно принципу суперпозиции можно выразить следующим образом

$$U = (U_1R_2 + U_2R_1)/(R_1 + R_2) = k_1U_1 + k_2U_2,$$

где $k_1 = R_2/(R_1 + R_2) = 0{,}001$; $k_2 = R_1/(R_1 + R_2) = 0{,}999$, т. е. $k_1 << k_2$.

Для исключения "плавающих" потенциалов к разомкнутым выводам подключают специальные резисторы, соединенные с источником питания или с общей точкой схемы. В первом случае это *"подтягивающие"* резисторы (Pull-up Resistors), во втором — *"заземляющие"* (Pull-down Resistors). Цепи с такими резисторами показаны на рис. 1.13, *в* и *г*. Если сопротивления подтягивающих и заземляющих резисторов велики, то сигналы, формируемые с их участием, относятся к слабым. Когда

к выводам подключаются сильные информационные сигналы, они преодолевают слабые сигналы, практически не влияющие на функционирование схем. Цепи фиксации потенциалов создают дополнительные токовые нагрузки на источники информационных сигналов, но эти нагрузки незначительны. Токи нагрузки в схеме с заземляющим резистором составляют U_1/R и U_0/R для единичного и нулевого состояний схемы, питающей рассматриваемый вывод (U_1 и U_0 — уровни кодирования 1 и 0). В схеме с подтягивающим резистором нагрузки для двух состояний схемы составляют $(U_{CC} - U_1)/R$ и $(U_{CC} - U_0)/R$.

Иногда к одним и тем же выводам подключают одновременно и подтягивающие и заземляющие резисторы, последовательно с которыми включены ключевые транзисторы. При программировании выбирается вариант фиксации потенциала вывода и согласно этому выбору замыкается один из ключевых транзисторов.

Выводы с запоминанием последнего значения сигнала

Кроме фиксации потенциалов с помощью подтягивающих и заземляющих резисторов применяют и схемы типа Weak Pin-keepers, которые позволяют сохранять на контакте прежнее значение потенциала после отключения контакта от источника сигнала (функция Pin Locking). Схемы типа Weak Pin-keepers основаны на применении обратной связи по слабому сигналу. Принцип их работы поясняется на рис. 1.14.

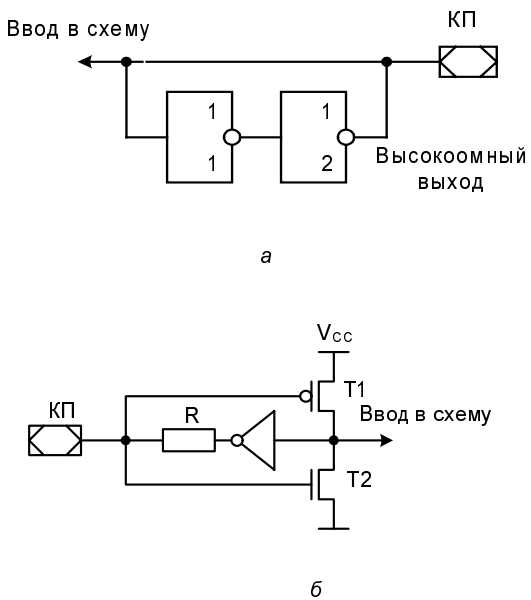


Рис. 1.14. Схемы входных цепей, обеспечивающие запоминание последнего значения информационного сигнала (*а*, *б*)

В цепи с автофиксатором (рис. 1.14, *а*) при подаче на вход логической единицы или нуля соответствующие напряжения дублируются сигналами на выходе второго инвертора. При отключении информационного сигнала, состояние входа не изменится, так как дублирующее напряжение поддерживает само себя, поскольку пара инверторов, соединенных в кольцо, представляет собою бистабильную ячейку (простейший триггер), имеющую свойство памяти. Чтобы выход второго инвертора не шунтировал информационные сигналы, он должен быть высокоомным (в практических схемах это сотни килоом).

В другом варианте (рис. 1.14, *б*) вводимые данные с контактной площадки подаются на затворы ключевых транзисторов T1 и T2. Высокий уровень напряжения открывает транзистор T2 с n-каналом и запирает транзистор T1 с p-каналом. При этом в линию входа подается нулевой сигнал, а в точку, связанную с контактной площадкой, — единичный, так как буфер является инвертирующим. Этот обратный сигнал подается на контактную площадку через высокоомное сопротивление R, благодаря чему он является слабым (обычно сопротивление резистора R составляет не менее 100 кОм). При отключении источника сигнала от контактной площадки, существовавший до отключения, потенциал будет поддерживаться слабым сигналом, и вводимая величина не изменится. При подключении источника сигнала к контактной площадке он будет преодолевать слабый сигнал цепи "буфер — резистор". На преодоление действия слабого сигнала обычно затрачиваются токи не более десятков микроампер.

§ 1.4. Паразитные связи по цепям питания. Фильтрация питающих напряжений

При проектировании и эксплуатации ЦУ необходимо предусматривать меры борьбы со сбоями из-за помех. Один из источников сбоев — токовые импульсы в цепях питания схемы. При переключениях элементов из-за перезаряда нагрузочных емкостей и сквозных токов в выходных каскадах в цепях питания создаются кратковременные импульсные токи, т. е. *одни элементы становятся источниками помех для других*.

Импульсный ток переключающегося логического элемента ЛЭ1 протекает от источника питания $U_{ИП}$ на общую точку схемы GND по линиям, имеющим полные сопротивления $Z_{ИП}$ и Z_{GND} (рис. 1.15, *а*). Главную часть полных сопротивлений в зависимости от типа цепи составляют индуктивные или резистивные составляющие. Для кабелей, выводов соединителей, выводов корпусов ИС и других внешних относительно кристаллов цепей основную роль играют индуктивности, на которых выделяются напряжения $U_L = L\ dI/dt$. Во внутрикристальных цепях распределения мощности главную роль играют резистивные составляющие, на которых падают напряжения IR. Для дальнейшей качественной иллюстрации происходящих процессов и понимания методов борьбы с помехами от импульсных токов конкретный характер сопротивлений не имеет принципиального значения.

Импульсный ток создает на линии питания отрицательный импульс напряжения, а на линии общей точки ("земли") — положительный. Эти импульсы воздействуют на подключенный вблизи элемента ЛЭ1 элемент ЛЭ2.

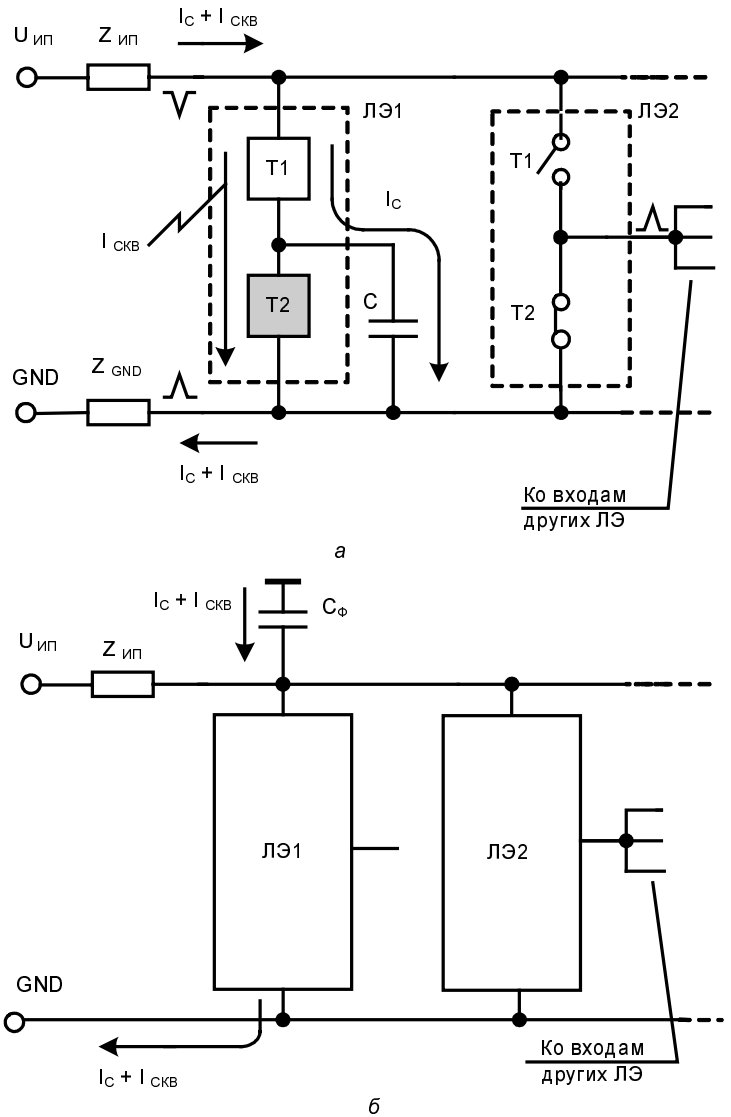


Рис. 1.15. Схемы, поясняющие процесс возникновения импульсных помех при переключении цифрового элемента (*а*), и пути протекания импульсного тока при наличии в схеме фильтрующего конденсатора (*б*)

На рис. 1.15, *а* показаны составляющие импульсного тока $I_{ИМП}$ и путь проникновения помех из цепей питания в сигнальные цепи схемы. *Импульсный ток состоит из сквозного тока $I_{СКВ}$ и тока перезаряда нагрузочной емкости $I_С$.* Для современ-

ных элементов субмикронной технологии основной составляющей импульсного тока является ток перезаряда емкостей.

На рисунке показан переход логического элемента ЛЭ1 из нулевого состояния в единичное, при котором верхний транзистор выходного каскада открывается, а нижний закрывается. Некоторая задержка запирания нижнего транзистора (это состояние иллюстрируется затемненным изображением транзистора T2) создает кратковременную проводимость между источником питания и землей и, соответственно, импульс сквозного тока.

Рост напряжения на выходе ЛЭ1 создает ток заряда емкости нагрузки C_L. Вместе эти составляющие образуют импульс тока. Если, как показано на рисунке, элемент ЛЭ2 находится в состоянии логического нуля, то его выход через открытый транзистор, отображаемый замкнутым ключом, связан с линией GND, следовательно, импульс с этой линии без каких-либо преград попадает на выход элемента ЛЭ2, откуда сможет распространяться и далее по обычным сигнальным цепям, ведущим ко входам последующих логических элементов. При единичном состоянии элемента ЛЭ2 на его выход аналогичным образом пройдет отрицательный импульс помехи с линии источника питания.

Система распределения мощности должна поставлять большие постоянные токи (современные кристаллы с низковольтным питанием могут потреблять десятки ампер каждый) при малом выходном сопротивлении для высокочастотных импульсов. Для достижения этой цели разработан ряд методов, которые можно разделить на активные и пассивные. Широко применяются простые пассивные методы. Их сущность состоит в создании *"хорошей земли" и фильтрации напряжений питания*.

Качество заземления

"Качество земли" улучшается конструктивными мерами, снижающими сопротивление Z_{GND}: шины "земли" делаются утолщенными, для их реализации отводят целые плоскости многослойных конструкций (плат и кристаллов), систему "заземления" соединяют с несколькими выводами корпуса, чтобы сократить пути прохождения токов в этой системе и т. д. Традиционная передача питающих напряжений от вводов на краях кристалла до отдельных элементов или устройств уже не обеспечивает требуемой для многих современных БИС/СБИС малой длины и, соответственно, малых сопротивлений цепей питания. В связи с этим широкое распространение получили кристаллы с распределением вводов питания по всей площади кристалла и такие же корпуса для них, например, BGA (Ball Grid Array). Число выводов для цепей питания и "земли" у одного корпуса может исчисляться несколькими десятками.

Наряду с конструктивными методами применяют и схемотехнические — в цепи выходных каскадов логических элементов и буферов добавляют небольшие сопротивления, ограничивающие сквозные токи и токи перезаряда емкостей; используют элементы с управляемой крутизной фронтов для уменьшения производных сигнальных напряжений и токов; применяют развязывающие каскады на выходах ИС

для ограничения емкостных нагрузок на этих выходах; используют *фильтрацию питающих напряжений.*

Фильтрация напряжений питания

Для фильтрации напряжений питания (устранения из них импульсных помех) между линиями U_{CC} и "землей" *включают конденсаторы.* Установка фильтрующих конденсаторов $C_Ф$ создает путь (рис. 1.15, *б*), по которому замыкаются импульсы сквозного тока и токи перезаряда емкостей, минуя сопротивление $Z_{ИП}$. Иначе говоря, конденсаторы играют роль резервуаров заряда, достаточного для кратковременного создания импульсного тока. Конденсаторы должны иметь достаточно большую емкость и малое сопротивление для высокочастотных сигналов.

Фильтрующие конденсаторы. Типы конденсаторов и их емкости определяются в соответствии с параметрами импульсов, которые следует подавлять. Необходимость подавления помех в разных частотных диапазонах ведет к применению по крайней мере трех разновидностей конденсаторов:

- МОП-конденсаторов, изготовленных на кристалле;
- керамических высокочастотных конденсаторов;
- электролитических конденсаторов большой емкости.

Схема замещения конденсатора помимо емкости C включает в себя паразитную индуктивность выводов L_C и последовательное сопротивление R_S. Полное сопротивление (импеданс) конденсатора

$$Z_C = j\omega L_C + R_S + 1/j\omega C = j\omega L_C + R_S - j/\omega C.$$

На частоте резонанса, при которой емкостное и индуктивное сопротивления, противоположные по знаку, по модулю равны друг другу, импеданс будет представлен только сопротивлением R_S. Ниже этой частоты импеданс имеет емкостной характер, выше — индуктивный, т. е. конденсатор сохраняет эффективность только в определенной области частот. Конденсаторы разных типов имеют разные рабочие диапазоны частот, поэтому для фильтрации напряжений питания применяют одновременно несколько разновидностей конденсаторов.

Переключение логического вентиля создает самые высокочастотные помехи (время переключения вентиля может составлять десятки пикосекунд). Для их фильтрации нет альтернатив конденсаторам, выполненным на самом кристалле в виде МОП-структур. Такие конденсаторы имеют верхнюю границу диапазона рабочих частот от единиц до десятков гигагерц и емкости порядка сотен фемтофарад или единиц пикофарад (фемтофарада равна 10^{-15} фарады).

Керамические конденсаторы работают в диапазоне частот до десятков мегагерц, имеют небольшое сопротивление R (порядка 0,1 Ом), до этой величины снижается их сопротивление в области резонанса. Они широко применяются для фильтрации питания на печатных платах, для чего подключаются к выводам питания корпусов микросхем возможно более короткими проводниками. Емкости керамических конденсаторов обычно составляют единицы-десятки нанофарад.

Электролитические конденсаторы обладают большой емкостью (в схемотехнике применяют конденсаторы с емкостями 10…1000 мкФ) и активными сопротивлениями порядка 1…0,05 Ом для конденсаторов меньшей и большей емкости соответственно. Их частотный диапазон — до сотен или единиц килогерц. Подключаются электролитические конденсаторы обычно к вводам питания для целой платы.

В совокупности иерархия фильтрующих конденсаторов подавляет помехи в широкой полосе частот. Рекомендации по числу, типу и емкости фильтрующих конденсаторов для микросхем стандартных серий приводятся в справочниках по их применению.

Фильтрация внутрисхемными конденсаторами. Существенное облегчение фильтрации на кристалле возникает благодаря тому, что *схемные емкости неперекючающихся логических элементов также работают как фильтрующие.* Поскольку в каждом такте работы не переключается основная масса элементов устройства (80…90%), полезная роль паразитных емкостей для фильтрации напряжений питания может оказаться настолько значительной, что изготовление на кристалле специальных МОП-конденсаторов может не потребоваться.

§ 1.5. Передача сигналов. Помехи в сигнальных линиях. Сигнальные линии повышенного качества

При работе ЦУ в линиях связи может возникнуть множество помех различного рода, способных нарушить нормальную работу схемы. К их числу относятся перекрестные помехи, электромагнитные наводки и паразитные колебания из-за несогласованности волновых сопротивлений линий связи.

Перекрестные помехи и электромагнитные наводки

Перекрестные помехи (Cross talks) порождаются взаимовлиянием близлежащих линий, передающих сигналы. Пусть линия — источник помехи — является близлежащей для линии, испытывающей воздействие помехи. Тогда между ними существует связь через паразитную емкость $C_{пом}$ (рис. 1.16, *а*). Схема замещения рассматриваемой цепи может быть представлена в виде рис. 1.16, *б*, где

$$R = R_{вых.1}R_{вх.2}/(R_{вых.1} + R_{вх.2}).$$

Если считать фронт помехи линейным, изменяющимся по закону $U_{пом}(t) = at$, где

$$a = (U_1 - U_0)/t_ф = U/t_ф,$$

то напряжение помехи на входе элемента ЛЭ2 будет определяться соотношением (для времен от нуля до $t_ф$)

$$U_{вх.2}(t) = a\,[1 - \exp(-t/RC)]\,RC,$$

т. е. будет пропорционально крутизне фронта.

Для борьбы с перекрестными помехами запрещается параллельное расположение близких и длинных сигнальных линий, между такими линиями размещаются экранирующие заземленные проводники (так, в частности, поступают при применении плоских кабелей), применяются коаксиальные кабели, витые пары, уменьшается крутизна фронтов передаваемых сигналов и др.

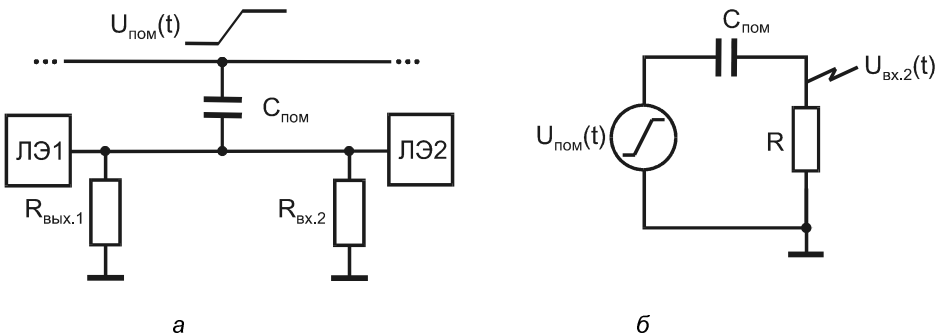


Рис. 1.16. Схема, поясняющая процесс возникновения перекрестных помех в цифровых устройствах (*а*), и схема замещения (*б*)

Электромагнитные наводки создаются внешними полями. Борьба с ними также ведется конструктивными методами — экранированием устройства.

Искажения сигналов в несогласованных линиях

Короткие и длинные линии связи. Паразитные колебания из-за несогласованности волновых сопротивлений возникают в связях, которые именуются длинными, причем речь не идет об абсолютных значениях длины, важно лишь соотношение длины линии и длины волны передаваемого сигнала. При их соизмеримости линии относятся к длинным. *Поведение длинной линии резко отличается от поведения короткой.*

Так как импульсные сигналы характеризуются широким спектром гармонических частот, говорить о длине волны для них затруднительно, и рекомендации по отнесению линий связи к коротким или длинным в значительной мере вырабатываются практикой. Например, граничную длину линии часто определяют по условию: время прохождения сигнала по линии должно быть на порядок меньше длительности передаваемого фронта. Скорость распространения сигнала в линии равна $V = V_c/\sqrt{\varepsilon}$, где V_c — скорость света в вакууме (30 см/нс); ε — диэлектрическая постоянная среды, в которой распространяется сигнал. Практически V = = 15...20 см/нс.

Распространение сигналов в длинных линиях. Схема замещения длинной линии без потерь состоит из цепочки LC-звеньев, где L и C — погонные параметры индуктивности и емкости (т. е. приходящиеся на единицу длины). Такая линия

(рис. 1.17, *а*) имеет волновое сопротивление $Z_0 = \sqrt{L/C}$. Величина Z_0 зависит от конструкции линии и обычно лежит в диапазоне от 50 до 100 Ом. На рис. 1.17, *б* в качестве примера показана реализация линии в виде коаксиального кабеля.

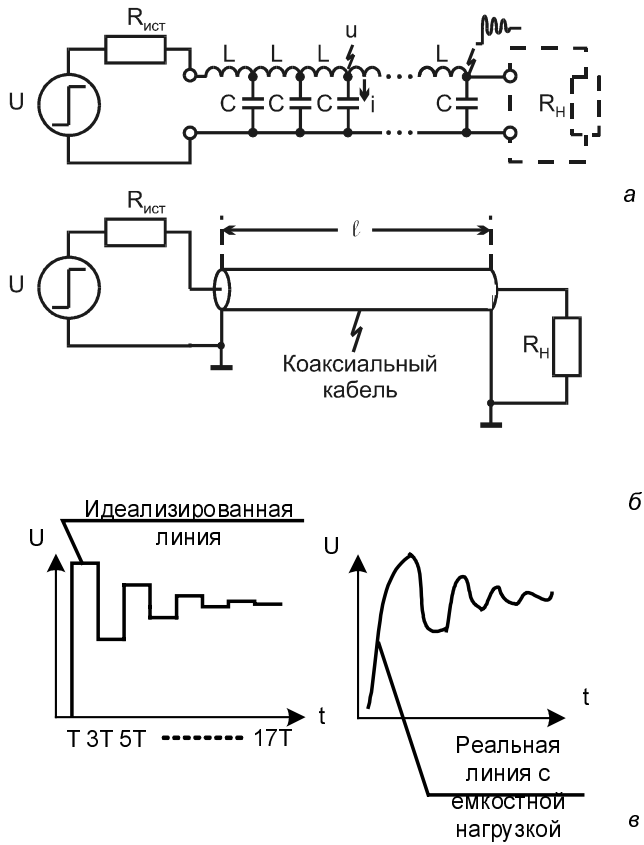


Рис. 1.17. Схема замещения длинной линии без потерь (*а*), схема с реализацией линии в виде коаксиального кабеля (*б*) и временные диаграммы сигналов в идеализированной и реальной линиях (*в*)

Физически волновое сопротивление соответствует отношению напряжения к току в точке линии, которой достигает распространяющаяся волна. Пока волна распространяется в линии, это отношение $u/i = Z_0$ остается неизменным. В конце линии ситуация зависит от подключенного к линии сопротивления. Если в конце линии подключено сопротивление $R_H = Z_0$, то отношение u/i сохраняется, падающая волна не встречает неоднородности и целиком поглощается нагрузкой.

Если в конце линии $R_H \neq Z_0$, то отношение u/i согласно закону Ома сохраниться не может, и должно произойти искажение волны. Оно трактуется как появление отраженной волны, параметры которой таковы, что сумма падающей и отраженной

волн соответствует условиям в конце линии (соблюдению закона Ома). Отношение амплитуд отраженной и падающей волн равно *коэффициенту отражения*

$$\rho = (R_H - Z_0)/(R_H + Z_0).$$

Величина коэффициента отражения меняется от минус единицы (при $R_H = 0$, т. е. у короткозамкнутого конца линии) до плюс единицы (при сопротивлении R_H равном бесконечности, т. е. у разомкнутого конца линии).

Отраженная волна распространяется обратно к началу линии. Если в начале линии подключено сопротивление, равное Z_0, то отраженная волна поглощается целиком, и режим линии устанавливается окончательно. В противном случае в начале линии также происходит отражение волны, которая вновь пойдет по линии от ее начала к концу. Амплитуды волн будут уменьшаться тем быстрее, чем меньше по абсолютной величине коэффициенты отражения в концах линии. Возможное многократное отражение способно затянуть переходные процессы в линии на время, равное десяткам T_0, где T_0 — время распространения сигнала по линии ($T_0 = \lambda / V$, где λ — длина линии).

Графики напряжений в конце и начале идеализированной ненагруженной линии имели бы ступенчатый характер, поскольку напряжения в указанных точках менялись бы скачком в моменты очередного прихода волны к концу и началу линии. В реальных условиях скачков напряжения не происходит, в первую очередь из-за характерных для цифровых устройств емкостных нагрузок линий связи. Действующая в конце линии паразитная емкость C_L создает постоянную времени Z_0C_L, ограничивающую скорость нарастания напряжений на емкости. Сглаживанию скачков напряжений содействуют и другие факторы (наличие паразитных колебательных контуров и др.). Таким образом, графики напряжений в конце идеализированной ненагруженной линии и реальной линии будут иметь вид, показанный на рис. 1.17, *в*.

Для устранения недопустимых при передаче цифровых сигналов паразитных колебаний используют параллельное или последовательное согласование волновых сопротивлений.

Параллельное согласование волновых сопротивлений

При параллельном согласовании в конце линии включают резистор (называемый *терминатором*), чтобы сделать сопротивление нагрузки равным волновому. Это дает полное устранение паразитных колебаний при времени передачи сигнала, равном T_0. Недостаток способа — потребление значительных статических токов от источника сигнала. После завершения переходных процессов на выходе линии устанавливается напряжение U_1 или U_0. Под этим напряжением постоянно находится резистор-терминатор, сопротивление которого равно Z_0 (входным сопротивлением приемника сигнала пренебрегаем, считая его большим и несоизмеримым с Z_0, в противном случае сопротивление терминатора подбирается так, чтобы параллельное соединение входного сопротивления приемника сигнала и резистора-терминатора давало величину Z_0). Сопротивление на выходе линии оказывается малым (ти-

пичные значения волновых сопротивлений линий передачи сигналов 50...100 Ом). Таким образом и в ходе переходных процессов и после их завершения источник сигнала нагружен на малое сопротивление. Ток на выходе источника сигнала может оказаться неприемлемо большим.

Для поиска приемлемого варианта включения резисторов на выходе линии можно просмотреть несколько схемных вариантов (рис. 1.18). Для варианта (рис. 1.18, *а*) статические токи в зависимости от логического состояния источника сигнала равны U_0/Z_0 или U_1/Z_0.

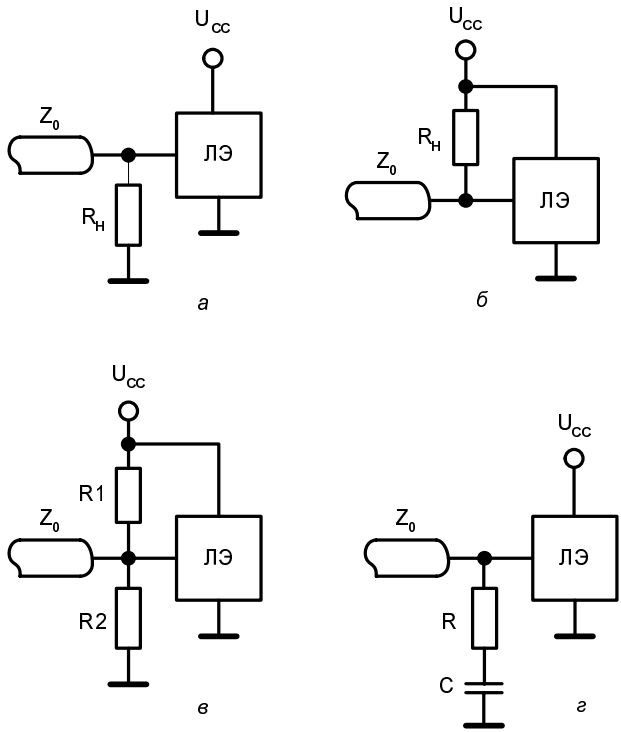


Рис. 1.18. Варианты параллельного согласования волновых сопротивлений линии передачи и нагрузки

Для другого варианта (рис. 1.18, *б*) токи равны $(U_{CC} - U_0)/Z_0$ и $(U_{CC} - U_1)/Z_0$ и также могут быть неприемлемо велики (заметим, что для выходных цепей многих элементов ТТЛ(Ш), находящихся в единичном состоянии, ток, втекающий извне, не возникает, т. к. на его пути включен диод в обратном направлении. Поэтому перегрузка током указанного направления не опасна, но отсутствие тока через резистор R_H приводит к повышению уровня логической единицы до U_{CC}).

Расчет сопротивлений делителя на выходе линии. В следующем варианте (рис. 1.18, *в*) на выходе линии включен делитель напряжения, составленный резисторами R_1 и R_2. Этот вариант обладает большей гибкостью, т. к. позволяет варьировать нагрузку на линию, изменяя отношение сопротивлений R_1/R_2. В данном случае линия нагружена на сопротивление параллельного соединения резисторов

R_1 и R_2, поэтому обязательным является условие $R_1R_2/(R_1+R_2) = Z_0$ (здесь, как и ранее, не учитываем входное сопротивление приемника сигнала, считая его достаточно большим). Переходя от сопротивлений к проводимостям, можно записать это же условие в более простом виде как

$$Y_1 + Y_2 = Y_0, \qquad (1.1)$$

где $Yi = 1/Ri$ $(i = 0, 1, 2)$.

Условие (1.1) может быть выполнено при разных значениях сопротивлений резисторов. Если удастся подобрать такие значения R_1 и R_2, при которых токи источника сигнала в обоих логических состояниях не будут превышать допустимые значения, то задача согласования волновых сопротивлений линии и нагрузки будет успешно решена. Методику поиска приемлемых значений R_1 и R_2 (если они существуют) проиллюстрируем на примере схемы с КМОП-элементами, которая может быть представлена в виде рис. 1.19, *а*.

Для этой схемы имеем

$$I_{ИСТ.0} = I_1 - I_2 = (U_{CC} - U_0)Y_1 - U_0Y_2,$$
$$I_{ИСТ.1} = I_2 - I_1 = U_1Y_2 - (U_{CC} - U_1)Y_1.$$

В свою очередь токи $I_{ИСТ.1}$ и $I_{ИСТ.2}$ не должны превышать допустимые для данного источника значения, т. е. должны удовлетворять условиям

$$I_{ИСТ.0} \le I_{ИСТ.0.ДОП},$$
$$I_{ИСТ.1} \le I_{ИСТ.1.ДОП}.$$

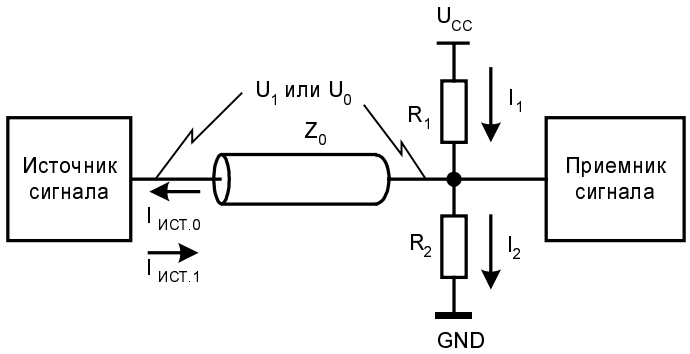
Из полученных равенств и неравенств легко получить условия работы источника сигнала без токовых перегрузок

$$Y_1 \le (I_{ИСТ.0.ДОП} - U_0Y_2)/(U_{CC} - U_0), \qquad (1.2)$$
$$Y_1 \ge (U_1Y_2 - I_{ИСТ.1.ДОП})/(U_{CC} - U_1). \qquad (1.3)$$

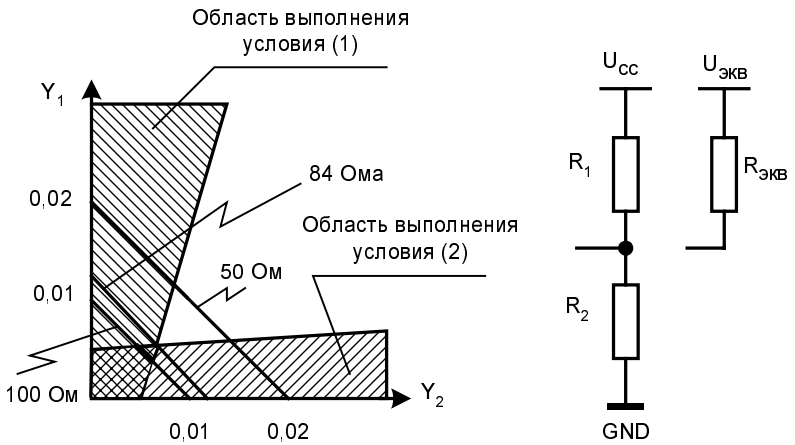
Условия (1.1), (1.2) и (1.3) должны выполняться одновременно. Все они выражаются линейными зависимостями одной проводимости от другой, что позволяет представить решаемую задачу графически. В координатах Y_1,Y_2 условие (1.1) будет представлено диагональной линией, а неравенства (1.2) и (1.3) — областями (зонами). Приемлемая рабочая точка должна обязательно лежать на диагональной линии и находиться одновременно в обеих допустимых областях.

На рис. 1.19, *б* приведено графическое решение задачи согласования волновых сопротивлений с помощью делителя напряжения для элемента КР1554ЛА3 (элемента 2И-НЕ). Напряжение питания элемента принято равным 4,5 В, уровни U_1 и U_0 — отличающимися от U_{CC} и "земли" на 10% от напряжения питания, допустимые токи в обоих логических состояниях 24 мА (допустимая норма для состояний с длительностью более 20 мс). При этих параметрах условия (1.1), (1.2) и (1.3) приобретают вид

$$Y_1 = Y_0 - Y_2,$$
$$Y_1 \le 0{,}0059 - 0{,}11Y_2,$$
$$Y_1 \ge 9Y_2 - 0{,}053.$$



а



б *в*

Рис. 1.19. К пояснению методики согласования волновых сопротивлений линии и нагрузки с помощью резистивного делителя напряжения

Из рисунка видно, что при волновых сопротивлениях линии менее 84 Ом согласование невозможно. Для линии с волновым сопротивлением 100 Ом утолщенным участком диагональной линии показан отрезок, на котором можно выбирать пары значений Y_1 и Y_2.

Известно, что делитель напряжения по теореме Тевенина можно заменить эквивалентной цепью, как показано на рис. 1.19, *в*. При этом

$$R_{ЭКВ} = R_1R_2/(R_1 + R_2),$$

$$U_{ЭКВ} = U_{CC}R_2/(R_1 + R_2).$$

На основании теоремы Тевенина данные, полученные для параметров делителя напряжения, можно использовать для схемы с одним резистором, имеющим сопро-

тивление Z_0, если применить для схем согласования источник питания с напряжением, равным $U_{ЭКВ}$. В последнее время такое решение в схемах БИС/СБИС встречается все чаще.

Наряду с чисто резисторными цепями пользуются также цепочками с включением последовательно с резистором емкости С, которая предотвращает потребление тока в статике (см. рис. 1.18, *г*). Неудобством при этом является появление такого элемента, как конденсатор. Емкость конденсатора должна быть достаточно большой, чтобы постоянная времени Z_0C была многократно больше длительности такта передачи данных. Кроме того, при частых переключениях емкость перестает играть роль эффективного ограничителя тока, поскольку не успевает полностью перезарядиться, и следовательно, все время проводит ток.

Последовательное согласование волновых сопротивлений

При последовательном согласовании в начале линии последовательно включается резистор $R_{доп}$, сопротивление которого совместно с выходным сопротивлением источника сигнала $R_{ист}$ дает величину Z_0 (рис. 1.20). На выходе линии включено высокое входное сопротивление элемента-приемника, следовательно, там коэффициент отражения приблизительно равен единице, и амплитуда отраженной волны приблизительно равна амплитуде падающей.

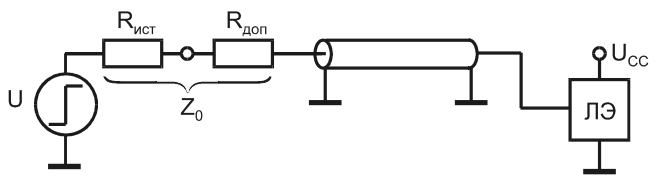


Рис. 1.20. Схема последовательного согласования волновых сопротивлений

Переходный процесс в этом случае протекает следующим образом. Ступенчатое изменение напряжения источника сигнала U создает на входе линии перепад напряжения U/2 (т. к. $R_{ист} + R_{доп} = Z_0$). Перепад половинной амплитуды распространяется по линии и через время T_0 достигает ее конца. Коэффициент отражения в конце линии равен единице ($R_{вх} >> Z_0$ и влиянием $R_{вх}$ пренебрегаем). Амплитуда отраженной волны равна также U/2, в итоге в конце линии сразу и окончательно устанавливается напряжение U. Отраженная волна возвращается к началу линии, где поглощается. Таким образом, на выходе линии процесс заканчивается через время T_0, а на входе через $2T_0$.

При последовательном согласовании отсутствуют статические токи нагрузки на источник сигнала, характерные для параллельного согласования. Однако в переходном процессе вход линии проявляет себя как сопротивление, равное Z_0. Следовательно, в переходном процессе источник сигнала нагружен током $U/2Z_0$, где U —

перепад напряжения источника. Поэтому на высоких частотах передач и при последовательном согласовании возникают проблемы токовой нагрузки источника сигнала.

Последовательное сопротивление в сигнальной цепи может уменьшить амплитуду передаваемых напряжений, если входное сопротивление приемника недостаточно велико, так что для схем на элементах с ощутимым входным током (ТТЛШ) требуется проконтролировать эту возможность. Если от линии связи берутся отводы в середине или начале линии, то задержка передачи сигнала может достигать величины $2T_0$.

Реальное положение в технике борьбы с отражениями в длинных линиях сложнее, чем было описано, т. к. сопротивления в схемах непостоянны и зависят от логического состояния элемента, уровня сигнала и т. д. В частности, выходные сопротивления источников сигнала в состояниях логического нуля и логической единицы обычно неодинаковы. Поэтому последовательное согласование не может быть вполне правильным и приходится довольствоваться приближенным решением.

Согласование волновых сопротивлений в конце и начале линии

Если требуется высокая степень подавления искажений сигнала из-за несогласованности волновых сопротивлений, то применяют одновременно оба вида согласований — параллельное и последовательное (это характерно для линий передач высшего быстродействия). Одновременное применение обоих типов согласований *резко улучшает форму передаваемых сигналов, но уменьшает их амплитуду в два раза.* Действительно, в этом случае происходят следующие процессы: в начале линии входной перепад делится на два согласующим сопротивлением и входным сопротивлением линии, которые равны. Половинный перепад распространяется по линии к приемнику сигнала, где падающая волна поглощается без отражения, поскольку нагрузка согласована с волновым сопротивлением линии. На этом все и заканчивается. Такой характер происходящих процессов исключает возможность включения линии между двумя элементами цифрового типа, т. к. сигнал половинной амплитуды непригоден для восприятия логическим элементом. Поэтому линия с одновременным согласованием сопротивлений в начале и конце должна работать на приемник аналогового типа (усилитель), который может восстанавливать нормальную амплитуду сигнала. Заметим, что используются и варианты, в которых применены одновременно более чем два вида согласований. Такие варианты будут показаны далее.

Линии передачи сигналов

При разработке ЦУ следует уделять надлежащее внимание линиям передачи сигналов.

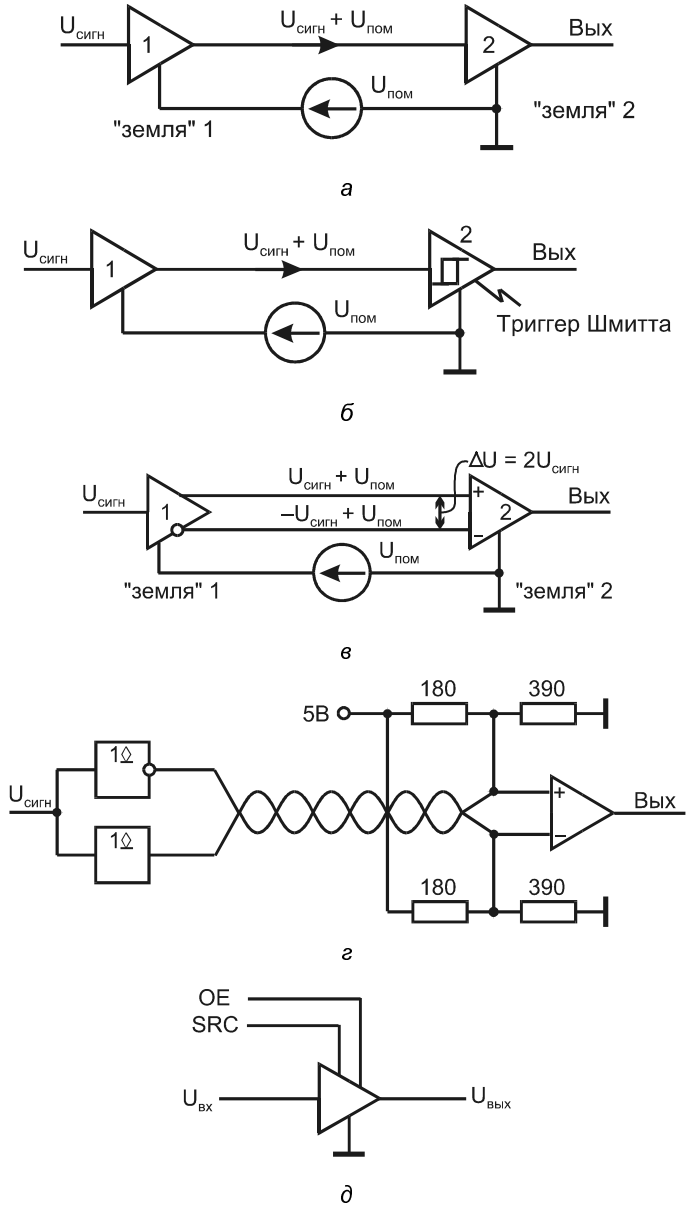


Рис. 1.21. Простейшая схема передачи цифрового сигнала (*а*), схема с гистерезисным приемником (*б*), передача сигнала дифференциальным способом (*в*), пример схемы помехоустойчивой передачи сигнала (*г*), буфер с регулируемой крутизной фронта (*д*)

Это важно при проектировании печатных плат, и становится особенно острой проблемой в БИС/СБИС, где преобладающая часть площади кристалла, задержек сигналов и потребляемой мощности зачастую относится именно к системе межсоединений. Ряд рекомендаций по вопросам передачи сигналов уже был высказан ранее

(ограничения на параллельные размещения сигнальных линий, согласование волновых сопротивлений в длинных линиях). Отметим теперь особенности технической реализации межсоединений.

На платах межсоединения выполняются одиночными проводниками над "земляной" плоскостью, двумя проводниками, витыми парами, микрополосковыми линиями, коаксиальными кабелями малого диаметра и др.

Схема соединения одиночным проводником (рис. 1.21, *а*) изображена с учетом напряжения помехи, которая может возникать между "землями" двух элементов. В этом случае помеха передается на вход приемника сигнала.

Помехоустойчивость передачи повышается, если элемент-приемник обладает *гистерезисными свойствами*, как, например, триггер Шмитта (рис. 1.21, *б*). Благодаря гистерезисной характеристике приемника, для переключения в состояние логической "1" нужно подать на вход напряжение, значительно превышающее пороговое, а для переключения в "0" — значительно меньше, чем пороговое. Это повышает уровень допустимых помех, причем тем больше, чем шире петля гистерезиса.

Значительное улучшение может дать передача сигнала по двум линиям *(дифференциальная передача)*, показанная на рис. 1.21, *в*. Приемником сигнала служит дифференциальный усилитель (или компаратор). На его верхнем входе действует напряжение $U_{сигн} + U_{пом}$, а на нижнем напряжение $-U_{сигн} + U_{пом}$. Дифференциальный приемник воспринимает разность напряжений между входами, которая равна $2U_{сигн}$ и не содержит напряжения помех. Перекрестные помехи в данном случае также значительно ослабляются, поскольку появляются в обоих проводниках близкими по величине, так что их разность, ощущаемая приемником, мала.

На рис. 1.21, *г* для примера приведена схема помехоустойчивой дифференциальной передачи сигнала по витой паре. По волновому сопротивлению витая пара согласуется резистором-терминатором, выполненным в виде делителя (резисторы 180 и 390 Ом), эквивалентное сопротивление которого относительно выхода равно 120 Ом.

Витые пары часто применяются в ЦУ. Для примера укажем параметры витой пары проводников типа МНВ $2 \times 0{,}05$ мм^2:

- волновое сопротивление 100 Ом;
- сопротивление проводника постоянному току 0,35 Ом/м;
- коэффициент перекрестной помехи 0,15;
- время задержки сигнала 6 нс/м.

На рис. 1.21, *д* изображен буфер с третьим состоянием и регулировкой крутизны нарастания выходного сигнала. Введением/снятием третьего состояния управляет вход OE (Output Enable), крутизной фронтов — сигнал SRC (Slew Rate Control). Пологий фронт желателен, поскольку замедление изменений токов и напряжений снижает помехи из-за токовых импульсов в цепях питания, перекрестные помехи и др. В то же время в критичных для быстродействия путях замедленные переключения элементов нежелательны, и поэтому в них устанавливают режимы крутых

фронтов. Буферные каскады с регулировкой крутизны фронтов достаточно часто применяют в современных схемах. Встречаются и более изощренные способы регулировки скорости изменения сигналов в буферных элементах по специально подобранным нелинейным законам.

Большие проблемы связаны с реализацией межсоединений в СБИС. Уменьшение размеров схемных элементов ведет к уменьшению поперечного сечения проводников, что увеличивает их погонное сопротивление. Резистивность и емкости связей ограничивают гипотезу их эквипотенциальности. Распространение потенциала вдоль проводника подчиняется уравнению диффузии, чему соответствует падение скорости распространения сигнала по мере удаления от источника и квадратичная зависимость задержки от длины проводника. Поэтому в длинных связях иногда включают через определенные расстояния усилители-повторители сигнала. Для БИС/СБИС, реализованных по технологии с минимальным размером 0,25 мкм, задержки в связях становятся в среднем равными задержкам в логических элементах, а при меньших минимальных размерах задержки в связях составляют основную часть общих задержек.

Линии связи с гальваническими развязками

Электрические токи могут протекать только в замкнутых цепях, для образования которых устройства, обменивающиеся данными, должны иметь общую схемную "землю" (общую точку). Объединение схемных земель взаимодействующих устройств — обычное решение, применяемое в подавляющем большинстве систем, но ему свойственны и известные недостатки. В электрически связанных цепях возникает множество помех — перекрестные помехи, помехи по цепям питания и др. Кроме того, нередко электронные приборы управляют работой мощных высоковольтных агрегатов и при аварийных пробоях изоляции высокие напряжения могут полностью выводить их из строя. В силу отмеченных обстоятельств, т. е. по соображениям электробезопасности и борьбы с помехами, в некоторых ситуациях применяют линии связи с гальваническими развязками. В этих случаях в тракте передачи имеется участок, на котором данные передаются с помощью оптических средств без электрической связи между источником и приемником. Основными элементами обеспечения электрических развязок служат *оптроны* (оптические изоляторы) — приборы с размещением светодиода и фоточувствительного элемента (диода или транзистора) в одном корпусе. Оптический изолятор преобразует входной сигнал в выходной при полном гальваническом разделении цепей входа и выхода. Связь выходных цепей со входными осуществляется следующим образом — под воздействием входного сигнала генерируется электромагнитное излучение (обычно в инфракрасной области спектра), которое затем воспринимается фоточувствительным элементом, преобразующим его в электрический сигнал. С помощью такого процесса можно передавать по линии не только цифровые, но и аналоговые сигналы.

На рис. 1.22 показана схема цифровой связи источника сигнала (например, процессора) с приемником (например, внешним устройством микропроцессорной системы).

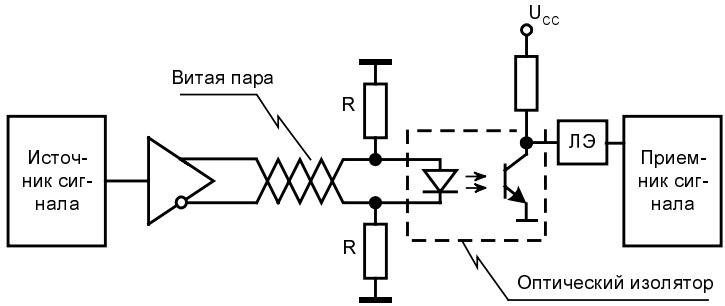


Рис. 1.22. Схема линии связи с гальванической развязкой

Сигнал на оптический изолятор передается от источника через буфер по витой паре с согласованием волновых сопротивлений резисторами-терминаторами R. Диодно-транзисторный оптический изолятор обеспечивает гальваническую развязку. Импульсы тока преобразуются светодиодом в импульсное излучение, воспринимаемое фоточувствительным транзистором, ток которого изменяется соответственно облучению базы. С коллектора транзистора снимается электрический сигнал и подается на приемник через логический элемент ЛЭ.

Линии типа "токовая петля"

Наряду с отображением логических сигналов уровнями напряжения при передаче последовательных данных применяют и отображение сигналов наличием или отсутствием тока в цепи. В частности, вариант "токовая петля" используется при передачах на большие расстояния (до десятков километров), хотя и с относительно невысокой скоростью. При этом двоичные данные отображаются уровнями токов в петле 0 и 20 мА (имеются варианты с токами 0 и 40 мА).

Стандарты сигналов ввода/вывода данных

С развитием схемотехники изменяются и параметры сигналов в трактах передачи. Для скоростных линий с высокими тактовыми частотами и малой потребляемой мощностью разработаны многие стандарты ввода/вывода, которые используются для межсоединений в процессорах, памяти, периферийных схемах, печатных платах, БИС/СБИС программируемой логики и т. д. Способность микросхем к поддержке одновременно многих промышленных стандартов ввода/вывода сокращает время разработки продукции на их основе.

Скоростная передача сигналов требует тщательного согласования волновых сопротивлений в линиях связи. Для эффективного подавления отражений применяются

схемы с комбинациями параллельных и последовательных согласований. На рис. 1.23 показаны варианты с двойным параллельным согласованием (*а*), последовательным согласованием в начале линии и параллельным в ее конце (*б*) и последовательно-параллельным согласованием в начале линии и параллельным в ее конце (*в*).

Схема с двумя видами согласований, КМОП-инвертором в качестве источника сигнала и дифференциальным аналоговым приемником показана на рис. 1.23, *а*. Для нее требуется питание от трех различных напряжений: V_{CCIO} — для выходного буфера (на схеме не показано), V_{TT} — для подключения резисторов-терминаторов и V_{REF} (опорного напряжения, подключаемого ко второму входу дифференциального приемника). На рис. 1.23, *б* изображена схема с двумя видами согласования — последовательным и параллельным, свойства которой рассмотрены ранее. Схема на рис. 1.23, *в* имеет три вида согласований — два в начале линии и одно в конце.

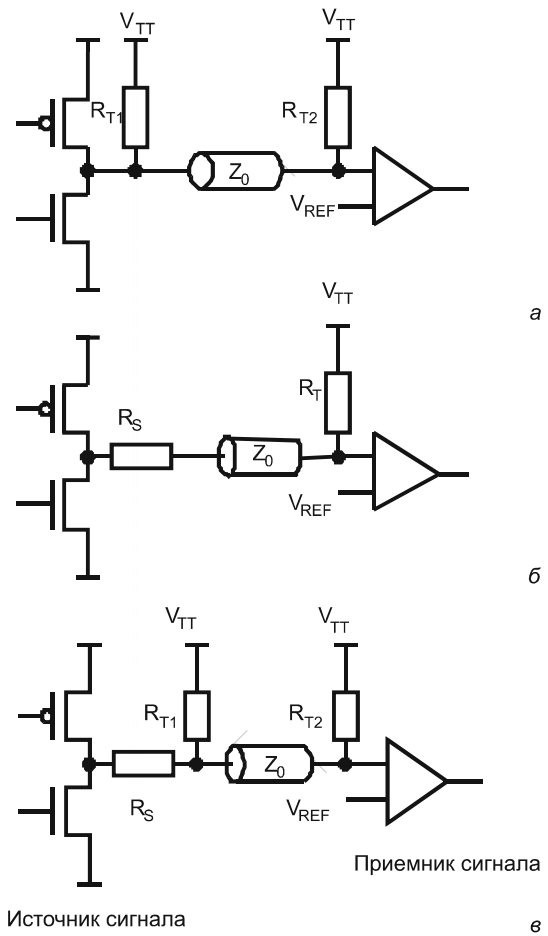


Рис. 1.23. Схемы с комбинированным подавлением волновых отражений

Стандарты ввода/вывода отличаются друг от друга как величинами используемых напряжений и уровнями нагрузочных токов, так и составом схемных элементов. Стандартизацией средств ввода/вывода занимается организация JEDEC (Joint Electron Device Engineering Council), создавшая стандарты JESD8. Стандарт JESD8-B определяет требования к интерфейсу ИС с напряжением питания 3,3 В, стандарт JESD8-5 определяет параметры интерфейса для ИС, работающих при напряжениях питания 2,5 В, а стандарт JESD8-7 рассчитан на схемы с питанием 1,8 В. К числу стандартов ввода/вывода принадлежат, в частности, следующие:

- LVTTL и LVCMOS с напряжениями питания 3,3; 2,5; 1,8 В — однополюсные линии для логических интерфейсов общего назначения. Не используют опорные напряжение и терминаторы. Имеют варианты с выходными токами в диапазоне 2…24 мА. Имеется вариант LVCMOS с напряжением питания 1,5 В. Работают на частотах до 225…350 МГц.
- 3,3V PCI / PCI-X — в стандарте PCI, основанном на схемах LVTL, при напряжении питания 3,3 В реализуются шины разрядностью 32 и 64 при частоте тактирования 33 или 66 МГц. PCI-X — усовершенствованная версия стандарта PCI с поддержкой более высоких частот передачи сигналов (66 и 133 МГц).
- LVDS — стандарт дифференциальных передач с высоким быстродействием и малым потреблением мощности. Имеются две версии промышленного стандарта (с полосой пропускания 250 Мбит/с и с полосой пропускания в диапазоне от 644 до 840 Мбит/с). Используется напряжение V_{CCIO} = 3,3 В и резистор-терминатор с сопротивлением 100 Ом. Опорное напряжение не используется.
- LVPECL — стандарт дифференциальной передачи, сходный со стандартом LVDS, но более высокочастотный и дорогостоящий.
- GTL+ — линия с двойным согласованием волновых сопротивлений, высокоскоростная стандартная шина, часто используемая для интерфейсов между процессорами и при коммуникациях на печатной плате. Требуется опорное напряжение 1,0 В и напряжение V_{TT} = 1,5 В. Сигнал на линию подается от каскада с открытым стоком. Минимальное значение питания цепей ввода V_{CCIO} должно быть не менее 3,0 В.
- SSTL-3, SSTL-2 и SSTL-18 классов 1 и 2 (Stub-Series Terminated Logic) — линии с согласованием волновых сопротивлений с напряжениями питания 3,3; 2,5 и 1,8 В, требующие применения опорных напряжений и напряжений V_{TT} (1,25 или 1,5 В). Используются для интерфейсов с микросхемами высокоскоростной синхронной динамической памяти SDRAM, в том числе с интерфейсами DDR1, DDR2.
- HSTL (High Speed Transceiver Logic) классов 1 и 2 — линии с согласованием волновых сопротивлений, использующие выходной буфер с напряжением питания V_{CCIO} = 1,5 В, опорное напряжение 0,75 В, напряжение V_{TT} = 0,75 В. Применяются для передач сигналов в цифровых ИС с частотами 150…250 МГц.

- AGP, соответствует спецификации Advanced Graphics Port Interface Specification, введен фирмой Intel для графических применений. Требует опорного напряжения 1,32 В и напряжения питания V_{CCIO} = 3,3 В. Не требует применения резисторов-терминаторов.

На рис. 1.24 приведены примеры схемных реализаций для некоторых стандартов сигналов ввода/вывода.

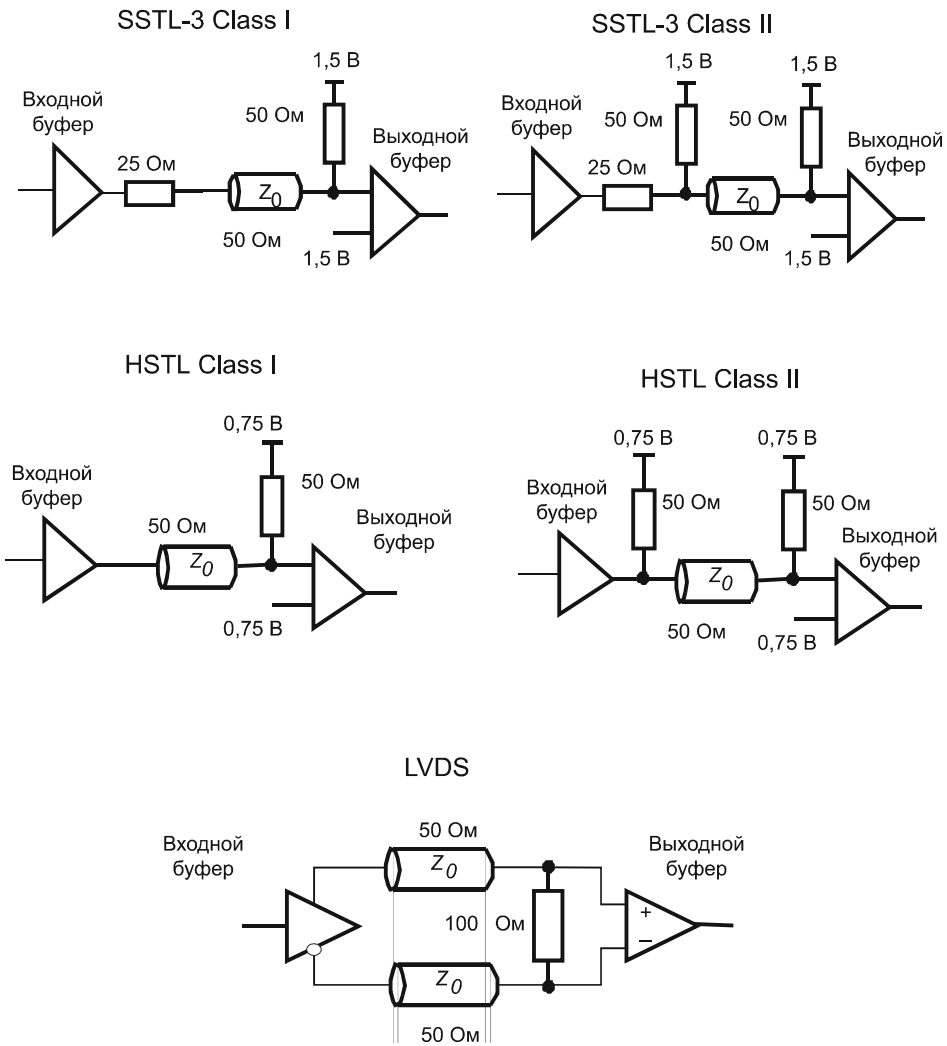
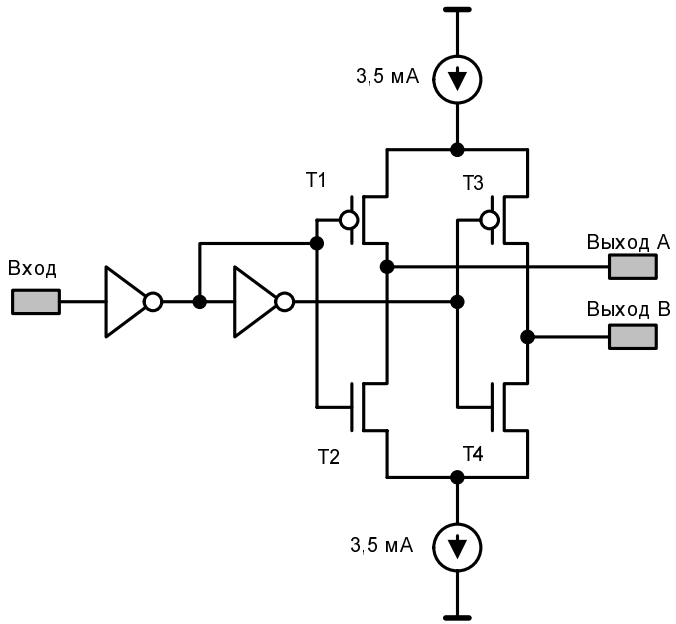
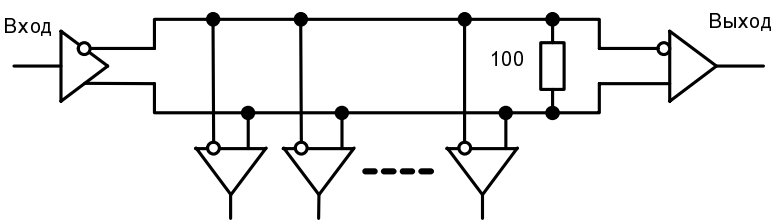


Рис. 1.24. Примеры схем, реализующих стандарты сигналов ввода/вывода

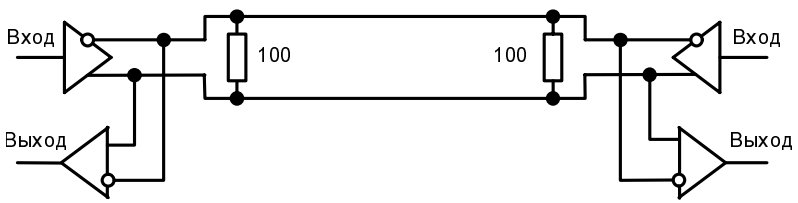
На рис. 1.25 показана реализация стандарта LVDS для симплексной связи "один источник — один приемник".



а



б



в

Рис. 1.25. Схемная реализация стандарта LVDS

Входной сигнал с помощью двух инверторов преобразуется в парафазный и подается на управление каскадом, образованным транзисторами T1…T4. Каскад питается от источников, выполненных в виде генераторов тока, так как токи можно пе-

реключать быстрее, чем напряжения. В зависимости от значения входного напряжения открывается одна из пар диагонально расположенных транзисторов (T1—T4 или T2—T3) и в контуре через резистор с сопротивлением 100 Ом ток течет по часовой стрелке или против нее, создавая на входе дифференциального усилителя приемника напряжение того или иного знака.

Терминирование на кристалле

Проблема терминирования линий связи, т. е. введения для них согласующих сопротивлений, создала большие затруднения для схем с БИС/СБИС. Большое число выводов у корпусов современных БИС/СБИС (особенно у корпусов типа BGA) не приспособлено к подключению внешних резисторов-терминаторов. Потребовался перенос резисторов-терминаторов на кристалл. Терминирование может быть фиксированным (фирмы Altera, TI и др.) либо программируемым, т. е. с *цифровым управлением выходными сопротивлениями источников сигнала* (технология DCI, Digitally Controlled Impedance).

При последовательном согласовании в сигнальный провод на входе линии обычно включают сопротивление R, чтобы сделать суммарное сопротивление цепи (выход буфера плюс R) равным волновому сопротивлению линии (для получения 50 Ом обычно требуется добавочное сопротивление около 30 Ом). Функция DCI позволяет изменять выходное сопротивление буфера, причем в таких пределах, которые нужны для согласования сопротивлений. При этом могут быть *исключены внешние резисторы*, подключаемые к ножкам корпуса ИС, и *схемы терминирования переносятся на кристалл,* что при современных плотностях монтажа и большом числе выводов корпуса весьма важно.

Кроме последовательного согласования, о котором говорилось ранее, применяется и параллельное. Функция DCI обеспечивает и этот вид согласования. *Резисторы-терминаторы для параллельного согласования реализуются на кристалле и автоматически подбираются соответственно требованиям линии.* При выборе для блоков ввода/вывода того или иного стандарта активизируются и средства DCI.

Банки ввода/вывода

Для удобства поддержки многочисленных стандартов блоки ввода/вывода разбиваются на группы, называемые *банками*, в которых объединяются схемы обеспечения совместимых стандартов. Совместимость определяется уровнями напряжений, используемых схемами стандартов. На рис. 1.26 приведена иллюстрация такого разбиения. С каждым банком ассоциируется группа контактов V_{CCO}, но все они должны подключаться к одному и тому же напряжению. В разных банках можно использовать различные напряжения V_{CCO}. Для банка все контакты подачи напряжения V_{REF} соединены друг с другом, так что совместимыми стандартами могут быть такие, у которых совпадают напряжения V_{CCO} и V_{REF} (или, может быть, какое-либо из этих напряжений в стандарте не используется).

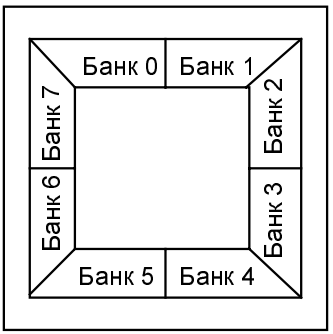


Рис. 1.26. Разбиение блоков ввода/вывода на банки

Передача данных с двойной скоростью (технология DDR)

Частота передач сигналов на печатной плате и кристалле ограничена возможностями линий связи и не может превышать определенных значений. В то же время устройства обработки данных, возможно, способны работать на более высоких частотах. Пропускную способность линий передач удалось повысить в два раза в рамках технологии DDR (Double Data Rate).

В обычном варианте передач (SDR, Single Data Rate) данные принимаются под воздействием только одного фронта тактирующего сигнала (положительного или отрицательного). Интервал времени между приемами данных равен тактовому периоду. Технология DDR состоит в приеме данных под воздействием фронтов обоих знаков. Такая возможность обеспечивается специальными схемотехническими мерами, а интервал между моментами приема соседних данных становится равным полупериоду тактирующих импульсов.

Технология DDR позволяет удвоить частоту передач сигналов при данной частоте тактирования или же снизить вдвое эту частоту при сохранении частоты передачи данных. Схемы сопряжения устройств, работающих по технологии DDR, с устройствами, работающими по технологии SDR, достаточно просты, хотя и требуют более жесткого соблюдения допусков на параметры и тщательного проектирования с точки зрения борьбы с помехами. Реализация передач с двойной скоростью показана на рис. 1.27 (возможна и организация выходов с третьим состоянием, которая здесь не рассматривается).

Для ввода/вывода по технологии DDR (или более поздним технологиям DDR-2, DDR-3, работающим на более высоких частотах) в схеме имеются специальные регистры. Входы и выходы выполняются с помощью пар регистров. При вводе один из регистров пары тактируется положительным фронтом синхросигнала, а второй — отрицательным, поскольку на него поступает инвертированный синхросигнал (противофазные синхросигналы можно получать и от блоков PLL или DLL,

рассматриваемых далее). Биты принимаемого слова идут поочередно на разные регистры. В одном из регистров фиксируются четные биты, в другом — нечетные. Слово принимается вдвое быстрее, чем при обычном методе SDR.

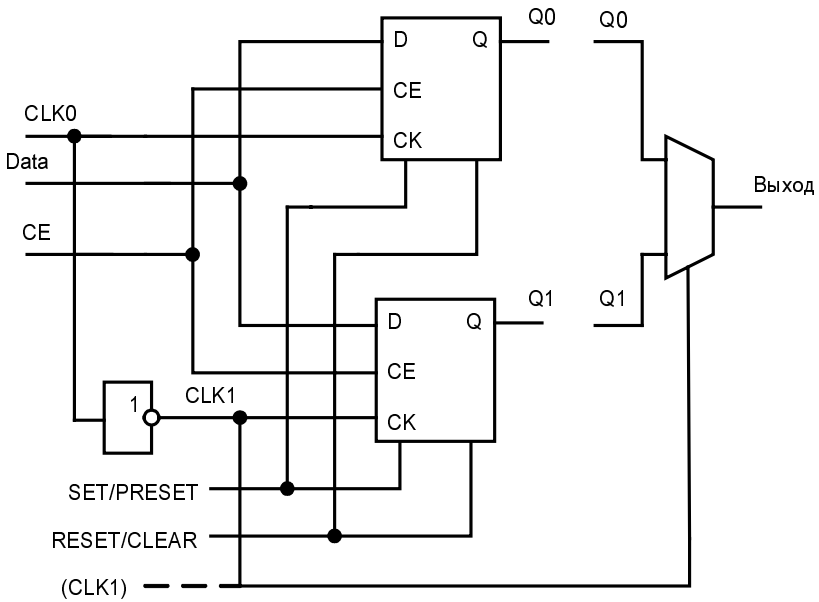


Рис. 1.27. Схемная реализация передач сигналов по технологии DDR при вводе и выводе данных

Вывод данных с удвоенной скоростью организуется с помощью мультиплексора, поочередно передающего на выход биты слова из пары регистров. Мультиплексор получает в качестве адресующего входа синхросигнал. При одном из уровней этого сигнала на выход поступает бит с одного из регистров (например, четный), при другом — с другого (нечетный). В результате слово выводится с удвоенной скоростью относительно обычного метода SDR.

Быстродействующие низковольтные дифференциальные линии LVDS с двойной скоростью передачи данных обладают скоростями до нескольких гигабит в секунду при экономичной схемной реализации и небольшой потребляемой мощности. Конечно, все большее распространение быстродействующих передач последовательных данных не означает отказ от многоразрядных шинных структур. В зависимости от конкретных условий преимущество имеет то или иное решение.

О разрядностях высокоскоростных шин

Пропускная способность шины зависит от частоты и разрядности передаваемых по ней слов и определяется их произведением. В прошлом пропускную способность шин традиционно повышали, увеличивая их разрядность. Но с ростом частот уве-

личиваются скорости изменения сигналов и усложняется ситуация с помехами, что особенно сказывается на многоразрядных шинах. В многоразрядных шинах труднее решать задачи борьбы с перекрестными помехами, помехами по цепям питания и помехами из-за отраженных волн в несогласованных линиях. Поэтому для многоразрядных шин частота тактирования ниже, чем для малоразрядных. Вследствие указанных обстоятельств в современных условиях изменился и подход к выбору рациональной разрядности для шин с высокой пропускной способностью.

В последнее время прослеживается *тенденция к уменьшению разрядности шин при одновременном повышении частоты их работы*. Замена многоразрядных шин на малоразрядные может дать существенное *упрощение и удешевление* систем при сохранении или даже повышении производительности шины. Упрощение и удешевление систем могут быть весьма большими, так как сокращение числа контактов микросхем при переходе к малоразрядным шинам может оказаться многократным. Современные проектные нормы позволяют получать логические элементы очень высокого быстродействия, что также является условием успешного перехода к малоразрядным шинам. Действительно, уменьшение разрядности передаваемых слов сопряжено с необходимостью последующего ее увеличения соответственно требованиям устройств обработки данных. Для этого нужны блоки преобразования последовательных данных в параллельные и наоборот (*блоки SERDES*, Serialiser-Deserialiser). Благодаря быстродействию современных схем блоки SERDES при использовании дифференциальных линий связи уже могут работать на частотах в несколько гигагерц. В настоящее время еще не все проблемы создания высокоскоростных блоков SERDES полностью решены, однако скоростные последовательные связи становятся обычными для многих цифровых БИС/СБИС.

> Оценить получаемые при переходе на малоразрядные шины результаты можно на следующем примере. Пусть 64-разрядная шина заменяется на 8-разрядную с дифференциальными связями (дифференциальные линии требуют двух контактов для каждой линии). При указанной замене вместо 64 контактов для шины потребуется 16, и на каждом канале передачи, которых может быть несколько, будет сэкономлено 48 контактов, что существенно упрощает не только микросхему, но и монтаж на печатной плате. Частота формирования байтов, т. е. время накопления байтов, на которые была разделена 64-разрядная шина, будет приблизительно в восемь раз ниже, чем частота тактирования последовательных дифференциальных связей. Если, например, последовательные связи работают на частоте 1,6 ГГц, то в результате будет получен эквивалент 64-разрядной шины, работающей на частоте 200 МГц.

Работа на высоких частотах предъявляет жесткие требования к синхронизации процессов в блоках ЦУ. Проблемы синхронизации приобрели важнейшее значение в цифровой технике высокой производительности. Для поддержания синфазности синхронизирующих и информационных сигналов разработаны специальные средства (блоки PLL и DLL) и технологии подстройки друг к другу сдвинутых во времени синхросигналов и передаваемых данных (CDR и т. д.), рассмотренные в *главе 3*.

§ 1.7. Элементы задержки, формирования, обнаружения и генерации импульсов

Элементы задержки

Задержки цифровых сигналов требуются прежде всего для согласования времен распространения сигналов по различным путям с целью борьбы с критическими состязаниями, нарушающими работоспособность ЦУ.

Техническая реализация элементов задержки зависит от требуемых значений их параметров (величины, стабильности, регулируемости задержки и т. д.). Применяют различные варианты реализации задержек: отрезки обычных или специальных коаксиальных кабелей, цепочки логических элементов, искусственные электромагнитные линии задержки, RC-цепочки, одновибраторы, схемы деления частоты тактовых сигналов. Остановимся на самых типичных для ЦУ вариантах — цепочках логических элементов и RC-цепочках.

В первом случае используется естественная инерционность логических элементов. В последовательной цепочке из нескольких элементов их задержки суммируются. Для целей задержки естественно применять простейшие элементы — инверторы или повторители. Задержку можно регулировать дискретно, изменяя число элементов в цепочке. Если цепочка составлена из инверторов, то при четном их числе получается просто задержка сигнала, при нечетном — задержка с инверсией. Порядок величин получаемых задержек обычно соответствует необходимому, т. к. требуется компенсация разновременности распространения сигналов в цепях, также составленных из логических элементов. Точность задержки ограничивается разбросом собственных задержек элементов и невысока.

При получении переменной задержки с помощью цепочки повторителей (рис. 1.28, *а*), каждый из которых имеет задержку t_d (индекс d от delay, что означает задержку), в зависимости от числа повторителей можно получать задержки от t_d до nt_d. Выбор величины задержки осуществляется мультиплексором (однополюсным ключом на n положений).

Задержку можно получить с помощью RC-цепочки в цепи передачи сигнала (рис. 1.28, *б*, *в*), где она формирует экспоненциальные процессы перезаряда емкости через резистор R с постоянной времени RC. Если считать пороговым напряжением середину логического перепада, то время задержки $t_d = RC \cdot \ln 2 = 0{,}7RC$. После RC-цепочки включены три инвертора для формирования крутых фронтов. В схемах ЦУ задержки на RC-цепочках могут составлять величины порядка микросекунд.

Имеется существенная разница в условиях применения RC-цепочек в схемах на МОП- и биполярных транзисторах. В первом случае входные токи элементов пренебрежимо малы и включение на входе элемента даже большого сопротивления вполне допустимо. Во втором случае входные токи элементов значительны, поэтому в их входные цепи можно включать лишь малые сопротивления (иначе произойдут недопустимые изменения уровней напряжения U_0 и U_1 из-за падений на-

пряжения на резисторе R), а постоянную времени придется увеличивать за счет больших емкостей C, что не всегда удобно по конструктивным соображениям.

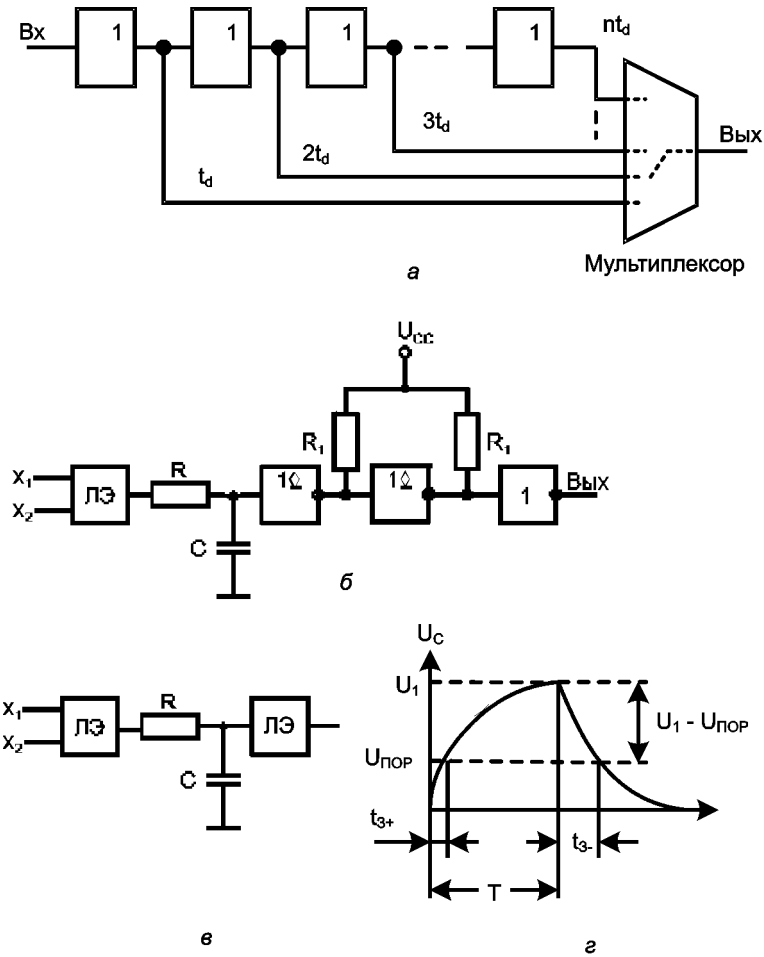


Рис. 1.28. Схема реализации переменной задержки на основе цепочки повторителей (*а*), схема задержки с RC-цепочкой и формирующими элементами, предотвращающими протекание сквозных токов (*б*), схема задержки при достаточно быстрых изменениях напряжения на емкости (*в*), к расчету времен задержек положительного и отрицательного фронтов входного напряжения (*г*)

Точность задержки ограничивается допусками на параметры R и C и разбросом порогового напряжения логических элементов или буферных каскадов, применяемых в схеме. С увеличением постоянной времени RC график изменения напряжения на емкости становится все более пологим. При этом разброс пороговых напряжений элементов будет вызывать все больший разброс задержек. Таким образом, чем больше задержка, тем менее точной она становится. Кроме того, для некоторых элементов (типа КМОП) слишком длительные фронты входных сигналов недопустимы по пас-

портным данным. Нежелательны затянутые фронты и для элементов ТТЛ(Ш) с их сквозными токами. Поэтому в схеме (рис. 1.28, *б*) первые элементы цепи формирования имеют выходы с открытым электродом, в которых не возникают сквозные токи.

Перед повторным срабатыванием схема должна восстановиться, для чего минимальная длительность постоянного уровня входного напряжения должна быть около 3RC. Порогом логического элемента приближенно считают середину логического перепада. Однако фактически порог отличается от этого значения (обычно он несколько меньше). Поэтому задержки положительного и отрицательного фронтов отличаются друг от друга. На рис. 1.28, *г* приведены временные диаграммы напряжения на емкости для расчета задержек фронтов ($t_{З+}$ и $t_{З-}$), на основании которых можно получить соотношения:

$$t_{З+} = \tau \ln[U_1/(U_1 - U_{пор})],$$

$$t_{З-} = \tau \ln(U_1/U_{пор}).$$

Например, если $U_{пор} = 0{,}4U_1$, то $t_{З+} = 0{,}5\tau$, а $t_{З-} = 0{,}9\tau$.

При близко расположенных фронтах задержки ограничиваются тем, что сближение фронтов сопровождается уменьшением амплитуды сигналов на емкости, т. к. напряжения на ней не успевают восстановиться к моменту появления следующего фронта. Если допустить уменьшение сигнала на входе элемента, подключенного к емкости, не более чем на 10%, то одним каскадом можно задерживать фронты на время не более 12% периода их повторения T.

RC-цепочки используются и во времязадающих цепях одновибраторов (ОВ), которые также применяются для задержек цифровых сигналов (фронтов). Одновибраторы имеют одно устойчивое состояние. Входной сигнал переводит ОВ в квазиустойчивое состояние, в котором он находится в течение времени, определяемого параметрами его времязадающей цепи (RC-цепочкой). Затем ОВ возвращается в свое устойчивое состояние. При этом формируется фронт, который служит выходным сигналом. Значит, длительность квазиустойчивого состояния ОВ, т. е. длительность формируемого им одиночного импульса, и есть время задержки сигнала. Одновибратор является релаксационной схемой, способной формировать крутые фронты выходного импульса.

Задержку сигнала в цифровых устройствах при наличии в них синхросигналов можно получить с помощью счетчиков. При этом входной сигнал должен разрешать работу счетчика, находящегося в нулевом исходном состоянии. Счетчик начнет подсчитывать синхросигналы, а при его переполнении выработается выходной сигнал. Таким образом, осуществится задержка $t_d = NT$, где N — емкость счетчика, T — период синхроимпульсов.

С помощью элементов задержки и простых логических схем решаются задачи формирования импульсов по длительности и генерации импульсных последовательностей.

Формирование импульсов по длительности

К задачам формирования импульсов по длительности относятся расширение, сужение и стандартизация. Эти операции реализуются схемой (рис. 1.29, *а*). Если функция F является дизъюнкцией, то, как видно из временных диаграмм на рис. 1.29, *б*, схема будет расширять входной импульс на интервал, равный времени задержки t_d. Если функция F является конъюнкцией, то схема дает сужение входного импульса на величину t_d (рис. 1.29, *в*). Если $F = x_1 \cdot \overline{x}_2$, то при $t_{вх} > t_d$ будет выполнена стандартизация длительности импульса. Выходной импульс будет иметь длительность t_d.

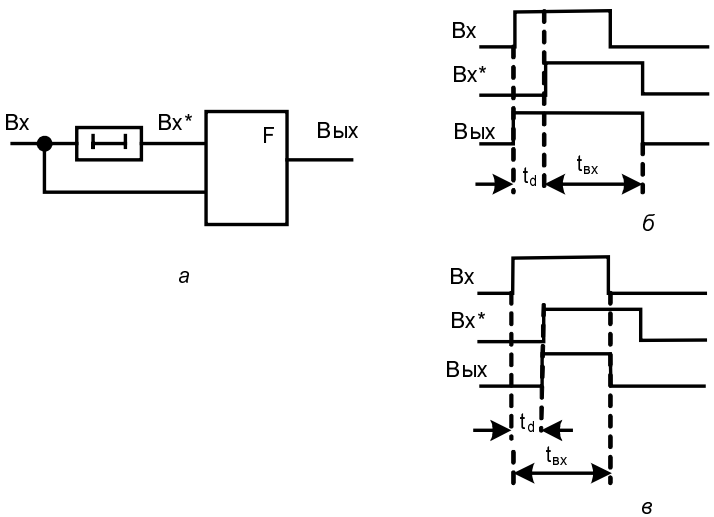


Рис. 1.29. Схема формирования импульса по длительности (*а*) и временные диаграммы реализации операций расширения (*б*) и сужения (*в*) импульсов

Разностные преобразователи и детекторы событий

Разностными преобразователями называют схемы, формирующие стандартные импульсы в ответ на перепад (изменение) входного напряжения.

Эта операция есть не что иное, как фиксация факта появления перепада напряжения на входе схемы. Подробнее этот режим представлен на рис. 1.30. Принцип работы схем обнаружения положительных и отрицательных фронтов выходного сигнала иллюстрируется приведенными на рисунке временными диаграммами.

В схему *детектора* событий (рис. 1.31) входят два разностных преобразователя, один из которых обнаруживает положительные фронты входного сигнала, отмечая их появление отрицательными импульсами длительностью в три задержки инвер-

тора. Выходные импульсы запаздывают относительно момента появления фронта на одну элементарную задержку (задержку вентиля И-НЕ). Второй преобразователь обнаруживает отрицательные фронты входного сигнала, отмечая их импульсами с теми же параметрами, что и первый преобразователь. Выходы обоих преобразователей поступают на элемент И-НЕ, формирующий положительные импульсы при появлении любого изменения входного сигнала (из нуля в единицу или наоборот), что и требуется от детектора событий.

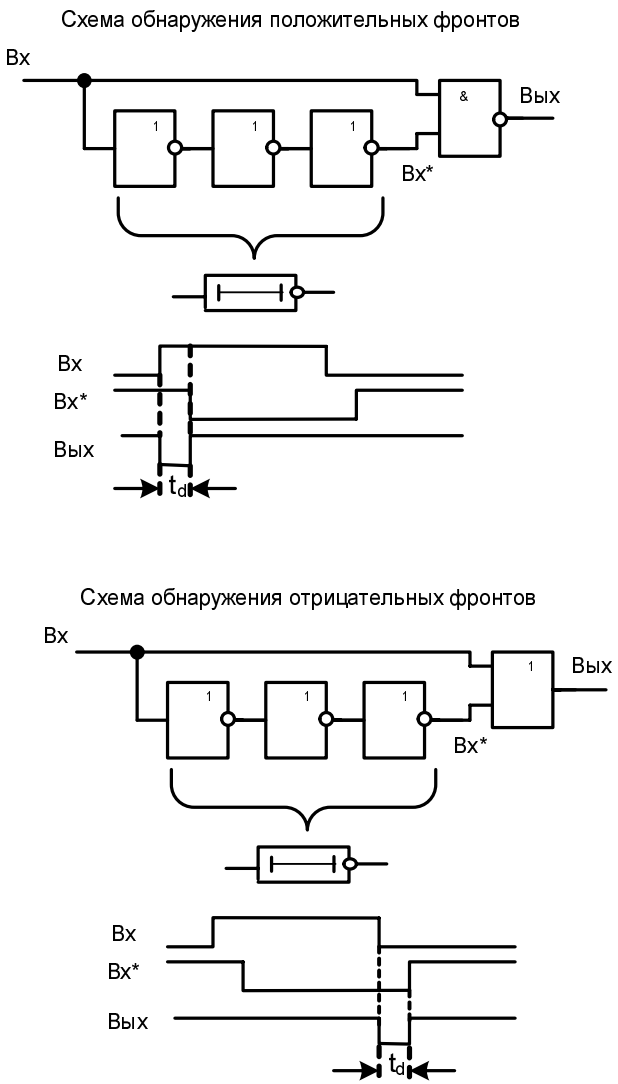


Рис. 1.30. Разностные преобразователи

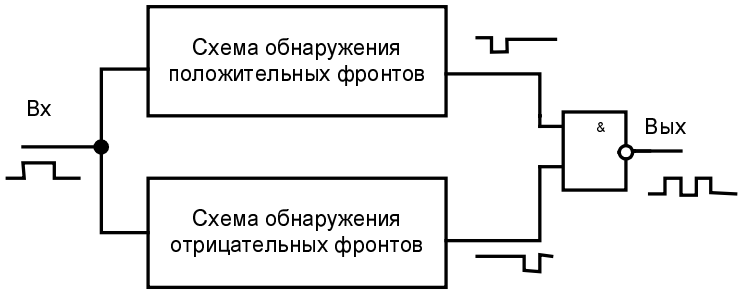


Рис. 1.31. Схема детектора событий

Заметим, что детектор событий *может выполнять умножение частоты на два*, поскольку в периоде входного сигнала содержатся два фронта, а на каждый фронт детектор событий откликается выдачей импульса, т. е. двумя фронтами. Это свойство детектора событий весьма полезно, т. к. умножение синхрочастот используется в работе ряда устройств, рассматриваемых в последующих разделах. Каскадируя детекторы событий, можно получать умножители частоты на 2^n, где n — число детекторов в последовательной цепи.

Кольцевые генераторы

На элементах задержки и логических элементах строятся кольцевые генераторы импульсных последовательностей. Простейший вариант показан на рис. 1.32, *а*. При нулевом значении сигнала управления Упр на выходе элемента И-НЕ имеется логическая единица, которая через обратную связь с задержкой на t_d передается на верхний вход элемента. Таким образом, в исходном состоянии верхний вход элемента И-НЕ находится в состоянии логической единицы. Изменение управляющего сигнала является командой для начала работы генератора. Появление единицы на нижнем входе Упр элемента И-НЕ дает совпадение единиц на обоих входах, что переводит выход схемы в нулевое состояние. Это состояние длится в течение интервала t_d, т. к. после него нуль с выхода схемы по обратной связи пройдет на верхний вход элемента и поставит его в единичное состояние, которое также сохранится на время t_d, после чего изменится из-за воздействия по цепи обратной связи. Следовательно, схема будет генерировать симметричные импульсы с периодом повторения $2t_d$ и длительностями импульса и паузы, равными t_d (рис. 1,32, *б*).

Частота генерации для кольцевых схем зависит от задержек элементов, и может быть весьма высокой при использовании элементов с малыми задержками. Из-за большого разброса задержек расчетное значение частоты дает лишь приблизительные результаты. Стабильность частоты соответствует стабильности задержек элементов.

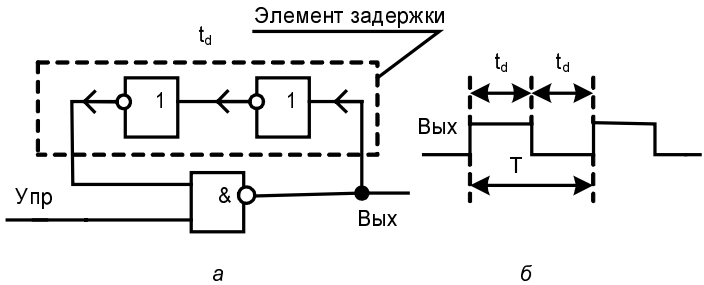


Рис. 1.32. Схема генератора симметричных импульсов (*а*) и временная диаграмма его выходных сигналов (*б*)

§ 1.8. Элементы визуальной индикации

Элементы индикации на светодиодах

Для общения с оператором ЦУ снабжаются средствами визуальной индикации символьных данных. Среди них имеются как сложные устройства, такие как экранные дисплеи, так и простые, такие как светодиодные индикаторы.

Преобразование электрических сигналов в видимое изображение может быть основано на разных физических явлениях: светоизлучении полупроводниковых структур, оптических явлениях в жидких кристаллах, электролюминесценции, процессах в газовом разряде и др.

Индикаторы на светодиодах. Светодиоды изготовляются на основе специальных полупроводниковых материалов (арсенида галлия, фосфида галлия, арсенид-фосфида галлия и др.). Пропускание тока через диод вызывает его свечение. Цвет свечения зависит от материала, из которого изготовлен диод. Яркость свечения зависит от величины тока. Обычно достаточен ток от единиц до приблизительно двадцати миллиампер при падении напряжения на диоде около 1...2 В. Как правило, последовательно со светодиодом включается резистор, задающий и стабилизирующий ток диода.

Из нескольких диодов составляются индикаторы и матрицы, отображающие буквы и цифры. Широко применяются *семисегментные индикаторы*, в которых семь сегментов-диодов расположены так, что при зажигании определенной их комбинации высвечивается тот или иной символ (рис. 1.33, *а*).

Выпускаются семисегментные индикаторы (ССИ) с общим анодом или общим катодом (рис. 1.33, *б*, *в*). Для управления сегментами удобны элементы с открытым выходом (типа ОК или ОС), поскольку в этом случае имеется внешняя цепочка с резистором, сопротивление которого можно задать для получения нужного тока светодиодов.

В схеме (рис. 1.34, *а*) показано управление сегментом ССИ. Диод зажигается, когда на выходе управляющего элемента напряжение равно U_0. Через диод будет проте-

кать ток $I_д = (U_{cc} – U_д – U_0)/R$, следовательно, для его задания требуется условие $R = (U_{cc} – U_д – U_0)/I_д$. Для этой схемы требуются ССИ с общим анодом. Необходим управляющий элемент с достаточно большим выходным током в нулевом состоянии ($I_{вых.0} \geq I_д$).

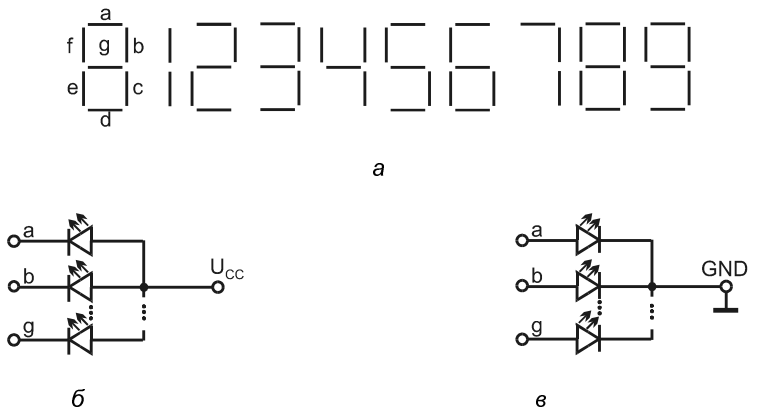


Рис. 1.33. Семисегментный индикатор и отображаемые им цифры (*а*), варианты индикатора с общим анодом (*б*) и общим катодом (*в*)

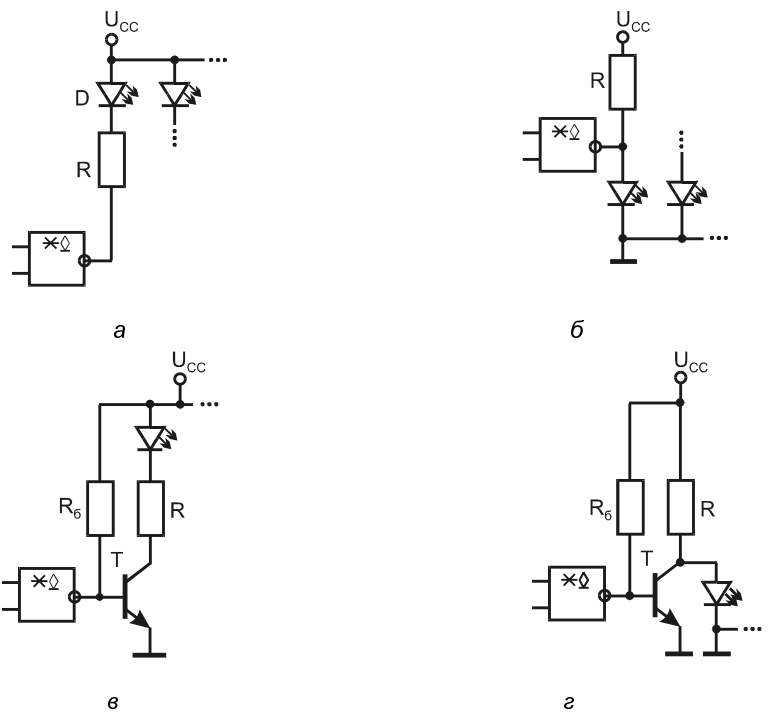


Рис. 1.34. Схемы управления сегментом индикатора с общим анодом (*а*), общим катодом (*б*) и использованием усилительных каскадов (*в*, *г*)

В схеме (рис. 1.34, *б*) диод зажигается, когда выходной транзистор управляющего элемента запирается. Через диод течет ток $I_д = (Ucc - U_д)/R$, откуда следует $R = (U_{cc} - U_д)/I_д$. Для этой схемы требуется ССИ с общим катодом. Выход управляющего элемента должен удовлетворять условию $I_{вых.0} \geq (U_{cc} - U_0)/R$.

Если выходные токи управляющих элементов недостаточны для управления диодом, между выходом элемента и сегментом индикатора можно включить буферный каскад на транзисторе. Примеры приведены на рис. 1.34, *в*, *г*.

Для логического управления семисегментными индикаторами имеются стандартные дешифраторы ССИ, работающие согласно табл. 1.2. Таблица приведена для такой схемы управления светодиодами, в которой зажиганию сегмента соответствует единичное значение логического управляющего сигнала. Для схем управления, в которых светодиоды зажигаются нулевыми сигналами на выходе логического элемента, таблица видоизменяется — все значения функций *a* ... *g* должны быть проинвертированы.

Таблица 1.2

Десятичная цифра	Входной двоичный код	Возбуждаемые сегменты						
		a	**b**	**c**	**d**	**e**	**f**	**g**
	0000	1	1	1	1	1	1	0
1	0001	0	1	1	0	0	0	0
2	0010	1	1	0	1	1	0	1
3	0011	1	1	1	1	0	0	1
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
8	1000	1	1	1	1	1	1	1
9	1001	1	1	1	1	0	1	1

Индикаторы на жидких кристаллах

Второй тип индикаторов, имеющих обычные для ИС уровни управляющих сигналов, — *жидкокристаллические индикаторы* (ЖКИ). Ранее они применялись преимущественно в электронных часах, калькуляторах и измерительных приборах. С появлением портативных компьютеров с автономным питанием энергетическая экономичность ЖКИ стала особенно важной, и на них стали делать дисплеи — сложные устройства отображения информации для компьютеров и других систем.

Жидкий кристалл при низких температурах представляет собою обычный твердый кристалл, при высоких температурах переходит в жидкое состояние, а в среднем диапазоне температур является вязким веществом с анизотропными свойствами. Молекулы этого вещества под действием электрического поля приобретают упорядоченную ориентацию в пространстве, что изменяет оптические свойства вещества.

Индикатор ЖКИ — две параллельно расположенные стеклянные пластины, на внутренних сторонах которых наклеены прозрачные электроды нужного рисунка и один общий электрод. Между пластинами размещается жидкий кристалл. Передний электрод всегда прозрачен, задний либо прозрачен (работа на просвет), либо является отражающим (работа на отражение). Если между электродами создается электрическое поле, то жидкокристаллическое вещество между ними изменяет прозрачность и находящийся под напряжением электрод становится видимым. ЖКИ не излучает свет, он лишь модулирует подаваемый на него световой поток. В конструкциях с отсутствием внешнего светового потока (например, в электронных часах) создаваемое в ЖКИ изображение не видно в темноте. ЖКИ обладает большим достоинством — он потребляет очень малую мощность. Для управления этим индикатором достаточны напряжения в единицы вольт при токах порядка 10 мкА, что на несколько порядков меньше, чем токи, требуемые для управления светодиодными индикаторами. Отсюда видна и высокая ценность ЖКИ для устройств с автономными источниками питания, в частности для портативной аппаратуры с батарейным питанием. ЖКИ используются в семисегментных и других индикаторах, подобных тем, которые рассмотрены ранее применительно к индикаторам на светодиодах, в дисплеях, экранах и т. д.

Напряжение, подаваемое на электроды ЖКИ, не должно содержать постоянной составляющей, так как это приводит к быстрому выходу ЖКИ из строя. В цифровой аппаратуре для возбуждения сегментов индикатора на них подается переменное напряжение в виде прямоугольных импульсов с частотой повторения в десятки герц. При такой частоте вследствие инерционности зрения мелькание сегментов уже не ощущается. Применение более высоких частот ведет к риску появления в напряжениях, питающих сегменты, постоянной составляющей. Действительно, для высокочастотных импульсов доля времени, приходящаяся на фронты, растет, а так как фронты импульсов фактически не идентичны (импульсы неидеальны по форме), и площадь положительной полуволны напряжения несколько отличается от площади отрицательной полуволны, возникающая постоянная составляющая пропорциональна частоте повторения импульсов.

Для получения переменного напряжения возбуждения сегментов ЖКИ используется фазовый метод управления. При этом на электроды передней и задней пластин ЖКИ подаются прямоугольные импульсы. Если фазы этих импульсов одинаковы, то между электродами напряжение будет отсутствовать, и сегмент не будет возбужден. Если же импульсы противофазны, то между электродами появляется переменное напряжение удвоенной амплитуды, и сегмент возбуждается. Схема управления для ЖКИ (рис. 1.35) имеет на одном из входов опорное напряжение $U_{оп}$, а на другом управляющее напряжение $U_{упр}$.

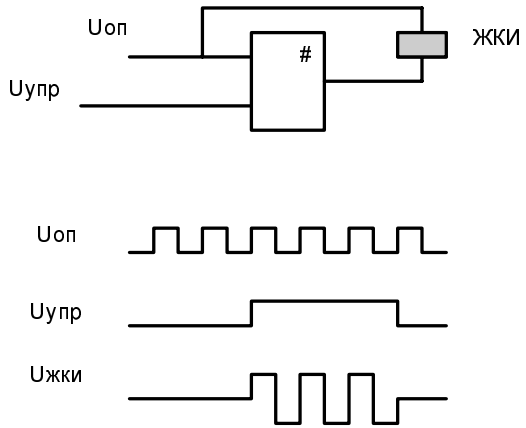


Рис. 1.35. Схема управления жидкокристаллическим индикатором

Опорное напряжение поступает на верхний вход элемента сложения по модулю 2, а управляющее — на нижний. Если управляющее напряжение равно нулю, то выходное напряжение элемента совпадает со входным ($X \oplus 0 = 0$), а при управляющем напряжении, соответствующем логической единице, выходное напряжение элемента инвертируется ($X \oplus 1 = \overline{X}$). На рис. 1.35 показаны также временные диаграммы действующих в схеме напряжений.

На основе светодиодов или жидкокристаллических индикаторов изготовляются как семисегментные изображения символов, так и более сложные, отображаемые возбуждением определенных сегментов из поля матрицы. Число строк и столбцов матрицы может быть различным. Для примера на рис. 1.36 показано поле размерностью 7×5, причем матрица неполная, из нее исключены 8 сегментов (дважды по 4), поскольку они не используются при отображении символов. Принципы формирования изображения при управлении сегментами матрицы те же, что и при управлении ССИ, а именно: входные коды специальным дешифратором преобразуются в сигналы возбуждения отдельных сегментов.

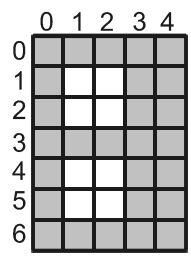


Рис. 1.36. Неполная матрица индикатора 7×5

При реализации индикаторов, содержащих несколько ССИ, удобно использовать *мультиплексное управление*, при котором одни и те же управляющие схемы поочередно обслуживают различные ССИ, выбирая их в определенной последовательно-

сти. При этом каждый индикатор возбуждается импульсно, в течение времени 1/n, где n — число индикаторов. Иллюзия постоянного свечения всех символов создается из-за инерционности человеческого зрения. Если частота возбуждения символов составляет десятки герц (современные средства визуальной индикации имеют частоты в 70...100 Гц), то мерцания изображений неощутимы.

§ 1.9. О некоторых типовых ситуациях

Разработанная проектировщиком функционально-логическая схема подлежит реализации на стандартных ИС или на наборе библиотечных микросхем. В обоих случаях возможны несовпадения имеющихся элементов с элементами реализуемой схемы. Типовыми ситуациями здесь являются наличие у элементов "лишних" (неиспользуемых в данном случае) входов, наличие в корпусах ИС лишних элементов или, напротив, нехватка у имеющихся элементов необходимого числа входов или нагрузочной способности.

Режимы неиспользуемых входов

Вопрос о режиме "лишних" входов решается с учетом используемой схемотехнологии. Пусть, например, нужно получить конъюнкцию пяти переменных. В стандартных сериях нет конъюнкторов с пятью входами, и придется взять элемент с восемью входами, у которого окажется три "лишних" входа. Принципиально возможно поступить следующим образом: не обращать внимания на "лишние" входы (т. е. оставить их разомкнутыми), подсоединить их к задействованным входам или подать на них некоторые константы. С точки зрения логических операций все три возможности правомерны (рис. 1.37, *а*). Если же учесть особенности той или иной схемотехнологии, то выбор варианта действий становится более определенным.

Для ЭСЛ неиспользуемые входы оставляют разомкнутыми. Это объясняется тем, что в схемах самих элементов уже предусмотрены специальные резисторы, связанные с источником питания, которые обеспечивают необходимые режимы "лишним" входам.

Для КМОП и ТТЛ(Ш) неиспользуемые входы разомкнутыми не оставляют. Для КМОП это строгая рекомендация, т. к. у них на разомкнутые входы легко наводятся паразитные потенциалы, которые могут изменять работу схемы. Для ТТЛ(Ш) строгого запрета на разомкнутые входы нет, но они ухудшают быстродействия элемента. Подсоединение "лишних" входов к задействованным для КМОП и ТТЛ(Ш) принципиально возможно, но приводит к увеличению нагрузки на источник сигнала, что также сопровождается уменьшением быстродействия схемы.

Таким образом, для КМОП и ТТЛ(Ш) рекомендуемый режим неиспользуемых входов — *подсоединение их к константам (логическим единицам или нулям), не изменяющим работу схемы для задействованных входов.* При этом уровни напряжений U_1 и U_0 для КМОП близки к уровням U_{CC} и "земли", к которым и под-

ключают неиспользуемые входы. У элементов ТТЛ(Ш) уровень U_1 заметно ниже уровня U_{CC}, поэтому следует поинтересоваться справочными данными и посмотреть, разрешается ли непосредственное подключение неиспользуемых входов к напряжению питания. Для некоторых серий элементов рекомендуется подключение через резистор, сопротивление которого указывается. Через один резистор могут подключаться несколько входов.

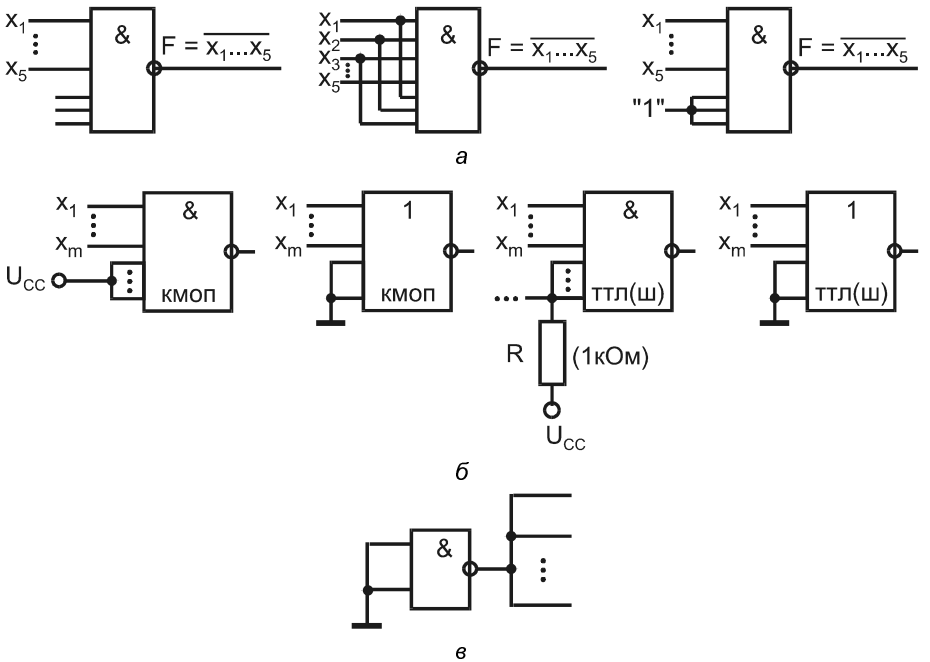


Рис. 1.37. Принципиально возможные (*а*) и рекомендуемые (*б*) режимы неиспользуемых входов логических элементов, схема формирования сигналов логической единицы (*в*)

Примеры подключения неиспользуемых выводов ИС показаны на рис. 1.37, *б*. Сигналы логической единицы можно получать от специального элемента (рис. 1.37, *в*), причем если это мощный элемент, то он может иметь большой коэффициент разветвления.

Согласование уровней сигналов при сопряжении разнотипных элементов

Иногда в одних и тех же устройствах приходится по тем или иным соображениям применять элементы разных схемотехнологических типов, что требует рассмотрения их совместимости по уровням напряжений, токов, быстродействию и т. д. Самая типичная ситуация — одновременное использование элементов КМОП и ТТЛ(Ш).

Выходные уровни U_1 и U_0 элементов КМОП близки соответственно к уровню питания и нулевому уровню. При подключении к таким элементам, имеющим U_{CC} = = 3…5 В, элементов ТТЛ(Ш) зачастую приемлема прямая передача сигналов (рис. 1.38, *а*). При этом низкий уровень, поступающий от КМОП-элемента, оказывается более "хорошим", чем аналогичный уровень, получаемый от элемента ТТЛ(Ш). Высокий уровень логической единицы у элементов КМОП близок к напряжению питания, а у элементов ТТЛ(Ш) этот уровень значительно меньше. Повышение уровня логической единицы благоприятно для помехоустойчивости схемы, но может быть опасно из-за возможности пробоя входных цепей. Если повышение уровня U_1 допустимо, то прямое управление элементом ТТЛ(Ш) от элемента КМОП вполне приемлемо. В частности, такое управление рекомендуется для серий микросхем КР1533 и КР1554.

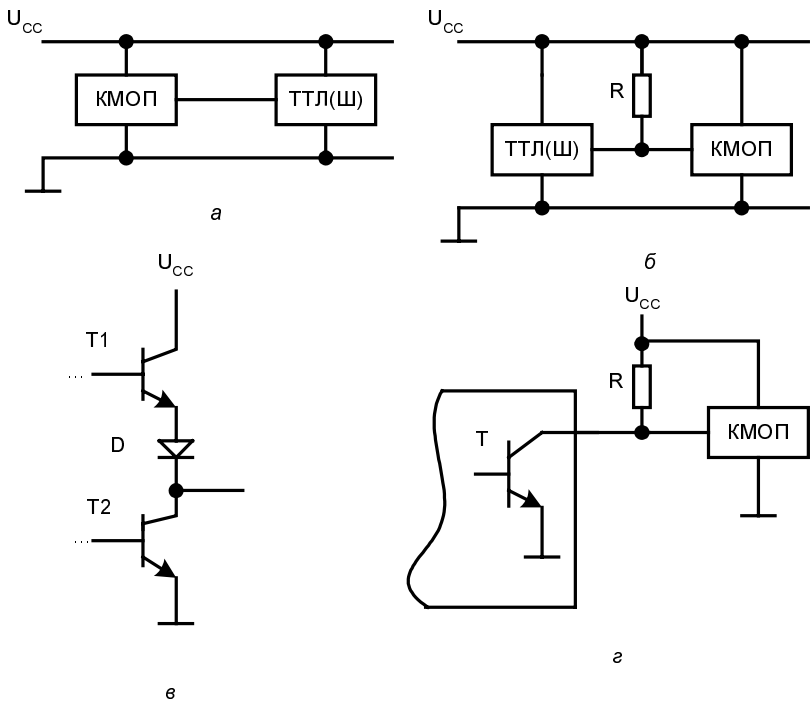


Рис. 1.38. Схемы согласования элементов КМОП и ТТЛ(Ш) (*а*, *б*) и пояснения к их работе (*в*, *г*)

В сочетаниях ТТЛ(Ш) — КМОП напряжение высокого уровня, формируемое выходным каскадом ТТЛ(Ш), обычно недостаточно для надлежащего управления элементами КМОП, и должно быть увеличено. В типовой схеме сопряжения (рис. 1.38, *б*) это выполняется с помощью цепочки U_{CC} — R. На первый взгляд схема рис. 1.38, *б* может показаться странной, поскольку в ней дополнительная цепочка U_{CC} — R должна воздействовать на выходное напряжение элемента ТТЛ(Ш). Выходные сопротивления элементов в рабочем режиме малы, что позволяет им не

поддаваться внешним воздействиям, сохраняя выработанные сигналы. Поэтому, как правило, никакие сигналы на выходы элементов не подаются. Однако в рассматриваемом случае ситуация иная. Типичная выходная цепь элемента ТТЛ(Ш) имеет вид рис. 1.38, *в*. При формировании высокого уровня напряжения транзистор T1 работает в схеме эмиттерного повторителя, и создает малое выходное сопротивление для тока, *вытекающего* из выходной цепи. Ток, втекающий извне в выходной электрод, напротив, встречает чрезвычайно высокое сопротивление запертого транзистора T2 и сопротивление обратно включенного диода D. Такая резкая асимметрия выходных сопротивлений каскада для вытекающего тока (обычного рабочего режима) и втекающего, создаваемого цепочкой U_{CC} — R, позволяет этой цепочке определять напряжение в линии связи между элементами, задавая уровень U_1, приблизительно равный U_{CC}, что и требуется для элемента КМОП. Рекомендуемые для сопряжения элементов серий КР1533 и КР1554 значения сопротивления резистора равны приблизительно 5 кОм. Информация, связанная с согласованием разнотипных элементов, приводится в соответствующих стандартах.

Режимы неиспользуемых элементов

Если не все элементы, имеющиеся в корпусе ИС, использованы, то лишние также подключены к напряжению питания, общему для всего корпуса. Если мощности, потребляемые элементами в состояниях нуля и единицы, не равны, то имеет смысл поставить неиспользуемый элемент в состояние минимальной мощности, подав на какой-либо из его входов соответствующую константу.

Наращивание числа входов

Работающий в схеме логический элемент принимает сигналы от нескольких предыдущих элементов и выдает сигналы нескольким следующим. Число входов такого элемента называют коэффициентом объединения ($К_{ОБ}$ или FanIn), а число выходов — коэффициентом разветвления ($К_Р$ или FanOut).

Наращивание числа входов для операций И и ИЛИ не представляет трудностей: для получения нужного числа входов берется несколько элементов, выходы которых объединяются далее элементом того же типа. Наращивание числа входов для операций И-НЕ, ИЛИ-НЕ, в сущности, производится аналогичным методом, но в схеме появляются дополнительные инверторы (рис. 1.39, *а*). На этом рисунке звездочка обозначает операцию Шеффера или Пирса.

Снижение нагрузок на выходах логических элементов

Снижение нагрузок может понадобиться, если они превышают допустимые значения, а также для повышения быстродействия схем, на которое нагрузки оказывают самое непосредственное влияние. Для предотвращения потерь быстродействия из-

за нагрузок на выходах сильно нагруженных элементов применяют буферизацию или разделение нагрузки (рис. 1.39, *б*, *в*).

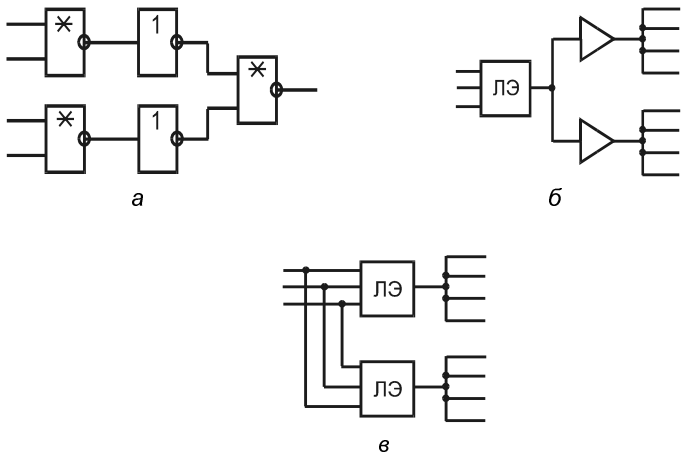


Рис. 1.39. Схемы наращивания числа входов (*а*)
и снижения нагрузки на выходах логических элементов (*б*, *в*)

Введение буферных каскадов облегчает и ускоряет работу источника сигнала, но вносит собственную задержку в тракт его передачи. Будет ли в конечном счете эффект ускорения, определяется конкретным расчетом.

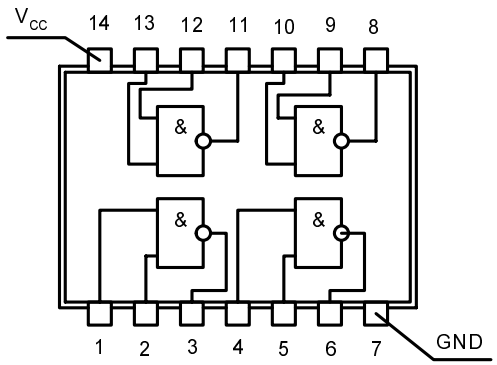
При разделении нагрузки новые элементы с задержками в тракт передачи сигнала не вводятся, но увеличивается нагрузка на тот источник сигнала, который питает рассматриваемую схему. Поэтому и здесь эффективность должна оцениваться конкретным расчетом.

§ 1.10. Прошлое и настоящее малых и средних интегральных схем. Логические примитивы в системах автоматизированного проектирования

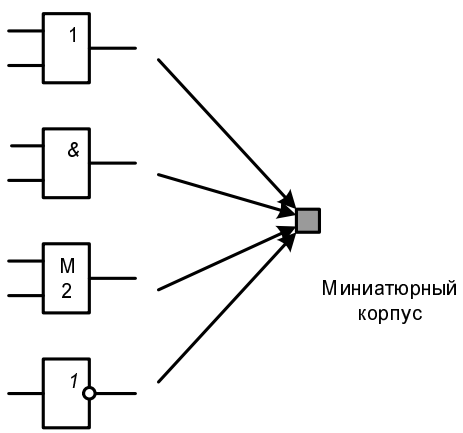
Малые и средние интегральные схемы (МИС и СИС) — старейшие на рынке цифровых элементов. Микросхемы ТТЛ появились в 1963 году, в последующие 10…15 лет были разработаны сотни их разновидностей. Позднее были освоены микросхемы типа КМОП, которые в то время проигрывали биполярным схемам по быстродействию, но отличались высокой компактностью, энергетической экономичностью, высокой помехоустойчивостью, способностью работать при изменениях питающего напряжения в широких пределах. Элементы КМОП по мере повышения их быстродействия стали все более вытеснять микросхемы ТТЛ, оставляя за ними схемотехнику буферных, согласующих и других элементов, которые должны

сохранять высокое быстродействие при больших нагрузках. Для особо скоростных устройств широко применялись и элементы ЭСЛ. Другие типы элементов применялись гораздо реже.

В течение нескольких десятилетий микросхемы ТТЛ, КМОП, ЭСЛ выпускались в виде МИС и СИС и проектировщики успели основательно привыкнуть к этому виду элементной базы. Сейчас выпуск ИС невысокого уровня интеграции не прекратился (МИС и СИС производятся 9 крупными фирмами), но их роль изменилась.



а



б

Рис. 1.40. Традиционное (*а*) и одновентильное (*б*) корпусирование малых интегральных схем

Возможность размещать на кристалле все больше и больше транзисторов привела к развитию микросхем высокого уровня интеграции с программируемой изготовителем или пользователем структурой. Новые кристаллы стали заменять все большее

и большее число стандартных МИС и СИС. Основная часть логических операций стала выполняться большими и сверхбольшими интегральными схемами (БИС и СБИС). И где-то в середине 90-х годов разработка новых функциональных вариантов МИС и СИС фактически прекратилась (до этого времени серии стандартных элементов постоянно пополнялись новыми типами микросхем). Сейчас МИС и СИС не исчезли, и в ограниченной степени применяются. Появились иные принципы их корпусирования — так называемое *одновентильное корпусирование* (Single-gate Logic).

Ранее проектировщики стремились получить в одном корпусе побольше элементов и изготовители старались максимально наполнить имеющиеся в их распоряжении корпуса, размещая в них несколько идентичных элементов. Например, в корпусе с 14 выводами размещали 4 элемента 2И-НЕ (рис. 1.40, *а*).

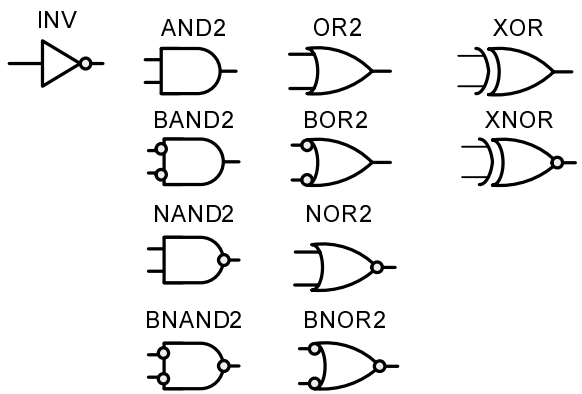


Рис. 1.41. Примитивы логических элементов фирмы Altera

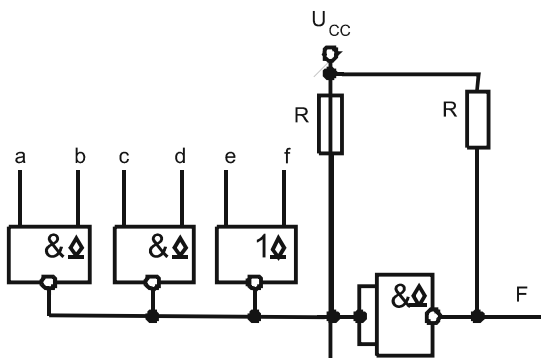
Такой подход был естественным для ситуации, когда МИС и СИС были основой цифровых устройств. Сейчас положение иное — основная работа ложится на схемы высокого уровня интеграции. Однако ограничиться только ими при построении устройств и систем не удается. Практически всегда возникает потребность в реализации одиночных логических функций или преобразовании уровней сигнала или буферизации линий интерфейса и т. д. Для удовлетворения подобных потребностей одновентильные схемы, т. е. схемы, реализующие одну несложную функцию, размещают в сверхминиатюрные корпуса с поверхностным монтажом (рис. 1.40, *б*). Малые размеры микросхем в сочетании с совершенными технологическими процессами их изготовления позволяют получить элементы высокого быстродействия (с задержками единицы наносекунд). Напряжения питания сверхминиатюрных МИС варьируются соответственно напряжениям питания БИС/СБИС.

При использовании систем автоматизированного проектирования (САПР) проектируемые схемы отображаются с помощью элементов, входящих в состав библиотек.

При этом применяются условные графические обозначения элементов, отличающиеся от рекомендуемых отечественным ГОСТом. На рис. 1.41 приведен состав логических элементов (примитивов), соответствующий библиотеке одной из ведущих фирм в области САПР.

Контрольные вопросы и упражнения

1. Какими параметрами характеризуются статические свойства логического элемента?
2. Какими параметрами логического элемента определяется его статическая помехоустойчивость? Запишите формулы для определения допустимых значений положительной и отрицательной статических помех.
3. Какими параметрами характеризуются динамические свойства логического элемента?
4. Что такое внутренняя емкость элемента типа КМОП?
5. На какие параметры логических элементов типа КМОП существенно влияет емкостная нагрузка?
6. Зачем нужны третьи состояния выходов (состояния "отключено") логических элементов и буферных схем?
7. Сравните свойства элементов с открытым электродом и с третьим состоянием при их применении в магистральных структурах.
8. Можно ли применять для выхода на одну и ту же магистраль одновременно два типа элементов: с открытым электродом и с третьим состоянием?
9. Какую функцию реализует показанная схема?



10. Перечислите способы устранения неопределенных состояний разомкнутых входов элементов типа КМОП.

11. Какие причины делают необходимой тщательную фильтрацию напряжений питания цифровых схем?

12. Почему для повышения эффективности фильтрации напряжений питания цифровых схем используют одновременно конденсаторы разных типов?

13. По каким причинам дифференциальные линии связи обладают более высокими быстродействием и помехозащищенностью в сравнении с однополюсными?

14. Для показанной на рисунке схемы передачи идеализированного (прямоугольного) сигнала U по линии с волновым сопротивлением Z_0 = 50 Ом, выходным сопротивлением источника сигнала $R_И$ и сопротивлением нагрузки R требуется построить на интервале от 0 до 10T (T — время пробега волны по линии) временные диаграммы напряжений в начале и конце линии при следующих условиях:

a) передается положительный фронт, $R_И$ = 20 Ом, R = 200 Ом.

b) передается отрицательный фронт, $R_И$ = 20 Ом, R = 200 Ом.

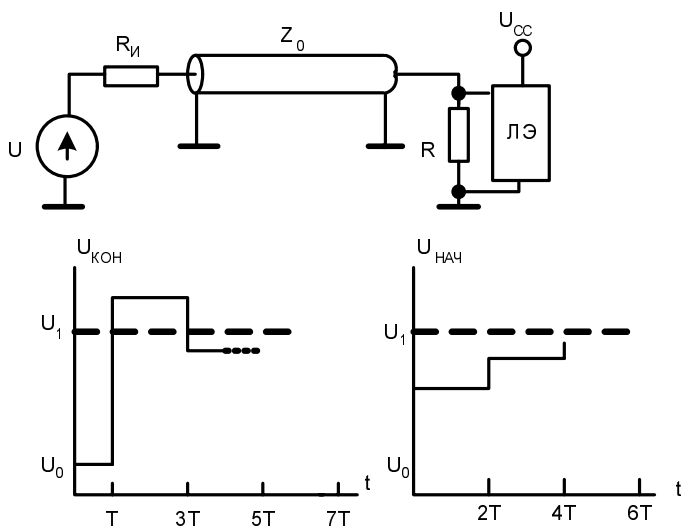
c) передается положительный фронт, $R_И$ = 10 Ом, R = 300 Ом.

d) передается отрицательный фронт, $R_И$ = 10 Ом, R = 300 Ом.

e) передается положительный фронт, $R_И$ = 30 Ом, R = 100 Ом.

f) передается отрицательный фронт, $R_И$ = 30 Ом, R = 100 Ом.

Ориентировочный вид диаграмм показан на рисунке. Диаграммы следует изображать с соблюдением масштаба.



15. Назовите несколько популярных стандартов сигналов ввода/вывода.

16. В чем состоит сущность технологии DDR (передачи данных с двойной скоростью)? Какие дополнительные схемные элементы нужны для реализации этой технологии?

17. Какими способами реализуются элементы задержки цифровых сигналов? В чем состоят достоинства и недостатки этих способов?

18. Какие схемы называются разностными преобразователями и детекторами событий?

19. Составьте логическую схему выработки сигнала управления одним из сегментов ССИ и схему подключения сегмента к управляющему сигналу, обеспечивающую требуемый ток зажигания диода. В логической схеме используются микросхемы КР1533ЛА23, в корпусе которых содержится четыре элемента 2И-НЕ с повышенной нагрузочной способностью и открытым коллектором. Индикатор с общим анодом обозначается как ОА, с общим катодом как ОК. При необходимости уменьшить ток, потребляемый от логического элемента, управляющего сегментом, сегмент следует подключать через транзисторный усилительный каскад, принимая коэффициент передачи тока для схемы с общим эмиттером равным 20.

Варианты заданий приведены в табл. 1.3.

Таблица 1.3

№ задания	Сегмент	Тип индикатора	Диапазон отображаемых цифр	Прямое напряжение на сегменте, В	Ток сегмента, мА
1	***a***	ОА	0…7	1,5	10
2	***b***	ОА	3…9	2	20
3	***c***	ОК	2…5	1,5	20
4	***d***	ОК	0…5	1,5	10
5	***e***	ОА	3…8	2	20
6	***f***	ОА	1…6	1,5	20
7	***g***	ОК	0…4	2	20

Выборочные сведения о статических параметрах микросхем КР1533ЛА23 представлены в табл. 1.4.

Таблица 1.4

Обозначение параметра	Наименование параметра	Норма	Единица измерения
U_{CC}	Напряжение питания	5 (допуск 10%)	В
U_{OL}	Выходное напряжение низкого уровня	$\leq 0{,}5$	В
I_{OH}	Выходной ток высокого уровня	$\leq 0{,}1$	мА
I_{OL}	Выходной ток низкого уровня	≤ 24	мА

Литература к главе: [10], [26], [48], [49], [57], [60].

Глава 2

Функциональные узлы комбинационного типа

§ 2.1. Проблематика проектирования комбинационных схем

Комбинационные цепи и автоматы с памятью

Как и все цифровые устройства, *функциональные узлы,* выполняющие типовые операции, делятся на *комбинационные* и *последовательностные*. В дальнейшем комбинационные узлы будем обозначать через КЦ (комбинационные цепи), а последовательностные через АП (автоматы с памятью). *Различия между КЦ и АП имеют фундаментальный характер.*

Выходные величины КЦ *в установившемся режиме* зависят только от текущего значения входных величин (аргументов). Предыстория значения не имеет. После завершения переходных процессов на выходах КЦ устанавливаются величины, на которые характер переходных процессов влияния не оказывает. С этой точки зрения переходные процессы в КЦ не опасны. Но в цифровых устройствах в целом КЦ функционируют совместно с АП, что кардинально меняет ситуацию.

Во время переходных процессов на выходах КЦ временно появляются сигналы, не предусмотренные логическими формулами и называемые *рисками сбоя* (или просто *рисками*). Риски возникают из-за временных состязаний сигналов в цепях схемы. Со временем ложные сигналы исчезают, и выход КЦ приобретает значение, предусмотренное логической формулой, описывающей работу цепи.

Риски могут быть восприняты и зафиксированы элементами памяти АП, изменение состояния которых может радикально нарушить нормальную работу ЦУ, предусмотренную алгоритмом его функционирования.

Структурно КЦ представляют собою цепи из логических элементов, в которых *отсутствуют обратные связи*, которые могут придавать этим цепям свойства памяти, т. е. создавать последовательностные схемы.

Риски сбоя

Различают *статические* и *динамические* риски сбоя (далее просто *риски*). Статические риски — это кратковременные изменения сигнала, который должен был бы оставаться неизменным (единичным или нулевым, соответственно чему говорят о единичном риске или нулевом риске). Если же состояние выхода должно измениться, но вместо однократного перехода происходят многократные, то имеет место динамический риск. При динамических рисках первый и последний переходы всегда совпадают с алгоритмическими, предусмотренными логикой работы схемы. Статический риск такого свойства не имеет.

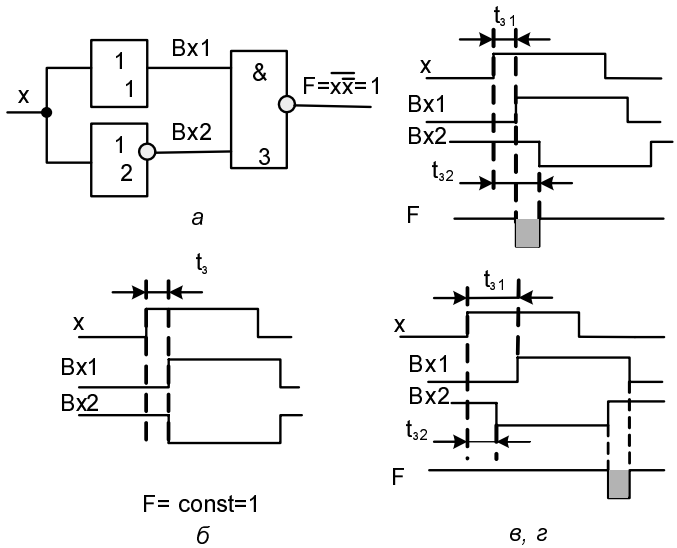


Рис. 2.1. Схема, иллюстрирующая механизм возникновения статического риска в комбинационной цепи (*а*), и временные диаграммы ее работы (*б*, *в*, *г*)

Простейший пример возникновения статического риска соответствует выработке функции "константа 1" по формуле $F = x\bar{x} = 1$ (рис. 2.1, *а*). В статике при любом значении x на одном из входов элемента И-НЕ имеется логический нуль, обеспечивающий единичное значение выхода. В переходных процессах возможен статический единичный риск. Не учитывая задержку элемента 3, которая здесь не играет принципиальной роли, рассмотрим временные диаграммы переходных процессов для случаев равенства задержек элементов 1 и 2 ($t_{31} = t_{32}$) (рис. 2.1, *б*), а также их неравенства ($t_{31} < t_{32}$ и $t_{31} > t_{32}$), показанные на рис. 2.1, *в*, *г*. Видно, что при различных задержках элементов возникает статический риск после положительного или отрицательного перепада входного сигнала в зависимости от того, задержка какого элемента больше.

Сигналы синхронизации

Для исключения возможных сбоев из-за явлений риска имеются два пути. Первый состоит в синтезе схем, свободных от рисков, и требует сложного анализа процессов в схеме и введения избыточных элементов для исключения рисков. Этот путь редко используется в практике.

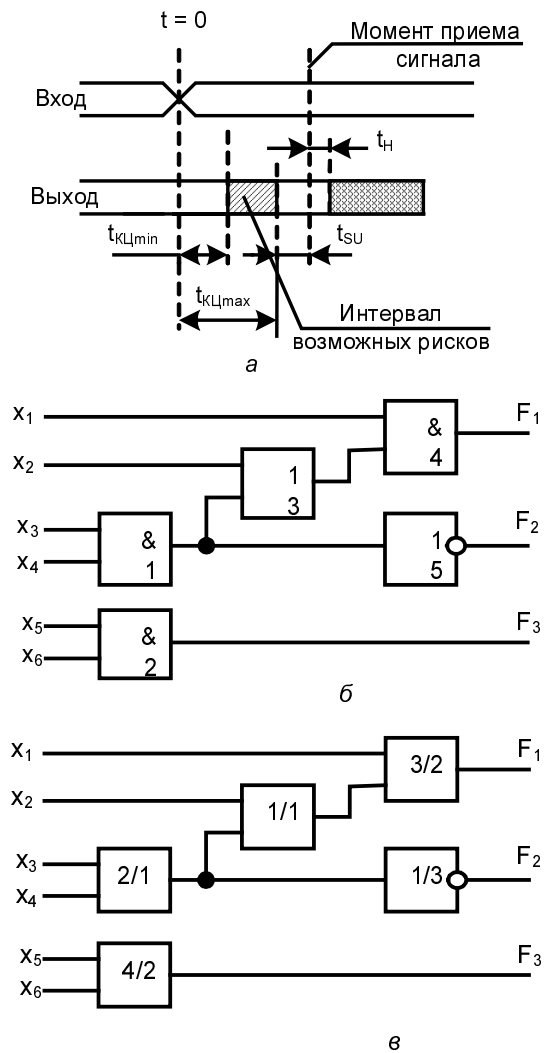


Рис. 2.2. Распространение сигналов в комбинационных цепях (*а*) и схемы для расчета длительности переходного процесса в комбинационной цепи (*б*, *в*)

Второй путь, основной для современной схемотехники, предусматривает *запрещение восприятия сигналов КЦ элементами памяти на время переходных процессов*. Прием информации с выходов КЦ разрешается только специальным сигналом синхронизации, подаваемым на элементы памяти после окончания

переходных процессов в КЦ. Таким образом, исключается воздействие ложных сигналов на элементы памяти. Иными словами, основная идея здесь может быть выражена словами *"переждать неприятности"*. Соответствующие структуры называются *синхронными*.

На рис. 2.2, *а* приведены временные диаграммы распространения сигналов в КЦ. В момент времени t = 0 обновляется набор входных переменных. После завершения времени распространения сигналов по самой короткой цепи $t_{КЦ\ min}$ на выходах начинаются изменения сигналов. По истечении времени распространения сигналов по самой длинной цепи $t_{КЦ\ max}$ переходные процессы заканчиваются и возможные ложные сигналы рисков исчезают. Выходные данные *становятся действительными* и могут быть использованы для запоминания. При этом до и после момента приема данных элементами памяти существуют интервалы, на которых выходные данные *должны оставаться неизменными*. Нарушение этого требования ведет к неправильной работе элементов памяти, принимающих данные от КЦ. Интервал неизменности данных, расположенный до момента их приема, называется *временем предустановки* t_{SU} (индекс SU происходит от слов Set-Up), а интервал после момента приема — *временем удержания (выдержки)* t_H (индекс H происходит от термина Hold).

Распространение сигналов в комбинационных цепях

Для определения временного интервала, на котором происходят переходные процессы, т. е. возможны ложные сигналы, следует оценить задержки на путях распространения сигналов от входов к выходам КЦ (рис. 2.2, *б*, *в*). Нужно найти пути с минимальной и максимальной задержками. Если на входе КЦ изменение аргументов произошло в нулевой момент времени, то по самому короткому пути до выхода F_3 сигнал может пройти за время $t_{з.2min}$, которое и обозначит начало интервала переходных процессов. На самом длинном пути (до выхода F_1) сигнал задержится не более, чем на время $t_з = t_{з.1.max} + t_{з.3.max} + t_{з.4.max}$, по истечении которого переходные процессы завершатся.

Для оценки задержек сигнала на самом коротком и самом длинном путях нужны сведения о минимальных и максимальных значениях задержек элементов. К сожалению, изготовитель, как правило, не приводит сведений о минимальных задержках. При их отсутствии многие разработчики принимают в качестве минимальной задержки *одну треть от типовой*.

Наиболее полно описывались бы задержки статистическими характеристиками, но они, как правило, неизвестны. Если известны только максимальные задержки, то зона возможного появления ложных сигналов расширяется влево и начинается от момента времени t = 0. Это увеличивает неопределенность ситуации, искажает реальную оценку работоспособности схем и может вынудить принять не лучшие схемотехнические решения.

В рассмотренном для схемы рис. 2.2, *б* примере не учитываются зависимости задержек от направления переключения элемента и предполагается близость значений задержек для всех элементов. На рис. 2.2, *в* дана уточненная версия той же схемы. Внутри логических элементов проставлены в виде дроби два числа, в числителе дается длительность переключения элемента из нулевого состояния в единичное, в знаменателе — из единичного в нулевое. При этом для величин задержек находится наибольший общий делитель, который принимается за единицу, а значения задержек выражаются целыми числами таких единиц. Предполагается, что набор входных переменных меняется в нулевой момент времени. Процесс переключения протекает различно в зависимости от исходного и нового набора входных переменных, и для сложной схемы с большим числом возможных комбинаций найти границы диапазона задержек не так-то просто. Для САПР разработаны алгоритмы решения таких задач. Показанная на рис. 2.2, *в* модель особенно актуальна для схем ТТЛ(Ш), имеющих существенно различные задержки при разных направлениях переключения. Для схемотехники КМОП такого различия нет и может оказаться приемлемой модель типа рис. 2.2, *б*.

Этапы разработки и средства реализации комбинационных цепей

В состав ЦУ обычно входят типовые функциональные узлы и некоторое количество логических схем, специфичных для данного конкретного проекта (как иногда говорят — произвольной логики), которые являются объектами индивидуального проектирования. Проектирование комбинационной цепи производится по этапам.

Прежде всего, задается характер функционирования КЦ. Это может быть сделано различными способами, чаще всего пользуются таблицами функционирования (таблицами истинности), задающими значение искомых функций на всех наборах аргументов. От таблицы легко перейти к СДНФ искомых функций (*СДНФ* — совершенная дизъюнктивная нормальная форма, т. е. дизъюнкция конституент единицы). Для этого составляют логическую сумму тех наборов аргументов, на которых функция принимает единичное значение.

Например, для функции четырех аргументов, заданной табл. 2.1, получим следующее выражение:

$$F = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3x_4 \vee x_1x_2\bar{x}_3x_4 \vee x_1x_2x_3x_4.$$

Таблица 2.1

x_1	x_2	x_3	x_4	**F**	x_1	x_2	x_3	x_4	**F**
0	0	0	0	**1**	1	0	0	0	**1**
0	0	0	1	**1**	1	0	0	1	**1**
0	0	1	0	**1**	1	0	1	0	**0**

Таблица 2.1 *(окончание)*

x_1	x_2	x_3	x_4	F	x_1	x_2	x_3	x_4	F
0	0	1	1	**1**	1	0	1	1	**0**
0	1	0	0	**0**	1	1	0	0	**0**
0	1	0	1	**0**	1	1	0	1	**1**
0	1	1	0	**0**	1	1	1	0	**0**
0	1	1	1	**0**	1	1	1	1	**1**

Дальнейшие действия зависят от средств реализации функций, к которым в современной схемотехнике относятся:

- логические блоки табличного типа (LUTs, Look-Up Tables);
- логические блоки в виде последовательности матриц элементов И и ИЛИ (*PLA* — Programmable Logic Array; *PAL* — Programmable Array Logic);
- универсальные логические блоки на основе мультиплексоров;
- логические блоки, собираемые из логических элементов некоторого базиса (*SLC* — Small Logic Cells).

Логические блоки табличного типа

Если КЦ будет реализована на основе логических блоков табличного типа, то *СДНФ явится окончательным выражением функции*, и никаких дальнейших преобразований этой формы не потребуется. Дело в том, что табличный блок представляет собою память, в которой имеется столько ячеек, сколько необходимо для хранения всех значений функций, т. е. 2^m, где m — число аргументов функции. Набор аргументов является адресом той ячейки, в которой хранится значение функции на этом наборе (0 или 1). СДНФ как раз и содержит все адреса, по которым нужно хранить единичные значения функции. Если искомая функция выражена в какой-либо сокращенной форме, то следует перевести ее в СДНФ. Для этого конъюнктивные члены, не содержащие переменной x_i, умножаются на равную единице дизъюнкцию $x_j \vee \overline{x}_j$. Например,

$$F(x_1, x_2, x_3) = x_1 \vee x_2\overline{x}_3 = x_1(x_2 \vee \overline{x}_2)(x_3 \vee \overline{x}_3) \vee x_2\overline{x}_3(x_1 \vee \overline{x}_1) =$$
$$= x_1x_2x_3 \vee x_1\overline{x}_2x_3 \vee x_1x_2\overline{x}_3 \vee x_1\overline{x}_2\overline{x}_3 \vee x_1x_2\overline{x}_3 \vee \overline{x}_1x_2\overline{x}_3.$$

Блок памяти для воспроизведения функции m переменных имеет вид рис. 2.3, *а*. Если требуется воспроизвести n функций, то в каждой ячейке нужно будет хранить n бит (по одному биту для каждой функции), и блок памяти будет организован, как показано на рис. 2.3, *б*.

а *б*

Рис. 2.3. Блоки памяти для воспроизведения одной (*а*) и нескольких (*б*) логических функций

Логические блоки с матрицами И и ИЛИ

Если проект реализуется на последовательно включенных матрицах элементов И и ИЛИ либо их эквиваленте в другом базисе, то может возникнуть необходимость поиска кратчайшей дизъюнктивной нормальной формы — *ДНФ*. Логические блоки с матрицами И и ИЛИ воспроизводят системы переключательных функций и имеют параметры: число входов, выходов и термов. Число входов (аргументов воспроизводимых функций) и число выходов (самих функций) от формы выражения функций не зависят и предопределены заданием. Число термов (конъюнкций) зависит от формы представления функций. Если число термов при данной форме представления функций превышает возможности имеющегося логического блока или блок проектируется специально для данной функции, то возникает вопрос о минимизации, целью которой будет *сокращение числа конъюнктивных термов в данной системе функций, т. е. поиск кратчайших дизъюнктивных форм*. Практически это сводится к поиску минимальных дизъюнктивных нормальных форм (ДНФ), о чем говорится далее, и отбору среди них вариантов с наименьшим числом термов. При реализациях на уже готовых средствах поиск можно прекратить, как только найдется форма с приемлемым числом термов, т. к. дальнейшее уменьшение числа термов эффекта уже не даст.

Логические блоки на основе мультиплексоров рассмотрены в *§ 2.5* после ознакомления с ИС мультиплексоров.

Блоки на основе типовых логических элементов

Синтез КЦ на логических блоках типа SLC, т. е. на вентильном уровне, является самым традиционным и изученным (термином *"вентиль"* называют базовые логические ячейки, выполняющие простейшие операции).

В этом варианте проектирование КЦ завершают следующие хорошо известные этапы:

- ❒ минимизация логических функций;
- ❒ переход к заданному логическому базису.

Заметим, что термин "минимизация" здесь понимается в широком смысле, т. е. как такое преобразование логических функций, которое упрощает их в смысле заданного критерия.

О КРИТЕРИЯХ МИНИМАЛЬНОСТИ (СЛОЖНОСТИ) ЦИФРОВЫХ СХЕМ

Исторически первым было стремление минимизировать число логических элементов в схеме (элементы были наиболее дорогими компонентами устройств), что приводит к критерию сложности схемы в виде числа букв в реализуемых выражениях. Этот критерий учитывается так называемой *ценой по Квайну* — суммарным числом входов всех логических элементов схемы. Для минимизации по этому критерию разработано несколько методов, в их числе как аналитические, основанные на преобразованиях математических выражений, так и графические, основанные на применении специальных карт (карт Карно, диаграмм Вейча), удобных, если число аргументов функции не превышает 6.

С переходом к ИС основным критерием аппаратной сложности схемы стала *площадь кристалла,* затрачиваемая на ее размещение.

Для функциональных узлов, целиком реализуемых на одном кристалле, площадь имеет прямой физический смысл и измеряется, например, в квадратных миллиметрах. Для устройств, реализуемых на печатной плате с использованием нескольких кристаллов, "площадь" измеряется числом корпусов. Так как корпуса ИС неодинаковы, их следует приводить к некоторым эквивалентным корпусам. Приведение учитывает число выводов корпуса, так, например, корпус с 24 выводами в 1,5 раза сложнее корпуса с 16 выводами. Операции приведения соответствует оценка суммарной площади корпусов *общим числом* всех выводов корпусов ИС.

Минимизация по числу букв в реализуемом выражении перестала точно соответствовать новому критерию, хотя между обоими критериями сохраняется известная связь.

Следующий этап проектирования — переход к заданному логическому базису от исходных выражений, которые обычно получают в булевском базисе (И, ИЛИ, НЕ). Правила такого перехода основаны на применении теоремы де-Моргана. В частности, для перехода к базису И-НЕ используется соотношение

$$F(x_1,x_2,x_3,x_4) = x_1x_2 \vee x_3x_4 = \overline{\overline{x_1x_2 \vee x_3x_4}} = \overline{\overline{x_1x_2} \cdot \overline{x_3x_4}},$$

а для перехода к базису Пирса удобно вначале получить исходную булевскую форму для инверсии искомой функции, а затем от нее перейти к базису ИЛИ-НЕ по соотношениям

$$\overline{F} = x_1x_2 \vee x_3x_4,$$

$$F = \overline{x_1x_2 \vee x_3x_4} = \overline{\overline{x_1x_2} \vee \overline{x_3x_4}} = \overline{\overline{x}_1 \vee \overline{x}_2} \vee \overline{\overline{x}_3 \vee \overline{x}_4}.$$

Традиционные методы минимизации приводят к каноническим формам функций алгебры логики, соответствующим (если входные переменные заданы и прямыми и инверсными значениями) двухъярусной реализации путем последовательного выполнения операций И и ИЛИ. Переход к базисам И-НЕ и ИЛИ-НЕ ярусность схем не изменяет. Для некоторых задач каноническое представление может оказаться слишком громоздким. Для упрощения выражений можно применять к ним *факторизацию*

(вынесение общих множителей за скобки и группирование членов), различного рода эквивалентные подстановки и др. Упрощение функций путем факторизации может дать упрощение схемы, но при этом увеличивается ее ярусность и, следовательно, возрастает задержка в выработке результата.

Возможные преобразования функций порождают множество вариантов, причем наиболее ценные иногда нелегко отыскать.

В работе [36] сказано (с. 6): "Примером исчисления, которым широко пользуются в процессе синтеза логических схем, являются преобразования алгебры логики. Набор правил говорит лишь о том, как можно преобразовать исходное булево выражение, но ничего не говорит о том, как нужно его преобразовать, чтобы на данном логическом базисе получить минимальную задержку или минимальное число корпусов или некоторый компромисс между этими требованиями".

Интегральные критерии качества цифровых схем

К проблематике проектирования ЦУ относится и вопрос о *критериях их качества*. Поскольку одну и ту же задачу можно решить многими способами, возникают альтернативные варианты проекта, которые нужно уметь сравнивать между собой. Объективная сложность сравнительной оценки вариантов обусловлена тем, что при этом имеет значение целый набор свойств для каждого варианта — *частных критериев* его качества. Каждый частный критерий имеет ясный, определенный смысл (аппаратная сложность, быстродействие, потребляемая мощность, помехоустойчивость и др.), но не может исчерпывающим образом охарактеризовать вариант. А чтобы учесть несколько частных критериев качества, нужно сформировать *общий критерий* (интегральный, многоцелевой, функцию качества, функцию ценности). Формирование такого критерия — чрезвычайно ответственная задача, не имеющая формального решения. *В любую форму общего критерия качества входят коэффициенты, назначаемые субъективно*. Таким образом, возникает ситуация, когда для оценки устройства применяется критерий, а для него самого оценки качества не существует. Поэтому в практике проектирования сложные общие критерии качества не популярны. Достаточно признанным можно, пожалуй, считать лишь критерий AT, где A — аппаратная сложность устройства, T — время решения задачи. Но и здесь проявляется общий недостаток, свойственный всем общим критериям — в них может происходить взаимная компенсация частных критериев, и уменьшение одного может быть скомпенсировано ростом другого, что формально равноценно, но не всегда разумно.

§ 2.2. Двоичные дешифраторы

Двоичные дешифраторы преобразуют двоичный код в код "1 из N", в кодовых комбинациях которого одна позиция занята единицей, а все остальные — нулевые. Например, код "1 из 4" содержит 4 кодовых комбинации следующего вида: 1000, 0100, 0010, 0001. Для дешифратора с инверсными выходами код "1 из 4" будет представлен комбинациями 0111, 1011, 1101, 1110. Таким образом, *в зависимости*

от входного двоичного кода на выходе дешифратора возбуждается одна из выходных цепей.

Из сказанного видно, что двоичный дешифратор, имеющий n входов, должен иметь 2^n выходов, соответствующих числу разных комбинаций в n-разрядном двоичном коде. Если часть входных наборов не используется, то дешифратор называют *неполным*, и у него число выходов меньше 2^n.

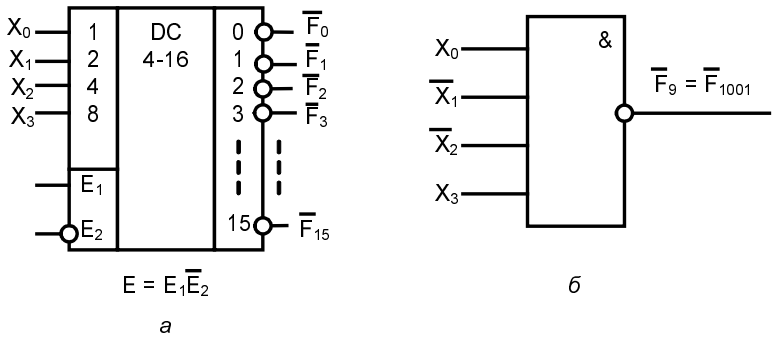


Рис. 2.4. Условное обозначение (*а*), элемент И-НЕ в цепи одного из выходов (*б*)

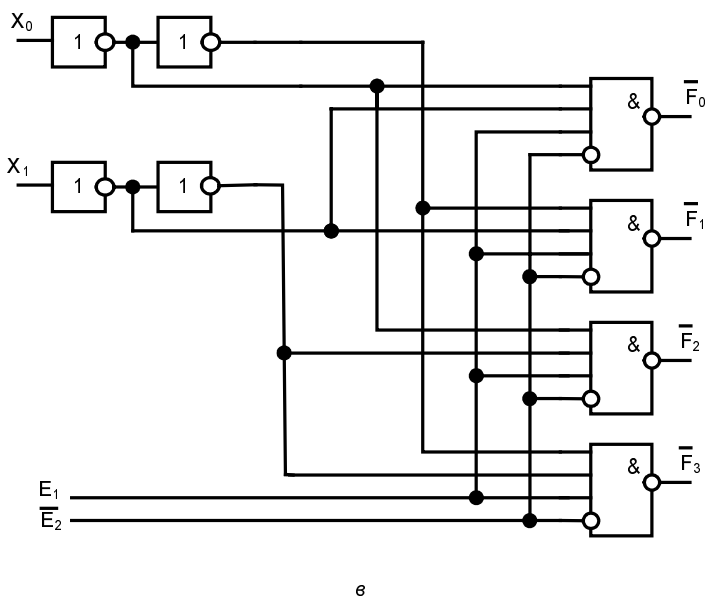


Рис. 2.4. Схемная реализация (*в*) двоичного дешифратора с инверсными выходами

В условном обозначении дешифраторов проставляются буквы DC (от Decoder). Входы дешифратора удобно отмечать их двоичными весами. Кроме информаци-

онных входов дешифратор обычно имеет один или более входов разрешения работы, обозначаемых как EN или E (от Enable). При разрешении по этому входу дешифратор работает описанным выше образом, при его отсутствии все выходы дешифратора пассивны. Если входов разрешения несколько, то общий сигнал разрешения работы образуется как конъюнкция этих сигналов. Дешифраторы, имеющие входы разрешения (один или несколько), называют также *дешифраторами-демультиплексорами*, поскольку с их помощью можно организовать режим демультиплексирования (см. *§ 2.4*).

Условное графическое обозначение полного двоичного дешифратора показано на рис. 2.4, *а*. Большинство дешифраторов имеют инверсные выходы, т. к. элементы И-НЕ схемотехнически выигрывают у элементов И. В этом случае только один (активный) выход имеет нулевое значение, а все остальные единичное. При запрещении работы такого дешифратора на всех его выходах будет присутствовать логическая единица.

Функционирование дешифратора с инверсными выходами и прямым входом разрешения EN описывается системой конъюнкций:

$$\overline{F}_0 = \overline{x}_0\overline{x}_1\overline{x}_2\ldots\overline{x}_{n-2}\overline{x}_{n-1}EN,$$

$$\overline{F}_1 = \overline{x}_0\overline{x}_1\overline{x}_2\ldots\overline{x}_{n-2}x_{n-1}EN,$$

$$\overline{F}_2 = \overline{x}_0\overline{x}_1\overline{x}_2\ldots\overline{x}_{n-2}x_{n-1}EN,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\overline{F}_{2^n-2} = x_0x_1x_2\ldots x_{n-2}\overline{x}_{n-1}EN,$$

$$\overline{F}_{2^n-1} = x_0x_1x_2\ldots x_{n-2}x_{n-1}EN.$$

Схемотехническая реализация дешифраторов

Схемотехнически дешифратор представляет собою совокупность конъюнкторов (или элементов И-НЕ в дешифраторах с инверсными выходами), не связанных между собой. Каждый конъюнктор (или элемент И-НЕ) вырабатывает одну из выходных функций (на рис. 2.4, *б* для примера показан без учета входа разрешения элемент И-НЕ девятого выхода дешифратора с инверсными входами). Дешифратор снабжен также схемами для выработки парафазных сигналов из однофазных (прямых), поступающих на входы ИС. Заметим, что входная прямая переменная непосредственно в схеме не используется, а вырабатывается повторно как двойная инверсия от входной. Это сделано для того, чтобы *максимально разгружать линии внешних входов* (здесь внешние входы нагружены только на один вход инвертора). Внешние линии входов максимально разгружаются всегда, поскольку они и без того нагружены емкостью из-за своей относительно большой длины и конструкции выводов корпуса ИС, что снижает скорость передачи сигнала по линии.

Схема дешифратора "2—4" с инверсными выходами и двумя входами разрешения, в которой общий разрешающий сигнал вырабатывается как логическое произведение $E_1\overline{E}_2$, приведена на рис. 2.4, *в*.

Время установления выходного сигнала дешифратора

$$t_{DC} = 2t_{з.инв.} + \max(t_з^{10}, t_з^{01}),$$

где $t_{з.инв.}$ — задержка сигнала в инверторе; $t_з^{10}$, $t_з^{01}$ — задержки переключений логического элемента.

Для размещения в недорогом корпусе с небольшим числом выводов годится только дешифратор с 4 информационными входами. Более "размерных" дешифраторов в стандартных сериях ИС нет. При реализации дешифратора как внутрикристального узла его размерность может быть значительно большей.

Неполные дешифраторы с числом выходов меньше 2^n применяются в различных ситуациях, в частности при работе с двоично-десятичными кодами, когда четырем входам (тетраде) соответствует не шестнадцать выходов, как в двоичной системе счисления, а только десять.

Пример применения дешифратора

Дешифраторы широко применяются в схемах адресации памяти, что проиллюстрировано в *главе 6*. Используются они и в других случаях. В качестве примера на рис. 2.5, *а* показана схема подключения нескольких источников данных к линии связи коллективного пользования, работающей в режиме разделения времени. Источники имеют выходы с третьим состоянием и L-активные сигналы разрешения работы.

К общей линии подключены восемь источников, имеющих выходы с тремя состояниями. Адресный код a_2—a_0 определяет номер источника, которому разрешается выход на общую линию. Все остальные источники в это время должны быть запрещены. Эта функция выполняется дешифратором "3—8".

На рис. 2.5, *б* представлено решение этой же задачи для источников с открытым выходом (стоковым или коллекторным). Специального входа разрешения работы здесь не требуется: один из входов элемента И-НЕ может управлять его подключением к линии. При нулевом значении этого входа на выходе элемента независимо от состояния других входов будет формироваться логическая единица, представляемая запертым транзистором, и элемент отключится от линии, не мешая работе других источников. При единичном состоянии управляющего входа он не влияет на работу элемента, подключающегося к линии, по другим входам.

Наращивание размерности дешифратора

Из малоразрядных дешифраторов можно построить схему, эквивалентную дешифратору большей разрядности. Для этого входное слово делится на поля. Разрядность поля младших разрядов соответствует числу входов имеющихся дешифрато-

ров. Оставшееся поле старших разрядов служит для получения сигналов разрешения работы одного из дешифраторов, декодирующих поле младших разрядов. В качестве примера на рис. 2.6 приведена схема дешифрации пятиразрядного двоичного кода с помощью дешифраторов "3—8" и "2—4". Для получения нужных 32 выходов составляется столбец из четырех дешифраторов "3—8". Дешифратор "2—4" принимает два старших разряда входного кода. Возбужденный нулевой выход этого дешифратора отпирает один из дешифраторов столбца по его входу разрешения. Выбранный дешифратор столбца расшифровывает три младших разряда входного слова.

Каждому входному слову соответствует возбуждение только одного выхода. Например, при дешифрации слова $x_4x_3x_2x_1x_0 = 11001_2 = 25_{10}$ на входе дешифратора первого яруса имеется код 11, возбуждающий его выход номер три (показано крестиком), что разрешает работу DC4. На входе DC4 действует код 001, поэтому нуль появится на его первом выходе, т. е. на 25 выходе схемы в целом, что и требуется. Общее разрешение или запрещение работы схемы осуществляется по входу E дешифратора первого яруса.

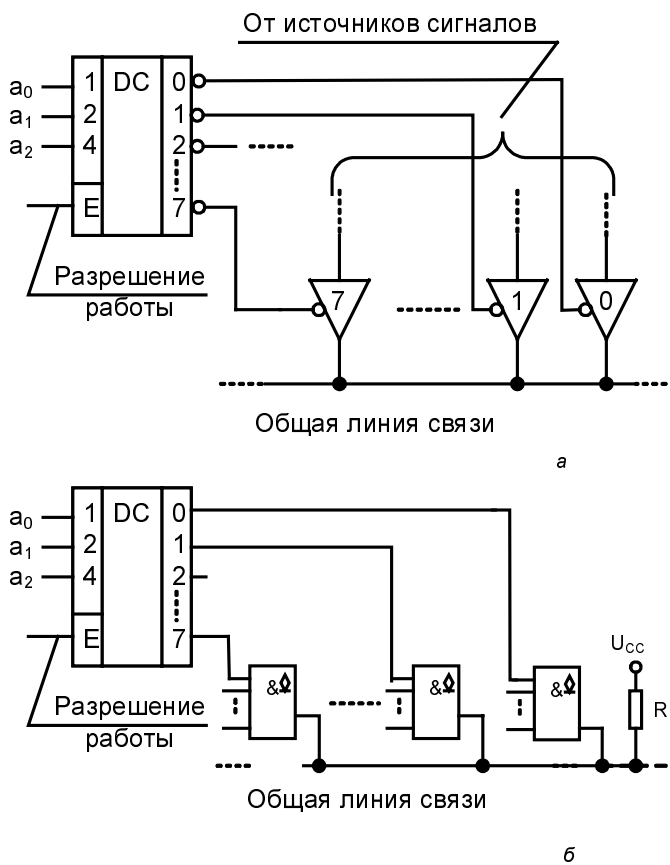


Рис. 2.5. Подключение нескольких источников сигнала к общей линии связи

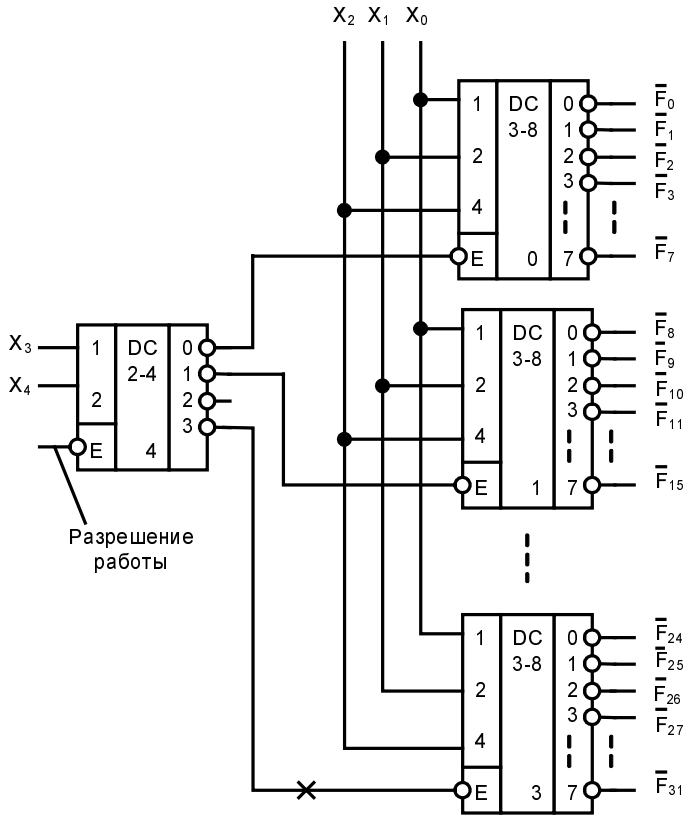


Рис. 2.6. Схема наращивания размерности двоичного дешифратора

Воспроизведение логических функций

Дешифраторы совместно со схемами ИЛИ можно использовать для воспроизведения произвольных логических функций. Действительно, на выходах дешифратора вырабатываются все минтермы, которые только можно составить из данного числа аргументов. Логическая функция в СНДФ есть дизъюнкция некоторого числа таких минтермов. Собирая нужные минтермы по схеме ИЛИ, можно получить любую функцию данного числа аргументов.

На рис. 2.7 в качестве примера показана схема выработки двух функций $F_1 = \overline{x}_3\overline{x}_2 \vee x_3x_1$ и $F_2 = \overline{x}_3\overline{x}_2x_1 \vee x_2\overline{x}_1$. Такое решение может быть целесообразным при необходимости выработки нескольких функций одних и тех же аргументов. В этом случае для выработки дополнительной функции добавляется только один дизъюнктор. Заметим, что для проверки правильности схемы рис. 2.7 удобно перевести функции F_1 и F_2 в СДНФ.

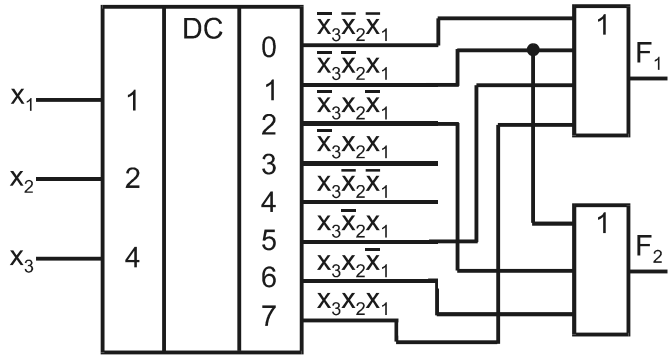


Рис. 2.7. Схема воспроизведения произвольных логических функций с помощью дешифратора и дизъюнкторов

§ 2.3. Приоритетные и двоичные шифраторы. Указатели старшей единицы

Шифраторы. Двоичные шифраторы выполняют операцию, обратную по отношению к операции дешифратора: они преобразуют код "1 из N" в двоичный. При возбуждении одного из входов шифратора на его выходе формируется двоичный код номера возбужденной входной линии. Полный двоичный шифратор имеет 2^n входов и n выходов.

Приоритетные шифраторы выполняют более сложную операцию. При работе ЭВМ и других устройств часто решается задача определения приоритетного претендента на пользование каким-либо ресурсом. Несколько конкурентов выставляют свои запросы на обслуживание, которые не могут быть удовлетворены одновременно. Нужно выбрать запрос, которому предоставляется право первоочередного обслуживания. Простейший вариант решения указанной задачи — присвоение каждому источнику запросов фиксированного приоритета. Например, группа из восьми запросов $R_7...R_0$ (R от английского Request) формируется так, что высший приоритет имеет источник номер семь, а далее приоритет уменьшается от номера к номеру. Самый младший приоритет у нулевого источника — он будет обслуживаться только при отсутствии всех других запросов. Если имеются одновременно несколько запросов, обслуживается запрос с наибольшим номером. *Приоритетный шифратор вырабатывает на выходе двоичный номер старшего запроса.*

Легко видеть, что при наличии всего одного возбужденного входа приоритетный шифратор работает так же, как двоичный. Поэтому в сериях и библиотеках элементов двоичный шифратор как самостоятельный элемент может отсутствовать. Режим его работы — частный случай работы приоритетного шифратора.

Указатели старшей единицы решают в сущности ту же задачу, что и приоритетные шифраторы, но вырабатывают результат в иной форме — в виде кода "1 из N". Таким образом, *при наличии на входах нескольких возбужденных линий (запросов) на выходе будет возбуждена лишь одна, соответствующая старшему запросу.* Чис-

ло входов в этом случае равно числу выходов схемы. Указатели старшей единицы применяются в устройствах нормализации чисел с плавающей точкой и т. д.

В промышленных сериях элементов имеются шифраторы приоритета для восьмиразрядных и десятиразрядных слов. Функционирование их отображается в табл. 2.2, где через X обозначено произвольное значение сигнала.

Таблица 2.2

EI	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0	A_2	a_1	a_0	G	EO
1	1	X	X	X	X	X	X	X	1	1	1	1	0
1	0	1	X	X	X	X	X	X	1	1	0	1	0
1	0	0	1	X	X	X	X	X	1	0	1	1	0
1	0	0	0	1	X	X	X	X	1	0	0	1	0
1	0	0	0	0	1	X	X	X	0	1	1	1	0
1	0	0	0	0	0	1	X	X	0	1	0	1	0
1	0	0	0	0	0	0	1	X	0	0	1	1	0
1	0	0	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	X	X	X	X	X	X	X	X	0	0	0	0	0

Таблица полностью характеризует работу приоритетного шифратора при всех возможных комбинациях сигналов: EI — сигнала разрешения работы данного шифратора; EO — сигнала, вырабатываемого на выходе данного шифратора при отсутствии запросов на его входах для разрешения работы следующего (младшего) шифратора при наращивании размерности шифраторов; G — сигнала, отмечающего наличие запросов на входе данного шифратора; $R_7...R_0$ — запросов на входах шифратора; $a_2...a_0$ — значений разрядов выходного двоичного кода, формирующего номер старшего запроса. Все перечисленные сигналы формируются при условии EI = 1 (работа шифратора разрешена). При EI = 0 независимо от состояний входов запросов все выходные сигналы шифратора становятся нулевыми.

Из таблицы можно получить следующие выражения для функций a_2, a_1, a_0, EO, G:

$$a_2 = (R_7 \vee \overline{R}_7 R_6 \vee \overline{R}_7 \overline{R}_6 R_5 \vee \overline{R}_7 \overline{R}_6 \overline{R}_5 R_4)EI,$$

$$a_1 = (R_7 \vee \overline{R}_7 R_6 \vee \overline{R}_7 \overline{R}_6 \overline{R}_5 \overline{R}_4 R_3 \vee \overline{R}_7 \overline{R}_6 \overline{R}_5 \overline{R}_4 \overline{R}_3 R_2)EI,$$

$$a_0 = (R_7 \vee \overline{R}_7 \overline{R}_6 R_5 \vee \overline{R}_7 \overline{R}_6 \overline{R}_5 \overline{R}_4 R_3 \vee \overline{R}_7 \overline{R}_6 \overline{R}_5 \overline{R}_4 \overline{R}_3 \overline{R}_2 R_1)EI,$$

$$EO = \overline{R}_7 \overline{R}_6 \overline{R}_5 \overline{R}_4 \overline{R}_3 \overline{R}_2 \overline{R}_1 \overline{R}_0 EI,$$

$$G = (R_7 \vee R_6 \vee R_5 \vee R_4 \vee R_3 \vee R_2 \vee R_1 \vee R_0)EI.$$

Повторным применением для функций в скобках известного соотношения алгебры логики $a \vee F\,\bar{a} = a \vee F$ можно получить выражения:

$$a_2 = (R_7 \vee R_6 \vee R_5 \vee R_4)EI,$$
$$a_1 = (R_7 \vee R_6 \vee \bar{R}_5\bar{R}_4R_2 \vee \bar{R}_5\bar{R}_4R_3)EI,$$
$$a_0 = (R_7 \vee \bar{R}_6R_5 \vee \bar{R}_6\bar{R}_4R_3 \vee \bar{R}_6\bar{R}_4\bar{R}_2R_1)EI,$$

определяющие внутреннюю структуру основной части шифратора приоритета.

Наращивание размерности приоритетного шифратора

На рис. 2.8 изображено наращивание числа входов шифратора приоритета вдвое (от 8 до 16). При этом условными обозначениями показаны шифраторы с инверсными входами и выходами, как это свойственно большинству серий элементов.

Шифратор 2 — старший по приоритету, его работа всегда разрешена подачей нуля на вход EI. Если на входах $\bar{R}_8 \ldots \bar{R}_{15}$ есть хотя бы один запрос, то разрешения на работу младшего шифратора 1 нет (EO = 1), его выходы пассивны, т. е. имеют единичные значения.

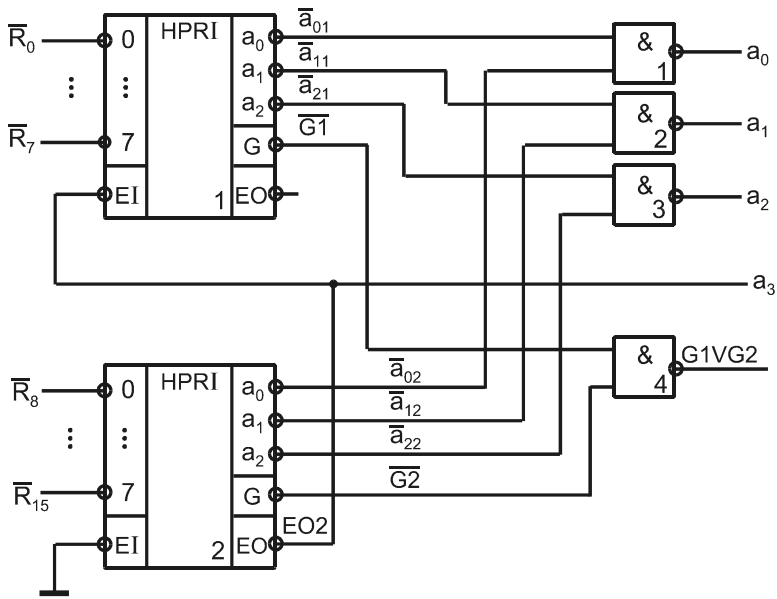


Рис. 2.8. Схема наращивания размерности приоритетного шифратора

При этом элементы И-НЕ с номерами 1, 2, 3 играют роль инверторов для сигналов a_{i2} (i = 0, 1, 2). Поэтому на выходах a_0, a_1, a_2 схемы формируются сигналы от

нуля до семи соответственно номеру старшего запроса в шифраторе 2, что вместе с единицей на выходе EO2, играющем роль разряда a_4 выходного кода, дает номера от 8 до 15.

Если на входах шифратора 2 запросов нет, он разрешает работу младшего, вырабатывая сигнал $\overline{EO2} = 0$ и приводит свои выходы a_0, a_1, a_2 в пассивное единичное состояние. Теперь на выходы a_i схемы в целом передаются инвертированные значения выходов a_{01}, a_{11}, a_{21} младшего шифратора, что вместе с нулем в разряде a_3 соответствует номерам от нуля до семи.

Таким образом, строится схема с 16 входами запросов, причем вход $\overline{R}_{15}$ имеет старший приоритет. Выход элемента 4 принимает единичное значение при наличии хотя бы одного запроса в любом из шифраторов и может использоваться как сигнал запроса на прерывания для процессора с последующим указанием процессору номера старшего запроса.

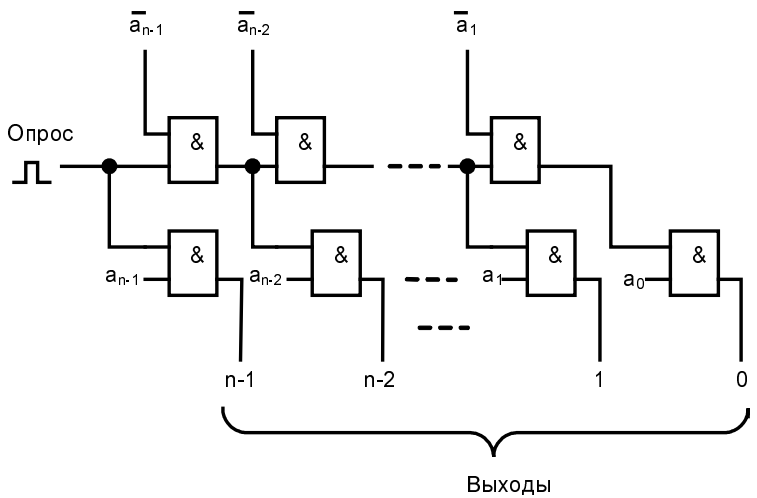


Рис. 2.9. Схема указания старшей единицы

Указатели старшей единицы. Такие указатели могут быть реализованы подключением двоичного дешифратора к выходу шифратора приоритета, но эту же задачу можно решить с помощью специальной цепочечной схемы (рис. 2.9) путем последовательного опроса разрядов, начиная со старшего, и прекращения дальнейшего опроса при выявлении первой же единицы.

В этой схеме единичный сигнал опроса, подаваемый со стороны старшего разряда a_{n-1}, может распространяться вправо только до первого разряда, содержащего единицу. Разряд, содержащий ноль, пропускает сигнал опроса, на его выходе остается нулевой уровень. На выходе единичного разряда конъюнктор блокируется нулевым значением инвертированной переменной, и дальнейшее распространение переноса прекращается. Одновременно на выходе разряда возникает единичный сигнал.

§ 2.4. Мультиплексоры и демультиплексоры

Мультиплексоры

Мультиплексоры осуществляют подключение одного из входных каналов к выходному под управлением адресующего кода. Разрядности каналов могут быть различными, мультиплексоры для коммутации многоразрядных слов составляются из одноразрядных.

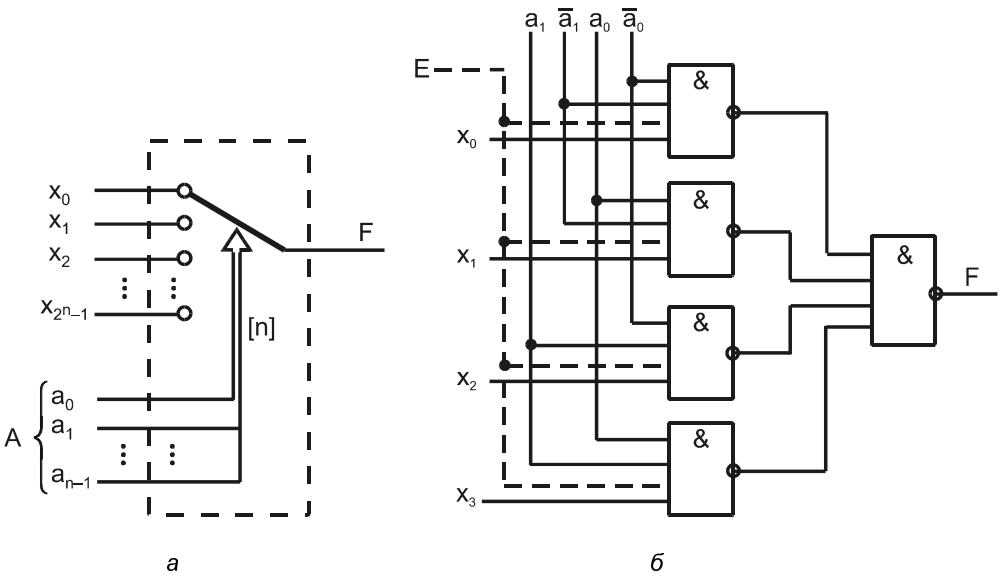


Рис. 2.10. Упрощенное представление мультиплексора многопозиционным ключом (*а*) и реализация мультиплексора на элементах И-НЕ (*б*)

Входы мультиплексора делятся на две группы: информационные и адресующие (управляющие). Работу мультиплексора можно упрощенно представить с помощью многопозиционного ключа. Для одноразрядного мультиплексора это представлено на рис. 2.10, *а*. Адресующий код A задает переключателю определенное положение, соединяя с выходом F один из информационных входов x_i. При нулевом адресующем коде переключатель занимает верхнее положение x_0, с увеличением кода на единицу переходит в соседнее положение x_1 и т. д.

Работа мультиплексора описывается соотношением

$$F = x_0\bar{a}_{n-1}\bar{a}_{n-2}\ldots\bar{a}_1\bar{a}_0 \vee x_1\bar{a}_{n-1}\bar{a}_{n-2}\ldots\bar{a}_1\bar{a}_{n-2}\ldots\bar{a}_1 a_0 \vee \ldots \vee x_{2^n-1}a_{n-1}a_{n-2}\ldots a_1 a_0,$$

которое иногда называют *мультиплексной формулой*. При любом значении адресующего кода все слагаемые, кроме одного, равны нулю. Ненулевое слагаемое равно x_i, где i — значение текущего адресного кода.

Схемотехнически мультиплексор реализует электронную версию показанного переключателя, имея, в отличие от него, только одностороннюю передачу данных. На рис. 2.10, *б* показан мультиплексор с четырьмя информационными входами, двумя адресными входами и входом разрешения работы. При отсутствии разрешения работы (E = 0) выход F становится нулевым независимо от информационных и адресных сигналов. В стандартных сериях размерность мультиплексоров не более 16×1.

Мультиплексоры в КМОП-схемотехнике

Схемотехника мультиплексоров на элементах КМОП имеет свои особенности. В число элементов схемы здесь наряду с обычной логикой входят специальные двунаправленные ключи, что позволяет строить экономичные структуры как мультиплексоров, так и других устройств (в частности, триггеров).

Двунаправленный ключ. Ключ (рис. 2.11) составлен из двух параллельно включенных транзисторов с взаимоинверсными напряжениями C и $\overline{C}$ на затворах. Один из транзисторов имеет n-канал, а другой — p-канал.

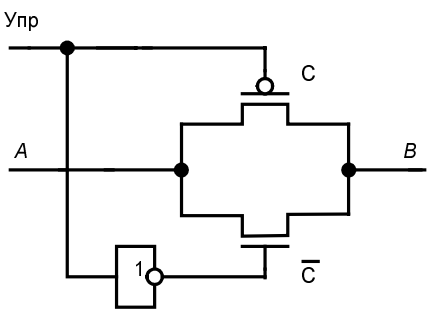


Рис. 2.11. Двунаправленный ключ

Такая схема позволяет получать замкнутое состояние ключа с приблизительно *постоянным и малым сопротивлением* независимо от величины переключаемого напряжения (т. е. напряжения $U_A \approx U_B$, что имеет место в замкнутом ключе). При этом ключ может работать при идентичных уровнях переключаемого и управляющего напряжений, как это и требуется для цифровых схем.

Обычный однотранзисторный ключ указанного свойства не имеет. У него уровни входного (управляющего) и переключаемого напряжений не могут быть одинаковыми. Если, например, на затвор транзистора с n-каналом подать отпирающее напряжение U_1 (уровень логической единицы), то переключаемое напряжение ($U_A \approx U_B$) должно быть меньше чем U_1, по меньшей мере на величину порогового напряжения, т. е. приблизительно вдвое. Иначе нарушится условие возникновения в транзисторе проводящего канала (Uзи >Uпор, где Uзи — напряжение на затворе относительно истока и Uпор — пороговое напряжение транзистора). Практически же максимальное переключаемое напряжение долж-

но быть еще меньше, т. к. при приближении к порогу резко увеличивается сопротивление канала и ухудшаются условия передачи сигналов. Указанное обстоятельство исключает применение однотранзисторных ключей для передачи логических сигналов под управлением других логических сигналов, поскольку в этом случае не обеспечивается превышение сигналов на затворах ключей над переключаемыми сигналами.

Иначе выглядит ситуация для ключей с параллельным включением двух разнотипных транзисторов. Подача на затвор транзистора с p-каналом низкого напряжение, а на затвор транзистора с n-каналом — высокого напряжения приводит ключ в замкнутое состояние. Изменение переключаемого напряжения при постоянстве напряжений на затворах изменяет напряжения "затвор-исток" и "затвор-сток", но при этом сопротивление одного из транзисторов растет, а другого падает, так что общее сопротивление ключа остается низким и приблизительно постоянным и, тем самым, решается проблема успешной передачи логических сигналов без потери их амплитуды. При подаче на затвор транзистора с p-каналом высокого напряжения, а на затвор транзистора с n-каналом — низкого напряжения, оба транзистора запираются и ключ размыкается.

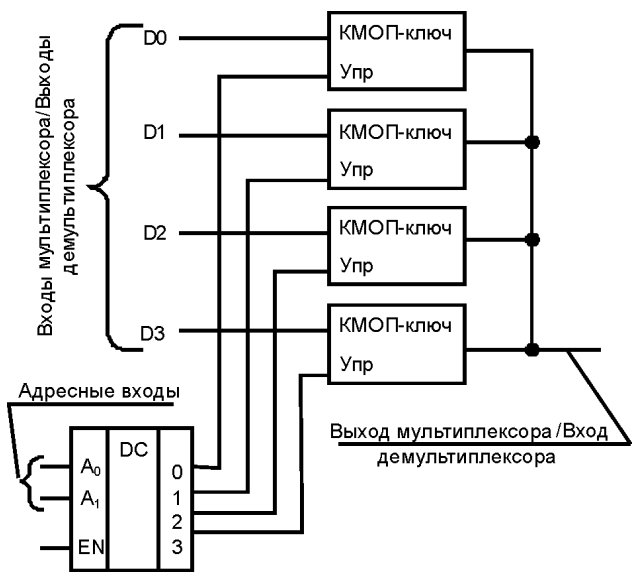


Рис. 2.12. Структура мультиплексора, реализованного в КМОП-схемотехнике

В КМОП-схемотехнике одна и та же схема функционирует как мультиплексор или демультиплексор (при перемене местами входов и выходов). Кроме того, схема может работать не только с цифровыми, но и с аналоговыми сигналами, поскольку двунаправленный ключ на двух транзисторах обладает высоким качеством и способен передавать уровни сигналов с весьма высокой точностью. Схема мультиплексора/демультиплексора показана на рис. 2.12.

Рассмотрим режим мультиплексора "4—1". Входные сигналы D0…D3 подключены к КМОП-ключам, выходы которых соединены друг с другом в точке выхода мультиплексора. Адресующий код A_1A_0 дешифрируется и вырабатывает управляющий

сигнал разрешения работы, т. е. замыкания, для ключа, номер которого соответствует коду A_1A_0. Остальные ключи остаются запертыми. Таким образом, на выход передается один из входных сигналов под управлением адресующего кода. Сигнал EN разрешает или запрещает работу мультиплексора.

Многоразрядные мультиплексоры

Для коммутации многоразрядных данных строятся мультиплексоры, представляющие собою совокупность одноразрядных схем. В этом случае мультиплексор характеризуется числом входных каналов *n* и их разрядностью *m* (рис. 2.13, *а*).

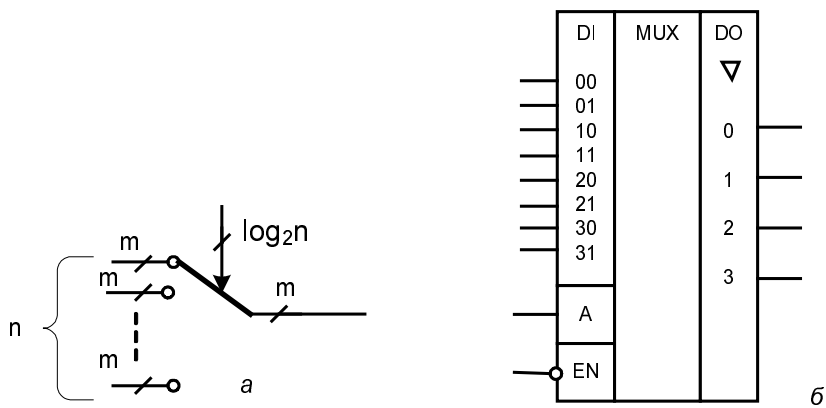


Рис. 2.13. Структура (*а*) и условное графическое обозначение (*б*) многоразрядного мультиплексора

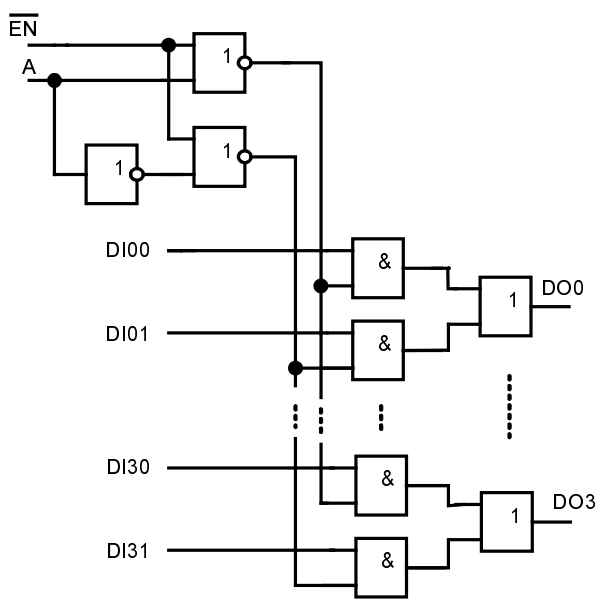


Рис. 2.14. Функциональная схема двухканального четырехразрядного мультиплексора

Условное графическое обозначение двухканального четырехразрядного мультиплексора представлено на рис. 2.13, *б*. Мультиплексор работает в соответствии с табл. 2.3. В нумерации сигналов первая цифра относится к разряду передаваемых слов, а вторая к каналу.

Таблица 2.3

EN_L	A	DO0	DO1	DO2	DO3
1	X	0	0	0	0
0	0	DI00	DI10	DI20	DI30
0	1	DI01	DI11	DI21	DI31

Функциональная схема мультиплексора показана на рис. 2.14.

Наращивание размерности мультиплексоров

Наращивание размерности мультиплексоров возможно с помощью пирамидальной структуры. При этом первый ярус схемы представляет собой столбец, содержащий столько мультиплексоров, сколько необходимо для получения нужного числа информационных входов. Все мультиплексоры столбца адресуются одними и теми же младшими разрядами адресного кода (если число информационных входов схемы равно 2^n, то общее число адресных разрядов равно n, младшее поле n_1 адресного кода используется для адресации мультиплексоров первого яруса). Старшие разряды адресного кода, число которых равно $n - n_1$, используются во втором ярусе, мультиплексор которого обеспечивает поочередную работу мультиплексоров первого яруса на общий выходной канал.

Пирамидальная схема, выполняющая функции мультиплексора "32—1" и построенная на мультиплексорах меньшей размерности, показана на рис. 2.15 (сокращение MUX от английского MUltipleXer).

Демультиплексоры

Демультиплексоры выполняют операцию, обратную операции мультиплексоров — передают данные из одного входного канала в один из нескольких каналов-приемников. Многоразрядные демультиплексоры составляются из нескольких одноразрядных. Условное обозначение демультиплексоров на примере размерности "1—4" показано на рис. 2.16.

Нетрудно заметить, что дешифратор со входом разрешения работы будет работать в режиме демультиплексора, если на вход разрешения подавать информационный сиг-

нал. Действительно, при единичном значении этого сигнала адресация дешифратора (подача адресного кода на его входы) приведет к возбуждению соответствующего выхода, при нулевом — нет. А это и соответствует передаче информационного сигнала в адресованный выходной канал.

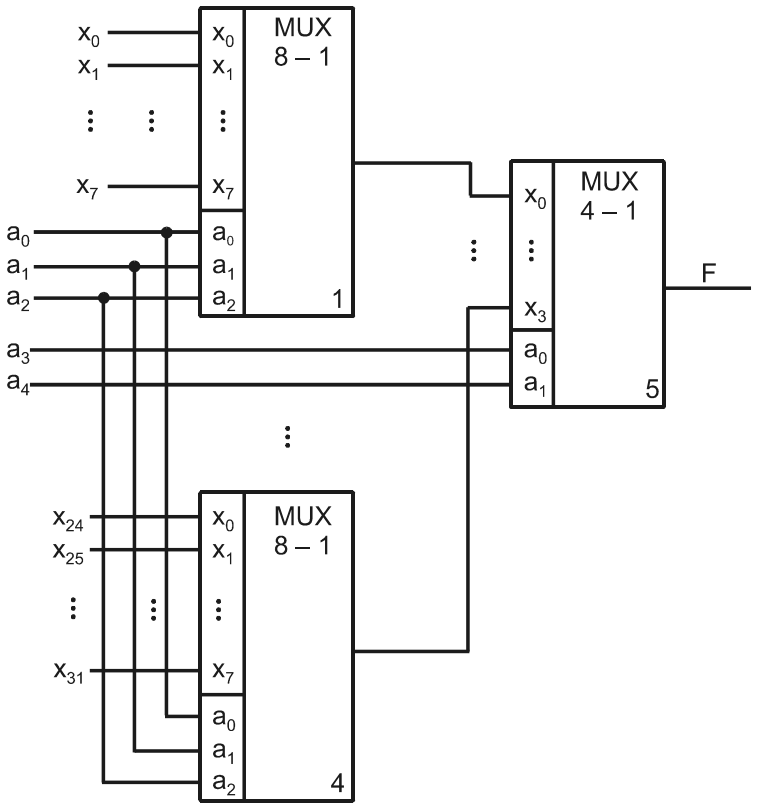


Рис. 2.15. Наращивание размерности мультиплексоров

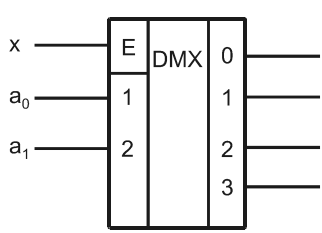
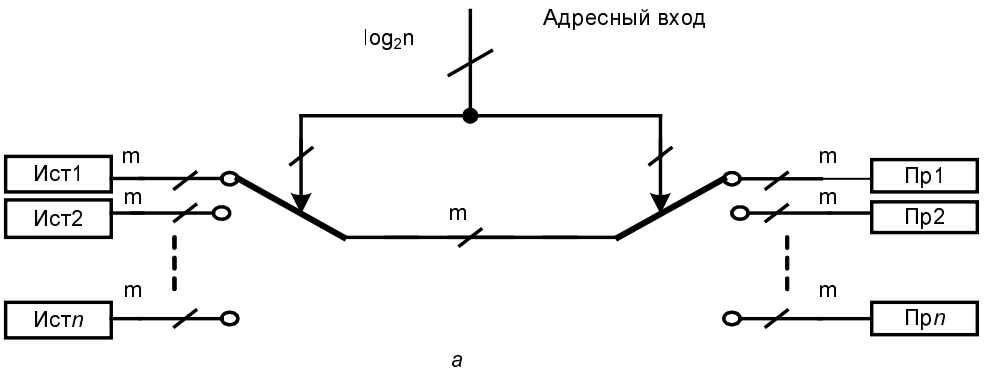


Рис. 2.16. Условное обозначение дешифратора-демультиплексора

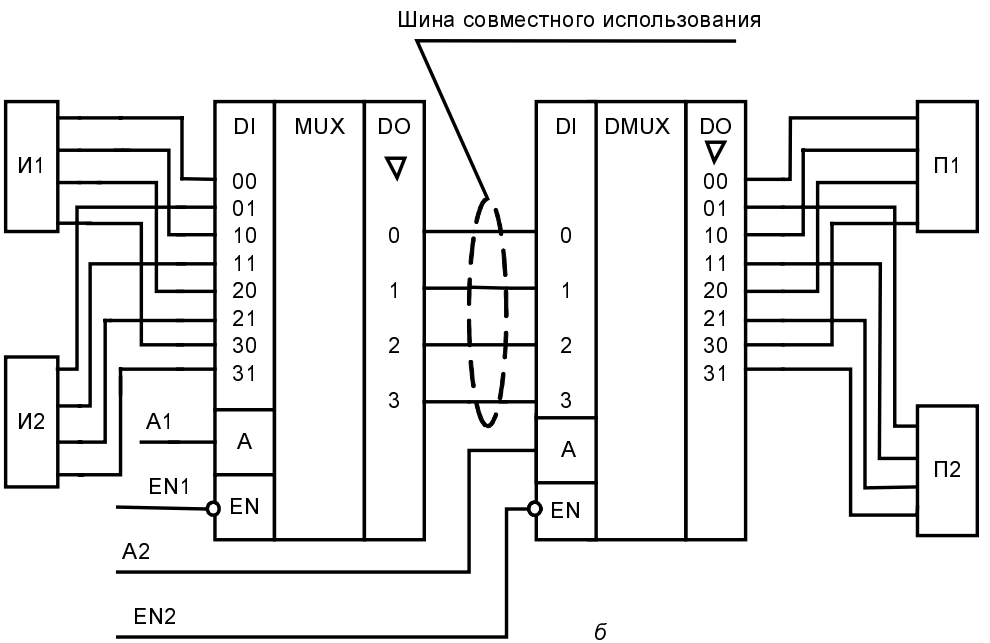
В связи с указанным, в сериях элементов демультиплексоры часто отсутствуют, а *дешифраторы со входом разрешения играют роль дешифраторов-демультиплексоров*.

Мультиплексоры и демультиплексоры в системах коммутации

Системы коммутации — главная область применения мультиплексоров и демультиплексоров. В частности, они позволяют организовать совместное использование шин несколькими источниками и приемниками сигналов. На рис. 2.17, *а* изображена структура схемы с поочередным обменом данными для *n* шин с разрядностью *m*.



а



б

Рис. 2.17. Структура коллективного пользования одной шиной (*а*) и функциональная схема шины, общей для двух четырехразрядных источников (*б*)

Общий для мультиплексора и демультиплексора адресный вход согласованно подключает к шине пару источник-приемник (Ист-Пр) с одинаковыми номерами. Смена адресного кода передает шину другой паре. Соответствующая структуре рис. 2.17, *а* функциональная схема для n = 2 и m = 4 приведена на рис. 2.17, *б*.

В этой схеме адресный сигнал A1 выбирает источник сигнала (И1 или И2), четырехразрядное слово которого передается на выход мультиплексора. Адресный сигнал A2 выбирает приемник (П1 или П2), которому будет передано это слово. Для всех передач используется одна и та же шина, соединяющая мультиплексор и демультиплексор. Сигнал разрешения работы EN1 может отключать мультиплексор от этой шины, отдавая ее в распоряжение других устройств, которые могут быть подключены к шине.

§ 2.5. Универсальные логические модули на основе мультиплексоров

Универсальные логические модули (УЛМ) на основе мультиплексоров относятся к устройствам, настраиваемым на воспроизведение той или иной функции. Универсальность их состоит в том, что для заданного числа аргументов можно настроить УЛМ на любую функцию. Известно, что общее число функций n аргументов выражается как 2^{2^n} и с ростом n растет чрезвычайно быстро. Хотя практический интерес представляют не все существующие функции, возможность получить любую функцию свидетельствует о больших возможностях УЛМ.

Первый способ настройки УЛМ

Первый способ настройки УЛМ — фиксация некоторых входов. Найдем для этого способа соотношение между числом аргументов функции n и числом настроечных входов m. Пусть требуется настройка на любую из функций. Тогда число комбинаций для кода настройки, равное числу функций, есть 2^{2^n}. Для двоичного кода число комбинаций равно 2^m, где m — разрядность кода. Приравнивая число воспроизводимых функций к числу комбинаций кода настройки, получаем соотношение $m = 2^n$.

На адресные входы УЛМ следует подавать аргументы функции, а на информационные входы — сигналы настройки (рис. 2.18). Таким образом, *для использования мультиплексора в качестве УЛМ следует изменить назначение его входов*.

Рисунок 2.18 иллюстрирует возможность воспроизведения с помощью мультиплексора любой функции n аргументов. Действительно, каждому набору аргументов соответствует передача на выход одного из сигналов настройки. Если этот сигнал есть значение функции на данном наборе аргументов, то задача решена. Разным функциям будут соответствовать разные коды настройки. Алфавитом настройки будет {0, 1} — настройка осуществляется константами 0 и 1. На

рис. 2.19 показан пример воспроизведения функции неравнозначности $x_1 \oplus x_2$ с помощью мультиплексора "4—1".

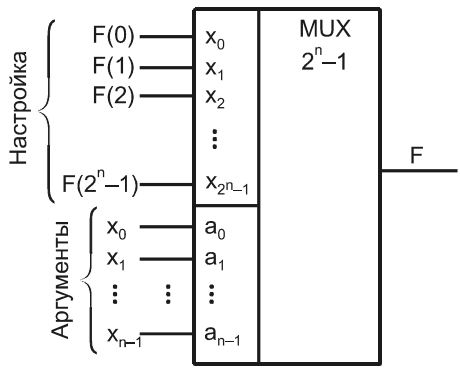


Рис. 2.18. Схема использования мультиплексора в качестве УЛМ

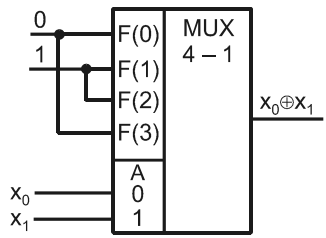


Рис. 2.19. Пример воспроизведения функции при настройке константами

Большое число настроечных входов затрудняет реализацию УЛМ. Для УЛМ, расположенных внутри кристалла, можно вводить код настройки последовательно в сдвигающий регистр, к разрядам которого подключены входы настройки. Тогда внешним входом настройки будет всего один, но настройка будет занимать не один такт, а 2^n тактов. Возможны и промежуточные последовательно-параллельные варианты ввода кода настройки.

Второй способ настройки УЛМ

Большое число входов настройки наталкивает на поиск возможностей их уменьшения. Такие возможности существуют и заключаются в *расширении алфавита настроечных сигналов.* Если от алфавита {0, 1} перейти к алфавиту $\{0,1, \widetilde{x}_i\}$, где $\widetilde{x}_i$ — литерал одного из аргументов, то число входов аргументов сократится на единицу, а число настроечных входов — вдвое. Напомним, что под *литералом* переменной понимается либо сама переменная, либо ее инверсия. Перенос одного из аргументов в число сигналов настройки не влечет за собою каких-либо схемных изменений. На том же оборудовании будут реализованы функции с числом аргументов на единицу больше, чем при настройке константами.

Для алфавита $\{0,1, \tilde{x}_i\}$ код настройки находится следующим образом. Аргументы за исключением $\tilde{x}_i$ подаются на адресующие входы, что соответствует их фиксации в выражении для искомой функции, которая становится функцией единственного аргумента $\tilde{x}_i$. Эту функцию, которую назовем *остаточной*, и нужно подавать на настроечные входы.

Если искомая функция зависит от n аргументов и в число сигналов настройки будет перенесен один из аргументов, то возникает n вариантов решения задачи, т. к. в сигналы настройки может быть перенесен любой аргумент. Спрашивается, какой именно аргумент целесообразно переносить в сигналы настройки? Здесь можно опираться на рекомендацию: в настроечные сигналы следует переносить аргумент, который имеет *минимальное число вхождений в термы функции*. В этом случае будут максимально использованы внутренние логические ресурсы мультиплексора, а среди сигналов настройки увеличится число констант, что благоприятно для схемной реализации УЛМ.

Проиллюстрируем сказанное примером воспроизведения функции трех аргументов $F = x_1x_2x_3 \vee \bar{x}_2\bar{x}_3$. Минимальное число вхождений в выражение функции имеет переменная x_1, которую и перенесем в число сигналов настройки. Остаточная функция определится табл. 2.4, *а*.

Таблица 2.4

x_2	x_3	$F_{ост}$
0	0	1
0	1	0
1	0	0
1	1	x_1

а

x_4	x_3	$F_{ост}$
0	0	$x_1\, x_2$
0	1	1
1	0	$x_1\, x_2$
1	1	$x_1\, x_2$

б

Схема УЛМ приведена на рис. 2.20.

По пути расширения алфавита сигналов настройки можно идти и дальше, но при этом понадобятся дополнительные логические схемы, воспроизводящие остаточные функции, которые будут уже зависеть более чем от одного аргумента.

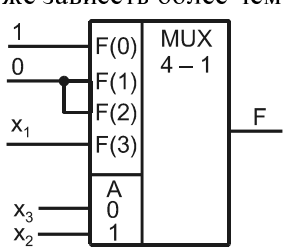


Рис. 2.20. Пример воспроизведения функции при переносе одного аргумента в число сигналов настройки

Если в сигналы настройки перевести два аргумента, то дополнительные логические схемы будут двухвходовыми, что мало усложняет УЛМ и может оказаться приемлемым решением. Пример реализации функции $F = x_1x_2 \vee x_3\bar{x}_4$ при алфавите настройки $\{0,1, \tilde{x}_1, \tilde{x}_2\}$ показан на рис. 2.21. Таблица остаточной функции для этого примера приведена в табл. 2.4, *б*.

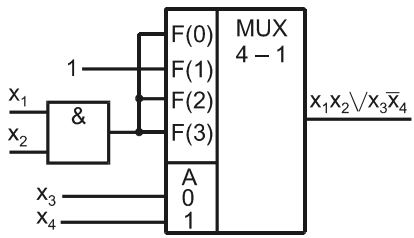


Рис. 2.21. Пример воспроизведения функции четырех аргументов на мультиплексоре "4—1"

Структуры УЛМ, содержащие несколько мультиплексоров

Дальнейшее расширение алфавита настройки путем переноса переменных в сигналы настройки требует вычислений остаточных функций трех или более аргументов. Вычисление остаточных функций с помощью мультиплексоров приводит к пирамидальной структуре, в которой мультиплексоры первого яруса реализуют остаточные функции, а мультиплексор второго яруса вырабатывает искомую функцию (такие структуры подробно рассмотрены во втором издании этой книги).

Пирамидальные структуры — каноническое решение, которое приводит к нужному результату, но не претендует на оптимальность. Варианты построения схем из нескольких мультиплексоров разнообразны, но алгоритм поиска структуры, оптимальной по затратам оборудования или какому-либо другому критерию, отсутствует. Имеются работы, в которых найдены эффективные решения, но это результаты изобретений, а не формализованного поиска структур.

В некоторых микросхемах программируемой логики используются функциональные преобразователи на мультиплексорах, причем воспроизводимость всех функций данного числа аргументов не преследуется, что упрощает схемы преобразователей. Схемы характеризуются порождающей функцией, реализуемой, когда все их входы используются как информационные (т. е. для подачи аргументов). Эта функция при введении настройки, когда часть входов занята под настроечные сигналы, порождает некоторый список подфункций, зависящих от меньшего числа аргументов.

На рис. 2.22 показан логический блок, используемый в микросхемах программируемой логики фирмы Actel (США). Изображены обозначения фирмы для мультиплексоров "2—1" (адресующие входы расположены сбоку). При S = 0 на выход передается сигнал верх-

него входа, при S = 1 — нижнего. Функциональная характеристика (порождающая функция) для этого блока имеет вид

$$F = \overline{S_0 \vee S_1}(\overline{S_A}A_0 \vee S_A A_1) \vee (S_0 \vee S_1)(\overline{S_B}B_0 \vee S_B B_1).$$

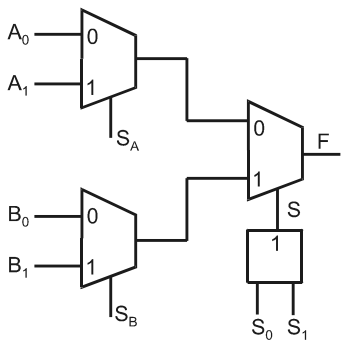


Рис. 2.22. Логические блоки, используемые в микросхемах фирмы Actel

Варьируя подачу на входы блока констант и входных переменных, можно реализовать более 700 практически полезных переключательных функций.

§ 2.6. Компараторы

Компараторы (устройства сравнения) определяют *отношения между двумя словами*. Основными отношениями, через которые можно выразить остальные, можно считать два — "равно" и "больше".

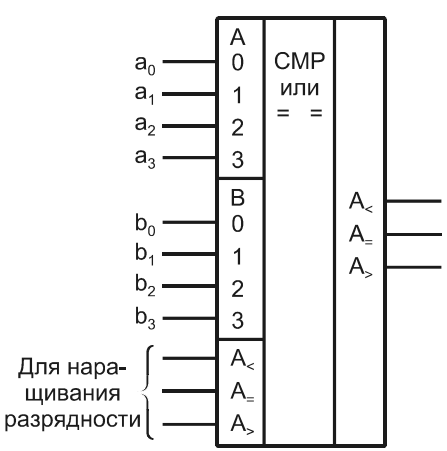


Рис. 2.23. Условное обозначение компаратора с тремя выходами

Определим выходные функции компараторов следующим образом: они принимают единичное значение (истинны), если соблюдается условие, указанное в индексе

обозначения функции. Например, функция $F_{A=B} = 1$, если A = B и принимает нулевое значение при $A \neq B$.

Приняв в качестве основных отношения "равно" и "больше", для остальных можно записать:

$$F_{A\neq B} = \overline{F}_{A=B};\ F_{A<B} = F_{B>A};\ F_{A\geq B} = \overline{F}_{B>A};\ F_{A\leq B} = \overline{F}_{A>B}.$$

Отношения используются как логические условия в микропрограммах, в устройствах контроля и диагностики ЭВМ и т. д.

В сериях цифровых элементов обычно имеются компараторы с тремя выходами: "равно", "больше" и "меньше" (рис. 2.23). Для краткости записей в индексе выходных функций указывается только слово A.

Сравнение на равенство

Устройства сравнения на равенство строятся на основе поразрядных операций над одноименными разрядами обоих слов. Слова равны, если в них равны все одноименные разряды, т. е. если в обоих нули или единицы. Признак равенства разрядов

$$r_i = a_i b_i \vee \overline{a}_i \overline{b}_i = \overline{a_i \overline{b}_i \vee \overline{a}_i b_i} = \overline{\overline{a_i b_i} \cdot \overline{\overline{a}_i \overline{b}_i}} = \overline{a_i \oplus b_i}.$$

Признак неравенства разрядов

$$\overline{r}_i = a_i \overline{b}_i \vee \overline{a}_i b_i = \overline{a_i b_i \vee \overline{a}_i \overline{b}_i} = \overline{\overline{a_i \overline{b}_i} \cdot \overline{\overline{a}_i b_i}} = a_i \oplus b_i.$$

Признак равенства слов $R = r_{n-1}\, r_{n-2} \ldots r_0$.

Схема компаратора на равенство в базисе И-НЕ показана на рис. 2.24, *а*.

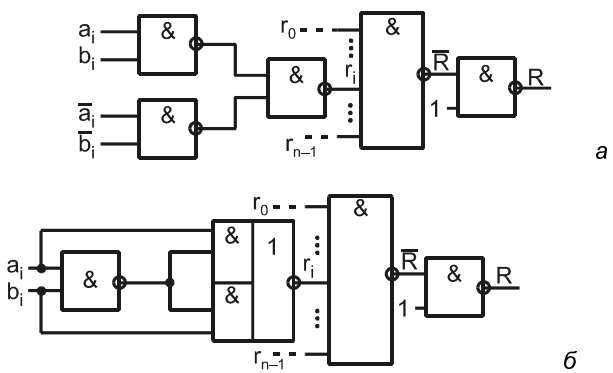


Рис. 2.24. Схемы компараторов на равенство

Схема без парафазных входов (рис. 2.24, *б*) основана на выражениях для r_i, преобразованных следующим образом:

$$r_i = \overline{a_i \overline{b}_i \vee \overline{a}_i b_i} = \overline{a_i (\overline{a}_i \vee \overline{b}_i) \vee b_i (\overline{a}_i \vee \overline{b}_i)} = \overline{a_i \overline{a_i b_i} \vee b_i \overline{a_i b_i}}.$$

Сравнение на "больше"

Построение компаратора на "больше" для одноразрядных слов (табл. 2.5) требует реализации функции $F_{A>B} = a\bar{b}$.

Таблица 2.5

a	b	$F_{A>B}$
0	0	0
0	1	0
1	0	1
1	1	0

Функцию $F_{A>B}$ для многоразрядных слов проще всего получить на основе рассуждений. Пусть нужно сравнить двухразрядные слова. Если старшие разряды a_1 и b_1 не равны, то результат известен независимо от младших разрядов: при $a_1 = 1$ и $b_1 = 0$ имеем $A > B$, а при $a_1 = 0$ и $b_1 = 1$ имеем $A < B$.

Если же $a_1 = b_1$, то результат еще неизвестен, и требуется анализ следующего разряда по тому же алгоритму. Поэтому для двухразрядных слов можно записать $F_{A>B} = a_1\bar{b}_1 \vee r_1 a_0 \bar{b}_0$.

Подобный же подход справедлив и для слов любой разрядности — к анализу следующего разряда нужно переходить только при равенстве предыдущих. Таким образом, для общего случая n-разрядных слов имеем

$$F_{A>B} = a_{n-1}\bar{b}_{n-1} \vee r_{n-1}a_{n-2}\bar{b}_{n-2} \vee \ldots \vee r_{n-1}r_{n-2}\ldots r_1 a_0 \bar{b}_0.$$

Примечание

Правильно рассуждая, мы получили правильный результат. Однако цель минимизации формул при этом не ставилась и на самом деле выражения для $F_{A>B}$ не минимальны. В минимальном варианте признаки равенства r_i можно заменить более простыми функциями $d_j = a_j \vee \bar{b}_j$. Однако для построения компаратора с тремя выходами ("равно", "больше" и "меньше") полученный нами вариант остается предпочтительным, поскольку функции r_i все равно нужны для сравнения на "равно", и для операций сравнения на "больше" они могут быть взяты в готовом виде.

Пример реализации компаратора

Пример реализации компаратора с тремя выходами для двухразрядных слов приведен на рис. 2.25. Выработка признака $A > B$ в этой схеме производится по соотношению (штрихом отмечены функции с выходов младшей группы)

$$F_{A>B} = a_i\bar{b}_i \vee r_i a_{i-1}\bar{b}_{i-1} \vee F'_{A>B} r_i r_{i-1} = \overline{\overline{a_i\bar{b}_i} \cdot \overline{a_{i-1}\bar{b}_{i-1}r_i} \cdot \overline{F'_{A>B} r_i r_{i-1}}}.$$

Компараторы для слов большой разрядности получают включением нескольких ИС последовательно по тракту сигналов A <, A =, A >, т. е. каскадированием идентичных схем.

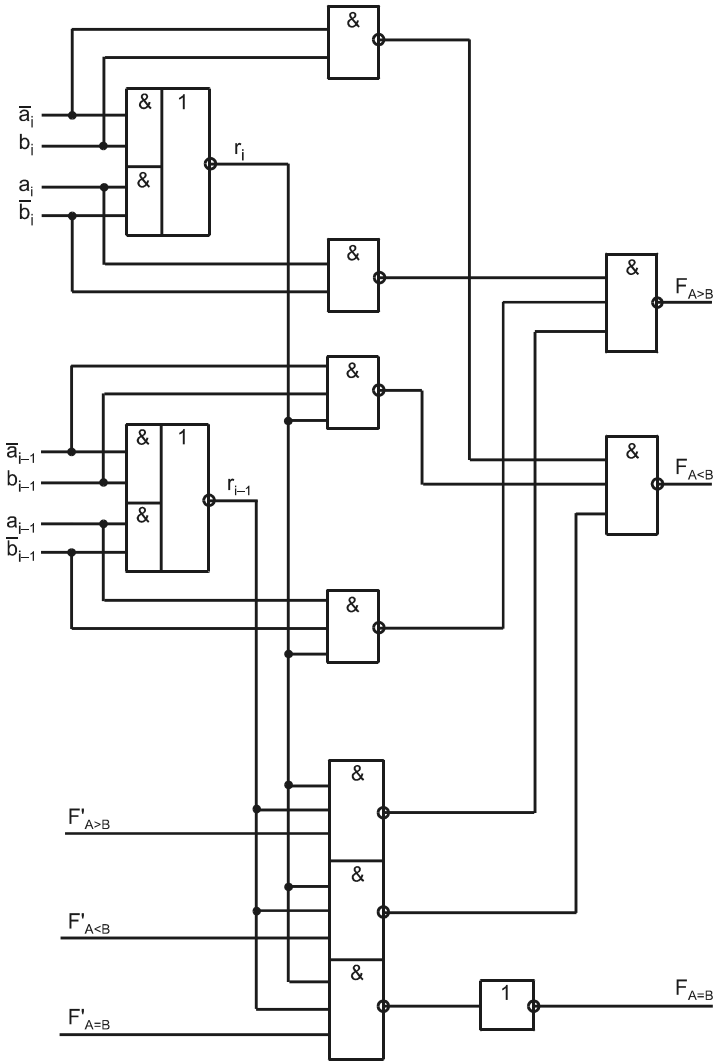


Рис. 2.25. Пример построения компаратора

§ 2.7. Схемы контроля

Сложность ЭВМ и многих других ЦУ определяет важность операций контроля и диагностики их функционирования. В некоторых случаях контроль жизненно важен (авиационные приборы, управление мощными энергетическими установками, мониторинг пациентов в клиниках и др.).

Причинами нарушения нормальной работы ЦУ могут быть отказы (т. е. нарушения из-за возникших неисправностей, имеющих постоянный характер) и сбои (т. е. нарушения из-за проявлений неблагоприятных факторов, в частности, помех, которые в дальнейшем могут и не проявиться). Когда для рассматриваемых

вопросов конкретный характер ошибок несущественен, будем говорить об ошибках функционирования вообще.

Цели и задачи контроля

Цели и задачи контроля, диагностики и исправления ошибок в ЦУ могут быть разными.

Можно ставить задачу *предотвращения ошибок* в работе ЦУ. Для этого необходимы такие меры, как применение высококачественных элементов схем, стабилизация условий окружающей среды и т. п. Но даже при всех стараниях вряд ли возможно полностью избавиться от ошибок.

Имея в виду неизбежность возникновения ошибок, следует позаботиться об их выявлении. Задачи *выявления ошибок* решаются разными методами. Можно, например, воспользоваться дублированием ЦУ и сравнением результатов работы двух идентичных устройств. Несовпадение результатов в этом случае рассматривается как признак ошибки (хотя вероятность того, что ошибка появилась в контролируемом устройстве, а не в контролирующем, равна всего 50%). Метод дублирования универсален, но дорог, для него избыточность составляет около 100%.

Для выявления ошибок широко используются специальные коды, более сложные, чем двоичные.

И, наконец, можно ставить задачи *маскирования (исправления)* ошибок. В этом случае наличие ошибок определенного типа и количества не нарушает работу устройства, поскольку их влияние устраняется автоматически. В этой области используется, например, троекратное резервирование устройств с выработкой результата путем "голосования" с помощью мажоритарных элементов. Эти элементы вырабатывают выходные данные "по большинству" входных. Если из трех устройств одно стало работать неправильно, это не скажется на результате. Только ошибка в двух из трех каналов проявляется в результате. Для исправления ошибок широко используются специальные корректирующие коды.

Отметим, что добавление к функциям устройств функций контроля всегда связано с *избыточностью* — платой за новые возможности будут дополнительные аппаратные или временные затраты. Вводимая избыточность — это цена контроля.

В этом параграфе затронуты темы, связанные с пониманием работы ИС, выпускаемых для использования в системах контроля. К таким схемам относятся мажоритарные элементы, схемы контроля по модулю 2 и схемы кодирования-декодирования для кодов Хемминга.

Мажоритарные элементы

Задача мажоритарного элемента — произвести среди входных величин "голосование" и передать на выход величину, соответствующую большинству из входных.

Ясно, что мажоритарный элемент может иметь только нечетное число входов. Практически выпускаемые элементы имеют по три или по пять входов. Функционирование мажоритарного элемента, на входы которого поступают величины F_1, F_2, и F_3 и по результатам голосования вырабатывается выходная величина F, представлено в табл. 2.6. Если имеется в виду контроль многоразрядных слов, то в каждом разряде ставится элемент рассматриваемого типа.

Таблица 2.6

F_1	F_2	F_3	F	a_1	a_0
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	1	0	0

Кроме выхода F, в таблице приведены и выходы a_1, a_0 — разряды кода, указывающего номер отказавшего канала (рис. 2.26б).

Из таблицы легко получить функции, которые после несложных преобразований приводятся к следующим:

$$F = F_1F_2 \vee F_1F_3 \vee F_2F_3, \quad a_1 = F_2 \oplus F_3, \quad a_0 = F_1 \oplus F_3.$$

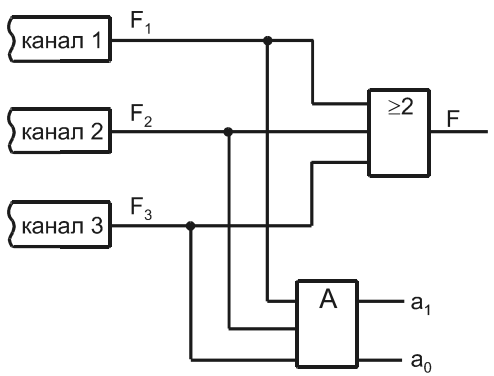


Рис. 2.26. Схема голосования с мажоритарным элементом

В схемах типа рис. 2.26 от мажоритарного элемента требуется особенно высокая надежность, т. к. его отказ делает бесполезной всю схему резервирования.

Контроль по модулю 2

Контроль правильности передач и хранения данных — важное условие нормальной работы ЦУ. В этой области простейшим и широко применяемым методом является контроль по модулю 2.

Определим некоторые понятия, связанные с помехоустойчивыми кодами. *Кодовая комбинация* — набор из символов принятого алфавита. *Код* — совокупность кодовых комбинаций, используемых для отображения информации. *Кодовое расстояние* между двумя кодовыми комбинациями — число разрядов, в которых эти комбинации отличаются друг от друга. *Минимальное кодовое расстояние* — минимальное кодовое расстояние для любой пары комбинаций, входящих в данный код. *Кратность ошибки* — число ошибок в данном слове (число неверных разрядов). *Вес кодовой комбинации* — число единиц в данной комбинации.

Из теории кодирования известны условия обнаружения и исправления ошибок при использовании кодов:

$$d_{min} = r_{обн} + 1;\ d_{min} = 2r_{испр} + 1;\ d_{min} = 2r_{испр} + r_{обн} + 1,$$

где d_{min} — минимальное кодовое расстояние кода; $r_{обн}$ и $r_{испр}$ — кратность обнаруживаемых и исправляемых ошибок соответственно.

Для двоичного кода минимальное кодовое расстояние $d_{min} = 1$, поэтому он не обладает возможностями какого-либо контроля производимых над ним действий. Чтобы получить возможность обнаруживать хотя бы ошибки единичной кратности, нужно увеличить минимальное кодовое расстояние на 1. Это и сделано для кода контроля по модулю 2 (контроля по четности/нечетности).

При этом способе контроля *каждое слово дополняется контрольным разрядом, значение которого подбирается так, чтобы сделать четным (нечетным) вес каждой кодовой комбинации*. При одиночной ошибке в кодовой комбинации четность (нечетность) ее веса меняется, а такая комбинация не принадлежит к данному коду, что и обнаруживается схемами контроля. При двойной ошибке четность (нечетность) комбинации не нарушается — такая ошибка не обнаруживается. Легко видеть, что у кода с контрольным разрядом $d_{min} = 2$. Хотя обнаруживаются ошибки не только единичной, но вообще нечетной кратности, на величину d_{min} это не влияет.

При контроле по четности вес кодовых комбинаций делают четным, при контроле по нечетности — нечетным. Логические возможности обоих вариантов абсолютно идентичны. В зависимости от технической реализации каналов передачи данных, может проявиться предпочтительность того или иного варианта, поскольку один из

вариантов может позволить отличать обрыв всех линий связи от передачи нулевого слова, а другой — нет.

Значения контрольного разряда ρ при контроле по четности ($\rho_ч$) и нечетности ($\rho_н$) приведены для четырехразрядного информационного слова в табл. 2.7.

Таблица 2.7

a_3	a_2	a_1	a_0	$\rho_ч$	$\rho_н$
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
…	…	…	…	…	…
1	1	1	1	0	1

Как видно из таблицы, $\rho_ч = a_3 \oplus a_2 \oplus a_1 \oplus a_0$; $\rho_Н = \overline{a_3 \oplus a_2 \oplus a_1 \oplus a_0}$.

После передачи слова или считывания его из памяти вновь производится сложение разрядов кодовой комбинации по модулю 2 (свертка по модулю 2) и проверяется, сохранилась ли четность (нечетность) веса принятой комбинации. Если четность (нечетность) веса комбинации изменилась, фиксируется ошибка операции.

Из приведенных сведений следует, что контроль по модулю 2 эффективен там, где вероятность единичной ошибки много больше, чем вероятность двойной (или вообще групповой).

В частности, для полупроводниковой основной памяти компьютеров такая ситуация справедлива, т. к. каждый бит слова хранится в своей собственной ячейке, и наиболее вероятны единичные ошибки. А для памяти на магнитных носителях информации (диски, ленты) дефекты таковы, что обычно затрагивают площадь, на которой размещено несколько бит данных, поэтому для этой памяти контроль по модулю 2 неэффективен.

Схемы свертки

Контроль по модулю 2 реализуется с помощью схем свертки. Схема пирамидального типа для свертки байта показана на рис. 2.27, *а*.

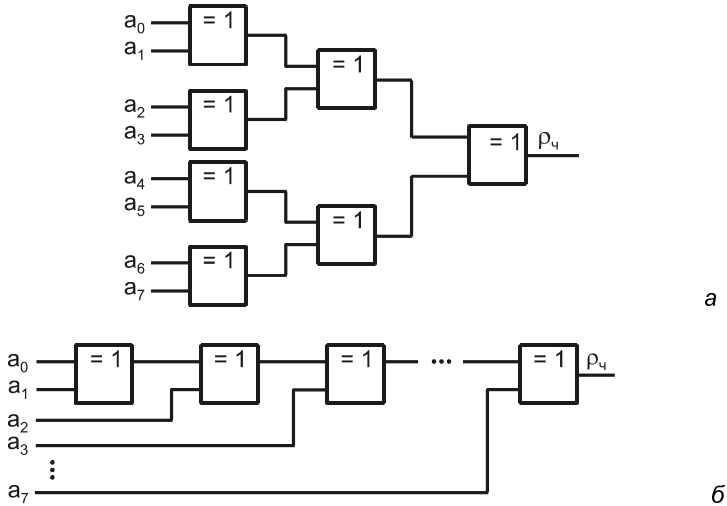
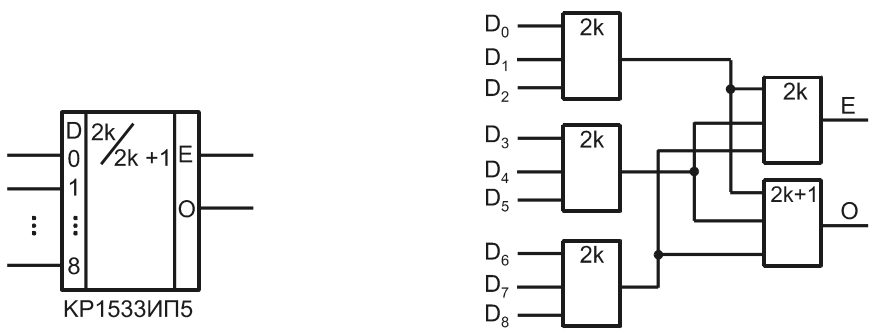


Рис. 2.27. Схемы свертки пирамидального (*а*) и последовательного (*б*) типов



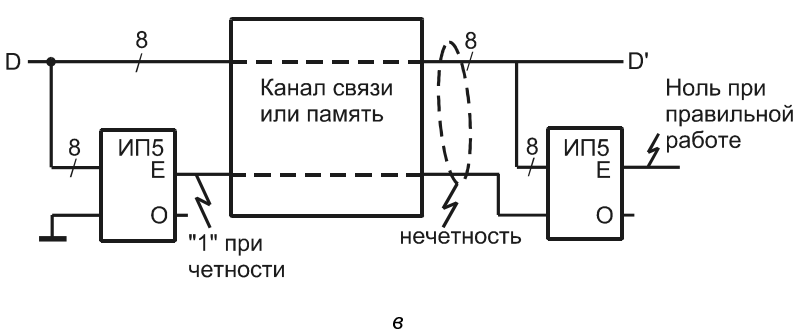


Рис. 2.28. Микросхема ИП5 (*а*, *б*) и ее применение в схеме контроля (*в*)

Для оценки аппаратной сложности и быстродействия подобных схем при разрядности свертываемого слова 2^n (n — произвольное целое число) легко получить соотношения:

$$N_{лэ} = n/2 + n/4 +...+ n/n = n(1/2 + 1/4 +...+ 1/n) = n - 1;\ L = \log_2 n,$$

где $N_{лэ}$ — число логических элементов в схеме; L — ее логическая глубина.

Для последовательных данных, когда слова передаются по одной линии последовательно разряд за разрядом, целесообразно применять схему свертки (рис. 2.27, *б*), которая выдает результат всего через одну задержку после поступления последнего разряда a_7.

Примером ИС свертки по модулю 2 может служить микросхема ИП5 серии КР1533 (рис. 2.28, *а*), представляющая собой пирамидальную структуру из трехвходовых элементов типа четность/нечетность (рис. 2.28, *б*). Схема имеет 9 входов, что допускает свертку байта с девятым контрольным разрядом. Двумя выходами схемы являются E (Even) и O (Odd). Если вес входной комбинации четный, то E = 1 и O = 0, и наоборот, если вес нечетный.

Передача данных с контролем по модулю 2

Передача данных или их запись/считывание (если речь идет о памяти) с контролем по модулю 2 показаны на рис. 2.28, *в*. Входные данные обозначены через D, на выходе из канала связи или памяти данные обозначены через D', поскольку вследствие ошибок они могут измениться.

Контроль по модулю 2 применим не только для операций передачи и хранения слов, но и для некоторых более сложных операций. В этих случаях недостаточно просто добавить к информационному слову контрольный разряд, а требуются более развитые операции.

Контроль логического преобразователя

На рис. 2.29 показан пример контроля логического преобразователя ЛП, воспроизводящего систему переключательных функций от m переменных. Для осуществления контроля к системе добавляется еще одна функция $F_{доп} = F_1 \oplus F_2 \oplus ... \oplus F_n$, которая воспроизводится на индивидуальных элементах (во избежание маскирования контролируемых ошибок). Затем выработанные функции $F_1...F_n$ свертываются по модулю 2, и результат сравнивается с дополнительной функцией $F_{доп}$. Ясно, что при отсутствии ошибок должны сравниваться одинаковые величины. Если они различны, то на выходе элемента сложения по модулю 2 возникнет сигнал ошибки.

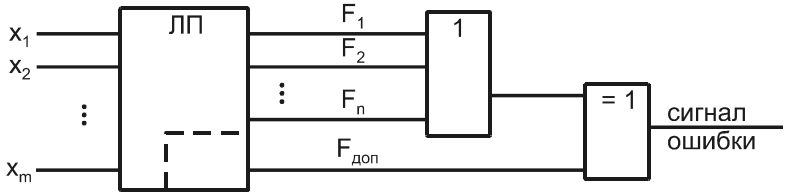


Рис. 2.29. Схема контроля логического преобразователя по модулю 2

Ряд операций контролируется в условиях, когда контрольный разряд не постоянен, а изменяется по определенному закону.

Это возможно, если установлена закономерность изменения контрольного разряда при выполнении операции. Например, при работе счетчика его содержимое меняется по известному закону. Если к слову, содержащемуся в счетчике, добавлять контрольный разряд, также изменяющийся по известному закону, то свертка содержимого счетчика вместе с контрольным разрядом покажет единичную ошибку в работе счетчика. Такой же подход возможен для контроля сумматоров.

Контроль с использованием кодов Хемминга

Применение кодов Хемминга позволяет *исправлять единичные ошибки.* Добавление к коду Хемминга контрольного разряда, обеспечивающего четность/нечетность всей кодовой комбинации в целом, приводит к модифицированному коду Хемминга, с помощью которого можно исправлять единичные ошибки и обнаруживать двойные.

Методы контроля с помощью кодов Хемминга основаны на тех же идеях, что и контроль по модулю 2. Отсюда и область эффективного применения кодов Хемминга — устройства, в которых вероятность единичных ошибок много больше, чем вероятность групповых.

Для получения кодовой комбинации кода Хемминга *к информационному слову добавляется несколько контрольных разрядов.* Для простоты просмотра кодовых комбинаций с целью определения значений контрольных разрядов примем, что контрольные разряды занимают позиции с номерами 2^i (i = 0, 1, 2, ...).

Каждый контрольный разряд ассоциируется с некоторой группой разрядов кодовой комбинации и выводит вес группы, в которую он входит, на четность/нечетность.

Первый контрольный разряд входит в группу разрядов с номерами XX...XX1, где X означает произвольное значение. Иными словами в первую группу входят разряды с нечетными номерами: 1, 3, 5, 7, 9, ...

Второй контрольный разряд входит в группу разрядов с номерами, имеющими единицу во втором справа разряде, т. е. номерами XX...X1X. Это номера 2, 3, 6, 7, 10, 11, ...

Третий контрольный разряд входит в группу, у которой номера разрядов имеют единицу в третьем справа разряде: XX...1XX, т. е. с номерами 4, 5, 6, 7, 12, 13, 14, 15, ...

Таблица 2.8

	8	**7**	**6**	**5**	**4**	**3**	**2**	**1**
	ρ	a_3	a_2	a_1	ρ_3	a_0	ρ_2	ρ_1
	0	0	0	0	0	0	0	0
	1	0	0	0	0	1	1	1
	1	0	0	1	1	0	0	1
	0	0	0	1	1	1	1	0
	1	0	1	0	1	0	1	0
	0	0	1	0	1	1	0	1
$a_3a_2a_1a_0 = 0110$	0	0	1	1	0	0	1	1
	1	0	1	1	0	1	0	0
	0	1	0	0	1	0	1	0
.....	....	....	....	....	....	....	....	....
	1	1	1	1	1	1	1	1

Контрольные разряды выводят веса своих групп на четность/нечетность. Далее для определенности примем, что контроль осуществляется по четности. *После выполнения операции (например, считывания кодовой комбинации из памяти) производится столько проверок по модулю 2, сколько контрольных разрядов в кодовой комбинации*, т. е. проверяется сохранение четности весов групп. Если в кодовой комбинации произошла ошибка, то в одних проверках она скажется, а в других — нет. Это и позволяет определить разряд, в котором произошла ошибка. Для восстановления правильного значения слова теперь остается только проинвертировать ошибочный разряд. Такова идея построения и использования кода Хемминга.

Пример составления кода Хемминга для четырехразрядного информационного слова $A = a_3a_2a_1a_0$ приведен в табл. 2.8.

Через ρ в таблице обозначен общий контрольный разряд для всей кодовой комбинации, через ρ_1, ρ_2, ρ_3 — первый, второй и третий групповые контрольные разряды.

Для коротких слов избыточность кода Хемминга получилась значительной (здесь на четыре информационных разряда приходится четыре контрольных), но это нетипично, поскольку реально контролируются слова большей разрядности, для которых избыточность (относительная) быстро уменьшается с ростом разрядности слов. Короткое слово взято, чтобы пример не был громоздким.

Рассмотрим теперь процесс исправления и выявления ошибок. Пусть, например, передавалось информационное слово $0110 = 6_{10}$. Не учитывая пока разряд ρ, получим, что правильная кодовая комбинация имеет вид:

7	6	5	4	3	2	1
0	1	1	0	0	1	1

Пусть во втором слева разряде произошла ошибка и принята комбинация:

7	6	5	4	3	2	1
0	0	1	0	0	1	1

Первая проверка (по группе разрядов с нечетными номерами) показывает сохранение четности, т. е. в этой группе ошибок нет, результат этой проверки отмечается нулем.

Вторая проверка (по разрядам 2, 3, 6, 7) обнаруживает нарушение четности веса комбинации, ее результат отмечается единицей.

Третья проверка (по разрядам 4, 5, 6, 7) также обнаруживает нарушение четности, ее результат отмечается единицей.

Результаты проверок образуют слово, называемое синдромом. Синдром указывает номер разряда, в котором произошла ошибка. Во взятом примере результаты проверок дают слово $110 = 6_{10}$. Проинвертировав разряд номер 6, возвращаемся к правильной кодовой комбинации — ошибка исправлена.

Минимальное кодовое расстояние обычного кода Хемминга равно трем. Добавление разряда проверки общей четности веса комбинации приводит к *модифицированному коду Хемминга* с минимальным кодовым расстоянием, равным 4, и, соответственно, добавляет возможность обнаружения двойной ошибки. Обнаружение двойной ошибки основано на сопоставлении наличия или отсутствия признаков ошибки в синдроме и общей четности. Если обозначить через S любое ненулевое значение синдрома, то возможные ситуации, используемые для обнаружения двойной ошибки, окажутся следующими (табл. 2.9).

Таблица 2.9

Синдром	Свертка кодовой комбинации	Характеристика результата
0	0	Все правильно, слово можно использовать
S	1	Была единичная ошибка, исправлена, слово можно использовать

Таблица 2.9 *(окончание)*

Синдром	Свертка кодовой комбинации	Характеристика результата
S	0	Эти ситуации могут возникать только вследствие ошибок двойной или большей кратности, слово использовать нельзя
0	1	

Схемы кодера и декодера для кода Хемминга

На рис. 2.30 показана схема кодирования и декодирования для кодов Хемминга. Верхняя часть схемы показывает выработку контрольных разрядов для составления кода Хемминга.

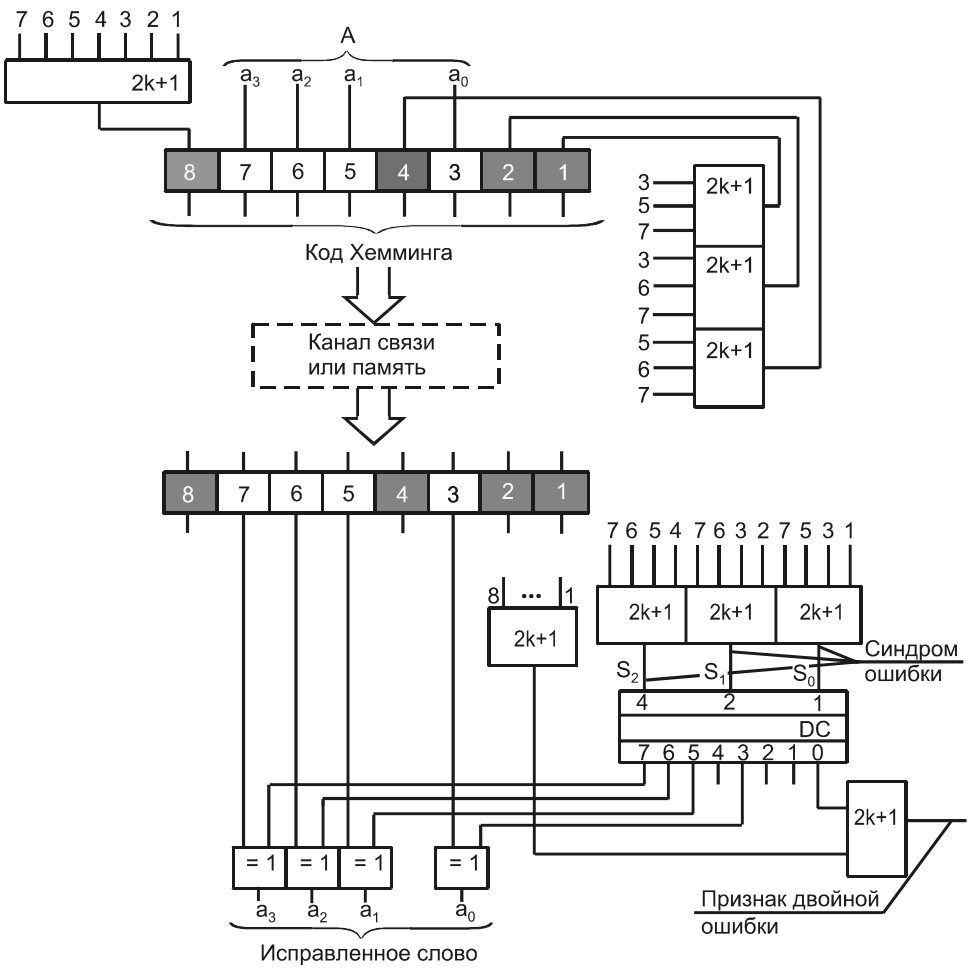


Рис. 2.30. Схема кодирования и декодирования для кодов Хемминга

Нижняя часть содержит три четырехразрядных схемы свертки для проведения групповых проверок (разрядов синдрома). Синдром поступает на дешифратор, который вырабатывает единичный сигнал на линии, соответствующей номеру ошибочного разряда. Эта единица выполняет инвертирование ошибочного разряда слова А, поступая на второй вход элемента сложения по модулю 2, через который данный разряд передается на выход схемы. Таким образом, нижняя часть схемы представляет собой декодирующее устройство для кода Хемминга.

Двойная ошибка обнаруживается элементом 2k согласно логике ситуаций, указанной выше.

Кодирование-декодирование для 16-разрядных слов с формированием 6 контрольных разрядов модифицированного кода Хемминга реализуется микросхемой ВЖ1 серий К555, 533. Время кодирования-декодирования составляет для этой ИС 50...60 нс.

Код Хемминга принадлежит к так называемым цикличным кодам и в рамках этого типа кодов относится к числу простых. Есть много более сложных кодов с большими корректирующими возможностями.

§ 2.8. Сумматоры

Сумматоры выполняют *арифметическое* сложение чисел. Они являются также ядром арифметико-логических устройств (АЛУ), входящих в состав процессоров.

Аппаратная сложность и быстродействие сумматора — важные параметры, и поэтому разработано множество вариантов сумматоров с различными соотношениями сложности и быстродействия. Рассмотрим следующие типы сумматоров:

- одноразрядный;
- многоразрядный для последовательных операндов;
- многоразрядные для параллельных операндов:
  - с последовательным переносом;
  - с параллельным переносом;
  - с групповой структурой;
  - с условным переносом;
  - накапливающий.

Наряду с сумматорами могут быть реализованы *вычитатели*, однако это почти никогда не делается, поскольку вычитание удобнее выполняется через сложение с применением дополнительных либо обратных кодов.

Одноразрядный сумматор

Одноразрядный сумматор имеет три входа (два слагаемых и перенос из предыдущего разряда) и два выхода (суммы и переноса в следующий разряд). Таблица истинности одноразрядного сумматора имеет вид, представленный в табл. 2.10.

Таблица 2.10

a_i	b_i	c_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Аналитические выражения функций суммы и переноса (перенос обозначен через С от английского Carry) таковы:

$$S_i = \bar{a}_i \bar{b}_i c_{i-1} \vee \bar{a}_i b_i \bar{c}_{i-1} \vee a_i \bar{b}_i \bar{c}_{i-1} \vee a_i b_i c_{i-1},$$

$$C_i = a_i b_i \vee a_i c_{i-1} \vee b_i c_{i-1}.$$

В базисе Шеффера функции S_i и C_i выражаются следующим образом:

$$S_i = \overline{\overline{\bar{a}_i \bar{b}_i c_{i-1}} \cdot \overline{\bar{a}_i b_i \bar{c}_{i-1}} \cdot \overline{a_i \bar{b}_i \bar{c}_{i-1}} \cdot \overline{a_i b_i c_{i-1}}},$$

$$C_i = \overline{\overline{a_i b_i} \cdot \overline{a_i c_{i-1}} \cdot \overline{b_i c_{i-1}}}.$$

Непосредственное воспроизведение полученных формул на элементах двухступенчатой логики И-ИЛИ-НЕ приводит к применению элемента 2-2-2И-ИЛИ-НЕ для выработки переноса $\bar{C}_i$ и элемента 3-3-3-3И-ИЛИ-НЕ для выработки суммы $\bar{S}_i$. Такое решение используется, но более популярно другое, приводящее к некоторому сокращению аппаратной сложности схемы при сохранении минимальной задержки по цепи переноса. Идея этого решения состоит в использовании уже полученного значения $\bar{C}_i$ в качестве вспомогательного аргумента при вычислении $\bar{S}_i$.

Из табл. 2.10 видно, что во всех строчках, кроме первой и последней, $S_i = \bar{C}_i$. Чтобы сделать эту формулу справедливой также в первой и последней строчках, нужно убрать единицу в строчке нулевых входных величин и добавить единицу в строчку единичных входных величин, что приводит к соотношению

$$S_i = \bar{C}_i (a_i \vee b_i \vee c_{i-1}) \vee a_i b_i c_{i-1}.$$

Схема сумматора, построенного по этому соотношению, показана на рис. 2.31, *а*.

Из табл. 2.10 видно, что функции суммы и переноса самодвойственны: при инвертировании всех аргументов инвертируется и значение функции, т. е.

$$S(\bar{X}) = \bar{S}(X) \text{ и } C(\bar{X}) = \bar{C}(X).$$

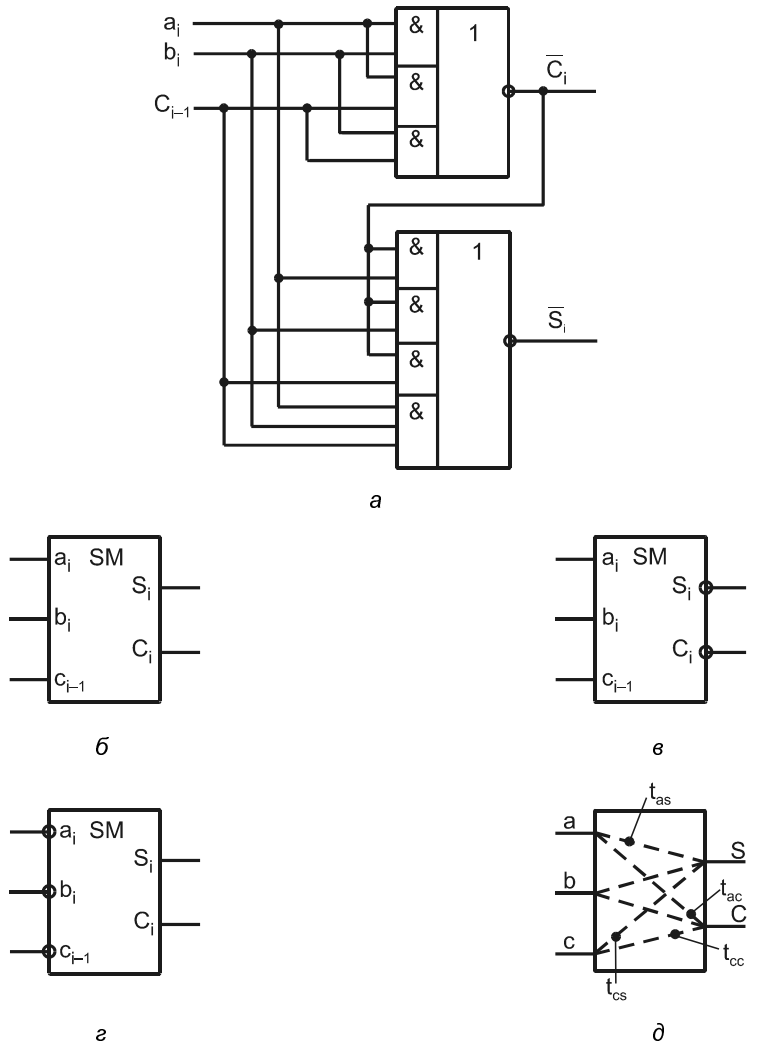


Рис. 2.31. Схема (*а*), условные обозначения (*б*, *в*, *г*) и пути распространения сигналов одноразрядного сумматора (*д*)

Условное обозначение одноразрядного сумматора показано на рис. 2.31, *б*. Для варианта с выработкой инвертированных значений суммы и переноса на основании свойства самодвойственности можно пользоваться двумя вариантами обозначений для одной и той же схемы (рис. 2.31, *в*, *г*).

Быстродействие одноразрядного сумматора оценивается задержками по шести трактам распространения сигналов: от первого слагаемого до выходов суммы и переноса, от второго слагаемого до тех же выходов и от входа переноса до выходов переноса и суммы (рис. 2.31, *д*). Так как тракты от обоих слагаемых обычно одинаковы, достаточно знать четыре задержки, отмеченные на рис. 2.31, *д* надписями t_{as}, t_{ac}, t_{cc} и t_{cs}.

На рис. 2.32 показана схема сумматора, входящая в одну из библиотек для микросхем программируемой логики.

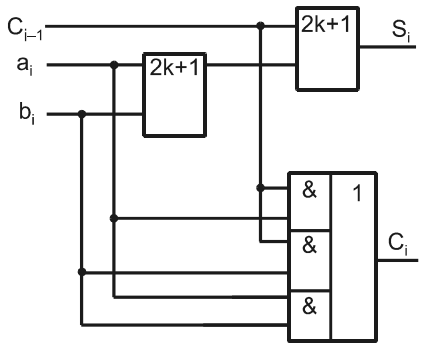
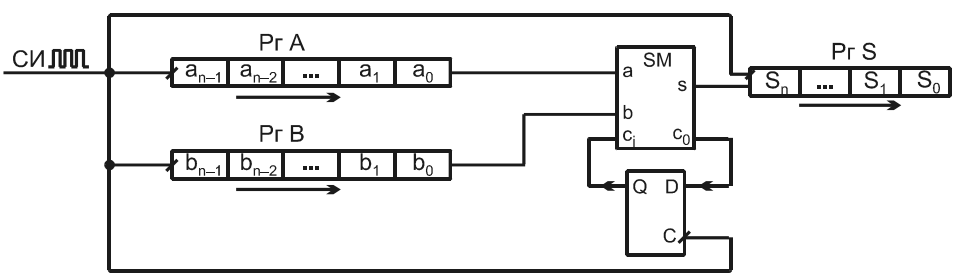


Рис. 2.32. Схема одноразрядного сумматора из библиотеки схемных решений для микросхем программируемой логики

Сумматор для последовательных операндов

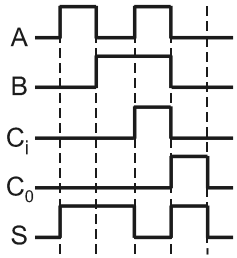
В сумматоре для последовательных операндов единственный одноразрядный сумматор, обрабатывает поочередно разряд за разрядом, начиная с младшего. Сложив младшие разряды a_0 и b_0, сумматор вырабатывает сумму S_0 для младшего разряда результата и перенос, который запоминается D-триггером на один такт. В следующем такте складываются вновь поступившие разряды слагаемых a_1 и b_1 с переносом c_0 из младшего разряда и получается следующий разряд суммы S_1 и перенос из него и т. д. Схема сумматора (рис. 2.33, *а*), помимо одноразрядного сумматора SM, содержит сдвигающие регистры слагаемых и суммы, а также триггер запоминания переноса. Регистры и триггер тактируются синхроимпульсами СИ.

На рис. 2.33, *б* показана временная диаграмма, соответствующая операции сложения двух операндов 0101 + 0110 = 1011 или в десятичном выражении 5 + 6 = 11.



а

Рис. 2.33. Схема сумматора для последовательных операндов (*а*)



б

Рис. 2.33. Временная диаграмма
для суммирования последовательных операндов (*б*)

Сумматор параллельных операндов с последовательным переносом

Сумматор параллельных операндов с последовательным переносом строится как цепочка одноразрядных сумматоров, соединенных последовательно по цепям переноса (рис. 2.34, *а*). Для схемы с одноразрядными сумматорами, вырабатывающими инверсии суммы и переноса, такая цепочка имеет вид, приведенный на рис. 2.34, б, поскольку функции суммы и переноса самодвойственны. Там, где в разряд сумматора должны подаваться инверсные аргументы, в их линиях имеются инверторы, а там, где вырабатывается инверсная сумма, инвертор включен в выходную цепь. Важно, что инверторы не входят в цепь передачи переноса — они при этом не замедляют работу сумматора в целом.

Длительность суммирования для этой схемы в наихудшем случае распространения переноса по всей цепочке разрядов составит

$$t_{SM} = t_{ac} + (n - 2)t_{cc} + t_{cs},$$

где n — разрядность сумматора.

Как и в других схемах с последовательным распространением сигналов от разряда к разряду, здесь время суммирования при достаточно больших n практически пропорционально разрядности сумматора.

Если одноразрядные сумматоры выполнены по схеме (см. рис. 2.31, *а*), то время суммирования составит

$$t_{SM} = (n + 1)t_{ЛР},$$

где $t_{ЛР}$ — задержка элемента И-ИЛИ-НЕ, обозначенная индексом ЛР, поскольку именно эти буквы входят в маркировку элементов данного типа. Если одноразрядные сумматоры выполнены по схеме (см. рис. 2.32), то $t_{SM} = nt_{ЛР}$.

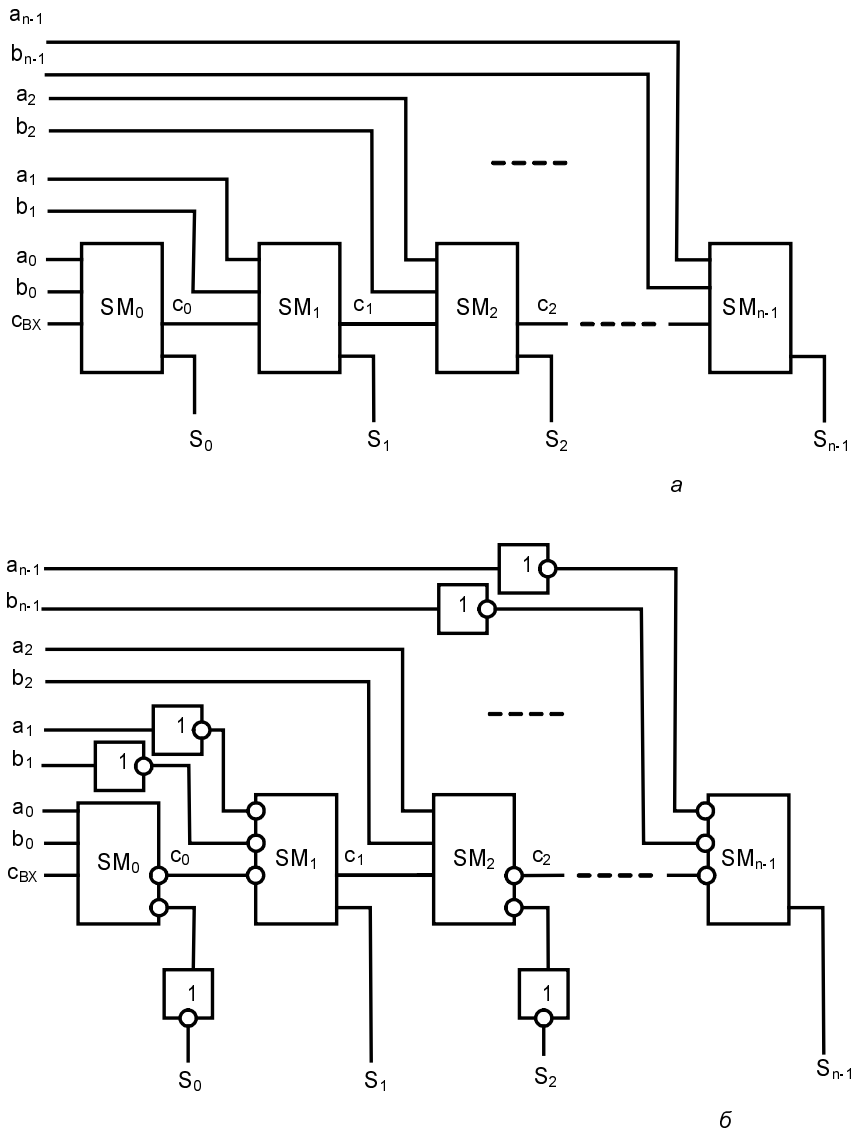


Рис. 2.34. Схемы сумматоров с последовательным переносом

Сумматор с передачей сигнала переноса по цепочке замкнутых ключей

В первом приближении этот вариант сумматора оценивается как обладающий простотой сумматора с последовательным переносом при быстродействии, близком к быстродействию сумматора с параллельным переносом [64].

Предложенный подход предусматривает вычисление сигналов переноса в два этапа. На первом этапе одновременно для всех разрядов задаются условия вычисления переносов, на втором происходит простая передача информации по образованной на первом этапе цепи.

При вычислении переноса Ci возможны два случая: ai = bi или ai ≠ bi. В первом случае перенос от предыдущего разряда не имеет значения (не влияет на перенос из данного разряда) и его можно исключить из рассмотрения. Действительно, при ai = bi = 0 перенос Ci = 0, а при ai = bi = 1 перенос Ci = 1. Следовательно, в качестве переноса может использоваться значение любого операнда (ai или bi). Примем, что Ci = ai. Во втором случае сигнал переноса из данного разряда совпадает с сигналом переноса в данный разряд, т. е. Ci = Ci–1.

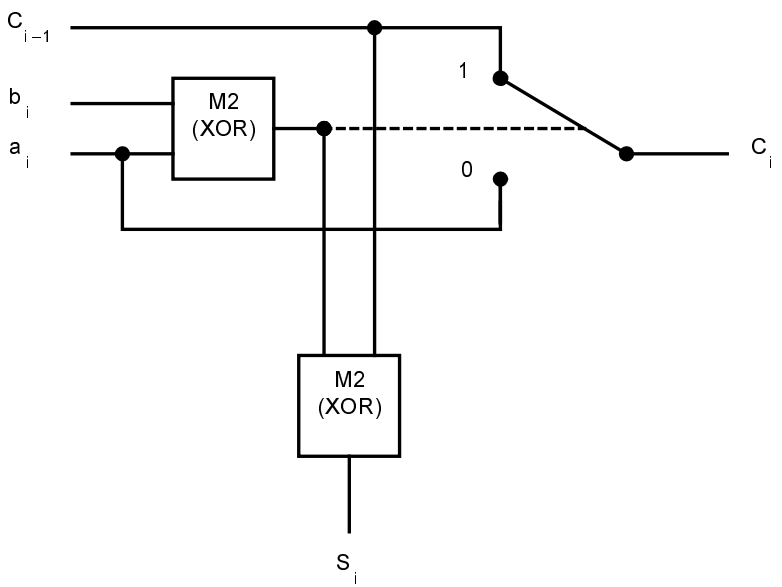


Рис. 2.35. Схема разряда сумматора с передачей сигнала переноса по цепочке замкнутых ключей

В обоих случаях после установления факта равенства или неравенства разрядов операндов происходит только передача информации по цепи, заготовленной на предыдущем этапе, что может происходить достаточно быстро.

Разряд сумматора (рис. 2.35) содержит ключ, управляемый от элемента сложения операндов ai и bi по модулю 2, выявляющего равенство или неравенство этих операндов. В зависимости от выходного сигнала этого элемента переносу присваиваются значения ai или Ci. Этот же элемент используется для вычисления совместно со вторым элементом сложения по модулю 2 значения суммы данного разряда по формуле: Si = ai ⊕ bi ⊕ Ci-1 (обозначение XOR (от eXclusive OR) — "исключающее ИЛИ" — является синонимом обозначения ⊕).

Автор публикации [64] проверил работу сумматора в схемотехнике ТТЛ. Результат оказался эффективным. Для сумматоров с 2, 4 и 8 разрядами время сложения практически не зависело от разрядности. По скорости четырехразрядный сумматор выиграл у традиционного варианта с последовательным переносом в 2 раза, а восьмиразрядный в 3,5 раза. Эффективность метода для сумматора на элементах КМОП, для которых параметры ключей отличаются от параметров ТТЛ-ключей, автором данной публикации не рассмотрена.

Сумматор параллельных операндов с параллельным переносом

Сумматоры для параллельных операндов с параллельным переносом разработаны для получения максимального быстродействия.

> Этот тезис требует пояснений. Дело в том, что рассматриваемые сумматоры — комбинационные схемы и вырабатываемые ими функции могут быть представлены в нормальных формах, например в ДНФ, что приводит к двухъярусной реализации при наличии парафазных аргументов и к трехъярусной при однофазных аргументах. Таким образом, предельное быстродействие оценивается (2...3) элементарными задержками. Однако построение многоразрядных сумматоров на основе нормальных форм дало бы неприемлемо громоздкие схемы. Реальные схемы имеют модульную структуру, т. е. состоят из подсхем (разрядных схем), что резко упрощает их, но не дает предельно возможного быстродействия. Именно в рамках указанных ограничений рассматриваемый здесь сумматор в определенном диапазоне разрядностей операндов обладает максимальным быстродействием.

Сумматоры с параллельным переносом не имеют последовательного распространения переноса вдоль разрядной сетки. *Во всех разрядах результаты вырабатываются одновременно*, параллельно во времени. Сигналы переноса для данного разряда формируются специальными схемами, на входы которых поступают все переменные, необходимые для выработки переноса, т. е. те, от которых зависит его наличие или отсутствие. Ясно, что это внешний входной перенос $C_{вх}$ (если он есть) и значения всех разрядов слагаемых, младших относительно данного. От одноразрядных сумматоров, имеющихся в схеме, выход переноса не требуется, достаточно одного выхода суммы (рис. 2.36).

Для перехода от идеи построения схемы к ее конкретному виду удобно ввести две вспомогательные функции: генерации и прозрачности.

Функция генерации принимает единичное значение, если перенос на выходе данного разряда появляется независимо от наличия или отсутствия входного переноса. Очевидно, что эта функция $g_i = a_i b_i$.

Функция прозрачности (транзита) принимает единичное значение, если перенос на выходе данного разряда появляется только при наличии входного переноса. Строго говоря, функция прозрачности должна выражаться как "исключающее ИЛИ", т. е. $h_i = a_i\bar{b}_i \vee \bar{a}_i b_i$, но допустимо заменить эту функцию на дизъюнкцию и принять, что

$h_i = a_i \vee b_i$. Действительно, разница между функциями ИЛИ и "исключающее ИЛИ" проявляется только при $a_i = b_i = 1$, т. е. в ситуации, где перенос все равно формируется из-за $g_i = 1$, так что эта разница не играет роли.

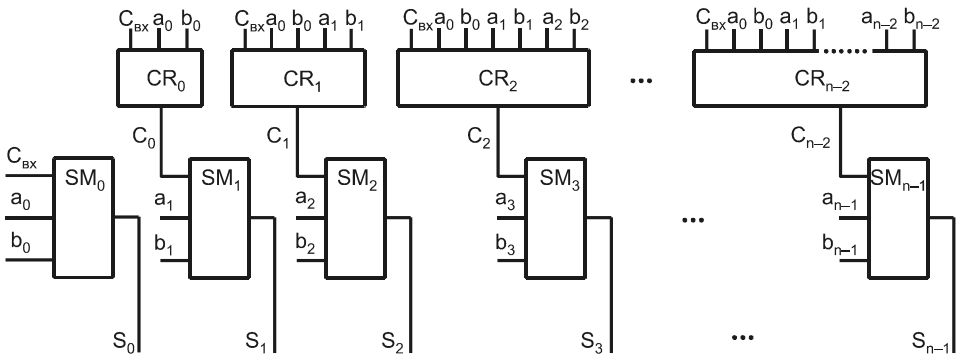


Рис. 2.36. Структура сумматора с параллельным переносом

Теперь выражение для сигнала переноса можно записать в виде $C_i = g_i \vee h_i C_{i-1}$. Словесно полученное соотношение можно выразить так: перенос на выходе разряда либо генерируется им, либо проходит от младшего разряда при прозрачности данного.

На основе полученного выведем функции переноса C для нулевого, первого и второго разрядов с последующим их обобщением. Перенос на выходе младшего разряда

$$C_0 = g_0 \vee C_{вх} h_0,$$

согласно чему он либо генерируется самим разрядом ($g_0 = 1$), либо пропускается через него ($h_0 = 1$ и $C_{вх} = 1$).

Аналогичным образом для переноса C_1 на выходе следующего разряда справедливо соотношение:

$$C_1 = g_1 \vee C_0 h_1.$$

Подставив в это соотношение выражение для C_0, получим

$$C_1 = g_1 \vee g_0 h_1 \vee C_{вх} h_1 h_0.$$

Для следующего разряда произведем те же действия

$$C_2 = g_2 \vee C_1 h_2 = g_2 \vee g_1 h_2 \vee g_0 h_2 h_1 \vee C_{вх} h_2 h_1 h_0.$$

Выведенные формулы имеют ясный физический смысл — перенос на выходе разряда сгенерируется в нем или придет от предыдущих разрядов при прозрачности тех, через которые он распространяется.

Для произвольного разряда с номером i можно записать

$$C_i = g_i \vee g_{i-1} h_i \vee g_{i-2} h_i h_{i-1} \vee \ldots \vee g_0 h_i h_{i-1} \ldots h_1 \vee C_{вх} h_i h_{i-1} \ldots h_0.$$

Функции переноса имеют нормальную дизъюнктивную форму и могут быть реализованы элементами двухступенчатой логики (И-ИЛИ либо И-ИЛИ-НЕ) однако у них для построения многоразрядного сумматора обычно недостаточно входов по И. Большее число входов имеют элементы И-НЕ (у стандартных элементов до восьми входов). Перевод полученных выражений в базис И-НЕ с одновременным раскрытием функций генерации дает выражения

$$C_0 = \overline{\overline{g_0} \cdot \overline{C_{вх} h_0}} = \overline{\overline{a_0 b_0} \cdot \overline{C_{вх} h_0}},$$

$$C_1 = \overline{\overline{a_1 b_1} \cdot \overline{a_0 b_0 h_1} \cdot \overline{C_{вх} h_1 h_0}},$$

$$C_2 = \overline{\overline{a_2 b_2} \cdot \overline{a_1 b_1 h_2} \cdot \overline{a_0 b_0 h_2 h_1} \cdot \overline{C_{вх} h_2 h_1 h_0}}.$$

Схема сумматора (рис. 2.37) соответствует полученным выражениям. Время суммирования складывается из времен формирования функций прозрачности (одна задержка $t_{ЛА}$ элемента И-НЕ), функций переноса ($2t_{ЛА}$) и задержки упрощенных одноразрядных сумматоров ($t_{ЛР}$), что в результате дает $t_{SM} = (4...5)\, t_{ЛА}$.

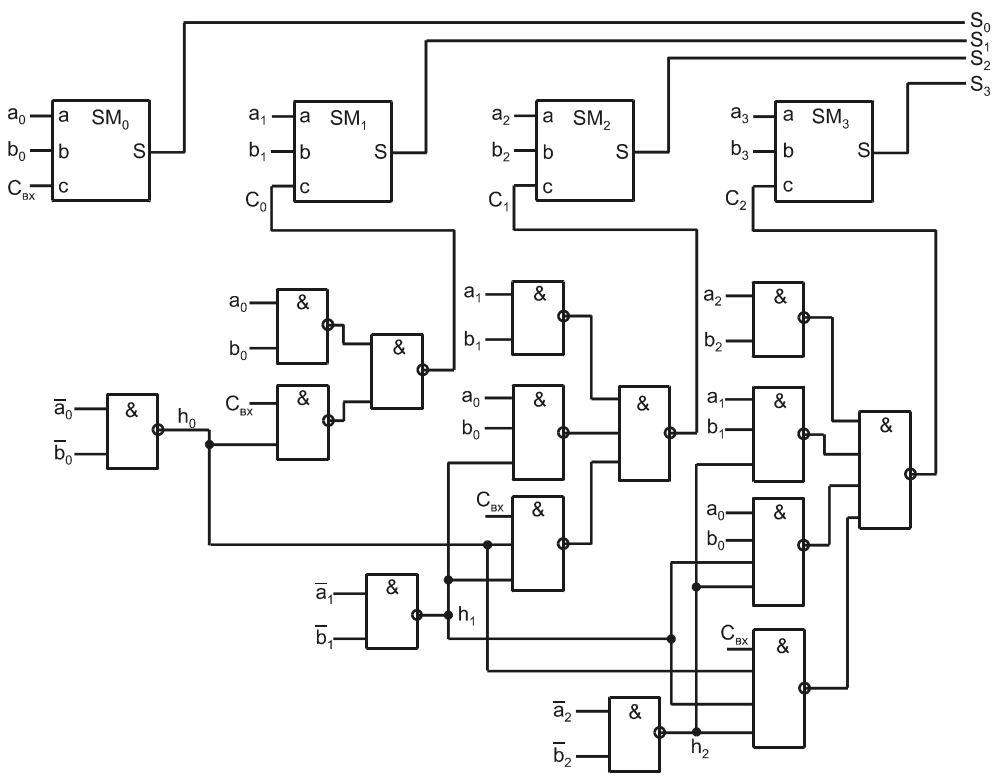


Рис. 2.37. Вариант схемы сумматора с параллельным переносом

Длительность суммирования, полученная из рассмотрения логической схемы, не зависит от разрядности сумматора, что является характерным для структур с параллельными переносами. Однако фактически это не совсем так, поскольку с ростом разрядности сумматора *увеличивается нагрузка элементов схемы, что*

увеличивает их задержки. В частности, коэффициент разветвления элементов, вырабатывающих функции прозрачности, равен $n^2/4$, т. е. квадратично зависит от разрядности сумматора. Поэтому *рост разрядности замедляет процесс суммирования.*

Диапазон разрядностей, в которых проявляются достоинства сумматоров с параллельным переносом, невелик. До n = 3...4 преимущества имеют более простые схемы сумматоров с последовательным переносом, после n = 8 появляются перегруженные элементы и элементы с большим числом входов, что замедляет работу сумматора, требует введения развязывающих элементов с их задержками и т. п. Для суммирования многоразрядных операндов целесообразно перейти к групповым структурам.

Сумматоры групповой структуры

В сумматорах групповой структуры схема с разрядностью n делится на ℓ групп по m разрядов (n = ℓm). В группах и между ними возможны различные виды переносов, что порождает варианты групповых сумматоров.

Сумматор с цепным переносом. Групповой сумматор с цепным переносом при ℓ группах имеет ℓ – 1 блок переноса. Блоки переноса включены последовательно и образуют тракт передачи переноса (рис. 2.38). Слагаемые разбиты на m-разрядные поля, суммируемые в группах. Результат также составляется из m-разрядных полей.

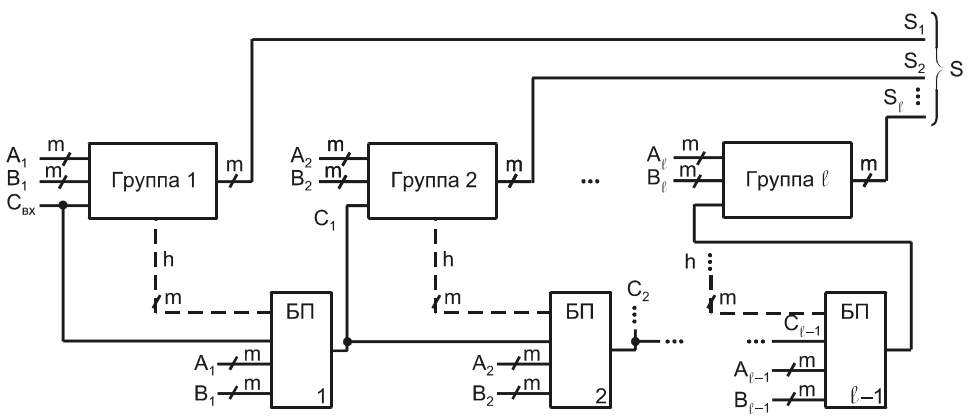


Рис. 2.38. Схема группового сумматора с цепным переносом

Блоки переноса $БП_i$ (i = 1, 2, ... , ℓ–1) анализируют слагаемые в пределах группы, и если из группы должен быть перенос, то он появляется на выходе блока для подачи на вход следующей группы и в цепочку распространения переноса от младших групп к старшим.

Переносы определяются формулами, полученными выше для сумматоров с параллельным переносом, но сумматоры благодаря делению на группы существенно упрощаются — у них все $БП_i$ имеют одинаковую сложность (все блоки анализируют

m-разрядные операнды), тогда как в сумматоре с параллельными переносами сложность схем переноса непрерывно возрастает при переходе от предыдущего разряда к последующему (последняя схема переноса требует анализа операндов с разрядностью n – 1).

Максимальная длительность суммирования для варианта с цепным переносом $t_{SM} = (\ell - 1)\, t_{БП} + t_{ГР}$.

Функции прозрачности разрядов h, необходимые для блоков переносов, вырабатываются в этих блоках либо берутся из групп, если в них организован параллельный перенос (штриховые линии на рис. 2.38). Имея в виду реализацию блоков переноса и групп, показанную ранее для базиса И-НЕ, формулу для времени суммирования можно представить в виде:

$$t_{SM} = t_h + (\ell - 1)\, 2t_{ЛА} + (4...5)t_{ЛА} = (2\ell - 1)\, t_{ЛА} + (4...5)\, t_{ЛА}.$$

Для сумматора 16-разрядных слов, в частности, при его разбиении на 4 группы получим $t_{SM} = (11...12)\, t_{ЛА}$.

Сумматор с параллельными межгрупповыми переносами

Такой сумматор строится по структуре, сходной со структурой сумматора с параллельными межразрядными переносами, но роль одноразрядных сумматоров играют группы, которые также характеризуются функциями генерации и прозрачности. Обозначив эти функции большими буквами, можно записать соотношение:

$$H = h_{m-1}h_{m-2} \ldots h_1,$$

согласно которому группа прозрачна при прозрачности всех ее разрядов, и выражение

$$G_{гр} = g_m \vee g_{m-1}h_m \vee g_{m-2}h_m h_{m-1} \vee \ldots \vee g_1 h_m h_{m-1} \ldots h_2,$$

справедливость которого видна из определения функции генерации.

Из групп так же, как ранее из одноразрядных сумматоров, собирается схема с выработкой на выходе группы с номером i параллельных межгрупповых переносов согласно выражению:

$$C_i = G_i \vee G_{i-1}H_i \vee G_{i-2}H_{i-1}H_i \vee \ldots \vee G_1 H_2 \ldots H_i \vee C_{вх}H_1 H_2 \ldots H_i.$$

Аппаратная сложность сумматоров с параллельными межгрупповыми переносами выше, чем сложность предыдущего варианта, но при больших разрядностях они дают преимущества по быстродействию.

Время суммирования для схемы, принятой нами в качестве примера,

$$t_{SM} = t_h + t_G + t_C + t_{гр} = 10t_{ла}.$$

Групповой сумматор с параллельным межгрупповым переносом можно без существенных трудностей построить и для достаточно большой разрядности, хотя схемы выработки переноса и усложняются с ростом i. Если число разрядов очень велико, можно распространить способ организации параллельных переносов и на

схему с тремя уровнями, считая групповой сумматор с двумя уровнями как бы новой группой и организуя параллельный перенос между новыми группами. Структура группового сумматора с параллельными межгрупповыми переносами достаточно сложна, но находила применение в больших ЭВМ прошлых поколений. Групповые сумматоры разных типов описаны в работе [36].

Сумматор с условным переносом

Сумматор с условным переносом (рис. 2.39) — давно известная структура, которая со временем вышла из широкого применения, но в современных условиях вновь привлекает внимание разработчиков. Эта структура улучшает быстродействие сумматоров с последовательным переносом. Реализация в современных микросхемах цепей переносов с малыми задержками возродила интерес к структурам с последовательным переносом и, соответственно, к методам улучшения их быстродействия.

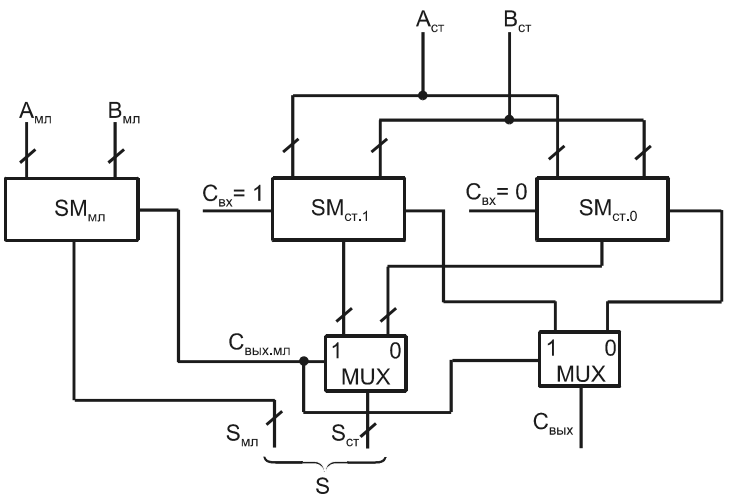


Рис. 2.39. Схема сумматора с условным переносом

Идею построения сумматора с условным переносом рассмотрим на примере сумматора с двумя группами разрядов. В этом случае сумматор с n разрядами делят на две равные группы с разрядностями n/2. Старшую группу дублируют, так что в схему входят три сумматора с разрядностью n/2. На одном суммируются младшие поля операндов $A_{мл}$ и $B_{мл}$. На втором — старшие поля операндов при условии $C_{вх} = 1$, на третьем — старшие поля операндов при условии $C_{вх} = 0$. После получения результата в младшем сумматоре становится известным фактическое значение переноса в старший сумматор, и из двух заготовленных заранее результатов выбирается тот, который нужен в данном случае. Цепь последовательного переноса здесь как бы укорачивается вдвое, т. к. обе половины сумматора работают параллельно во времени (см. рис. 2.39).

Накапливающий сумматор

Накапливающий сумматор представляет собой сочетание комбинационного сумматора и регистра, работающее по формуле S := S + A, согласно которой к содержимому сумматора добавляется очередное слагаемое, и результат замещает старое значение суммы. Структура накапливающего сумматора показана на рис. 2.40. Очередное прибавление слагаемого тактируется синхроимпульсами СИ. Учитывая особенности функционирования, накапливающие сумматоры называют иногда *аккумуляторами*.

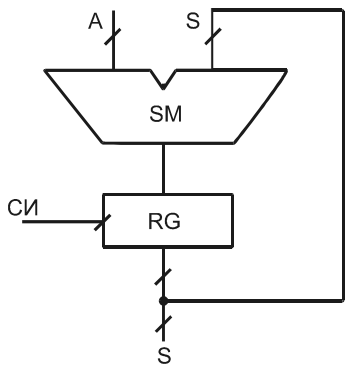


Рис. 2.40. Структура накапливающего сумматора

Микросхемы сумматоров

В сериях элементов имеются одноразрядные сумматоры, в том числе с дополнительной входной логикой, а также двух- и четырехразрядные сумматоры. Примером могут служить микросхемы, содержащие четырехразрядный сумматор с последовательным переносом и блок переноса (рис. 2.41), непосредственно пригодные для построения группового сумматора с цепным переносом.

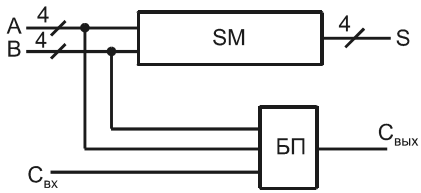


Рис. 2.41. Структура микросхемы ИМ3

Микросхемы четырехразрядных сумматоров можно объединять и в групповую структуру с межгрупповым параллельным переносом с помощью специальных блоков ускоренного переноса (см. *§ 2.9*). В некоторых сериях элементов сумматоры отсутствуют. Причиной этого обычно является наличие арифметико-логического устройства, для которого режим суммирования есть один из рабочих режимов.

§ 2.9. Арифметико-логические устройства и блоки ускоренного переноса

Арифметико-логические устройства, или сокращенно АЛУ (ALU, Arithmetic-Logic Unit), выполняют над словами ряд действий. Основой АЛУ служит сумматор, схема которого дополнена логикой, расширяющей функциональные возможности АЛУ и обеспечивающей его перестройку с одной операции на другую.

Для наращивания разрядности АЛУ объединяются в вариантах с последовательными или параллельными переносами. Логические возможности АЛУ разных технологий (ТТЛШ, КМОП) идентичны. В силу самодвойственности выполняемых операций условное обозначение и таблица истинности АЛУ встречаются в двух вариантах, отличающихся взаимно инверсными значениями переменных.

АЛУ (рис. 2.42) имеет входы операндов A и B, входы выбора операций S, вход переноса $\overline{C}_i$ и вход M (Mode), сигнал которого задает тип выполняемых операций: логические (M = 1) или арифметико-логические (M = 0). Результат операции вырабатывается на выходах F, выходы G и H дают функции генерации и прозрачности, используемые для организаций параллельных переносов при наращивании размерности АЛУ. Сигнал $\overline{C}_0$ — выходной перенос, а выход A = B есть выход сравнения на равенство с открытым коллектором.

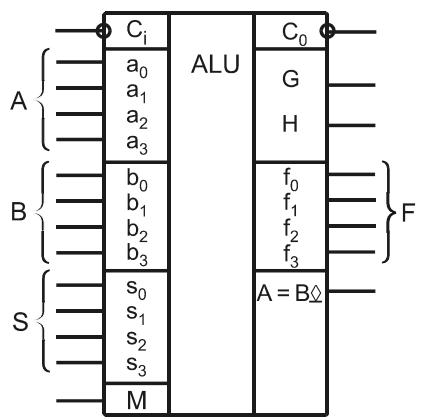


Рис. 2.42. Условное обозначение АЛУ

Перечень выполняемых АЛУ операций дан в табл. 2.11. Двоичные числа $s_3s_2s_1s_0$ представлены их десятичными эквивалентами. Под утолщенными цифрами 1 и 0 следует понимать наборы 1111 и 0000, входной перенос поступает в младший разряд слова, т. е. равен $000C_i$. Логические операции поразрядные, т. е. операция над словами A * B означает, что a_i * b_i при отсутствии взаимовлияния разрядов. При арифметических операциях учитываются межразрядные переносы.

Таблица 2.11

S	Логические функции (M = 1)	Арифметико-логические функции (M = 0)
0	$\overline{A}$	$A + C_i$
1	$\overline{A \vee B}$	$A \vee B + C_i$
2	$\overline{A}B$	$A \vee \overline{B} + C_i$
3	**0**	**1** + C_i
4	$\overline{AB}$	$A + A\overline{B} + C_i$
5	$\overline{B}$	$A \vee B + A\overline{B} + C_i$
6	$A \oplus B$	$A + \overline{B} + C_i$
7	$A\overline{B}$	$A\overline{B} + \mathbf{1} + C_i$
8	$\overline{A} \vee B$	$A + AB + C_i$
9	$\overline{A \oplus B}$	$A + B + C_i$
10	B	$A \vee \overline{B} + AB + C_i$
11	AB	$AB + \mathbf{1} + C_i$
12	**1**	$A + A + C_i$
13	$A \vee \overline{B}$	$A \vee B + A + C_i$
14	$A \vee B$	$A \vee \overline{B} + A + C_i$
15	A	$A + \mathbf{1} + C_i$

Шестнадцать логических операций позволяют воспроизводить все функции двух переменных. В логико-арифметических операциях встречаются и логические и арифметические операции одновременно.

Запись типа $A \vee \overline{B} + AB$ следует понимать так: вначале поразрядно выполняются операции инвертирования ($\overline{B}$), логического сложения ($A \vee \overline{B}$) и умножения (AB), а затем полученные указанным образом два четырехразрядных числа складываются арифметически. Наличие в таблице некоторых экзотических операций, мало связанных с потребностями практики, объясняется тем, что они получаются "бесплатно", т. е. реализуются при данном сочетании сигналов схемой, которая строилась для воспроизведения набора нужных операций.

При операциях над словами большой размерности АЛУ соединяются друг с другом с организацией последовательных (рис. 2.43, *а*) или параллельных (рис. 2.43, *б*) переносов. В последнем случае совместно с АЛУ применяют блоки ускоренного переноса (CRU, Carry Unit), получающие от отдельных АЛУ функции генерации и прозрачности, а также входной перенос и вырабатывающие сигналы переноса по формулам, приведенным в предыдущем параграфе.

Блок ускоренного переноса CRU вырабатывает также функции генерации и прозрачности G и H для всей группы обслуживаемых им АЛУ, что при необходимости

позволяет организовать параллельный перенос на следующем уровне (между несколькими группами из четырех АЛУ каждая).

На рис. 2.43, *в* показаны способы выработки сигналов сравнения слов для группы АЛУ. Выход сравнения на равенство выполняется по схеме монтажной логики И для выходов типа ОК. Комбинируя сигнал равенства слов с сигналом переноса или заема на выходе группы при работе АЛУ в режиме вычитания, легко получить функции $F_{A\geq B}$ и $F_{A\leq B}$. Если A < B, то при вычитании возникает L-активный заем из старшего разряда и $F_{A\leq B} = 1$. Если заем отсутствует (A > B), то получим $F_{A\geq B} = 1$.

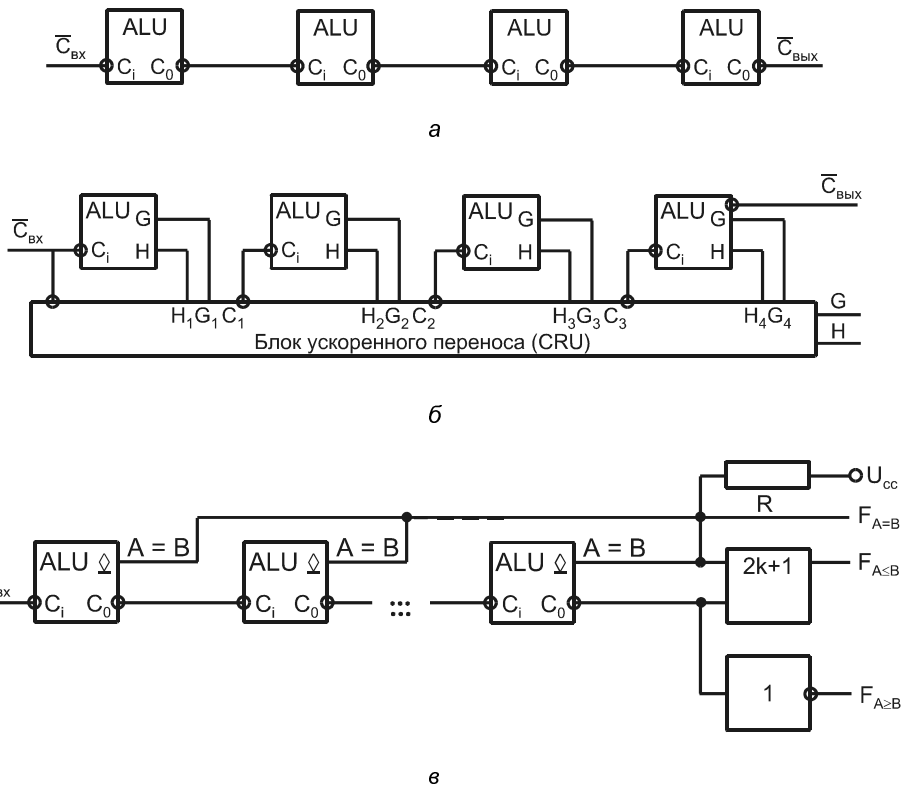


Рис. 2.43. Схемы наращивания АЛУ при последовательном (*а*) и параллельном (*б*) переносах и реализация функций компаратора для группы АЛУ (*в*)

§ 2.10. Матричные умножители

Микросхемы множительных устройств появились в 1980-х годах, когда достигнутый уровень интеграции позволил разместить на одном кристалле достаточно большое количество логических элементов. Структура матричных умножителей тесно связана со структурой математических выражений, описывающих операцию

умножения. Пусть имеются два целых двоичных числа без знаков $A_m = a_{m-1} \ldots a_0$ и $B_n = b_{n-1} \ldots b_0$. Их перемножение выполняется по известной схеме "умножения столбиком". Если числа четырехразрядные, т. е. $m = n = 4$, то восьмиразрядное произведение образуется следующим образом:

				a_3	a_2	a_1	a_0
				b_3	b_2	b_1	b_0
				a_3b_0	a_2b_0	a_1b_0	a_0b_0
+			a_3b_1	a_2b_1	a_1b_1	a_0b_1	
		a_3b_2	a_2b_2	a_1b_2	a_0b_2		
	a_3b_3	a_2b_3	a_1b_3	a_0b_3			
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

В общем случае произведение выражается числом $P_{m+n} = p_{m+n-1}\ p_{m+n-2} \ldots p_0$. Члены вида a_ib_j, где $i = 0 \ldots (m-1)$ и $j = 0 \ldots (n-1)$, вырабатываются параллельно во времени конъюнкторами. Их сложение в столбцах, которое можно выполнять разными способами, определяет почти целиком время перемножения.

Матричные перемножители могут быть просто *множительными блоками* (МБ) или *множительно-суммирующими блоками* (МСБ), последние обеспечивают удобство наращивания размерности умножителя. МСБ реализует операцию $P = A_m \times B_n + C_m + D_n$, т. е. добавляет к произведению два слагаемых: одно разрядности m, совпадающей с разрядностью множимого, другое разрядности n, совпадающей с разрядностью множителя.

Множительно-суммирующие блоки

Множительно-суммирующий блок для четырехразрядных операндов без набора конъюнкторов, вырабатывающих члены вида a_ib_j, показан на рис. 2.44, *а*, где для одноразрядного сумматора принято обозначение (рис. 2.44, *б*).

Для построения МСБ чисел равной разрядности требуется n^2 конъюнкторов и n^2 одноразрядных сумматоров.

Максимальная длительность умножения — сумма задержек сигналов в конъюнкторах для выработки членов a_ib_j и задержки в наиболее длинной цепочке передачи сигнала в матрице одноразрядных сумматоров, равной $2n - 1$ ($m + n - 1$ в случае разной разрядности операндов). Таким образом, $t_{MPL} = t_K + (2n - 1)t_{SM}$.

Схема множительного блока отличается от схемы МСБ тем, что в ней отсутствуют сумматоры правой диагонали, т. к. при $C_m = 0$ и $D_n = 0$ они не требуются.

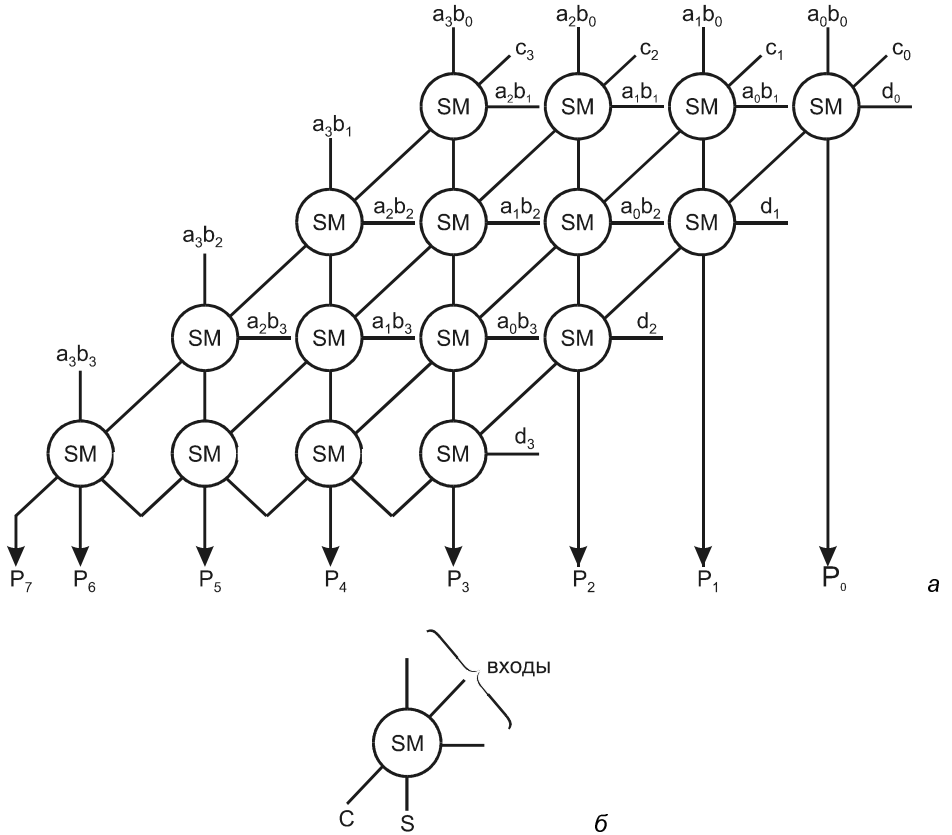


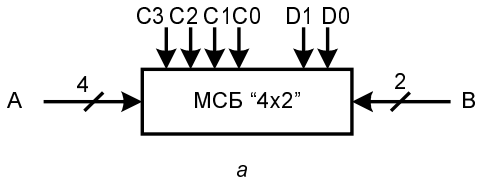
Рис. 2.44. Схема множительно-суммирующего блока для четырехразрядных сомножителей (*а*), обозначение одноразрядного сумматора для данной схемы (*б*)

Наращивание размерности матричных умножителей

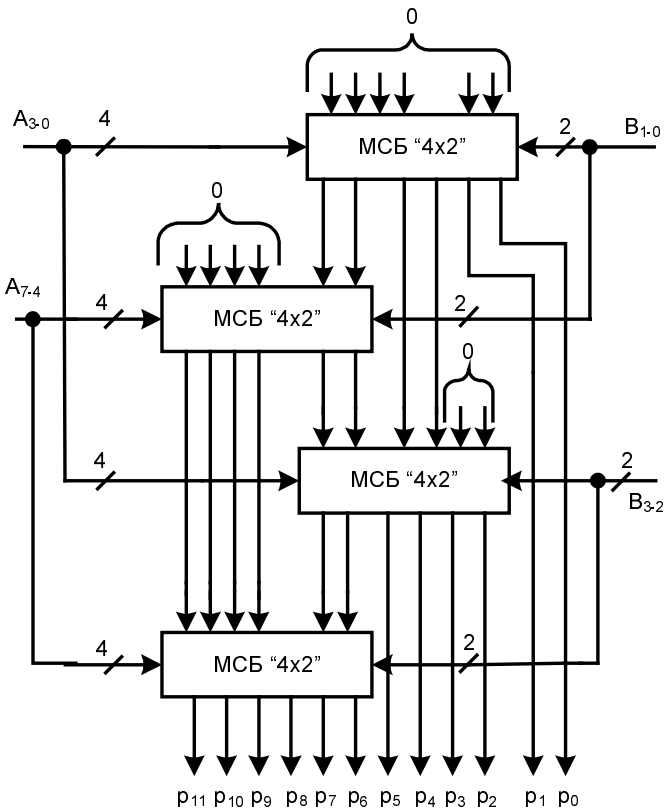
Построение умножителей большей размерности из умножителей меньшей размерности на основе МБ требует введения дополнительных схем, называемых *"деревьями Уоллеса"*, микросхемы которых имеются в некоторых зарубежных сериях. При использовании МСБ дополнительные схемы не требуются. Принцип наращивания размерности умножителя состоит в следующем. При построении умножителя с разрядностями сомножителей n и m из умножителей с разрядностями сомножителей q и r, где $q < n$ и $r < m$, разрядные сетки n и m разбиваются на поля размерами q и r соответственно.

Далее сомножители A_n и B_m представляются в виде сумм полей. Произведение таких сумм дает соотношение, в котором искомый результат представляется суммой нескольких произведений, размерности которых отвечают возможностям имеющихся умножителей. Проиллюстрируем сказанное примером построения с помощью множительно-суммирующих блоков с разрядностями сомножителей 4 и 2, выполняющих

операцию $P_{5\text{-}0} = A_{3\text{-}0} \times B_{1\text{-}0} + C_{3\text{-}0} + D_{1\text{-}0}$, умножителя $P_{11\text{-}0} = A_{7\text{-}0} \times B_{3\text{-}0}$ с разрядностями сомножителей 8 и 4.



а



б

Рис. 2.45. Условное обозначение множительно-суммирующего блока (*а*) и схема умножителя "8 × 4", построенная на множительно-суммирующих блоках "4 × 2" (*б*)

Множители $A_{7\text{-}0}$ и $B_{3\text{-}0}$ следует представить в виде:

$$A_{7\text{-}0} = A_7A_6A_5A_40000 + 0000A_3A_2A_1A_0 = A_{7\text{-}4}2^4 + A_{3\text{-}0},$$

$$B_{3\text{-}0} = B_3B_200 + 00B_1B_0 = B_{3\text{-}2}2^2 + B_{1\text{-}0}.$$

Перемножив полученные соотношения, имеем:

$$P_{11\text{-}0} = A_{7\text{-}0} \times B_{3\text{-}0} = (A_{7\text{-}4} \times B_{3\text{-}2})2^6 + (A_{7\text{-}4} \times B_{1\text{-}0})2^4 + (A_{3\text{-}0} \times B_{3\text{-}2})2^2 + (A_{3\text{-}0} \times B_{1\text{-}0}).$$

Согласно последнему выражению, схема умножителя будет содержать четыре МСБ размерности 4×2, выходы которых занимают в разрядной сетке результата смещенные положения, указываемые коэффициентами 2^i ($i = 6, 4, 2$), и суммируются с учетом этих смещений. Пользуясь условным обозначением МСБ, показанным на рис. 2.45, *а*, схему умножителя можно представить в виде рис. 2.45, *б*. Заметим, что верхний блок МСБ работает как множительный (МБ), и выполнен как множительно-суммирующий только для единообразия всех блоков схемы.

Схемы ускоренного умножения

Для ускорения умножения разработан ряд алгоритмов, большой вклад в их разработку внес Э. Бут (E. Boot). Рассмотрим процесс умножения по так называемому модифицированному алгоритму Бута (*умножение сразу на два разряда*).

Из изложенного ранее видно, что основную задержку в процесс выработки произведения вносит суммирование частичных произведений. Уменьшение их числа сократило бы время суммирования. К этому приводит алгоритм, основанный на следующих рассуждениях.

Пусть требуется вычислить произведение

$$P = A \times B = A \times (b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + ... + b_0 2^0). \qquad (а)$$

Непосредственное воспроизведение соотношения (а) связано с выработкой частичных произведений вида Ab_i2^i ($i = 0 ... n - 1$). Число таких произведений равно разрядности множителя n.

Выражение (а) можно видоизменить с помощью соотношения

$$b_i2^i = b_i2^{i+1} - 2b_i2^{i-1}, \qquad (б)$$

справедливость которого очевидна.

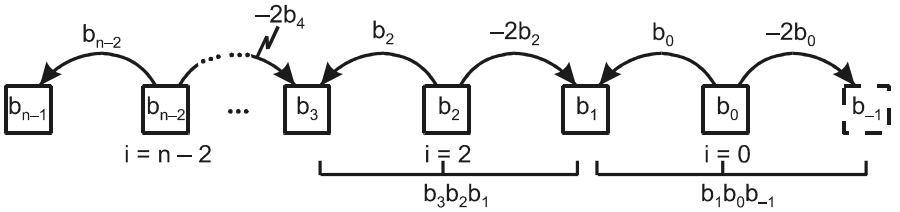
Это соотношение позволяет разреживать последовательность (спектр) степеней в сумме частичных произведений. Можно, например, исключить четные степени, как показано на рис. 2.46, *а*. Исключение четных (или нечетных) степеней не только изменяет значения оставшихся частичных произведений, но и сокращает их число примерно вдвое, что, в конечном счете, ускоряет выработку произведения. Для того чтобы "разнести по соседям" член со степенью 2^0, расширим разрядную сетку, введя слагаемое $b_{-1}2^{-1}$ (нулевой разряд с номером –1).

Оставшиеся частичные произведения имеют вид

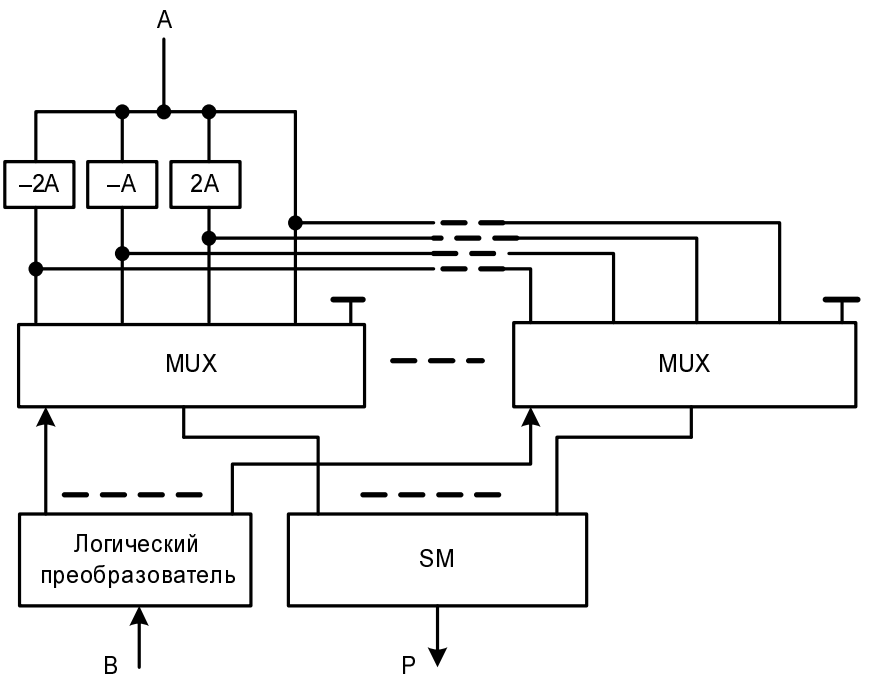
$$R_i = A(-2b_{i+1} + b_i + b_{i-1})2^i.$$

Так как число частичных произведений уменьшилось примерно вдвое, при применении этого алгоритма говорят об умножении сразу на два разряда.

Для всех возможных сочетаний b_{i+1}, b_i, b_{i-1} можно составить таблицу (табл. 2.12) частичных произведений.



а



б

Рис. 2.46. К пояснению принципа быстрого умножения "сразу на два разряда" (*а*) и схема быстрого умножения (*б*)

Таблица 2.12

b_{i+1}	b_i	b_{i-1}	**Значение скобки**	$R_i/2^i$	**Операция для получения $R_i/2^i$**
0	0	0	0	0	Заменить A нулем
0	0	1	1	A	Скопировать A
0	1	0	1	A	Скопировать A

Таблица 2.12 *(окончание)*

b_{i+1}	b_i	b_{i-1}	Значение скобки	$R_i/2^i$	Операция для получения $R_i/2^i$
0	1	1	2	2A	Сдвинуть A влево
1	0	0	–2	–2A	Сдвинуть A влево и преобразовать в дополнительный код
1	0	1	–1	–A	Преобразовать A в дополнительный код
1	1	0	–1	–A	Преобразовать A в дополнительный код
1	1	1	0	0	Заменить A нулем

Пример

Пусть требуется умножить 1010_2 на 0111_2, т. е. 10×7. При разреживании частичных произведений оставим только нечетные, как показано на рис. 2.39, *а*. Расширив разрядную сетку множителя, имеем $B = b_4b_3b_2b_1b_0b_{-1}b_{-2} = 0011100$.

Первому частичному произведению соответствует тройка $b_0b_{-1}b_{-2} = 100$. Из табл. 2.11 получаем, что этой тройке соответствует частичное произведение $-2A\,2^{-1} = -A$, для получения которого требуется перевести A в дополнительный код. Сама величина A в пределах разрядной сетки произведения должна быть записана как 00001010, ее обратный код 11110101 и дополнительный код 11110110.

Второму частичному произведению соответствует тройка $b_2b_1b_0 = 111$, следовательно, второе частичное произведение равно нулю (табл. 2.11).

Третьему частичному произведению соответствует тройка $b_4b_3b_2 = 001$, следовательно, оно имеет вид $A\,2^3 = 01010000$. Для получения результата заданного умножения требуется сложить частичные произведения:

$$\begin{array}{l} 11110110 \\ 01010000 \\ \hline 01000110 = 2^6 + 2^2 + 2^1 = 64 + 4 + 2 = 70. \end{array}$$

Схема, реализующая алгоритм быстрого умножения сразу на два разряда, показана на рис. 2.46, *б*.

Множимое A поступает в этой схеме на ряд преобразователей, заготавливающих все возможные варианты частичных произведений (–2A, –A, 2A), кроме самого A и нуля, которые не требуют схемной реализации. Множитель B поступает на логический преобразователь ЛП, который анализирует тройки разрядов, декодирует их и дает мультиплексорам сигналы выбора того или иного варианта частичных произведений. Окончательный результат получается суммированием частичных произведений с учетом их взаимного сдвига в разрядной сетке. Размерность умножителя "4×4".

Учет знаков сомножителей

Приведенные выше примеры множительных устройств касались операций с прямыми кодами. В этом случае умножение знакопеременных чисел сведется только к выработке знакового разряда как суммы по модулю 2 знаковых разрядов сомножителей. Если же числа представлены не прямыми кодами со знаковыми разрядами, а, например, дополнительными кодами, то, имея рассмотренные выше умножители, можно дополнить их преобразователями дополнительного кода в прямые на входах и преобразователем прямого кода в дополнительный на выходе или использовать схемы, непосредственно реализующие алгоритмы умножения дополнительных кодов (см., например, [47]).

> Разработке матричных умножителей уделяют внимание многие фирмы. В отечественных сериях МИС/СИС имеются умножители малой размерности (2×2, 4×4, 4×2 и др.). В сериях БИС размерности умножителей значительно больше: 8×8, 12×12, 16×16 (в серии 1802), 32×32 (микросхема КА1843ВР1). Зарубежные фирмы разработали много вариантов умножителей. Кроме того, немало умножителей реализовано в качестве библиотечных элементов для систем и устройств, размещаемых на одном кристалле.

§ 2.11. Быстрые сдвигатели

Сдвиги слов в разрядной сетке — типовая операция, используемая при вычислениях с представлением чисел с плавающей точкой, умножениях чисел на константу и в других случаях. Сдвиг на несколько разрядов можно реализовать с помощью сдвигающих регистров как результат нескольких одноразрядных сдвигов, но это занимает много времени. Быстрые сдвигатели перемещают слово сразу на несколько разрядов в соответствии с указанным значением сдвига.

Схемотехника быстрых сдвигателей насчитывает ряд вариантов, из которых основными являются сдвигатели, управляемые кодом "1 из N" (Barrel Shifters), и сдвигатели, управляемые двоичным кодом (Logarithmic Shifters).

Сдвигатель, управляемый кодом "1 из N"

Структура этого варианта показана на рис. 2.47, *б* с обозначением ключевых транзисторов согласно рис. 2.47, *а*. Ключевые транзисторы (Pass transistors) для наглядности представления структуры изображены кружками, причем по прямой линии расположены выводы истока и стока (т. е. в эту линию включен управляемый ключ), а боковой вывод связан с затвором, и на него подается управляющее напряжение. Обозначение управляющего напряжения, замыкающего ключ, записывается в кружке в сокращенном виде. Если ключ замыкается сигналом "Сдвиг N", т. е. при команде сдвига на N разрядов, то в кружке ставится символ N.

Основа схемы сдвигателя — матрица ключевых транзисторов, управляемых кодами "1 из N" согласно табл. 2.13 (сдвиги производятся в сторону разрядов с меньшими номерами).

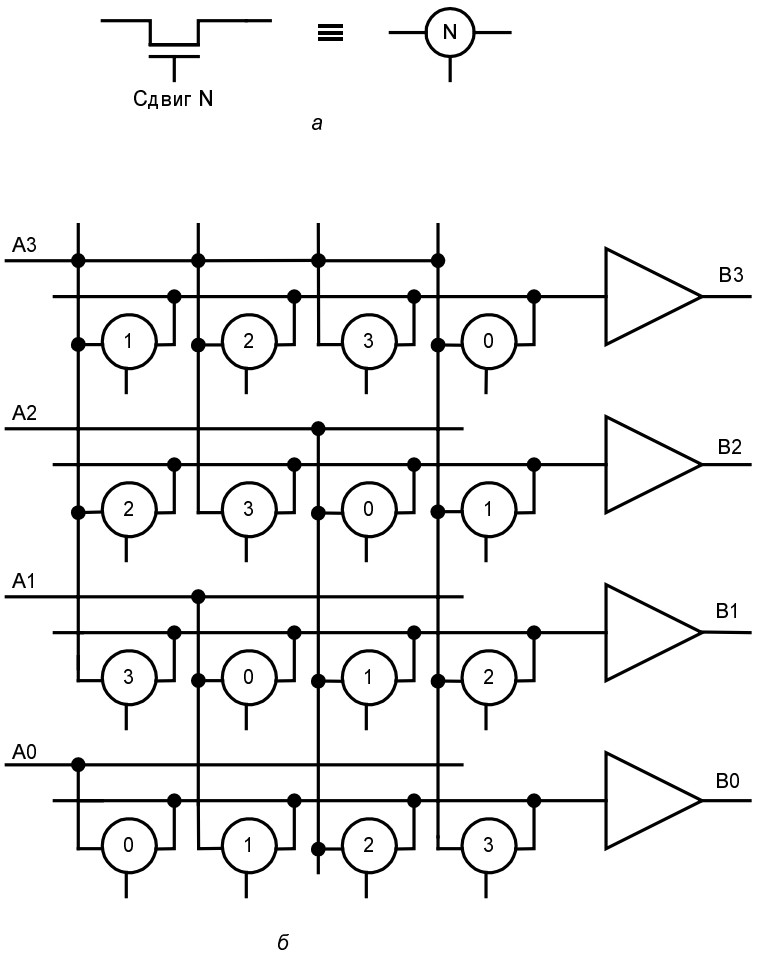


Рис. 2.47. Условное обозначение ключевых транзисторов, принятое при изображении структуры сдвигателя, управляемого кодом "1 из N" (*а*), и структура сдвигателя (*б*)

Таблица 2.13

Операция	**Сигналы управления**			
	Сдвиг 0	**Сдвиг 1**	**Сдвиг 2**	**Сдвиг 3**
Передача без сдвига	1	0	0	0
Сдвиг на 1 разряд	0	1	0	0

Таблица 2.13 *(окончание)*

Операция	**Сигналы управления**			
	Сдвиг 0	**Сдвиг 1**	**Сдвиг 2**	**Сдвиг 3**
Сдвиг на 2 разряда	0	0	1	0
Сдвиг на 3 разряда	0	0	0	1

Требуемый результат получается за счет определенной схемы соединений ключей. В схеме рис. 2.47, *б* предусмотрено автоматическое повторение знакового разряда и на выходы B передаются следующие коды:

- при сдвиге на 0 разрядов $A_3\ A_2\ A_1\ A_0$;
- при сдвиге на 1 разряд $A_3\ A_3\ A_2\ A_1$;
- при сдвиге на 2 разряда $A_3\ A_3\ A_3\ A_2$;
- при сдвиге на 3 разряда $A_3\ A_3\ A_3\ A_3$.

Сдвигатель типа Barrel Shifter отличается высоким быстродействием, т. к. сигнал сдвига замыкает для каждой передачи входного бита на выходной буфер цепь только из одного ключа. Однако это справедливо, если управление ведется кодом "1 из N". В большинстве случаев управляющие коды исходно вырабатываются как двоичные, и перед сдвигателем требуется включать дешифратор, что увеличивает как сложность схемы, так и задержки в ее работе. В этих случаях, особенно при больших величинах сдвигов, преимущество по технико-экономическим показателям обычно оказывается на стороне сдвигателя, непосредственно управляемого двоичными кодами и называемого логарифмическим.

Сдвигатель, управляемый двоичным кодом

В отличие от предыдущего варианта этот сдвигатель в качестве основы имеет не единую матрицу ключевых транзисторов, а схему, состоящую из нескольких каскадов (ступеней). Число ступеней N связано со значением максимального сдвига S зависимостью $N = \log_2 S$, что и объясняет название "логарифмический" применительно к данному типу сдвигателя. Требуемое значение величины сдвига образуется в этом сдвигателе как композиция нескольких сдвигов, каждый из которых равен числу 2^i, где i = 1, 2, 3, … .

Структура логарифмического сдвигателя показана на рис. 2.48 на примере четырехразрядного фрагмента.

Обозначения сигналов управления ключевыми транзисторами совпадает с принятыми на рис. 2.47. Как видно из схемы, сигналы от разрядов входного слова до входов выходных буферов проходят через цепочку последовательно включенных транзисторов, число которых равно $\log_2 S$. Это замедляет процесс сдвига, а поскольку задержка цепочки из RC-звеньев, которая создается последовательно

включенными транзисторами, зависит от их числа сильнее, чем по линейному закону, в длинные цепочки целесообразно вводить промежуточные буферные каскады, разбивая цепочки на более короткие части.

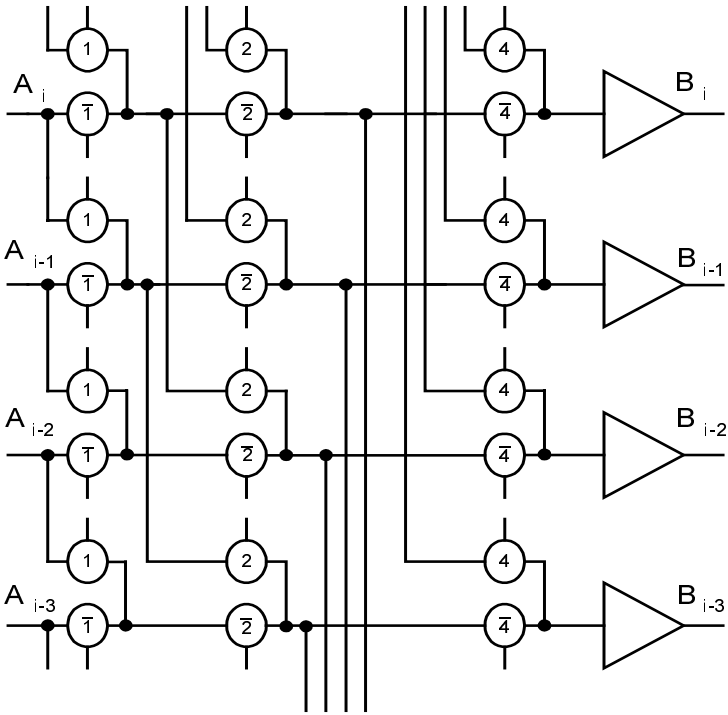


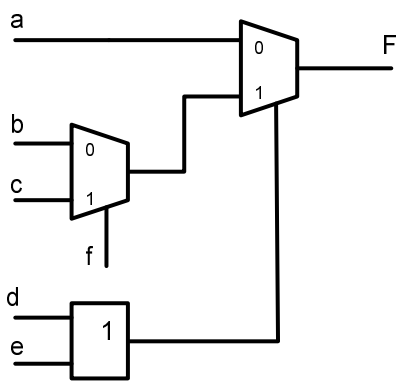
Рис. 2.48. Фрагмент структуры логарифмического сдвигателя

Для иллюстрации работы сдвигателя рассмотрим сдвиг на 9 разрядов. В двоичном коде число 9 представляется кодом $1001 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$, соответственно чему первая ступень устанавливается в режим сдвига, вторая и третья в режим пропуска сигнала без сдвига и третья в режим сдвига.

Контрольные вопросы и упражнения

1. Какие явления называются рисками сбоя? Укажите временной интервал, на котором после смены входных сигналов существуют риски?
2. Какими методами можно бороться с нарушениями нормальной работы цифровых устройств из-за влияния рисков? Какой метод является основным для современной схемотехники?
3. Какая характеристика служит показателем сложности схемы при ее реализации на одном кристалле и с помощью нескольких корпусов на печатной плате?

4. Какой дешифратор можно называть дешифратором-демультиплексором?
5. Составьте схему, реализующую умножение двух двухразрядных чисел с помощью двоичного дешифратора размерностью "4—16" с инверсными выходами и логических элементов И-НЕ.
6. Нарисуйте схему воспроизведения функции $F = X_1\overline{X}_2 \vee X_3$ с помощью дешифратора размерностью "3—8" и логических элементов ИЛИ.
7. Чем работа шифратора приоритетов отличается от работы двоичного шифратора?
8. Составьте схему, реализующую умножение двух двухразрядных чисел с помощью мультиплексоров размерностью "8—1" и логических элементов И-НЕ.
9. Нарисуйте схему воспроизведения функции $F = \overline{X}_1 \vee X_2 \vee \overline{X}_3X_4$ с помощью следующих микросхем:
   - а) мультиплексора размерностью "16—1";
   - б) мультиплексора размерностью "8—1" и элементов ИЛИ-НЕ;
   - в) мультиплексора размерностью "4—1" и элементов ИЛИ-НЕ.
10. Составьте схему компаратора на "больше" для двух двухразрядных слов $A = a_1a_0$ и $B = b_1b_0$ с помощью мультиплексора с алфавитом настройки $\{0, 1, a_1\}$.
11. Определите функциональную характеристику логического блока, изображенного на рисунке



12. Для каких областей применения эффективны контроль по модулю 2 и коды Хемминга? Каковы возможности этих методов?
13. Составьте кодовую комбинацию, соответствующую информационному слову 1101 при контроле по четности с применением модифицированного кода Хемминга.

14. При передаче по каналу связи четырехразрядного слова с контролем по четности и применением модифицированного кода Хемминга получено слово 01000110. Оцените эту ситуацию. Были ли ошибки передачи и если были, то какие? Можно ли пользоваться полученным результатом при наличии декодера кода Хемминга?

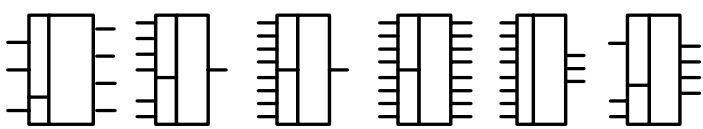
15. По методике, аналогичной методике построения сумматора с последовательным переносом, постройте четырехразрядный вычитатель.

16. Необходимо реализовать 24-разрядный сумматор с использованием микросхем 4-разрядных сумматоров, показанных на рис. 2.41 и имеющих задержку распространения сигнала по тракту $C_{ВХ}$-$C_{ВЫХ}$ 40 нс. Схемы 4-разрядных сумматоров имеют последовательный перенос и структуру, показанную на рис. 2.32. Найдите максимальное время суммирования для 24-разрядного сумматора.

17. Первоначально в проекте предполагалось применить 32-разрядный сумматор с последовательным переносом, в котором разряды выполнены по схеме, задерживающей сигнал переноса на $1{,}2t_з$ и вырабатывающей сигнал суммы за $2{,}4t_з$. Затем было решено перейти к групповой структуре сумматора (8 групп по 4 разряда) с цепным переносом, в которой цепь переноса имеет задержку $2t_з$. Оцените быстродействие указанных вариантов.

18. Составьте схему для перемножения восьмиразрядных чисел, с помощью множительно-суммирующих блоков размерностью 4×4.

19. Определите, какие блоки изображены здесь в виде условных обозначений без надписей



Литература к главе: [3], [5], [10], [26], [33], [34], [36], [39], [46], [47], [48], [52], [64].

Глава 3

Триггеры. Тактирование и синхронизация в цифровых устройствах

Триггеры — важнейшие элементы автоматов с памятью (АП), а тактирование и синхронизация — важнейшие процессы, обеспечивающие работоспособность автоматов. Изучение триггеров и процессов тактирования и синхронизации создает общую основу для понимания особенностей функционирования АП.

Автоматы с памятью АП (или просто *автоматы*) имеют фундаментальные отличия от комбинационных цепей. Реакция АП на входные сигналы зависит от его внутреннего состояния, и *одни и те же входные сигналы приводят к различным результатам, если автомат находился в разных внутренних состояниях.* Таким образом, новое состояние и новые выходные сигналы зависят от исходного состояния и входных воздействий. Поэтому текущее состояние и выходы определяются предысторией автомата — его начальным состоянием и всей последовательностью предыдущих входных сигналов. Преобразование последовательности входных сигналов в последовательность выходных, реализуемое автоматами, определило и их название — "последовательностные устройства".

§ 3.1. Триггеры. Основные сведения. Внешнее поведение

Триггеры — элементарные автоматы с емкостью памяти в один бит, содержащие собственно элемент памяти (бистабильную ячейку, фиксатор) и схему управления.

Бистабильная ячейка

Бистабильная ячейка строится из двух замкнутых в кольцо инверторов, так что выход одного соединен со входом другого (рис. 3.1, *а*). Та же схема при другом начертании представляется парой инверторов, связанных друг с другом "накрест" (рис. 3.1, *б* без учета штриховых линий). Такое соединение дает цепь с двумя устойчивыми состояниями. Действительно, соединенные в кольцо инверторы всегда находятся в противоположных состояниях, и состояние одного поддерживает со-

стояние другого. При этом возможны два состояния ячейки (0-1 или 1-0), любое из которых может существовать неограниченно долго. Состояние ячейки читается по прямому выходу Q, второй выход $\overline{Q}$ является инверсным.

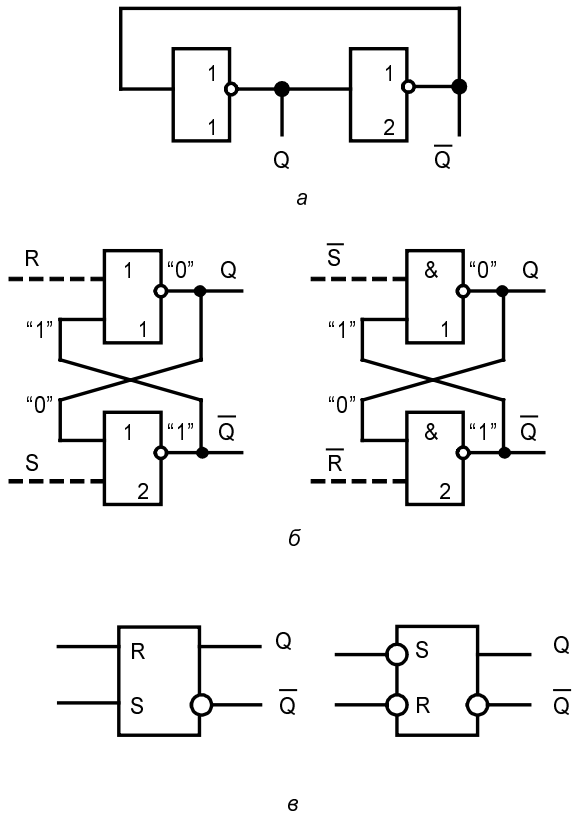


Рис. 3.1. Схемы бистабильных ячеек на инверторах (*а*), на элементах ИЛИ-НЕ и И-НЕ с входами управления (*б*) и их условные обозначения (*в*)

Переходное состояние, в котором инверторы имеют усилительные свойства, неустойчиво. Ячейка обязательно перейдет в одно из стабильных состояний, т. к. из-за наличия в схеме сильной положительной обратной связи любое изменение режима вызывает лавинообразное продолжение в том же направлении, пока ячейка не перейдет в устойчивое состояние. В состояниях 0 и 1 инверторы теряют усилительные свойства, что и делает стабильные состояния устойчивыми. При *достаточно определенном воздействии* на ячейку она переключается из одного состояния в другое с высокой скоростью.

Простейший триггер

Простейший триггер — это бистабильная ячейка, состоянием которой можно управлять, снабдив ее дополнительными входами. Для этого инверторы замещают-

ся инвертирующими элементами с двумя входами (ИЛИ-НЕ либо И-НЕ). На вторые входы элементов поступают внешние установочные (управляющие) сигналы.

Управляющие сигналы показаны на рис. 3.1 штриховыми линиями. Бистабильная ячейка с управляющими входами образует RS-триггер. Буквой R (от Reset) обозначен сигнал сброса триггера в нуль, а буквой S (от Set) — сигнал установки в единичное состояние. Состояние триггера считывается по значению прямого выхода Q.

> Для ячеек на элементах ИЛИ-НЕ *установочным сигналом служит единичный,* поскольку только он приводит элемент в нулевое состояние независимо от сигналов на других входах элемента. Для ячеек на элементах И-НЕ *установочным сигналом является нулевой*, как обладающий свойством однозначно задавать единичное состояние элемента независимо от состояний других входов.

Условные обозначения триггеров с входами установки/сброса показаны на рис. 3.1, *в*. Для триггерных схем на элементах И-НЕ и ИЛИ-НЕ существуют равноценные дуальные варианты. В приводимых далее примерах, как правило, используются варианты, построенные на элементах И-НЕ.

> Оценим необходимую *длительность входных сигналов* триггера. Для надежного переключения триггера входной сигнал должен сохраняться до прихода по цепи обратной связи сигнала, дублирующего входное воздействие, т. е. в течение времени, равного двум задержкам элементов, на которых собрана схема.
>
> Для RS-триггера существует *запрещенная комбинация* входных сигналов R = S = 1. Что же произойдет, если она возникнет? В этом случае оба выхода триггера (для схемы на элементах И-НЕ) станут единичными. Если после запрещенной комбинации появится комбинация R = S = 0 (режим хранения), то возникнет непредсказуемая ситуация. В конечном счете, схема перейдет в одно из устойчивых состояний, когда один из элементов имеет нулевое состояние, а другой — единичное. В какое именно? Происходит "противоборство" элементов, каждый из которых стремится навязать соседу свою "волю". Исход борьбы заранее неизвестен. Именно это заставляет считать комбинацию R = S = 1 запрещенной, т. к. пользоваться схемой, поведение которой непредсказуемо, нельзя.

Все серии цифровых ИС и библиотеки систем автоматизированного проектирования (САПР) содержат готовые триггеры, и задача проектировщика — их *правильное использование*, а не разработка. Поэтому важное значение приобретают не столько детали внутреннего устройства триггеров, сколько их классификация, характеристики и особенности функционирования.

Классификация триггеров

Классификация триггеров (рис. 3.2) проводится по трем признакам:

- ❒ логическому функционированию;
- ❒ способу приема и выдачи информации;
- ❒ схемотехнологии.

Классификация триггеров по логическому функционированию

Широко применяются триггеры RS, D, T, JK. Существуют также комбинированные триггеры, в которых совмещаются одновременно несколько типов, и триггеры со сложной входной логикой (группами входов, связанных между собой логическими зависимостями).

Триггер RS имеет два информационных входа — установки (S) и сброса (R). Одновременная подача сигналов S и R (запрещенная комбинация) не допускается.

Триггер D (от delay — задержка) имеет один информационный вход и принимает входной сигнал только по разрешению тактового сигнала C (Clock).

Триггер T (от toggle — переключение) или *счетный* триггер имеет один информационный вход и изменяет свое состояние каждый раз при поступлении входного сигнала.

Триггер JK универсален, имеет информационные входы установки (J) и сброса (K), подобные входам триггера RS. В отличие от последнего не имеет запрещенной комбинации входных сигналов и допускает одновременную подачу единиц на оба входа (J = K = 1). В этом режиме работает как счетный триггер относительно третьего (тактового) входа.

В *комбинированных* триггерах совмещаются несколько режимов. Например, триггер типа RST — счетный триггер, имеющий также входы установки и сброса.

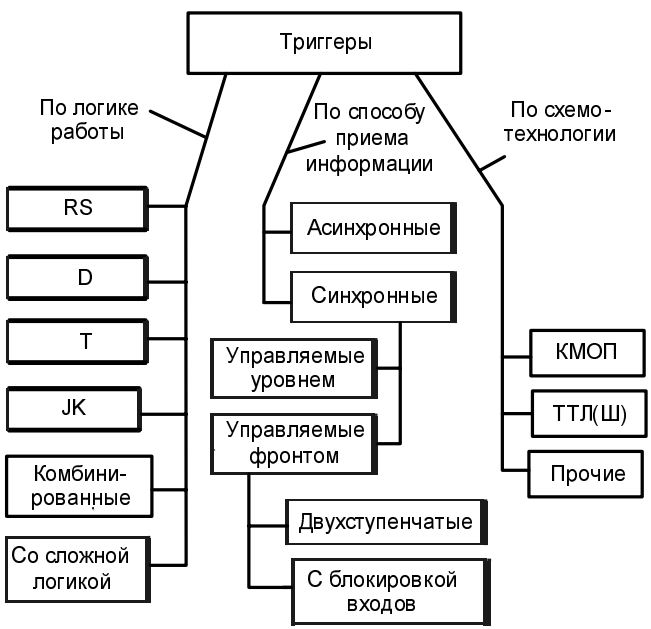


Рис. 3.2. Классификация триггеров, используемых в практической схемотехнике

Пример триггера *со сложной входной логикой* — JK-триггер с группами входов $J_1J_2J_3$ и $K_1K_2K_3$, соединенными операцией конъюнкции, так что $J = J_1J_2J_3$, $K = K_1K_2K_3$.

В практике проектирования цифровых устройств *доминируют триггеры D и JK.*

Классификация триггеров по способу приема информации

По этому признаку различают *асинхронные (не тактируемые)* и *синхронные (тактируемые)* триггеры. В асинхронных триггерах переход в новое состояние вызывается непосредственно изменениями входных информационных сигналов. В тактируемых, имеющих специальный вход (синхровход), восприятие информационных сигналов происходит только по разрешению тактовых сигналов C (Clock), подключаемых к синхровходу.

В схемотехнике *главную роль играют тактируемые триггеры*. Асинхронные процессы используются обычно только для организации входов установки/сброса и внутренних фрагментов триггерных схем.

Тактируемые триггеры делятся на *управляемые уровнем* и *управляемые фронтом*. Управление уровнем означает, что при одном уровне тактового сигнала триггер воспринимает входные сигналы, а при другом не воспринимает и остается в неизменном состоянии. Триггеры, управляемые уровнем, называют также триггерами со *статическим управлением*. При управлении фронтом разрешение на переключение (прием входных воздействий) дается только в момент перепада тактового сигнала. В остальное время независимо от уровня тактового сигнала триггер не воспринимает входные сигналы и остается в неизменном состоянии. Триггеры, управляемые фронтом, называют также триггерами с *динамическим управлением*.

Тактирующие входы могут быть *прямыми или инверсными*. При прямом статическом входе разрешающим уровнем служит уровень логической единицы, при инверсном — логического нуля. Прямое динамическое управление означает разрешение на переключение при положительном фронте тактового сигнала (его изменении с нулевого значения на единичное), инверсное — при отрицательном фронте тактового сигнала (его изменении с единичного значения на нулевое).

На рис. 3.3 показаны процессы, происходящие в тактируемых триггерах. На диаграммах тактовых импульсов отмечено содержание процессов на отдельных этапах, здесь же даны обозначения входов для соответствующих триггеров. При этом логический тип триггера и число его информационных входов не определены (операция отмечена решеткой, входы — двойной стрелкой).

Принадлежность схемы к классу триггеров отмечается в условных обозначениях буквой T (согласно стандарту эту букву можно и не проставлять, если характер входов ясно указывает на то, что данный элемент является триггером). Для триггеров с динамическими входами показаны два варианта обозначений. Оба допускаются стандартом.

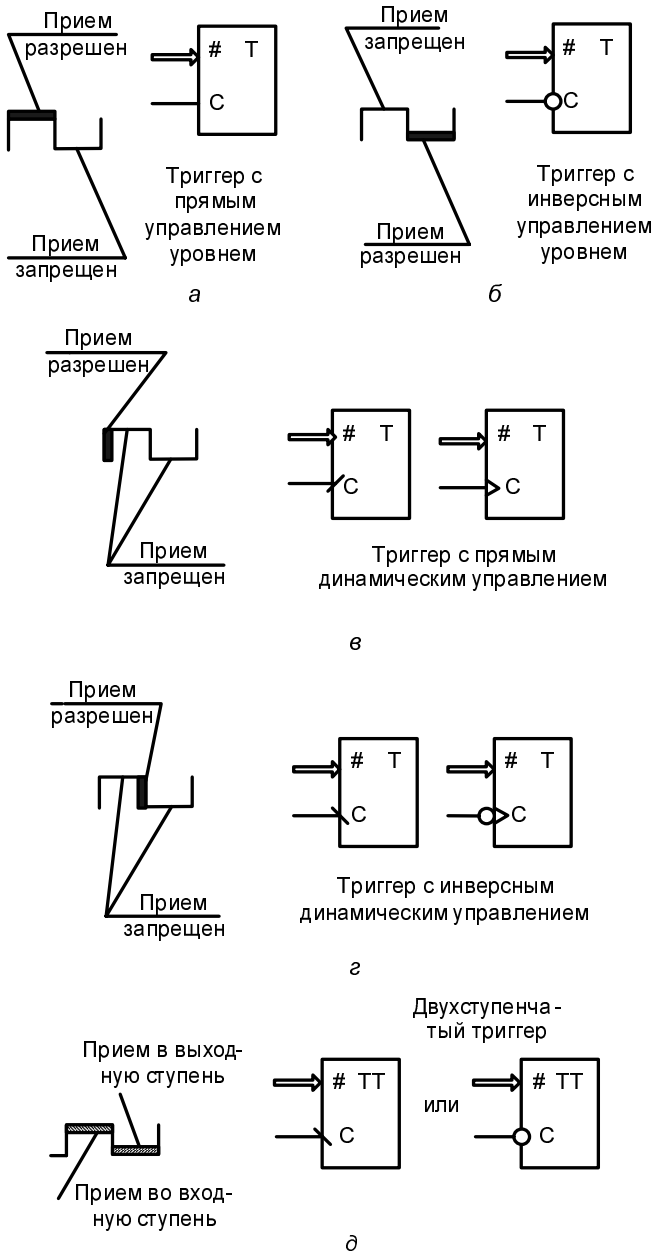


Рис. 3.3. Временные диаграммы, поясняющие работу синхронных триггеров, и условные обозначения тактирующих входов для триггеров с прямым управлением уровнем (*а*), инверсным управлением уровнем (*б*), прямым динамическим управлением (*в*), инверсным динамическим управлением (*г*) и двухступенчатых (*д*)

Далее будем использовать преимущественно вариант с косыми черточками, как более простой в начертании.

Один из схемотехнических способов достижения управляемости фронтом — построение *двухступенчатых* триггеров. Двухступенчатый триггер при необходимости обозначается двумя буквами Т (т. е. как ТТ). Разрешающим уровнем тактового сигнала для двухступенчатого триггера будем считать тот, который переносит информацию из входной ступени в выходную, т. к. именно при этом новая информация появляется на выходе триггера. Двухступенчатые триггеры называют также *триггерами типа MS* (от английского Master-Slave, т. е. "хозяин — раб"). Эта аббревиатура отражает характер работы триггера: входная ступень вырабатывает новое значение выходной переменной Q, а выходная его копирует.

На рис. 3.3, *д* приведены два обозначения двухступенчатых триггеров. Причина состоит в том, что некоторые (старые) разновидности двухступенчатых триггеров из-за явлений "захвата единицы" и "захвата нуля" [34] по своему поведению отличаются от триггеров с динамическим управлением. Современные двухступенчатые триггеры, свободные от "захватов", по внешнему поведению можно трактовать, как *управляемые фронтам*и. В связи со сказанным, условные обозначения двухступенчатых триггеров даны на рис. 3.3, *д* в двух вариантах — для триггеров, трактуемых как управляемые фронтом, и для триггеров с явлениями "захватов" единиц или нулей.

Замечания о терминологии. В связи с классификацией триггеров уточним сложившуюся в настоящее время терминологию. В отечественной литературе преобладает применение термина "триггер" ко всем элементарным автоматам, которые затем подразделяются на ряд типов в зависимости от логики работы и способа приема информации. В иностранной литературе положение иное. Элементарные автоматы типа D с управлением уровнем, имеющие режим прозрачности, из которого их можно перевести в режим хранения, называют "защелками" (latch). Элементарные автоматы, управляемые фронтами, называют "триггерами" (flip-flop). Термин flip-flop возник во времена, когда функции, свойственные триггерам, выполнялись электромагнитными реле, шелест контактов которых ассоциируется по звучанию со словами flip-flop. Иностранная терминология начинает преобладать, поскольку именно она используется в пакетах программ автоматизированного проектирования. В частности, в терминологии для примитивов библиотек САПР триггеры D с управлением уровнем обычно обозначаются как LATCH или DLATCH, а различные триггеры с управлением фронтом имеют в обозначении буквы FF (DFF, TFF, JKFF, SRFF).

По схемным и технологическим признакам триггеры образуют те же типы, что и логические элементы. С учетом преобладающих в настоящее время вариантов в классификации выделены только триггеры схемотехнологий КМОП и ТТЛ(Ш). Остальные типы (ЭСЛ, $И^2Л$, n-МОП и др.) отнесены к прочим.

Тактирование уровнем. Режим прозрачности. Круговые гонки

Синхронные D-триггеры, *управляемые уровнем* (*защелки*), имеют *режим прозрачности*. В режиме прозрачности прием данных в триггер разрешен тактовым сигналом и триггер все время воспринимает входные информационные сигналы, отслеживая их изменения, т. е. повторяет входной сигнал. При изменении тактового сигнала режим прозрачности прекращается, слежение за входом заканчивается, и триггер переходит в режим хранения (защелкивается).

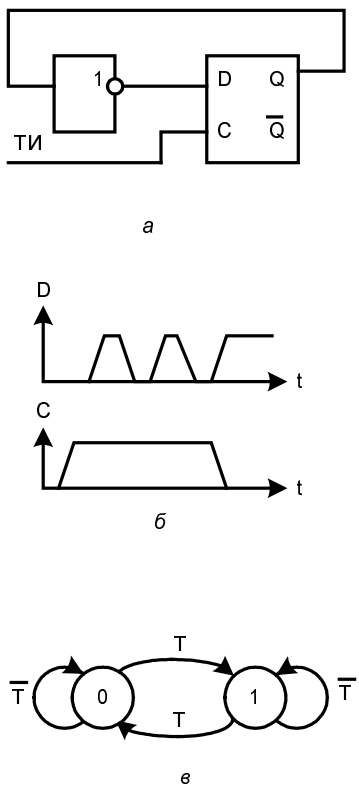


Рис. 3.4. Схема, иллюстрирующая возникновение круговых гонок (*а*), временные диаграммы процесса генерации (*б*) и соответствующий граф переходов (*в*)

Режим прозрачности в ряде случаев полезен и целесообразен. В то же время есть ситуации, в которых он создает серьезные проблемы. Рассмотрим для примера схему (рис. 3.4, *а*), в которой сигнал обратной связи подается через инвертор с выхода D-триггера на его вход. Пока синхросигнал имеет высокий логический уровень, прием информации с входа D разрешен, и в схеме возникнет генерация импульсов с чередованием состояний 0-1-0-1-0-1-…, поскольку триггер все время

принимает с выхода сигнал противоположного состояния. Временные диаграммы процесса генерации показаны на рис. 3.4, *б*, а граф переходов на рис. 3.4, *в*. Это явление можно назвать *круговыми гонками.*

Круговые гонки не возникнут, если длительность разрешающего уровня синхросигнала будет меньше, чем задержка сигнала в петле обратной связи, т. к. в этом случае к моменту изменения значения D на входе триггера, он уже будет блокирован запрещающим уровнем синхросигнала. К сожалению, такой метод борьбы с круговыми гонками обычно непригоден, поскольку синхросигналы, как правило, являются общими для большого количества триггеров, имеющих разброс по быстродействию. При этом длительность тактового импульса, необходимая для переключения наиболее инерционного триггера, может оказаться такой, что быстродействующий триггер успеет за это время переключиться не один раз, что недопустимо. Иными словами, разброс быстродействия не позволяет подобрать длительность единого синхросигнала, пригодную одновременно для множества триггеров.

Предотвращение круговых гонок. Триггеры MS и ET. Круговые гонки можно исключить применением в схемах с обратными связями триггеров, не имеющих режима прозрачности: двухступенчатых (типа MS, Master-Slave) или управляемых фронтом (типа ET, Edge Triggered).

Времена предустановки и выдержки

Тактируемые триггеры имеют два важных параметра — *время предустановки* t_{SU} (Set-Up Time) и *время выдержки* или иначе *время удержания* t_H (Hold Time). Время t_{SU} — это интервал *до поступления синхросигнала*, в течение которого информационный сигнал должен оставаться неизменным (рис. 3.5).

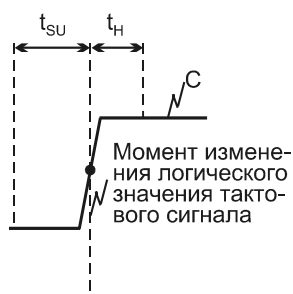


Рис. 3.5. К пояснению параметров предустановки и выдержки для синхронных триггеров

Время выдержки t_H — это время *после поступления синхросигнала*, в течение которого информационный сигнал должен оставаться неизменным. Соблюдение времен предустановки и выдержки обеспечивает правильное восприятие триггером входной

информации. Для надежного проектирования тактируемых устройств желательно обеспечивать соблюдение интервалов предустановки и выдержки с запасами. Соблюдение требований по предустановке и выдержке информационных сигналов для всех триггеров, входящих в цифровые устройства, является одной из важнейших и сложнейших задач их проектирования, особенно для устройств высокого быстродействия. Нарушение таких требований чревато возникновением аномальных состояний триггеров, несовместимых с нормальным функционированием аппаратуры.

Метастабильные состояния триггеров

При применении тактируемых триггеров приходится сталкиваться с проблемой аномальных состояний. Триггер быстро принимает одно из устойчивых состояний при достаточно определенном воздействии на него. Чтобы избежать неопределенных воздействий, вводятся *запретные зоны*, в которых информационные сигналы не должны изменяться — времена предустановки и выдержки. Ясно, что при приеме *асинхронных информационных сигналов*, появляющихся в произвольные моменты времени, соблюдать требования по временам предустановки и выдержки невозможно, и триггер может попадать под неопределенные воздействия. Например, D-триггер может получить сигнал переключения по информационному входу одновременно с переходом синхросигнала в состояние запрета приема информации. Процесс переключения может начаться, но затем прекратиться в некотором промежуточном состоянии, т. к. синхросигнал отключит триггер от информационных входов. Триггер, предоставленный самому себе, рано или поздно перейдет в одно из устойчивых состояний (вернется в исходное состояние или перейдет в противоположное). Однако если его оставить в точке, *очень близкой к равновесию*, то выход из нее окажется аномально длительным.

Аномалии разделяются *на метастабильные и колебательные*. В первом случае напряжения на обоих выходах триггера близки к пороговым напряжениям логических элементов, из которых собран триггер. Во втором случае выходные напряжения триггера медленно колеблются вокруг пороговых напряжений элементов. Поскольку имеются схемы триггеров, не входящие в режим аномальных колебаний, дальше будут обсуждаться только проблемы метастабильных состояний.

Длительность метастабильных состояний может быть несопоставимой с нормальными длительностями переходных процессов. Метастабильности большой длительности менее вероятны, чем более короткие, т. к. возникают при попадании триггера в более узкую зону вблизи точки равновесия.

Метастабильные состояния — неустранимые явления, определяющие неизбежность сбоев при работе тактируемых схем с асинхронными сигналами. Можно принимать меры лишь для снижения частоты (вероятности) их возникновения. Вероятность возникновения метастабильных состояний снижается при уменьшении

интервалов предустановки и выдержки, что свойственно триггерам высокого быстродействия. Параметры триггера, характеризующие его качество с точки зрения вероятности возникновения метастабильных состояний, особенно важны для триггеров, работающих в схемах синхронизаторов. В этих схемах не избежать ситуаций, вызывающих метастабильность. В число параметров триггера в подобных случаях вводят *среднее время между отказами* из-за метастабильных состояний MTBF (Mean Time Between Failures), которое характеризует свойства триггера на основе статистики, получаемой в экспериментах.

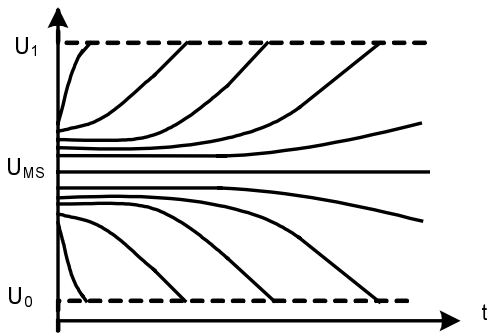


Рис. 3.6. Траектории выходных напряжений триггера, попадающего в метастабильные состояния

На рис. 3.6 приведены траектории выходных сигналов триггеров, попадающих в метастабильные состояния. Точке идеально равновесного состояния схемы U_{MS} (точке метастабильности) соответствует метастабильное состояние бесконечной длительности. Чем дальше исходное состояние от точки метастабильности, тем быстрее изменяются выходные сигналы триггера, стремясь к своим установившимся состояниям. Задерживая опрос состояния триггера относительно момента его тактирования, можно уменьшать число метастабильных состояний, которые будут наблюдаться на выходе триггера. На этом основаны методы построения схем ввода асинхронных сигналов в синхронные системы.

Способы описания триггеров

Логическое функционирование триггеров описывается общими для автоматов способами: таблицами истинности, картами Карно, характеристическими уравнениями, диаграммами состояний, "словарями", временными диаграммами.

Ниже рассматривается логика работы наиболее распространенных триггеров JK, D и, менее подробно, триггера T. Работа триггера RS совпадает с работой JK-триггера во всем за исключением наличия у триггера RS запрещенного состояния. Таблицу истинности триггера JK можно записать в полном (табл. 3.1) или сжатом виде (табл. 3.2). Через Q_H обозначено новое состояние триггера (после переключения).

Таблица 3.1

J	K	Q	$Q_н$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Таблица 3.2

J	K	$Q_н$
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Карта Карно для JK-триггера показана на рис. 3.7. Из нее можно получить *характеристические уравнения триггера,* описывающие его поведение в базисе И, ИЛИ, НЕ:

$$Q_{н} = J\overline{Q} \vee Q\overline{K} \quad \text{и} \quad \overline{Q}_{н} = \overline{J\overline{Q}} \vee QK .$$

Переведя уравнения в логический базис элементов, из которых строится триггер, получим *структурные уравнения* триггера, определяющие конфигурацию его схемы.

Примечание

Реальные схемы триггеров, особенно при проектировании на транзисторном уровне, являются, как правило, не просто воплощением схем, получаемых формальным путем, но результатом их изобретательских усовершенствований.

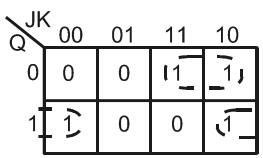


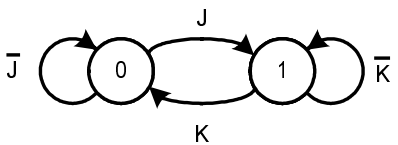
Рис. 3.7. Карта Карно для JK-триггера

Таблица 3.3

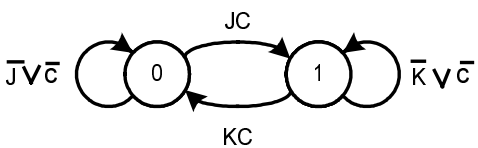
Переход	J	K
$0 \to 0$	0	X
$0 \to 1$	1	X
$1 \to 0$	X	1
$1 \to 1$	X	0

Словарь (табл. 3.3) определяет информационные сигналы, обеспечивающие переходы триггера из одного состояния в другое, и является удобным инструментом проектирования схем, содержащих триггеры.

Диаграмма состояний или *граф переходов* (рис. 3.8, *а*) отражает наличие у триггера двух устойчивых состояний и информационные сигналы, обеспечивающие переходы из одного состояния в другое, т. е. предоставляет ту же информацию в графической форме. На рис. 3.8, *б* показана диаграмма состояний с учетом сигналов тактирования C (Clock). Эту диаграмму можно получить из таблицы функционирования триггера, дополнив число входов сигналом тактирования C, но нетрудно сделать это и на основе простых рассуждений: информационные сигналы воздействуют на триггер только при наличии разрешения сигналом C, а к режимам хранения состояния добавляется режим отсутствия разрешения на прием информации.



а



б

Рис. 3.8. Диаграммы состояний (графы переходов) для триггера JK

Для D-триггера сокращенная таблица истинности дана в табл. 3.4, а словарь в табл. 3.5. Характеристическое уравнение триггера $Q_H = D$.

Таблица 3.4

D	Q_H
0	0
1	1

Таблица 3.5

Переход	D
$0 \to 0$	0
$0 \to 1$	1
$1 \to 0$	0
$1 \to 1$	1

Таблица 3.6

Переход	R S	T
$0 \to 0$	X 0	0
$0 \to 1$	0 1	1
$1 \to 0$	1 0	1
$1 \to 1$	0 X	0

Словари триггеров RS и T имеют вид (табл. 3.6).

Важным способом описания функционирования триггеров (как и других автоматов) являются *временные диаграммы*, отражающие не только логическое функционирование схемы, но и ее поведение во времени. Это поведение другими способа-

ми описания работы триггеров не отображается, и поэтому в ряде случаев временные диаграммы незаменимы. Временные диаграммы будут рассматриваться при описании конкретных схем.

§ 3.2. Схемотехника триггерных устройств

При изложении темы рассмотрим два ее аспекта:

- конфигурации схем, обеспечивающих требуемое логическое функционирование триггера;
- меры устранения режимов генерации в схемах, имеющих обратные связи (помимо обратных связей в бистабильной ячейке);
- ознакомление со схемами триггеров и методами их построения полезно не только само по себе, но и как пример, иллюстрирующий методику разработки цифровых устройств.

Триггеры в биполярной схемотехнике

На реализацию триггеров существенно влияет отсутствие (в триггерах RS и D) или наличие (в триггерах T и JK) обратных связей (обратные связи в схеме самой бистабильной ячейки во внимание не принимаются).

Простые RS-триггеры и защелки

Асинхронные RS-триггеры. Эти триггеры представляют собой бистабильные ячейки со входами сброса и установки (см. рис. 3.1).

Асинхронные D-триггеры. Эти триггеры применения не имеют, т. к. просто повторяют входные сигналы и с точки зрения функционирования не представляют интереса.

Синхронные RS- и D-триггеры, управляемые уровнем. Эти триггеры содержат асинхронный RS-триггер и вентили разрешения/запрещения передачи информационных сигналов, управляемые тактирующим сигналом (рис. 3.9).

При C = 0 для асинхронного RS-триггера создается режим хранения (рис. 3.9, *а*, *в*). Если же C = 1, то для информационных сигналов создаются цепи их передачи на асинхронный RS-триггер, соответствующие требуемой логике работы триггеров. Таким образом, переключение разрешается только при C = 1. Условное обозначение триггера RS показано на рис. 3.9, *б*.

Синхронный D-триггер, управляемый уровнем (защелка) получается из триггера RS, если подавать на вход S значение D, а на вход R его инверсию (рис. 3.9, *в*). Такая схема строится на пяти логических элементах. Однако есть возможность один элемент исключить, если при C = 1 вентиль 2 использовать "по совместительству" как инвертор (рис. 3.9, *г*).

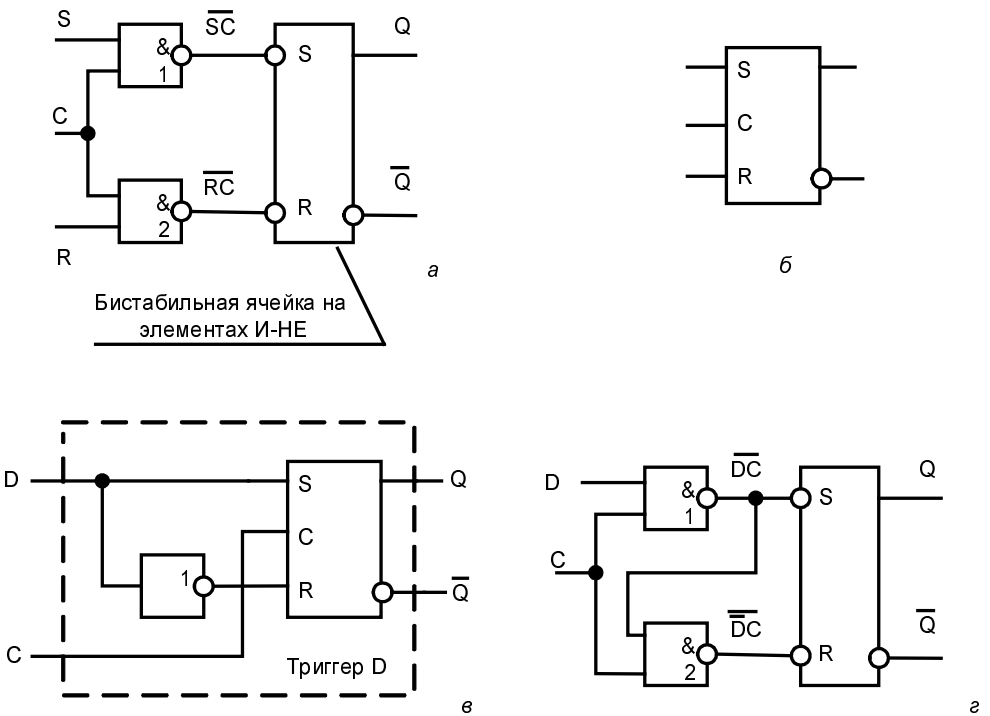


Рис. 3.9. Схема синхронного RS-триггера с управлением уровнем (*а*), ее условное обозначение (*б*), схема информационных связей, образующих триггер D (*в*) и реализация триггера D на четырех вентилях (*г*)

Синхронные RS- и D-триггеры на элементах ИЛИ-НЕ строятся принципиально так же, как и на элементах И-НЕ.

Логические структуры триггеров T и JK

В триггерах T и JK выходные сигналы используются как сигналы обратных связей (помимо обратных связей в бистабильной ячейке).

В триггере T на основе триггера RS (рис. 3.10, *а*) тактирующий вход играет роль счетного. При каждом разрешении приема информации триггер по обратным связям принимает состояние, противоположное текущему, т. е. переключается. Триггер T аналогичным способом можно получить и на основе D-триггера (рис. 3.10, *б*). Для исключения круговых гонок в таких схемах применяют *непрозрачные триггеры.*

Схему информационных связей, создающую JK-триггер на основе триггера D, определим, пользуясь характеристическим уравнением JK-триггера $Q_н = J\overline{Q} \vee Q\overline{K}$. Для D-триггера $Q_н = D$, следовательно, для построения триггера JK (рис. 3.11, *а*) нужно выполнить условие:

$$D = J\overline{Q} \vee Q\overline{K}.$$

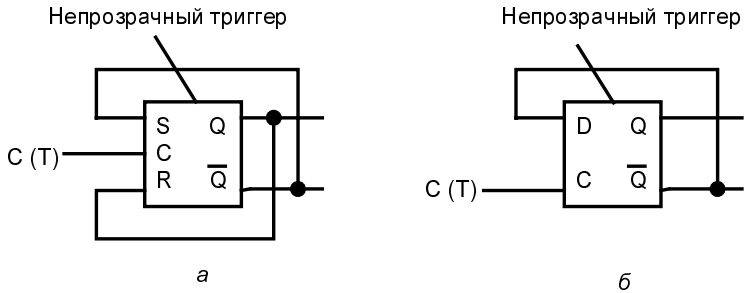


Рис. 3.10. Схемы информационных связей, образующих триггеры T: на основе RS-триггера (*а*) и D-триггера (*б*)

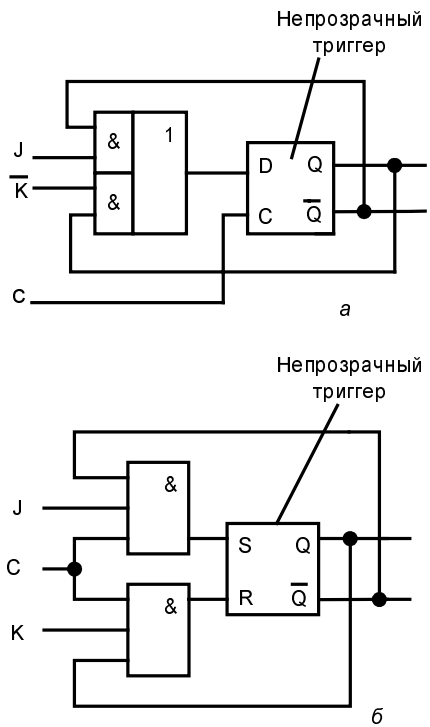


Рис. 3.11. Схемы информационных связей, образующих структуры триггеров JK на основе триггера D (*а*) и триггера RS (*б*)

Нетрудно получить схему информационных связей для построения триггера JK на основе RS-триггера. Такая структура имеет недостаток — явления захватов единиц и нулей [34], но еще встречается в практике. Для триггера RS характеристические уравнения имеют вид: $Q_{н} = S \vee Q\overline{R}$ и $\overline{Q}_{н} = R \vee \overline{Q}\,\overline{S}$. Сравнивая эти уравнения с характеристическими уравнениями для JK-триггера, можно видеть, что они совпадут, если вырабатывать значения S и R по соотношениям: $S=J\overline{Q}$, $R=\overline{K}Q$, что и реализовано в схеме (рис. 3.11, *б*).

Счетные режимы триггеров JK. Триггер JK можно поставить в счетный режим двумя способами. Если подать на входы J и K единичные сигналы (рис. 3.12, *а*), то относительно тактирующего входа триггер будет работать как триггер T. Такую схему следует называть *асинхронной*, т. к. она реагирует непосредственно на изменения информационного сигнала, хотя он и подается на вход тактирования.

Если же соединить входы J и K друг с другом и использовать их общий вывод для подачи информационного сигнала T (рис. 3.12, *б*), то получится вариант, который является *синхронным* T-триггером, поскольку реакция на информационный сигнал будет наступать только по разрешению сигнала тактирования C.

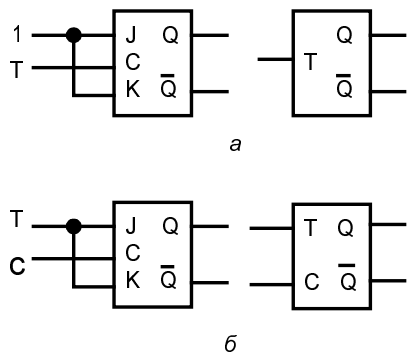


Рис. 3.12. Варианты включения JK-триггера в режим триггера T

Двухступенчатые триггеры

Вариант двухступенчатого триггера, называемый "схемой с инвертором", показан на рис. 3.13. Логический тип триггера первой ступени не определен (показан символом "решетка"), т. к. в данном случае он не имеет значения.

Триггер *не имеет режима прозрачности*, поскольку его ступени принимают информационные сигналы (открываются) только поочередно, и ни один из уровней тактирующего сигнала не создает пути для сквозного прохождения информации со входа на выход. Прием данных во входную ступень запрещается, когда разрешаются изменения выходного напряжения. Иными словами, петля обратной связи всегда прерывается входной или выходной ступенью. При этом независимо от длительности разрешающего уровня синхросигнала *круговые гонки не возникнут*. Для примера на рис. 3.14 изображена диаграмма состояний T-триггера с двухступенчатой структурой. Первая цифра кода состояния соответствует входной ступени, вторая — выходной. Видно, что в такой структуре режим генерации не возникает при любой длительности входного сигнала T.

С давних пор двухступенчатые триггеры радикально избавляли синхронные автоматы от опасных состязаний. В схемотехнике ТТЛ(Ш) параметры одноступенчатых триггеров с динамическим управлением (нулевое время выдержки t_h и др.) по-

зволили строить на них надежные схемы автоматов, что сузило область применения двухступенчатых триггеров. В схемотехнике КМОП двухступенчатые триггеры вновь играют основную роль [41].

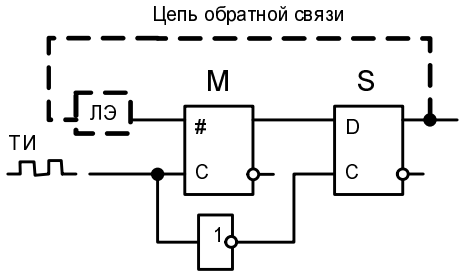


Рис. 3.13. Схема двухступенчатого триггера, охваченного обратной связью

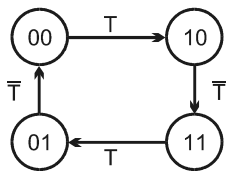


Рис. 3.14. Диаграмма состояний двухступенчатого Т-триггера

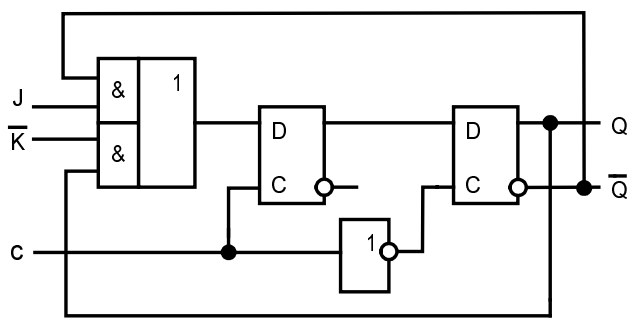


Рис. 3.15. Схема триггера JK, построенного на основе двухступенчатого триггера D

На рис. 3.15 показана схема JK-триггера, построенного на основе двухступенчатого триггера D, реализованного по "схеме с инвертором".

Двухступенчатые триггеры с инвертором не свободны от временных состязаний, в их схемах инвертор состязается с входной ступенью. Для правильной работы схемы инвертор должен сформировать сигнал запрещения приема данных в выходную ступень раньше, чем на выходе входной ступени начнутся очередные изменения сигналов.

В двухступенчатых триггерах с *разнополярным управлением ступенями* одна имеет прямое, другая инверсное управление. Один и тот же сигнал тактирования подается

на обе ступени, и никаких элементов, задерживающих передачу запрещающего сигнала, нет. Состязания отсутствуют. Однако это удобство сопровождается потерей идентичности ступеней. В варианте *с запрещающими связями* активные сигналы записи во входную ступень (инверсные по отношению к тактирующим) передаются одновременно и на блокирование передач в выходную ступень.

Одноступенчатые триггеры, управляемые фронтом

Управление фронтом в одноступенчатых триггерах достигается следующими методами:

- блокированием информационных входов после прохождения фронта синхросигнала;
- формированием в схеме триггера внутренних коротких синхроимпульсов.

Первый способ применяется в так называемой "*схеме трех триггеров*", другим названием которой является "*шестиэлементный триггер*". Второй способ характерен для "*триггеров с внутренней задержкой*".

Триггеры по схеме "трех триггеров". На рис. 3.16 иллюстрируется способ, которым достигается прием входного сигнала триггера только под действием положительного перепада тактового сигнала. После этого перепада схема становится нечувствительной к входному сигналу вплоть до нового положительного перепада тактового сигнала. Для сравнения штриховой линией показан вход, соответствующий прямому статическому управлению (для примера взят вход установки триггера S). При прямом статическом управлении и C = 1 вентиль 2 для сигнала $S_{СТ.УПР}$ играет роль инвертора, через который входной сигнал передается на бистабильную ячейку (при $S_{СТ.УПР} = 1$ ячейка получает сигнал установки).

Иначе дело обстоит для динамического управления. Здесь при нулевом уровне тактового сигнала выход A вентиля 2 единичен, т. е. пассивен и не может влиять на бистабильную ячейку. При этом сигнал $\overline{S}_{ДИН.УПР}$ проходит через вентиль 1, инвертируется им и на верхнем входе вентиля 2 присутствует сигнал $S_{ДИН.УПР}$. Переход тактового сигнала к единичному значению пропускает инверсию входного сигнала на бистабильную ячейку. При $S_{ДИН.УПР}$, равном единице, происходит установка триггера, поскольку формируется A = 0. Сигнал A одновременно с воздействием на бистабильную ячейку блокирует воздействие изменений входного сигнала на состояние схемы, поскольку нижний вход вентиля 1 становится нулевым. Новый прием входного сигнала возможен только при новом положительном фронте тактового сигнала. Таким образом, чтобы триггер имел чувствительность к входному сигналу только во время фронта тактового сигнала C, нужно автоматически отключать цепь входа сразу же после прохождения этого фронта. И до и после тактирующего фронта цепь передачи входных информационных сигналов должна быть блокирована. Только появление тактирующего фронта должно кратковременно разрешать прохождение информационного сигнала в триггер.

На основе рассмотренного способа строятся триггеры D, T и JK по схеме "трех триггеров".

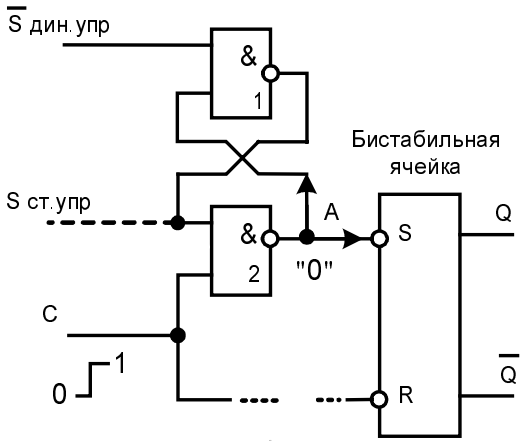


Рис. 3.16. Схема с управлением фронтом тактового сигнала

Триггеры с внутренней задержкой. Триггеры с внутренней задержкой также воспринимают информационные сигналы только по разрешению перепада тактирующего сигнала. Для этого применяется формирование внутренних коротких импульсов установки/сброса бистабильной ячейки, длительность которых определяется заданным соотношением задержек элементов, а момент появления синхронизирован с активным фронтом тактирующего сигнала.

> ***Примечание***
>
> Ранее отмечалось, что для борьбы с круговыми гонками практически не удается применять укорачивание длительности синхросигналов из-за разброса характеристик триггеров, управляемых одним и тем же синхросигналом. В триггерах с внутренней задержкой фактически используется именно сокращение длительности сигнала, разрешающего прием данных, но такие сигналы *вырабатываются в каждом триггере индивидуально*, и их длительность согласована с быстродействием элементов именно этой схемы. Поэтому проблема непригодности единых коротких синхросигналов для управления множеством триггеров снимается — единой длительности синхроимпульсов вообще не будет.

Подробнее схемы триггеров с управлением фронтом рассмотрены в предыдущем издании этой книги и работах [34] и др.

Входы установки/сброса и разрешения тактирования

В рассмотренных ранее примерах не были показаны *входы установки/сброса и входы разрешения тактирования,* являющиеся дополнительными средствами управления, типичными для синхронных триггеров. Асинхронные входы установки/сброса обозначаются как S и R (или Preset и Clear в тех случаях, когда триггер имеет и асинхронные и синхронные входы установки/сброса). Эти входы являются доминирующими, воздействие по ним осуществляется в любое время независимо от сигналов на других входах (в том числе тактовых), которые при этом игнориру-

ются. Когда асинхронные сигналы снимаются, обусловленное ими состояние триггера сохраняется до первого активного фронта синхросигнала, определяющего новое состояние триггера в соответствии с его информационными входами.

Сигналы асинхронных входов установки/сброса подаются на дополнительные входы элементов схемы так, чтобы однозначно определить состояние триггера при любых значениях других сигналов. Например, шестиэлементный триггер устанавливается L-активным сигналом по дополнительным входам элементов, входящих в состав бистабильной ячейки. Для реализации синхронных входов применяют варианты с формированием сигналов установки/сброса как конъюнкций с тактовыми сигналами.

Сигналы разрешения тактирования CE (Clock Enable) разрешают или запрещают воздействие на триггер тактовых импульсов. Входы CE организуют различными способами, в том числе с помощью мультиплексоров (рис. 3.17).

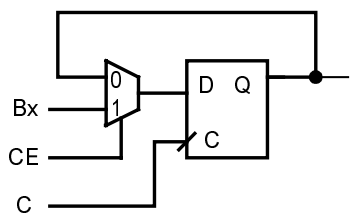


Рис. 3.17. Схема организации входа разрешения тактирования с помощью мультиплексора

Триггеры в схемотехнике КМОП

Схемотехника триггеров на элементах КМОП имеет свои особенности. В число элементов схемы здесь наряду с обычной логикой входят двунаправленные ключи, позволяющие передавать логические сигналы без потери их амплитудных значений (см. *§ 2.4*). Применение таких ключей позволяет строить экономичные структуры триггеров.

Триггер-защелка

Идея построения D-триггера с управлением уровнем (защелки) показана на рис. 3.18, *а*. При замкнутом ключе входной сигнал передается прямо на выход, т. е. реализуется режим прозрачности. При разомкнутом ключе на выход передается состояние ячейки памяти, которое она имела перед размыканием ключа — реализуется режим хранения.

Структура триггера на вентильном уровне приведена на рис. 3.18, *б*. В зависимости от уровня тактового сигнала C, являющегося адресным для мультиплексора, на вход цепочки из двух инверторов подается входной сигнал D или выходной Q. В первом случае сигнал D передается на выход Q, а точка соединения инверторов играет роль инверсного выхода. Во втором случае инверторы

замыкаются в кольцо, представляющее собою бистабильную ячейку, и триггер переходит в режим хранения.

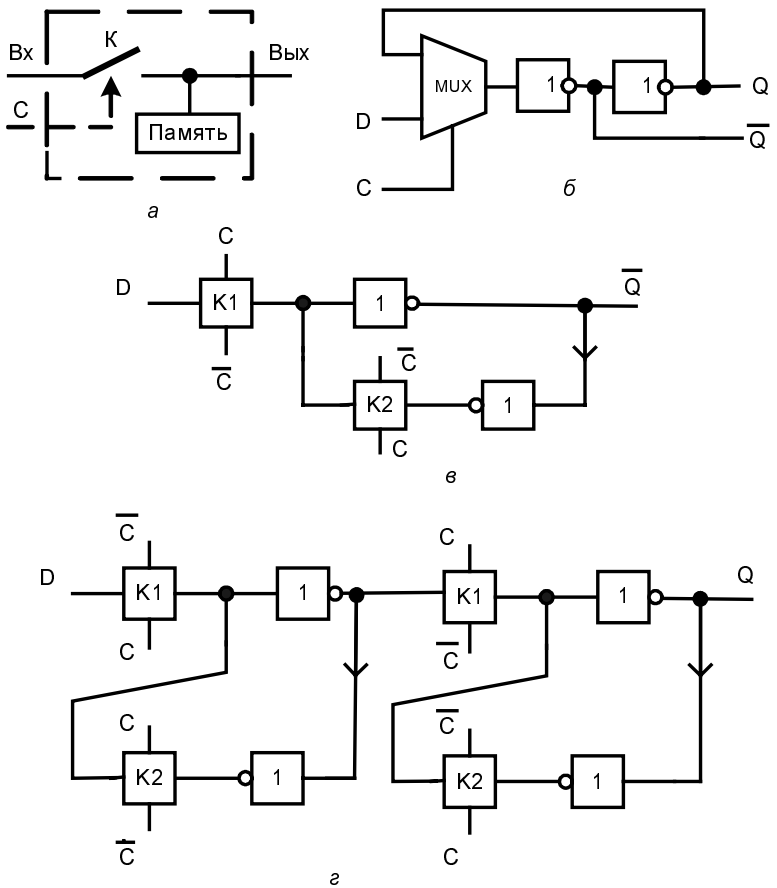


Рис. 3.18. Идея построения триггера-защелки в КМОП-схемотехнике (*а*), структура триггера на вентильном уровне (*б*), схема триггера-защелки (*в*) и двухступенчатого триггера (*г*)

Структура относится к *псевдостатическим*. Память в динамических схемах обеспечивается хранением зарядов конденсаторов, а в статических — обратными связями. Динамическая память кратковременна, а статическая сохраняет данные, пока находится под питанием. В рассматриваемой схеме используются оба вида памяти. Долговременную память создает бистабильная ячейка из двух инверторов. При коммутации ключей, когда сигнал D замещается величиной Q, возникает кратковременная ситуация, в которой инверторы предоставлены самим себе. Они не теряют при коммутации своего состояния благодаря инерционности, вызванной сохранением зарядов в емкостях схемы.

В схему триггера-защелки (рис. 3.18, *в*) входят ключи К1 и К2, которые всегда находятся в противоположных состояниях. Когда ключ К2 разомкнут, а ключ К1 замкнут,

цепочка последовательно соединенных инверторов подключается ко входному напряжению так, что обеспечивается выработка выходных напряжений $Q = D$ и $\overline{Q} = \overline{D}$ (режим прозрачности триггера). Когда ключ К1 разомкнут, а ключ К2 замкнут, схема представляет собою бистабильную ячейку на инверторах. Это состояние схемы соответствует режиму хранения. Триггер "защелкивает" в фиксаторе последнее значение входного напряжения.

Двухступенчатый триггер

Двухступенчатый триггер (рис. 3.18, *г*) функционирует как триггер, управляемый фронтом, и представляет собою последовательное соединение двух триггеров-защелок с антисинхронным тактированием.

Применение двунаправленных ключей дает возможность обогащать функциональные возможности триггеров, в частности, вводить в их схемы цепи переключения полярностей тактирующих сигналов или получать схемы с третьим состоянием выхода, для чего достаточно включить в цепь выходного сигнала ключ, управляемый сигналом разрешения OE (Output Enable).

Примеры стандартных триггеров. Примитивы триггеров в системах автоматизированного проектирования цифровых устройств

На рис. 3.19 показаны примеры стандартных JK-триггеров. Триггер ТВ1 указанных серий — двухступенчатый, свободный от "захватов", в связи с чем трактуется как управляемый фронтом. Триггеры ТВ6, ТВ9 — с внутренними задержками, одноступенчатые, переключаемые отрицательным фронтом синхросигнала.

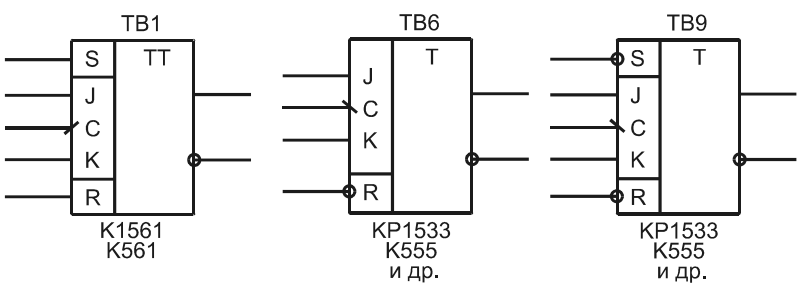


Рис. 3.19. Схемы стандартных триггеров типа JK

В справочниках функционирование триггеров обычно поясняется таблицей, в которой для каждого набора входных переменных и исходного состояния триггера указывается его новое состояние. Например, для триггера типа К561ТВ9 функционирование триггера представлено табл. 3.7.

Таблица 3.7

S	R	C	J	K	Q_H	$\overline{Q}_H$	Режим работы
L	H	X	X	X	H	L	Асинхронная установка
H	L	X	X	X	L	H	Асинхронный сброс
L	L	X	X	X	X	X	Запрещенная комбинация
H	H	↓	H	H	$\overline{Q}$	Q	Счетный
H	H	↓	L	H	L	H	Загрузка нуля (сброс)
H	H	↓	H	L	H	L	Загрузка единицы (установка)
H	H	↓	L	L	Q	$\overline{Q}$	Хранение
H	H	H	X	X	Q	$\overline{Q}$	Хранение
H	H	L	X	X	Q	$\overline{Q}$	Хранение

Символ ↓ означает отрицательный перепад синхросигнала.

Для ориентировки в табл. 3.8 даны некоторые данные о быстродействии и потребляемой мощности для микросхем JKRS-триггеров схемотехнологий ТТЛШ и КМОП.

Таблица 3.8

Тип ИС	Схемотехнология	Активный фронт	Число триггеров	Средняя задержка, нс	Максимальная частота, МГц	Потребляемая мощность, мВт
К1533ТВ15	ТТЛШ	Положительный	2	12	40	24
К1554ТВ9	КМОП	Отрицательный	2	10	140	0,035 (статическая)

Примитивы триггеров, используемые в САПР фирмы Altera, приведены на рис. 3.20.

В обозначениях примитивов триггеров входы D, T, J, K, S, R — информационные, соответствующие логике работы триггера. Входы, снабженные стрелками-треугольниками, — тактирующие, указывают на прямое динамическое управление приемом информации. Входы PRN (от Preset Negativ) и CLRN (от Clear Negativ) — L-активные (инверсные) асинхронные входы установки и сброса соответственно. Входы ENA (от Enable) разрешают или запрещают прием информации. При подаче на этот вход единицы в триггерах-защелках вводится режим прозрачности, а при

подаче нуля триггер переводится в режим хранения. В других случаях единичное значение сигнала ENA позволяет триггеру работать так, как это было бы при отсутствии входа ENA, а нулевое значение ENA переводит триггер в режим хранения, при котором входные сигналы не оказывают воздействия на состояние триггера. Однако это не относится к асинхронным сигналам установки и сброса PRN и CLRN, которые воздействуют на триггеры в любом режиме. Благодаря этой особенности при подаче сигнала ENA = 0 триггеры разных типов можно превратить в асинхронный RS-триггер, в котором роль входа S играет вход PRN, а роль входа R — вход CLRN.

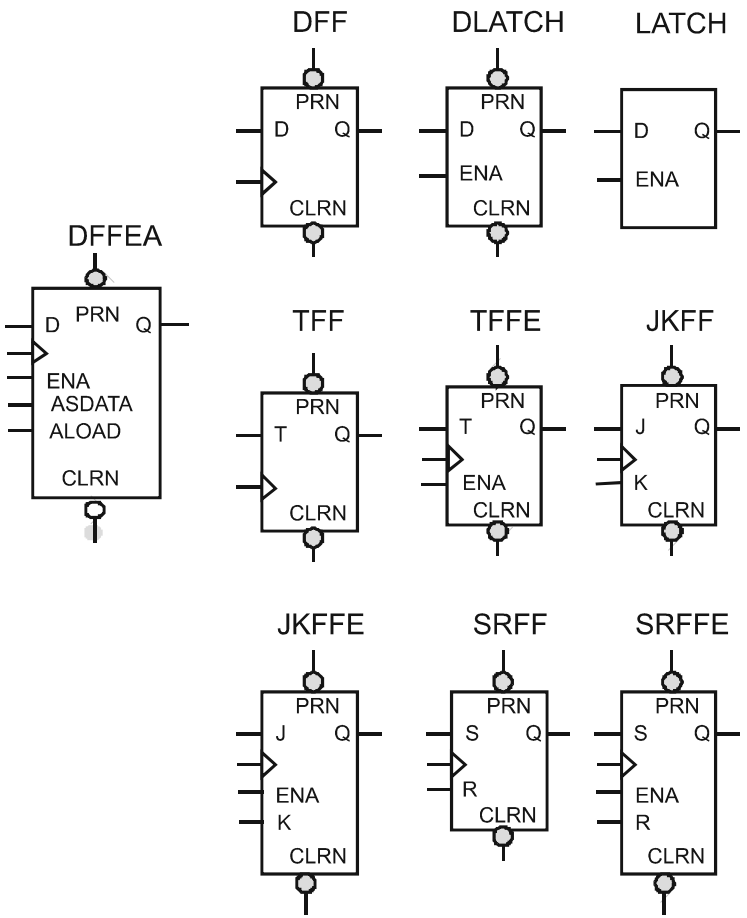


Рис. 3.20. Примитивы триггеров в САПР фирмы Altera

Примитив DFFEA снабжен дополнительными входами ADATA (асинхронной загрузки данных) и ALOAD (разрешения асинхронной загрузки). Функционирование примитива DFFEA иллюстрируется табл. 3.9.

Таблица 3.9

Входы							Выход
CLRN	**PRN**	**ALOAD**	**ASDATA**	**ENA**	**D**	**CLK**	Q_H
0	1	X	X	X	X	X	0
1	0	X	X	X	X	X	1
1	1	1	0	X	X	X	0
1	1	1	1	X	X	X	1
1	1	0	X	0	X	X	Q
1	1	0	X	1	0	Перепад 0→1	0
1	1	0	X	1	1	Перепад 0→1	1
1	1	0	X	1	X	0	Q
1	1	0	X	1	X	1	Q

§ 3.3. Примеры использования триггеров

Применение триггеров в регистрах, счетчиках, схемах памяти и других устройствах рассматривается в последующих главах. В этом разделе рассмотрены примеры использования отдельных триггеров в схемах, выполняющих несложные типовые операции, связанные главным образом со вводом данных в ЦУ.

Ввод логических сигналов от механических ключей

Ввод логических сигналов от механических ключей — одно из типовых действий, позволяющих оператору воздействовать на цифровое устройство (ЦУ).

Механические ключи имеют упругость, их коммутация — сложный процесс. После первого соударения контактов происходит ряд упругих отскоков, называемых *дребезгом контактов*, поэтому вместо однократного перепада напряжения ключи создают целую серию импульсов (рис. 3.21, *а*). Длительность упругих колебаний ключей зависит от их конструкции, обычно она лежит в диапазоне 1...10 мс. Сигнал с дребезгом нельзя вводить в цифровое устройство, т. к. он может создать множество ложных переключений.

Для получения сигнала, "очищенного" от дребезга контактов, принимают специальные меры — программные или схемные. Программный метод — введение пау-

зы между нажатием ключа и использованием формируемого ключом сигнала (серия пустых команд NOP в программе, выполняемой системой). В схемных методах борьбы с дребезгом контактов используются свойства триггеров, причем очищенный от дребезга сигнал формируется сразу, ожидания окончания дребезга не требуется.

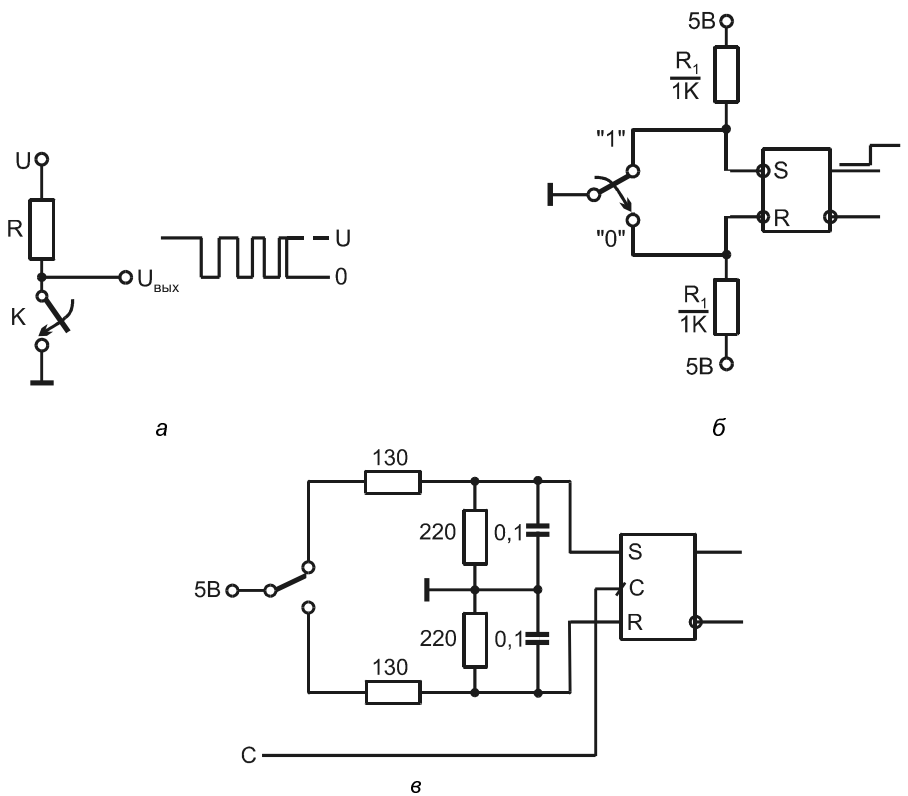


Рис. 3.21. Сигнал, формируемый механическим ключом (*а*), и схемы устранения дребезга контактов (*б*, *в*)

С помощью триггеров выходное напряжение ключа очищается от паразитных колебаний и превращается в стандартный логический сигнал. При работе с однополюсными ключами на два положения (рис. 3.21, *б*) верхнее положение ключа устанавливает триггер, т. к. на входе нулевое напряжение, а на входе — высокое, задаваемое от источника 5В через резистор R1 (номиналы напряжений и сопротивлений приводятся здесь с ориентацией на схемотехнику стандартных ТТЛ). Нижнее положение ключа ведет к сбросу триггера (на входе $\overline{R}$ нулевое напряжение, на входе $\overline{S}$ — высокое). При изменении состояния ключа первое же соударение приводит триггер в соответствующее состояние, а при отскоке ключа, когда он находится в воздухе, оба входа триггера получают пассивные сигналы логической единицы (высокие напряжения от цепочек "источник-резистор"), т. е. триггер попадает

в режим хранения уже установленного правильного состояния. С помощью этой схемы производится асинхронный ввод сигнала от ключа. Резисторы R_1 могут быть достаточно высокоомными, т. к. через них замыкаются только относительно малые токи входов триггера, находящихся в единичном состоянии.

Тактируемый ввод показан на рис. 3.21, *в*. Видоизменения схемы объясняются тем, что синхронный триггер имеет прямые входы установки и сброса, и тем, что для повышения помехоустойчивости схемы добавлены фильтрующие конденсаторы. Принцип работы схемы сохраняется. Первый же тактовый импульс, пришедший после переключения ключа, формирует выходной сигнал. Для схем типа ТТЛ(Ш) резисторы низкоомны, поскольку через них замыкаются входные токи триггера при обоих значениях логического сигнала на них, в том числе и значительные по величине токи $I_{вх\,0}$. Отношение сопротивлений плеч делителей обеспечивает подачу на входы триггера необходимых уровней U_1.

Иногда сигналы от механических ключей вводят с помощью более простых схем, содержащих интегрирующие RC-цепочки. В этом случае переходный процесс при идеальном замыкании ключа имел бы форму экспоненты, а в реальной ситуации эта экспонента будет прерываться горизонтальными участками, соответствующими отскокам ключа, когда он находится в воздухе, и ток емкости нулевой. Так как подобная кривая монотонна, после достижения ею порогового значения логический элемент переключается однократно.

Синхронизаторы

В синхронных цифровых устройствах (ЦУ) моменты восприятия данных и временные интервалы, в которых допускается их изменение, четко определены. Данные на входах тактируемых триггеров на интервалах предустановки и выдержки в окрестностях фронтов синхросигналов должны быть неизменны во избежание метастабильных состояний. Однако внешние сигналы могут поступать на входы ЦУ в любой момент, нарушая требования предустановки и выдержки и порождая метастабильные состояния. В подобных ситуациях невозможно избавиться от метастабильных состояний, можно лишь снижать вероятность их появления, вводя в ЦУ асинхронные сигналы с помощью синхронизаторов.

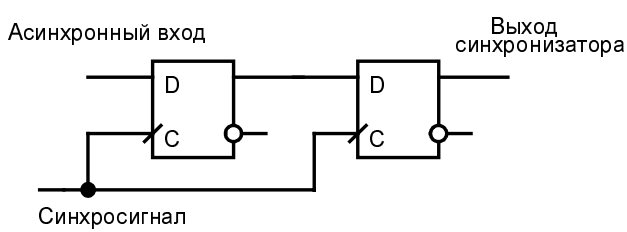


Рис. 3.22. Схема ждущего синхронизатора (типа Brute-Force Synchronizer)

Широко применяемые синхронизаторы *ждущего типа* (в английской терминологии Brute-Force Synchronizers) представляют собою цепочку D-триггеров с непосредственными связями между ними.

В синхронизаторе с двумя триггерами (рис. 3.22) первый триггер принимает асинхронный сигнал и при его попадании в запрещенную область входит в метастабильное состояние. Частота появления метастабильностей определяется параметрами триггера и частотой тактирования системы. В зависимости от близости к точке равновесия длительность метастабильности на выходе первого триггера различна. Наиболее вероятны короткие метастабильности, чем длительнее метастабильность, тем меньше вероятность ее возникновения (см. рис. 3.8).

Второй триггер воспринимает сигнал от первого на следующем такте, т. е. через время T, равное периоду синхросигналов. За это время *часть состояний метастабильности* первого триггера *успеет затухнуть* и не повлияет на второй триггер. Так как вероятность появления коротких метастабильностей значительно больше, чем вероятность появления длительных, то большая часть всех метастабильностей затухает. Такова идея *снижения вероятности* аномалий вводимого сигнала. При удлинении цепочки вероятность появления метастабильности на ее выходе уменьшается. В целом, степень уменьшения этой вероятности зависит от величины nT, где n — число триггеров в цепочке. Плата за получаемый результат — задержка вводимого сигнала — он появляется *после прохождения n активных фронтов синхросигнала.*

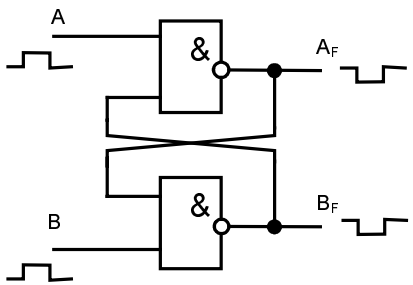
Арбитры

Арбитры определяют порядок событий для двух входных сигналов. Выяснение того, какое событие произошло первым, необходимо при решении таких задач, как предоставление ресурса тому или другому претенденту.

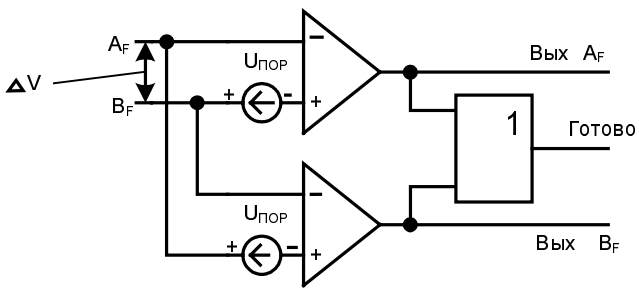
Арбитром может служить асинхронный RS-триггер. Такой триггер, реализованный на элементах И-НЕ (рис. 3.23, *а*), при A = B = 0 имеет единичные состояния обоих выходов. Пусть первым появляется сигнал A. Условие A = 1 приводит к переходу выхода A_F в нулевое состояние (A_F — аббревиатура от AFirst). Это блокирует какие-либо изменения выходов при изменениях входа B, и схема останется в состоянии, заданном первым сигналом A ($A_F = 0$, $B_F = 1$). Точно так же, если первым появится сигнал B, схема сформирует сигнал $B_F = 0$ и перейдет в состояние $B_F = 0$, $A_F = 1$. Выработка одновременно обоих сигналов исключается.

Что произойдет, если оба сигнала появятся почти одновременно? В этом случае воздействие на триггер становится слабо определенным, и в нем возникает состояние метастабильности. Разность времен появления сигналов Δt создает первоначальную малую разность напряжений на выходе триггера ΔV_1, пропорциональную Δt. Затем разность напряжений на выходах триггера ΔV нарастает по экспоненциальному закону с некоторой постоянной времени τ, т. е. $\Delta V = \Delta V_1 \exp(t/\tau)$. Для принятия решения (т. е. для нарастания ΔV до определенного значения) потребуется

большее или меньшее время в зависимости от значения ΔV_1, т. е., в конечном счете, от значения Δt.



а



б

Рис. 3.23. Схема арбитра на элементах И-НЕ (*а*)
и схема фиксации момента принятия решения при попадании арбитра
в состояние метастабильности (*б*)

Определить момент принятия решения можно с помощью схемы, показанной на рис. 3.23, *б. З*авершение процесса отмечается моментом отклонения ΔV от порогового напряжения $U_{пор}$ на заданную величину, когда помехи уже не могут вернуть триггер в состояние метастабильности. При $|\Delta V| > |U_{пор}|$ срабатывает верхний или нижний компаратор. Момент принятия решения с помощью элемента ИЛИ фиксируется сигналом "Готово". Интересно отметить, что, несмотря на кажущуюся сложность схемы рис. 3.23, *б*, имеющей "плавающие " источники опорных напряжений, ее реализация на транзисторном уровне очень проста. Вся схема может быть выполнена на четырех транзисторах, так как роль "встроенного " плавающего источника $U_{пор}$ играют смещения потенциалов, свойственные характеристикам транзисторов.

Фиксация момента принятия решения является особенно полезной для асинхронных режимов (с процедурой рукопожатия).

§ 3.4. Тактирование и синхронизация. Общие сведения

Как уже отмечалось, основным методом борьбы с опасными состязаниями сигналов в цифровых устройствах в настоящее время является синхронизация.

Примечание

В современной литературе термин "синхронизация" применяется к процессам, имеющим существенные отличия. С содержательной точки зрения следовало бы использовать два понятия: *"тактирование"*, подразумевающее выработку командных сигналов для выполнения действий, и *"синхронизация"*, относящееся к временному согласованию взаимодействующих сигналов. В этой главе различие между понятиями "тактирование" и "синхронизация" отмечается, но не всегда, поскольку сохранена и общепринятая терминология, в которой различия между тактированием и синхронизацией стираются.

Тактирование процессов

Проблемы тактирования в ЦУ и методы их решения перечислены на рис. 3.24.

Рис. 3.24. Классификация проблем тактирования и методов их решения

В современных ЦУ применяют два подхода к формированию тактовых сигналов в устройствах-приемниках информации:

- с передачей в приемник тактовых сигналов (Source-Synchronous Clocking);
- с выработкой тактовых сигналов самим приемником путем их "извлечения" из передаваемой информации (Clock-Data Recovery).

Системы с передачей в приемник тактовых сигналов

В этом традиционном варианте сгенерированные *тактовые* (тактирующие) сигналы распределяются по всем частям устройства и разрешают или запрещают прием данных элементам памяти — синхронным триггерам. Тактирование определяет *последовательность и моменты выполнения действий при обработке информации*. Темп обработки задается частотой тактовых импульсов ТИ, называемых в литературе также *синхросигналами.* В англоязычной литературе тактовый сигнал обозначается термином *Clock*.

Система с единым тактовым генератором, непосредственно питающим цепи распределения ТИ, все участки которых эквипотенциальны, соответствует идеалу, но для более или менее сложных устройств практически неосуществима. Для управления большим числом элементов памяти пришлось бы делать единые цепи тактирования мощными, и они создавали бы *помехи* недопустимо высокого уровня. Для подведения тактовых импульсов к элементам памяти на кристалле, печатной плате и в других конструкциях создаются специальные цепи, обычно представляющие собою разветвленную сеть той или иной конфигурации с буферными усилителями.

Цепи распределения тактирующих сигналов являются одновременно *самыми скоростными и самыми нагруженными*. Частота сигналов в них максимальна для данной системы, т. к. они изменяются в каждом такте. Нагрузка очень велика, поскольку к линиям тактирования подключаются порою многие тысячи элементов памяти, а сами линии имеют большую длину (даже на кристаллах длина тактирующих цепей может доходить до единиц сантиметров). Обе причины создают для линий тактирования емкостную нагрузку, которая может на много порядков превышать нагрузку других цепей. Большая емкостная нагрузка влечет за собою значительные задержки распространения ТИ.

Паразитные задержки в линиях передачи ТИ исключают эквипотенциальность всех участков этих линий и в зависимости от расстояний от генератора ТИ меняется и фаза синхросигнала. Чем меньше период ТИ, тем больший *фазовый сдвиг* создают одни и те же задержки (фаза определяется относительной величиной сдвига, т. е. тем, какую часть периода составляет сдвиг), так что особенно острую проблему создают задержки тактирующих сигналов для быстродействующих устройств. Фазовые сдвиги тактирующих импульсов *(Clock Skew)* нарушают предусмотренные проектом длительности этапов обработки информации или даже их очередность. Они либо жестко ограничиваются при проектировании и конструировании устройства, либо заставляют применять специальные усложненные способы тактирования систем.

Выработка тактовых сигналов в приемнике данных

В последнее время возникли новые подходы к тактированию процессов в сложных системах. В частности, в технологиях CDR (Clock-Data Recovery), CDS (Clock-Data Synchronization) *сигналы тактирования создаются в приемниках с помощью самих передаваемых данных.* Для правильного восприятия данных устройством-приемником необходимо лишь правильное *взаимное* положение тактирующих и информационных сигналов. Информационные сигналы при их распространении сдвигаются во времени. Если тактирующие сигналы сдвинутся на ту же величину, то правильное функционирование устройства сохранится. Целью блоков CDR, CDS является согласование фазы вырабатываемых ими синхросигналов с приходящими в приемник данными.

Синхронизация сигналов

Области синхронизации. Фазовые сдвиги сигналов приводят к их разбиению на *области синхронизации.* К одной и той же области *(Clock Domain)* относятся сигналы, синхронизируемые одними и теми же тактовыми импульсами (точнее, импульсами, фазовыми расхождениями которых можно пренебречь). Все сигналы одной области могут изменяться только по команде одних и те же ТИ, и все информационные сигналы остаются неизменными (стабильными) в *запрещенном окне, т. е. на интервалах предустановки и выдержки* вблизи фронта синхросигнала. Сигналы одной области синхронизации без проблем взаимодействуют друг с другом в логических структурах и схемах памяти.

> ***Примечание***
>
> Выделение различных областей синхронизации не всегда является вынужденным следствием задержек ТИ. Иногда для разных устройств специально устанавливают различную скорость работы, т. е. и разные частоты ТИ. Это также порождает различные области синхронизации.

Так или иначе, в силу необходимости или по желанию проектировщика, но сигналы синхронных ЦУ могут оказаться в разных областях синхронизации и возникает задача их взаимодействия — *пересечения информационными сигналами границ между областями синхронизации или, иначе говоря, ввода сигналов из иной области синхронизации.* Такая задача решается более или менее сложно в зависимости от типа синхросигналов разных областей.

Ввод сигналов. Среди задач синхронизации (рис. 3.25) можно выделить ввод сигналов и арбитраж. Сложность задачи ввода зависит от характера сигналов. Сложнее всего дело обстоит с вводом асинхронных сигналов, появляющихся в произвольные моменты времени. Для "полусинхронных" сигналов (мезохронных, плезиохронных и периодических), моменты изменений которых в той или иной мере ограничены, добиться успеха проще, хотя схемы ввода могут быть и не очень простыми.

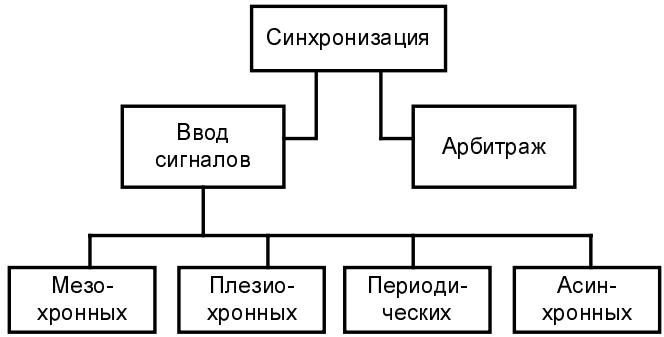


Рис. 3.25. Задачи синхронизации, решаемые при построении цифровых устройств

Арбитраж. Средства арбитража позволяют ответить на вопрос "Кто первый? " применительно к двум сигналам. Такая задача принадлежит к типовым при обработке информации. Проблемы арбитража во многом сходны с проблемами тактирования, в частности, в связи с явлениями метастабильности триггеров.

§ 3.5. Тактирование сигналами, выработанными генератором

Общие сведения. Возможные решения

Задача такого тактирования — выделить с помощью тактовых сигналов определенные интервалы времени для обработки данных в комбинационных цепях и их восприятия элементами памяти. Идентичные в идеале тактовые сигналы передаются всем тактируемым элементам.

В рамках указанной проблемы с учетом типа применяемых элементов памяти (триггеров) решаются задачи:

- выбора типа (концепции) тактирования (глобальное, конвейерное и т. д.);
- выбора фазности системы тактирования, т. е. числа используемых тактовых последовательностей;
- выбора между разомкнутыми и замкнутыми системами;
- обеспечения надлежащего качества тактовых сигналов (стабильности частоты, допустимого фазового сдвига, приемлемой формы импульсов и т. д.).

Концепции тактирования

Применяются различные *концепции* тактирования. При *глобальном сквозном* тактировании *(Global Clocking)* период ТИ определяется временем обработки данных устройством в целом. После подачи на входы устройства очередного набора данных выдерживается время, необходимое для их обработки и фиксации резуль-

тата на выходе устройства, и лишь после этого допускается подача нового набора аргументов на входы устройства. При глобальном тактировании все сигналы подчиняются единому тактированию.

Вторая концепция — *конвейерное* тактирование *(Pipeline Clocking)*, эффективное при обработке данных, продвигаемых по тракту обработки в одном направлении. В этом случае тракт разбивается на ступени, что позволяет повысить частоту тактовых сигналов, поскольку время обработки информации в пределах ступеней меньше, чем время обработки трактом в целом, а новые данные на вход ступени можно подавать по мере обработки в ней предыдущих, не дожидаясь завершения работы последующих ступеней.

Тактирование конвейера в свою очередь может осуществляться импульсами глобального типа (т. е. общими для всех ступеней, но увеличенной частоты ввиду разбиения устройства на ступени) или с выработкой для каждой ступени своих ТИ с оптимальным расположением относительно обрабатываемых данных, т. е. с формированием для ступеней своих областей синхронизации.

Фазность тактирования

В цифровых устройствах для управления процессами во времени могут применяться как одна, так и несколько тактовых последовательностей. Соответственно их числу системы тактирования можно разделить на *однофазные, двухфазные, многофазные* (с числом тактовых последовательностей 4, 6, 8 и т. д.).

С развитием техники возрастают частоты тактирования и изменяются схемы триггеров, что ведет к смене популярности систем разной фазности. Сравнительно недавно (в эпоху ТТЛ(Ш)) была популярна однофазная система тактирования с одноступенчатыми триггерами, управляемыми фронтами. Вследствие повышения тактовых частот обострилась проблема фазовых сдвигов синхросигналов, а триггеры с управлением фронтом в схемотехнике КМОП стали выполняться как двухступенчатые. Эти обстоятельства поддерживают двухфазное тактирование, которое стало наиболее распространенным в устройствах небольшой и средней сложности и в микропроцессорной технике. Многофазность тактирования характерна для систем высокой сложности и быстродействия, где идут на усложнения ради максимального использования ресурсов времени.

Разомкнутые и замкнутые системы тактирования

Для систем тактирования (в первую очередь глобальных и высокочастотных) важнейшее значение имеет борьба с фазовыми сдвигами тактовых сигналов (Clock Skew). С этой точки зрения системы тактирования можно разделить на следующие типы:

- *разомкнутые* (Open-Loop), не имеющие средств автоматического управления параметрами тактирующих импульсов;

- *замкнутые* (Closed-Loop), в которых автоматически измеряется фазовый сдвиг тактирующих импульсов в разных частях системы и подбираются задержки в цепях их передачи с целью борьбы с их расфазированием (эта задача решается блоками PLL и DLL).

Медленные и быстрые сдвиги фаз ТИ

Фазовый сдвиг тактовых импульсов можно разделить на следующие составляющие:

- *статическую* (медленную), к которой относится уже упоминавшийся Clock Skew;
- *динамическую*, к которой относится так называемое *дрожание фронтов (*Jitter), вызываемое высокочастотными помехами и другими быстроизменяющимися факторами.

Обобщенный тракт обработки данных

Обобщенный тракт тактируемой обработки данных можно представить чередованием комбинационных цепей КЦ и элементов памяти ЭП. Такая структура отражает работу ЦУ как при пространственном чередовании КЦ и ЭП (рис. 3.26, *а*), так и при последовательном выполнении операций в разных тактах на одном и том же оборудовании, что свойственно операционным блокам процессоров, содержащим АЛУ и блок регистров (рис. 3.26, *б*). КЦ преобразуют данные по тем или иным логическим зависимостям, а ЭП принимают данные *после окончания переходных процессов*, т. е. установления на выходах КЦ истинных (действительных, стабильных) значений сигналов.

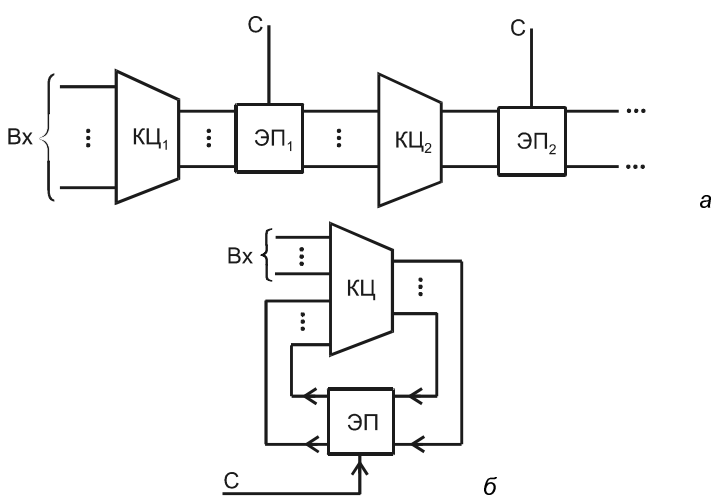


Рис. 3.26. Обобщенные структуры тракта обработки информации в цифровых устройствах (*а*, *б*)

В КЦ пути от входов к различным выходам не идентичны. Для расчета системы тактирования нужно оценивать минимальную и максимальную задержки сигналов в КЦ. Для оценки минимальной задержки следует учесть минимальные задержки элементов (т. е. учесть разброс задержек) и найти самый короткий (в смысле времени его прохождения) путь от входов к одному из выходов КЦ. Самый длинный путь сигнала к выходу КЦ оценивается с учетом максимальных задержек элементов. Таким образом, должны быть определены задержки $t_{кц.min}$ и $t_{кц.max}$. Естественно, следует учитывать и задержки в связях, роль которых в современных условиях может оказаться доминирующей.

Тракт обработки информации разбивается на секции (ступени), в которые входят КЦ, ЭП и связи, по которым передаются сигналы. Время передачи сигналов от одной секции до другой зависит и от задержек в элементах памяти (триггерах), поэтому для проектирования системы тактирования необходимы также сведения о максимальных и минимальных задержках триггеров $t_{тг.min}$ и $t_{тг.max}$.

Параметры тактовых импульсов

Период тактовых импульсов T складывается из длительностей импульса и паузы.

Длительности импульса и паузы

Длительность импульса $t_и$ должна быть достаточной для надежной записи информации в триггер, этот параметр задается в паспортных данных триггера. Обозначив его через t_{wc}, можем записать условие:

$$t_и \geq t_{wc}.$$

Новое состояние триггеры гарантированно примут после фронта ТИ по истечении максимальной из задержек $t_з^{01}$ и $t_з^{10}$ их переключения. Параметры t_{wc} и задержки переключения триггеров зачастую близки, но могут и отличаться друг от друга. Разность $\max\{t_з^{01}, t_з^{10}\} - t_{wc}$ обозначим через $\Delta t_{тг}$ (рис. 3.27).

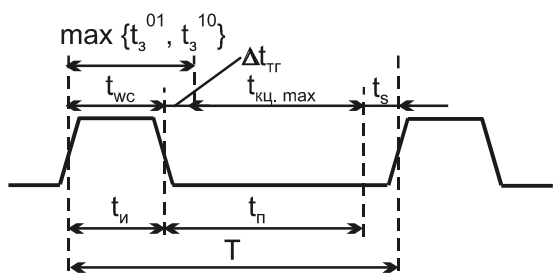


Рис. 3.27. Иллюстрация к определению параметров синхроимпульсов

Приняв новое состояние, триггеры тем самым формируют на входах КЦ новые значения аргументов. После этого, до нового приема данных, должно пройти время, достаточное для прохождения сигнала по самому длинному пути в КЦ плюс

время предустановки t_s (задержку в связях учитываем, приплюсовывая ее к задержке КЦ). Поэтому для длительности паузы имеем соотношение:

$$t_п \geq \Delta t_{тг} + t_{кц.max} + t_s.$$

Минимальный период тактовых импульсов

$$T_{min} = t_{и.min} + t_{п.min},$$

а их частота $f_{max} = 1/T_{min}$.

На интервале от $t_{кц.min}$ до $t_{кц.max}$ после переключения триггеров выходные сигналы КЦ не соответствуют ни старому, ни новому значению (данные нестабильны).

Задержки сигналов в линиях связи зависят от топологии соединений. Поэтому на ранних стадиях проектирования расчет параметров ТИ может быть только ориентировочным.

Стабильность частоты

В системах тактирования часто используют генераторы с кварцевой стабилизацией, позволяющие без затруднений обеспечить относительную нестабильность частоты порядка $10^{-4}...10^{-5}$. В более простых генераторах нестабильность частоты существенно выше, что приводит к потере быстродействия устройства. Действительно, частоту синхроимпульсов можно выразить соотношением: $f = f_0(1 \pm \delta f)$, где f_0 — номинальное значение частоты и $\delta f = \Delta f/f_0$ — ее относительный уход (допуск). Ширина поля допуска равна $2\delta f$. Даже максимальная частота не должна превышать допустимого значения, т. е. $f_0(1 + \delta f) \leq f_{доп}$. Если же частота будет равна нижнему пределу $f_0(1 - \delta f)$, то она окажется на $2\delta f$ ниже допустимой. То есть *возможная относительная* потеря быстродействия устройства из-за нестабильности частоты синхроимпульсов составляет $2\delta f$.

Крутизна фронтов

Определенные требования предъявляются и к крутизне фронтов ТИ. Она не должна снижаться ниже допустимого предела. Причины этого ограничения заключаются в том, что при слишком пологих фронтах выходные цепи элементов могут слишком долго оставаться под действием сквозных токов и, во-вторых, в том, что при малой крутизне фронтов ТИ разброс порогов срабатывания ЭП приводит к большому разбросу моментов их переключения. Особенно важно это обстоятельство для схем на элементах типа КМОП, для которых характерен повышенный разброс порогов срабатывания. Разброс моментов срабатывания (т. е. как бы разброс моментов поступления тактовых сигналов на разные элементы, питаемые одним и тем же синхросигналом), определяется выражением

$$t_2 - t_1 = (U_{пор2} - U_{пор1})/К,$$

где К — крутизна фронта синхроимпульса; $U_{пор2}$ и $U_{пор1}$ — пороговые напряжения элементов, для которых вычисляется эквивалентный сдвиг синхросигналов (раз-

брос пороговых напряжений элементов на МОП-транзисторах может доходить приблизительно до одной четвертой от величины логического перепада).

Для показа опасности сбоев из-за малой крутизны фронтов синхроимпульсов рассмотрим передачу данных в цепочке триггеров (сдвиг слова). В этой цепочке (рис. 3.28) очередной синхроимпульс должен передавать состояние триггера соседу справа. Предположим, что пороговое напряжение триггера T_i минимально, а триггера T_{i+1} максимально. Тогда триггер T_i переключится раньше, чем придет сигнал приема данных для триггера T_{i+1}, и этот триггер не сможет принять старое состояние от соседа слева — информация будет утеряна.

С другой стороны, нежелательна и излишняя крутизна фронтов, создающая *повышенные импульсные помехи*. Это тоже существенное обстоятельство, и при проектирования сложных схем на кристалле обычно стремятся к работе с пологими фронтами везде, где это допустимо. Часто применяют программируемые буферы, в которых регулируется крутизна формируемых сигналов.

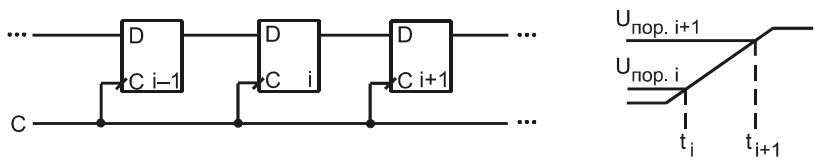


Рис. 3.28. Схема передачи данных в цепочке синхронных триггеров

§ 3.6. Структура и элементы систем тактирования

Структура системы тактирования

Система тактирования (рис. 3.29) содержит задающий генератор ЗГ, формирователь опорных сигналов ФОС и размножитель сигналов РС. Блок ФОС (распределитель тактов РТС) служит для выработки необходимого числа импульсных последовательностей в зависимости от фазности системы тактирования. О блоке РС говорится далее.

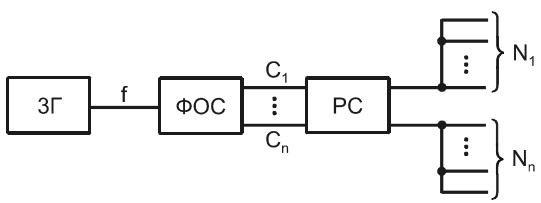


Рис. 3.29. Обобщенная структура системы тактирования

Выбор конкретного варианта первичного (задающего) генератора тактовых сигналов зависит от требуемой стабильности частоты. При допустимой нестабильности

порядка процентов можно применить простые схемы *кольцевых генераторов*, построенные на основе логических элементов и элементов задержки (см. *§ 1.7*), *релаксационные генераторы* с RC-цепочками ([10] и др.) или генераторы с LC-контурами. При жестких требованиях к стабильности частоты применяют генераторы с *кварцевой стабилизацией*, обладающие высокими качественными показателями. При этом кварцевый резонатор является внешним элементом, подключаемым к выводам микросхемы.

Кварцевые генераторы

Структура генератора. По своей структуре классический кварцевый генератор является усилительным звеном с положительной обратной связью, в котором возбуждаются периодические колебания (рис. 3.30, *а*). Частота колебаний определяется резонансными частотами цепи обратной связи с кварцевым резонатором, представляющим собою колебательный LC-контур с потерями. Усилитель компенсирует потери в контуре, благодаря чему в схеме возникают незатухающие колебания. Усилитель имеет коэффициент усиления K_U и сдвигает сигнал по фазе на угол $\varphi_{ус}$. К выходу усилителя подключено сопротивление нагрузки R_L (в частности, входы формирователей тактовых импульсов), входное сопротивление усилителя обозначено через $R_{вх}$.

Для существования в схеме незатухающих колебаний должны выполняться условия:

- баланс фаз;
- баланс амплитуд.

Первое условие означает, что колебания возникают только при точном совпадении фазы напряжения обратной связи с фазой входного напряжения усилителя. Количественно это условие формулируется следующим образом

$$\varphi_{ус} + \varphi_{ос} = 2\pi k \quad (k = 0, 1, 2, \ldots),$$

где $\varphi_{ос}$ — фазовый сдвиг, вносимый цепью обратной связи.

Второе условие означает, что генератор может возбуждаться только при достаточном усилении в петле обратной связи, при котором усилитель полностью компенсирует потери в контуре. При этом коэффициент петлевого усиления $G = K_U\beta = 1$ (β — коэффициент обратной связи, т. е. та часть выходного напряжения, которая через цепь обратной связи подается на вход усилителя). При $G = 1$ генерируются незатухающие гармонические колебания. Если $G < 1$, то колебания затухают, если $G > 1$, то амплитуда колебаний нарастает.

В реальных условиях петлевое усиление не может задаваться абсолютно точно, так как оно зависит от изменяющихся во времени и под действием внешних факторов параметров схемных элементов. Чтобы избежать недопустимой ситуации с $G < 1$, следует принять соотношение $G > 1$, создав тем самым запас усиления, не позволяющий попадать в режим затухающих колебаний. Это же условие обеспечивает самовозбуждение генератора. При $G > 1$ рост амплитуды не будет бесконечным, он ограничится диапазоном выходных напряжений усилителя, а форма выходного напряжения исказится, приблизившись к трапецеидальной ("обрежутся" верхушки

синусоиды). Из-за попадания в области насыщения уменьшается K_U, и это объясняет стабилизацию амплитуды сигнала (устанавливается среднее значение G = 1).

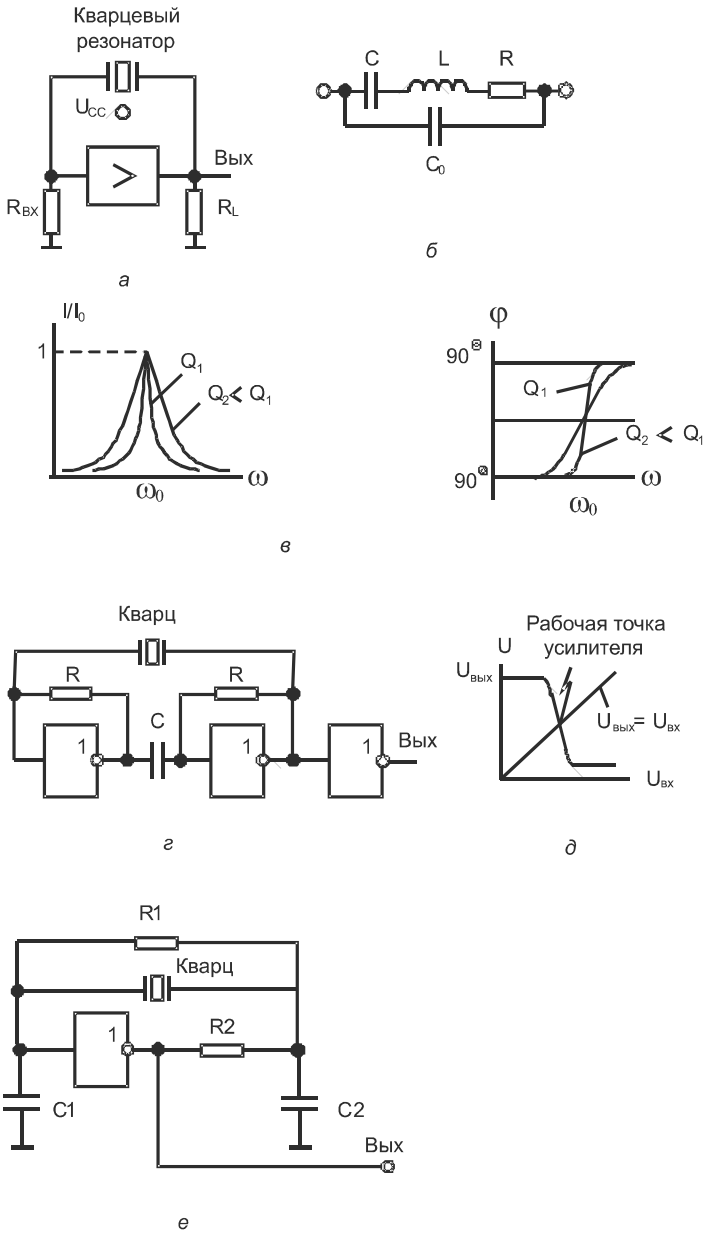


Рис. 3.30. Структура кварцевого генератора (*а*), схема замещения кварцевого резонатора (*б*), резонансные кривые (*в*), схема кварцевого генератора, построенного на основе инверторов (*г*), вывод рабочей точки инвертора в область усилительного режима (*д*), вариант схемы кварцевого генератора (*е*)

Поскольку в конечном счете нужно вырабатывать синхросигнал с крутыми фронтами, его трапецеидальная форма оказывается даже полезной. Однако при искажениях сигнала, формируемого генератором, снижается стабильность его частоты, поэтому может оказаться желательным поддержание близких к гармоническим колебаний и формирование сигналов с крутыми фронтами, схемами, подключаемыми к выходу генератора.

Кварцевый резонатор. В схеме замещения кварцевого резонатора (рис. 3.30, *б*) величины индуктивности L и емкости C определяются пьезоэлектрическими свойствами кварцевой пластинки, закрепленной между электродами. Малое омическое сопротивление R отображает потери в контуре резонатора. Величина C_0 — емкость выводов резонатора (электродов и проводов, подводимых к кварцу). Частоту колебаний генератора определяют явления резонансов в кварцевой пластине. Основным является *резонанс напряжений* в цепи CLR (последовательный резонанс). Наличие емкости C_0 ведет к появлению второго (параллельного) резонанса, однако параметр C_0 менее точен, чем параметры кристалла кварца, поэтому предпочтителен последовательный резонанс, который состоит в следующем. Импеданс цепи из последовательно включенных индуктивности L, емкости C резистора R выражается формулой:

$$Z = j\omega L + 1/j\omega C + R = j\omega L - j/\omega C + R = j(\omega L - 1/\omega C) + R.$$

Резонанс происходит на частоте $\omega_0 = 1/(LC)^{1/2}$, определяемой из условия равенства модулей индуктивного и емкостного сопротивлений:

$$1/(\omega_0 C) = \omega_0 L.$$

Участок CL при $\omega = \omega_0$ будет как бы закорочен, поскольку напряжение на нем будет нулевым. Цепь будет проявлять себя как малое активное сопротивление R. Это и есть *резонанс напряжений*.

На резонансной частоте сопротивление контура минимально и равно R. При отклонении от частоты резонанса индуктивное и емкостное сопротивления перестают компенсировать друг друга, сопротивление цепи и фазовый сдвиг растут, причем тем быстрее, чем больше отношение реактивных сопротивлений к омическому — *добротность контура Q*, причем

$$Q = \omega_0 L/R = 1/(\omega_0 CR).$$

Крутизна фазовой и острота амплитудной характеристик контура определяются параметром его добротности.

На рис. 3.30, *в* показаны резонансная кривая (отношение тока в цепи к максимальному току при резонансе) и фазовая характеристика для двух значений добротности контура, причем $Q_1 > Q_2$. Чем острее амплитудно-частотная характеристика и чем круче фазовая, тем стабильнее будет частота генератора. Это легко видеть из достаточно наглядных представлений, например, связанных с формой фазовой характеристики. Нестабильность частоты вызывается изменениями параметров схемы. Пусть в результате температурных воздействий, старения элементов или по каким-либо иным причинам фазовый сдвиг, вносимый усилителем, изменился на $\Delta\varphi$. Для сохранения баланса фаз это изменение сдвига будет скомпенсировано соответствующим изменением фазового сдвига в цепи обратной связи. Изменение

фазового сдвига в цепи обратной связи определится как произведение крутизны фазовой характеристики при резонансе на девиацию частоты, т. е.

$$\Delta\varphi = (d\varphi/d\omega)\Delta\omega = K_\phi\,\Delta\omega,$$

или

$$\Delta\omega = \Delta\varphi/\,K_\phi,$$

где K_ϕ — крутизна фазовой характеристики в области резонанса.

При большой крутизне фазовой характеристики для изменения фазы на $\Delta\varphi$ потребуется сдвиг частоты генерации на малое значение $\Delta\omega$, т. е. стабилизация частоты будет эффективной. Причина высокой стабильности частоты кварцевых генераторов состоит в том, что *добротности колебательных контуров кварца несоизмеримо выше, чем добротности контуров каких-либо других типов.*

Пример схемы генератора. На рис. 3.30, *г* показана схема кварцевого генератора, реализованного на типовых инверторах. Инверторы работают в режиме усилителей, что обеспечивается резисторами R. Благодаря разделительной емкости C каждый из инверторов имеет независимый режим по постоянному току, причем, пренебрегая падением напряжения на резисторах (из-за малости входных токов усилителей), можно считать, что для инверторов рабочая точка соответствует условию $U_{вых} = U_{вх}$. В этом случае, как видно из рис. 3.30, *д*, инвертор имеет усилительные свойства ($|\Delta U_{вых}/\Delta U_{вх}| >> 1$). Последовательное включение двух инвертирующих каскадов образует неинвертирующий усилитель, необходимый для введения положительной обратной связи через кварцевый резонатор. Третий инвертор играет роль формирователя, увеличивающего крутизну фронтов синхросигналов.

Для подстройки частоты в небольших пределах последовательно с кварцем включают конденсатор относительно большой емкости C_S, изменяющий частоту генератора на величину

$$\Delta f = f\cdot C/[2(C_0 + C_S)].$$

На рис. 3.30, *е* приведен вариант схемы генератора, построенной на одном инверторе, в которой для получения требуемого фазового сдвига в петлю обратной связи наряду с одним инвертором включены две RC-цепочки. Существует трактовка этой схемы как схемы с элементом задержки в цепи обратной связи инвертора. Такая схема часто используется, в том числе для генерации высоких частот.

Вторичные тактовые сигналы

На выходе первичного генератора обычно формируется одна импульсная последовательность с идентичными по длительности импульсом и паузой. В то же время для тактирования могут понадобиться как несколько сдвинутых во времени тактирующих последовательностей, так и несимметричные импульсы (обычно с длительностью паузы, превышающей длительность импульса). Для получения таких (вторичных) тактовых сигналов из исходной последовательности применяют специальные схемы.

На рис. 3.31, *а* показана схема, вырабатывающая две тактовые последовательности с относительной длительностью импульсов 1/4 (т. е. со скважностью 4), сдвинутые

относительно друг друга на 1/2 периода (такие сигналы характерны, например, для систем двухфазного тактирования). Частота синхросигналов задается генератором опорной частоты (сигналом С).

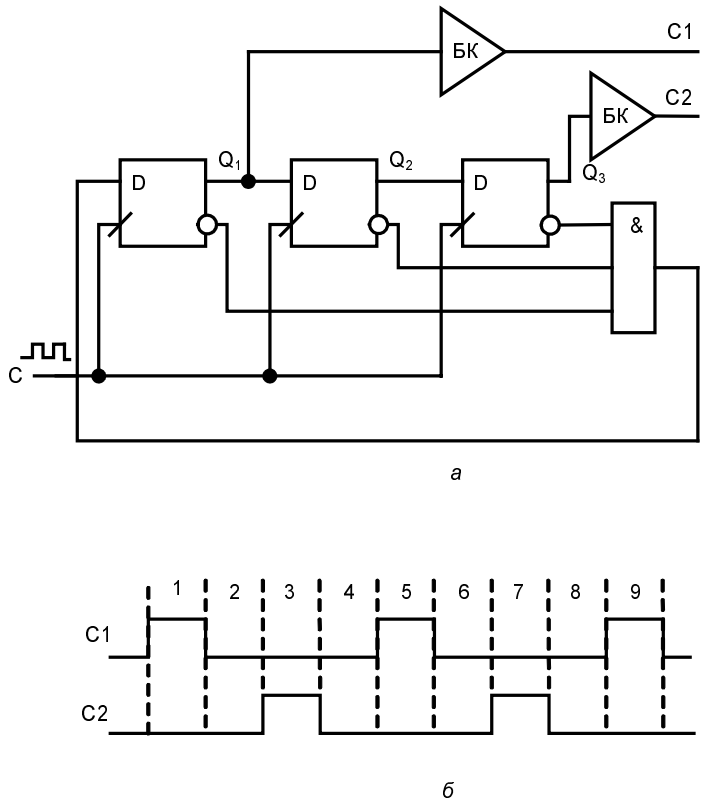


Рис. 3.31. Схема генератора вторичных тактирующих сигналов (*а*) и временные диаграммы его выходных сигналов (*б*)

Функционирование схемы хорошо иллюстрируется наблюдением за сменой ее состояний (табл. 3.10).

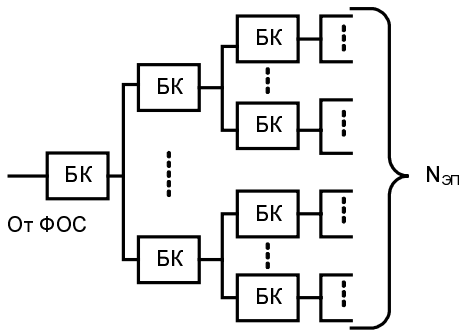
Таблица 3.10

Выходы D-триггеров	**Состояния схемы на выходах D-триггеров**				
	Начальное состояние	**1 такт**	**2 такт**	**3 такт**	**4 такт**
Q_1 Q_2 Q_3	0 0 0	1 0 0	0 1 0	0 0 1	0 0 0

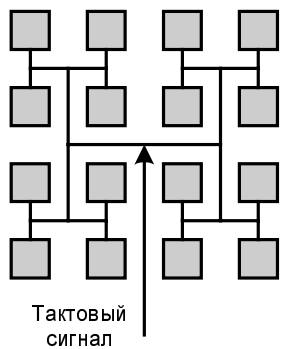
Согласно таблице на выходе $Q_1 = C1$ чередуются состояния 0100010001…, а на выходе $Q_3 = C2$ состояния 00010001…, что соответствует временным диаграммам на рис. 3.31, *б*.

Размножение тактовых импульсов

Традиционное решение — размножение тактовых импульсов разветвляющейся пирамидальной схемой (рис. 3.32, *а*), число ярусов которой зависит от числа тактируемых элементов памяти и коэффициентов разветвления задающего генератора и буферных каскадов БК. При определении числа ярусов для традиционных конструкций целесообразно ставить ярусы в соответствие каким-либо конструктивным единицам (ТЭЗам, панелям, рамам и т. п.).



а



б

Рис. 3.32. Схемы размножения тактовых импульсов: пирамидальные (*а*) и матричного типа (*б*)

В каждом БК фронты импульсов задерживаются, причем из-за разброса задержек неодинаково. Если задержки обоих фронтов в БК идентичны, то при прохождении БК длительность импульса не изменится, и сигналы разных выходов будут различаться лишь смещением во времени (фазовым сдвигом), причем максимальный сдвиг между сигналами произвольных выходов $\Delta t_{max} = m\Delta t_{БК}$, где m — число ярусов в схеме РС; $\Delta t_{БК} = (t_{БК.max} - t_{БК.min})$ — разброс задержек БК.

Временные сдвиги между тактовыми импульсами, подаваемыми на различные ЭП, существенно усложняют работу ЦУ, в связи с чем минимизации сдвигов уделяют большое внимание (применяют специальные элементы повышенного быстродействия, ограничивают обмен данными между элементами, тактируемыми отдаленными выходами схемы размножения, тщательно подбирают длины соединительных проводников или вводят специальные задержки для выравнивания путей импульсов, используют следящие системы для обеспечения синфазности тактовых сигналов и т. д.).

В современной схемотехнике получили широкое распространение регулярные структуры матричного типа, элементы которых идентичны и, чаще всего, довольно просты. Для таких структур хорошим вариантом распределения тактовых сигналов является так называемое *H-дерево* (H-tree), показанное на рис. 3.32, *б* на примере матрицы 4×4. В такой структуре близкие и взаимно сообщающиеся элементы получают тактовые сигналы в идеале идентичной фазы, т. к. они эквидистантны относительно источника. Если сами элементы просты и сдвиги сигналов внутри них малы, структура двоичного дерева H-типа особенно удобна.

§ 3.7. Однофазное и двухфазное тактирование

В этом разделе анализ систем тактирования проводится в два этапа — вначале без учета паразитных фазовых сдвигов тактовых сигналов, затем с их учетом.

Однофазное тактирование

В однофазной системе на все элементы памяти подаются тактовые сигналы одной и той же импульсной последовательности.

Однофазное тактирование с триггерами-защелками. Однофазное тактирование с триггерами-защелками представлено на рис. 3.33, *а*. Тактовые импульсы, как и всегда, должны иметь длительность $t_и$, достаточную для надежного переключения триггеров ($t_и \geq t_{wc}$, где t_{wc} — паспортный параметр триггера).

Триггеры должны принять от КЦ и сохранить сигналы, которые будут обрабатываться в текущем такте. После переключения триггеров (через время $t_{кц.min}$) на входах КЦ появляются новые данные, а через $t_{кц.min}$ после этого начинают изменяться сигналы на выходах КЦ, т. е. и на входах триггеров следующей ступени. Эти изменения в данном такте не должны восприниматься триггерами, так как они "догоняют" и разрушают уже принятую "правильную" информацию. Тактовые импульсы, разрешающие триггерам прием данных, должны закончиться на время предустановки t_h раньше, чем начнут изменяться выходы КЦ, иначе состояния триггеров могут повторно измениться, что недопустимо. Поэтому должно соблюдаться следующее условие работоспособности

$$t_{wc} \leq t_и \leq t_{тг\,min} + t_{кц.min} - t_h,$$

где $t_{тг\,min}$ — минимальное время переключения триггера.

Как видно, в данном случае необходимо строгое ограничение длительности импульсов снизу и сверху, т. к. за время существования импульса должен переключиться даже самый инерционный триггер и, в то же время, информация не должна успеть пройти через самый быстродействующий каскад обработки данных (триггер плюс КЦ). Это условие должно соблюдаться во всем диапазоне изменений условий эксплуатации устройства. Выполнение неравенства практически может обеспечиваться только инерционностью комбинационных цепей, т. е. задержкой $t_{кц.min}$. Расчету условий работоспособности данного варианта тактирования обычно препятствует отсутствие сведений о минимальных задержках элементов. Отмеченные трудности делают рассмотренную систему тактирования редкой для практики.

На практике однофазное тактирование обычно применяется в схемах с триггерами, не имеющими режимов прозрачности (с динамическим управлением, ET-типа).

Однофазное тактирование с триггерами, управляемыми фронтом. В этом случае (рис. 3.33, *б*) данные принимаются по фронту тактового импульса, и, согласно свойствам триггеров с динамическим управлением, неизменность данных требуется лишь в малом интервале времени после фронта (в течение времени выдержки t_h). За это время до входов триггеров даже по кратчайшему пути не должны дойти происходящие изменения информационных сигналов. Если это не обеспечивается, возможен сбой.

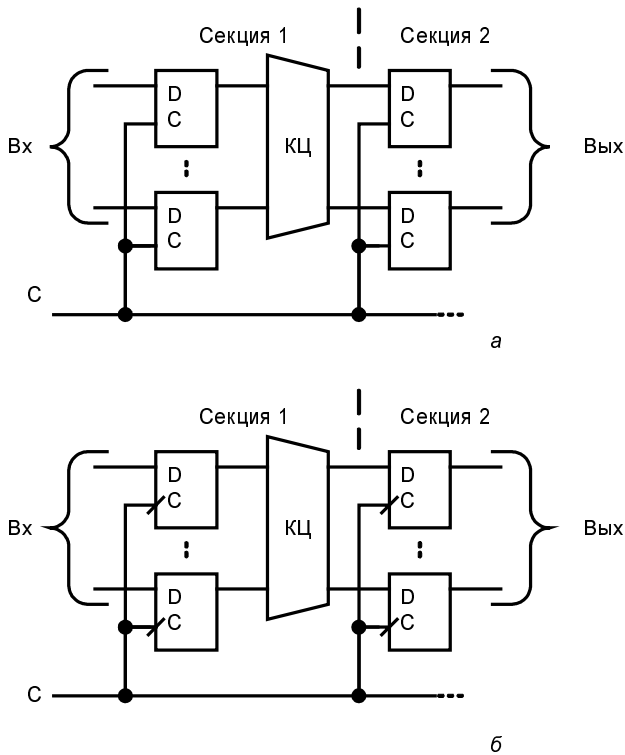


Рис. 3.33. Фрагменты тракта обработки данных с однофазным тактированием

Таким образом, и в этом варианте однофазной системы требуется соблюдение определенного условия работоспособности:

$$t_{тг\ min} + t_{кц.\ min} \geq t_h.$$

Легко заметить, что обеспечить это условие работоспособности гораздо проще, чем предыдущее, т. к. величина t_h мала. Более того, для ряда триггеров, в частности, для JK-триггеров с внутренними задержками, $t_h = 0$. А это значит, что при их применении *работоспособность систем с однофазным тактированием гарантирована.*

Учет сдвигов однофазных тактирующих сигналов. Вернемся к фрагменту обобщенного тракта обработки данных (рис. 3.34, *а*).

Временные диаграммы на рис. 3.34, *б* показывают положения фронтов синхросигналов для регистров RG1 и RG2 с учетом их сдвига. Фронт синхросигнала поступает на RG1 в момент времени t_1, а на регистр RG2 в момент t_2, запаздывая на величину

$$\Delta = t_2 - t_1.$$

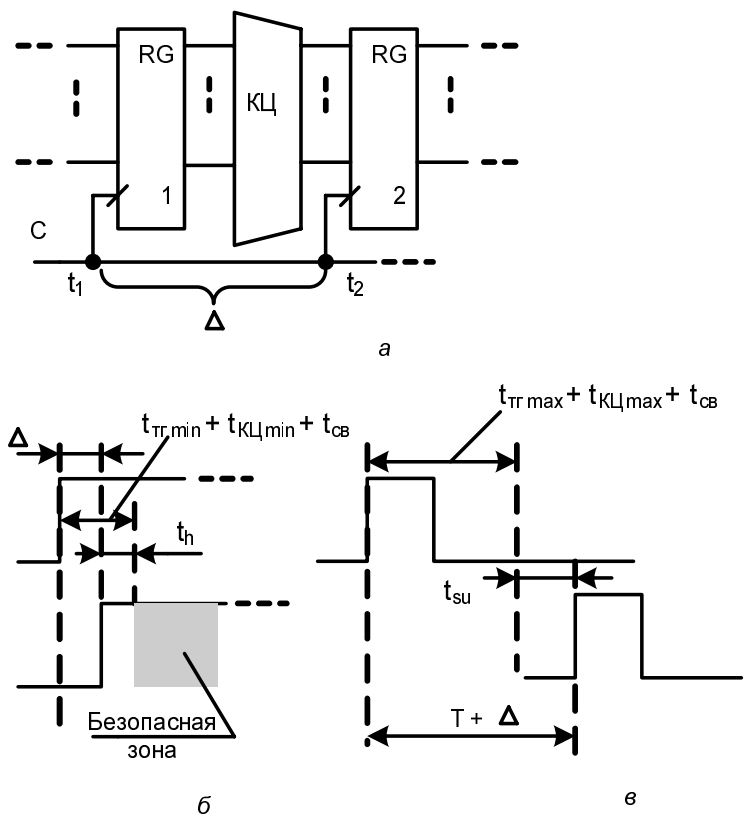


Рис. 3.34. Фрагмент тракта обработки данных (*а*)
и временные диаграммы тактирования
с учетом временных сдвигов сигналов (*б, в*)

Регистры второй секции должны принять данные в начале такта, и не должны реагировать повторно на их изменения в этом же такте. Поэтому данные на входах регистра не должны изменяться до момента $\Delta + t_h$, в который триггеры теряют чувствительность к информационным сигналам (интервалы здесь и далее отсчитываются от момента t_1). Новые данные на входах регистра RG2 появляются не ранее, чем через время $t_{тг\ min} + t_{КЦ\ min} + t_{св}$ после поступления фронта синхросигнала на регистры первой секции ($t_{св}$ — задержка в связях). Следовательно, должно соблюдаться неравенство

$$t_{тг\ min} + t_{КЦ\ min} + t_{св} \geq \Delta + t_h,$$

из которого следует условие

$$\Delta \leq t_{тг\ min} + t_{КЦ\ min} + t_{св} - t_h.$$

Важно отметить, что полученное выражение не содержит величины T, иначе говоря, при сдвиге синхросигналов, превышающем допустимые пределы, *нельзя добиться работоспособности схемы, снижая частоту тактирующих импульсов.* Надежная работа схемы обеспечивается лишь при достаточно хорошей системе распределения тактовых сигналов, поддерживающей требуемые пределы их сдвигов. Сами же пределы зависят от величин, среди которых наиболее управляемой является задержка в комбинационной цепи (могут вводиться ограничения на минимальное число элементов в путях от входов до выходов КЦ).

Временные диаграммы на рис. 3.34, *в* позволяют оценить ограничения для максимальных задержек элементов, определяющих длительность периода синхросигналов. Временной сдвиг синхросигналов соседних секций на Δ делает интервал между фронтами смежных тактов равным $T + \Delta$. К моменту приема данных в регистры, они должны стабилизироваться с учетом интервала предустановки t_{su}, для чего должно быть выполнено неравенство

$$t_{тг\ max} + t_{КЦ\ max} + t_{св} \leq T + \Delta - t_{su}$$

или

$$T \geq t_{тг\ max} + t_{КЦ\ max} + t_{св} + t_{su} - \Delta.$$

Это условие показывает, что максимальные задержки в схеме не являются фактором, препятствующим работоспособности системы, поскольку увеличение T всегда позволяет выполнить необходимое неравенство. Иными словами, максимальные задержки влияют на быстродействие системы, жертвуя которым можно всегда получить ее работоспособный вариант.

Влияние направления передачи тактовых сигналов. В рассмотренной схеме данные и тактовые сигналы распространялись в одном и том же направлении, как показано на рис. 3.35, *а*.

Однако данные и синхросигналы могут передаваться и в противоположных направлениях, как показано на рис. 3.35, *б*. Оценить свойства такого варианта можно по полученным ранее неравенствам, изменив знак Δ на противоположный. Видно, что условие, связанное с минимальными задержками, при отрицательных значениях Δ легко выполняется, а при применении триггеров с $t_H = 0$ *схема будет работоспособной независимо от сдвигов синхросигналов*, т. к. задержки элементов всегда

положительны. В то же время отрицательные значения сдвига Δ плохо влияют на быстродействие системы, поскольку для выполнения неравенства, связанного с максимальными задержками элементов, придется увеличить период T на | Δ |.

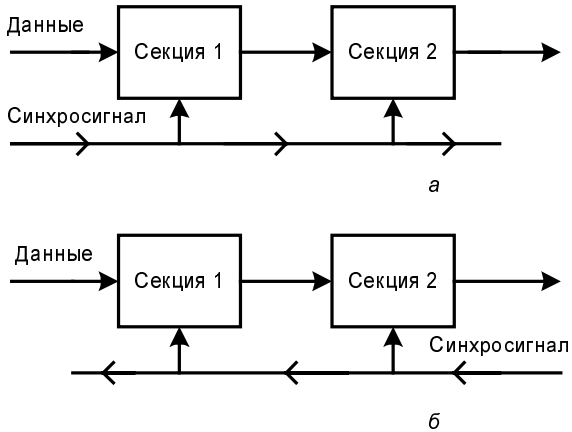


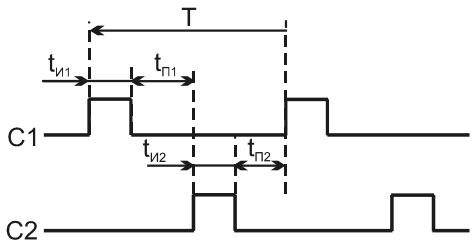
Рис. 3.35. Варианты схем однофазной синхронизации с противоположными направлениями передачи синхросигналов

Однофазное тактирование с двухступенчатыми триггерами. При однофазном тактировании с двухступенчатыми триггерами один уровень тактового сигнала открывает входные ступени триггеров, оставляя неизменными состояния выходных ступеней. При этом данные с предыдущих каскадов записываются во входные ступени следующих. Такую запись можно вести в течение необходимого времени без каких-либо опасностей временных состязаний сигналов. Переход тактового сигнала на другой уровень переносит состояния входных ступеней в выходные и изменяет тем самым переменные на входе КЦ, которые вырабатывают новые сигналы для триггеров следующего каскада. Этот процесс также можно вести без каких-либо опасений, поскольку входные ступени всех триггеров закрыты. Очередной переход тактового сигнала на новый уровень вновь запишет информацию во входные ступени триггеров и т. д. Таким образом, *временные состязания сигналов в системе с двухступенчатыми триггерами отсутствуют.* Этот вариант тактирования близок к варианту двухфазной системы с простейшими триггерами-защелками, выбор должен проводиться с учетом влияния фазовых сдвигов синхросигналов в секциях тракта обработки информации и схемной реализации триггеров.

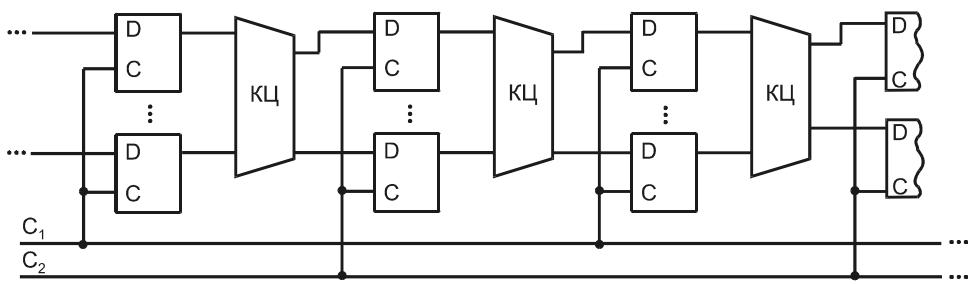
Двухфазное тактирование

Такое тактирование характеризуется применением двух последовательностей тактовых импульсов (рис. 3.36, *а*), сдвинутых во времени друг относительно друга. Интервал между импульсами обеих последовательностей отводится для работы комбинационных цепей. Соседние каскады получают разноименные серии тактовых импульсов (рис. 3.36, *б*).

При возбуждении фазы С2 данные с триггеров фазы С1 через КЦ передаются на триггеры фазы С2. При возбуждении фазы С1 триггеры этой фазы через КЦ принимают данные от триггеров фазы С2. Поочередное возбуждение фаз обеспечивает передачу данных по тракту их обработки без каких-либо временных состязаний, т. к. выдача данных производится триггерами, не изменяющими своих состояний в данной фазе, а прием данных осуществляется после завершения переходных процессов в КЦ.



а



б

Рис. 3.36. Временная диаграмма тактовых сигналов (*а*) и схема тактирования элементов памяти для двухфазной системы тактирования (*б*)

Достоинство двухфазной системы — возможность применения простых одноступенчатых триггеров, управляемых уровнем (защелок). Двухфазность тактирующей системы несколько усложняет схему устройства, но находит широкое применение, т. к. обеспечивает надежное тактирование даже при значительных фазовых сдвигах синхросигналов (естественно, ограниченных некоторыми пределами). Недостатком системы можно считать распределение логических схем по двух тактирующим последовательностям, ограничивающее взаимосвязи между схемами (сигналы, тактированные одной фазой, можно подавать только на схемы, тактируемые другой).

Расчет параметров тактовых импульсов для двухфазной системы основан на той же стратегии, что и расчет для однофазной, т. е. на недопущении повторного срабатывания элементов памяти в данном такте и обеспечении неизменности информационных сигналов на входах триггеров в интервалах t_{su} и t_h даже при наихудшем сочетании допусков на положения фронтов тактирующих сигналов и задержек в КЦ.

Параметры сигналов двухфазного тактирования. На рис. 3.37, *а* показан фрагмент схемы с двухфазным тактированием для передачи данных от секции 1 к секции 2, а на рис. 3.37, *б* приведены временные диаграммы тактирующих импульсов двух фаз с учетом их возможного паразитного сдвига.

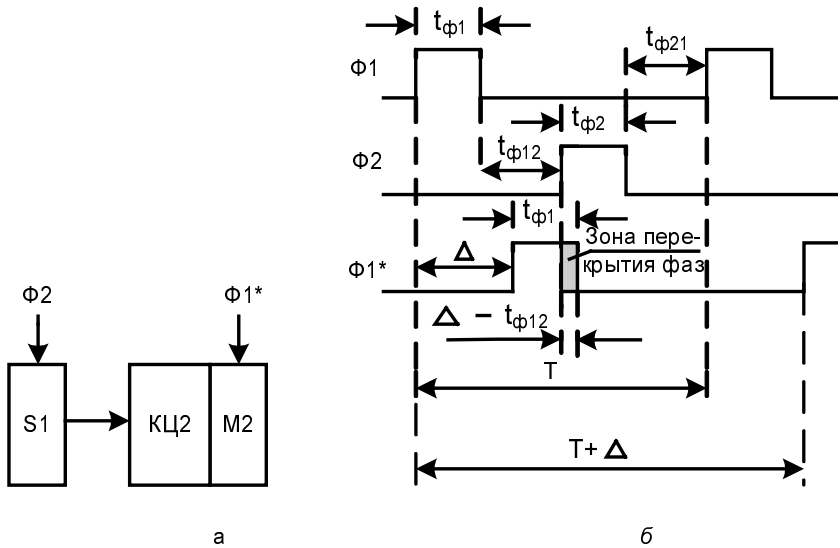


Рис. 3.37. Схема передачи данных между секциями обобщенного тракта обработки информации с двухфазным тактированием (*а*) и временные диаграммы ее работы (*б*)

Нормальное положение импульсов первой и второй фаз показано на диаграммах Ф1 и Ф2, сдвинутые импульсы первой фазы — на диаграмме Ф1*. Величина сдвига обозначена через Δ. Нормальное положение импульсов соответствует отсутствию перекрытия фаз, причем интервалы неперекрытия обозначены через $t_{ф12}$ и $t_{ф21}$. Из-за сдвига фазы на интервал Δ может возникнуть зона перекрытия фаз, ширина которой составляет величину $\Delta - t_{ф12}$.

Рассмотрим передачу данных из первой секции во вторую. Положительный фронт импульса фазы Ф2 разрешает прием данных в защелку секции S1. На выходе этой защелки после времени ее задержки появляются новые данные, являющиеся входными для комбинационной цепи КЦ2. Выходные сигналы КЦ2 принимаются защелкой М2 по положительному фронту импульсов фазы Ф1*.

Из-за запаздывания фазы Ф1* на время Δ может возникнуть перекрытие фаз, т. е. обработка данных в КЦ2 будет идти при открытой защелке М2. Если при этом распространение сигналов через логику КЦ2 может оказаться слишком быстрым, то сигналы переходных процессов могут достичь защелки М2 до спада сигнала Ф1* и нарушить правильную работу схемы. Для предотвращения указанной ситуации должно соблюдаться неравенство:

$$t_{гг\,min} + t_{КЦ\,min} + t_{св} \geq \Delta - t_{ф12} + t_{h}.$$

Пока $\Delta + t_h < t_{ф12}$, правая часть неравенства отрицательна, и оно безусловно соблюдается, поскольку задержки элементов положительны. Следовательно, *интервал $t_{ф12}$ выступает как буфер, компенсирующий влияние сдвигов фаз синхросигналов на работу схемы.* Увеличивая этот интервал, можно добиться работоспособности схемы при наличии сдвигов фаз, синхронизирующих прием данных в разных секциях обобщенного тракта обработки информации.

Для оценки ситуации, связанной с максимальными задержками элементов схемы, следует учесть, что интервал от положительного фронта фазы Ф2 до положительного фронта фазы Ф1* должен обеспечивать подготовку новых данных для защелки М2, т. е. необходимо выполнение неравенства:

$$T \geq t_{тг\,max} + t_{КЦ\,max} + t_{св} + t_{su} - \Delta,$$

совпадающего по форме с выражением, полученным ранее для системы однофазной синхронизации.

Многофазное тактирование

Многофазная синхронизация характеризуется использованием более чем двух серий синхроимпульсов и применяется, в частности, для увеличения быстродействия систем путем организации работы их частей с разной скоростью. Это осуществляется разбиением периода основной частоты на части, формированием сигналов с уменьшенными периодами и использованием в отдельных блоках системы более высокочастотных тактовых сигналов. Многофазное тактирование свойственно, как правило, системам высокой сложности и быстродействия.

Кроме того, многофазность тактирования появляется и как следствие применения в системе нескольких независимых частот, управляющих работой различных блоков. Например, с разными тактирующими частотами могут работать процессор, шины PCI и USB, графические блоки и т. д. Подробное рассмотрение многофазного тактирования выходит за рамки целей этой книги.

§ 3.8. Блоки PLL, DLL и DCM

Проблема расфазирования тактовых импульсов для быстродействующих ЦУ настолько остра, что современные БИС/СБИС снабжаются специальными довольно сложными блоками, улучшающими синфазность тактовых сигналов в различных областях схем и при необходимости выполняющими и преобразования их частоты (умножение, деление).

Это схемы коррекции фазы тактовых сигналов Phase Locked Loops (PLLs) и Delay Locked Loops (DLLs). Между PLL и DLL есть разница в технической реализации, но на них возлагаются идентичные задачи — *коррекция фазы синхросигналов и, при необходимости, умножение или деление их частоты.* Благодаря введению PLL или DLL, удается снижать фазовый сдвиг тактовых сигналов системы (Clock Skew) до малых значений.

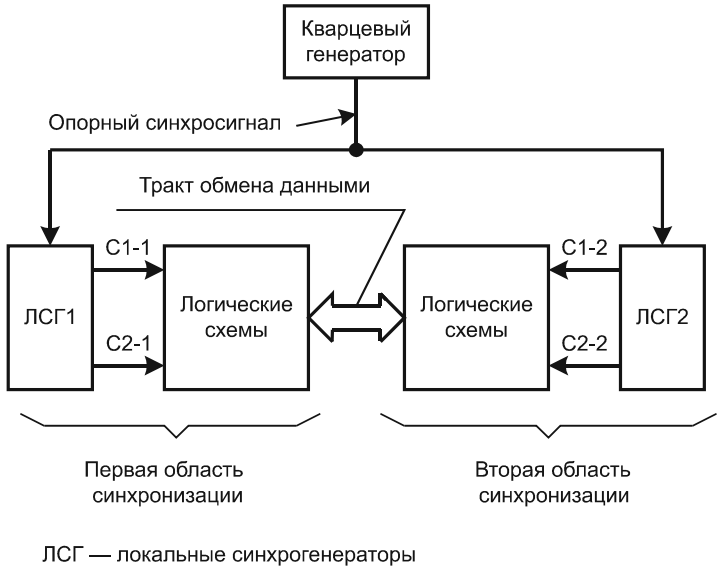


Рис. 3.38. Схема обмена данными между двумя частями системы

Для более полного понимания роли PLL (или DLL) рассмотрим схему, использующую общий опорный синхросигнал и локальные схемы формирования синхросигналов для отдельных частей системы (рис. 3.38).

В этой схеме опорный тактовый сигнал от стабильного кварцевого генератора подается в различные области синхронизации (в частности, на различные кристаллы) по специальной ненагруженной линии, т. е. *без ощутимых фазовых сдвигов*. Для синхронизации работы каждой области вырабатываются пары синхросигналов C1 и C2 с требуемой скважностью. Синхросигналы C1-1 и C1-2, а также C2-1 и C2-2 в силу ряда причин имеют заметные фазовые сдвиги относительно опорного сигнала и относительно друг друга. Эти фазовые сдвиги вносятся неидентичностью схем локальных синхрогенераторов ЛСГ1 и ЛСГ2 (в первую очередь их выходными буферами), разными задержками во входных контактах и межсоединениях. *Блоки PLL* (Phase Locked Loop) или *блоки DLL* (Delay Locked Loop), в значительной степени устраняют фазовые сдвиги между контролируемыми синхросигналами, "привязывая" их фронты в разных частях системы к фронтам опорного сигнала, выравнивая тем самым их фазы (устраняя Clock Skew).

Блоки PLL

Следует отметить, что PLL (Phase Locked Loop), по крайней мере, в реализуемых сейчас вариантах, — аналоговые устройства. Действительно, компенсируемый сдвиг фазы синхросигналов может иметь любое значение (является непрерывной величиной), поэтому и система его точной отработки должна работать в аналоговом режиме. С этой точки зрения PLL трудны для реализации. Являясь замкнутыми нелинейными системами, PLL могут оказаться неустойчивыми, что требует тща-

тельной отработки их для обеспечения надежной устойчивой работы во всем диапазоне изменения температуры и других воздействий.

В структуре PLL (рис. 3.39, *а*) опорный и корректируемый сигналы подаются на фазовый детектор, определяющий фазовые соотношения между ними. Простейшим фазовым детектором может служить логический элемент "неравнозначность" (сложение по модулю 2). Если сигналы синфазны, то выходное напряжение такого элемента равно нулю. При фазовом сдвиге между синхросигналами появляются интервалы, на которых выходное напряжение элемента становится высоким, причем длительности таких интервалов пропорциональны величине фазового сдвига. Более сложные схемы фазовых детекторов определяют и знак рассогласования фаз синхропоследовательностей. Именно такой фазовый детектор предполагается в схеме рис. 3.39, *а*.

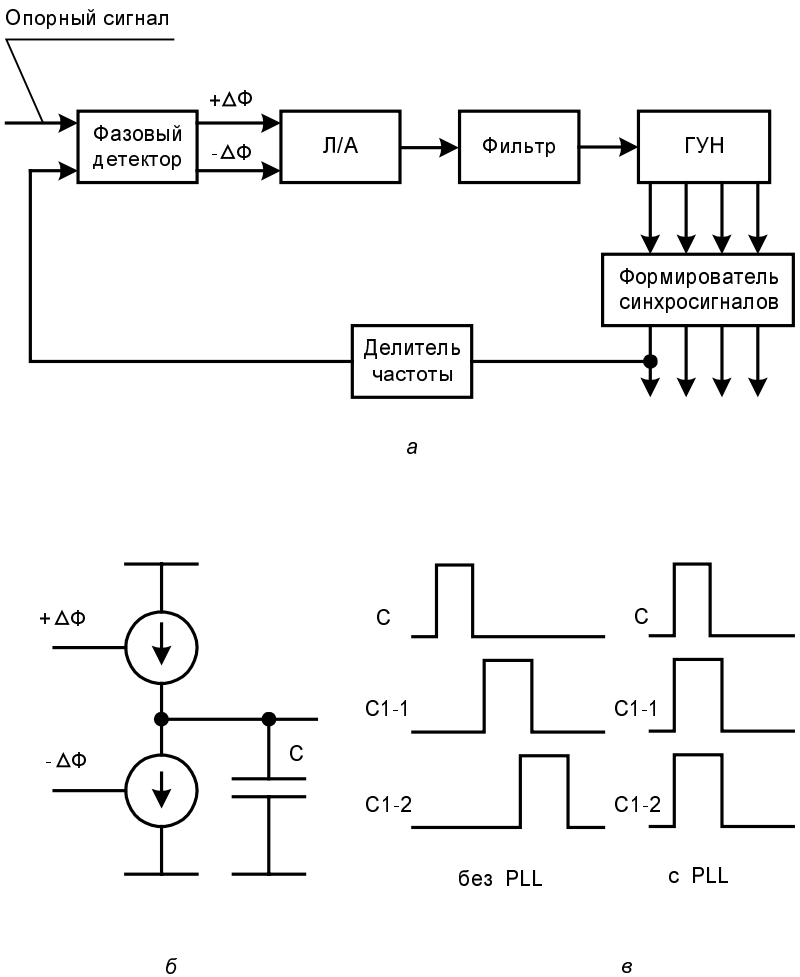


Рис. 3.39. Структура PLL и пояснения к ее функционированию

Для последующих блоков PLL нужны аналоговые сигналы, поэтому выходы фазового детектора подаются на блок Л/А преобразования "логический—аналоговый" (Charge Pump), в состав которого (рис. 3.39, *б*) входят генераторы токов, ключи и конденсатор. Логический сигнал от фазового детектора подключает к конденсатору тот или иной генератор тока в зависимости от знака рассогласования фаз синхросигналов (сигналы Up и Down), заставляя тем самым напряжение на емкости расти или уменьшаться по линейному закону.

Аналоговое напряжение поступает далее на *генератор, управляемый напряжением*, (ГУН, VCO — Voltage Controlled Oscillator). ГУН вырабатывает импульсную последовательность переменной фазы, подбираемой так, чтобы минимизировать сдвиг входных сигналов фазового детектора. Перед блоком ГУН включается фильтр низких частот, устраняющий высокочастотные составляющие напряжения, поскольку при отсутствии фильтра появляется дрожание фронтов выходного напряжения (Jitter). ГУН вырабатывает скорректированные по фазе синхросигналы, один из которых используется как входной для фазового детектора. С помощью PLL удается свести рассогласование фаз опорного и корректируемого сигналов к весьма малым значениям (рис. 3.39, *в*).

Введение в цепь обратной связи делителя частоты приводит к умножению частоты, формируемой блоком распределения синхросигналов, на коэффициент, равный коэффициенту деления.

Блоки DLL

Схема DLL (Delay Locked Loop) (рис. 3.40) имеет два входа для синхросигналов — эталонного и обратной связи. На выходе формируются семь сигналов, два из которых (CLK 0 или CLK 2X) могут участвовать в формировании сигналов обратной связи. Выход CLK 0 формирует сигналы, аналогичные входным по частоте и фазе, а выход CLK 2X генерирует сигналы удвоенной частоты. На выходе CLK DV формируются сигналы пониженной частоты. Сигнал обратной связи с помощью схемы управления и цепочки из *n* элементов задержки приводится к эталонному.

Функция (Clock Skew) устранения сдвига фаз между эталонными сигналами и сигналами тактирования в данной области синхронизации реализуется следующим образом. Сигнал CLK 0 проходит через схемы распределения тактирующих импульсов данной области синхронизации и возвращается к DLL как сигнал обратной связи. Схема управления оценивает фазовую погрешность между входами детектора и активизирует такое число элементов задержки, которое дает устранение этой погрешности. Сфазировав оба сигнала, DLL устанавливает сигнал Locked.

Для получения частоты, умноженной на N, в цепь обратной связи вводится делитель частоты на N. Реализуются и другие операции по формированию синхросигналов с различными параметрами.

Сопоставляя структуры PLL и DLL, можно отметить, что аналоговые PLL способны работать с более высокой точностью, поскольку в их схемах фазовые сдвиги непрерывны и могут иметь любые значения (в пределах рабочего диапазона), но для PLL,

как замкнутых систем, характерны проблемы возможной неустойчивости, накопления фазовых погрешностей и чувствительности к помехам. Схемы DLL с дискретными элементами задержки отрабатывают фазовые сдвиги более грубо, но не являются замкнутыми аналоговыми системами, не накапливают фазовые погрешности и более просты в разработке.

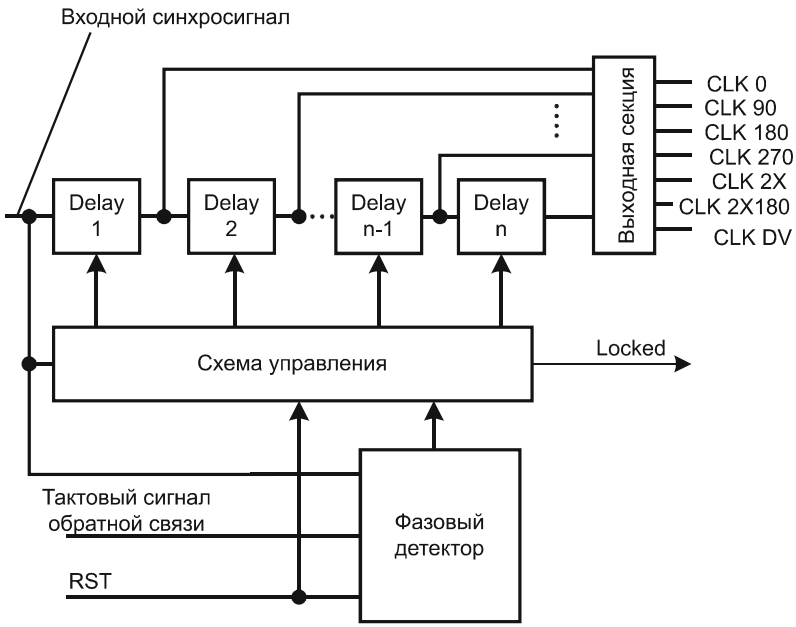


Рис. 3.40. Структура DLL

Блоки DCM

Усложнение функций, возлагаемых на устройства повышения качества синхросигналов, привело к появлению *блоков DCM* (Digital Clock Manager), которые имеют расширенные возможности. В DCM реализованы функции PLL или DLL, кроме того, могут быть синтезированы новые синхросигналы заданных частот и фаз. Блок DCM состоит из четырех функциональных элементов: сдвигателя фаз — блока PS (Phase Shifter), блока DLL, блока цифрового синтезатора частоты — блока DFS (Digital Frequency Synthesizer) — и логических схем состояния (Status Logic).

Блок DLL вырабатывает выходные сигналы, назначение которых было рассмотрено при его описании. Блок DFS синтезирует частоты, подвергая входную частоту одновременно умножению и делению на определяемые пользователем целые числа, так что выходная частота задается отношением двух этих чисел, т. е. коэффициенты умножения или деления могут быть дробными. Блок DFS может работать совместно с DLL или без него, причем в последнем случае не обеспечиваются определенные фазовые соотношения между выходными сигналами блока и входным сигналом CLKIN. Блок PS управляет соотношением фаз выходных

сигналов и входного сигнала CLKIN, сдвигая фазу всех сигнальных выходов на фиксированную часть периода входного сигнала, величина которой вводится при конфигурировании схемы. Логические схемы состояния отражают текущее состояние DCM сигналами Locked и Status [2:0]. Сигнал Locked показывает, синфазны ли выходные сигналы со входным, а статусные оповещают о функциях, выполняемых блоками DCM.

§ 3.9. Тактирование сигналами, выработанными в приемниках информации

Выработка тактовых сигналов без передачи эталонов

Рассмотрим процесс восприятия информации с коррекцией тактовых сигналов самими данными.

Место блоков DR (Data Recovery) в системе иллюстрируется на рис. 3.41. В схеме имеется локальный синхрогенератор, номинальная частота которого соответствует частоте принимаемого потока данных (номинальное равенство частот фактически означает лишь незначительность их различия). Блок DLL формирует локальные синхросигналы CLK0 и CLK90, причем версия CLK0 представляет основной сигнал генератора, а версия CLK90 задержана относительно CLK0 на 90° (на ¼ периода). Синхросигналы CLK0 и CLK90 используются в блоках DR, которые согласуют тактирование сигналов, поступающих по последовательным каналам, с требованиями принимающей системы.

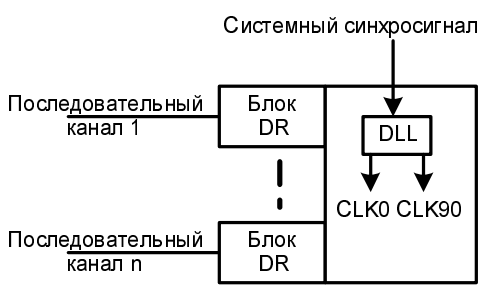


Рис. 3.41. Включение блоков DR в систему

Оба синхросигнала показаны на рис. 3.42, *а*. Там же приведены 4 возможных варианта моментов появления входных данных. Структура блока DR представлена на рис. 3.42, *б*. Рассмотрим основную идею работы схемы. Входные данные поступают на 4 канала, причем входные триггеры каналов тактируются различно — в верхнем канале положительным фронтом сигнала CLK0 и далее следующим образом: положительным фронтом CLK90, отрицательным фронтом CLK0, отрицательным фронтом CLK90. Такая схема создает 4 точки восприятия данных (A, B, C, D), отстоящие

друг от друга на ¼ периода. Далее в каждом канале размещены синхронизаторы ждущего типа (производятся повторные фиксации входных данных, чтобы снизить вероятность попадания канала в метастабильное состояние).

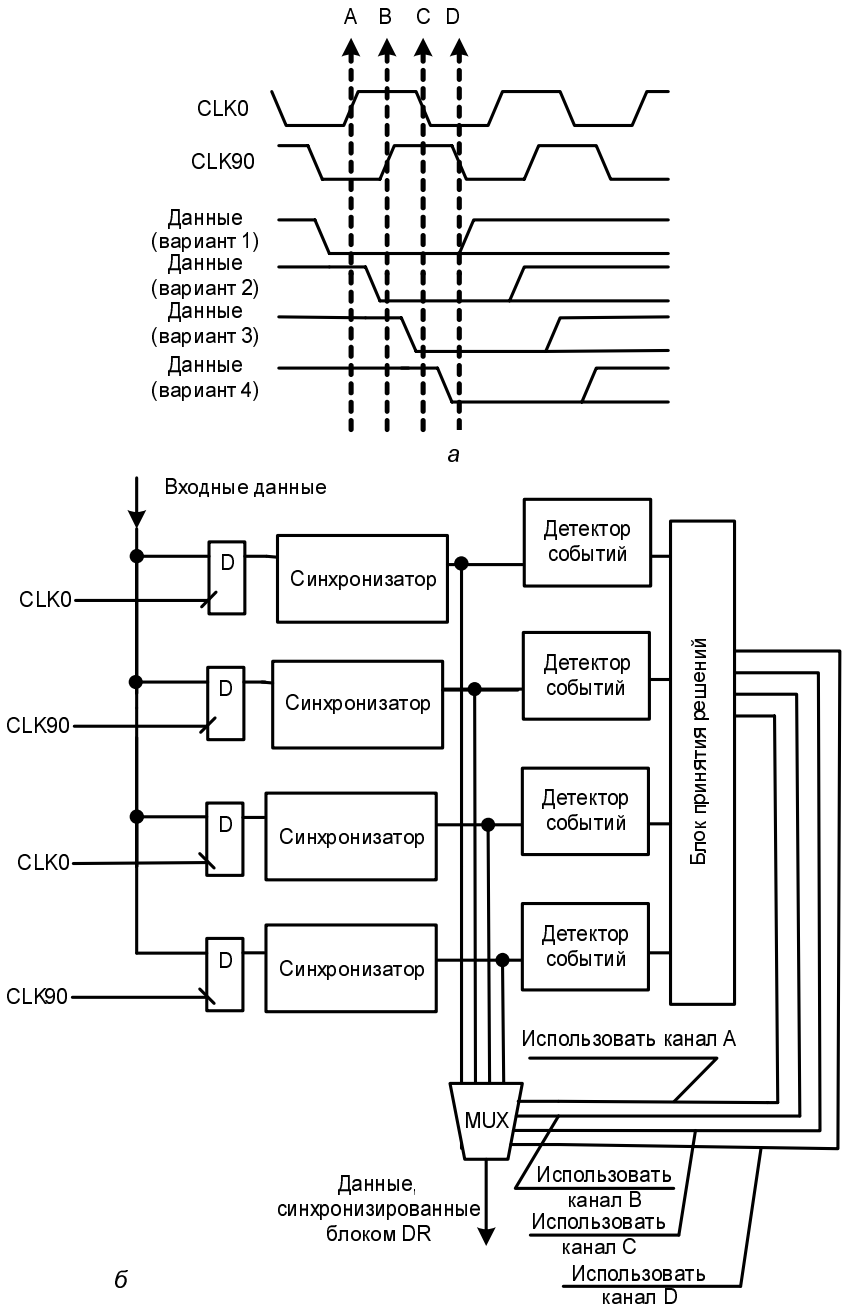


Рис. 3.42. Структура блока DR

Выходы каналов подключены к детекторам событий, выявляющим как положительные, так и отрицательные фронты сигнала (P- и N-сигналы). Детекторы событий предоставляют сигналы для решения о наилучшем варианте тактирования. На основе анализа ситуаций принимается решение. Для примера отметим две ситуации. Так, единичные значения всех P-сигналов или N-сигналов означают, что сигнал появился до первой точки восприятия. Наилучшим вариантом в этом случае будет прием по каналу C, у которого точка фиксации сигнала находится близко к середине периода (она отстает от A на ½ периода). Если же сигналы каналов B, C, D будут идентичны, а сигналы канала A противоположны им, то первым "увидел" данные канал B, и для использования следует выбрать канал D. Таким образом, для всех возможных положений входных данных выбирается наилучший вариант тактирования. Выбор наилучшего канала осуществляется мультиплексором под управлением выходов блока принятия решения.

В полной схеме блока DR имеется дополнительная логика учета специфических ситуаций. Кроме того, следует учитывать, что для правильной работы схемы за время, соответствующее накоплению сдвига фаз между сигналами генераторов передающей и принимающей систем на ¼ периода, должно произойти, по крайней мере, одно событие. При необходимости это обеспечивается специальным кодированием входных данных (например, применением кодов 8b/10b).

Выработка тактовых сигналов с передачей эталона

Коррекция фазы тактовых сигналов с помощью потока поступающих данных может производиться и следующим образом. В месте приема данных формируются *несколько тактирующих последовательностей* с разными фазами. Затем передается *слово-эталон*. Это известное слово принимается несколько раз с поочередным использованием сформированных тактирующих последовательностей. "Плохие" последовательности дадут прием с ошибками, "хорошие" — без ошибок. На основании этого опыта можно принять в качестве рабочего наилучший из имеющихся вариантов тактирующих сигналов. Таким образом, тактирующие сигналы будут приспособлены к данным.

О самосинхронизирующихся схемах

Как радикальную меру борьбы с расфазированием синхросигналов можно рассматривать и *переход к так называемым самосинхронизирующимся схемам (Self-Timed Design)*, которые фактически являются разновидностью *асинхронных систем*, в которых квитирование при обработке данных переведено с уровня устройств на уровень элементов памяти. Пока что этот метод мало применяется, т. к. цена введения самосинхронизации (за счет дополнительных аппаратных затрат) слишком велика.

§ 3.10. Ввод внешних сигналов в синхронные устройства. Синхронизаторы

Ввод асинхронных сигналов

К проблемам синхронизации относится также *арбитраж*, при котором применительно ко входным сигналам требуется ответить на вопрос "Кто первый?".

При вводе в систему асинхронные данные должны быть "привязаны" к системным тактовым сигналам, для чего можно пропустить вводимые данные через D-триггеры, тактируемые от системных ТИ. Запрещается, как обычно, изменение входных сигналов триггера в окне $t_{su}...t_h$ в окрестности перепада тактового сигнала. Однако при вводе асинхронных сигналов нет возможности обеспечить соблюдение этого запрета, поэтому попаданий триггеров в аномальные метастабильные состояния не избежать.

Временные диаграммы (рис. 3.43) показывают ситуации с поступлением асинхронных входных данных до, после или вблизи тактового воздействия. Сигналы, поступившие до активного фронта синхросигнала, триггером воспринимаются, а сигналы, поступившие после фронта — игнорируются. При поступлении сигналов в запретном интервале возможно попадание триггера в метастабильное состояние, отмеченное затемненной зоной. Вероятность того, что прием асинхронного сигнала приведет к метастабильному состоянию, т. е. того, что изменение принимаемого сигнала окажется в запретной зоне, равна ее относительной длительности t_a/T, где t_a — длительность запретной (апертурной) зоны и T — период синхросигнала. Длительности метастабильных состояний имеют случайный характер и лежат в широких пределах. Вероятность попадания в метастабильное состояние снижается способами, иллюстрируемыми далее при рассмотрении схем синхронизаторов.

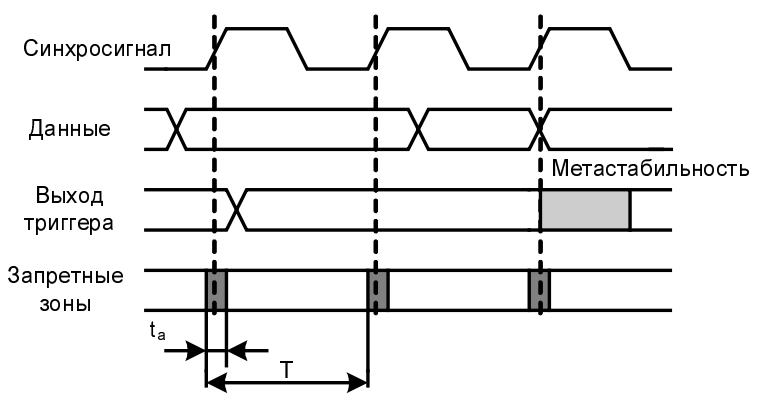


Рис. 3.43. Временные диаграммы ввода асинхронных данных

Ширина апертурного окна зависит от схемы триггера, технологического уровня его изготовления, условий его работы и нагрузки на выходе. Для триггеров, принимающих асинхронные сигналы, выделяют больше площади на кристалле и больше потребляемой мощности с целью реализации быстродействующей схемы с узким апертурным окном и без колебательных аномалий. Но при всех стараниях все же невозможно полностью избежать состояний метастабильности. Обмен информацией между областями синхронизации всегда сопровождается сложностями и требует применения специальных схем.

Ждущие синхронизаторы (Brute-Force Synchronizers, Waiting Synchronizers) состоят из двух или большего числа последовательно включенных D-триггеров с непосредственной связью между ними. Свойства таких синхронизаторов тесно связаны с параметрами триггеров и они рассмотрены ранее (см. рис. 3.21).

В схемах *с буфером FIFO* синхронизация достигается тем, что вводимые данные записываются в буфер под управлением тактовой частоты передающей системы, а потом считываются из буфера под управлением тактовых импульсов принимающей системы. Основой таких синхронизаторов служит буфер FIFO — одна из типовых структур запоминающих устройств, рассмотренная в *главе 5*.

Синхронные, асинхронные и "полусинхронные" сигналы

С точки зрения передачи между областями синхронизации сигналы можно разделить на следующие разновидности:

- *полностью синхронные*;
- *мезохронные*;
- *плезиохронные*;
- *периодические*;
- *асинхронные*.

Передачи сигналов из одной области синхронизации (тактирования) в другую имеют большую универсальность в том смысле, что полностью синхронным можно сделать лишь "островок в асинхронном мире", и взаимодействие разных областей синхронизации практически неизбежно. Сложность решения задачи пересечения сигналами границ областей синхронизации зависит от типа сигналов.

Крайними разновидностями сигналов с точки зрения определенности их положения во времени являются *полностью синхронные* (принадлежащие одной области синхронизации) и *асинхронные* (моменты их изменения произвольны). Промежуточный характер имеют три типа "полусинхронных" сигналов (мезохронные, плезиохронные и периодические).

Мезохронные сигналы имеют одну и ту же частоту синхронизации, но разную фазу, причем разность их фаз хотя и произвольна, но постоянна. Именно мезохронные сигналы образуются из-за задержек синхросигналов одного генератора в цепях их распределения по кристаллу, печатной плате и т. д. Для успешного пересечения

мезохронными сигналами границы между областями синхронизации достаточно задержать ТИ или сигналы на некоторую постоянную величину.

Плезиохронные сигналы имеют близкие, но не идентичные частоты тактирования, поэтому разность их фаз медленно дрейфует. Такая ситуация типична для взаимодействия двух устройств или систем с *номинально одинаковыми* частотами тактирования, реально задаваемыми с теми или иными допусками, так что фактически частоты несколько различаются. В этом случае для пересечения границы между областями тактовые импульсы или информационные сигналы нужно задерживать на переменную величину.

Периодические сигналы имеют произвольную частоту тактирования. Разность фаз периодических тактирующих сигналов тоже периодична. Периодичность позволяет предсказывать ситуации с попаданием изменений сигналов в запрещенные зоны и на этой основе вырабатывать способы пересечения сигналами границы областей синхронизации, однако эти способы более сложны, чем для предыдущих случаев.

Ввод в систему *асинхронных* сигналов не поддается какому-либо предсказанию моментов их появления. Это затрудняет построение синхронизаторов. Мезохронность, плезиохронность или периодичность вводимых сигналов позволяет предсказывать будущие временные положения моментов изменения данных и тактирующих импульсов и дает тем самым дополнительные возможности для построения синхронизаторов, хотя и не избавляет полностью от возможных метастабильностей.

Схемы ввода "полусинхронных" сигналов усложняются соответственно усложнению самих сигналов по линии "мезохронные—плезиохронные—периодические". Основные приемы построения синхронизаторов видны уже из рассмотрения вариантов с мезохронными сигналами. Далее рассматриваются именно эти варианты и отмечаются способы их модификации для работы с другими сигналами. Более полные сведения о синхронизаторах для сигналов различного типа можно найти, например, в работе [57].

Синхронизаторы мезохронных сигналов

Синхронизаторы для мезохронных сигналов реализуются в следующих вариантах:

- с элементами задержек;
- с двумя регистрами;
- с круговым буфером.

Синхронизаторы с элементами задержек

Вариант синхронизатора с элементами задержек показан на рис. 3.44, *а*. Напомним, что мезохронные сигналы различаются лишь фазами, но не частотами тактирования.

Чтобы успешно ввести в систему мезохронный сигнал X, требуется вывести моменты его изменения из запретных зон в окрестности фронтов ТИ принимающей системы. Для этого входной сигнал передается через элемент регулируемой задержки с диапазоном изменения не менее периода тактирования. Требуемая величина задержки ав-

томатически подбирается в процессе синхронизации. Так как частоты тактирования поступающего сигнала и принимающей его системы идентичны, подбор задержки достаточно провести однократно. Последовательно с элементом задержки включен триггер, выход которого является выходом синхронизатора.

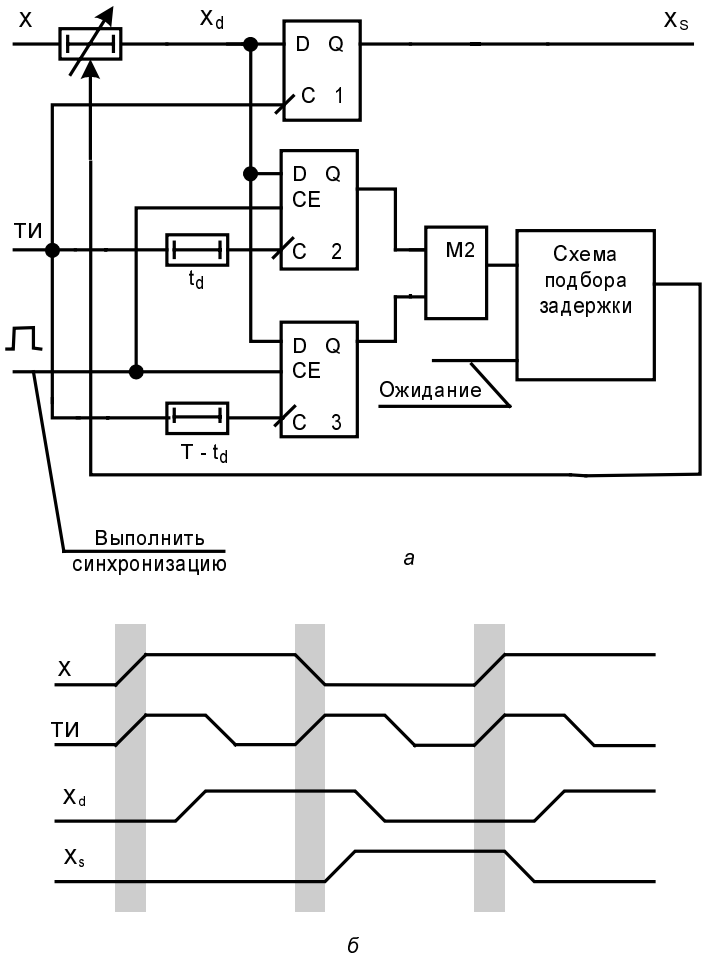


Рис. 3.44. Схема синхронизатора мезохронных сигналов с элементами задержек (*а*) и временные диаграммы ее работы (*б*)

Чтобы обнаружить изменения сигнала X в запретной зоне вблизи тактирующего фронта, этот сигнал принимается триггерами 2 и 3 на ее краях, поскольку задержки ТИ для этих триггеров составляют t_d и $T - t_d$. Ширина запретной зоны равна $2t_d$ (времена предустановки и выдержки приняты одинаковыми и равными t_d, иначе задержки на входах триггеров были бы равными t_h и $T - t_{SU}$). Если изменений сигнала в запретной области нет, то триггеры 2 и 3 примут один и тот же сигнал и выход схемы сложения по модулю 2 будет нулевым. Если же сигнал изменился в запретной области, то состояния триггеров 2 и 3 окажутся различными, и на выходе схемы М2 поя-

вится единица, которая заставит схему подбора задержки изменить ее значение и вывести тем самым принимаемый сигнал из запретной зоны. Можно сразу увеличить задержку на величину, близкую к T/2, максимально удалив сигнал от запретной зоны, или же увеличивать задержку малыми долями с целью уменьшения запаздывания передачи сигнала, вносимого синхронизатором. При приеме сигнала триггерами 2 и 3 не исключено их попадание в метастабильные состояния. Вероятность этого в данном случае можно снижать до весьма малого уровня, поскольку однократность процесса синхронизации позволяет анализировать состояния триггеров 2 и 3 после интервала ожидания, в котором будут затухать метастабильности.

Работа синхронизатора иллюстрируется рис. 3.44, *б*. Входной сигнал X изменяется в зоне, запретной для принимающей области (показано затененными полосами), а задержанный сигнал X_d и выходной сигнал X_S расположены во времени правильно.

Синхронизаторы с двумя регистрами

Вариант синхронизатора с двумя регистрами показан на рис. 3.45.

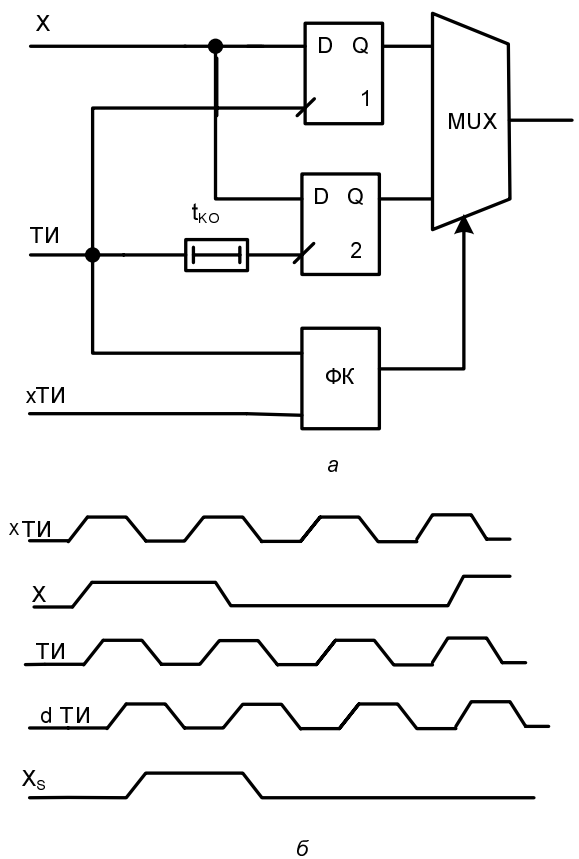


Рис. 3.45. Схема двухрегистрового синхронизатора мезохронных сигналов (*а*) и временные диаграммы ее работы (*б*)

Применение предыдущего варианта синхронизатора для многоразрядных шин требует введения элементов задержки в каждую линию шины. Избежать этого можно, если подбирать необходимую для синхронизации задержку не для сигнальных линий, а для линии тактирования, как и сделано в данном варианте. Синхронизатор содержит два триггера, верхний из которых принимает входные сигналы X по положительному фронту ТИ принимающей системы, а нижний — по положительным фронтам задержанных ТИ, причем задержка соответствует ширине запретной области $t_{КО}$. Если речь идет о вводе многоразрядных данных, то вместо верхнего триггера включается регистр требуемой разрядности. Если запретная область принимающего триггера меньше полупериода ТИ, запретные области обоих триггеров не перекрываются и входной сигнал надежно воспринимается одним из триггеров. Выбор такого триггера производится мультиплексором MUX, передающим на выход синхронизированный сигнал X_S.

"Надежный" триггер выбирается однократно при инициализации системы, когда фазовый компаратор ФК измеряет сдвиг между тактирующими импульсами передающей и принимающей схем (хТИ и ТИ). Если события в линии X не происходят в запретной области тактовых импульсов ТИ, то выбирается верхний триггер, иначе выбирается нижний. Временные диаграммы рис. 3.45, *б* соответствуют выбору нижнего триггера.

Синхронизатор с круговым буфером

Вариант синхронизатора *с круговым буфером* (Ring Buffer) основан на применении небольшого блока FIFO для разделения систем тактирования передающей и принимающей схем (рис. 3.46, *а*).

В рассматриваемом варианте применен двухразрядный круговой буфер (триггеры 1 и 2), на таких же принципах можно построить и синхронизатор с большей разрядностью буфера.

От передающей системы поступает информационный сигнал X, тактируемый импульсами хТИ. Тактирующие сигналы принимающей схемы ТИ поступают на счетный триггер и на триггер типа D, с которого снимается синхронизированный сигнал X_S. Счетный триггер принимающей стороны вырабатывает адресующий сигнал RP (Read Pointer) мультиплексора MUX, так что на выход X_M попеременно подаются величины X0 и X1. Фаза сигнала RP устанавливается специальной логикой так, что через мультиплексор всегда передается выход того триггера, который будет стабильным во время следующего такта.

Сигнал X_M синхронизирован с тактовой частотой принимающей системы, он может восприниматься комбинационной цепью КЦ и записываться в D-триггер по тактам ТИ.

Временные диаграммы, иллюстрирующие работу синхронизатора, приведены на рис. 3.46, *б*. Пять первых строк показывают, как на передающей стороне входной поток X расщепляется на перемежающиеся потоки X0 и X1. Следующие три строки показывают, как сигнал RP выбирает выход триггера передатчика, кото-

рый изменяется только в тех интервалах времени, в которых он не передается через мультиплексор.

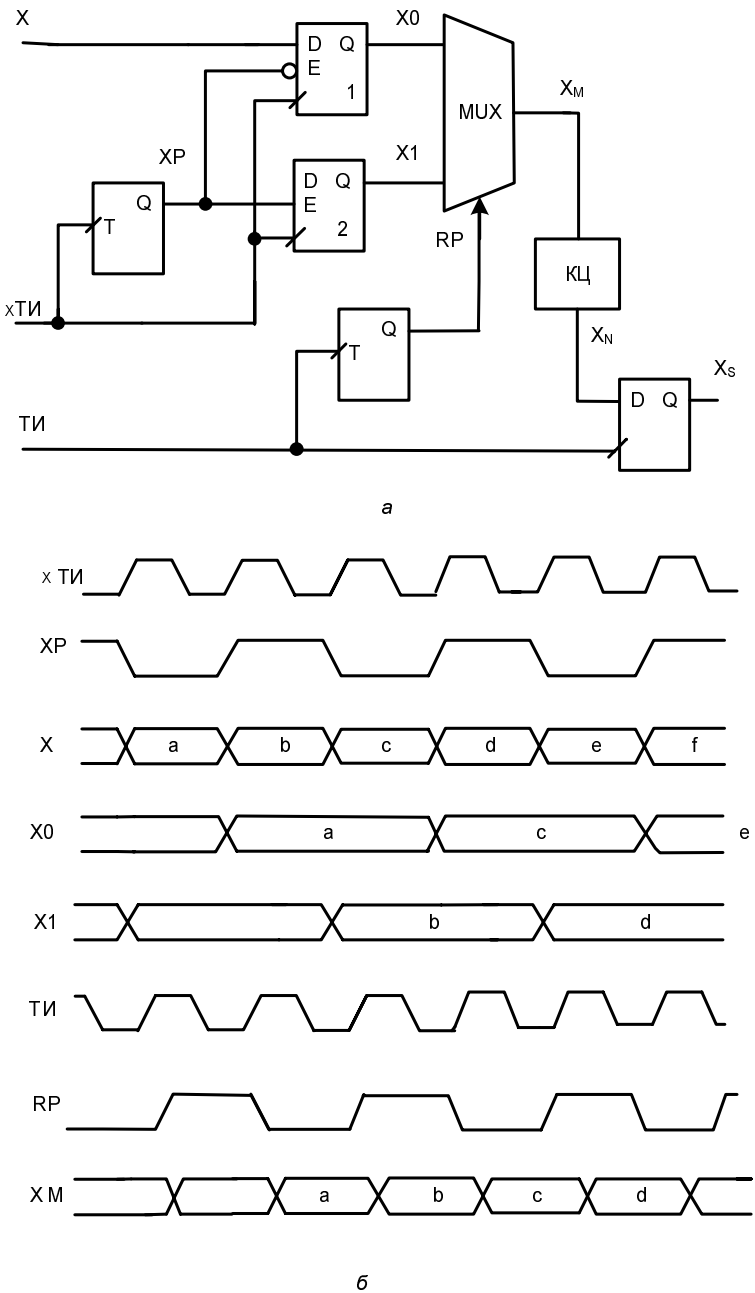


Рис. 3.46. Схема синхронизатора мезохронных сигналов с круговым буфером (*а*) и временные диаграммы ее работы (*б*)

Для правильной работы синхронизатора сигнал RP должен отставать от сигнала XP по меньшей мере на период тактовой частоты. При этом сигнал RP должен иметь высокий уровень во время положительного фронта XP, так чтобы при изменении X0 воспринимался выход X1 и наоборот. Это обеспечивается при инициализации схемы.

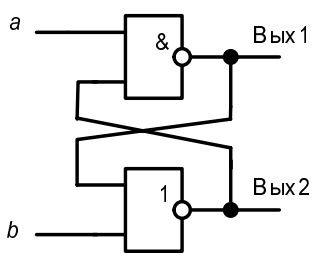
При совпадении (близости) фаз тактирующих импульсов передающей и принимающей систем в схеме с двухразрядным буфером возникают нестабильности сигнала X_M в начале или конце периода. Радикально устранить этот недостаток можно увеличением разрядности кругового буфера до трех.

Синхронизаторы плезиохронных сигналов

При передаче мезохронных сигналов через границу между областями синхронизации требуется однократно произвести временной сдвиг того или иного сигнала на постоянную величину. Если задержки в схемах неизменны, повторных сдвигов не потребуется. Для плезиохронных сигналов временной сдвиг медленно дрейфует, поэтому однократная его компенсация недостаточна, и по мере отклонения от исходных условий требуется коррекция первоначально установленных временных соотношений между сигналами. Эту задачу могут автоматически выполнять те же схемы, что и предназначенные для синхронизации мезохронных сигналов. При этом прохождение относительными фазовыми сдвигами сигналов значений, соответствующих целому числу периодов тактовой частоты, создает особенности, из-за которых происходит потеря или удвоение информационного символа. Поэтому в схемах для обработки плезиохронных сигналов появляются дополнительные элементы, предотвращающие отмеченные явления. Подробнее эти схемы, как и синхронизаторы для периодических сигналов, рассматриваются, в частности, в работе [57].

Контрольные вопросы и упражнения

1. Перечислите способы описания функционирования триггеров.
2. Какие свойства схемы делают ее триггером? Является ли триггером показанная на рисунке схема?



3. Дайте определения понятию "защелка" (latch).

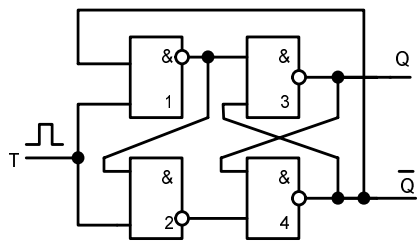
4. Какой триггер называется счетным? Что понимается под асинхронным и синхронным счетными триггерами?

5. В цифровых устройствах в зависимости от конкретных условий целесообразно применение триггеров RS, JK, D и T. В то же время в составе стандартных микросхем часто встречаются только триггеры D и JK. Чем объясняется это обстоятельство?

6. Что такое "круговые гонки"? В каких структурах триггеров они возникают?

7. Почему для схем триггеров на элементах И-НЕ установочным сигналом служит нулевой, а для схем на элементах ИЛИ-НЕ — единичный? Как асинхронные установочные сигналы воздействуют на триггер при одновременном наличии других сигналов?

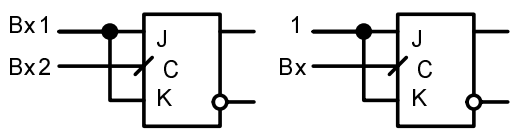
8. На рисунке приведена схема триггера T, построенная на основе триггера D со статическим управлением. В такой схеме может возникать режим генерации. Укажите, при каких условиях это будет происходить.



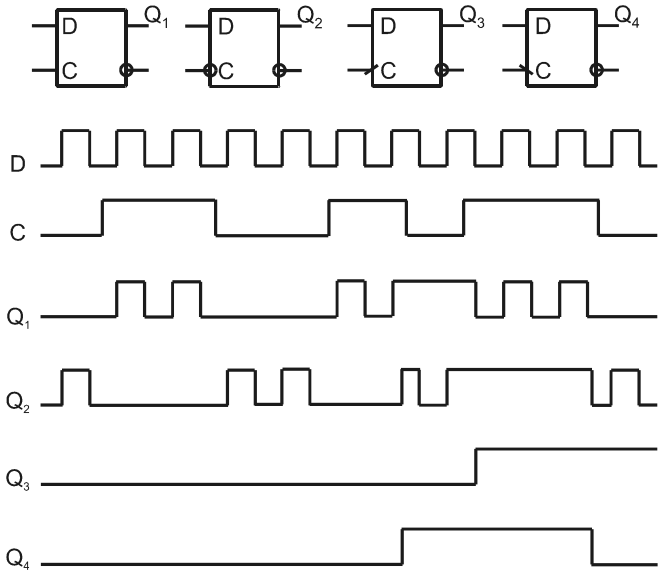
9. Определите понятия "время предустановки" и "время удержания".

10. Какое состояние триггеров называют метастабильностью? Какова роль явлений метастабильности в схемотехнике цифровых устройств?

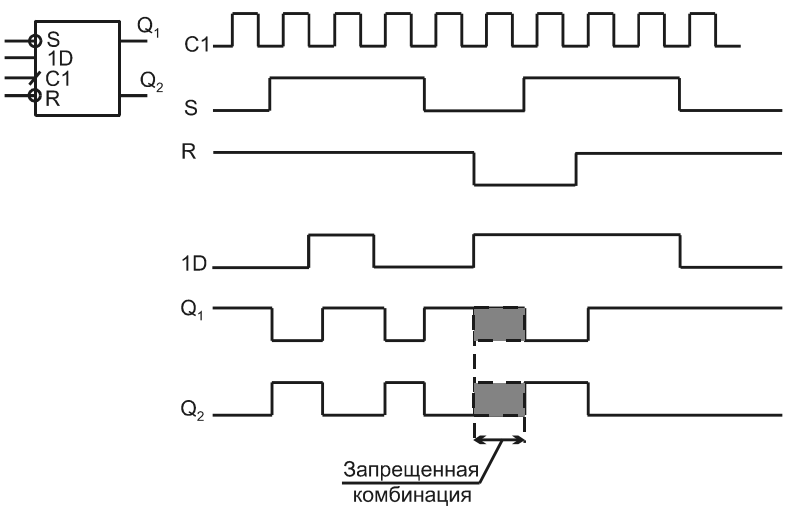
11. На рисунке показаны две схемы включения JK-триггера. Определите назначение входов и функциональные типы триггеров для этих включений.



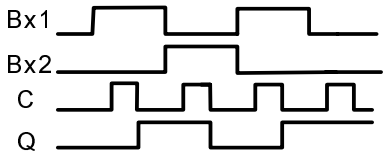
12. На рисунке показаны временные диаграммы, иллюстрирующие реакцию триггеров D разных типов на показанные входные сигналы D и C. Проверьте по этим диаграммам правильность своих представлений о функционировании таких триггеров.



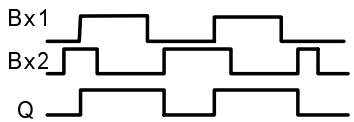
13. На рисунке показаны временные диаграммы, иллюстрирующие реакцию RSD-триггеров на показанные входные сигналы. Проверьте по этим диаграммам правильность своих представлений о функционировании таких триггеров.



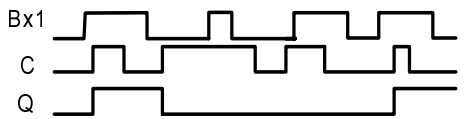
14. Определите тип триггера по приведенной временной диаграмме и нарисуйте его условное графическое обозначение (УГО). Укажите функциональное назначение пронумерованных входов (через C обозначен тактирующий вход, через Q — выход триггера).



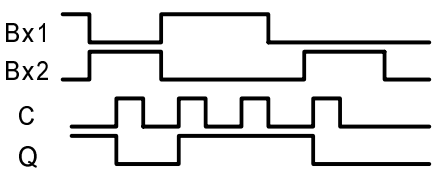
15. Определите тип триггера по приведенной временной диаграмме и нарисуйте его условное графическое обозначение (УГО). Укажите функциональное назначение пронумерованных входов (через Q обозначен выход триггера).



16. Определите тип триггера по приведенной временной диаграмме и нарисуйте его условное графическое обозначение (УГО). Укажите функциональное назначение пронумерованного входа (через C обозначен тактирующий вход, через Q — выход триггера).



17. Определите тип триггера по приведенной временной диаграмме и нарисуйте его условное графическое обозначение (УГО). Укажите функциональное назначение пронумерованных входов (через C обозначен тактирующий вход, через Q — выход триггера).



18. На каких особенностях явления метастабильности триггеров основана эффективность ждущего синхронизатора?

19. От каких временных параметров блоков КЦ и элементов памяти тракта синхронной обработки информации зависят длительность импульса и длительность паузы тактирующих сигналов?

20. Какие варианты триггеров (с точки зрения способа приема информации) целесообразно применять в системах с однофазным тактированием? С двухфазным?

21. Как направление передачи тактирующих сигналов (попутное с передачей данных или встречное) влияет на критичность схемы к фазовым сдвигам тактирующих импульсов и на быстродействие устройства?
22. В чем состоит сущность процесса, называемого Clock-Data Recovery?
23. Объясните смысл понятия "область синхронизации"
24. Какие сигналы называются мезохронными? Плезиохронными?
25. Какие задачи решают блоки PLL и DLL?

Литература к главе: [3], [10], [26], [33], [34], [36], [39], [43], [52], [57], [60], [66].

Глава 4

Функциональные узлы последовательностного типа (автоматы с памятью)

§ 4.1. Введение в проблематику проектирования автоматов с памятью

Устройства, содержащие *элементы памяти* (ЭП), имеют некоторое внутреннее состояние Q, определяемое совокупностью состояний всех ЭП. В зависимости от внутреннего состояния (далее называемого просто состоянием), *автомат с памятью* (АП) или, короче, *автомат* различно реагирует на один и тот же вектор входных сигналов X. Набор входных сигналов и внутреннее состояние АП определяют новое состояние

$$Q_н = f(Q, X)$$

и вектор выходных переменных

$$Y = \varphi(Q, X),$$

где Q и $Q_н$ — состояния АП до и после подачи входных сигналов (индекс "н" от "новое").

Переходы АП из одного состояния в другое начинаются с некоторого исходного состояния Q_0, которое также является частью задания автомата. В любой момент времени состояние и выходы автомата зависят от его начального состояния и всех векторов X, поступавших на автомат ранее. Таким образом, входная последовательность векторов X преобразуется автоматом в выходную последовательность векторов Y. Это объясняет применяемое для АП название "*последовательностные схемы*" (не "последовательные").

Автоматы строятся из логических элементов, и структурно отличаются от комбинационных цепей наличием *обратных связей*, вследствие чего в них появляются свойства памяти (вспомним схемы триггеров).

Автоматы в каноническом представлении разделяют на две части: *память* и *комбинационные цепи* (КЦ). КЦ вырабатывают выходные сигналы и сигналы перевода элементов памяти в новое состояние.

Принципиальным является деление АП на *асинхронные* и *синхронные*. В асинхронных автоматах (рис. 4.1, *а*) роль элементов памяти играют элементы задержки, через которые сигналы состояния передаются на входы КЦ, чтобы совместно с новым набором входных переменных определить следующую пару значений Y и Q на выходе. Элементы АП переключаются здесь под непосредственным воздействием изменений информационных сигналов.

В синхронном АП (рис. 4.1, *б*) имеются специальные тактирующие сигналы C (Clocks), разрешающие элементам памяти прием данных только в определенные моменты времени. Элементами памяти служат синхронные триггеры. Процесс обработки информации упорядочивается во времени, и в течение одного такта возможно распространение процесса переключения только в строго определенных пределах тракта обработки информации.

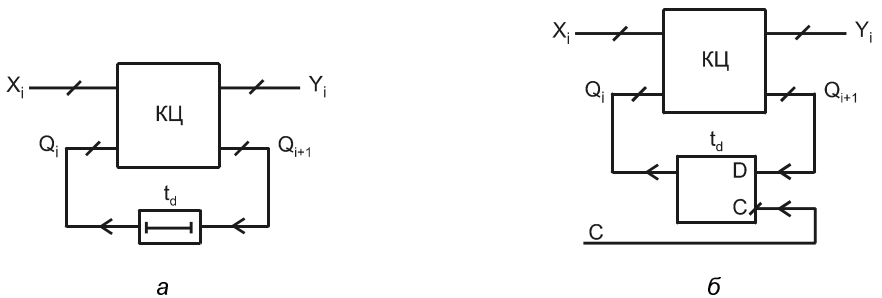


Рис. 4.1. Асинхронный (*а*) и синхронный (*б*) автоматы с памятью

Практическое применение асинхронных автоматов существенно затруднено влиянием на их работу задержек сигналов в цепях АП, создающих статические и динамические риски, гонки элементов памяти (неодновременность их срабатывания даже при одновременной подаче на них входных сигналов) и др. При переходе асинхронного автомата из одного устойчивого состояния в другое он обычно проходит через промежуточные нестабильные состояния. Нельзя сказать, что методы борьбы с нежелательными последствиями рисков и гонок в асинхронных АП отсутствуют, но все же обеспечение предсказуемого поведения АП — сложная проблема. В более или менее сложных АП асинхронные схемы встречаются очень редко, хотя в простейших случаях и применяются (примером могут служить асинхронные RS-триггеры).

В синхронных автоматах каждое состояние устойчиво и переходные временные состояния не возникают. Концепция борьбы с последствиями рисков и гонок в синхронных автоматах проста — прием информации в элементы памяти разрешается только после завершения в схеме переходных процессов. Это обеспечивается параметрами тактирующих импульсов, задающих интервалы времени для выполнения тех или иных процессов. В сравнении с асинхронными, синхронные АП значительно проще в проектировании.

На сегодняшний день и достаточно длительную перспективу основным путем построения АП следует считать применение синхронных автоматов.

В теории автоматов проводится их классификация по ряду признаков. Не вдаваясь в подробности, отметим, что у *автоматов Мура*, выходы являются функциями только состояния автомата. Для этих автоматов

$$Q_н = f(Q, X),$$
$$Y = \varphi(Q).$$

Зависимость выходов и от состояния автомата и от вектора входных переменных свойственна *автоматам Мили*, для которых

$$Q_н = f(Q, X),$$
$$Y = \varphi(Q, X).$$

Автономные автоматы не имеют информационных входов, и под действием тактовых сигналов переходят из одного состояния в другое по алгоритму, определяемому структурой автомата.

О проектировании автоматов

Проектирование АП предполагает выполнение следующих этапов:

- ❒ исходное задание функционирования;
- ❒ формализованное задание функционирования;
- ❒ минимизация состояний;
- ❒ кодирование состояний;
- ❒ составление таблицы переходов;
- ❒ определение функций возбуждения элементов памяти (триггеров);
- ❒ минимизация функций возбуждения триггеров;
- ❒ переход к базису выбранных для реализации комбинационных схем;
- ❒ составление логической схемы;
- ❒ проверка функционирования автомата.

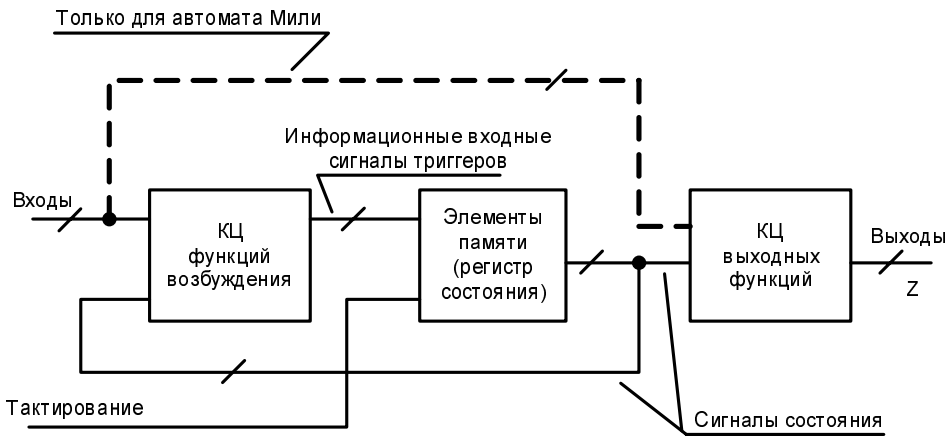
Исходное задание автомата может иметь различную форму, в том числе и словесную. От нее переходят к формализованному заданию — таблицам, формулам, диаграммам состояния (графам переходов) и т. п. Далее выполняются минимизация и кодирование состояний автомата, в результате получается таблица переходов, на основании которой можно найти функции возбуждения триггеров.

Минимизация и кодирование состояний в общем случае задача, решение которой может потребовать значительных усилий, однако при проектировании узлов ЭВМ и цифровой автоматики ее решение чаще всего подсказывается самой формулировкой задания на проектирование. *Широко применяется кодирование состояний автомата двоичными кодами*, экономное по затратам триггеров. Для некоторых СБИС, снабженных большим числом триггеров, экономия их числа несущественна. Для таких случаев нередко предпочтительны *коды "1 из N", приводящие к более быстродействующим схемам*, хотя и требующим значительного числа триггеров. Для исключения чреватых помехами ситуаций с одновремен-

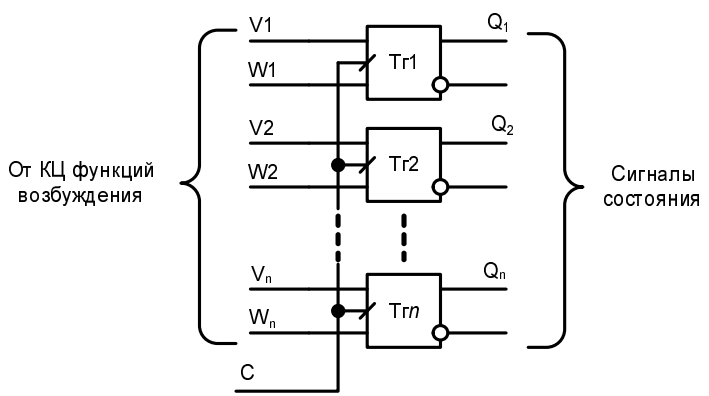
ным переключением многих элементов памяти (ЭП) применяют кодирование *кодом Грея* или *Либау-Крейга (Джонсона)*.

Функции возбуждения триггеров, обеспечивающие переходы АП из одного состояния в другое, реализуются его комбинационной частью. В процессе проектирования они минимизируются и переводятся в базис выбранных средств реализации автомата, в зависимости от которых требования к формам представления функций возбуждения могут существенно различаться (*см. § 2.1*). Элементами памяти синхронных автоматов служат синхронные триггеры, причем любой автомат можно построить на любом типе триггера (D, JK, RS, T и др.), хотя и с различной эффективностью.

Выполнение указанных действий дает логическую схему АП. Заканчивается процесс проверкой работы АП с помощью моделирования или макетирования.



а



б

Рис. 4.2. Структурные схемы автоматов Мура и Мили (*а*) и схема регистра состояния (*б*)

Рассмотрим более подробно *методику проектирования автоматов, содержащих триггеры* (рис. 4.2, *а*).

При кодировании состояний двоичным кодом или кодом Грея число триггеров в схеме автомата равно $n =]\log_2 N[$, где N — число состояний автомата и]…[— знак округления до ближайшего справа целого числа. При кодировании кодом "1 из N" число триггеров равно числу состояний автомата: n = N, т. к. каждому состоянию соответствует один триггер в единичном состоянии при нулевом состоянии остальных.

Будем считать, что закон функционирования автомата определен, и кодирование его состояний произведено. Значит, известна последовательность состояний триггеров, принимаемых ими в каждом такте под управлением входных сигналов x_1, x_2,..., x_k и текущего состояния Q_1, Q_2,..., Q_n. Предмет синтеза — получение для входов всех триггеров функций возбуждения V_i и W_i, обеспечивающих необходимые переходы автомата (рис. 4.2, *б*).

Выходы автомата Мура зависят только от его состояния, поэтому сигналы Z формируются комбинационной схемой, на которую подаются только выходы триггеров Q_1, Q_2,..., Q_n. Все триггеры тактируются общим синхросигналом C. После завершения выработки функций возбуждения поступает очередной тактовый сигнал C, переводящий триггеры в новое состояние. Вид функций возбуждения зависит от логического типа триггеров. Поэтому одним из средств синтеза служат "словари" для триггеров (*см. § 3.1*).

При поиске функций возбуждения триггеров вначале составляется таблица (табл. 4.1), содержащая следующие данные.

Таблица 4.1

Входы в момент времени t				**Состояния триггеров**								**Необходимые сигналы на всех входах каждого триггера**				
				Q (старое)				**Q_H (новое)**								
x_1	x_1	...	x_k	Q_1	Q_2	...	Q_n	Q_1	Q_2	…	Q_n	V_1	W_1	…	V_n	W_n

Столбцы V_1, W_1,..., V_n, W_n определяют функции возбуждения триггеров.

Многовариантность реализаций автомата связана с выбором типа триггеров и комбинационной части. Наиболее распространены триггеры JK и D. Триггер JK обладает более развитыми логическими возможностями, поэтому для него функции возбуждения в среднем более просты, но число их вдвое больше, чем для триггера D. Что же даст более простое решение, заранее неизвестно. Комбинационная часть автомата может быть построена на логических элементах, мультиплексорах, ИС программируемой памяти, программируемых логических матрицах и т. д.

Автомат можно построить, приспособив к необходимому функционированию типовую ИС среднего уровня интеграции (счетчик, сдвигающий регистр), добавив к ним специально спроектированную логическую часть.

Реализации автомата с использованием ИС памяти, программируемых логических матриц и тому подобных устройств поясняются далее (после рассмотрения соответствующих средств). Далее приведены примеры построения автоматов на рассмотренных в предыдущих разделах типовых элементах при кодировании состояний двоичными кодами и кодами "1 из N".

Примеры проектирования

Пусть необходимо спроектировать автомат с двумя режимами работы, управляемый входным сигналом M. При M = 0 автомат работает как двоичный счетчик с модулем счета 4, при M = 1 как счетчик в коде Грея.

Примечание

Код Грея используется в счетчиках, системах контроля ЦУ, преобразователях механических перемещений в цифровой код и т. д. При переходе от предыдущей кодовой комбинации к следующей в коде Грея изменяется только один разряд. Для получения следующей кодовой комбинации находят самый младший разряд, изменение которого дает новую комбинацию, еще не встречавшуюся среди предыдущих. Последовательность кодовых комбинаций для кода Грея можно получить и по соотношению $g_i = b_i \oplus b_{i+1}$, где g_i — значение разряда кода Грея; b_i — значение разряда двоичного кода, преобразуемого в код Грея. Разряд левее старшего для двоичного кода считается нулевым. Первые восемь комбинаций кода Грея представлены в табл. 4.2.

Таблица 4.2

Десятичная цифра	Код Грея			Десятичная цифра	Код Грея		
0	0	0	0	4	1	1	0
1	0	0	1	5	1	1	1
2	0	1	1	6	1	0	1
3	0	1	0	7	1	0	0

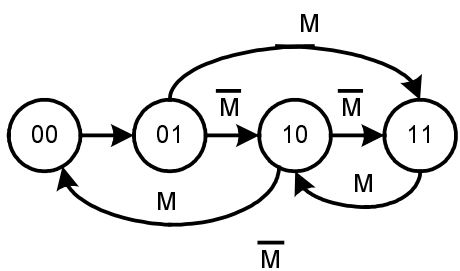


Рис. 4.3. Диаграмма состояний автомата для примера проектирования

Кодирование состояний автомата, являющегося автоматом Мура, определяется здесь самой постановкой задачи. Диаграмма состояний автомата показана на рис. 4.3. Изменение управляющего сигнала M сразу же ведет к изменению режима, т. е. следующее состояние будет принадлежать уже другому коду. Тактирующие сигналы на диаграмме не показаны, т. к. они являются условиями всех без исключения переходов из одного состояния в другое.

Вариант 1

Автомат, построенный на триггерах D и элементах И-НЕ

Таблица переходов автомата (табл. 4.3) соответствует диаграмме его состояний.

Таблица 4.3

M	Исходное состояние		Новое состояние		M	Исходное состояние		Новое состояние	
	Q_1	Q_0	Q_{1H}	Q_{0H}		Q_1	Q_0	Q_{1H}	Q_{0H}
0	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	1	1	1
0	1	0	1	1	1	1	0	0	0
0	1	1	0	0	1	1	1	1	0

Сигналы возбуждения для триггеров D совпадают с их новыми состояниями, т. е. $D_1 = Q_{1H}$ и $D_0 = Q_{0H}$. Карты Карно для функций D_1 и D_0 показаны на рис. 4.4.

Из карт Карно получаем выражения для функций возбуждения триггеров и приводим их к базису И-НЕ:

$$D_1 = \overline{M}Q_1\overline{Q}_0 \vee \overline{Q}_1Q_0 \vee MQ_0 = \overline{\overline{\overline{M}Q_1\overline{Q}_0} \cdot \overline{\overline{Q}_1Q_0} \cdot \overline{MQ_0}},$$

$$D_0 = \overline{M\overline{Q}_0} \vee M\overline{Q}_1 = \overline{\overline{M\overline{Q}_0} \cdot \overline{M\overline{Q}_1}}.$$

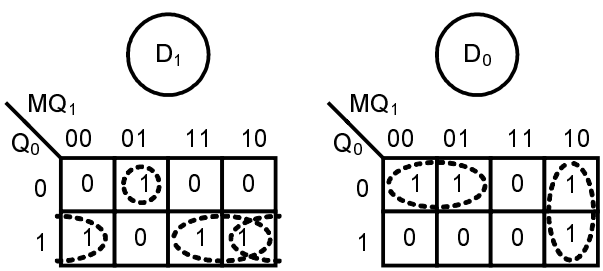


Рис. 4.4. Карты Карно для функций возбуждения триггеров D проектируемого автомата

Схема автомата приведена на рис. 4.5.

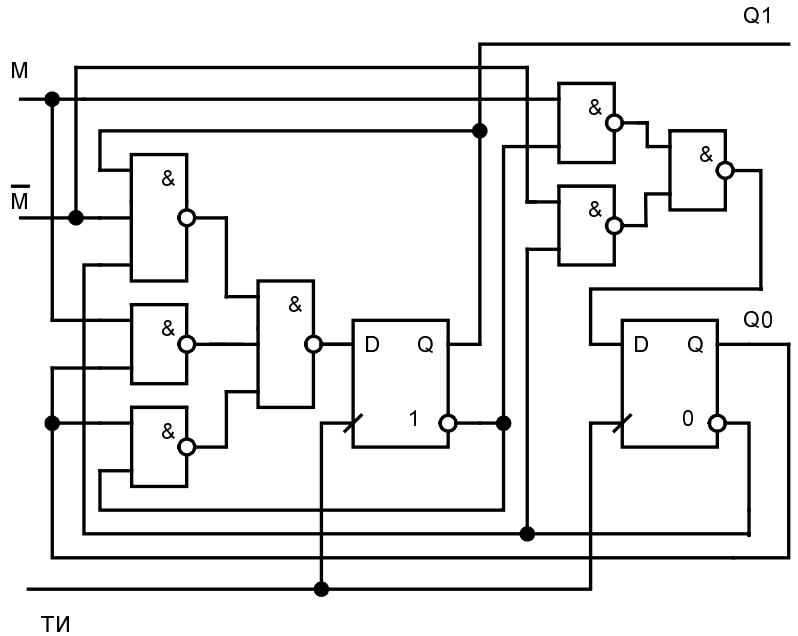


Рис. 4.5. Схема автомата, реализованная с применением D-триггеров

Вариант 2

Автомат, построенный на JK-триггерах и элементах И-НЕ

Синтез функций возбуждения для автоматов с триггерами JK имеет интересную особенность. Они могут быть получены не только указанным ранее путем, но и непосредственно из выражений для функций Q_H. Из данных о функционировании автомата можно получить функцию переходов каждого триггера

$$Q_{нi} = F\ (x_1, x_2,..., x_k, Q_1, Q_2,..., Q_n).$$

Эту функцию можно разложить следующим образом

$$Q_{Hi} = f_i\overline{Q}_i \vee g_i Q_i, \qquad (4.1)$$

где функции f и g уже не содержат соответственно переменных Q_i и $\overline{Q}_i$.

Характеристическое уравнение триггера типа JK имеет вид:

$$Q_{Hi} = J_i\overline{Q}_i \vee \overline{K}_i Q_i. \qquad (4.2)$$

Сопоставляя выражения (4.1) и (4.2), получим $f_i = J_i$, $\overline{g}_i = K_i$.

Следовательно, положив в функции переходов триггера $Q_i = 0$, сразу получаем функцию возбуждения для входа J:

$$Q_{Hi}\Big|_{Q_i = 0} = f_i = J_i,$$

а приняв условие Q = 1, можно получить функцию возбуждения для входа K_i:

$$Q_{Hi}\Big|_{Q_i = 1} = g_i = \overline{K}_i, \text{ т. е. } \overline{K}_i = g_i.$$

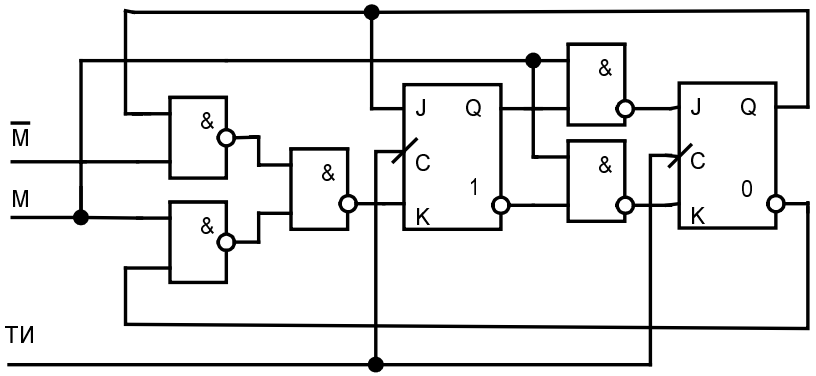


Рис. 4.6. Схема автомата, реализованная с применением JK-триггеров

Для нашего примера получаем:

$$J_1 = Q_0, K_1 = M\overline{Q}_0 \vee \overline{M}Q_0, = \overline{\overline{M\overline{Q}_0} \cdot \overline{\overline{M}Q_0}},$$

$$J_0 = \overline{M} \vee \overline{Q}_1 = \overline{MQ_1}, \; K_0 = \overline{M} \vee Q_1 = \overline{M\overline{Q}_1}.$$

Схема автомата приведена на рис. 4.6.

Вариант 3

Автомат, реализованный на D-триггерах и мультиплексорах

Структура с мультиплексорами на входах D-триггеров отличается концептуальной простотой и наглядностью, для ее проектирования не требуется разработка логических преобразователей, обеспечивающих необходимые переходы автомата.

Задача решается, в сущности, табличным методом. Переменные состояния, снимаемые с триггеров, и входные сигналы образуют слово, служащее для мультиплексора адресным входом. По этому адресу в каждом мультиплексоре выбирается переменная (0 или 1), необходимая для перевода D-триггера в новое состояние. Ясно, что при этом данные для информационных входов мультиплексоров берутся прямо из таблицы переходов ($D_i = Q_{iн}$). Достоинство структуры — легкость перестройки автомата на новый алгоритм работы, недостаток — быстрый рост размерности мультиплексоров с ростом числа состояний и входов автомата. Структура с мультиплексным управлением триггерами показана на рис. 4.7. Входные сигналы $X_0 ... X_{m-1}$ и значения разрядов слова старого состояния $Q_0 ... Q_{n-1}$ образуют управ-

ляющее (адресное) входное слово мультиплексора, по которому выбираются значения разрядов нового слова состояния. Поступление тактового импульса ТИ вводит новое слово состояния в триггеры.

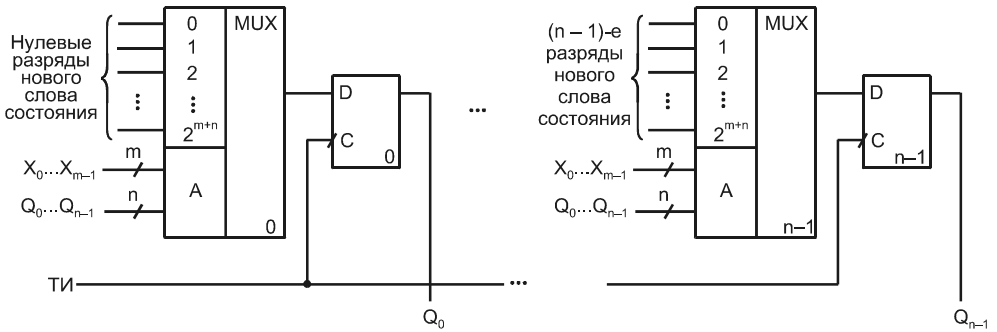


Рис. 4.7. Структура автомата на триггерах с мультиплексным управлением

Конкретная реализация автомата для рассматриваемого примера (рис. 4.8) не требует особых пояснений.

При нулевых исходных состояниях триггеров и M = 0 на адресные входы мультиплексоров поступает код 000 и на входах триггеров формируется комбинация сигналов 01. Поступление тактового импульса вводит эту комбинацию в триггеры. Теперь адресом для мультиплексоров становится комбинация 001, по которой с них снимается комбинация 10, поступающая по разрешению следующего тактового импульса в триггеры (регистр состояния). Так реализуется режим двоичного счетчика.

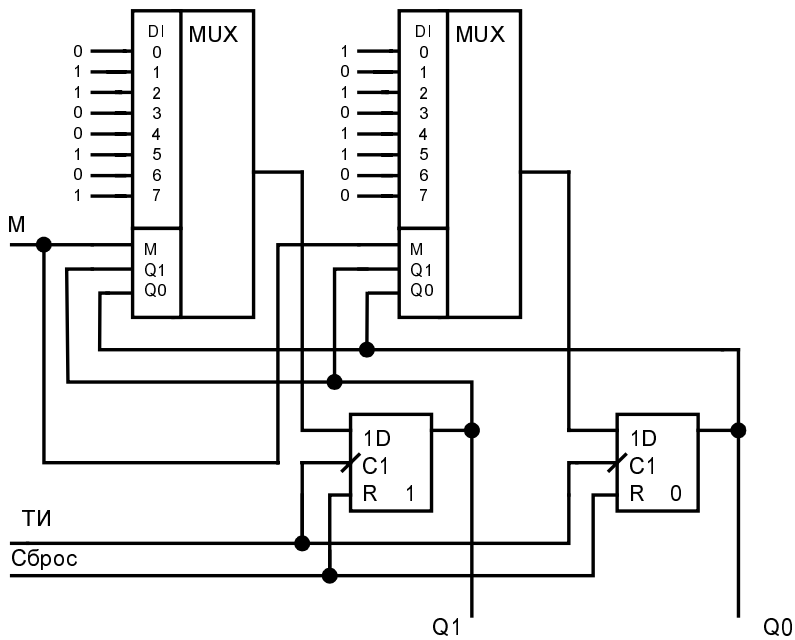


Рис. 4.8. Схема проектируемого автомата с мультиплексорами на входах триггеров

Изменение управляющего сигнала M меняет режим работы автомата. Если, например, при слове состояния 10 сигнал M становится единичным, то адрес мультиплексоров изменяется с 010 на 110 и с их выходов снимется комбинация 00, соответствующая следующему состоянию при работе счетчика в коде Грея.

Структура и работа автомата отличаются большой наглядностью, переход к другому алгоритму функционирования требует только смены сигналов на информационных входах мультиплексоров.

Вариант 4

Автомат с состояниями, кодируемыми в коде "1 из N"

Структура с кодированием типа "1 из N" (с кодированием ONE, One-Hot Encoding) содержит максимальное число триггеров, т. к. в ней для каждого состояния предусматривается триггер, находящийся в активном ("горячем") состоянии (пусть это состояние 1) при пассивном состоянии всех остальных. Рост числа триггеров при переходе от двоичного кодирования к ONE сопровождается *упрощением логики, обеспечивающей переходы автомата.* Функции возбуждения триггеров при ONE зависят от меньшего числа аргументов, т. к. декодирование состояния автомата не требует анализа состояний всех триггеров, и для декодирования достаточно одной переменной.

Уменьшение числа аргументов на входах логических преобразователей, вырабатывающих функции возбуждения, особенно полезно при реализации автомата на микросхемах программируемой логики типа FPGA, типовые блоки которых позволяют получать любые функции небольшого числа аргументов (чаще всего не более четырех). Еще одно достоинство ONE-кодирования — *высокое быстродействие автомата*, поскольку упрощение функций возбуждения триггеров сокращает логическую глубину соответствующих схем и коэффициенты разветвления их элементов, что позволяет повысить частоту тактирования схемы. Набор триггеров автомата образует структуру типа сдвигающего регистра — узла, допускающего эффективное размещение и трассировку в топологии СБИС. При увеличении числа состояний автомата его быстродействие остается приблизительно неизменным.

Проектирование автоматов с ONE-кодированием имеет особенности. *Диаграмма состояний автомата становится непосредственным инструментом проектирования.* Из рассмотренного ранее перечня этапов разработки автомата многие этапы исключаются, остаются лишь следующие:

- функциональное описание автомата;
- составление диаграммы его состояний (графа переходов);
- построение логической схемы.

В нашем примере граф переходов автомата имеет вид рис. 4.9, *а*, где состояния S0…S3 кодируются следующим образом: S0 = 1000, S1 = 0100, S2 = 0010, S3 = = 0001. Для каждого SI (I = 0…3) соответствующий триггер находится в единичном состоянии, а остальные три — в нулевом.

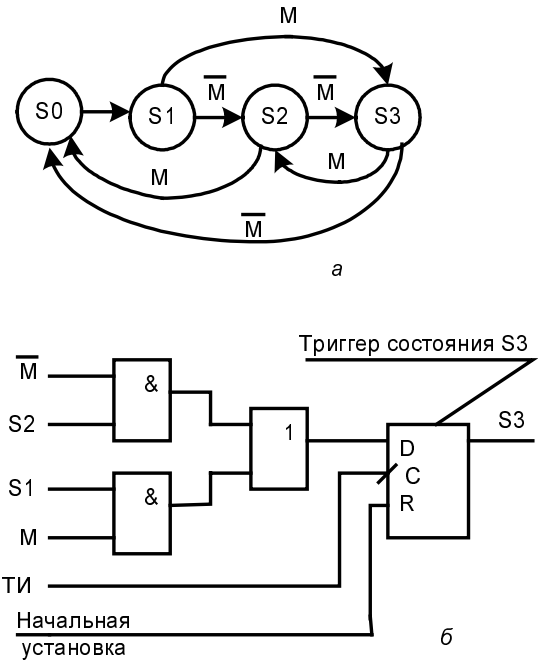


Рис. 4.9. Диаграмма состояний проектируемого автомата с ОНЕ-кодированием (*а*) и схема реализации состояния автомата (*б*)

Для получения схемы автомата следует найти функции возбуждения всех триггеров. Поиск этих функций проводится с помощью одних и тех же приемов и может быть проиллюстрирован на примере какого-либо состояния. Возьмем для примера состояние S3. Вначале для рассматриваемого состояния подсчитывается число входящих условных дуг, в данном случае их две. На входе триггера включается элемент ИЛИ с числом входов, равным числу входящих условных дуг. Далее для каждого входа элемента ИЛИ формируется конъюнкция сигналов от предыдущего состояния и условий перехода. В результате получается схема реализации состояния, показанная на рис. 4.9, *б*. Переход в следующее состояние происходит как переход единственной единицы из триггера в соседний триггер.

Схема автомата в целом при реализации функций возбуждения триггеров на элементах И-НЕ показана на рис. 4.10. Вентили ИЛИ на выходе схемы служат для перехода при необходимости от ОНЕ-кодирования к двоичному кодированию состояний, аналогичному применявшемуся в предыдущих вариантах реализации автомата.

Примечание

Для упрощения рисунка применена так называемая "линия групповых связей", показанная утолщенной линией (ранее она еще не встречалась). К ней подключаются выходы и входы коммутируемых элементов, что позволяет задать все их межсоединения, не проводя индивидуальные линии связей. Для изображений более или менее сложных схем применение линий групповых связей является стандартным приемом.

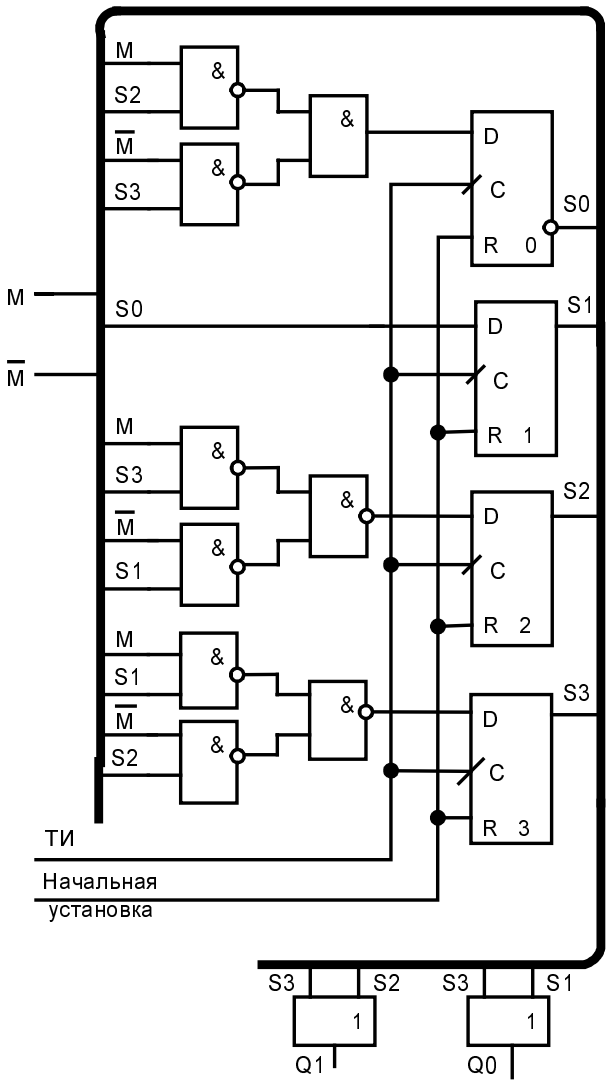
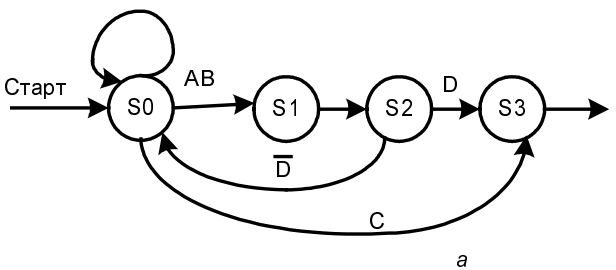


Рис. 4.10. Схема автомата с ОНЕ-кодированием состояний

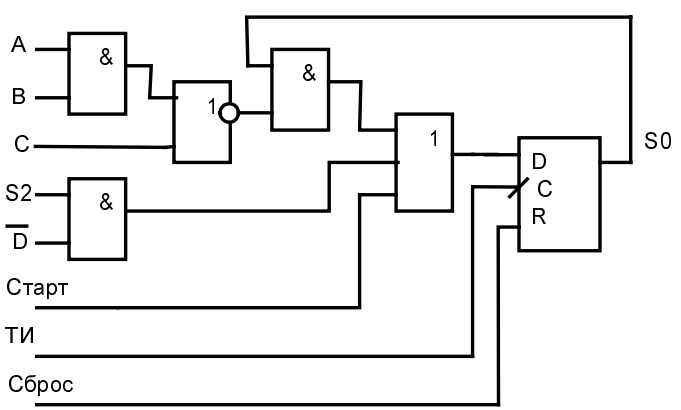
Специфическая ситуация может возникать при установке исходного состояния автомата. Если набор триггеров выполнен как единая ИС, то сброс сигналом начальной установки переведет все триггеры в нулевое состояние, тогда как первый слева триггер должен получать единичное состояние. Для создания эквивалента нужной ситуации можно взять выход левого триггера с инверсного вывода, что после сброса даст на выходах триггеров состояние 10…0. Чтобы не изменилось функционирование схемы, на вход левого триггера также следует подавать инвертированный сигнал (см. рис. 4.10).

Следует отметить и еще одну специфическую ситуацию, которая отсутствует в рассмотренном варианте автомата, но может встретиться в других. Речь идет об обес-

печении режима хранения состояния автомата в случае отсутствия условий перехода к следующим состояниям, но при сохранении сигналов тактирования. Диаграмма состояний автомата с такой ситуацией изображена на рис. 4.11, *а.* Состояние S0 имеет "дугу-ухо", отображающую его сохранение при отсутствии условий перехода к следующим состояниям. Чтобы реализовать требуемую работу автомата, в этом случае ко входам элемента ИЛИ, включенного на входе соответствующего триггера, добавляется еще один вход, логические условия для которого обеспечивают сохранение неизменного состояния при отсутствии условий перехода. Схема реализации состояния S0 приобретает вид, показанный на рис. 4.11, *б*.



а



б

Рис. 4.11. Диаграмма состояний автомата (*а*) и реализация состояния с режимом хранения (*б*)

§ 4.2. Регистры и регистровые файлы

Регистры — самые распространенные узлы цифровых устройств. Они оперируют со связанными переменными, составляющими слово. Над словами выполняется ряд операций: прием, выдача, хранение, сдвиг в разрядной сетке, поразрядные логиче-

ские операции. Регистры состоят из разрядных схем, в которые входят триггеры и, чаще всего, также и логические элементы.

По числу линий передачи сигналов регистры делятся на однофазные и парафазные, по системе синхронизации на однотактные, двухтактные и многотактные. Однако главным классификационным признаком является способ приема и выдачи данных. По этому признаку различают *параллельные* (статические) регистры, *последовательные* (сдвигающие) и *параллельно-последовательные*.

В параллельных регистрах прием и выдача слов производятся по всем разрядам одновременно. В последовательных регистрах слова принимаются и выдаются разряд за разрядом. Их называют *сдвигающими*, т. к. тактирующие сигналы при вводе и выводе слов перемещают их в разрядной сетке. Сдвигающий регистр может быть нереверсивным (с однонаправленным сдвигом) или *реверсивным* (с возможностью сдвига в обоих направлениях).

Последовательно-параллельные регистры имеют входы-выходы последовательного и параллельного типа. Имеются варианты с последовательным входом и параллельным выходом (*SIPO*, Serial Input — Parallel Output), параллельным входом и последовательным выходом (*PISO*, Parallel Input — Serial Output), а также варианты с возможностью любого сочетания способов приема и выдачи слов.

В параллельных (статических) регистрах схемы разрядов не обмениваются данными между собой. Общими для разрядов обычно являются цепи тактирования, сброса/установки, разрешения выхода или приема данных, т. е. цепи управления. Пример схемы статического регистра, построенного на триггерах типа D с прямым динамическим тактированием, имеющего входы сброса $\overline{R}$, разрешения выхода OE и выходы с третьим состоянием показан на рис. 4.12.

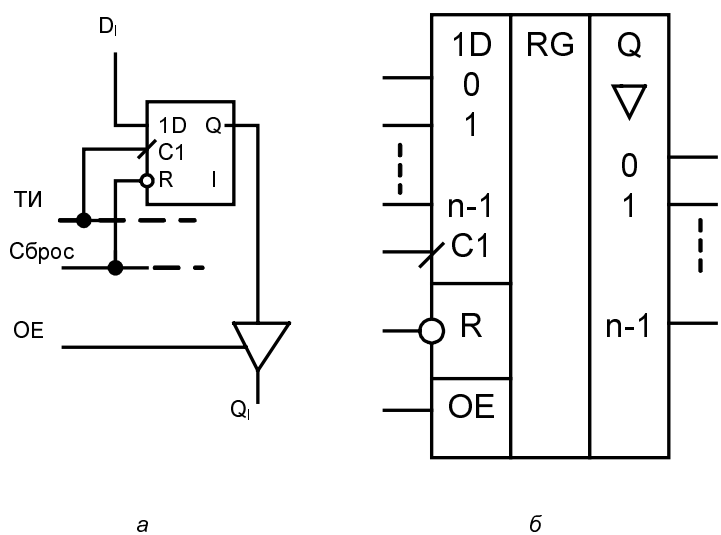


Рис. 4.12. Схема одного разряда (*а*)
и условное графическое обозначение (*б*) статического регистра

Для современной схемотехники характерно построение регистров именно на D-триггерах, преимущественно с динамическим управлением. Многие имеют выходы с третьим состоянием, некоторые регистры относятся к числу буферных, т. е. рассчитаны на работу с большими емкостными и/или низкоомными активными нагрузками. Это обеспечивает их работу непосредственно на магистраль (без дополнительных схем интерфейса).

Регистровые файлы

Из статических регистров составляются блоки регистровой памяти — *регистровые файлы*. В микросхеме ИР26 (серии КР1533 и др.) можно хранить 4 четырехразрядных слова с возможностью независимой и одновременной записи одного слова и чтения другого. Информационные входы регистров соединены параллельно (рис. 4.13). Входы адресов записи WA и WB (W от Write) дают четыре комбинации, каждая из которых разрешает "защелкнуть" в соответствующем регистре данные, присутствующие в настоящее время на выводах $D_{1–4}$. Содержимое регистра появляется на выходах $Q_{1–4}$ в соответствии с сигналами, подаваемыми на адресные входы мультиплексора RA и RB (R от Read).

Если на входе разрешения записи $\overline{WE}$ (Write Enable) действует активный низкий уровень, то данные поступают в соответствующий регистр, при высоком уровне $\overline{WE}$ входы для данных и адресов запрещены. Выходные данные выдаются в прямом коде.

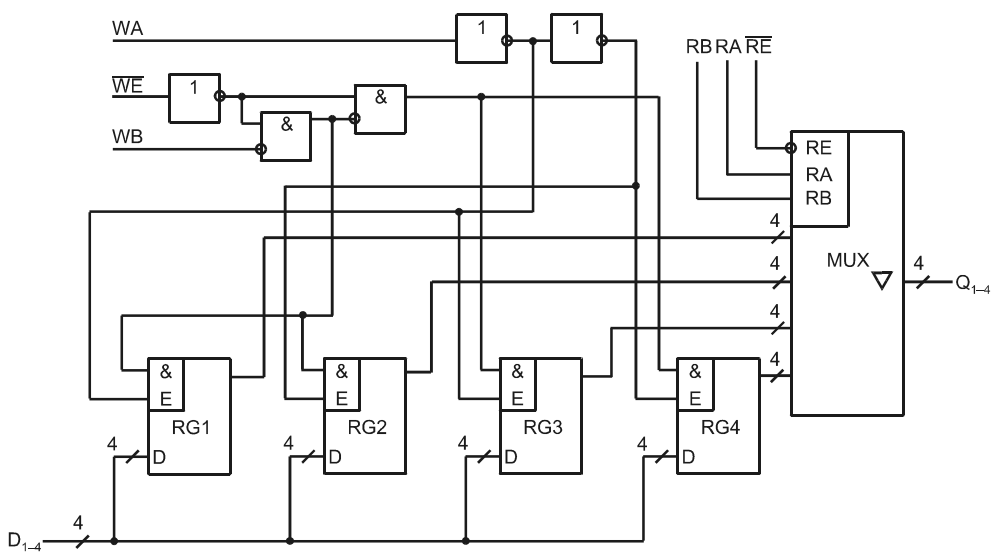


Рис. 4.13. Схема регистрового файла

Регистровую память можно наращивать, составляя ее из нескольких блоков. При наращивании числа хранимых слов выходы отдельных ИС с тремя состояниями соединяются в одной точке. Допускается соединять до 128 выходов, что дает 512 храни-

мых слов. При наращивании разрядности слов параллельно соединяют входы разрешения и адресации нескольких ИС, выходы которых в совокупности дают единое слово.

Сдвигающие регистры

Последовательные (сдвигающие) регистры представляют собою цепочку разрядных схем, связанных цепями переноса.

В однотактных регистрах со сдвигом (вправо — рис. 4.14, *а* и влево — рис. 4.14, *б*) слово сдвигается при поступлении синхросигнала "Сдвиг". Вход и выход последовательные (DSR — Data Serial Right, DSL — Data Serial Left). На рис. 4.14, *в* иллюстрируется принцип построения реверсивного регистра, в котором имеются связи триггеров с обоими соседними разрядами, но соответствующими сигналами разрешается работа только одних из этих связей (команды "влево" и "вправо" одновременно не подаются).

Следует отметить, что направление сдвига в регистрах не является геометрическим понятием, имеется в виду лишь направление сдвига в сторону старших или младших разрядов. В зависимости от варианта записи слова в регистр одна и та же схема может быть как регистром со сдвигом влево, так и регистром со сдвигом вправо. Поэтому схемотехнически существуют фактически два типа последовательных регистров: сдвигающие и реверсивные. Далее отдельно говорится о регистрах со сдвигами влево и вправо лишь для удобства последующего пояснения структуры реверсивного регистра.

Согласно требованиям тактирования, рассмотренным в предыдущей главе, в сдвигающих регистрах, не имеющих логических элементов в межразрядных связях, нельзя применять одноступенчатые триггеры, управляемые уровнем, поскольку некоторые триггеры могут за время действия разрешающего уровня синхросигнала переключиться неоднократно, что недопустимо (в этом случае слово или его части "поедут" по разрядной сетке, смещаясь не на один разряд, как это требуется, а на большее неконтролируемое число разрядов). Триггеры с динамическим управлением или двухступенчатые обеспечивают работоспособность регистра.

Появление в межразрядных связях логических элементов и, тем более, логических схем неединичной глубины упрощает выполнение условий работоспособности регистров, при этом иногда становится возможным и применение простейших триггеров.

Многотактные сдвигающие регистры управляются несколькими тактовыми последовательностями. Из их числа наиболее известны двухтактные с основным и дополнительным регистрами, построенными на простых одноступенчатых триггерах, управляемых уровнем. По такту С1 содержимое основного регистра переписывается в дополнительный, а по такту С2 возвращается в основной, но уже в соседние разряды, что соответствует сдвигу слова. По затратам оборудования и быстродействию этот вариант близок к однотактному регистру с двухступенчатыми триггерами.

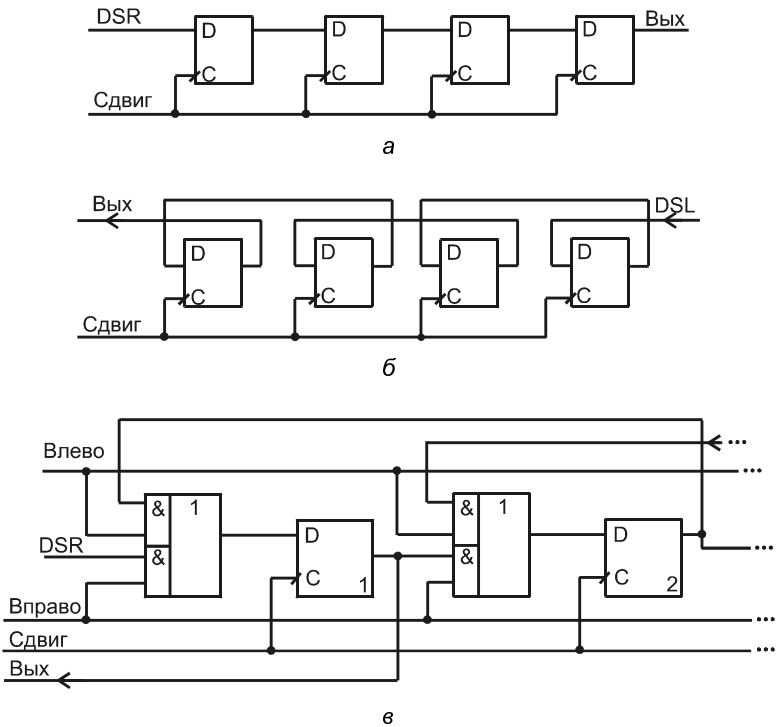


Рис. 4.14. Схемы регистров сдвига вправо (*а*), влево (*б*) и реверсивного (*в*)

Универсальные регистры

В сериях ИС и библиотеках БИС/СБИС имеется много вариантов регистров, в том числе универсальные (многорежимные, многофункциональные), способные выполнять набор операций. Многорежимность достигается композицией в одной и той же схеме частей, необходимых для выполнения различных операций. Управляющие сигналы, задающие вид выполняемой в данное время операции, активизируют необходимые для этого части схемы.

Таблица 4.4

Режим	Входы							Выходы				
	C	$\overline{R}$	SO	S1	DSR	DSL	D_n	Q_0	Q_1	...	Q_6	Q_7
Сброс	X	L	X	X	X	X	X	L	L	...	L	L
Хранение	⌐¬	H	L	L	X	X	X	Q_0	Q_1	...	Q_6	Q_7
Сдвиг вправо	⌐¬	H	H	L	X	L	X	Q_1	Q_2	...	Q_7	L
		H	H	L	X	H	X	Q_1	Q_2		Q_7	H

Таблица 4.4 *(окончание)*

Режим	Входы							Выходы				
	C	$\overline{R}$	SO	S1	DSR	DSL	D_n	Q_0	Q_1	...	Q_6	Q_7
Сдвиг влево	⌐	H	L	H	L	X	X	L	Q_0	...	Q_5	Q_6
		H	L	H	H	X	X	H	Q_0		Q_5	Q_6
Парал-лельная загрузка	⌐	H	H	H	X	X	D_n	D_0	D_1	...	D_6	D_7

Типичным представителем многорежимных регистров является микросхема ИР13 серии КР1533 и других (рис. 4.15). Это восьмиразрядный регистр с возможностью двусторонних сдвигов. Имеет также параллельные входы и выходы, вход асинхронного сброса $\overline{R}$ и входы выбора режима S_0 и S_1, задающие четыре режима (параллельная загрузка, два сдвига и хранение). Во избежание ошибок синхронизации изменение уровней сигналов на входах S_0 и S_1 должно осуществляться только при высоком уровне напряжения на тактовом входе C. Функционирование регистра определяется табл. 4.4. Условное обозначение приведено на рис. 4.16.

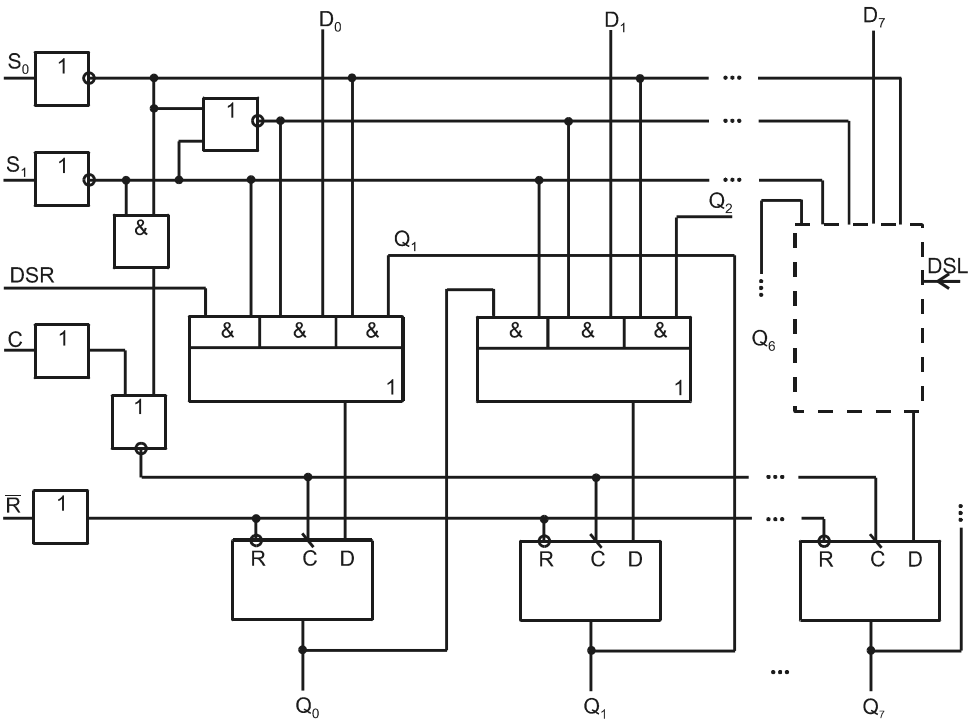


Рис. 4.15. Схема многорежимного регистра

Регистры, имеющие разнотипные вход и выход, служат основными блоками *преобразователей параллельных кодов в последовательные и обратно*. На рис. 4.17 пока-

зана схема преобразователя параллельного кода в последовательный на основе восьмиразрядного регистра типа SI/PI/SO. В этой схеме отрицательный стартовый импульс St, задающий уровень логического нуля на верхнем входе элемента 1, создает единичный сигнал параллельного приема данных на вход L (Load — загрузка), по которому в разряды 1...7 регистра загружается преобразуемое слово $D_{1–7}$, а в нулевой разряд — константа 0. На последовательный вход DSR подана константа 1.

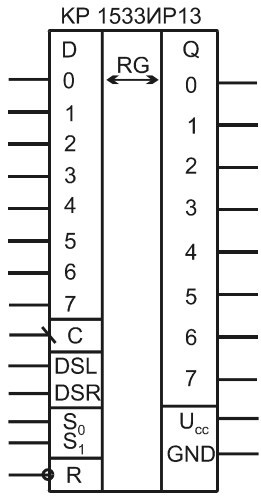


Рис. 4.16. Условное обозначение универсального регистра

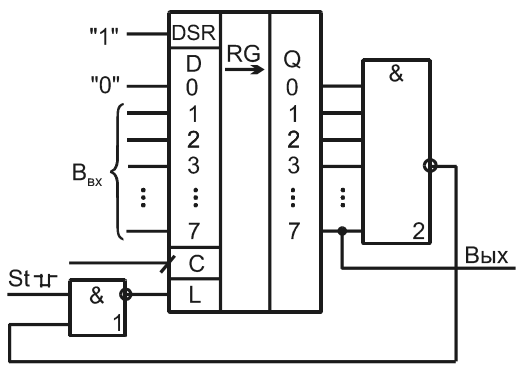


Рис. 4.17. Схема преобразователя параллельного кода в последовательный

Таким образом, после загрузки в регистре формируется слово $0D_1D_2...D_7$. Тактовые импульсы, поступающие на вход C, вызывают сдвиги слова вправо (для условного обозначения это соответствует сдвигу вниз). Сдвиги выводят слово в последовательной форме через выход Q_7. Вслед за информационными разрядами идет ноль (константа "0"), после которого цепочка единиц. Пока ноль не выведен из регистра, на выходе элемента 2 действует единичный сигнал. После вывода нуля все входы элемента 2 становятся единичными, его выход приобретает нулевое значение и через элемент 1 фор-

мирует сигнал автоматической загрузки следующего слова, после чего цикл преобразования повторяется.

В перечень операций, выполняемых регистрами, входят поразрядные *логические операции*. Современные регистры мало приспособлены для выполнения этих операций, однако при необходимости их можно выполнить, пользуясь регистрами на RS-триггерах. Для выполнения операции ИЛИ на S-входы статического регистра с исходным нулевым состоянием подается первое слово A, единичные разряды которого устанавливают соответствующие триггеры. Затем без сброса регистра на S-входы подается второе слово B. Ясно, что в результате получим результат Q = A\/B.

При выполнении поразрядной операции И в первом такте на S-входы регистра подается слово A, устанавливающее те разряды регистра, в которых слово A имеет единицы. Затем следует подать на регистр слово B. Чтобы в регистре сохранились единицы только в тех разрядах, в которых единицы имеют оба слова, слово B подается на входы R триггеров в инверсном виде.

Сложение по модулю 2 может быть выполнено схемой с триггерами типа T в разрядах путем последовательной во времени подачи на нее двух слов A и B.

§ 4.3. Основные сведения о счетчиках. Двоичные счетчики

Понятие "счетчик" является очень широким. *К счетчикам отнесем автоматы, которые под действием входных импульсов переходят из одного состояния в другое, фиксируя тем самым число поступивших на их вход импульсов в том или ином коде.*

Специфичной для счетчиков операцией является изменение их содержимого на единицу (может быть и условную). Прибавление такой единицы соответствует операции *инкрементации*, вычитание — операции *декрементации*. Обычно счетчиками выполняются также и другие операции — сброс, установка, параллельная загрузка и др.

Счетчик характеризуется *модулем счета M (емкостью)*, т. е. *числом возможных состояний счетчика*. После поступления на счетчик M входных сигналов начинается новый цикл, повторяющий предыдущий.

Классификация и режимы работы счетчиков

Классификация счетчиков. По *способу кодирования состояний* различают двоично-кодированные счетчики, счетчики Джонсона, счетчики с кодом "1 из N", счетчики с кодом Грея и др.

По *направлению счета* счетчики делятся на *суммирующие* (прямого счета), *вычитающие* (обратного счета) и *реверсивные* (с изменением направления счета).

Соответственно *организации межразрядных связей* различают счетчики с последовательным, параллельным и комбинированными переносами.

По *одновременности или разновременности срабатывания разрядов* различают синхронные и асинхронные счетчики.

Способы использования счетчиков. Возможные способы использования счетчика:

- регистрация числа поступивших на счетчик сигналов;
- деление частоты.

В первом варианте результат — содержимое счетчика, во втором выходными сигналами являются импульсы переполнения счетчика.

Быстродействие счетчика характеризуется временем установления в нем нового состояния (для первого режима), а также максимальной частотой входных сигналов f_{max}.

Как и любой автомат, счетчик можно строить на триггерах любого типа, однако удобнее всего использовать для этого триггеры типа Т (счетные) и JK, имеющие при J = K = 1 счетный режим.

Состояние счетчика читается по выходам разрядных схем как слово $Q_{n-1}Q_{n-2}...Q_0$, входные сигналы поступают на младший разряд счетчика.

> ***Примечание***
>
> Согласно стандарту входы цифровых устройств располагают с левой стороны их условных обозначений. Входные сигналы счетчиков поступают на их младшие разряды, поэтому положение чисел в счетчике оказывается обратным по отношению к привычному — старшие разряды находятся справа.

Двоичные счетчики

Двоичные счетчики имеют модуль $M = 2^n$, где n — целое число. Коды состояний счетчика образуют последовательность двоичных чисел, десятичными эквивалентами которых будут числа 0, 1, 2, 3, …, M – 1. Максимальное число в счетчике равно M – 1.

Асинхронные счетчики

Схему *асинхронного двоичного счетчика* можно получить наглядным путем с помощью рассмотрения его работы. Таблица истинности двоичного счетчика — последовательность двоичных чисел от нуля до M – 1. Наблюдение за разрядами чисел, составляющих таблицу, приводит к пониманию структурной схемы двоичного счетчика. Состояния младшего разряда при его просмотре по соответствующему столбцу таблицы показывают чередование нулей и единиц вида 01010101..., что естественно, т. к. младший разряд принимает входной сигнал и переключается от каждого входного воздействия. В следующем разряде наблюдается последовательность пар нулей и единиц вида 00110011... . В третьем разряде образуется последовательность из четверок нулей и единиц 00001111... и т. д. Из этого наблюдения

видно, что следующий по старшинству разряд переключается с частотой, в два раза меньшей, чем данный.

Известно, что счетный триггер делит частоту входных импульсов на два. Сопоставив этот факт с указанной выше закономерностью, видим, что счетчик может быть построен в виде *цепочки последовательно включенных счетных триггеров* (рис. 4.18, *а*). Представление счетчика цепочкой Т-триггеров справедливо как для суммирующего, так и для вычитающего вариантов, поскольку закономерность по соотношению частот переключения разрядов сохраняется как при просмотре таблицы сверху вниз (прямой счет), так и снизу вверх (обратный счет). Различия при этом состоят в направлении переключения предыдущего разряда, вызывающего переключение следующего.

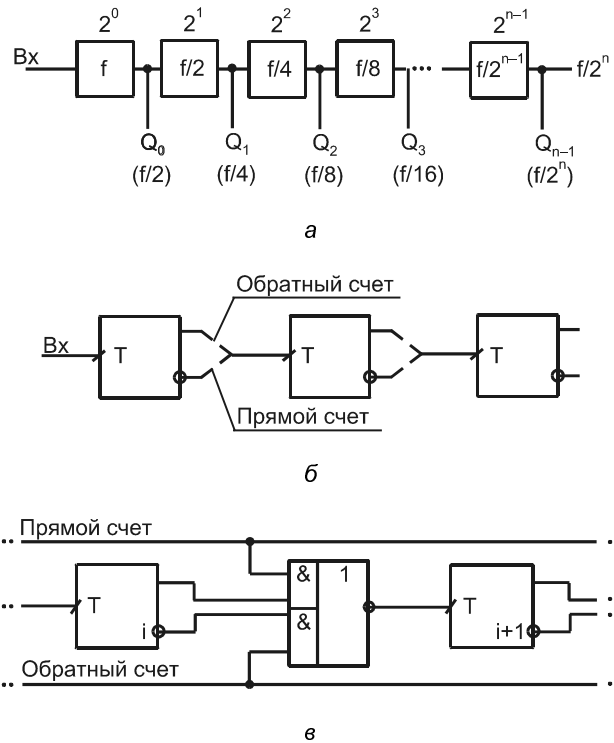


Рис. 4.18. Структура последовательного счетчика (*а*), ее реализация на триггерах с прямым динамическим управлением (*б*) и межразрядные связи реверсивного счетчика (*в*)

При прямом счете следующий разряд переключается при переходе предыдущего в направлении 1-0, а при обратном — при переключении 0-1. Следовательно, различие между вариантами заключается в разном подключении входов триггеров к выходам предыдущих. Если схема строится на счетных триггерах с прямым динамическим управлением, то характер подключения следующих триггеров к предыдущим для получения счетчиков прямого и обратного счета будет соответствовать рис. 4.18, *б*.

Из различия вариантов прямого и обратного счета следует также и способ построения *реверсивного счетчика* (рис. 4.18, *в*) путем переноса точки съема сигнала на противоположный выход триггера под действием управляющего сигнала (с помощью элемента И-ИЛИ-НЕ, как показано на рисунке, либо элемента И-ИЛИ).

Полученные структуры относятся к *асинхронным счетчикам*, т. к. в них каждый триггер переключается выходным сигналом предыдущего, и эти переключения происходят не одновременно. Переключение одного триггера за другим есть не что иное, как распространение переноса по разрядам числа при изменении содержимого счетчика. В худшем случае перенос распространяется по всей разрядной сетке от младшего разряда к старшему, т. е. для установления нового состояния должны переключиться последовательно все триггеры. Отсюда видно, что время установления кода в асинхронном счетчике составит величину $t_{уст} \le nt_{тг}$. Другим названием асинхронного счетчика является название *"последовательный счетчик"*.

Максимальная частота входных импульсов в режиме деления частоты ограничивается возможностями триггера младшего разряда, т. к. все последующие разряды переключаются с более низкими частотами.

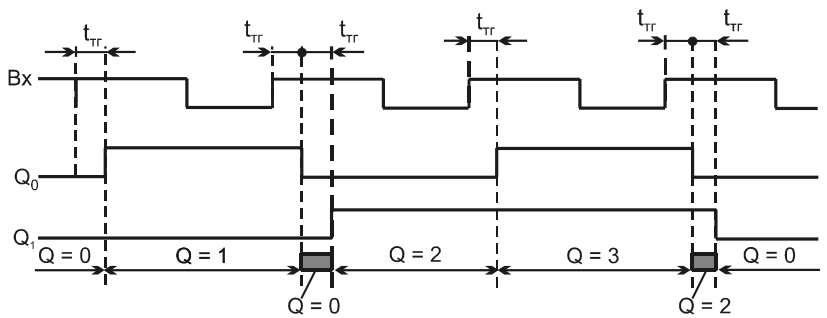


Рис. 4.19. Временные диаграммы работы последовательного двоичного счетчика

Особенностью последовательных счетчиков является возникновение в переходных процессах ложных состояний из-за задержек переключения триггеров. На рис. 4.19 показана временная диаграмма работы двухразрядного суммирующего счетчика на триггерах с прямым динамическим управлением, построенная с учетом задержек переключения триггеров $t_{тг}$. Читая состояние счетчика Q по потенциалам на выходах триггеров Q_0 и Q_1, видим, что после состояний 1 и 3 появляются ложные состояния 0 и 2 (показаны штриховкой). Опасность воздействия коротких ложных импульсов на ЦУ заставляет прибегать при необходимости к стробированию выхода счетчика.

Синхронные счетчики

Максимальным быстродействием обладают *синхронные счетчики с параллельным переносом*. Структура такого счетчика может быть задана в виде совокупности *n*-разрядных схем, содержащих триггеры и логические схемы для выработки функ-

ций возбуждения триггеров. Функции возбуждения можно найти формальным путем, как указано в предыдущих разделах. Рассмотрим трехразрядный счетчик. Функционирование счетчика определяется табл. 4.5.

Таблица 4.5

Q_2	Q_1	Q_0	$Q_{2н}$	$Q_{1н}$	$Q_{0н}$	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Из таблицы получаем следующие выражения для функций возбуждения триггеров:

$$T_0 = 1,$$

$$T_1 = \overline{Q}_2\overline{Q}_1Q_0 \vee \overline{Q}_2Q_1Q_0 \vee Q_2\overline{Q}_1Q_0 \vee Q_2Q_1Q_0 = Q_0,$$

$$T_2 = \overline{Q}_2Q_1Q_0 \vee Q_2Q_1Q_0 = Q_1Q_0.$$

Согласно полученному результату функция возбуждения триггера представляет собой конъюнкцию выходных сигналов всех предыдущих разрядов (младший разряд переключается от каждого входного воздействия и для него T = 1). Эта закономерность может быть обобщена на произвольное число разрядов, т. е. можно записать формулу:

$$T_i = Q_{i-1}Q_{i-2} \ldots Q_0.$$

Схема четырехразрядного счетчика с JK-триггерами, работающими в режиме счетных триггеров, приведена на рис. 4.20, *а*.

Время установления нового состояния для счетчиков с параллельным переносом (если не учитывать зависимости задержек элементов от нагрузок на них) не зависит от их разрядности и равно времени переключения триггера $t_{тг}$. Длительность цикла равна $t_{тг} + t_k$, где t_k — задержка коньюнктора.

Схема межразрядной связи для реверсивного счетчика с сигналом U/D (Up/Down, т. е. прямо/обратно) показана рис. 4.20, *б*.

С ростом числа разрядов реализация параллельных счетчиков затрудняется — требуются вентили с большим числом входов, растет нагрузка на выходы триггеров и вследствие этого падает быстродействие счетчика.

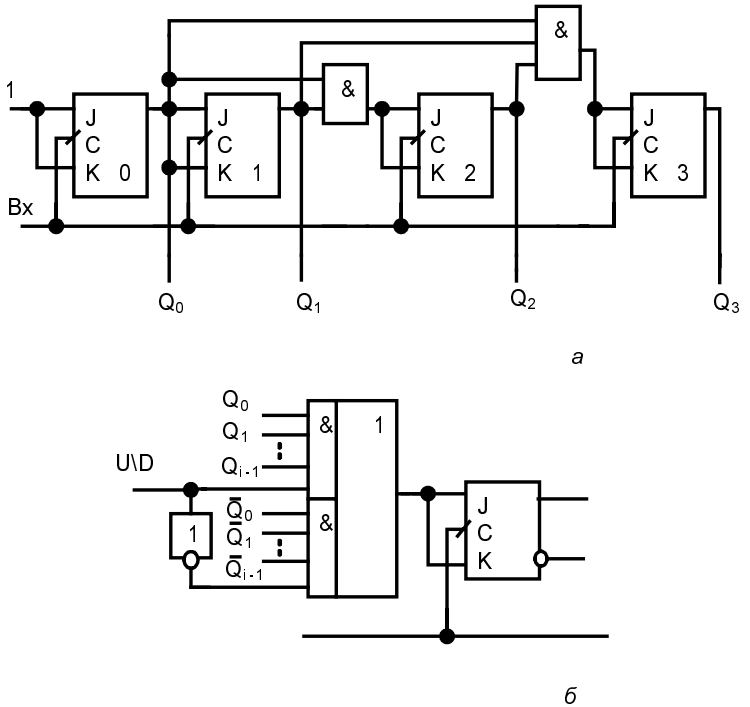


Рис. 4.20. Схемы параллельных счетчиков прямого счета (*а*) и реверсивного (*б*)

Счетчики с групповой структурой

Широкое распространение получили счетчики с групповой структурой, в которых счетчик разбивается на группы, связанные цепями межгруппового переноса (рис. 4.21, *а*). При единичном состоянии всех триггеров группы приход очередного входного сигнала создаст перенос из этой группы. Эта ситуация подготавливает межгрупповой конъюнктор к прямому пропусканию входного сигнала на следующую группу. В наихудшем для быстродействия случае, когда перенос проходит через все группы и поступает на вход последней,

$$t_{уст} = t_k (\ell - 1) + t_{гр},$$

где ℓ — число групп; $t_{гр}$ — время установления кода в группе.

Если уменьшить разрядность группы до единицы и использовать синхронные Т-триггеры, то получится схема *синхронного счетчика с последовательным переносом* (рис. 4.21, *б*). Схема относится к числу синхронных, т. к. все триггеры срабатывают одновременно под действием единого входного сигнала. В этом проявляется быстрая реакция схемы на входной сигнал, такая же, как и в счетчике с параллельным переносом. Однако по максимальной частоте входных сигналов эта схема существенно отличается от схемы с параллельным переносом, т. к. до подачи

нового входного сигнала требуется дать цепочке вентилей установиться в новое состояние путем их последовательного переключения.

В развитых сериях ИС обычно имеется по 5...10 вариантов двоичных счетчиков, выполненных в виде 4-разрядных групп (секций). Каскадирование секций может выполняться путем их последовательного включения по цепям переноса, организации параллельно-последовательных переносов или для более сложных счетчиков с двумя дополнительными управляющими входами разрешения счета и разрешения переноса путем организации параллельных переносов и в группах и между ними (см. например, [32]).

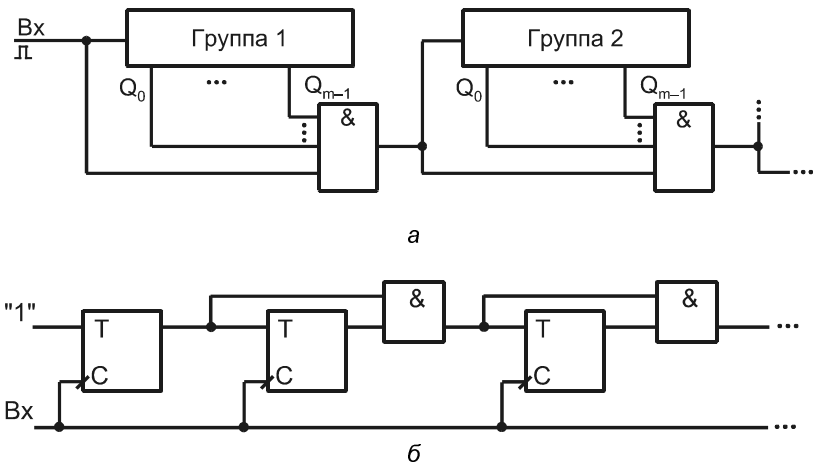


Рис. 4.21. Схемы счетчиков групповой структуры

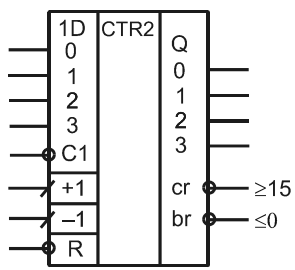


Рис. 4.22. Условное обозначение двоичного реверсивного счетчика со сбросом, параллельной загрузкой и выходами переноса и заема (carry, borrow)

Особенность двоичных счетчиков синхронного типа — наличие ситуаций с одновременным переключением всех его разрядов (например, для суммирующего счетчика при переходе от кодовой комбинации 11...1 к комбинации 00...0 при переполнении счетчика и выработке сигнала переноса). *Одновременное переключение многих триггеров создает значительный токовый импульс в цепях питания* ЦУ и может привести к сбою в их работе (*см. § 1.3*). Поэтому в руководящих материалах

по использованию микросхем иногда указываются ограничения на разрядность двоичных счетчиков (например, 16). Для счетчиков большей разрядности рекомендуется переходить к коду Грея, у которого переходы от одной кодовой комбинации к другой сопровождаются переключением всего одного разряда. Не более двух разрядов одновременно переключаются и в счетчике Джонсона. Правда, для получения результата счета в двоичном коде в этих случаях придется использовать дополнительно преобразователь кода, но зато исключаются токовые импульсы большой интенсивности в цепях питания.

Пример условного обозначения счетчика приведен на рис. 4.22.

§ 4.4. Двоично-кодированные счетчики с произвольным модулем

Счетчики с модулем, не равным целой степени числа 2, т. е. с произвольным модулем, реализуются на основе нескольких методов.

Для построения счетчика с произвольным модулем M исходной структурой служит двоичный счетчик с модулем 2^n, превышающим требуемый и ближайшим к нему, так что разрядность счетчика n =] $\log_2$M [, где]…[— знак округления до ближайшего справа целого числа. Такой двоичный счетчик имеет 2^n – M = L лишних (неиспользуемых) состояний, подлежащих исключению.

Способы *исключения лишних состояний* разнообразны, и для любого M можно предложить много реализаций счетчика. Исключая некоторое число первых состояний, получим ненулевое начальное состояние счетчика, что приводит к отсутствию естественного порядка счета и регистрации в счетчике кода с избытком. Исключение последних состояний позволяет сохранить естественный порядок счета. Сложность обоих вариантов принципиально одинакова, поэтому далее будем ориентироваться на схемы с естественным порядком счета. Состояния счетчиков предполагаем закодированными двоичными числами, т. е. будем рассматривать двоично-кодированные счетчики.

В счетчиках с исключением последних состояний счет ведется обычным способом вплоть до достижения числа M – 1. Далее последовательность переходов счетчика в направлении роста регистрируемого числа должна быть прервана, и следующее состояние должно быть нулевым. При этом счетчик будет иметь M внутренних состояний (от 0 до M – 1), т. е. его модуль будет равен M.

Остановимся на двух способах построения счетчиков с произвольным модулем: *модификации межразрядных связей* и *управлении сбросом*. При построении счетчика с модифицированными межразрядными связями последние, лишние, состояния исключаются непосредственно из таблицы функционирования счетчика. Далее обычным для синтеза автоматов способом строится счетчик, специфика которого состоит в нестандартных функциях возбуждения триггеров и, следовательно, в нестандартных связях между триггерами, что и объясняет название способа. Схема получается как специализированная, изменение модуля счета требует изменения самой схемы,

т. е. легкость перестройки с одного модуля на другой отсутствует. В то же время реализация схемы счетчика может оказаться простой.

При управлении сбросом выявляется момент достижения содержимым счетчика значения M – 1. Это является сигналом сброса счетчика в следующем такте, после чего начинается новый цикл. Такой вариант обеспечивает легкость перестройки счетчика на другие значения модуля, т. к. требуется изменять лишь код, с которым сравнивается содержимое счетчика для выявления момента сброса.

Счетчики с модифицированными межразрядными связями

Построение счетчика проиллюстрируем примером для M = 5, начав с таблицы (табл. 4.6).

Таблица 4.6

Исходное состояние			**Следующее состояние**			**Функции возбуждения**					
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

При нахождении функций возбуждения триггеров использован "словарь" (*см. § 3.1*). Имея в виду, что вместо символа произвольного сигнала X можно подставлять любую переменную (0 или 1), на основании таблицы запишем: $J_2 = Q_1Q_0$ (в столбце J_2 оставлена всего одна единица), $J_1 = Q_0$, $J_0 = \overline{Q}_2$. Для функций K_i (i = 0, 1, 2) выберем варианты с наибольшим числом констант, чтобы меньше нагружать источники сигналов. Примем, что $K_2 = 1$, $K_1 = J_1$ и $K_0 = 1$.

Схема счетчика приведена на рис. 4.23.

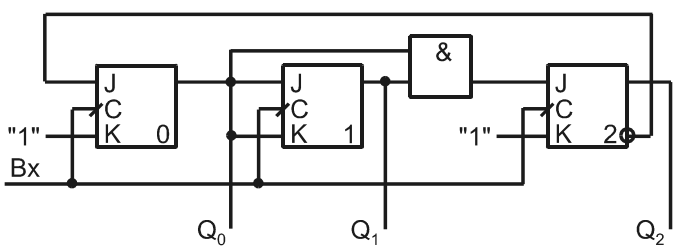


Рис. 4.23. Схема счетчика с модулем 5

Анализ "лишних" состояний. В спроектированной схеме лишние состояния исключены в том смысле, что они не используются при нормальном функционировании счетчика. Но при сбоях или подаче на схему напряжения питания в начале ее работы *лишние состояния могут возникать*. Поэтому полезно определить поведение схемы, в которой возникло лишнее состояние. Имея схему, можно полностью предсказать ее поведение во всех возможных ситуациях. Сделаем это для полученной схемы счетчика с модулем 5.

Взяв каждое лишнее состояние, найдем для него функции возбуждения триггеров, определяющие их переходы в следующее состояние. При необходимости найдем таким же способом следующий переход и т. д. Для взятого примера лишними являются состояния 101, 110 и 111.

В состоянии 101 $Q_2 = 1$, $Q_1 = 0$ и $Q_0 = 1$. Зная функции возбуждения триггеров, находим, что $J_0 = 0$, $K_0 = 1$, $J_1 = K_1 = 1$, $J_2 = 0$, $K_2 = 1$. Следовательно, триггеры 0 и 2 сбросятся, а триггер 1 переключится в противоположное текущему состояние и из лишнего состояния 101 счетчик перейдет в состояние 010.

Аналогичным способом можно получить результаты для состояний 100 и 111. В итоге можно построить диаграмму состояний счетчика (граф переходов), в которой учтен не только рабочий цикл (он показан кружками), но и поведение схемы, попавшей в неиспользуемые состояния (показаны прямоугольниками). Из диаграммы (рис. 4.24) видно, что рассматриваемый счетчик обладает *свойством самозапуска* (самовосстановления после сбоя) — независимо от исходного состояния после начала работы он приходит в рабочий цикл. При разработке некоторых схем в них вводят специальные элементы или подсхемы для придания свойств самозапуска.

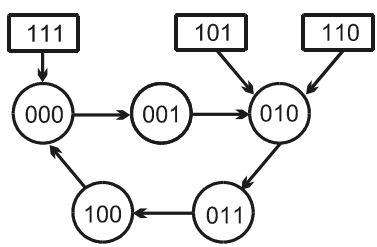


Рис. 4.24. Диаграмма состояний счетчика с модулем 5

Среди счетчиков с произвольным модулем особое место занимают двоично-десятичные, имеющие модуль 10. В сериях ИС обычно имеются идентичные по прочим признакам счетчики с модулями 16 и 10. Счетчик с модулем 10 нетрудно построить формально проиллюстрированным выше методом.

Пример стандартной микросхемы счетчика

Схема счетчика ИЕ2, которая появилась в самых первых сериях ИС и до сих пор повторяется в более новых (рис. 4.25), состоит фактически из двух секций с модулями 2 и 5, представленных триггером T_0 и группой $T_1T_2T_3$, на которой собран счетчик с модулем 5. Секции можно использовать по отдельности или соединять

последовательно с помощью внешней коммутации выводов для получения двоично-десятичного счетчика. Имеется сброс по конъюнкции сигналов R_0R_1 и установка в состояние 1001 по конъюнкции сигналов S_0S_1. Режимы неиспользуемых входов не показаны.

Соединение счетчиков в порядке mod2-mod5 дает двоично-десятичный счетчик с естественной последовательностью счета, который в режиме делителя частоты формирует импульсы со скважностью 5. Соединение в порядке mod5-mod2 дает, естественно, тот же модуль счета, но состояния счетчика образуют последовательность чисел 0, 1, 2, 3, 4, 8, 9, 10, 11, 12, после которой цикл повторяется. В режиме делителя частоты формируются импульсы со скважностью 2. Таким образом, разбиение счетчика на две секции предоставляет возможность получить как обычный двоично-десятичный счетчик, так и два делителя частоты на 10 — с формированием узких импульсов ($t_и$ = T/5) и симметричных импульсов ($t_и$ = T/2), где T — период повторения импульсов.

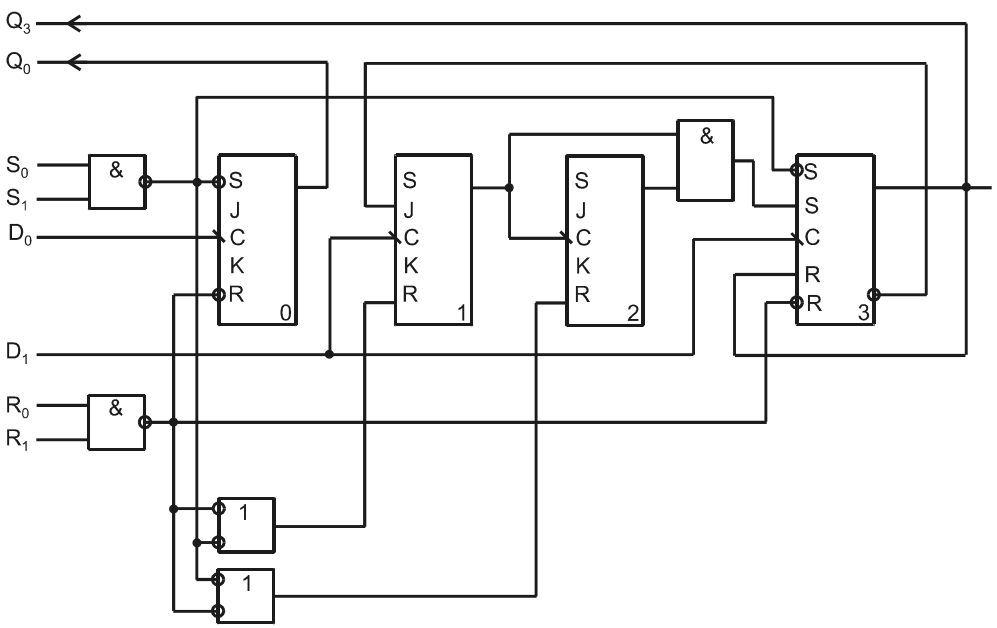


Рис. 4.25. Схема счетчика ИЕ2 серии КР1533

Наряду с секционированным двоично-десятичным счетчиком в сериях ИС имеются и обычные с различными сочетаниями классификационных признаков (до 5...10 вариантов).

Счетчики с управляемым сбросом

Второй метод построения счетчиков с произвольным модулем — метод управляемого сброса — позволяет изменять модуль счета очень простым способом, не требующим изменений самой схемы счетчика.

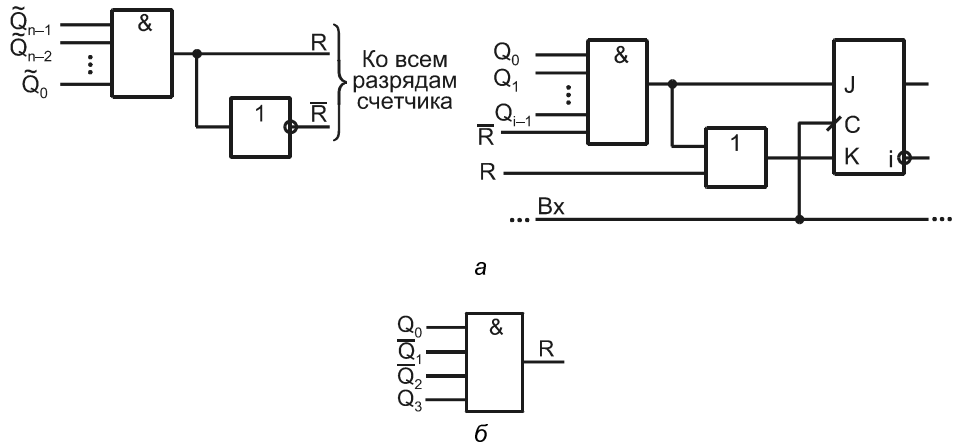


Рис. 4.26. Схема счетчика с управляемым сбросом (*а*) и схема выработки сигнала сброса для двоично-десятичного счетчика (*б*)

Рассмотрим этот способ применительно к реализации синхронного счетчика с параллельным переносом. Функции возбуждения двоичного счетчика указанного типа имеют вид $J_i = K_i = Q_0Q_1...Q_{i-1}$ (в младшем триггере $J_0 = K_0 = 1$). Введем в эти функции сигнал сброса R, изменив их следующим образом:

$$J_i = (Q_0Q_1...Q_{i-1})\overline{R},$$

$$K_i = J_i \vee R.$$

Пока сигнал сброса отсутствует (R = 0), функции J_i и K_i не отличаются от соответствующих функций двоичного счетчика. Когда сигнал R приобретает единичное значение, все функции J_i становятся нулевыми, а K_i — единичными, что заставляет все триггеры сброситься в следующем такте. Если сигнал R возникнет при появлении в счетчике числа M – 1, то будет реализована последовательность счета 0, 1, 2, ..., M – 1, 0..., т. е. счетчик с модулем M.

Схемы разрядов счетчика с управляемым сбросом не зависят от модуля счета. Кроме разрядных схем, счетчик содержит один конъюнктор, вырабатывающий сигнал сброса при достижении содержимым счетчика значения M – 1 (рис. 4.26, *а*).

Если, например, имеется четырехразрядный счетчик, и на входы конъюнктора выработки сигнала сброса подключены выходы триггеров, как показано на рис. 4.26, *б*, то сброс произойдет после достижения счетчиком числа 1001 = 9, т. е. счетчик будет работать как двоично-десятичный.

§ 4.5. Счетчики с недвоичным кодированием

Наибольшее практическое значение среди счетчиков с недвоичным кодированием состояний имеют счетчики с кодом Грея, счетчики Джонсона и счетчики с кодом "1 из N".

Счетчики в коде Грея

В коде Грея при переходе от любой кодовой комбинации к следующей изменяется только один разряд. Этот код известен с 70-х годов XIX века, однако оказался востребованным и получил свое название только в 50-х годах XX века, когда Ф. Грей применил его для построения преобразователя механических перемещений в цифровой код. Механические перемещения преобразуются в цифровые коды с помощью масок, примеры которых приведены на рис. 4.27.

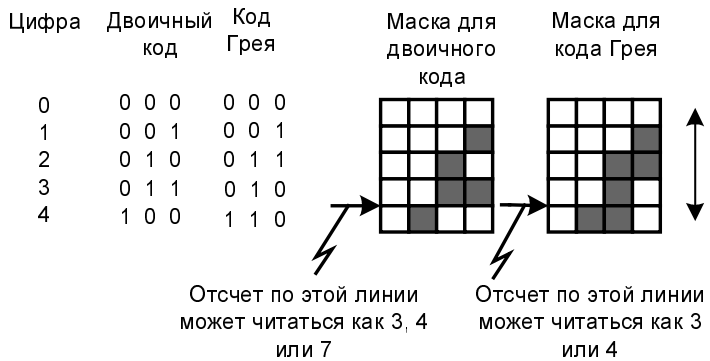


Рис. 4.27. Маски преобразователей перемещений в цифровые коды

Маски перемещаются относительно линейки считывающих элементов (оптических, индукционных и др.), так что под считывающими элементами оказывается отображение той или иной кодовой комбинации. Для наглядности на рис. 4.27 показаны фрагменты масок для восприятия поступательных движений, хотя чаще всего в цифровые коды нужно преобразовывать угловые перемещения. В этом случае маски выполняются в виде дисков, секторы которых соответствуют строкам прямоугольных масок, принципиальных изменений в ситуацию это не вносит.

Маски для двоичных кодов вносят в работу преобразователей грубые ошибки. Дело в том, что на границе строк возможны несколько вариантов считывания информации, поскольку граница в равной мере принадлежит двум смежным строкам. Для изображенного фрагмента маски двоичного кода, например, на границе четвертой и пятой строки могут считываться не только значения соседних строк (3 или 4), но и грубо неверная комбинация (7). Такие ситуации для построения преобразователей неприемлемо.

Маски для кода Грея на границах между строками, где возникают неопределенности отсчетов, дают одну из соседних цифр, отличающихся друг от друга ценой младшего разряда кода, что совершенно естественно для отображения дискретных данных. Грубых ошибок маски для кодов Грея не дают, и их применение позволило создать преобразователи механических перемещений в цифровую форму.

В схемотехнике счетчиков применение кода Грея устраняет одновременное переключение многих разрядов, характерное для двоичных счетчиков при их перепол-

нениях и создающее в цепях питания токовые импульсы, которые могут вызывать сбои в работе схемы (*см. § 1.3*).

Структура счетчика Грея обычна — это n разрядов, схема каждого из которых состоит из триггера и схем выработки функций возбуждения. Функции возбуждения можно получить рассмотренным ранее способом (*см. пример построения автомата в § 4.1*), исходя из таблицы переходов счетчика. Однако в данном случае удобно применить искусственный прием [III], дополнив число аргументов функций возбуждений еще одной переменной d, как бы фиктивным добавочным разрядом кода, который переключается под воздействием каждого входного импульса. Функционирование счетчика с учетом переменной d определяется табл. 4.7.

Таблица 4.7

Отсчет	Q_3	Q_2	Q_1	Q_0	D
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	1	0
3	0	0	1	0	1
4	0	1	1	0	0
5	0	1	1	1	1
6	0	1	0	1	0
7	0	1	0	0	1
8	1	1	0	0	0
9	1	1	0	1	1
10	1	1	1	1	0
11	1	1	1	0	1
12	1	0	1	0	0
13	1	0	1	1	1
14	1	0	0	1	0
15	1	0	0	0	1

Наблюдение за изменениями данных в таблице показывает, что разряд Q_0 переключается, когда $d = 0$; разряд Q_1 переключается, когда $d = 1$ и $Q_0 = 1$ (т. е. при условии $dQ = 1$); разряд Q_2 переключается при $d = 1, Q_0 = 0, Q_1 = 1$ (т. е. при условии

$d\bar{Q}_0Q_1 = 1)$; разряд Q_3 переключается при $d = 1, Q_0 = 0, Q_1 = 0, Q_2 = 1$ (т. е. при условии $d\bar{Q}_o\bar{Q}_1Q_2 = 1)$.

Таким образом, выявляется закономерность, согласно которой триггеры разрядов счетчика, работающие в счетном режиме, имеют функции возбуждения вида

$$T_i = d\bar{Q}_0\bar{Q}_1 \ldots \bar{Q}_{i-2}Q_{i-1},$$

где i = 1…(n – 1). Схема четырехразрядного счетчика показана на рис. 4.28.

Схема снабжена преобразователем кода Грея в двоичный код, осуществляемым согласно формулам:

$$b_i = g_{n-1} \oplus g_{n-2} \oplus \ldots \oplus g_i$$

при i < (n – 1) и

$$b_{n-1} = g_{n-1}.$$

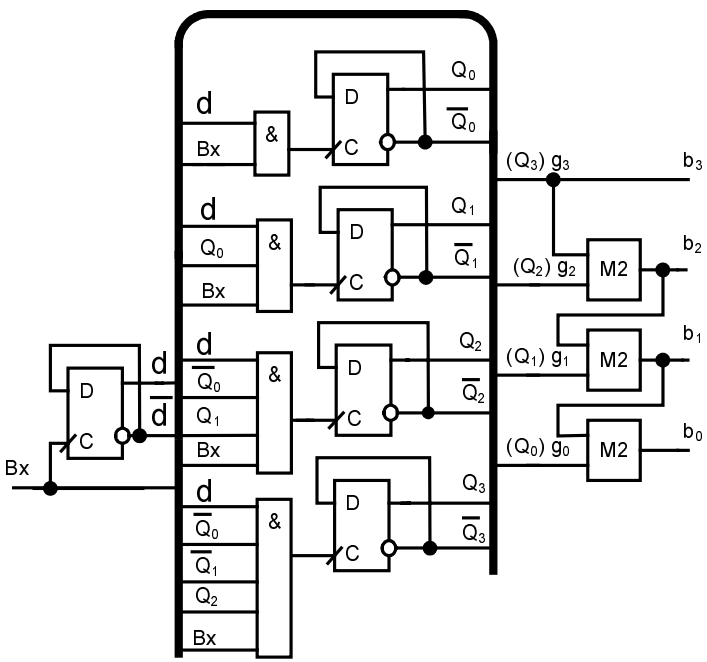


Рис. 4.28. Счетчик в коде Грея с преобразователем кода Грея в двоичный код

Счетчики в коде "1 из N"

Счетчики в коде "1 из N" находят применение в системах тактирования, управления и других. На их основе получают импульсные последовательности с заданными временными диаграммами. Для этого вначале разбивают период временной диаграммы на части ("кванты"), соответствующие минимальному интервалу диа-

граммы, применив для этого задающий генератор с частотой m/T, где m — число "квантов" в периоде диаграммы T. Выходные импульсы задающего генератора затем распределяются во времени и пространстве так, что каждый "квант" появляется в свое время и в своем пространственном канале.

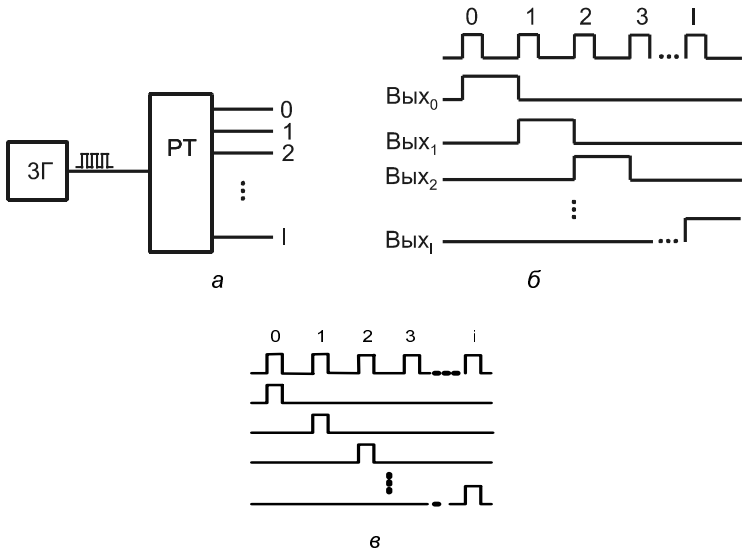


Рис. 4.29. Структура распределителя тактовых сигналов (*а*) и временные диаграммы распределения уровней (*б*) и импульсов (*в*)

Счетчик в коде "1 из N" имеет вход, на который подаются импульсы задающего генератора ЗГ, и N выходов, причем первый импульс генератора передается на первый выход счетчика (в первый канал), второй импульс во второй канал и т. д. Структура такого счетчика, называемого также *распределителем тактов РТ*, и временные диаграммы его работы показаны на рис. 4.29, *а*, *б*, *в*, причем диаграмма на рис. 4.29, *б* соответствует режиму распределения уровней РУ (паузы между активными состояниями каналов отсутствуют), а диаграмма на рис. 4.29, *в* — режиму распределения импульсов РИ. Распределители импульсов обычно реализуются на основе распределителей уровней путем включения в их выходные цепи конъюнкторов, на вторые входы которых подаются импульсы задающего генератора.

Последовательности, вырабатываемые распределителями тактов, применяются как непосредственно, так и для сборки из их "квантов" по схеме ИЛИ импульсных последовательностей с более сложными временными диаграммами.

Счетчики в коде "1 из N" на кольцевых регистрах

Счетчиком в коде "1 из N" (распределителем тактов) является *сдвигающий регистр, замкнутый в кольцо*, если записанное в регистр слово содержит всего одну

единицу. При сдвигах единица перемещается на соседний выход, циркулируя в замкнутом кольце. Число выходов РТ равно разрядности регистра. Недостаток схемы — потеря правильного функционирования при сбое. Если в силу каких-либо причин слово в регистре исказится, то возникшая ошибка станет постоянной. Схема не обладает свойством самозапуска.

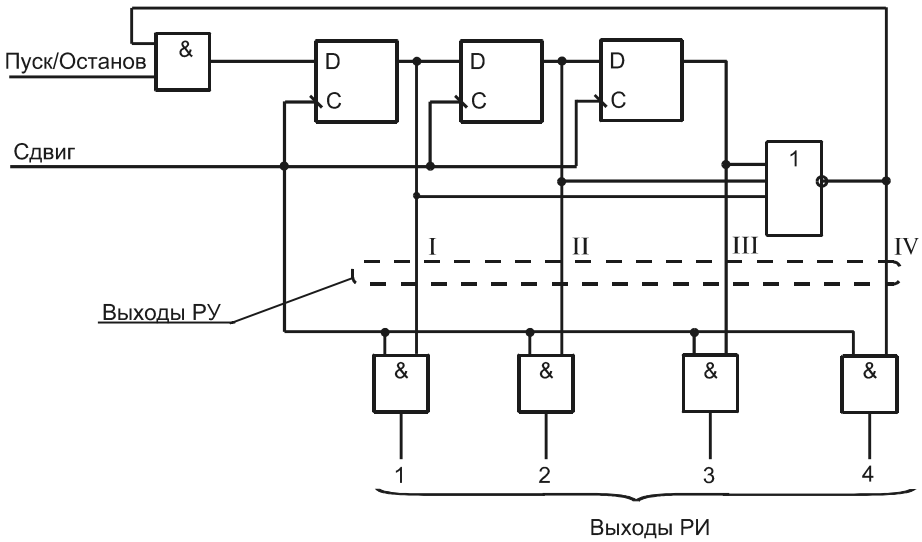


Рис. 4.30. Схема распределителя
с автоматическим вхождением в рабочий цикл

Возможны варианты РТ на кольцевом регистре с *самовосстановлением* режима. Схема такого РТ с самовосстановлением за несколько тактов (рис. 4.30) основана на том, что на вход регистра подаются нули, пока в нем имеется хотя бы одна единица. Таким образом, лишние возникшие единицы будут устранены. Когда регистр очистится, сформируется сигнал записи единицы на его входе. Следовательно, потеря единственной единицы также будет исключена. Выход логического элемента, выполняющего самовосстановление схемы, дает еще один дополнительный канал. На схеме показаны также цепи пуска/останова РТ и два варианта выхода — для распределителя уровней (непосредственно с триггеров и логического элемента ИЛИ-НЕ) и распределителя импульсов (после стробирования сигналов распределителя уровней импульсами сдвига на цепочке конъюнкторов).

Можно поставить задачу более *быстрого исправления сбоев*, в том числе в ближайшем же такте. Для этого нужно задать и реализовать соответствующую диаграмму состояний распределителя. Сделаем это для трехканального распределителя. Диаграмма состояний с указанием рабочего цикла кружками и ложных состояний прямоугольниками приведена на рис. 4.31, *а*. Ей соответствует следующая таблица истинности (табл. 4.8).

Таблица 4.8

Q_1	Q_2	Q_3	Q_{1H}	Q_{2H}	Q_{3H}	Q_1	Q_2	Q_3	Q_{1H}	Q_{2H}	Q_{3H}
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0	1	1	0	0
0	1	0	0	0	1	1	1	0	1	0	0
0	1	1	1	0	0	1	1	1	1	0	0

Выбрав для построения схемы триггеры типа D, учтем, что функция возбуждения этого триггера $D = Q_H$. Исходя из таблицы, для функций $D_i = Q_{iH}$ имеем следующие соотношения:

$$D_3 = \overline{Q}_1\overline{Q}_2\overline{Q}_3,\ D_2 = Q_1\overline{Q}_2\overline{Q}_3 \text{ и } D_1 = Q_3 \vee Q_1Q_2 \vee \overline{Q}_1\overline{Q}_2 .$$

Схема распределителя показана на рис. 4.31, *б*.

Распределители на кольцевых регистрах применяют при малом числе выходных каналов, когда необходимость иметь по триггеру на каждый канал не ведет к чрезмерно большим аппаратурным затратам. Достоинством распределителей на кольцевых регистрах является отсутствие в них дешифраторов и, как следствие, высокое быстродействие (задержка перехода в новое состояние равна времени переключения триггера).

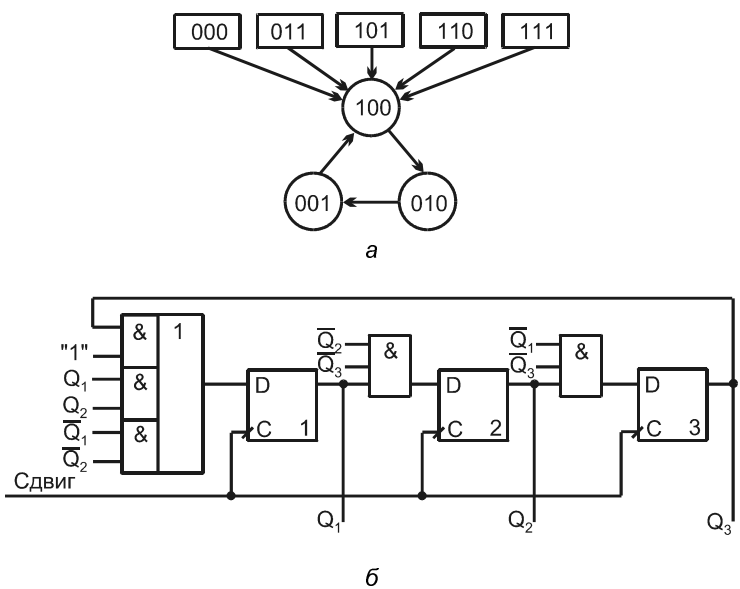
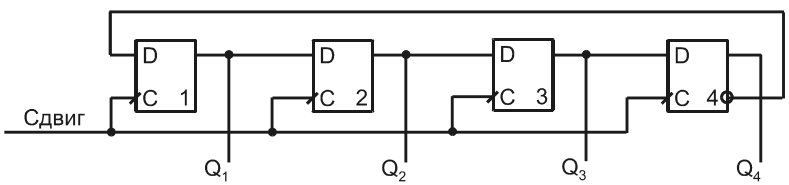


Рис. 4.31. Диаграмма состояний (*а*) и схема (*б*) распределителя с автоматическим вхождением в рабочий цикл за один такт

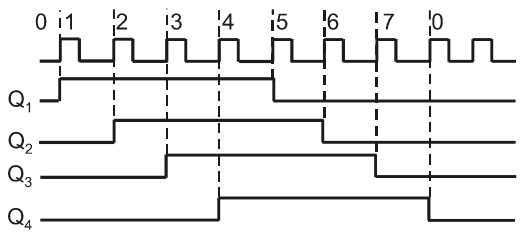
Счетчики в коде "1 из N" на основе счетчиков Джонсона

Счетчик Джонсона. *Кольцевой регистр с перекрестной обратной связью* (счетчик Джонсона, счетчик Мебиуса, счетчик Либау-Крейга) имеет обратную связь на первый триггер от инверсии выходного сигнала (рис. 4.32, *а*). Он имеет 2n состояний, т. е. при той же разрядности вдвое больше, чем обычный кольцевой регистр. В то же время выход счетчика Джонсона представлен не в коде "1 из N", что требует преобразования кодов для получения выходов распределителя тактов. Такие преобразователи очень просты, что обуславливает применение счетчиков Джонсона в составе распределителей.

Четырехразрядный счетчик Джонсона при начальном нулевом состоянии работает следующим образом. Первый импульс сдвига С установит первый триггер в единичное состояние ($\overline{Q}_4 = 1$), в остальных разрядах будут нули как результат сдвига нулей от соседних слева разрядов. Второй импульс сдвига сохраняет единичное состояние первого триггера, т. к. по-прежнему $\overline{Q}_4 = 1$. Второй разряд окажется в единичном состоянии, поскольку примет единицу от первого триггера. Остальные разряды будут нулевыми. Последующие сдвиги приведут к заполнению единицами всех разрядов счетчика, т. е. "волна единиц", распространяясь слева направо, постепенно приведет счетчик в состояние 1111. Следующий импульс сдвига установит первый разряд в ноль, т. к. теперь $\overline{Q}_4 = 0$. Этим начинается процесс распространения "волны нулей". После восьми импульсов повторится состояние 0000, с которого было начато рассмотрение работы счетчика. Временные диаграммы описанных процессов показаны на рис. 4.32, *б*.



а



б

Рис. 4.32. Схема счетчика Джонсона (*а*) и временные диаграммы его работы (*б*)

Особенность рассмотренной схемы — четное число состояний при любом n (2n — всегда число четное). Обычный кольцевой регистр такого ограничения не имеет.

Преобразование кода Джонсона в двоичный код. Преобразование выходного кода счетчика Джонсона в код "1 из N" требует добавления всего одного двухвходового элемента И либо И-НЕ для каждого выхода распределителя тактов. Принцип дешифрации состоит в выявлении положения характерной координаты временной диаграммы — границы между зонами единиц и нулей (табл. 4.9).

Таблица 4.9

Номер состояния	Q_1	Q_2	Q_3	Q_4	Номер состояния	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	4	1	1	1	1
1	1	0	0	0	5	0	1	1	1
2	1	1	0	0	6	0	0	1	1
3	1	1	1	0	7	0	0	0	1

В двух случаях (для слов, состоящих только из нулей или только из единиц) состояние выявляется анализом крайних разрядов. В остальных случаях анализируются разряды на границе зоны единиц и нулей.

Как видно из таблицы, преобразование выходного кода счетчика Джонсона в код "1 из N" осуществляется согласно выражениям:

$$F_0 = \overline{Q}_1\overline{Q}_4,\ F_1 = Q_1\overline{Q}_2,\ F_2 = Q_2\overline{Q}_3,\ F_3 = Q_3\overline{Q}_4,$$

$$F_4 = Q_1Q_4,\ F_5 = \overline{Q}_1Q_2,\ F_6 = \overline{Q}_2Q_3,\ F_7 = \overline{Q}_3Q_4,$$

где F_i (i = 0...7) — выходы распределителя тактов.

По полученным выражениям строится дешифратор. Рассмотрим дешифратор с элементами И-НЕ (с инверсными выходами). В таком дешифраторе можно дополнительно принять меры по предотвращению перекрытий импульсов в соседних каналах, возможных из-за различных задержек элементов. Используя элементы с тремя входами и "косыми связями" (рис. 4.33, *а*), можно запретить начало импульса в последующем канале до его завершения в предыдущем.

Распределитель тактов в целом (рис. 4.33, *б*) имеет выходные сигналы в коде "1 из N".

Снятие ограничения четности числа состояний. Для схем со счетчиками Джонсона могут возникнуть вопросы преодоления ограничения обязательной четности

числа состояний счетчика и обеспечения автоматического вхождения его в рабочий цикл (свойства самозапуска).

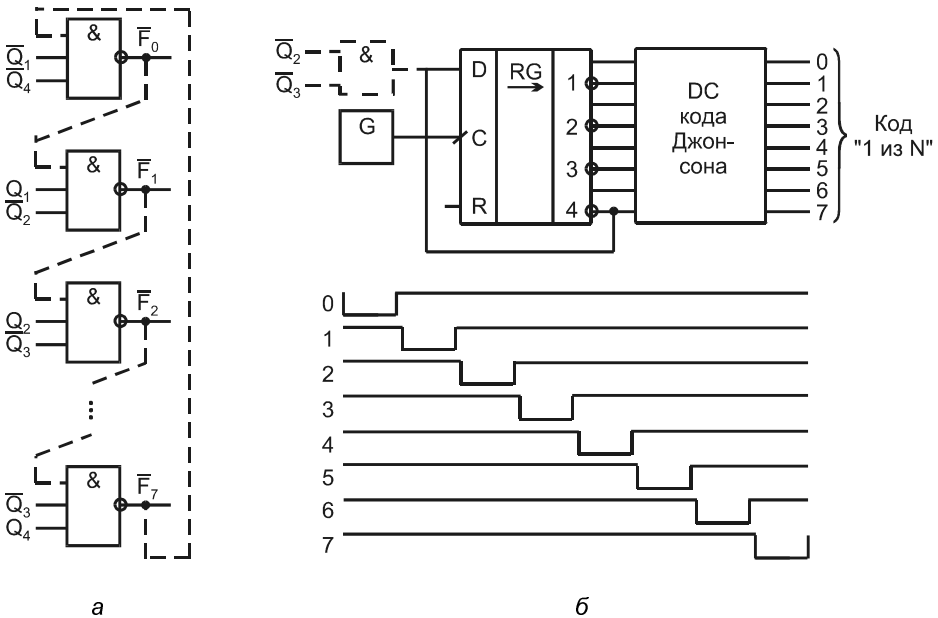


Рис. 4.33. Схемы преобразования кода Джонсона в код "1 из N" (*а*) и распределителя на основе счетчика Джонсона (*б*)

Первую задачу можно решить в рамках подхода, применявшегося при построении счетчиков с произвольным модулем, т. е. исключением одного "лишнего" состояния.

Получить *схему с исключенным состоянием* можно уже не раз показанным способом, переходя от таблицы функционирования к функциям возбуждения триггеров и далее к схеме. Однако в данном случае нетрудно сократить этот путь, пользуясь простыми рассуждениями. Пусть исключению подлежит состояние 11...11. Чтобы его исключить, нужно перейти к следующему состоянию не от состояния "все единицы", а от предыдущего состояния 11...10, которое создает единицу в предпоследнем разряде счетчика, т. е. ноль на инверсном выходе этого разряда. Подавая этот нулевой сигнал на вход счетчика вместе с основным сигналом обратной связи через конъюнктор (показан на рис. 4.33, *б* штриховой линией), исключим состояние 11...11 и получим счетчик с нечетным числом состояний $2n - 1$.

Обеспечение автоматического вхождения в рабочий цикл. Задача *обеспечения вхождения распределителя на основе счетчика Джонсона в рабочий цикл* связана с тем, что базовая схема, рассмотренная выше, свойством самозапуска не обладает. Например, распределитель с трехразрядным счетчиком Джонсона имеет общее число возможных состояний $2^3 = 8$, а число состояний в рабочем цикле $2 \cdot 3 = 6$. Неиспользуемыми являются два состояния: 010 и 101. Нетрудно видеть, что из со-

стояния 010 счетчик перейдет в состояние 101, а из состояния 101 в состояние 010. Таким образом, наряду с замкнутым рабочим циклом существует и замкнутый цикл из двух неиспользуемых состояний, попав в который схема без постороннего воздействия не сможет перейти в рабочий цикл.

Чтобы придать схеме свойство самозапуска, нужно модифицировать сигнал обратной связи, поступающий на вход счетчика. Понятно, что это можно сделать многими путями, поскольку траектория перехода из замкнутого цикла неиспользуемых состояний в рабочий неоднозначна. Одной из возможностей является выработка сигнала обратной связи согласно выражению

$$F_{OC} = D_1 = \overline{Q}_n \vee \overline{Q_{n-1} \ldots Q_2} Q_1.$$

Распределители на основе счетчиков Джонсона характеризуются небольшими аппаратурными затратами (1/2 триггера и один двухвходовой вентиль на канал) и высоким быстродействием (время установления — сумма задержек переключения триггера и вентиля). Счетчики Джонсона реализованы, в частности, в сериях элементов типа КМОП (микросхемы ИЕ9 и ИЕ19 серии К561 и др.), причем одной из причин их применения является отсутствие импульсов помех в выходном напряжении и пониженный уровень токовых импульсов в цепях питания, создаваемых микросхемами. Распределитель в целом реализован в ИС К561ИЕ8.

Следует заметить, что распределители могут быть получены в виде *сочетания обычного двоичного счетчика и дешифратора.* Такое решение наиболее очевидно. При большом числе выходных каналов эта структура может выигрывать у других, но при малом числе каналов преимущество по аппаратурной сложности и быстродействию, как правило, оказывается на стороне вариантов с кольцевыми регистрами или счетчиками Джонсона.

§ 4.6. Полиномиальные счетчики. Делители полиномов

Полиномиальные счетчики (сдвигающие регистры с линейными обратными связями, генераторы псевдослучайных последовательностей, линейные автоматы на основе сдвигающих регистров) используются в устройствах тестового диагностирования цифровых устройств, для решения математических задач методом Монте-Карло, при моделировании систем с учетом случайного разброса их параметров и в некоторых других случаях. Ряд названий, относящихся к полиномиальным счетчикам, связан с понятием линейных комбинационных функций, линейных автоматов и т. п. По определению к линейным переключательным функциям относятся те, для которых полином Жегалкина имеет степень не выше первой. Такие функции содержат конъюнкции единичной длины и могут быть представлены в виде

$$F(x_1, \ldots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \ldots \oplus a_n x_n,$$

где a_i принимают значения 0 или 1 (i = 0...n).

Автомат *линеен*, если схемы выработки функций выходов и функций возбуждения D-триггеров, образующих память автомата, линейны (эти схемы составлены только из сумматоров по модулю 2).

Возможная реализация автономного линейного автомата — сдвигающий регистр с обратными связями через сумматоры по модулю 2. Такие автоматы применяются в генераторах псевдослучайных последовательностей и устройствах тестового контроля ЦУ, где они обеспечивают получение циклических кодов.

Случайные сигналы могут быть аналоговыми или цифровыми. Цифровой сигнал при этом представляется случайной последовательностью, элементами которой могут быть символы 0 и 1 или многоразрядные числа. Первому варианту обычно присваивается наименование случайной последовательности, второму — случайных чисел.

Случайные сигналы характеризуются законами распределения, среди которых важное место занимает равномерный закон, т. к. сигналы с таким законом распределения имеют не только непосредственное применение, но и служат для получения сигналов с другими законами распределения путем определенной математической обработки.

Истинно случайные последовательности и числа, которые нельзя заранее предсказать, генерируются на основе физических явлений (флуктуационных шумов в электронных приборах, радиоактивного излучения и др.), что сложно реализуется и не обеспечивает стабильности статистических характеристик.

При генерации *псевдослучайных последовательностей и чисел* сигнал детерминирован, т. е. заранее известен, но его статистические характеристики близки к характеристикам истинно случайного сигнала. Генерация псевдослучайных сигналов проще и надежнее.

Структура сдвигающего регистра с линейной обратной связью. Эта структура показана на рис. 4.34.

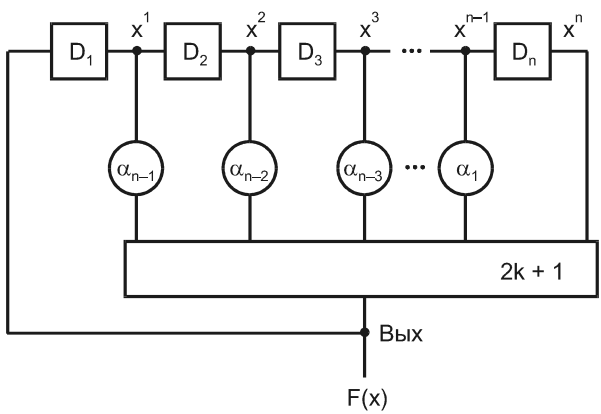


Рис. 4.34. Структура сдвигающего регистра с линейной обратной связью

Выходная последовательность снимается с входа триггера D_1, она же повторяется со сдвигами в других точках тракта, образованного D-триггерами. Аргументами линейной функции являются величины, различающиеся только сдвигами на то или иное число тактов, что отмечается показателем степени при аргументе *x*. Структуре схемы ставится в соответствие полином

$$F(x) = x^n + \alpha_1 x^{n-1} + ... + \alpha_{n-1} x + 1.$$

Начав с любого исходного состояния, можно вычислить последующие. На входе левого триггера окажется значение функции F(x), соответствующее исходному состоянию, в других — результат сдвига на один разряд. Как только опять возникнет состояние, идентичное исходному, все начнет повторяться, т. е. устройство работает периодично. Период последовательности зависит от коэффициентов α_i.

М-последовательности. Обычно желателен максимальный период псевдослучайной последовательности. Автомат с n-триггерами может иметь 2^n состояний. В данном автомате состояние всех нулей должно быть исключено, т. к. из него схема никогда не сможет выйти. Поэтому для данного автомата максимальный период составит $2^n - 1$, а соответствующая ему последовательность называется последовательностью максимальной длины или *М-последовательностью*.

К М-последовательности приводят многие варианты схемы. Их поиск основан на изучении полинома, соответствующего схеме. Чтобы генерировалась М-последовательность, полином должен быть неприводимым и примитивным. Таких полиномов много: при n = 8 их 16, при n = 16 уже 2048 и т. д. Среди множества полиномов целесообразно отыскать такие, у которых минимальное число ненулевых коэффициентов α_i, поскольку это упрощает схему.

Для генерации М-последовательностей схемой с одним элементом сложения по модулю 2 рассчитаны таблицы. Элемент имеет два входа, один из которых подключен к выходу последнего триггера регистра (иначе его наличие в схеме теряет смысл), а второй подключен к разряду с номером i. Если перевести вход элемента с выхода разряда номер i на выход разряда номер n – i, то будет генерироваться последовательность с обратным порядком следования двоичных символов, поэтому в приводимой таблице (табл. 4.10) указаны номера разрядов i или n – i.

Таблица 4.10

n	4	5	6	7	9	10	11	15	17	18	20
i или n–i	1	2	1	1,3	4	3	2	1, 4, 7	3	7	3

Схемы генераторов псевдослучайных последовательностей

Схема генератора псевдослучайных последовательностей (ГПСП), соответствующих первому столбцу таблицы (рис. 4.35, *а*), останавливается сбросом всех триггеров и запускается импульсом старта, записывающим единицу через элемент сло-

жения по модулю 2 в левый триггер. На рис. 4.35, *б* показана схема такого же ГПСП, но обладающего свойством самозапуска. С выхода любого триггера ГПСП можно снять последовательность 111101011001000, соответствующую $M = 2^4 - 1 = 15$. Для схемы ГПСП с 20 разрядами M = 1048575. Если длина последовательности превышает емкость памяти системы, то псевдослучайную последовательность не отличить от случайной.

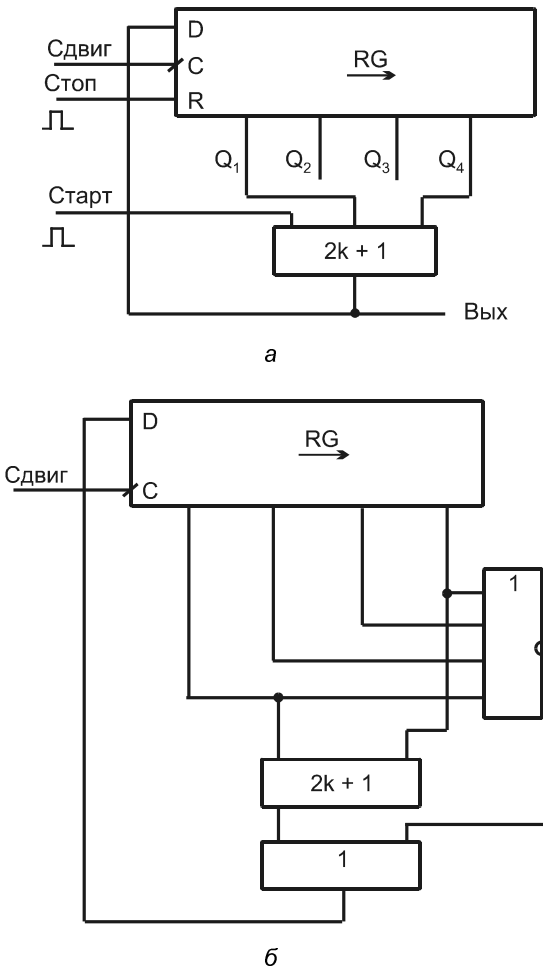


Рис. 4.35. Схемы четырехразрядных генераторов М-последовательностей: запускающегося стартовым импульсом (*а*) и самозапускающегося (*б*)

Генерируемые последовательности имеют число единиц, на единицу превышающее число нулей (следствие исключения состояния "все нули"); группы одинаковых символов появляются в них с той же частотой, что и в случайной последовательности равновероятных двоичных символов; любой набор из $m \le n$ смежных элементов встречается с равной вероятностью (за исключением набора из m ну-

лей); нормированная автокорреляционная функция качественно подобна этой функции белого шума $R(\tau) \approx 0$ при больших M и τ, не кратных M.

Генераторы псевдослучайных чисел (ГПСЧ)

Эти генераторы строят по последовательному, параллельному и смешанному способам. В первом случае число (слово) образуется за несколько тактов. Из образованной в регистре последовательности для получения m-разрядного слова получают результат путем S сдвигов, где $S \geq m$, что дает отсутствие корреляции между соседними словами. Период последовательности слов равен наименьшему общему кратному чисел S и M. Для получения максимального периода число S выбирается взаимнопростым к M.

В генераторах параллельного типа псевдослучайные числа генерируются в каждом такте. Очевидным решением было бы использование m генераторов псевдослучайных двоичных последовательностей для образования отдельных разрядов случайных чисел, однако существуют и более простые решения.

Кодеры и декодеры циклических кодов

Для контроля работы ЦУ и исправления их ошибок широко применяются *циклические коды* (CRC, Cyclic-Redundancy Codes). Они используются для контроля дисковых накопителей, сетей передачи данных и т. д. В дисковых накопителях каждый блок данных (обычно 512 байт) защищают циклическим кодом, что позволяет обнаруживать, а в определенных случаях и исправлять возникшие ошибки. В сетях передачи данных пакеты заканчиваются проверочными блоками циклических кодов. В обоих случаях циклические коды полезны благодаря своей способности *обнаруживать пачки ошибок*, т. к. для дорожек дисков и всплесков помех в сетях наиболее вероятны искажения не одиночных битов, а их групп (это объясняется соотношением дефектной площади дорожки и длительности всплеска помех с площадью одного бита и величиной битового временного интервала соответственно).

Кодеры и декодеры циклических кодов могут иметь различную структуру, один из вариантов показан на рис. 4.36.

Структуре схемы ставится в соответствие полином вида

$$x^m \oplus g_{m-1}x^{m-1} \oplus \ldots \oplus g_1 x \oplus 1,$$

называемый *порождающим*. Порождающий полином должен удовлетворять определенным требованиям (примитивность, неприводимость), от него зависят возможности образуемого циклического кода. Разрядность регистра сдвига соответствует числу контрольных разрядов в формируемых кодовых словах. Если входное информационное слово F(x) имеет разрядность n, а разрядность регистра m, то разрядность кодовой комбинации составит $n + m$.

Информационное слово также представляется полиномом. Например, слово $1011001 = 2^6 + 2^4 + 2^3 + 2^0$ будет представлено полиномом $x^6 \oplus x^4 \oplus x^3 \oplus 1$. Информационное слово последовательно разряд за разрядом подается на вход F(x). После передачи всего слова в регистре формируется остаток от его деления

на порождающий полином. Деление выполняется по правилам арифметики по модулю 2. Полученный *остаток дает контрольные разряды*, которые добавляются к информационному слову и вместе с ним составляют кодовую комбинацию циклического кода.

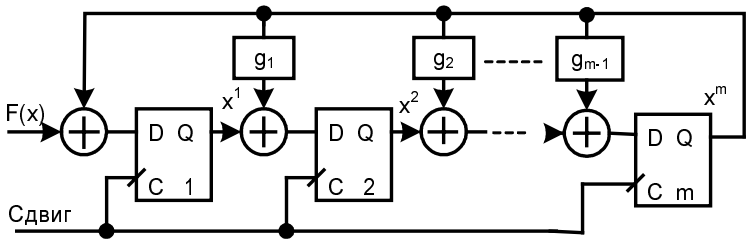


Рис. 4.36. Схема, применяемая при формировании и проверке циклических кодов

Декодирование осуществляется делением кодовой комбинации на порождающий полином. Если комбинация правильна, то остаток от деления будет нулевым. Если произошли ошибки определенного класса, то, как доказано, нулевого остатка быть не может. Имеются возможности локализации обнаруженных ошибок, т. е. и их исправления.

Схемы деления полиномов используются также в *сигнатурных анализаторах*, являющихся средствами тестового диагностирования цифровых устройств, требующих подачи на них специальных воздействий, с помощью которых проверяется правильность работы устройства. С ростом сложности устройств длина тестовых последовательностей и объем оборудования, обеспечивающего генерацию тестов и анализ результатов, увеличиваются, и возникает задача сжатия информации при тестировании. Эта задача решается, в частности, с помощью схем деления полиномов. Тестовая последовательность, снимаемая с выхода испытуемой схемы, поступает на вход схемы деления полиномов, а получаемый при делении остаток сравнивается с его эталонным значением. Равенство полученного и эталонного остатка трактуется как признак исправности испытуемой схемы. Остаток от деления называют *сигнатурой*, а схему деления полиномов — *сигнатурным анализатором*. С помощью специальных процедур наряду с обнаружением ошибок можно производить и их поиск.

Контрольные вопросы и упражнения

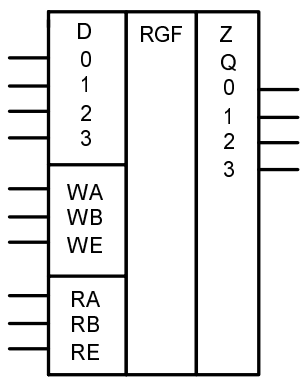
1. Спроектируйте схему тактируемого автомата, последовательность состояний которого имеет вид 000-011-101-110-111-000-… Используйте триггеры типа JK и логические элементы ИЛИ-НЕ. Постройте этот же автомат с применением триггеров типа D и логических элементов И-НЕ. Проанализируйте поведение автоматов при их попадании в нерабочие состояния.

2. Какой признак можно считать основным для классификации регистров? Какие разновидности регистров различают по этому признаку? Указать не менее четырех разновидностей.

3. Составьте схему трехразрядного многофункционального регистра с управляющими входами S_1S_0, определяющими выполняемую операцию, используя триггеры типа D с динамическим управлением и логические элементы, указанные в таблице. Регистр тактируется внешними импульсами C. Реализуются последовательные входы, режим хранения и операции, указанные в таблице.

№ задания	Операции				Логические элементы
	Параллельный прием	Параллельная выдача	Сдвиг влево	Сдвиг вправо	
1	+	–	–	+	И-ИЛИ-НЕ
2	+	–	+	–	Мультиплексор "4-1"
3	–	–	+	+	Мультиплексор "4-1"
4	–	+	–	+	И-НЕ
5	–	+	+	–	ИЛИ-НЕ

4. Как наращивается разрядность слов, хранимых в регистровых файлах? Как можно наращивать количество слов, хранимых в регистровых файлах? Нарисуйте схему для хранения 8 слов с помощью 2 схем регистровых файлов, каждый из которых может хранить 4 слова, и необходимых логических элементов. УГО (условное графическое обозначение) регистрового файла показано на рисунке (WA, WB — разряды адреса записи, WE — разрешение записи, RA, RB — разряды адреса чтения, RE — разрешение чтения).

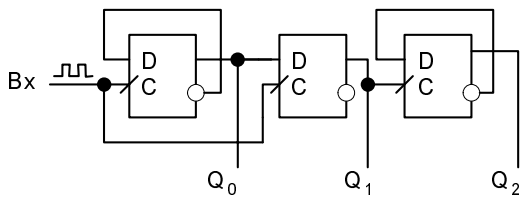


5. Какой параметр счетчиков называют модулем счета?

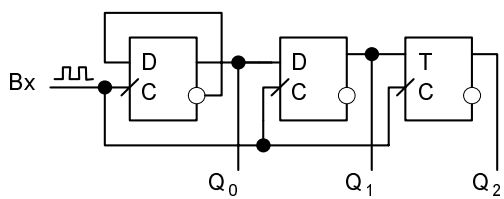
6. Какими параметрами оценивается быстродействие счетчиков в режимах счета и деления частоты? Как соотносятся друг с другом время установления счетчика и период максимальной частоты входного сигнала для счетчиков с последовательным переносом?

7. Какими способами исключаются лишние состояния при построении счетчиков с произвольным модулем на основе двоичных счетчиков? В чем состоят достоинства и недостатки этих способов?

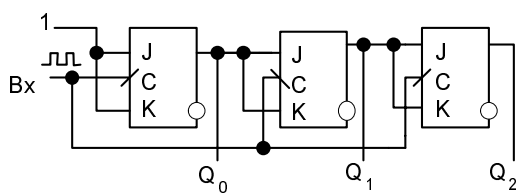
8. Построить временные диаграммы сигналов на выходах, показанных на рисунке схем, и нарисовать для них диаграммы состояний (графы переходов). Начальные состояния триггеров нулевые. Состояния кодируются десятичными эквивалентами двоичных чисел.



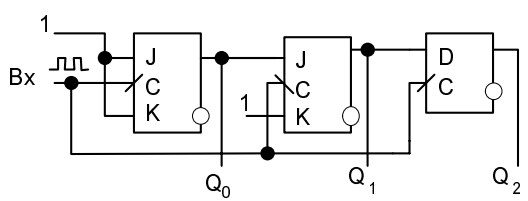
а



б



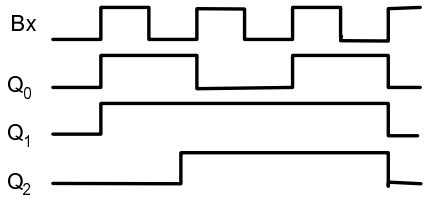
в



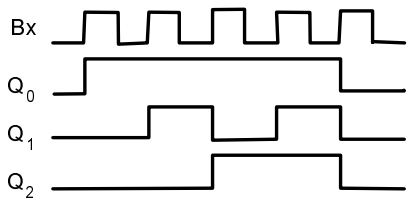
г

9. Спроектируйте схемы, на выходах Q_0, Q_1 и Q_2 которых формируются сигналы, показанные на временных диаграммах (*а*) и (*б*).

Используйте триггеры D и логические элементы И-НЕ.



а



б

10. Построить методом модификации межразрядных связей два варианта трехразрядного счетчика с управляющим входом M, при M = 0 работающего в режиме счетчика с модулем 6, а при M = 1 в режиме счетчика с модулем 5. Для построения схемы использовать элементы:
    - для варианта (*а*) триггеры типа D и логические элементы И-НЕ;
    - для варианта (*б*) триггеры типа JK и логические элементы ИЛИ-НЕ.
11. Для построения двоичных счетчиков имеются триггеры типа JK с задержками переключения 1,5 единицы и логические элементы И-НЕ с задержками 1 единица и числом входов не более 8. Определите минимально возможные интервалы между входными импульсами для работающих в режиме регистрации 32-разрядных счетчиков следующих типов:
    - с последовательным переносом;
    - с параллельным переносом;
    - с групповой структурой при разрядности групп, равной 4, и последовательными переносами в группах и между ними.
12. Сравните между собой двоичные счетчики, счетчики в коде Грея и счетчики Джонсона с точки зрения создания ими помех по цепям питания.
13. Спроектируйте на основе счетчика Джонсона распределитель тактов с семью выходными каналами и самовосстановлением после сбоя, используя триггеры типа JK и логические элементы И-НЕ.
14. Почему максимальное число состояний в схеме генерации псевдослучайных последовательностей, содержащей n-разрядный регистр, равно не 2^n, а $2^n - 1$?

Литература к главе: [3], [9], [10], [26], [33], [36], [40], [46], [48], [49], [51], [52].

Глава 5

Запоминающие устройства

§ 5.1. Основные сведения. Параметры. Классификация

Запоминающие устройства (ЗУ) служат для хранения информации и обмена ею с другими устройствами. Они играют важнейшую роль во многих системах и постоянно совершенствуются по архитектуре, схемотехнике и технологии. В настоящее время созданы и используются сотни различных типов ЗУ.

Важнейшие параметры ЗУ находятся в противоречии. Так, например, большая информационная емкость не сочетается с высоким быстродействием, которое в свою очередь не сочетается ни с большой емкостью, ни с низкой стоимостью. Поэтому системам памяти свойственна *многоступенчатая иерархическая структура*, и в зависимости от роли ЗУ его реализация может быть существенно различной.

В развитой иерархии памяти ЭВМ можно выделить следующие уровни:

- *регистровые ЗУ*, находящиеся в самом процессоре, благодаря которым уменьшается число обращений к другим уровням памяти, реализованным вне процессора и требующим большего времени для обращения к ним;
- *кэш-память*, особо быстродействующая память, служащая для хранения копий информации, используемой в текущих операциях;
- *основная память* (оперативная, постоянная), взаимодействующая с процессором (непосредственно или через кэш) и по возможности согласованная с ним по быстродействию. Исполняемый в текущий момент фрагмент программы обязательно находится в основной памяти;
- *специализированные* виды памяти, характерные для некоторых специфических архитектур (многопортовые, ассоциативные, видеопамять и др.);
- *внешняя память*, хранящая большие объемы информации. Обычно реализуется устройствами с подвижным носителем информации (магнитные и оптические диски, магнитные ленты и др.). В настоящем пособии эти устройства не рассматриваются.

Важнейшие параметры ЗУ

- *Информационная емкость* — максимально возможный объем хранимой информации. Выражается в битах или словах (в частности, в байтах). Бит хранится *запоминающим элементом* (ЗЭ), а слово — *запоминающей ячейкой* (ЗЯ), т. е. группой ЗЭ, к которым возможно лишь одновременное обращение. В вычислительной технике добавление к единице измерения множителя "К" (кило) означает умножение на $2^{10} = 1024$, множителя "М" (мега) — на 2^{20}, а множителя "Г" (гига) — на 2^{30}.
- *Разрядность ЗУ* — разрядность хранимых в нем слов.
- *Организация ЗУ* — произведение числа хранимых слов на их разрядность. При одной и той же информационной емкости организация ЗУ может быть различной, так что организация является самостоятельным важным параметром. Примеры организации памяти: 32×8, $128К \times 8$, $1М \times 1$.
- *Быстродействие ЗУ* оценивают временами считывания, записи, длительностями циклов чтения/записи и т. д.
- *Время считывания* — интервал между моментами появления сигнала чтения и слова на выходе ЗУ.
- *Время записи* — интервал после появления сигнала записи, достаточный для установления ЗЯ в состояние, задаваемое входным словом.
- *Цикл (чтения или записи)* — минимально допустимый интервал между последовательными повторными чтениями или записями. Длительности циклов могут превышать времена чтения или записи, т. к. после этих операций может потребоваться время для восстановления необходимого начального состояния ЗУ.

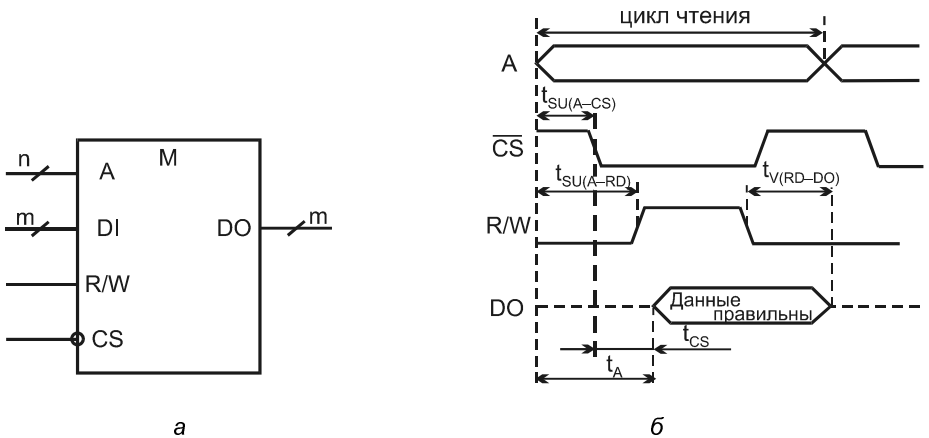
Для многих ЗУ перечисленные параметры дополняются новыми. Причиной является более сложный характер доступа к хранимым данным, когда обращение к первому слову некоторой группы слов (*пакета*) требует большего времени, чем обращение к последующим. Для таких режимов вводят параметры:

- *Время доступа при первом обращении* (*Latency*).
- *Темп передач* для последующих слов пакета. Темп передач в свою очередь оценивается двумя значениями — предельным (внутри пакета) и усредненным (с учетом Latency). С уменьшением пакета усредненный темп снижается, все более отличаясь от предельного.
- Применительно к ЗУ используется также параметр, называемый *полосой пропускания* или *производительностью* и определяемый как произведение числа считываемых (или записываемых) в секунду слов на их разрядность. Например, ЗУ с темпом передачи слов 50 МГц при их разрядности 8 имеет полосу пропускания 400 Мбит/с.

Перечисленные параметры являются *эксплуатационными* (измеряемыми). Кроме них, существует ряд *режимных параметров,* обеспечение которых необходимо для нормального функционирования ЗУ, поскольку для его входных сигналов должно

быть обеспечено определенное взаимное расположение во времени. Для этих сигналов задаются длительности и ограничения по взаимному положению во времени.

Возможный набор сигналов ЗУ для операции чтения показан на рис. 5.1, *а* при следующих обозначениях сигналов:



а *б*

Рис. 5.1. Типичные сигналы простейшего ЗУ (*а*) и их временные диаграммы (*б*)

- A — адрес, номер ячейки, к которой идет обращение. Число ячеек ЗУ (N) выражается целой степенью двойки ($N = 2^n$). Разрядность адреса $n = \log_2 N$. Например, ЗУ с информационной емкостью 64К = 2^{16} слов имеет 16-разрядные адреса, выражаемые словами $A = A_{15}A_{14}A_{13}...A_0$;
- CS — (Chip Select) или CE (Chip Enable), разрешает работу данной микросхемы;
- R/W — (Read/Write) задает выполняемую операцию (при единичном значении — чтение, при нулевом — запись);
- DI и DO (Data Input) и (Data Output) — шины входных и выходных данных, разрядность которых m определяется организацией ЗУ (разрядностью его ячеек). В некоторых ЗУ эти линии объединены. При этом единая двунаправленная шина передачи данных обозначается как DIO (Data Input/Output).

Примечание

Взаимное временное положение двух сигналов (A и B) задается временами предустановки, удержания и сохранения. *Время предустановки* $t_{SU\ (A-B)}$ — интервал между началами сигналов A и B. *Время удержания* $t_{H\ (A-B)}$ — интервал между началом сигнала A и окончанием сигнала B. *Время сохранения* $t_{V\ (A-B)}$ — интервал между окончаниями сигналов A и B. *Длительности сигналов* обозначаются как t_w (индекс от Width — ширина).

При обращении к ЗУ прежде всего подается адрес, чтобы последующие операции не коснулись какой-либо другой ячейки, кроме выбранной. Затем сигналами CS (CE) и R/W разрешается работа ЗУ и определяется операция (чтение или запись),

причем взаимное положение сигналов CS и R/W для разных ЗУ может быть различным. После подачи перечисленных сигналов ЗУ готовит выходные данные (при чтении) или принимает входные (при записи), что требует определенного времени. Затем задний фронт сигнала R/W считывает данные или фиксирует завершение записи. Пример временной диаграммы для рассмотренного набора сигналов ЗУ и операции чтения приведен на рис. 5.1, *б*.

Времена доступа — интервалы от появления управляющего сигнала до получения соответствующего результата согласно стандарту обозначаются индексом A (от Access). Время доступа *относительно адреса* обозначается как $t_{A(A)}$ (часто просто t_A). Время доступа *относительно сигнала CS* — $t_{A(CS)}$ часто обозначается просто как t_{CS}.

Кроме отмеченных параметров для ЗУ используется и ряд других (уровни напряжений, токи, емкости выводов, температурный диапазон и т. д), традиционных для цифровой схемотехники. Исключение составляет *свойство энергонезависимости*, т. е. способность ЗУ сохранять данные при отключении напряжения питания. Энергонезависимость может быть *естественной*, т. е. присущей самим ЗЭ, или *искусственной*, достигаемой введением резервных источников питания, автоматически подключаемых к накопителю ЗУ при снятии основного питания.

Классификация ЗУ

Важнейшие классификационные признаки ЗУ:

- способ доступа к данным;
- тип рабочих режимов (только чтение или чтение и запись);
- схемотехнология (КМОП, n-МОП, ТТЛ(Ш) и т. д.).

Классификация по способу доступа к данным и типу рабочих режимов приведена на рис. 5.2. При *адресном* доступе код на адресном входе указывает ячейку, к которой идет обращение. Все ячейки адресной памяти *равнодоступны*. Адресные ЗУ образуют самую многочисленную группу микросхем памяти. Другие виды памяти часто строят на основе адресных ЗУ.

В *ЗУ с последовательным доступом* записываемые данные образуют некоторую *очередь*. Считывание происходит из очереди слово за словом либо в порядке записи, либо в обратном порядке. Моделью такого ЗУ является последовательная цепочка запоминающих ячеек, в которой данные передаются между соседними ячейками.

Прямой порядок считывания имеет место в *буферах FIFO* с дисциплиной "первый пришел — первый вышел" (First In — First Out), а также в файловых и циклических ЗУ. Разница между памятью FIFO и файловым ЗУ состоит в том, что в FIFO запись в пустой буфер сразу же становится доступной для чтения, т. е. слово поступает в конец цепочки. В файловых ЗУ данные поступают в начало цепочки и появляются на выходе после числа обращений, равного числу элементов в цепочке. При независимости операций считывания и записи фактическое расположение данных в ЗУ

на момент считывания не связано с каким-либо внешним признаком. Поэтому записываемые данные объединяют в блоки, обрамляемые специальными символами конца и начала (файлы). Прием данных из ЗУ начинается после обнаружения приемником символа начала блока.

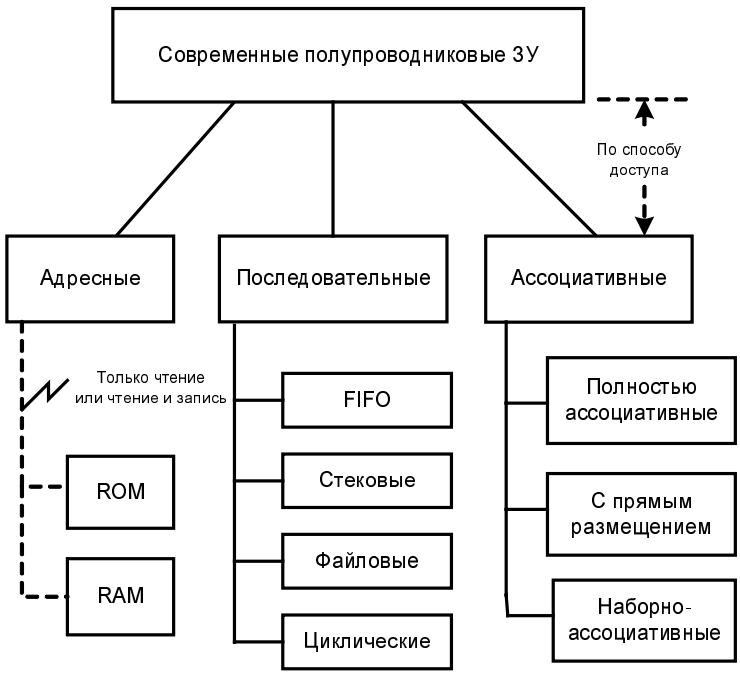


Рис. 5.2. Классификация ЗУ по способу доступа к данным и характеру рабочих режимов

Считывание в обратном порядке свойственно *стековым ЗУ*, для которых реализуется дисциплина "последний пришел — первый вышел". Такие ЗУ называют *буферами LIFO* (Last In — First Out).

В *циклических ЗУ* слова доступны одно за другим с постоянным периодом, определяемым емкостью памяти. К такому типу ЗУ относится *видеопамять* (VRAM).

Время доступа к конкретной единице хранимой информации в последовательных ЗУ представляет собою случайную величину. В наихудшем случае для такого доступа может потребоваться просмотр всего объема хранимых данных.

Ассоциативный доступ реализует поиск информации по некоторому *признаку*, а не по ее расположению в памяти (адресу или месту в очереди). В наиболее полной версии все хранимые в памяти слова одновременно проверяются на соответствие признаку. На выход выдаются слова, удовлетворяющие признаку. Дисциплина выдачи слов и записи новых данных могут быть разными, соответственно чему различают перечисленные в классификации типы ассоциативной памяти. Основная область применения ассоциативной памяти в современных ЭВМ — кэширование данных.

По типу рабочих режимов адресные ЗУ делятся на *RAM (Random Access Memory) и ROM (Read-Only Memory)*. Русские синонимы этих терминов — ОЗУ (оперативные ЗУ) и ПЗУ (постоянные ЗУ). Оба вида ЗУ участвуют в исполнении текущей программы. RAM позволяют и записывать и читать данные в рабочих режимах с приблизительно одинаковой скоростью, но, как правило, не обладают энергонезависимостью. В ROM содержимое либо вообще не изменяется, либо изменяется, но редко и в специальном режиме (с затратами времени, на порядки превышающими длительности рабочих процессов). Для рабочего режима ROM — это *"память только для чтения"*.

ЗУ типа ROM

Постоянная память ROM (рис. 5.3) программируется при изготовлении с помощью масок (ROM (M), масочные ПЗУ) или лазерной технологии (ROM (L)). Для потребителя это в полном смысле слова постоянная память, т. к. изменить ее содержимое он не может.

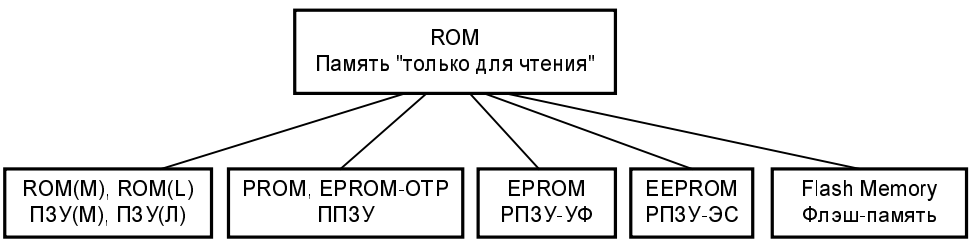


Рис. 5.3. Классификация постоянных ЗУ

В обозначениях следующих четырех разновидностей ROM присутствует буква P (от Programmable). Это *программируемая пользователем* память (ППЗУ — программируемые ПЗУ). В PROM и EPROM-OTP содержимое записывается однократно (OTP означает One-Time Programmable). В EPROM, EEPROM и Flash возможны стирание старой информации и запись новой. Новые данные записываются электрическими сигналами. Содержимое памяти EPROM (Erasable Programmable ROM) стирается облучением кристалла ультрафиолетовыми лучами, ее русское название РПЗУ-УФ. В EEPROM (Electrically Erasable Programmable ROM) и Flash-памяти стирание производится электрическими сигналами. Русское название EEPROM — РПЗУ-ЭС или ЭСППЗУ (электрически стираемое программируемое ПЗУ).

Микросхемы PROM, EPROM и EEPROM программируются в лабораторных условиях на специальных программаторах (PROM, EPROM) либо с помощью специальных режимов без изъятия микросхемы из устройства, в котором она используется. Память *Flash* (флэш) по принципу работы близка к EEPROM, но имеет структурные и технологические особенности, позволяющие выделить ее в отдельный вид.

ЗУ типа RAM

RAM делятся на *статические* (*SRAM* — Static RAM) *и динамические* (*DRAM* — Dynamic RAM). Статические ЗУ, хранящие информацию в триггерах, отличаются высоким быстродействием. Типичные области их применения — кэш-память, буферы FIFO и LIFO, память небольшой емкости для микроконтроллеров, быстродействующих коммуникационных устройств и т. п.

В динамических RAM данные хранятся в виде зарядов конденсаторов, образуемых элементами МОП-структур. Плотность упаковки динамических элементов в несколько раз превышает плотность упаковки статических, поэтому динамические ЗУ имеют значительно более высокую информационную емкость и в 4...5 раз дешевле статических. Однако быстродействие динамических ЗУ ниже, чем у статических. Саморазряд конденсаторов разрушает данные, поэтому они должны каждые несколько миллисекунд *регенерироваться* с помощью специальных контроллеров. Разработаны также динамические ЗУ с внутренними встроенными системами регенерации (*псевдостатические*), у которых внешнее поведение относительно управляющих сигналов становится аналогичным поведению статических ЗУ.

Классификация статических ЗУ

Статические ОЗУ (рис. 5.4) разделены в классификации на *асинхронные* и *синхронные.* Асинхронные ОЗУ названы также *стандартными*, так как ранее они были практически единственными представителями статических микросхем памяти и наиболее привычны для потребителя.

После обращения к *асинхронному ЗУ* до выдачи данных проходит время, являющееся параметром самой памяти, не связанным с тактированием процессора (системы). Отсутствие согласования моментов готовности памяти с тактовыми сигналами процессора может создать дополнительные задержки обмена. Например, возможна выдача данных из ЗУ вблизи, но после активного фронта тактовых импульсов процессора. В таком случае прием данных из памяти будет отложен до следующего активного фронта тактового сигнала, т. е. почти на целый такт.

Асинхронные статические ОЗУ могут различаться требованиями к управляющим сигналам: одни не требуют присутствия *импульсных сигналов*, другие в них нуждаются (обычно это касается сигнала CS). В первом случае соответствующий сигнал может задаваться как импульсами, так и уровнями, т. е. быть неизменным и разрешающим на протяжении многих идущих подряд циклов обращения к памяти. Во втором случае этот сигнал обязательно должен быть импульсным, например, CS в каждом цикле работы памяти должен возвращаться в пассивное состояния и затем переходить в активное, формируя перепад в каждом цикле.

В *синхронных ОЗУ* длительности рабочих этапов жестко связаны с тактовыми сигналами процессора, и это позволяет исключить неоправданные потери времени при обмене, организовать конвейерную обработку данных, а также избавить процессор от простоя в ожидании готовности памяти. Таким образом, синхронность памяти является средством повышения ее быстродействия. Синхронные структуры

является средством повышения ее быстродействия. Синхронные структуры применяются как в статических, так и в динамических микросхемах памяти.

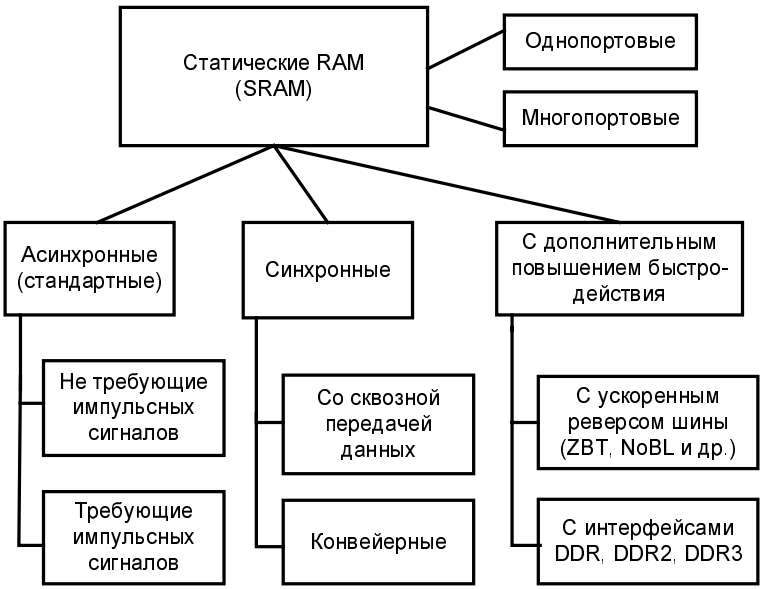


Рис. 5.4. Классификация статических ОЗУ

Среди других методов повышения быстродействия статических ОЗУ можно назвать *ускорение реверса шины* при переходе от передачи данных в одном направлении к другому и *использование интерфейсов DDR*, рассмотренных далее при описании динамических ЗУ повышенного быстродействия.

Статические ОЗУ выполняются как *однопортовые* (обычные) и *многопортовые*. В многопортовых ЗУ возможны одновременные обращения более чем к одной ячейке, например, в простых двухпортовых ЗУ возможно считывание информации из одной ячейки и одновременная запись в другую, а в истинно двухпортовых возможно сочетание любых двух операций. Подобные режимы полезны при разделении памяти между двумя или более абонентами.

Классификация динамических ОЗУ

Динамические ОЗУ (рис. 5.5) характеризуются максимальной информационной емкостью и невысокой стоимостью, поэтому именно они используются как *основная память компьютеров*.

Поскольку желательно получить от основной памяти максимально возможное быстродействие, разработаны многочисленные способы его повышения. Соответствующие архитектуры перечислены в классификации и подробнее рассмотрены в дальнейшем.

Рис. 5.5. Классификация динамических ЗУ

Технико-экономические параметры ЗУ существенно зависят от их схемотехнологической реализации. По этому признаку также возможна классификация ЗУ, однако удобнее рассматривать этот вопрос применительно к отдельным типам памяти.

Классификация перспективных ЗУ

Многие современные средства обработки информации требуют огромных емкостей памяти чрезвычайно высокого быстродействия, что делает актуальным поиск все более эффективных решений. Перспективные варианты новых типов ЗУ перечислены в классификации на рис. 5.6.

Наиболее зрелой, уже достигшей уровня промышленного производства, является *ферроэлектрическая* память (*FRAM*, Ferroelectric RAM), сочетающая высокую емкость и быстродействие с таким полезным свойством, как энергонезависимость. Такое сочетание свойств близко к идеалу, его не имеют ни статические, ни динамические ОЗУ, ни EEPROM, ни Flash-память.

MRAM (Magnetoresistive RAM) — это *магниторезистивные ОЗУ*, в которых запоминающим элементом является участок магнитного материала, способный сохранять состояние намагниченности при отсутствии питания схемы, так что физиче-

ские свойства используемых магнитных материалов придают MRAM естественную энергонезависимость, а также неразрушающее чтение информации. Степень зрелости MRAM ниже, чем у FRAM, но их производство уже начинается.

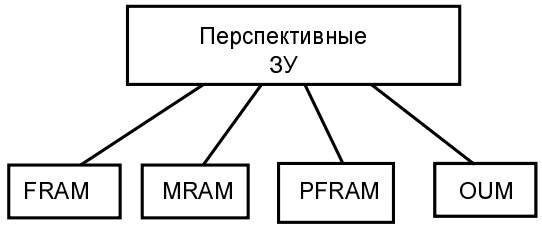


Рис. 5.6. Классификация перспективных ЗУ

Полимерные ферроэлектрические ЗУ (*PFRAM*, Polymeric Ferroelectric RAM), построены на основе материалов, образующих молекулярные цепи с диполями. Диполи служат запоминающими элементами и в зависимости от направления поляризации хранят бит информации. Простота запоминающего элемента и компактность конструкции в целом могут обеспечить PFRAM чрезвычайно высокие емкости при очень малой стоимости/бит. В то же время быстродействие PFRAM мало. Это исключает их применение в качестве ОЗУ, но для использования вместо дисковой памяти PFRAM весьма перспективны.

Память типа OUM (Ovonics Unified Memory) фирмы Ovonics реализована методами интегральной технологии с использованием запоминающих элементов, свойственных компакт-дискам (CD, DVD). Запоминающим элементом служит перемычка из халкогенидного сплава (GeSbTe), который может находиться в проводящем кристаллическом или непроводящем аморфном состоянии. Память считается весьма перспективной с точки зрения экономических показателей.

Наиболее распространенные виды современной памяти: DRAM, SRAM и Flash вместе занимают около 90% всего объема продаж на мировом рынке.

Модули памяти

Для получения ЗУ большой емкости микросхемы памяти объединяются в конструктивные модули, представляющие собою небольшие платы с печатным монтажом, имеющие для подключения к разъемам системной (материнской) платы "ножевой" печатный разъем. Платы модулей многослойные, чаще всего четырехслойные.

Модули *SIMM* (Single In Line Memory Module) с однорядными контактами уже устарели. У модулей *DIMM* (Dual In Line Memory Module) в отличие от модулей SIMM противолежащие контакты на разных сторонах платы электрически не связаны друг с другом, что удваивает число выводов. Микросхемы памяти у этих модулей также размещаются с обеих сторон платы. Имеются и более новые разновидности модулей. Модули подразделяются по напряжению питания и нагру-

зочной способности (буферизованные и не буферизованные). Варианты с большой нагрузочной способностью свойственны модулям памяти большого объема, т. к. в этом случае велики емкостные нагрузки линий передачи сигналов и, соответственно, требуются большие токи для быстрого изменения потенциалов шин при изменении сигналов.

§ 5.2. Основные структуры запоминающих устройств

Структуры адресных ЗУ имеют много общего, что делает рациональным изучение их обобщенных структур с последующим описанием запоминающих элементов, применяемых в различных структурах.

Структура 2D

В структуре 2D (рис. 5.7) запоминающие элементы ЗЭ образуют прямоугольную матрицу размерностью M = k × m, где M — информационная емкость памяти (бит); k — число хранимых слов; m — их разрядность.

Дешифратор адресного кода DC при наличии разрешающего сигнала CS (Chip Select) активизирует соответственно адресу A одну из выходных (словарных) линий, разрешая одновременный доступ ко всем элементам выбранной строки, хранящей адресуемое слово. Элементы столбцов соединены вертикальной линией (разрядной линией, линией записи/считывания) и хранят одноименные биты всех слов. Направление обмена определяется усилителями чтения/записи под воздействием сигнала R/W (Read /Write).

Структура типа 2D применяется лишь в ЗУ малой емкости, т. к. при росте емкости резко проявляются ее недостатки:

- чрезмерно *сложный дешифратор адреса* (число его выходов равно числу хранимых слов);
- *форма матрицы далека от квадратной.* Поскольку обычно число хранимых слов многократно превышает их разрядность, число строк матрицы многократно превышает число ее столбцов, и матрица приобретает вид узкой полоски. Реализация такой матрицы затруднена, и может оказаться совершенно неприемлемой. Квадратная форма матрицы дает большие преимущества.

Структура 3D

Структура 3D позволяет резко упростить дешифраторы адреса и получить матрицу квадратной формы с помощью *двухкоординатной выборки* запоминающих элементов. Принцип двухкоординатной выборки поясним на примере одноразрядной ROM (рис. 5.8, *а*). Здесь код адреса разрядностью n делится на две половины, каж-

дая из которых декодируется отдельно. Выбирается запоминающий элемент, находящийся на пересечении активных линий выходов обоих дешифраторов. Таких пересечений будет как раз $2^{n/2} \times 2^{n/2} = 2^n$.

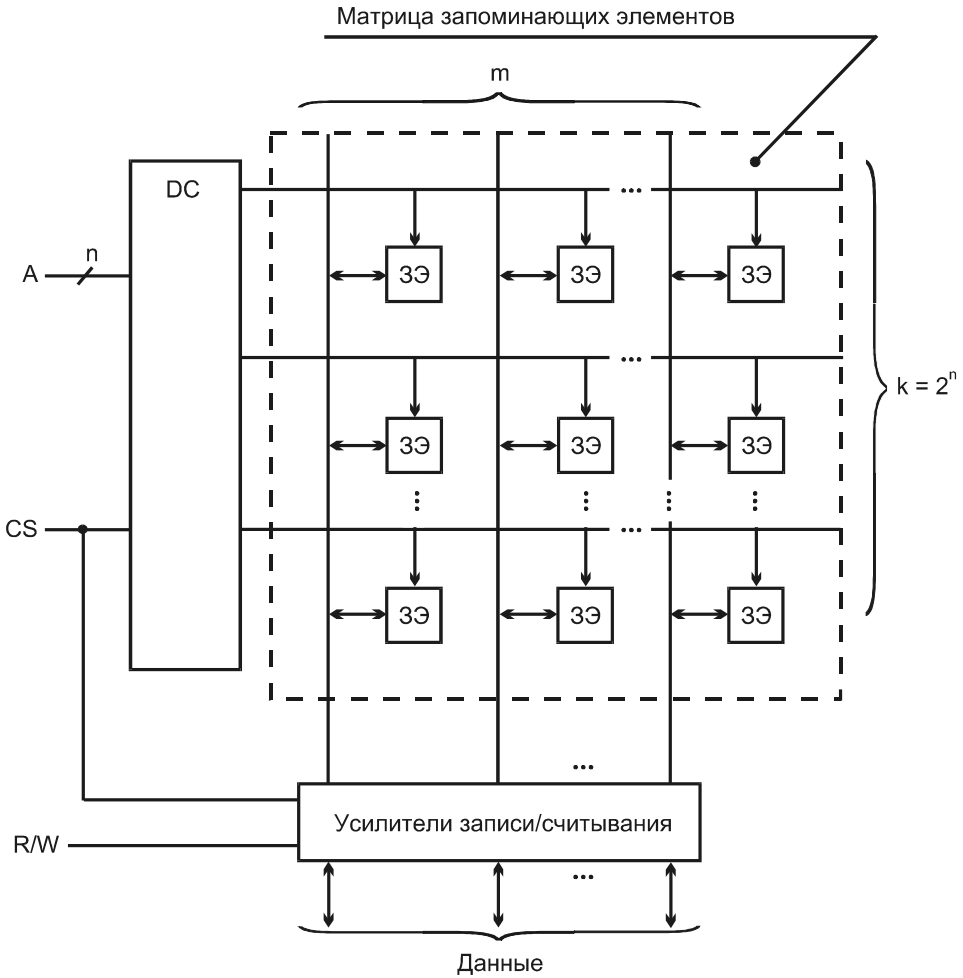


Рис. 5.7. Структура ЗУ типа 2D

Суммарное число выходов обоих дешифраторов составляет $2^{n/2} + 2^{n/2} = 2^{n/2+1}$, что гораздо меньше, чем 2^n.

Пример

Для структуры 2D при хранении 1К слов потребовался бы дешифратор с 1024 выходами, тогда как для структуры типа 3D нужны два дешифратора с 32 выходами каждый.

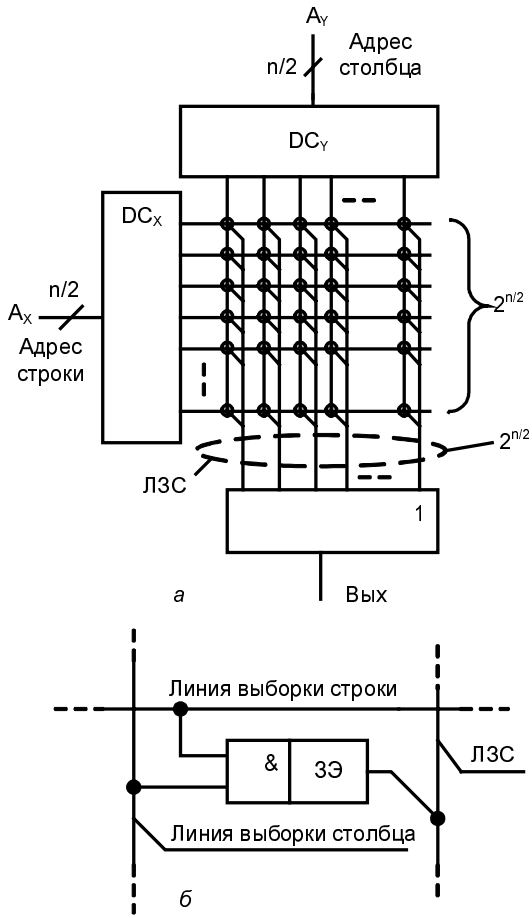


Рис. 5.8. Структура ЗУ типа 3D с одноразрядной организацией (*а*) и запоминающий элемент (*б*)

Недостатком структуры 3D в первую очередь является усложнение элементов памяти, имеющих двухкоординатную выборку. Фактически для выборки ЗЭ возбуждением двух линий к нему нужно добавить двухвходовой элемент И (рис. 5.8, *б*).

Структура 3D может применяться и в ЗУ с многоразрядной организацией (рис. 5.9), приобретая при этом "трехмерный" характер. В этом случае несколько матриц адресуются от одних и тех же дешифраторов, относительно которых они включены параллельно. Каждая матрица выдает один бит адресованного слова, а число матриц равно разрядности хранимых слов.

Структуры типа 3D, как и структуры 2D, имеют ограниченное применение, поскольку сильным конкурентом для них является *структура 2DM* (2D модифицированная).

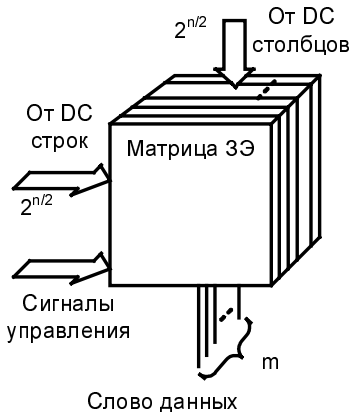


Рис. 5.9. Структура ЗУ типа 3D с многоразрядной организацией

Структура 2DM

В этой структуре сочетаются достоинства обеих рассмотренных структур — в сравнении со структурой 2D упрощается дешифрация адреса и улучшается форма матрицы, а в сравнении со структурой 3D упрощаются запоминающие элементы (не требуются элементы с двухкоординатной выборкой).

В структуре 2DM (рис. 5.10) матрица запоминающих элементов адресуется от дешифратора DC так же, как и в структуре 2D — возбужденный выход дешифратора выбирает целую строку. Однако, в отличие от структуры 2D, длина строки не равна разрядности m хранимых слов, а *многократно ее превышает* (на рисунке в k раз). При удлинении строк их число, естественно, уменьшается и, соответственно, уменьшается число выходов дешифратора, а матрица приближается к квадратной либо может быть сделана квадратной.

Для выбора строки нужна лишь часть адресного кода (разряды A_{n-1}... A_k). Остальные разряды (от A_{k-1} до A_0) при чтении выбирают с помощью мультиплексоров слово из того их множества, которое содержится в длинной строке. Длина строки равна $m2^k$, где m — разрядность хранимых слов. Из каждого "отрезка" строки длиной 2^k мультиплексор выбирает один бит. На выходах мультиплексоров формируется выходное слово. При записи сигнал R/W переключает блоки MUX/DMUX в режим демультиплексоров, через которые разряды входного слова подаются на соответствующие столбцы матрицы и затем записываются в ЗЭ выбранной строки.

Пример

Найдем параметры ЗУ структуры 2DM с организацией 64К × 16 и квадратной матрицей. Число хранимых битов равно 2^{20}, значит, в квадратной матрице будет 2^{10} строк и 2^{10} столбцов. "Длинная строка" будет включать в себя $2^{10}/2^4$, т. е. 64 слова, следовательно, потребуется 16 мультиплексоров размерности 64 × 1. Число вхо-

дов дешифратора строк составит 10, а число входов для адресации мультиплексоров — 6. Общая разрядность адреса равна 16.

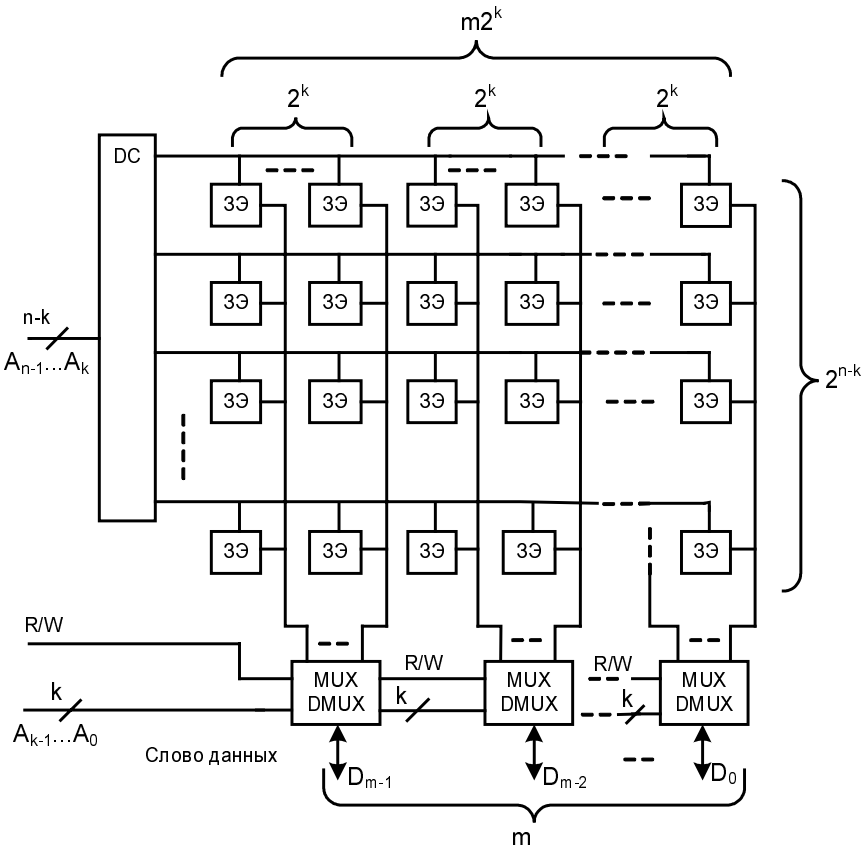


Рис. 5.10. Структура ЗУ типа 2DM

Блочные структуры

С ростом емкости ЗУ увеличиваются размеры матрицы запоминающих элементов и возрастают длины линий выборки и записи/считывания и емкостные нагрузки на них. Соответственно снижается быстродействие памяти. Для противодействия этому единую структуру ЗУ стали фрагментировать (разделять на несколько частей). При этом словарные и разрядные линии укорачиваются, значительно уменьшаются их паразитные емкости и сопротивления и, следовательно, задержки переключений, происходящих в схеме.

Уменьшение длины линии передачи сигнала в k раз снижает время распространения сигнала по линии приблизительно в k^2 раз. В динамических ЗУ укорачивание разрядных линий ведет также к увеличению амплитуды считываемых сигналов, линейно зависящей от отношения емкости запоминающего конденсатора к емкости

разрядной линии, что в свою очередь способствует ускорению работы усилителей считывания. Таким способом удается существенно *повысить быстродействие* ЗУ.

Благодаря разбиению матрицы на части при обращении к ЗУ можно активизировать не всю матрицу, а лишь тот блок, в котором находится искомая ячейка. Это позволяет значительно *снизить мощность,* потребляемую схемой, что особенно важно для памяти большой емкости. Усложнение схемы блочного ЗУ и соответствующее ему увеличение стоимости могут более чем окупаться указанными достоинствами структуры.

На рис. 5.11 сопоставлены структуры единого и блочного ЗУ.

В блочных ЗУ сигналы управления словарными линиями WL вырабатываются несколькими дешифраторами-драйверами, нагрузками для которых служат укороченные линии, а введение дополнительных усилителей чтения-записи УЗС приводит к укорачиванию разрядных линий. Усложнение ЗУ выражается в увеличении числа усилителей чтения-записи и декодеров-драйверов для управления "отрезками" линий.

В блочных структурах адрес обычно делится на три части: две части используются так, как это свойственно структуре 2DM, третья часть — адрес блока. Первые две части выбирают в блоках идентично расположенные ячейки, а третья часть выбирает определенный блок для записи или чтения в ячейку. Адресованный блок активизируется, остальные находятся в режиме уменьшения мощности.

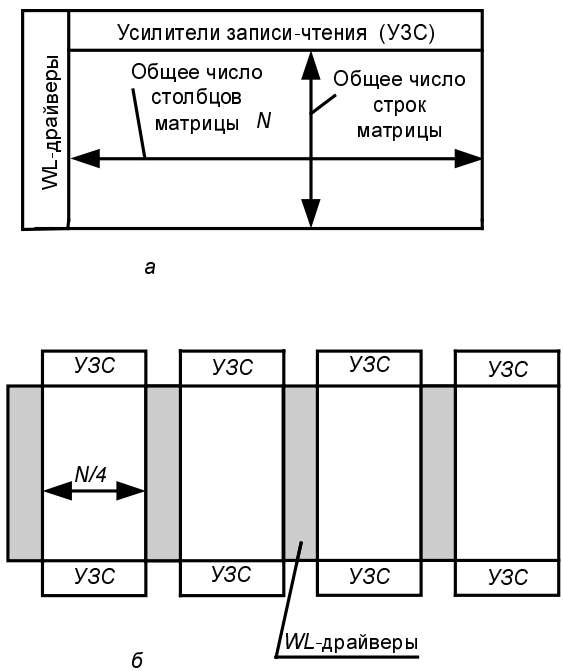


Рис. 5.11. Структуры единого (*а*) и блочного (*б*) ЗУ

Видеопамять

Видеопамять относится к ЗУ с последовательным доступом, в которых данные либо продвигаются в цепочке элементов (по подобию с регистрами сдвига), либо хранятся в адресном ЗУ при последовательном переборе адресов доступа.

Видеопамять работает циклично, на ее выходе последовательно в порядке сканирования экрана монитора появляются коды, задающие параметры светимости (цвет, яркость) элементарных точек экрана — *пикселов* (pixel). Текущее изображение на мониторе — кадр — представлено последовательностью слов, длина которой определяется числом пикселов экрана и синхросимволов строчной и кадровой синхронизации, встроенных в эту последовательность.

Разрядность слова, соответствующего одному пикселу, зависит от требуемого числа возможных состояний пиксела. Если достаточно иметь всего два состояния — светится или не светится — то для воздействия на пиксел нужны всего лишь однобитные слова. Если же речь идет о высококачественном воспроизведении цветных изображений, то используются 24-разрядные слова (по 8 разрядов на каждый из трех цветов, смешение которых дает все разнообразие цветов и оттенков), что соответствует более чем 16 миллионам состояний пиксела.

При реализации на основе адресной памяти циклический доступ к данным обеспечивается счетчиком адреса с модулем, равным числу запоминаемых слов. После каждого обращения адрес увеличивается на единицу, обеспечивая последовательное обращение ко всем ячейкам ЗУ. При переполнении счетчика формируется сигнал начала кадра для запуска кадровой синхронизации.

> ***Примечание***
>
> Для видеопамяти требуется кадровая и строчная синхронизация, но далее для простоты учитывается только кадровая.

Возможна пакетная или одиночная запись. В первом случае переполнение счетчика и его переход на начальный адрес являются сигналом начала передачи блока данных. Во втором случае адрес изменяемой ячейки (номер пиксела) и данные сохраняются в буфере, а в момент совпадения этого адреса и содержимого счетчика выполняется один цикл записи нового слова. Все остальное время ЗУ работает обычным образом.

В видеопамяти элементы хранения и перезаписи данных — регистры кратковременного хранения (рис. 5.12), которые экономично реализуются на основе динамической схемотехники.

При циклическом считывании хранимых данных выбран нижний канал мультиплексора MUX и данные постоянно переписываются с выхода на вход цепочки запоминающих регистров. В последовательность данных вводятся коды кадровых и строчных синхросигналов (на рис. 5.12 для пояснения принципа показан только кадровый). Появление кода синхросигнала на выходе обнаруживается компаратором и синхронизирует запуск развертки монитора.

Пакетная запись начинается после появления запроса передачи в момент прохождения кода кадрового синхросигнала. При этом вырабатывается сигнал разрешения передачи кадра из памяти ЭВМ на вход DI, а мультиплексор переключается на верхний канал. После приема целого кадра счетчик CTR, емкость которого равна длине кадра, переполняется, и под воздействием сигнала переполнения ЗУ возвращается в режим циклической перезаписи.

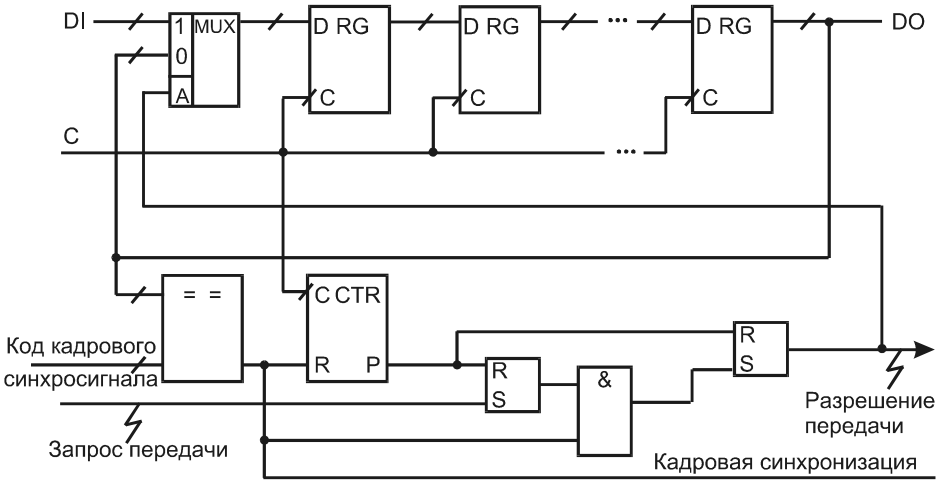


Рис. 5.12. Структура видеопамяти

При одиночных записях устройство должно иметь дополнительно схему сравнения кода счетчика и входного адресного кода (номера заменяемого кода пиксела). При их совпадении мультиплексор переключается на верхний канал на один такт работы, чем обеспечивается замена всего одного слова.

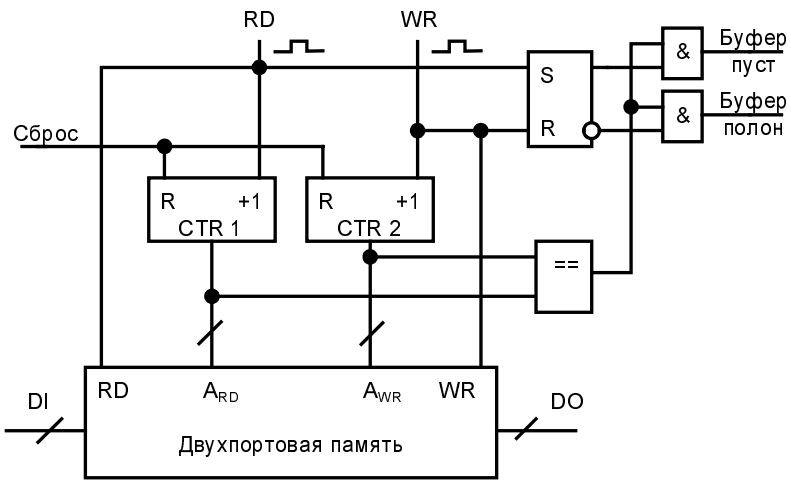
Буферы FIFO, LIFO, круговой

Буфер FIFO. Буфер представляет собою ЗУ для хранения очередей данных (списков) с порядком выборки слов, таким же, что и порядок их поступления. Моменты записи слова в буфер и считывания из него задаются внешними сигналами управления независимо друг от друга.

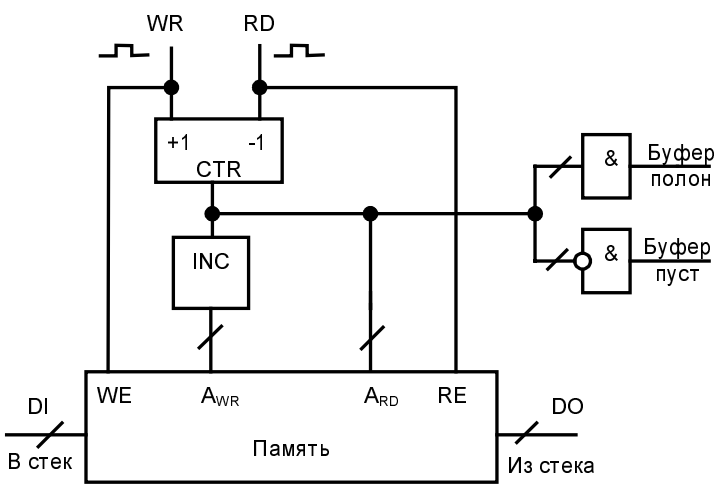
Разный темп приема и выдачи слов необходим, например, если приемник способен принимать данные, поступающие регулярно с некоторой частотой, а источник выдает слова в более быстром темпе и, может быть, к тому же не регулярно. Такие данные поступают в их темпе в буфер FIFO, а из него считываются регулярно с частотой, необходимой для приемника. Новое слово ставится в конец очереди, считывание осуществляется с начала очереди.

Пример буфера FIFO приведен на рис. 5.13, *а*. Перед началом работы оба счетчика адресов CTR 1 и CTR 2 сбрасываются. При записи адреса увеличиваются на еди-

ницу при каждом обращении, т. е. возрастают, начиная с нулевого. То же происходит при чтении слов, так что адрес чтения всегда "гонится" за адресом записи. Если адреса сравняются при чтении, то буфер пуст. Если адреса сравняются при записи, то буфер полон (адресами занята вся емкость счетчика). Эти ситуации отмечаются соответствующими сигналами. Если буфер полон, то нужно прекратить прием данных, а если пуст, то нужно прекратить чтение. Очередь удлиняется или укорачивается в зависимости от разности чисел записанных и считанных слов. Переход через нуль осложнений не вызывает.



а



б

Рис. 5.13. Структуры буферов FIFO (*а*) и LIFO (*б*)

Буфер LIFO. Это ЗУ для хранения очереди данных с порядком считывания, обратным порядку записи (последний пришел — первый вышел). Пример применения — сохранение состояний регистров компьютера при прерывании основной программы и переходе к другой и последующее возвращение данных в те же регистры при возобновлении выполнения основной программы.

Пример буфера LIFO приведен на рис. 5.13, *б*. Адреса слов при обращении к буферу формируются в реверсивном счетчике. При записи адрес увеличивается, а при чтении уменьшается на единицу. Схема INC (Incrementation) увеличивает передаваемое через нее число на единицу. После записей в счетчике находится некоторое число. Чтение будет происходить по тому адресу, который находится в счетчике, а запись по соседнему большему. Отсюда видно, что состояние счетчика указывает вершину стека — последнюю занятую ячейку блока памяти. Состояния "буфер полон" и "буфер пуст" обнаруживаются анализом содержимого счетчика. Код, состоящий из единиц, указывает на то, что буфер полностью заполнен, а код, состоящий из нулей — на то, что буфер пуст. Индикация этих кодов выполняется конъюнкторами с прямыми и инверсными входами.

Другой вариант реализации буфера LIFO — набор реверсивных регистров, число которых равно разрядности слов. При одном направлении сдвига данные вдвигаются в регистры (записываются), при другом выдвигаются (читаются). Первым выдвигается последнее записанное слово.

Круговой буфер. Этот буфер широко применяется при решении задач цифровой обработки сигналов (ЦОС), поскольку выполняет задержку данных, необходимую при реализации многих алгоритмов ЦОС.

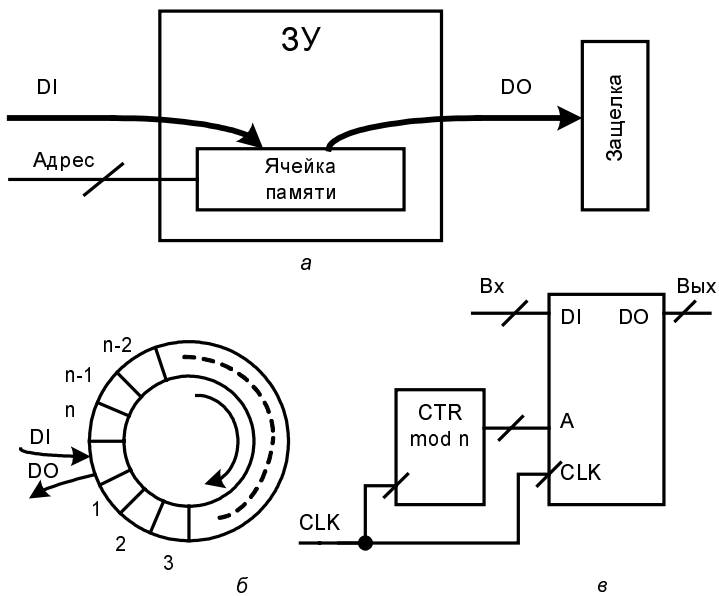


Рис. 5.14. Пояснение к режиму работы памяти Read_First (*а*), иллюстрация к работе кругового буфера (*б*) и его структура (*в*)

Для построения круговых буферов эффективна память, имеющая режим Read_First (сначала чтение), сущность которого поясняется на рис. 5.14, *а*. Такой режим удобно организовать для памяти, имеющей на выходе регистр-защелку. При записи в ячейку нового слова данных старое автоматически передается в защелку и появляется на выходе.

В круговом буфере (рис. 5.14, *б*) слова размещаются в кольце по адресам 0, 1, 2, …n – 2, n – 1, n. Кольцо вращается по стрелке, поворачиваясь на одну позицию в каждом такте. По нулевому адресу записываются входные данные, из этой же ячейки в режиме Read_First считываются выходные, отличающиеся от входных задержкой на n тактов.

В структуру кругового буфера (рис. 5.14, *в*) входят блок памяти и счетчик по модулю n, формирующий адреса чтения. Адреса последовательно изменяются от нулевого до максимального, что имитирует вращение кольца.

Кэш-память

Информация, оперативно используемая при выполнении программ, хранится в основной памяти — динамическом ОЗУ. Кэш-память (Cache) *хранит копии* той части информации, с которой в данное время работает процессор. Она в сравнении с основной памятью имеет небольшую емкость и более высокое быстродействие, позволяющее повысить производительность системы. Кэш-память (или просто кэш) технически представляет собой статическое ОЗУ (SRAM). Статические ОЗУ значительно дороже динамических, так что основную память в целом на них не выполняют.

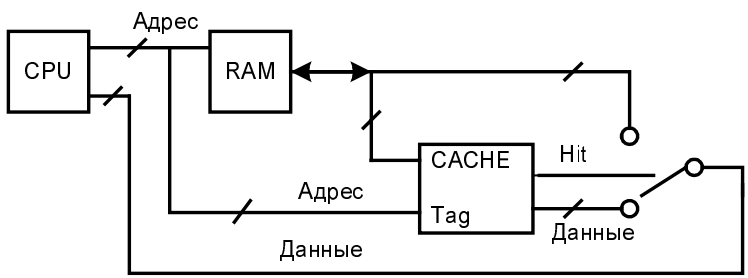


Рис. 5.15. Структура кэшированной памяти

В кэшированной памяти (рис. 5.15) при чтении сначала выполняется обращение к кэшу. Если в кэше имеется копия запрашиваемых данных, то он вырабатывает сигнал *Hit* (попадание) и выдает данные на общую шину данных. В противном случае сигнал *Hit* не вырабатывается и выполняется чтение из основной памяти RAM с одновременным обновлением содержимого кэша путем записи в него нового блока данных.

Эффективность кэширования обуславливается тем, что большинство прикладных программ удовлетворяют принципу *локальности* или, иначе говоря, имеют *гнездо-*

вой характер обращений, при котором адреса последовательных обращений к памяти образуют компактную группу. Поэтому после первого обращения к относительно медленной основной памяти и копирования из нее в кэш блока данных повторные обращения (уже к кэшу) требуют меньше времени. К тому же при работе процессора с кэш-памятью основная память освобождается, и могут выполняться регенерация данных в динамическом ЗУ или другие действия.

Кэш-память *"не видима" для процессора* CPU, который обращается к ней так же, как и к основной памяти (ОП), формируя обычный адрес и управляющие сигналы (в переводе *кэш* означает *скрытый*). Механизмы взаимодействия кэша и ОП реализуются аппаратно самой схемой кэшированной памяти.

С процессором кэш обменивается отдельными словами, а с ОП — блоками данных. Ширина шины связи кэша с процессором равна разрядности передаваемых слов W, а ширина шины связи с ОП обычно составляет величину WK, где K — число слов в блоке. По широким шинам блочные передачи осуществляются быстро.

Объем кэша много меньше емкости ОП, и любая единица информации, помещаемая в кэш, должна сопровождаться дополнительными данными (тегом), определяющими, копией содержания какой ячейки ОП она является. Кэш-память состоит из двух частей — *памяти данных* и *памяти тегов*. При распознавании тегов используется *ассоциативный поиск* (ассоциативные ЗУ).

Модели основной памяти и кэша

Представим ОП и кэш-память в виде (рис. 5.16, *а, б*), где (как и в дальнейшем) для примера численные данные соответствуют емкости ОП 4 Гбайта; емкости кэша 16 Кбайт; размеру блока 16 байт.

> ***Примечание***
>
> Разрядность хранимых слов не оказывает принципиального влияния на рассматриваемые далее структуры, поэтому в численных характеристиках байт может быть заменен на слова других разрядностей.

Основная память при емкости $N = 4Г = 2^{32}$ байт имеет разрядность адреса $n = 32$. Массив ОП разделен на L *блоков*, содержащих по M байт. Если $M = 16 = 2^4$, то $L = 2^{28}$, и адресация блоков требует 28 разрядов.

Кэш-память данных состоит из *строк*, размер которых *идентичен размеру блока ОП*. Число строк кэша C многократно меньше числа блоков ОП (в нашем примере $C = 16$ Кбайт/16 байт $= 1К = 2^{10}$ и для адресации строк требуется 10 разрядов). Для адресации байта в строке требуется 4 разряда. Общая разрядность адреса для памяти данных кэша составит $10 + 4 = 14$, т. е. на 18 разрядов меньше разрядности адресов ОП.

При размещении блоков в кэш-памяти строкам кэша не могут соответствовать определенные блоки ОП, т. к. их число многократно превышает число строк кэша.

Поэтому строкам кэша придается *тег, указывающий, копия какого блока хранится в них в данное время.* Тегом служит часть адреса, поступающего от процессора. Кэш-память *состоит из двух частей* — памяти данных и памяти тегов.

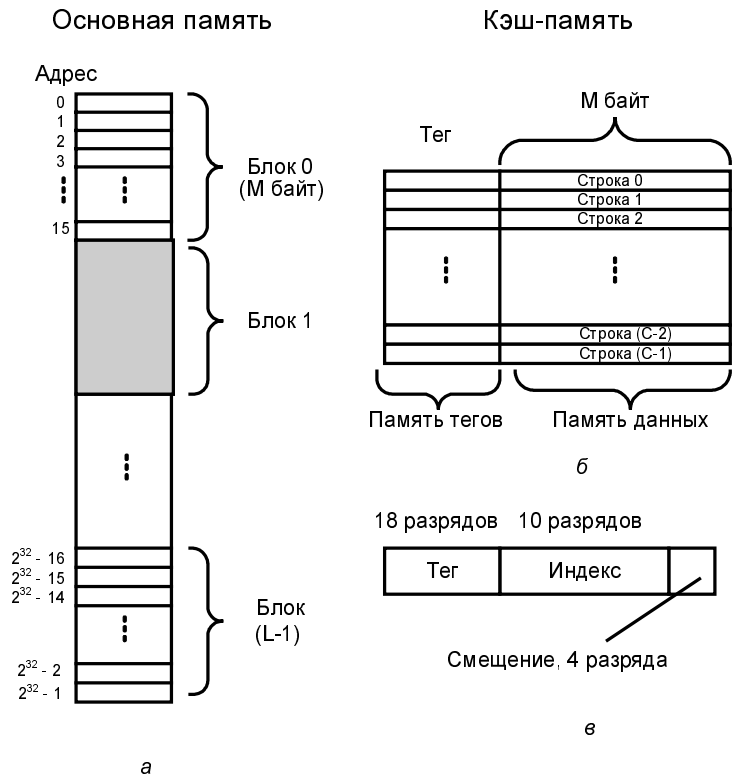


Рис. 5.16. Структуры ОП (*а*), кэш-памяти (*б*) и поля адреса, поступающего от процессора (*в*)

Поля адреса, поступающего от процессора, играют в кэшированной памяти разные роли (рис. 5.16, *в*). Поле адреса слова внутри блока длиной $\log_2 M$ (4 разряда для нашего примера), называемое *смещением*, совпадает с адресом слова в строке кэша, поскольку емкости строки и блока равны и слова расположены в них идентично. Поле длиной $\log_2 C$, называемое *индексом*, адресует строки кэша (10 разрядов для нашего примера). Оставшиеся старшие разряды называются *тегом* (18 разрядов для нашего примера).

Адрес ОП преобразуется в адрес кэша по-разному в зависимости от характера отображения ОП на кэш-память, которое может быть следующим:

- ❒ полностью ассоциативным;
- ❒ прямым;
- ❒ частично-ассоциативным.

Полностью ассоциативная кэш-память

В этой памяти (FACM — Fully Associated Cache Memory) каждая ячейка хранит данные, а в поле "тег" — *полный физический адрес* информации, копия которой записана (рис. 5.17). При любых обменах физический адрес запрашиваемой информации сравнивается с полями "тег" *всех ячеек* и при совпадении их в любой ячейке устанавливается сигнал *Hit*.

При чтении и *Hit* = 1 данные выдаются на шину данных, причем смещение адресует читаемое слово, выбирая его из блока с помощью мультиплексора. Если же совпадений нет (*Hit* = 0), то происходит чтение из основной памяти и одновременно блок данных вместе с адресом помещается в свободную или наиболее давно не используемую ячейку кэш-памяти.

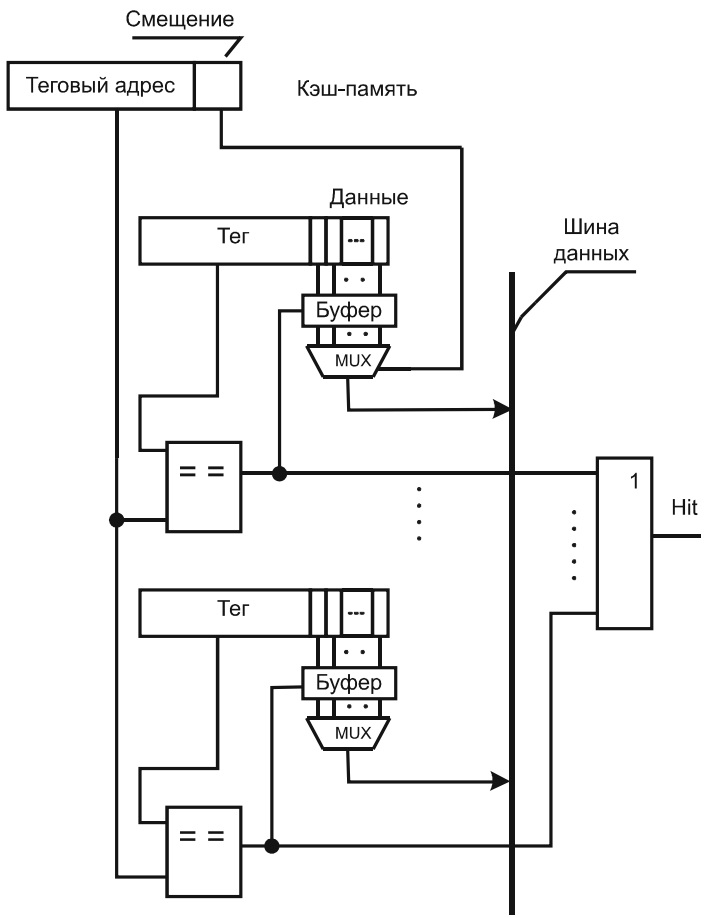


Рис. 5.17. Структура полностью ассоциативной кэш-памяти

При записи данные вместе с адресом сначала, как правило, размещаются в кэш-памяти (в обнаруженную ячейку при *Hit* = 1 и свободную при *Hit* = 0). Копирова-

ние данных в основную память выполняется под управлением специального контроллера, когда нет обращений к памяти.

Память FACM — весьма сложное устройство и используется только при малых емкостях, главным образом в специальных приложениях. В то же время этот вид кэш-памяти обеспечивает наибольшую функциональную гибкость и бесконфликтность адресов, т. к. любую единицу информации можно загрузить в любую ячейку кэш-памяти. Сложность FACM заставляет искать более экономичные структуры, к их числу относятся кэш-память с прямым размещением и кэш-память с частично-ассоциативной архитектурой.

Кэш-память с прямым размещением

В кэш-памяти с прямым размещением (с прямым отображением) несколько блоков основной памяти строго соответствуют одной строке кэша (рис. 5.18), т. е. в данной строке может оказаться лишь один из блоков некоторого заранее определенного подмножества. Подмножество составляют блоки, адреса которых, взятые по модулю, равному числу строк кэша, совпадают с номером строки. Для нашего примера модуль равен 1024. При этом в нулевой строке могут размещаться только блоки с адресами 0, 1024, 2048, 3072 и т. д., в первой строке блоки с номерами 1, 1025, 2049, 3073 и т. д. Так как занимать строку в одно и то же время может только один из этих блоков, нужен специальный признак его распознавания — тег.

Адрес от процессора делится на три части. Младшие разряды (смещение) определяют положение байта (слова) в строке. Средние (индекс) позволяют выбрать одну из строк кэш-памяти. Оставшиеся старшие образуют тег. По индексу производится считывание из кэша данных и тегов. Теговое поле считанной строки сравнивается с теговым адресом и, если есть совпадение, вырабатывается сигнал *Hit* выдачи информации и затем мультиплексированием из строки данных выбирается слово. При загрузке из внешней памяти заменяется вся строка.

Достоинство кэша с прямым размещением — аппаратурная экономичность. В то же время ограничения на расположение блоков в кэше могут не позволить сформировать в нем оптимальный набор страниц, т. к. передача в кэш блока вызывает удаление из него другого, который может быть также нужным.

Кэш-память с частично-ассоциативным отображением

Кэш-память с частично-ассоциативным отображением (с ассоциацией по нескольким направлениям, наборно-ассоциативная) — промежуточный по сложности и эффективности вариант между FACM и кэшэм с прямым размещением. В этом варианте несколько строк кэша объединяются в наборы, а средние разряды адреса определяют уже не одну строку, а набор (рис. 5.19).

Кэш делится на наборы с небольшим числом строк, равным степени двойки, т. е. 2, 4, 8 и т. д. (на рисунке это 2).

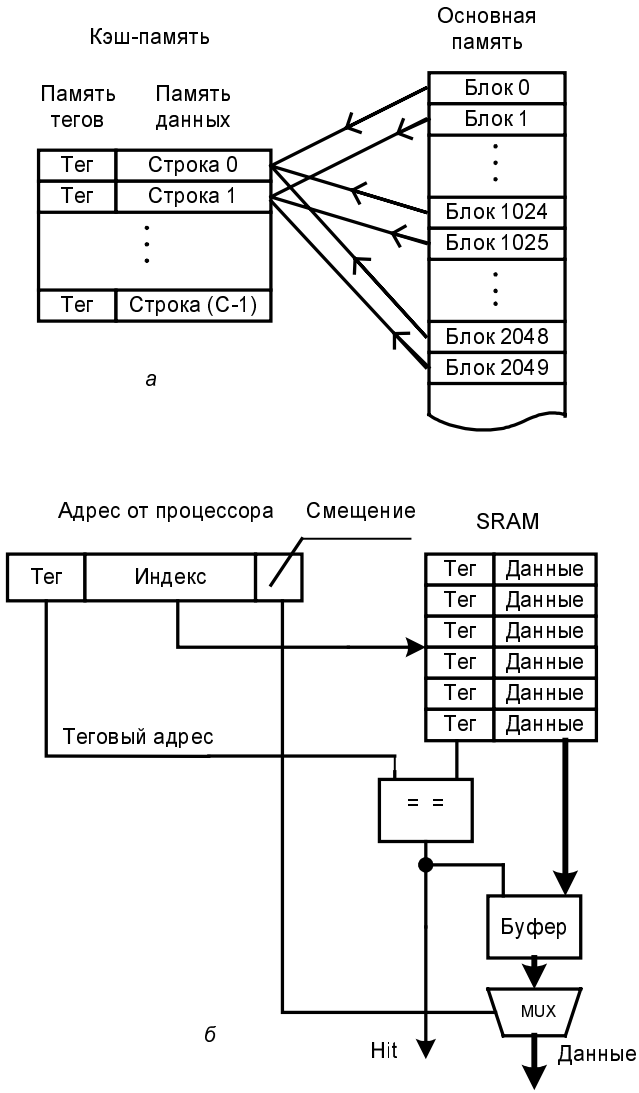


Рис. 5.18. Соответствие адресов блоков и строк (*а*) и организация кэш-памяти с прямым размещением (*б*)

Блок ОП можно поместить только в тот набор, номер которого равен адресу этого блока, взятому по модулю (в данном случае модуль равен 512). Место блока в наборе может быть произвольным. Сравнение тегов со старшими разрядами адреса производится только для строк, входящих в набор. По числу строк в наборе различают двухвходовые, четырехвходовые и т. д. структуры.

Для нашего примера требуются две отдельных SRAM (для четных и нечетных строк). Одновременно выбираются четные и нечетные строки набора (слова в них). Считывание идет от той SRAM, где имеется совпадение тега и тегового адреса. Из

строки через смещение выбирается адресованное слово. При отсутствии совпадений происходит обращение к ОП и замещение строки в одной из SRAM.

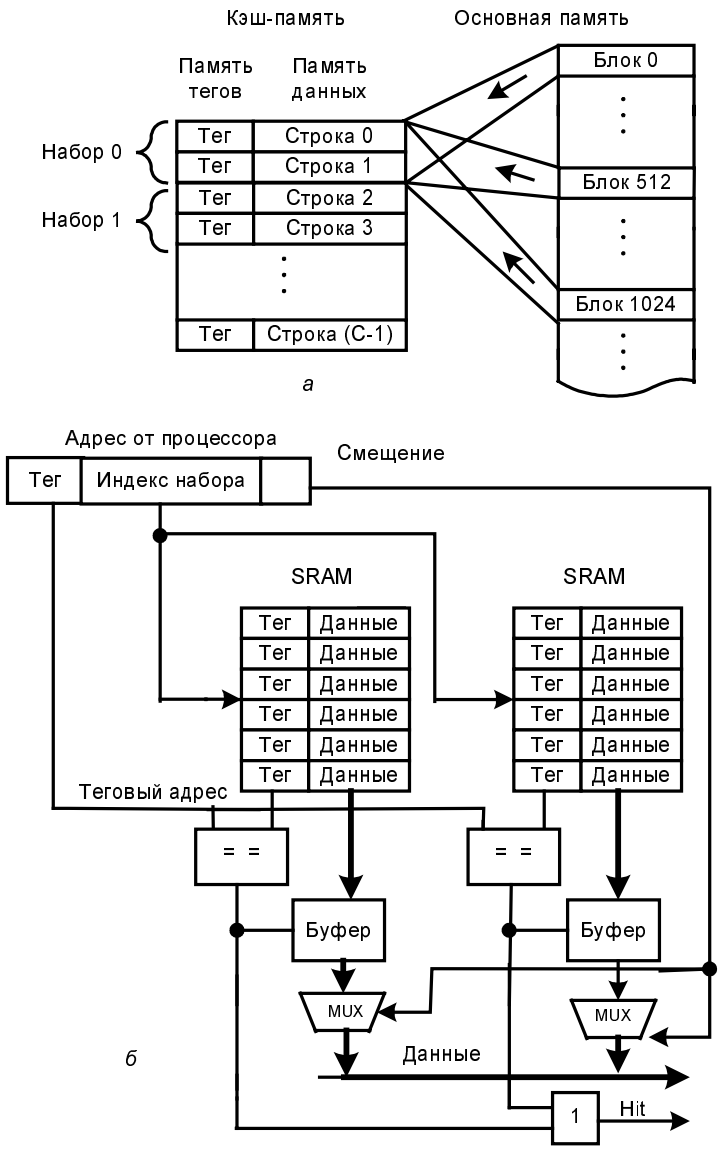


Рис. 5.19. Соответствие адресов блоков и строк (*а*) и организация кэш-памяти наборно-ассоциативного типа (*б*)

По сравнению с кэшем с прямым размещением кэш наборно-ассоциативного типа вследствие более свободного размещения блоков в наборе позволяет сформировать в кэше лучший состав страниц, т. к. имеется возможность выбрать ту или иную заменяемую страницу. В современных микропроцессорных системах кэш первого

уровня, обозначаемый L1, обычно имеет наборно-ассоциативную структуру, а кэш второго уровня L2 (внешний) — структуру с прямым размещением.

При промахе и обращении к ОП в кэш должен быть записан новый блок. При прямом отображении для этого может быть использована только одна определенная строка, которая и будет заменена. Для других вариантов кэша нужен какой-то алгоритм выбора удаляемой строки, т. к. здесь ситуация не однозначная. Самым применяемым является замещение строки, к которой *дольше всего не было обращений*. Подробнее с алгоритмами замещения, как и с другими аспектами работы кэш-памяти, можно ознакомиться по работам [50], [28] и др.

§ 5.3. Структурные методы повышения быстродействия запоминающих устройств

Постоянно растущие потребности вызвали к жизни ряд методов повышения быстродействия микросхем памяти. При этом наряду с технологическим совершенствованием запоминающих элементов и цепей доступа к ним идет и процесс модификации структур ЗУ, и то и другое существенно влияет на быстродействие памяти. Многие *структурные методы* приобрели универсальный характер. Такие методы полезно изучить в обобщенном виде.

Кучность следования адресов. Эффективность многих методов повышения быстродействия ЗУ зависит от характера изменения адресов при обращениях к памяти. При выполнении компьютерных программ чаще всего наблюдается *кучность адресов* — после обращения по некоторому адресу следующее обращение, вероятнее всего, будет по соседнему или близкому к нему адресу (свойство *локальности (гнездования) программ*). Кучность адресов нарушается командами переходов, но их доля среди команд программы невелика.

Произвольный доступ. Для реализации некоторых алгоритмов (сетевые задачи, телекоммуникации и др.) кучность адресов не свойственна. На адрес последующего обращения к памяти предыдущий адрес здесь не влияет. В этом случае будем говорить о работе памяти при *произвольном доступе* (не путать с термином RAM — Random Access Memory).

Быстрый страничный доступ

Этот способ применяется в условиях кучности адресов. При произвольном доступе такой способ совершенно бесполезен. Дело в том, что при кучности адресов обращение по следующему адресу чаще всего изменяет лишь часть разрядов адресного кода (некоторое число младших разрядов). Если обращение идет к близко расположенной ячейке, то старшие разряды адреса не изменятся. Адрес можно рассматривать как состоящий из двух частей — адреса страницы (старшие разряды) и адреса слова на странице (младшие разряды). Если страницей считать строку квад-

ратной матрицы запоминающих ячеек, то адресный код будет разделен на две равные части — адрес страницы (номер строки) и адрес слова в пределах страницы (номер столбца).

Первоначальное обращение к памяти требует обработки обеих частей адреса. Если следующее обращение происходит по адресу на той же странице, то обновляется только часть адреса, и цикл выборки ячейки сокращается. Действительно, неизменность части адресного кода означает, что часть схемы ЗУ останется в прежнем состоянии и не потребуется время на переключение ее элементов.

Быстрый страничный доступ в свое время появился в динамических ОЗУ и позволил повысить их быстродействие приблизительно на 40%.

Пакетная передача данных и команд

Этот способ также эффективен только при кучности адресов обращения к ЗУ. Пакетная передача означает, что при обращении по некоторому адресу из памяти последовательно извлекается не только адресованное слово, но и несколько соседних — пакет. Размер пакета может быть различным, часто он составляет четыре слова. Внутри пакета выборка слов идет быстро, так как адреса формируются специальным счетчиком *внутри самой схемы* (инкрементированием начального адреса), и не требуется для каждого слова передавать в микросхему его страничный адрес, что по сравнению с простым страничным доступом сокращает время выборки слов.

Работа ЗУ со страничным или пакетным доступом характеризуется цепочкой цифр, первая из которых отображает время первоначального доступа к памяти, а последующие — времена доступа к последующим словам в пределах страницы или пакета. Например, для пакетного доступа с размером пакета 4 такая цепочка (Timing) может иметь вид 5-2-2-2 или 5-1-1-1. Это означает, что первоначальный доступ занимает 5 некоторых единиц времени, а последующие 2 или 1.

Технологии DDR и QDR

Термином *DDR* (Double Data Rate) обозначают *интерфейс памяти* с удвоенной скоростью передачи данных. DDR повышает пропускную способность шин. Шины на печатной плате или кристалле имеют ограниченную частоту тактирования. Традиционно, т. е. в технологии SDR (Single Data Rate), активную роль играют перепады тактирующих сигналов только в каком-либо одном направлении — их положительные или отрицательные фронты. Технология DDR предусматривает *выдачу и восприятие данных по обоим фронтам тактирующего сигнала*, а это удваивает скорость передачи при той же тактовой частоте. При этом схемотехника внутри ЗУ может оставаться обычной (с тактированием передач фронтами одного знака), а согласование пропускной способности интерфейса и ЗУ возможно за счет разной разрядности трактов передачи данных вне и внутри ЗУ.

Разработаны также ЗУ с интерфейсом *QDR* (Quad Data Rate), в которых интерфейс DDR сочетается с двухпортовостью. В таких схемах за один такт выполняется четыре операции, поскольку одновременно производятся запись по одному адресу и чтение по другому.

Многобанковые структуры

Многобанковые структуры (Multibank Memory, Interleaving Memory) эффективны при кучности адресов и для таких ЗУ, которые после выполнения операции нуждаются в восстановлении исходного режима, т. е. длительность цикла которых включает в себя время восстановления начального состояния после обращения к ЗУ. Многобанковость позволяет исключать время восстановления из цикла обращения к ЗУ.

В многобанковых структурах память делится на части (банки). Эффект появляется уже при делении памяти на два банка, с увеличением их числа эффективность метода возрастает, хотя и нелинейно. В двухбанковой структуре в одном банке размещаются ячейки с нечетными адресами, в другом — с четными. Если обращение идет по последовательным адресам, то *банки работают поочередно*, и каждый из них имеет между обращениям к нему свободный интервал для восстановления исходного состояния. Поэтому после завершения одной операции можно сразу же приступать к следующей. Если следующий адрес не является соседним по отношению к предыдущему, то возможно повторное обращение к тому же самому банку, а это требует выполнения обычного (не сокращенного) цикла, поскольку придется тратить время на восстановление в банке исходного состояния. Чем больше банков, тем реже будут возникать ситуации с соседними обращениями в тот же банк, т. е. тем выше будет выигрыш в быстродействии памяти. В некоторых ЗУ используется до 32 банков.

Длительное восстановление исходного состояния характерно для динамической памяти, именно в ней находят основное применение многобанковые структуры.

Конвейеризация трактов передачи данных

Сущность *конвейеризации* заключается в разбиении трактов обработки информации на ступени. На рис. 5.20, *а* показана схема, содержащая входной и выходной регистры и логическую схему между ними. Исходя из тезиса о возможности подачи новых входных данных только после окончания обработки старых, получим минимальный период тактовых импульсов для этой схемы:

$$T_{min} = t_{RG} + t_{кц} + t_{SU},$$

где t_{RG} — задержка входного регистра на пути "такт-выход"; $t_{кц}$ — задержка сигнала в комбинационной цепи (логической схеме); t_{SU} — время предустановки выходного регистра.

Уменьшения T_{min}, т. е. повышения частоты тактирования схемы, можно добиться уменьшением задержки $t_{кц}$ путем расщепления логической схемы на ступени, разделенные регистрами (рис. 5.20, *б*). Если логическая схема расщепляется по глубине ровно пополам, то новое значение минимального периода тактовых импульсов определится тем же соотношением, что и для схемы, показанной на рис. 5.20, *а*, однако численное значение задержки логической схемы уменьшится вдвое, т. е. допустимая частота тактирования схемы увеличится. Конвейеризация увеличивает темп передач данных по тракту их обработки, хотя время нахождение каждой единицы данных в этом тракте не только не уменьшается, но даже увеличивается. Первая порция результатов в конвейеризованной системе появляется позднее, чем в сквозной, зато последующие поступают с более высокой частотой.

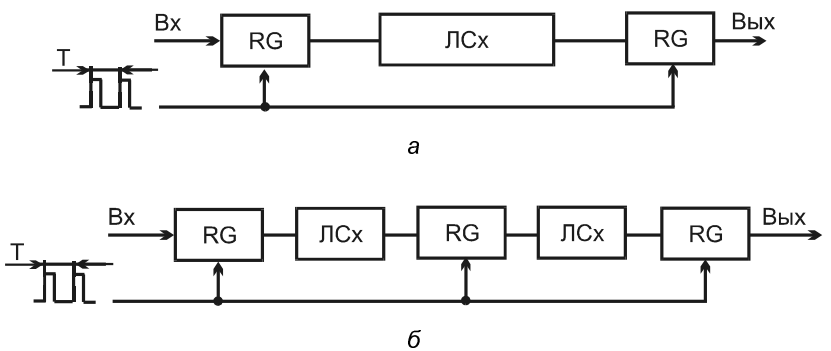


Рис. 5.20. Исходный (*а*) и конвейеризованный (*б*) тракты обработки информации

§ 5.4. Запоминающие устройства ROM, PROM, EPROM, EEPROM

ЗУ перечисленных типов (постоянные) хранят информацию, которая либо вообще не изменяется, либо изменяется редко и в специальном режиме программирования. В рабочем режиме содержимое этих видов памяти остается неизменным. Программирование всех видов постоянной памяти заключается в том или ином *размещении элементов связи* между горизонтальными и вертикальными линиями матрицы запоминающих элементов. Программирование выполняется изготовителем микросхем (в ROM) или их пользователем (в PROM, EPROM, EEPROM).

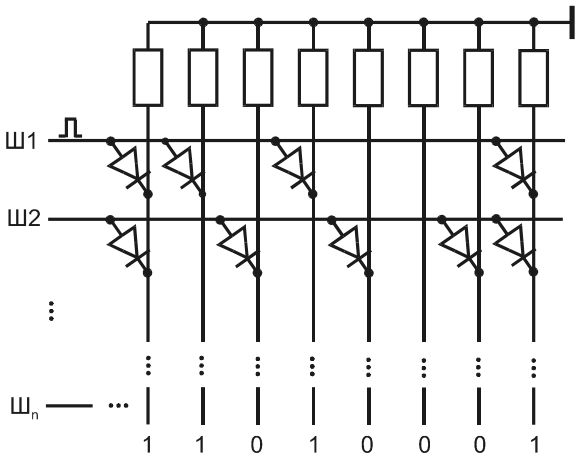
Постоянные ЗУ имеют многоразрядную организацию (обычно 8-, 16- или 32-разрядную) и чаще всего выполняются по структуре 2DM. Технологические типы постоянных ЗУ разнообразны — диодные матрицы, ТТЛ(Ш), КМОП, n-МОП и др.

ROM

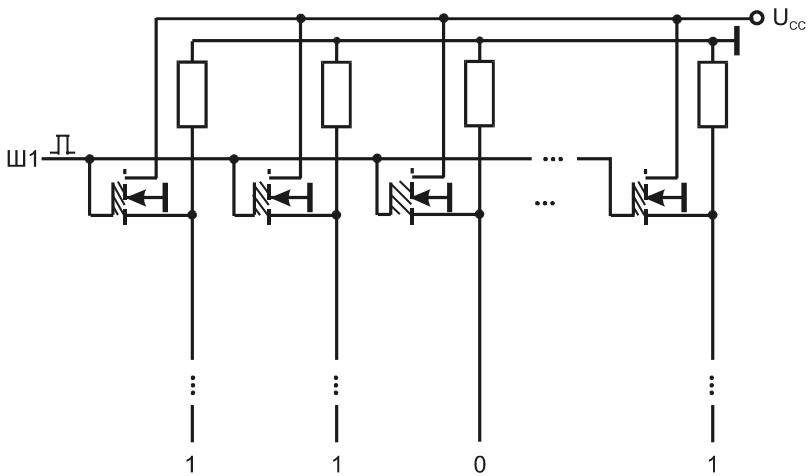
Эти ЗУ (ПЗУ) являются по-настоящему постоянными, т. к. их содержимое после изготовления никогда не изменяется.

Масочные ROM

В масочные ROM (ROM(M)) информация записывается на завершающих этапах технологического процесса.



а



б

Рис. 5.21. Матрицы диодных (*а*) и МОП-транзисторных (*б*) запоминающих элементов ROM

В матрице диодной ROM (рис. 5.21, *а*) горизонтальные линии служат для выборки слов, а вертикальные — для считывания их разрядов. Считываемое слово определяется расположением диодов в пересечениях линий. При наличии диода высокий потенциал выбранной горизонтальной линии передается на соответствующую вер-

тикальную линию, и в данном разряде слова появляется сигнал логической единицы. При отсутствии диода вертикальная линия имеет нулевой потенциал, т. к. она через резистор связана со схемной землей. В изображенной матрице при возбуждении шины выборки Ш1 считывается слово 11010001 (в ячейке номер один хранится это слово). При возбуждении шины Ш2 считывается слово 10101011 (оно хранится в ячейке номер 2). Шины выборки являются выходами дешифратора адреса, каждая адресная комбинация возбуждает свой выход дешифратора, что приводит к считыванию слова из адресуемой ячейки.

Для программирования ROM в одних узлах матрицы элементы связи изготовляются, в других — нет. Чтобы удешевить производство, при изготовлении матрицы стремятся варьировать только один шаблон (маску), так чтобы одни элементы связи были законченными, а другие — не завершенными и как бы отсутствующими. В матрицах с МОП-транзисторами для хранения нуля увеличивают толщину подзатворного окисла, что увеличивает пороговое напряжение транзистора. В этом случае рабочие напряжения ЗУ не в состоянии открыть транзистор, что равноценно его отсутствию. Матрица с МОП-транзисторами показана на рис. 5.21, *б*.

Масочные ЗУ отличаются компактностью запоминающих элементов и, следовательно, высоким уровнем интеграции. При больших объемах производства масочное программирование целесообразно, однако при недостаточной тиражности затраты на проектирование и изготовление шаблонов для технологического программирования ЗУ делают продукцию чрезмерно дорогой. Отсюда видна и область применения масочных ЗУ — хранение стандартной информации, имеющей широкий круг потребителей. В частности, масочные ЗУ имеют в качестве "прошивки" коды букв алфавитов (русского и латинского), таблицы типовых функций (синуса, квадратичной функции и др.), стандартное программное обеспечение и т. п.

Лазерные ROM

Альтернативой масочному программированию ROM служит лазерное. В этом случае матрица вначале содержит все элементы связи, а при программировании отключаются лишние путем пережигания лазером соответствующих проводников. Для лазерного программирования не требуется изготовление шаблона (маски) и при наличии соответствующего технологического оборудования заказ на программирование может быть выполнен быстро (масочное программирование сопровождается изготовлением шаблонов и занимает много времени). Однако стоимость ЗУ в целом оказывается более высокой, чем при масочном программировании, которое в настоящее время является основным вариантом изготовления ROM.

PROM и EPROM-OTP

В PROM и EPROM-OTP информация однократно *записывается потребителем*. Микросхемы PROM программируются пережиганием плавких перемычек (типа *fuse*) с помощью несложных программаторов. В исходной заготовке имеются все

перемычки, а после программирования остаются только необходимые. Металлические или поликремниевые перемычки в электродах запоминающих элементов расплавляются импульсами тока достаточно большой амплитуды и длительности. Программирование пользователем и невысокая стоимость в свое время обусловили широкое распространение PROM, которые сейчас почти сошли со сцены. Отечественные PROM серии К556 имеют емкость 1...64 Кбит и время доступа по адресу 70...90 нс.

Перемычка из двух встречно включенных диодов в исходном состоянии имеет для токов обоих направлений настолько большое сопротивление, что практически равноценна разомкнутой цепи, и запоминающий элемент хранит логический нуль. Для записи единицы к перемычке прикладывают повышенное напряжение, пробивающее диод, включенный в обратном направлении, с образованием в нем короткого замыкания (проводящей перемычки).

Запоминающие элементы с плавкими перемычками в цепях диодов и в цепях эмиттеров биполярных транзисторов (что дает элемент, усиливающий ток), а также перемычки с парами диодов показаны на рис. 5.22 в исходном состоянии и после программирования.

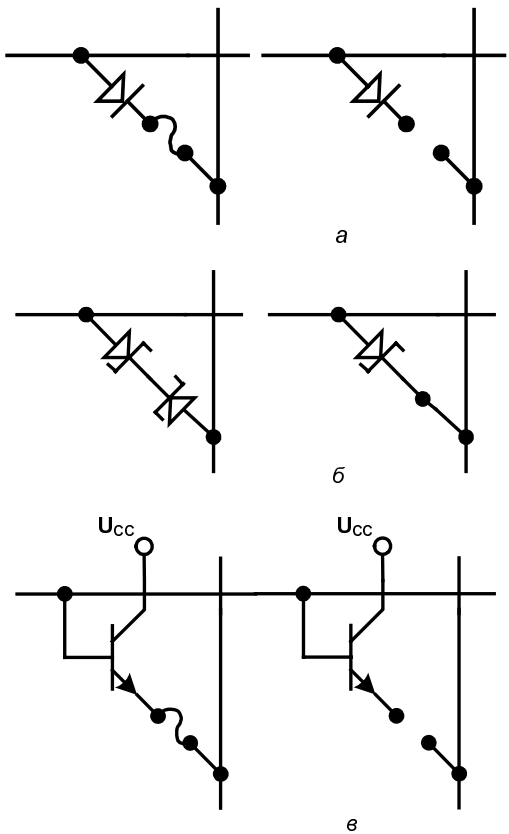


Рис. 5.22. Запоминающие элементы с плавкими перемычками (*а, в*) и диодными парами (*б*)

К микросхемам с однократным программированием, получившим признание в современной схемотехнике, относятся ЗУ типа EPROM-OTP — упрощенный вариант репрограммируемых EPROM, рассматриваемых в следующем разделе.

EPROM и EEPROM

В этих ЗУ возможно стирание старой информации и замена ее новой. Рабочий режим (чтение) — процесс, выполняемый с высокой скоростью. Замена же содержимого памяти — процесс длительный и сложный, для его проведения ЗУ выводится из рабочего режима. В EPROM старая информация стирается ультрафиолетовыми лучами, в EEPROM электрическими сигналами.

Запоминающие элементы обоих типов ЗУ — МОП-транзисторы, над каналами которых созданы области, *способные захватывать и удерживать электрический заряд*, изменяющий пороговые напряжения транзисторов. В транзисторах с *p*-каналом введение в область-ловушку электронов (отрицательного заряда) создает электрическое поле, способствующее возникновению между истоком и стоком проводящего канала. В транзисторах с *n*-каналом заряд электронов, напротив, затрудняет создание проводящего канала и увеличивает пороговое напряжение транзистора, который не сможет включаться под действием рабочего напряжения. Возможность *удалять или создавать заряд* в области-ловушке обеспечивает репрограммируемость ЗУ.

Конструктивно области-ловушки выполняются в виде:

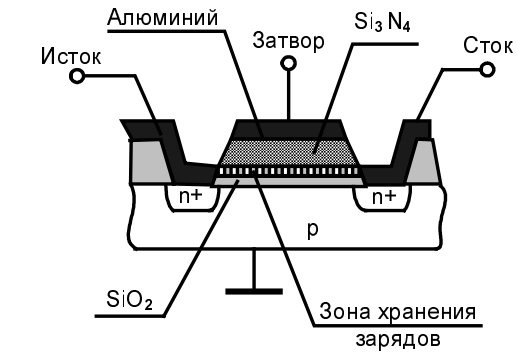
- границы между диэлектрическими слоями из разных материалов;
- плавающих затворов.

МНОП-транзисторы

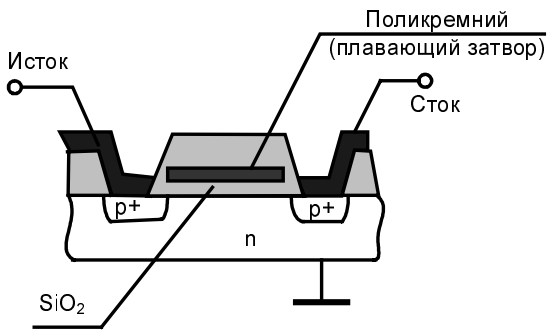
Эти транзисторы (транзистор со структурой "металл-нитрид-оксид-полупроводник" или MNOS (metal-nitride-oxide-semiconductor)) отличаются от обычных двухслойным подзатворным диэлектриком. На поверхности кристалла расположен тонкий слой двуокиси кремния SiO_2, далее более толстый слой нитрида кремния Si_3N_4 и затем уже затвор (рис. 5.23, *а*). На границе диэлектрических слоев SiO_2 и Si_3N_4 возникают центры захвата заряда. Благодаря туннельному эффекту, носители заряда могут проходить через тонкий слой SiO_2 (толщиной не более 5 нм) и скапливаться на границе раздела диэлектрических слоев. Этот заряд или его отсутствие влияют на пороговое напряжение транзистора и отображают информацию (0 или 1), хранимую транзистором. Заряд записывают созданием под затвором напряженности электрического поля, достаточной для возникновения туннельного перехода электронов через слой SiO_2. На границе раздела диэлектрических слоев можно создавать заряд любого знака в зависимости от направленности программирующего поля.

В транзисторе с n-каналом отрицательный заряд на границе слоев повышает пороговое напряжение (экранирует действие положительного напряжения на затворе,

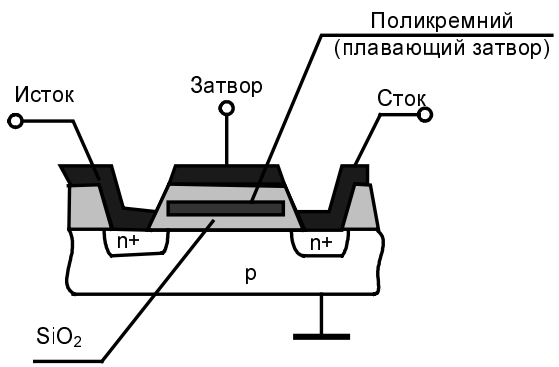
стремящегося открыть транзистор), так что рабочие напряжения на затворе не в состоянии создать в транзисторе проводящий канал. Транзистор, в котором заряд отсутствует или имеет другой знак, открывается рабочим напряжением затвора. Так осуществляется хранение бита: одно из состояний отображает логическую единицу, другое — нуль.



а



б



в

Рис. 5.23. Структуры транзисторов типов МНОП (*а*), ЛИЗМОП (*б*) и с двойным затвором (*в*)

После снятия повышенных программирующих напряжений туннельное прохождение электронов через диэлектрик прекращается, и заданное транзистору состояние остается неизменным.

Перед новой записью старая информация стирается записью нулей во все запоминающие элементы. РПЗУ на МНОП-транзисторах энергонезависимы и могут хранить информацию годами и десятками лет. После $10^4...10^6$ перезаписей МНОП-транзистор перестает устойчиво хранить заряд.

Среди зарубежных ЗУ схемы на МНОП-транзисторах сейчас представлены мало, но, согласно имеющимся данным, они *могут получить новую жизнь* в будущем в связи с применением новых материалов и чрезвычайно малыми размерами ЗЭ при выдающихся характеристиках.

Транзисторы с плавающим затвором

Плавающим затвором называют расположенную над каналом транзистора и отделенную от него тонким слоем диэлектрика проводящую область, в которую можно ввести заряд. Эта область со всех сторон окружена диэлектриком и не имеет контактов с какими-либо электродами. Введенный при программировании заряд сохраняется в плавающем затворе десятки и более лет. Плавающий затвор может быть *единственным* или *вторым*, дополнительным к обычному (управляющему) затвору. Транзисторы с одним плавающим затвором характерны для ЗУ с ультрафиолетовым стиранием зарядов, а транзисторы с двумя затворами — для ЗУ с электрическим стиранием зарядов, играющих в современных разработках основную роль.

В транзисторах с р-каналом без управляющего затвора (рис. 5.23, *б*) введение электронов в плавающий затвор создает проводящий канал, а их удаление ведет к его исчезновению. Ввод заряда инжекцией электронов из области лавинного пробоя стокового *р-n* перехода обусловил название транзисторов — *ЛИЗМОП* (добавление ЛИЗ к обозначению МОП происходит от слов "лавинная инжекция заряда"). Лавинный пробой перехода вызывается приложением к нему повышенного программирующего напряжения.

Транзисторы с двумя затворами

Такие транзисторы (рис. 5.23, *в*) имеют *n*-канал. При подаче на управляющий затвор и сток высоких программирующих напряжений в обратно смещенных *р-n* переходах возникает лавинный пробой, область которого насыщается свободными электронами. Электроны, имеющие энергию, достаточную для преодоления потенциального барьера тонкого слоя диэлектрика, проникают в плавающий затвор (проводящую область из поликремния). Снятие программирующего напряжения прекращает пробой, восстанавливает нормальное состояние областей транзистора и запирает электроны в плавающем затворе. Заряженный электронами плавающий затвор увеличивает пороговое напряжение транзистора настолько, что в диапазоне

рабочих напряжений на затворе проводящий канал в транзисторе не возникает. При отсутствии заряда в плавающем затворе транзистор работает в обычном ключевом режиме под управлением внешнего затвора.

Информация стирается подачей на управляющие затворы низкого (нулевого) напряжения, а на истоки — высокого напряжения программирования.

EPROM

Запоминающие элементы этих ЗУ состоят из двух последовательно включенных транзисторов: обычного ключевого транзистора для выборки адресованного элемента и ЛИЗМОП-транзистора, играющего, в зависимости от программирования, роль замкнутой или разомкнутой перемычки. Для стирания информации ультрафиолетовыми лучами (УФ-лучами) корпус ИС имеет специальное прозрачное окошко. Двуокись кремния и поликремний прозрачны для УФ-лучей, создающих в областях транзистора фото- и тепловые токи, что делает области проводящими и позволяет заряду покинуть плавающий затвор. После стирания старой информации окошко в корпусе заклеивают, чтобы избежать воздействия света на поверхность кристалла. Операция стирания информации этим способом занимает десятки минут, информация стирается сразу во всем кристалле. В схемах с УФ-стиранием *число циклов перепрограммирования существенно ограничено* (10…1000 циклов у приборов разного качества), т. к. под действием УФ-лучей свойства материалов постепенно изменяются.

Современные EPROM имеют емкости до 32 Мбит при временах доступа 70…100 нс и скорости записи 50…100 мкс/слово. Отечественные ЗУ с УФ-стиранием данных, среди которых наиболее известна серия К573, имеют емкости до 1 Мбит и времена доступа около 350 нс.

EPROM-OTP

Для стирания данных УФ-лучами нужны *специальные корпуса с прозрачным окошком*, имеющие высокую стоимость. Замена таких корпусов обычными дешевыми (без окошка) приводит к однократно программируемым EPROM-OTP (OTP от One Time Programmable), в которых запись производится электрическими сигналами путем заряда плавающих затворов. Изменить содержимое памяти уже нельзя, так как отсутствие в корпусе окошка исключает возможность облучения кристалла.

EEPROM

В этих ЗУ стирание информации удалением зарядов из плавающих затворов производится электрическими сигналами. Электрическое стирание имеет преимущества: можно стирать информацию не со всего кристалла, а выборочно (в EEPROM индивидуально для каждого адреса). Длительность процесса "стирание-запись" значительно меньше чем у EPROM, сильно ослабляются ограничения на число циклов пе-

репрограммирования (допускается $10^4...10^5$ таких циклов). Кроме того, перепрограммировать микросхему можно, не извлекая ее из устройства, в котором она работает. В последнее время электрическое стирание активно вытесняет УФ-стирание.

Подключение двухзатворных транзисторов к линиям выборки строк и линиям чтения в матрицах ЗУ показано на рис. 5.24. Логический нуль записывается путем заряда плавающего затвора инжекцией "горячих" электронов в режиме программирования. Стирание информации (удаление заряда из плавающего затвора) приводит к записи в запоминающие элементы логических единиц, т. к. в данном случае опрашиваемые транзисторы открываются и передают напряжение U_{CC} на линии считывания.

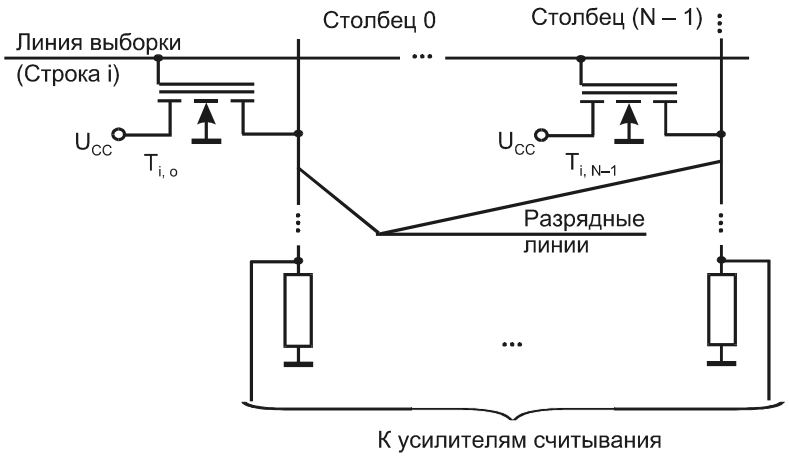


Рис. 5.24. Схема подключения транзисторов с двумя затворами к линиям выборки и считывания в РПЗУ

Среди отечественных ЗУ с электрическим стиранием данных имеются серии КР558 (на основе n-МНОП) и К1609, К1624, К1626 на основе транзисторов с плавающими затворами. Емкости отечественных микросхем памяти с электрическим стиранием данных достигают 64 Кбит, а время доступа составляет около 250 нс. На уровне мировой техники имеются EEPROM с информационной емкостью до 4 Мбит при тактовых частотах до 50 МГц и напряжениях питания 1,8…3,6 В.

Внешняя организация рабочих режимов для микросхем постоянной памяти

Эта организация проста: входными сигналами служат адресный код и сигнал выбора микросхемы CS. Во времени последовательность сигналов следующая: вначале подается адресный код (чтобы произошла дешифрация адреса и было исключено обращение к непредусмотренной ячейке), затем поступает сигнал выбора микросхемы CS

и после задержки, определяемой быстродействием схемы, на выходах данных устанавливаются правильные (действительные) значения считываемых сигналов.

Пример схемы ЗУ типа EPROM

На рис. 5.25 показана упрощенная схема ЗУ EPROM (фирма Cypress Semiconductor).

Ядро схемы — структура 2DM, содержащая адресный дешифратор, матрицу запоминающих элементов и блок мультиплексоров. Ядро дополнено регистром, принимающим данные по фронту синхросигнала CP, выходными буферами с третьим состоянием и логикой управления буферами. Введение в схему регистра вносит в ее работу элементы параллелизма операций и повышает быстродействие ЗУ. Действительно, сразу же после приема данных регистром можно изменять адрес и переходить к чтению следующего слова, не дожидаясь, пока предыдущее слово будет использовано приемником, так как во время выборки следующего слова предыдущее сохраняется регистром. Управление буферами с помощью двух сигналов разрешения E и E_S повышает гибкость использования микросхемы.

Конкретные параметры схемы соответствуют информационной емкости 4 Кбита при организации 512×8. Микросхемы имеют напряжение питания 5 В, время доступа по адресу 25 нс и, как сказано в их описании, программируемость 100%.

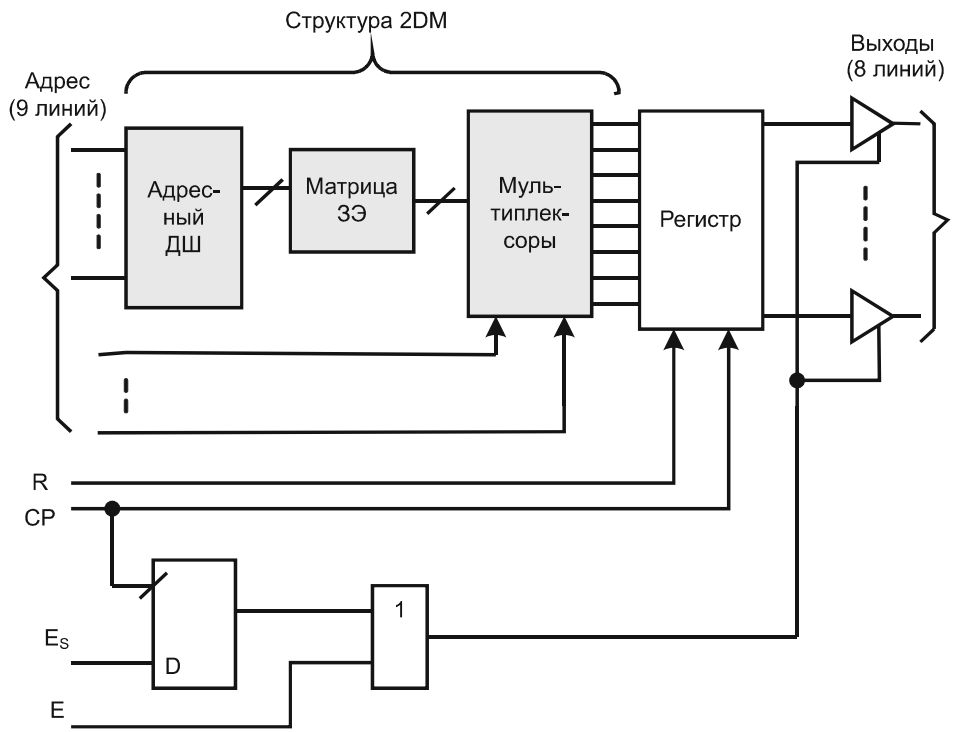


Рис. 5.25. Структура ЗУ типа EPROM

§ 5.5. Флэш-память

Флэш-память (Flash Memory) по типу запоминающих элементов и основным принципам работы сходна с EEPROM, однако имеет структурные и технологические особенности. Для заряда плавающих затворов используется лавинная инжекция, а для стирания — туннелирование электронов через тонкий слой диэлектрика (эффект Нордхайма-Фаули). Разработка флэш-памяти считается кульминацией развития энергонезависимых ЗУ с электрическим стиранием информации.

Блочное стирание. Для флэш-памяти *не предусмотрено стирание отдельных слов*, стирание информации вначале осуществлялось для всей памяти одновременно, позднее — для достаточно больших блоков. Исключение записи/стирания с разрешающей способностью до каждого слова (байта) упрощает схемы ЗУ и способствует повышению их уровня интеграции и быстродействия при снижении стоимости.

Проще всего стирать одновременно всю информацию, но это имеет недостаток — даже замена одного слова требует стирания и новой записи для всего ЗУ в целом. Сейчас схемы со стиранием всего массива данных уступили место схемам *с блочной структурой*, в которых массив памяти делится на блоки, стираемые независимо друг от друга. Вначале емкость блоков составляла 64…256 байт, затем размеры блоков увеличились до 64…128 Кбайт и более.

Число циклов репрограммирования для флэш-памяти хотя и велико, но ограничено. Чтобы увеличить долговечность памяти, используются специальные алгоритмы, способствующие "разравниванию" числа перезаписей по всем ее блокам.

> Термин flash возник в связи с характерной особенностью первых образцов этого вида памяти — одновременным стиранием всего ее объема. Название flash (вспышка, мгновение) подошло для памяти, обладавшей свойством быстрого стирания данных одним сигналом.

Основные разновидности

Соответственно областям применения флэш-память имеет архитектурные и схемотехнические разновидности. Основные направления эффективного использования флэш-памяти:

- ❒ хранение не очень часто изменяемых данных (обновляемых программ и т. п.);
- ❒ замена памяти на жестких магнитных дисках.

В микросхемах первого направления (*несимметричных блочных структурах*) массив памяти делится на неидентичные специализированные блоки. По имени Boot-блоков, в которых информация особо надежно защищена от случайного стирания, эти ЗУ называют также *Boot Block Flash Memory*. Boot-блоки хранят программы инициализации системы, позволяющие ввести ее в рабочее состояние после включения питания.

Микросхемы второго направления (*симметричные блочные структуры*) предназначены для замены жестких магнитных дисков (*Flash-File Memory*), содержат более развитые средства перезаписи информации и имеют идентичные блоки.

Накопители с ячейками ИЛИ-НЕ и И-НЕ

Схемотехника накопителей (матриц запоминающих элементов) флэш-памяти представлена двумя направлениями: на основе ячеек типа ИЛИ-НЕ (NOR) и на основе ячеек типа И-НЕ (NAND). Обе разновидности имеют хорошие перспективы развития.

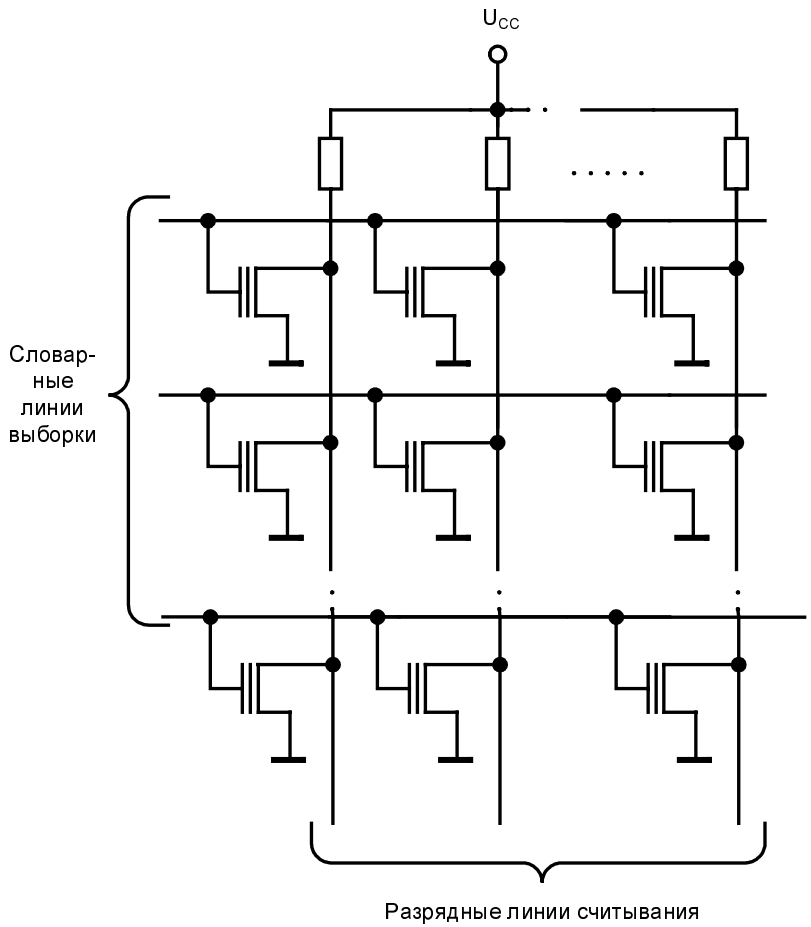


Рис. 5.26. Структура накопителя флэш-памяти на ячейках ИЛИ-НЕ

Накопители на ячейках ИЛИ-НЕ

Этим накопителям свойственна равнодоступность слов *при произвольной выборке*. Они приемлемы для разных применений, особенно удобны для хранения программ.

ЗУ на ячейках ИЛИ-НЕ (NOR) сохраняют полезную преемственность с ЗУ-предшественниками и типичные сигналы управления чтением с произвольной выборкой.

В матрице накопителя (рис. 5.26) каждый столбец представляет собой группу параллельно соединенных транзисторов, образующих ячейку ИЛИ-НЕ. При запертом состоянии всех транзисторов столбца напряжение на разрядной линии имеет высокий уровень. Для получения нулевого уровня достаточно включения хотя бы одного транзистора.

При работе такой ячейки-столбца в матрице накопителя все транзисторы, кроме транзистора выбранной строки, заперты, поскольку на их управляющих затворах действует низкое напряжение. Состояние же транзистора выбранной строки, определяющее и состояние выхода ячейки (напряжения на разрядной линии), зависит от наличия или отсутствия заряда электронов в плавающем затворе. В выбранной строке на затвор транзистора подается высокий уровень напряжения, и если в плавающем затворе заряда электронов нет, и, следовательно, пороговое напряжение транзистора имеет нормальное значение, транзистор откроется. Соответственно возникшему низкому уровню напряжения на разрядной линии в данном разряде прочитается сигнал логического нуля. Транзистор с зарядом электронов в плавающем затворе (с повышенным пороговым напряжением) останется запертым, и в данном разряде напряжение на разрядной линии останется высоким, т. е. будет считываться значение логической единицы. Аналогичные процессы в других разрядах матрицы обеспечивают считывание всей строки.

Накопители на ячейках И-НЕ

Ячейки И-НЕ (NAND) (рис. 5.27) более компактны, чем ячейки ИЛИ-НЕ, что позволяет получать на их основе микросхемы памяти большей емкости. На первый взгляд в обеих ячейках затраты на хранение бита идентичны — это один транзистор с двумя затворами. Однако в ячейке ИЛИ-НЕ каждый транзистор имеет контакты заземления и соединения с разрядной шиной, а в ячейке И-НЕ таких контактов нет, и транзистор как запоминающий элемент выступает "в чистом виде". Более того, при последовательном включении транзисторов сток одного и исток соседнего могут быть совмещены, поскольку они представляют собой электрически соединенные области одного типа проводимости. В результате *ячейки И-НЕ приблизительно вдвое компактнее ячеек ИЛИ-НЕ*.

Для считывания состояния адресованного транзистора необходимо все остальные держать в открытом состоянии. Тогда напряжение на разрядной линии будет определяться состоянием единственного адресованного транзистора — если он имеет проводящий канал, то выходное напряжение ячейки будет низким, в противном случае оно окажется высоким.

Не входя в подробности, отметим, что память на основе ячеек И-НЕ *не обеспечивает произвольный доступ* и используются там, где считывание хранимых данных имеет характер потока, например, для замены магнитных дисков, в видеопамяти

и т. п. Последовательный характер доступа к данным в памяти с ячейками И-НЕ приводит к задержке чтения первого слова массива на время порядка 1 мкс при задержках 60…80 нс для чтения последующих слов массива.

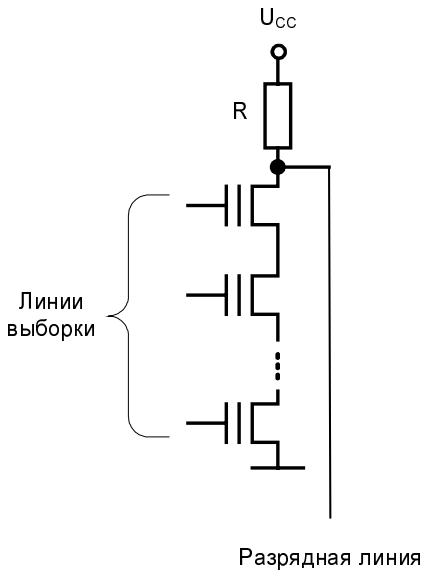


Рис. 5.27. Ячейка И-НЕ для схем флэш-памяти

Средства улучшения характеристик

Для улучшения технико-экономических характеристик в схемах флэш-памяти применяются различные средства и приемы (часть из них присуща и другим типам ЗУ).

- Прерывание записи при обращениях процессора для чтения (Erase Suspend). Без этого возникали бы длительные простои процессора, т. к. запись занимает большое время. После прерывания запись возобновляется под управлением внутренних средств флэш-памяти.
- Внутренняя очередь команд управления, позволяющая организовать конвейеризацию выполняемых операций и ускорить процессы чтения и записи.
- Программирование длины хранимых слов для согласования с различными портами ввода/вывода.
- Введение режимов пониженной мощности на время, когда к ЗУ нет обращений, в том числе режима глубокого покоя, в котором мощность снижается до крайне малых значений.
- Автоматическая приспособляемость к работе с разными питающими напряжениями.

- Введение в структуру микросхемы страничных буферов для быстрого приема данных, подлежащих записи. Два таких буфера могут работать в режиме "пинг-понг", когда один из них принимает данные, подлежащие записи, а другой в это время переписывает свое содержимое в память. Если второй уже освободится, в то время как первый заполнится, то они поменяются местами, и прием данных продолжится.
- Различные меры защиты от случайного или несанкционированного доступа.
- Применение специальных кодов для обнаружения и исправления ошибок, поскольку в схемах столь высокого уровня интеграции трудно полностью избежать появления неработоспособных запоминающих элементов.
- Выполнение операций чтения во время проведения в других блоках операций стирания/записи (в структурах Concurrent Flash Memory).

Флэш-память имеет перед EEPROM и EPROM ряд преимуществ и зачастую *наиболее удобна как энергонезависимая память для самых разных условий.* Вначале ее применение сдерживалось высокой стоимостью, но позднее положение изменилось. Сейчас флэш-память широко используется пользователями компьютеров, входит в схемы практически всех персональных компьютеров и сотовых телефонов, многих модемов и т. д. В перспективе сильным конкурентом флэш памяти будет ферроэлектрическая память FRAM (см. *§ 5.17*).

Команды управления

В микросхемах флэш-памяти помимо чтения непосредственно в системе выполняются стирание старых данных и запись новых, процессы сложные и длительные. Поэтому управление флэш-памятью имеет относительно сложный характер. В отличие от традиционного управления с помощью адресных и управляющих сигналов флэш-память имеет *дополнительное управление словами-командами*, записываемыми процессором в специальный *внутренний командный регистр.* Команды определяют работу *внутреннего автомата управления*, обеспечивающего для реализуемого режима необходимую последовательность действий. Типичный состав команд:

- две команды для стирания (подготовка стирания/стирание и проверка стирания);
- две команды для программирования (подготовка программирования/программирование и проверка программирования);
- две команды, задающие операцию чтения (для данных и кодов идентификатора);
- команда, задающая операцию сброса.

После команды стирания правильность операции проверяется чтением содержимого ячеек. Чтение правильного кода (состоящего из единиц) показывает, что все биты слова стерты. Если считывается иной код, выполняется повторное стирание и его проверка. Процесс проверки продолжается до достижения последнего адреса.

Программирование ведется слово за словом (последовательно или при произвольном доступе). При этом цикл чтения от процессора выводит записанные данные, ко-

торые сравниваются с заданными. Равенство байтов свидетельствует об успешном программировании. После этого процесс программирования переходит к следующему слову.

Команда сброса надежно устраняет возможное последействие команд стирания/программирования. После каждой из этих команд в регистр команд можно записать код операции сброса, что устранит возможность каких-либо влияний, связанных с указанными командами. Содержимое памяти уже не сможет изменяться. Для дальнейшего приведения схемы в желаемое состояние в регистр команд нужно записать соответствующую команду.

Память с несимметричными блоками

Блоки этой памяти *специализированы и имеют разные размеры*. В Boot-блоке (ББ) хранятся программы BIOS (Basic Input/Output System), необходимые для правильной эксплуатации и инициализации системы. Содержимое ББ аппаратно и программно защищено от случайного стирания. При этом в микросхему вводится пароль пользователя.

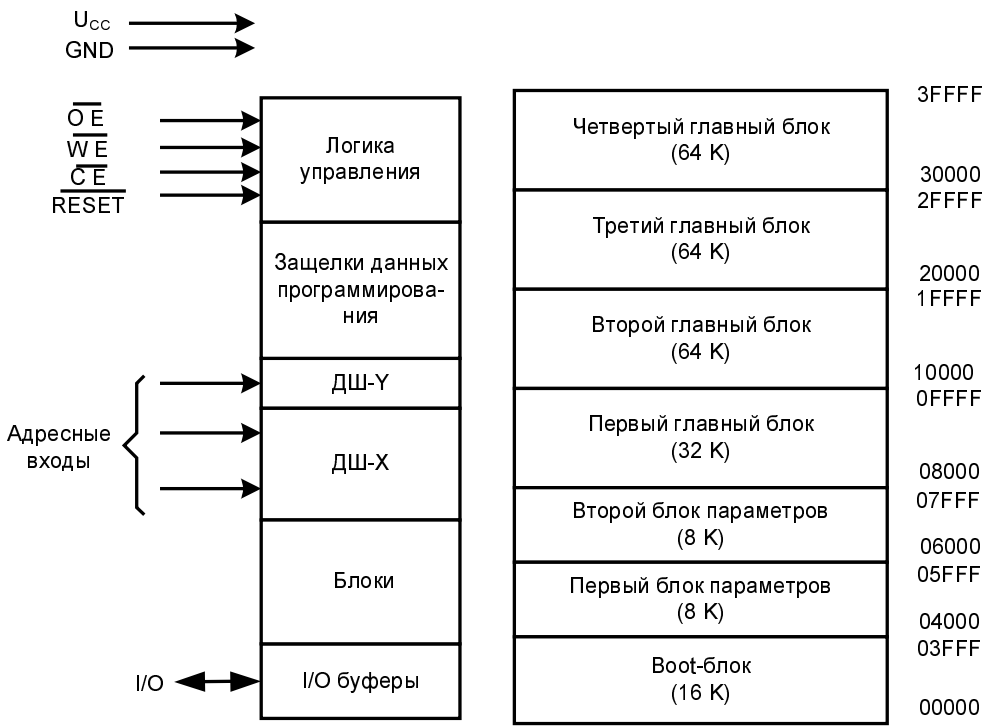


Рис. 5.28. Структура и внешняя организация флэш-памяти с несимметричными блоками

В составе блоков имеются также блоки параметров (БП), хранящие параметры системы (коды идентификаторов, диагностические программы и т. п.) и главные блоки (ГБ), хранящие, например, реализуемые программы.

На рис. 5.28 показана упрощенная структура и внешняя организация флэш-памяти с несимметричными блоками. Схема работает от одного источника питания как в режиме чтения, так и в режиме программирования. Для выполнения сложных операций стирания/записи имеются внутренние схемы управления и таймер. Boot-блок имеет емкость 16 Кбайт, два блока параметров имеют емкости по 8 Кбайт, три главных блока — по 64 Кбайта и один 32 Кбайта.

Адресные входы позволяют адресовать 256К ячеек, линии входов/выходов I/O имеют байтовый формат, L-активные сигналы $\overline{CE}$ (Chip Enable), $\overline{OE}$ (Output Enable), $\overline{WE}$ (Write Enable) и $\overline{RESET}$ (сброс) управляют работой схемы. Чтение данных обеспечивается сочетанием сигналов $\overline{OE}$ = $\overline{CE}$ = L (Low — низкий уровень сигнала), $\overline{WE}$ = H (High — высокий). При этом данные, хранимые в адресованной ячейке, появляются на выходах микросхемы. Если $\overline{CE}$ = H или $\overline{OE}$ = H, то выходы переводятся в третье состояние. Адреса и данные защелкиваются перепадами сигналов $\overline{WE}$ и $\overline{CE}$.

Блоки программируются последовательностью операций стирания данных в блоке, содержащем изменяемые байты, и затем программированием блока в режиме байт за байтом. Для инициализации стирания в микросхему вводится последовательность байтов (команда стирания), в которой фиксируется и адрес блока. Сама операция управляется внутренними средствами кристалла без использования внешних синхросигналов. Все элементы памяти в стираемой области приводятся в состояние логической единицы. При программировании состояние соответствующих элементов изменяется на нулевое.

Программирование завершается по истечении определенного времени, для его индикации предусмотрены специальные меры (контроль состояния одного из разрядов шины I/O). Если выявляется конец цикла программирования байта, то можно начинать новый доступ для операций чтения или программирования.

При включении питания в зависимости от управляющих сигналов схема окажется в режимах чтения или ожидания (Standby). Для перехода к другим режимам необходимо вводить командные последовательности, запись которых осуществляется нулевым импульсным сигналом, подаваемым на вход $\overline{WE}$ или $\overline{CE}$ при низком уровне другого входа и $\overline{OE}$ = H.

Вход $\overline{RESET}$ используется для упрощения некоторых системных операций. При высоком уровне напряжения на этом входе схема находится в обычном рабочем режиме. Переход сигнала $\overline{RESET}$ к низкому уровню останавливает текущие операции и переводит выходы схемы в третье состояние. Если такой переход произошел при выполнении операций стирания или программирования, то они не могут быть впоследствии продолжены и должны быть повторены с самого начала после возвращения сигнала сброса к высокому уровню. Возврат сигнала сброса к высокому

уровню приводит схему в режим чтения или ожидания в зависимости от состояния управляющих сигналов. При подаче на вход $\overline{\text{RESET}}$ повышенного напряжения Boot-блок может быть репрограммирован.

Микросхемы флэш-памяти с Boot-блоками предназначены для работы с разными микропроцессорами и для соответствия им имеют два варианта расположения этих блоков в адресном пространстве: вверху или внизу (как на рис. 5.28), что отображается в маркировке ИС буквами T (Top) или B (Bottom).

Память с симметричными блоками (файловая)

Важное место в иерархии ЗУ занимает *файловая флэш-память* (ФФП) с симметричными блоками. Хранение больших объемов данных традиционно возлагалось на хорошо отработанные и сравнительно недорогие внешние ЗУ на магнитных дисках. Во многих компьютерах основа системы памяти организована как сочетание жесткого магнитного диска (винчестера) с динамическим полупроводниковым ОЗУ.

Обладая рядом достоинств, дисковые ЗУ, как электромеханические устройства, не свободны от недостатков: чувствительны к ударам, вибрациям и загрязнениям, имеют ограниченное быстродействие и значительное потребление мощности. Эти недостатки особенно сказываются в портативных устройствах с автономным (батарейным) питанием. Достаточно отметить, что дисководы потребляют мощность около 3 Вт, что в системах с напряжениями питания 3,3...5 В означает потребление токов 0,6...0,9 А, что приводит к быстрому истощению питающих батарей.

Файловая флэш-память может быть *ориентирована на замену жестких дисков*, поскольку она в сотни раз сокращает потребляемую мощность, в той же мере увеличивает механическую прочность и надежность ЗУ, уменьшает их размеры и вес, на несколько порядков повышает скорость чтения данных. Использование ФФП для замены дисковой памяти в портативных компьютерах — один из важнейших факторов, способствующих развитию этого направления. За дисковой памятью пока остаются преимущества по информационной емкости и стоимости.

Накопитель ФФП делится на идентичные блоки — аналоги секторов магнитных дисков. Программные средства обеспечивают обмен между блоками, подобный обмену между секторами диска. В ФФП запись производится значительно чаще, чем в памяти с несимметричными блоками, и этой операции уделяется большое внимание — вводятся страничные буферы, позволяющие с высокой скоростью накапливать некоторый объем данных, подлежащих записи, для их последующей передачи в накопитель с меньшей скоростью.

Внешняя организация ФФП. Внешняя организация ФФП показана на рис. 5.29 на примере микросхемы с организацией $2^n \times m$. Для численных примеров приняты n = 24 и m = 8 или 16 в зависимости от варианта конфигурации памяти.

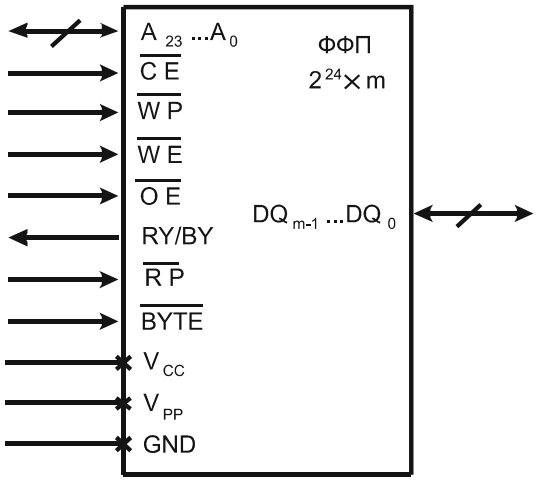


Рис. 5.29. Внешняя организация флэш-файловой памяти

Выводы и сигналы, показанные на рис. 5.29, имеют следующий смысл. Старшие разряды адреса выбирают один из блоков, младшие выбирают байт или слово в пределах блока.

Например, для нашего примера памяти с информационной емкостью 128 Мбит и блоками по 128 Кбайт для указания блока будут использованы 7 старших разрядов адреса, а для указания байта — 17 младших.

При организации памяти с использованием слов (словарной организации) разрядность адреса сокращается на единицу (количество слов вдвое меньше количества байтов) и линия A_0 (младший бит адреса) отключается. При адресации блочных данных от процессора поступает и запоминается начальный адрес блока. Текущий адрес ячейки формируется адресным счетчиком.

В 16-разрядной двунаправленной шине данных DQ_{15-0} линии DQ_{7-0} служат для ввода и вывода младшего байта данных, а также передачи в цикле записи команд для внутреннего автомата управления стиранием/записью (CUI). В циклах чтения по линиям DQ_{7-0} выводятся данные из буфера или регистров (идентификатора или состояния). Линии DQ_{15-8} передают старший байт при словарной организации памяти. По ним выводятся данные накопителя, буфера или идентификатора в соответствующем режиме чтения. Если кристалл не выбран или запрещен вывод, линии шины данных переходят в третье состояние.

Назначение управляющих сигналов:

- $\overline{CE}$ — (Chip Enable) вход разрешения кристалла, при его высоком уровне кристалл не выбран, и потребление мощности после завершения текущей операции записи или стирания снижается до уровня покоя.
- $\overline{OE}$ — (Output Enable) открывает выходные буферы при низком уровне и переводит их в третье состояние при высоком.

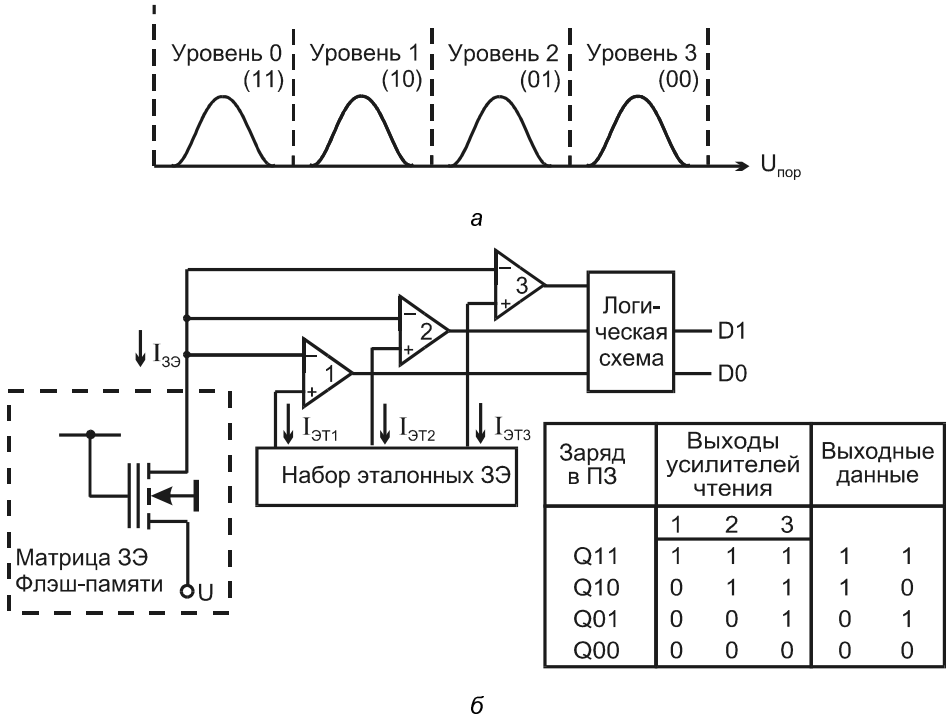
- $\overline{\text{WE}}$ — (Write Enable) управляет доступом к командному интерфейсу пользователя CUI, страничным буферам, регистрам очереди данных и защелкам очереди адресов.
- $\overline{\text{RP}}$ — (Reset/Power-Down) при переходе к низкому уровню прекращает операции автомата записи, сбрасывает схему и вводит ее в состояние глубокой экономии мощности. При выходе из этого состояния время восстановления достаточно велико (сотни наносекунд).
- RY/BY — (Ready/Busy) индицирует состояние внутреннего автомата записи. Низкий уровень означает занятость, высокий означает или готовность к новым операциям, или приостановление стирания, или состояние глубокой экономии мощности в зависимости от выполняемой операции.
- $\overline{\text{WP}}$ — (Write Protect) имеет следующий смысл. Каждый блок имеет бит запрещения записи (Lock-bit). Низкий уровень $\overline{\text{WP}}$ разрешает восприятие битов запрещения, при этом запись или стирание в блоке могут выполняться только при Lock-bit = 0. При высоком уровне сигнала $\overline{\text{WP}}$ в блоках могут выполняться операции записи и стирания независимо от состояния блокирующих битов.
- $\overline{\text{BYTE}}$ — низким уровнем вводит схему в байтовый режим, высоким — в словарный и выключает буфер линии A_0.

Микросхемы ФФП в настоящее время (2007 г.) имеют информационную емкость до 32 Гбит при байтовой или программируемой разрядности, временах доступа при чтении около 50 нс и времени программирования одного байта около 10 мкс. Длительности процессов стирания и записи блоков лежат в секундном диапазоне.

Память с многоуровневым хранением заряда

В 1997 г. компания Intel выпустила новый вид флэш-памяти, названный StrataFlash, в которой в одном запоминающем элементе хранятся два бита, а не один. Эта возможность обеспечивается тем, что в *плавающем затворе запоминающего транзистора фиксируется не только наличие или отсутствие заряда, но и его величина, которая может иметь несколько значений*. Различая четыре уровня, можно хранить в одном элементе два бита.

До изобретения элементов памяти с многоуровневым хранением заряда (MultiLevel Cells, *MLC*) для увеличения емкости ЗУ шли путем усовершенствований процессов литографии, уменьшения размеров схемных элементов и т. п. Память с MLC ознаменовала другой подход к этой проблеме. Хранения двух битов добились практически в тех же запоминающих элементах, которые ранее хранили один бит, преодолев трудности ужесточения допусков на величины вводимых в плавающий затвор зарядов. При этом от емкости 32 Мбита перешли к емкости 64 Мбита без заметных изменений площади кристалла.



Заряд в ПЗ	Выходы усилителей чтения			Выходные данные	
	1	2	3		
Q11	1	1	1	1	1
Q10	0	1	1	1	0
Q01	0	0	1	0	1
Q00	0	0	0	0	0

Рис. 5.30. Графики распределения пороговых напряжений в четырехуровневом запоминающем элементе (*а*) и схема чтения данных из этого элемента (*б*)

Запоминающие элементы памяти StrataFlash программируются введением в плавающий затвор одного из 4-х количеств заряда, каждое из которых соответствует паре двоичных цифр 11, 10, 01, 00. В зависимости от заряда, запоминающий транзистор имеет одно из четырех пороговых напряжений. При чтении к затвору транзистора прикладывают напряжение считывания. Ток запоминающего транзистора зависит от порогового напряжения. Определяя ток, можно выявить заряд плавающего затвора.

На рис. 5.30 показаны статистическое распределение пороговых напряжений в четырехуровневом запоминающем элементе (*а*) и схема чтения состояния запоминающего транзистора (*б*).

Переход к многоуровневому хранению заряда в плавающем затворе транзистора, дающий большие преимущества по емкости памяти при той же площади кристалла, приводит одновременно к уменьшению примерно на порядок числа допустимых циклов репрограммирования.

Память с зеркальным битом

В 2001 г. фирма AMD предложила новый вариант флэш-памяти, названный *схемой с зеркальным битом* (Mirror Bit). В этих схемах, как и в ячейках с четырьмя уров-

нями хранимого заряда, можно удвоить емкость памяти по сравнению со стандартными вариантами, но достигается это иным путем. В одном запоминающем элементе хранятся два бита данных *в виде индивидуальных зарядов, размещенных в разных местах подзатворного слоя одного и того же транзистора.* Иными словами, используется пространственное разделение двух зарядов, каждый из которых отображает свой бит хранимой информации (рис. 5.31).

Транзистор запоминающего элемента в схемах с зеркальным битом отличается от обычного идентичностью областей истока и стока (в обычных транзисторах исток и сток легируются различно). Запоминающая область модифицирована так, чтобы группы электронов могли независимо храниться в обеих ее сторонах. При этом транзистор проявляет себя как два обычных запоминающих элемента.

Поскольку многоуровневое хранение заряда не используется, проблема жестких допусков на величину вводимого заряда снимается, а это позволяет проводить процессы чтения, стирания и записи с сохранением высокой скорости. По сравнению с вариантом многоуровневого хранения заряда повышается и надежность сохранения данных, поскольку при многоуровневом хранении допустимые потери заряда, естественно, значительно меньше, чем при двухуровневом. Указанные факторы следует считать достоинствами варианта с зеркальным битом. В то же время разработчики многоуровневых ячеек хранения уже реализовали возможность хранения в одном элементе более чем двух бит (4 бит и более). В 2006 г. появились флэш-ЗУ, сочетающие многоуровневое хранение заряда с концепцией зеркального бита.

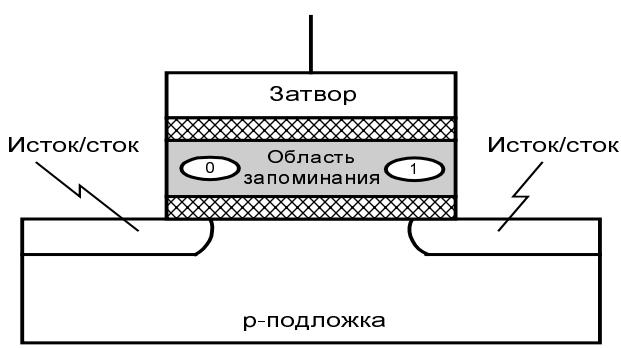


Рис. 5.31. Структура запоминающего элемента с зеркальным битом

Флэш-память с MLC-ячейками И-НЕ

Современная память высшего уровня интеграции — это флэш-память на ячейках И-НЕ с многоуровневым хранением заряда или даже с сочетанием этого метода и метода зеркального бита.

Основной функциональный блок матрицы флэш-памяти с ячейками И-НЕ и хранением четырех уровней заряда (MLC NAND-flash, Multilevel NAND-flash) [55] показан на рис. 5.32. Однотранзисторные запоминающие элементы (ЗЭ) хранят по два

бита, поскольку при программировании пороговое напряжение каждого транзистора задается одним из четырех возможных значений путем соответствующей инжекции заряда в его плавающий затвор. Каждый ЗЭ строки затвором подключен к одной из 32 линий CG (Control Gate) и входит в столбец одной из 32 разрядных линий (BL, Bit Lines). Кроме этих линий имеются линия SelL (Select Lines) для управления доступом к блоку и линия источника напряжения SL (Source Line).

Данные считываются из строки параллельно. При этом потенциал линий SelL и невыбранных линий CG должен быть высоким. Затем на выбранную линию затворов последовательно подаются четыре значения напряжения. Когда затворное напряжение превысит пороговое напряжение транзистора, по соответствующей линии BL пройдет ток, что и выявляет уровень хранимого заряда. Как видно, за увеличение объема хранимой информации в рассмотренной схеме с многоуровневым зарядом приходится платить снижением быстродействия — для считывания данных требуется подавать четыре импульса, а не один, как это было бы в схеме с одноуровневым зарядом.

В 2006 г. разработана схема ЗУ по NROM-технологии с подзатворным диэлектриком транзистора в виде чередования слоев "оксид-нитрид-оксид", допускающим локальное хранение зарядов, как это происходит в конструкции с зеркальным битом, где заряды хранятся в двух раздельных областях. Хранимые в каждой области заряды задаются четырьмя уровнями, так что в целом однотранзисторная ячейка хранит 4 бита.

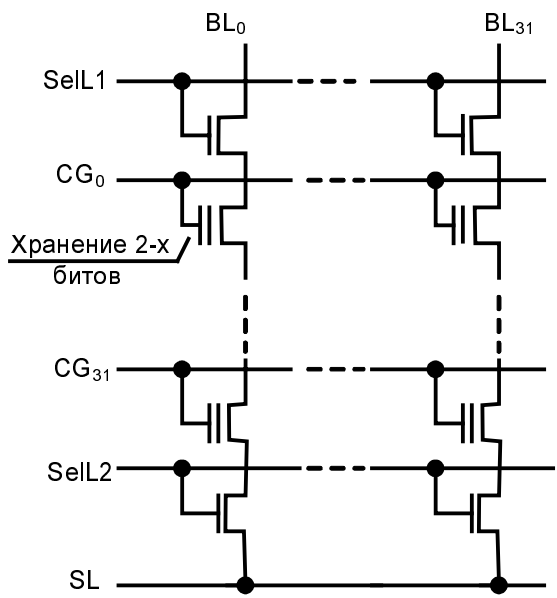


Рис. 5.32. Функциональный блок флэш-памяти на основе ячеек И-НЕ с многоуровневым хранением заряда

§ 5.6. Последовательные репрограммируемые ЗУ

В большинстве проектов энергонезависимая память представлена традиционными микросхемами EEPROM и флэш-памяти. Но существует и немало проектов со скромными требованиями к быстродействию памяти. Для таких проектов целесообразно заменить традиционные схемы значительно более простыми и дешевыми последовательными вариантами.

Переходя к последовательным вариантам, жертвуют быстродействием, но приобретают компактность и низкую стоимость микросхем, в которых исключаются многоразрядные шины. Адреса и данные вводятся в схемы по одной линии (через один контакт) последовательно, разряд за разрядом. Выходные слова также выводятся через один контакт в последовательной форме. *Длительность передач адресов и данных при этом резко увеличивается, но столь же резко упрощается корпус микросхемы, растет ее компактность и снижается стоимость.* Корпус микросхемы последовательной памяти может иметь *всего 5 выводов*: для напряжения питания, "земли", входных данных, выходных данных и сигнала разрешения работы (одна из первых микросхем с информационной емкостью 1 Кбит была выпущена в корпусе SOT-23 с пятью выводами площадью всего 5 мм^2, при этом для последующих вариантов уже разрабатывались корпуса с площадью 2 мм^2).

Последовательные EEPROM и флэш-память по информационной емкости перекрывают диапазон от сотен бит до нескольких мегабит и приобретают все более разнообразные свойства. С помощью блоков SERDES они могут обеспечивать работу с 8- или 16-разрядными словами, иногда и с возможностью перестройки организации памяти. Допустимое число циклов репрограммирования составляет для них $10^5 \ldots 10^6$, гарантированное время сохранения данных 10…100 лет. Очень мала и потребляемая мощность — при напряжении питания 1,8 В токи потребления в активном режиме составляют приблизительно 1 мА, а в режиме покоя (Standby) всего несколько микроампер. *Малые габариты и малая потребляемая мощность* — важнейшие качества микросхем, предназначенных для портативной аппаратуры с батарейным питанием.

Микросхемы последовательной памяти нуждаются в скоростных интерфейсах. С ростом информационной емкости памяти растет и разрядность слов, а это требует и повышения скоростей последовательных интерфейсов. Действительно, при той же частоте передачи слов темп передач битов пропорционален длине слов, поскольку при переходе к параллельным кодам за то же время в сдвигающие регистры преобразователей последовательного кода в параллельный и наоборот требуется вводить (или выводить из них) большее число битов.

> Например, при тактовой частоте преобразователей 10 МГц преобразование потока битов в 8-разрядные слова дает частоту потока слов приблизительно 1,25 МГц, а для получения той же частоты потока 16-разрядных слов требуется частота потока битов приблизительно 20 МГц.

Наряду с обычными базовыми интерфейсами для последовательной памяти реализованы и такие, в которых пакет выходных данных формируется слитно после загрузки в память стартового адреса. Такой режим исключает задержки на загрузку нового адреса для каждого слова данных. Так, например, в микросхемах семейства AT93CXX фирмы Atmel после получения команды чтения и адреса выходной контакт выходит из третьего состояния, на выходе формируется нулевой бит, после которого выдвигается адресованное слово. При сохранении активного сигнала разрешения работы адрес автоматически инкрементируется и выводится следующее слово данных. Этот процесс может продолжаться вплоть до достижения конца адресного пространства, после чего адрес становится нулевым и процесс чтения продолжается.

§ 5.7. Импульсное питание ROM

Энергонезависимость микросхем памяти, сохраняющих информацию при отключении питания, открывает возможности экономии мощности при их эксплуатации, что особенно важно для портативной аппаратуры.

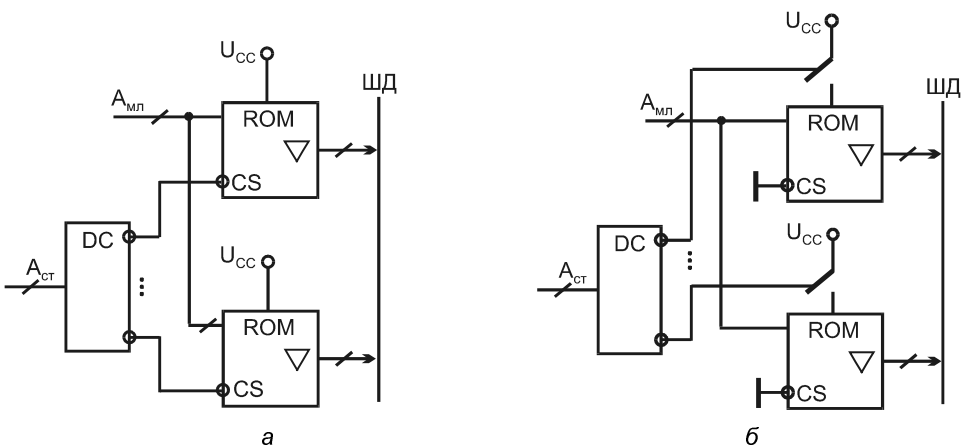


Рис. 5.33. Модули постоянной памяти с обычным (*а*) и импульсным (*б*) питанием

Питание можно подавать только на кристалл, к которому в данный момент происходит обращение. На рис. 5.33 показан обычный вариант модуля памяти, состоящего из нескольких ИС, и вариант с импульсным питанием. В обычном варианте напряжение U_{cc} подключено ко всем ИС постоянно, а выбор адресуемой ИС осуществляется сигналом $\overline{CS}$. В варианте с импульсным питанием работа всех ИС по входам $\overline{CS}$ постоянно разрешена, но питание подключается только к выбранной микросхеме с помощью ключа, управляемого от выходов адресного дешифратора, декодирующего старшие разряды адреса.

Режим импульсного питания может многократно уменьшить потребляемую модулем мощность, но, одновременно, увеличивает время обращения к ЗУ при одиночных произвольных обращениях, т. к. после включения питания необходимо время для установления режима микросхемы. При чтении данных, расположенных по близким адресам, когда старшие разряды адреса остаются неизменными, потерь времени не возникает.

§ 5.8. Использование программируемых ЗУ для решения задач обработки информации

Основное назначение запоминающих устройств — хранение информации. Однако память является также и средством решения задач *обработки* информации. Применимость памяти в указанной области определяется возможностью представления решения задачи в табличной форме. Эта форма решения возможна для задач самого разного характера.

Реализация логических функций

Память с организацией $2^m \times 1$ принимает m-разрядный адрес и выдает одноразрядный результат (0 или 1). Это непосредственно воспроизводит логическую функцию m переменных, т. к. для каждого входного набора аргументов (адреса) можно при программировании ЗУ назначить необходимую выходную переменную. Например, ЗУ с организацией $1К \times 1 = 2^{10} \times 1$ может быть использовано для воспроизведения функции 10 аргументов. Время выполнения операции — время считывания данных из ЗУ.

Память с организацией $2^m \times n$ по поступающему на его вход m-разрядному адресу выдает n-разрядное выходное слово. Такое ЗУ воспроизводит *систему переключательных функций*, число которых равно разрядности выходного слова. Действительно, в каждом разряде совокупности выходных слов может быть воспроизведена любая переключательная функция m-аргументов, а совокупность разрядов даст n различных функций.

В ЗУ функции реализуются в совершенной дизъюнктивной нормальной форме (СНДФ), каждая выходная линия дешифратора адреса представляет минтерм, значение которого при программировании вводится в выходную функцию. *Какой-либо минимизации функций при подготовке задачи к решению на основе ЗУ не требуется*, более того, если функции уже минимизированы, то для удобства подготовки данных для программирования ЗУ их придется развернуть до СДНФ. Это делается либо заполнением карты Карно и последующей записью функции без какого-либо объединения единиц, либо введением в каждую конъюнкцию недостающих переменных x_i путем домножения конъюнкции на равные единице выражения $x_i \vee \overline{x_i}$ с

последующим раскрытием скобок (x_i — вводимая переменная). Пример приведения функции в СДНФ:

$$F = x_1 \vee x_2 x_3 = x_1(x_2 \vee \bar{x}_2)(x_3 \vee \bar{x}_3) \vee (x_1 \vee \bar{x}_1)x_2 x_3 = x_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 x_2 x_3.$$

Для воспроизведения этой функции по пяти конъюнкциям-адресам в ППЗУ следует записать единицы, по остальным адресам — нули.

Представление функции в СДНФ определяет большие затраты элементов памяти, однако цена элемента памяти значительно ниже цены логического элемента, поэтому реализация на ЗУ может оказаться выгодной в сравнении с воспроизведением функции традиционным методом. ЗУ наиболее целесообразно применять для воспроизведения функций, не поддающихся существенной минимизации.

Реализация конечных автоматов

В канонической схеме автомата память может заменить комбинационную цепь, поскольку она способна воспроизводить логические функции. Поэтому структура автомата без потери общности может быть представлена также в виде, приведенном на рис. 5.34.

Начальная установка регистра задает исходное состояние S_H элементов памяти (автомата). По этому состоянию и входным сигналам из памяти считывается код нового состояния и функции выхода. В следующем такте эти процессы повторяются. В каждом очередном такте автомат переходит в новое состояние и вырабатывает выходные функции согласно таблицам переходов и выходов.

Емкость ППЗУ определяется объемом таблиц, задающих функционирование автомата. Сведя таблицы переходов и выходов в одну, получим общее число входов $m = k + q$ и число выходов $n = p + q$, следовательно, для реализации автомата требуется емкость памяти $M = 2^{k+q}(p + q)$.

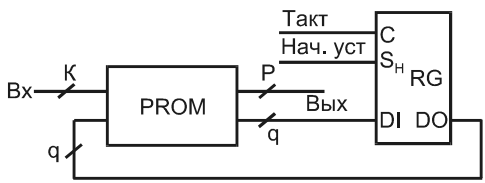


Рис. 5.34. Структура автомата, реализованного на основе микросхем памяти

Воспроизведение числовых функций

Под числовыми функциями будем понимать "обычные" (не логические) функции, которые могут быть заданы таблицей. Для функций одного аргумента объем памяти таблиц легко вычислить, зная разрядности аргумента и функции.

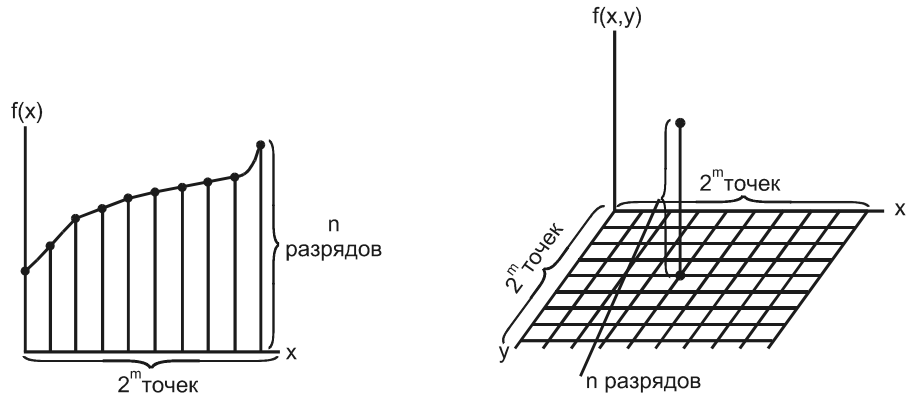


Рис. 5.35. К определению емкости памяти при воспроизведении табличным методом числовых функций одного (*а*) и двух (*б*) аргументов

При задании аргумента m-разрядным кодом число точек, в которых задана функция, составит 2^m (рис. 5.35, *а*). Если разрядность кода, представляющего функцию, равна n, то емкость памяти, воспроизводящей таблицу, будет равна $n2^m$ бит.

С ростом числа аргументов объем памяти для таблиц функций быстро растет. Для функции двух m-разрядных аргументов число точек, в которых задана функция, определится как произведение чисел точек по каждой из координат и составит 2^{2m} (рис. 5.35, *б*). Объем памяти таблицы в этом случае составит $M = n2^{2m}$. Для функций ℓ аргументов $M = n2^{\ell m}$.

С ростом разрядности слов и числа аргументов объем памяти таблиц быстро растет и чисто табличный метод решения задачи может стать неприемлемым. В этих случаях полезны таблично-алгоритмические методы, в рамках которых можно существенно снизить объем таблиц, введя небольшое число простых операций над данными.

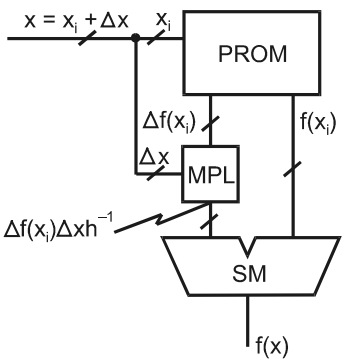


Рис. 5.36. Структура кусочно-линейного аппроксиматора функций одного аргумента

Для произвольных функций f(x) простейший таблично-алгоритмический метод — кусочно-линейная аппроксимация, когда запоминаются только узловые значения функции, а в промежутках между узлами функция вычисляется в предположении, что на промежутках она изменяется линейно. Число узлов назначается по соображениям точности линейной аппроксимации функции на участках. Кусочно-линейной аппроксимации с постоянным шагом соответствуют следующие представления аргумента и функции:

$$x = x_i + \Delta x,\ f(x) = f(x_i) + \Delta f(x_i)\Delta x h^{-1},$$

где xi — координата i-й узловой точки; Δx — разность значений x и координаты ближайшей слева узловой точки; $\Delta f(x_i)$ — приращение функции на участке от xi до xi+1; h — шаг аппроксимации (для удобства реализации цифровыми методами шаг берут равным целой степени числа 2).

Согласно приведенным формулам структура функционального преобразователя с кусочно-линейной аппроксимацией имеет вид, приведенный на рис. 5.36.

Для функций двух переменных можно применить кусочно-плоскостные аппроксиматоры.

§ 5.9. Статические оперативные ЗУ

Статические ОЗУ (SRAM) имеют высокую стоимость и применяются как память *высшего быстродействия* в кэш-памяти, при построении буферов FIFO и LIFO, как небольшие блоки памяти в микроконтроллерах и т. п. Эти ОЗУ чаще всего имеют структуру 2DM. Их запоминающими элементами служат триггеры. В связи с этим статические ОЗУ называют также *триггерными.* Триггеры и память в целом можно реализовать по любой схемотехнологии (ТТЛ(Ш), $И^2Л$, ЭСЛ, n-МОП, КМОП, AsGa и др.). На разных этапах на ведущие роли выдвигались различные схемотехнические типы ЗУ. Сейчас доминируют КМОП-ЗУ, поскольку при субмикронной технологии КМОП-схемы, сохраняя свои традиционные достоинства, приобрели также и высокое быстродействие.

Типичная номенклатура SRAM зарубежных фирм представлена сейчас микросхемами с информационной емкостью в диапазоне от 64 Кбит до 16 Мбит и временами доступа 10…15 нс. Среди отечественных микросхем можно отметить серии К537 технологии КМОП и К132 технологии n-МОП (эти серии являются старыми разработками с соответствующим уровнем технических характеристик).

Структура асинхронного (стандартного) ЗУ

Асинхронное ЗУ со структурой 2DM (рис. 5.37) в своей основе совпадает с обобщенной структурой (см. *§ 5.2*), отличаясь от нее лишь парафазностью сигналов считывания/записи и способом выборки столбцов матрицы запоминающих элементов.

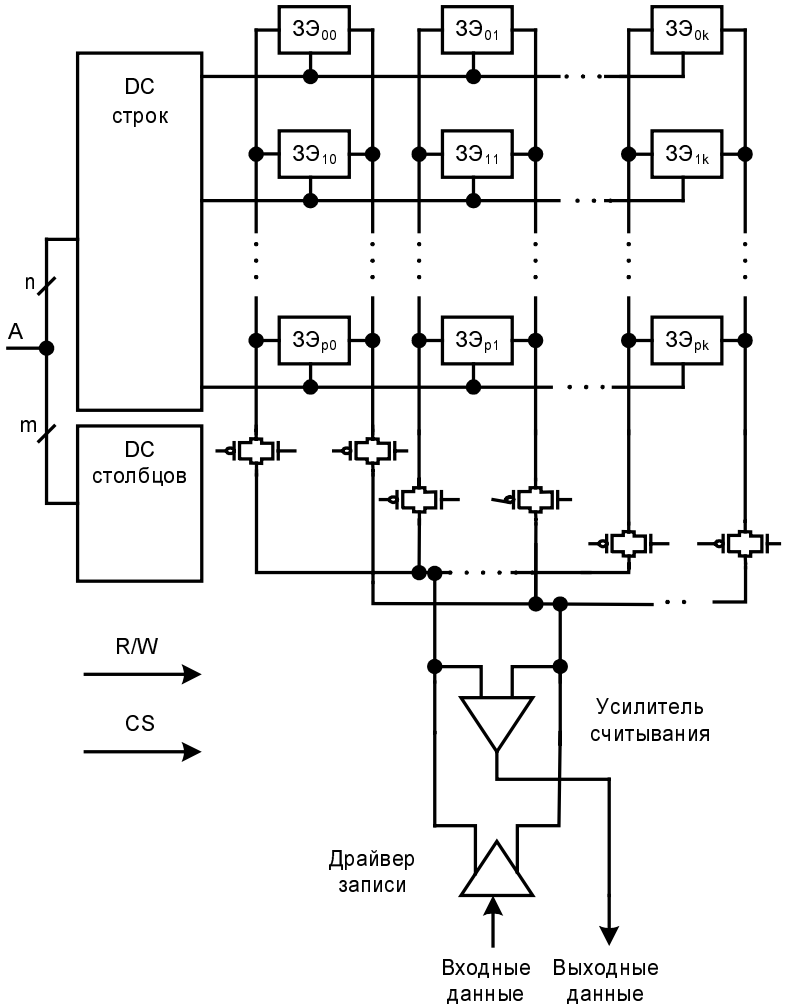


Рис. 5.37. Структура асинхронного статического ОЗУ

Адрес ячейки A делится на поля с разрядностями n и m. Дешифратор строк, получая n-разрядный код, выбирает в матрице одну из 2^n строк ($p = 2^n - 1$). Запоминающий триггер имеет прямой и инверсный выходы, поэтому столбец матрицы представлен двумя линиями (парафазно). Дешифратор столбцов управляет ключами, выбирающими один из 2^m столбцов ($k = 2m - 1$) для подключения к дифференциальному усилителю считывания данных (при чтении) или к дифференциальному выходу формирователя данных (драйверу) при записи. Во избежание громоздкости рисунка линии связей дешифратора столбцов и ключей не показаны. Соответственно режиму чтения или записи сигналом R/W разрешается работа усилителя считывания или драйвера записи. Организация ЗУ $2^{m+n} \times 1$.

Запоминающие элементы

Запоминающий элемент в схемотехнике КМОП

Этот ЗЭ (рис. 5.38, *а*) представляет собой RS-триггер, собранный на двух инверторах (транзисторы Т1, Т2 и Т3, Т4 соответственно), и ключи выборки (Т5 и Т6).

> ***Примечание***
>
> На рис. 5.38 и некоторых последующих приняты международные обозначения МОП-транзисторов. Транзистор с p-каналом открывается отрицательным напряжением и имеет на затворе символ инверсии. Транзистор с n-каналом открывается положительным напряжением на затворе и имеет прямой вход.

При обращении к ЗЭ высокий потенциал линии выборки открывает ключи по всей строке, и выходы ее триггеров соединяются со столбцовыми линиями считывания-записи (столбцовая линия D_j связана с прямым выходом триггера, $\overline{D}_j$ — с инверсным). Через столбцовые линии усилитель с дифференциальным входом считывает состояние триггера. При записи к этим же столбцовым линиям подключаются дифференциальные выходы драйвера записи, приводящие триггер в требуемое состояние. Так как записываемые сигналы должны преодолевать "сопротивление" триггера, если требуется изменить его состояние, транзисторы в цепях записи делаются более мощными, чем транзисторы в схемах триггеров.

При выборке со своими линиями столбцов соединяются все триггеры строки, но с цепями считывания/записи может связываться лишь часть столбцов в соответствии с адресной информацией, управляющей ключами (на рис. 5.38, *а* это ключи с параллельным соединением двух транзисторов с разными типами каналов). В нашем примере память одноразрядная и с цепями считывания/записи может связываться всего один столбец).

Запоминающий элемент в схемотехнике n-МОП

Второй вариант ЗЭ (рис. 5.38, *б*) — реализованный по n-МОП технологии RS-триггер (на транзисторах Т1 и Т2) с ключами выборки (Т3 и Т4). Этот ЗЭ функционирует принципиально так же, как и описанный ранее, но инверторы, входящие в схему триггера, в стоковых цепях имеют высокоомные поликремниевые резисторы, пространственно расположенные над транзисторами, что придает схеме как режим микротоков, так и высокую компактность. Недостаток схемы — наличие статического тока в схемах триггеров, чего нет в КМОП-схемах.

По занимаемой площади ЗЭ n-МОП приблизительно в 1,5 раза компактнее КМОП-элемента, по току покоя (в отсутствие переключений) заметно проигрывает ему (например, для одной из технологий: 10^{-12} А против 10^{-15} А).

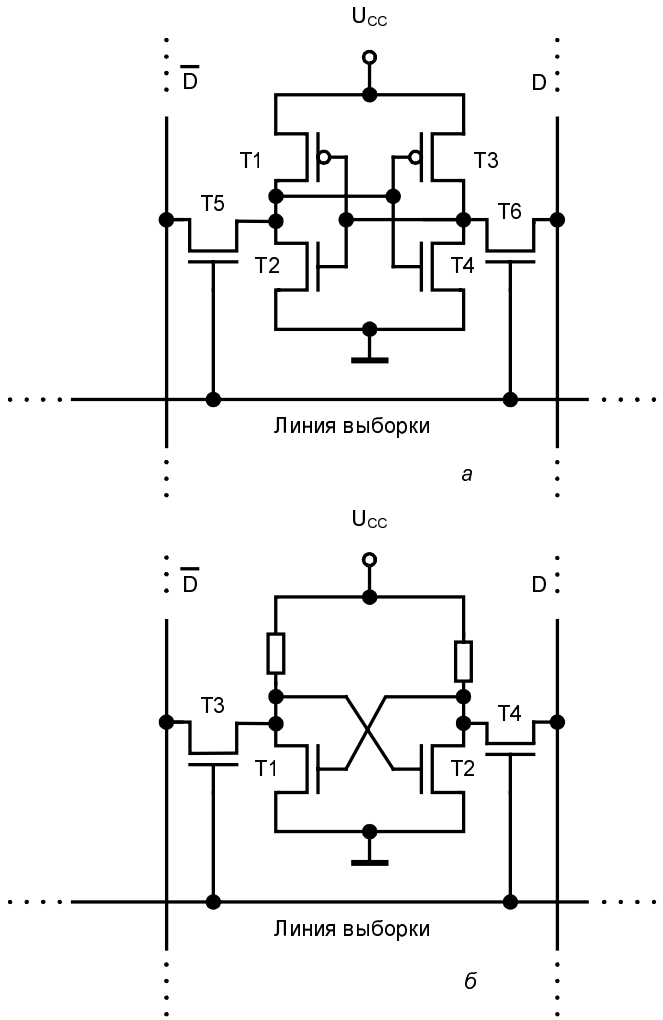


Рис. 5.38. Запоминающие элементы статического ЗУ в схемотехнике КМОП (*а*) и n-МОП (*б*)

Требования к усилителям считывания

Режим микротоков в ЗЭ нужен для ограничения потребляемой мощности, прежде всего для кристаллов высокого уровня интеграции, но создает и маломощность выходных сигналов. Эта особенность требует применения высокочувствительных усилителей считывания, в частности, так называемых *усилителей-регенераторов*, которые не только воспринимают слабые сигналы, но и *удерживают триггер-источник сигнала от изменения его логического состояния в процессе чтения* (усилители-регенераторы рассмотрены далее при описании динамических ЗУ, в которых и были впервые применены).

Внешняя организация и временные диаграммы

Выпускаются одноразрядные и многоразрядные статические ОЗУ. Внешняя организация ЗУ емкостью 64 Кбит (8К×8) показана на рис. 5.39. Назначение сигналов адреса $A_{12—0}$, выборки кристалла $\overline{CS}$ и чтения/записи R/W рассматривалось ранее. Входы и выходы данных совмещены в двунаправленной шине DIO. Вход $\overline{OE}$ разрешает вывод данных, его пассивное состояние ($\overline{OE}$ = H) переводит выходы в третье состояние. Работа ОЗУ отображается табл. 5.1.

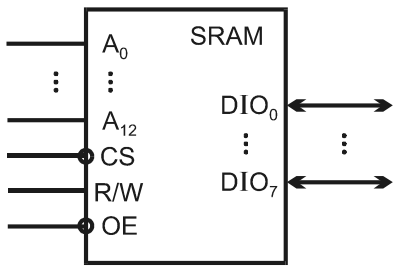


Рис. 5.39. Пример внешней организации статического ЗУ

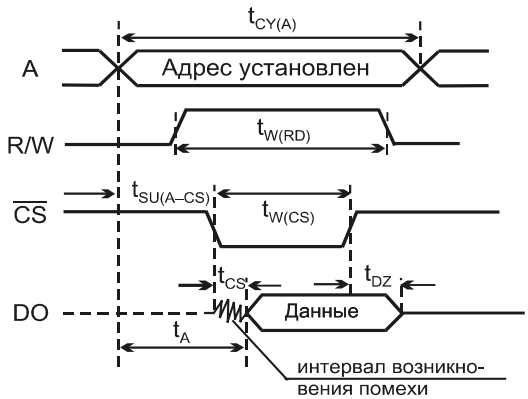
Таблица 5.1

$\overline{CS}$	$\overline{OE}$	R/W	A	DIO	Режим
1	X	X	X	Z	Хранение
0	X	0	A	DI	Запись
0	0	1	A	DO	Чтение

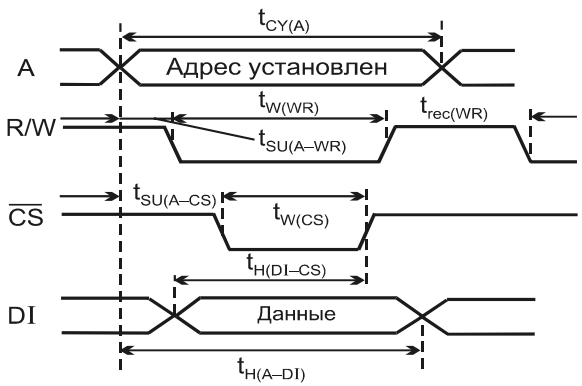
Управление памятью регламентируется временными диаграммами, устанавливаемыми изготовителем. Адрес подается раньше, чем другие сигналы, с опережением на время его декодирования и должен держаться в течение всего цикла обращения к памяти. Затем подаются сигналы $\overline{CS}$ и R/W, а также записываемые данные (если предполагается запись) или сигнал разрешения выхода $\overline{OE}$ (при чтении). *Стробирующий* сигнал (R/W или $\overline{CS}$) выделяет временной интервал непосредственного выполнения действия.

Требования к управляющим сигналам у разных ЗУ различны. Одни ЗУ требуют импульсного представления некоторых сигналов, для других такие требования отсутствуют (см. *§ 5.1 разд. "Классификация статических ЗУ"*).

Пример временных диаграмм для чтения и записи в асинхронном ЗУ показан на рис. 5.40.



а



б

Рис. 5.40. Временные диаграммы процессов чтения (*а*) и записи (*б*) в асинхронном статическом ЗУ

На диаграммах показаны времена доступа относительно адреса (t_A) и относительно сигнала $\overline{CS}$ (t_{CS}), длительности импульсов t_W различных сигналов и цикла адреса $t_{CY(A)}$, задержка t_{DZ} перехода выхода из активного состояния в состояние "отключено", времена предустановки t_{SU} и удержания t_H с указанием сигналов, для которых они отсчитываются. Приведено время восстановления $t_{rec(WR)}$, отсчитываемое как необходимая пауза между повторениями активных интервалов сигнала R/W. Асинхронные ЗУ работают по требованиям своих временных диаграмм, а процессор генерирует сигналы управления памятью согласно своим временным диаграммам. Поскольку (в отличие от ситуации с синхронными ОЗУ) эти *временные диаграммы независимы*, проектировщик должен позаботиться об их согласовании (см. *главу 6*).

Для проектирования модулей памяти необходимо также знать емкости их входов C_I, выходов C_O и предельно допустимую емкость нагрузки $C_{L\,max}$.

Пример асинхронного ЗУ

На рис. 5.41 показана схема асинхронного статического ОЗУ с информационной емкостью 1 Мбит, построенного по структуре 3D при организации 256К×4. Дополнительно к блокам, обычным для структуры 3D, схема имеет блок управления режимом глубокого понижения мощности (power-down), снижающим ее более чем на 65%, когда схема не выбрана, т. е. сигнал $\overline{CE}$ имеет высокий уровень. При $\overline{CE}$ = $\overline{WE}$ = L данные с выводов I/O0…I/O3 записываются по адресу, поданному на соответствующие входы микросхемы. Чтение происходит при $\overline{CE}$ = $\overline{OE}$ = L и $\overline{WE}$ = H. Это сочетание сигналов передает на выходы схемы содержимое ячейки, определяемой заданным адресом. При $\overline{CE}$ = H выводы I/O0…I/O3 переходят в состояние "отключено".

Синхронные ЗУ

Синхронные статические ЗУ (SSRAM) имеют модифицированный интерфейс для *согласования работы ЗУ с системой тактирования процессора* и могут работать со *сквозной* или *конвейерной* передачей данных. Согласованность временных диаграмм интерфейса (процессора) и ЗУ позволяет *исключить непроизводительные потери времени*, возможные в асинхронных ЗУ. В синхронных ЗУ моменты изменения всех сигналов точно известны, они фиксируются фронтами тактового сигнала CLK, вырабатываемого процессором. В частности, время приема данных от памяти при чтении точно известно, и в соответствующем такте процессор может без каких-либо потерь времени принять данные.

При сквозной передаче между обращениями к памяти выдерживается временной интервал, достаточный для формирования результата (например, до появления читаемых данных). В конвейеризованных устройствах (см. *§ 5.3*) положение иное: *не ожидая выработки конечного результата для данного цикла, можно начинать следующий*. При конвейеризации памяти между усилителями чтения и выходными буферами вводятся регистры, позволяющие увеличить частоту тактирования системы и расширить полосу пропускания памяти (напомним, что *полосой пропускания* называют произведение частоты передач данных на их разрядность). Повышению производительности систем с синхронными ЗУ способствует и наличие в синхронных микросхемах памяти регистров-защелок на входах и выходах, что позволяет процессору после выдачи для ЗУ адреса и сигналов управления сразу же переходить к другим операциям.

В структурах синхронных ЗУ их ориентация отображается на обмен данными, свойственный кэш-памяти (пакетный обмен). *Синхронные ЗУ с пакетным обменом* (Synchronous Burst SRAM) имеют внутренний счетчик адресов и в дополнение к обычным для асинхронных ЗУ управляющим сигналам следующие сигналы:

- тактовый сигнал процессора (системы) CLK;

- стробы, которыми процессор или контроллер записывают адрес во внутренний регистр микросхемы и инициируют очередной цикл обращения к памяти (пакетный или одиночный);
- сигнал для перехода к следующему адресу в пакете (последовательному или с чередованием банков).

Синхронные конвейеризованные ЗУ с пакетным обменом (Pipelined Burst SRAM). В этой структуре в тракт передачи данных введен дополнительный регистр. При этом, как и свойственно конвейеру, для первой передачи время увеличивается на дополнительный такт, но для последующих передач темп ускоряется.

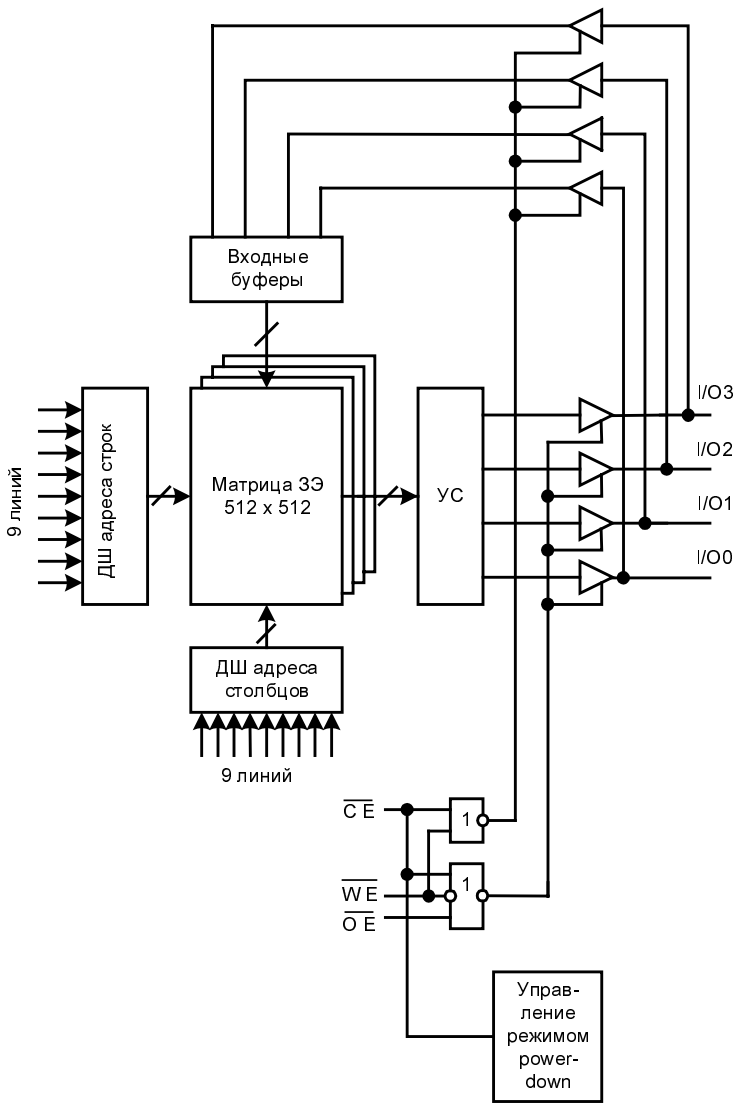


Рис. 5.41. Структура асинхронного статического ОЗУ

В быстродействующих SSRAM *сочетаются многие способы повышения быстродействия* (низковольтное питание, конвейеризованная структура, технология DDR, пакетные режимы, двухпортовость и т. д.).

Структура синхронных ЗУ

Синхронные ЗУ (рис. 5.42) отличаются от асинхронных тактируемым вводом/выводом адресов, данных, сигналов управления. К блокам стандартного ЗУ в данном случае добавляются регистры, включенные в тракты передачи адреса (RG A), сигналов управления (RG C) и входных данных. Благодаря этим регистрам все сигналы фиксируются и далее поступают в матрицу памяти по разрешению тактового сигнала CLK. Сигналы разрешения работы (CS) и разрешения записи (WE) также стробируются импульсами CLK. Входные данные всегда фиксируются в регистре, тогда как выходные либо поступают непосредственно на выводы DIO, либо (если выходы конвейеризуются) также фиксируются в регистре. Введен дополнительный сигнал $\overline{\text{ADS}}$, разрешающий загрузку регистра адреса (на рисунке не показан). При пассивном состоянии этого сигнала прекращается прием новых значений адреса. Сигнал $\overline{\text{GW}}$ (Global Write) своим переходом на активный уровень определяет операцию записи.

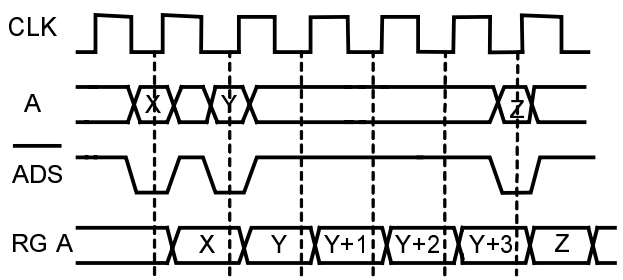


Рис. 5.42. Структура синхронного ЗУ

Временные процессы в SSRAM подчинены ее работе в составе кэш-памяти. Внутренние регистры воспринимают внешнюю информацию в следующем после ее появления такте. Регистр входных данных фиксирует их при записи, выходные данные при чтении фиксируются регистром, если он имеется в схеме (т. е. при конвейеризации выхода). На прием адресов в регистре RG A влияет сигнал $\overline{\text{ADS}}$. В следующем такте происходит обращение к матрице ОЗУ, так что на шине DIO при чтении после некоторой задержки появляются данные. При пакетном режиме данные считываются из последовательности ячеек, а регистр RG A работает как счетчик. Подачи новых адресов после каждого цикла не требуется (рис. 5.43).

При записи входные данные могут восприниматься только через такт после загрузки регистра RG A (временно хранятся в регистре входных данных). Для этого задерживается сигнал $\overline{\text{ADS}}$, что предотвращает изменение состояния RG A. Запись

происходит в такте после перехода сигнала $\overline{GW}$ на низкий уровень. При записи также возможен пакетный режим. В таком режиме (протокол с задержанной записью) матрица ОЗУ "простаивает" в течение одного такта, если запись происходит не по соседним адресам.

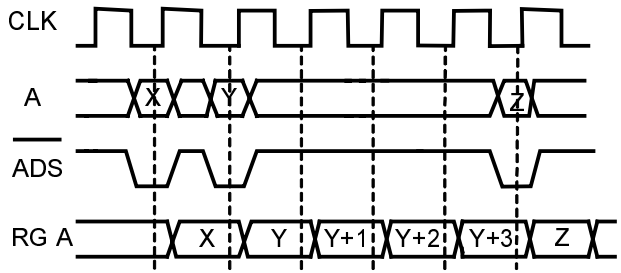


Рис. 5.43. Временные диаграммы сигналов SSRAM

ЗУ с ускоренным реверсом шины. Существуют архитектуры SSRAM с *уменьшенным временем реверсирования направления передач по шине данных* (с исключением потери такта при реверсе шины "чтение — запись"). К таким архитектурам относятся микросхемы ZBT (Zero Bus Turnaround), NoBL (No Bus Latency), NtRAM (No Turnaround RAM) и др. В этих архитектурах удалось исключить свободный такт, обычно возникающий при переходе от операции чтения к операции записи и наоборот.

§ 5.10. Искусственная энергонезависимость статических ОЗУ

Статические ОЗУ по своей природе энергозависимы — при снятии питания информация в триггерных запоминающих элементах теряется. Можно придать им искусственную энергонезависимость, причем с помощью разных методов.

Варианты с резервным источником питания

Этот вариант наиболее пригоден для ЗУ на элементах КМОП, т. к. они в режиме хранения потребляют чрезвычайно малую мощность.

Вариант с реакцией на изменение постоянного напряжения питания. Для подключения к накопителю ЗУ резервного источника питания разработчики памяти рекомендуют схему, приведенную на рис. 5.44, *а*. В этой схеме напряжение резервного источника немного ниже напряжения основного источника U_{CC}. В рабочем режиме накопитель питается от напряжения U_{CC}, при этом диод D1 проводит, а диод D2 заперт. При снижении рабочего напряжения включается диод D2, а диод D1

запирается, т. к. при малых значениях U_{CC} он попадает под обратное смещение. Так к накопителю автоматически подключается источник резервного питания.

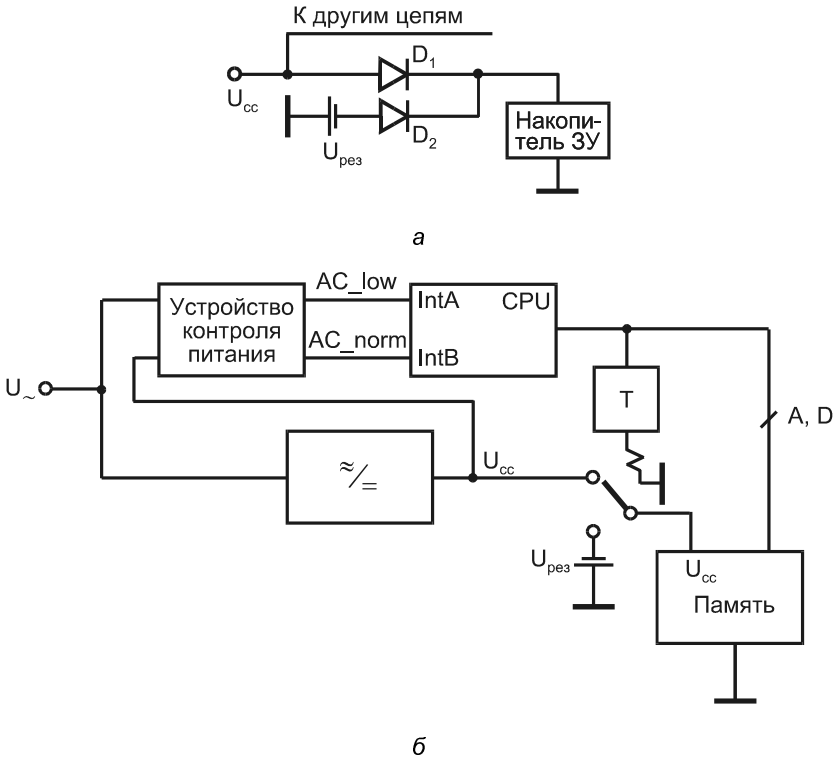


Рис. 5.44. Схемы подключения резервных источников питания к накопителям ЗУ

При использовании в микропроцессорных системах вариант ОЗУ, показанный на рис. 5.44, *а*, недостаточно надежен в связи со следующим обстоятельством. Напряжение питания системы U_{CC} вырабатывается схемой, на выходе которой обычно имеется сглаживающий фильтр со значительной инерционностью. Поэтому при аварии питания напряжение U_{CC} снижается относительно медленно. На начальном этапе этого процесса система продолжает работать, но в ее работе возможны ошибки. Желательно быстрее отреагировать на аварию питания, чтобы исключить указанную ситуацию.

Вариант с реакцией на изменение переменного напряжения питания. Ускорение реакции на аварию питания достигается с помощью специальной схемы подключения (рис. 5.44, *б*).

Здесь нарушение нормальной работы источника питания обнаруживается раньше, чем в предыдущей схеме, с помощью *контроля напряжения переменного тока* (AC — Alternate Current). Нарушение можно выявить за один-два периода переменного напряжения, *пока постоянное напряжение* U_{CC} *еще практически не изме-*

нилось. Признак нарушения AC_low служит запросом прерывания для процессора CPU. Получив запрос, процессор выполняет подпрограмму обслуживания прерывания A (Int A), в ходе которого передает содержимое своих регистров для временного хранения в стек накопителя и заканчивает подпрограмму установкой триггера T, что воздействует на обмотку реле, управляющего ключом. В результате память подключается к резервному источнику. При восстановлении нормального питания признак AC_norm вызывает свою программу обслуживания прерывания B (Int B), в ходе которой из стека возвращаются в процессор состояния регистров и сбрасывается триггер, что подключает память к основному источнику питания.

Память NV-SRAM

Память *NV-SRAM* (Non-Volatile SRAM) — вид "неразрушаемой" статической оперативной памяти, способной сохранять свое содержимое при снятии питания. Это свойство обеспечивается *добавлением к ОЗУ специального РПЗУ-ЭС (EEPROM)*. Такое репрограммируемое ПЗУ с электрическим стиранием (далее просто ПЗУ) служит для автоматического копирования в него данных из ОЗУ при снижении напряжения питания и обратной передачи данных в ОЗУ при восстановлении нормального уровня питающего напряжения. Получающаяся *комбинация двух типов памяти* дает сочетание SRAM, необходимого для скоростной оперативной работы с процессором, и постоянного ЗУ для хранения данных при снятии питания.

Микросхемы NV-SRAM выпускаются с 1996 г. В современных NV-SRAM запоминающие элементы ПЗУ фактически являются МНОП-транзисторами, а статическое ОЗУ выполнено на основе четырехтранзисторных триггерных запоминающих элементах (ЗЭ) с высокоомными нагрузками. Такое решение обеспечило как приемлемую плотность размещения запоминающих элементов на кристалле, так и достаточное быстродействие ОЗУ. В каждый элемент ОЗУ встраиваются два элемента ПЗУ для сохранения присущего триггерным ЗЭ дифференциальных сигналов и передач, дающих надежную работу ЗУ в широком диапазоне изменения температуры.

Данные копируются в ПЗУ параллельно во всех ЗЭ. При этом вначале ПЗУ, отключенное от ОЗУ, полностью очищается. Затем элементы ПЗУ и ОЗУ соединяются и происходит копирование данных в энергонезависимую память. Необходимые для стирания и программирования уровни напряжений вырабатываются встроенными преобразователями. В большинстве микросхем NV-SRAM реализован режим автоматического копирования данных в энергонезависимую память при снижении напряжения питания ниже заданного порогового уровня. Для копирования требуется около 10 мс и нужно, чтобы за это время напряжение питания снизилось не более чем на допустимую величину (приблизительно 0,7 В). Если это не соблюдается, то спад напряжения питания специально замедляется, например, включением на входе питания электролитического конденсатора с емкостью порядка 100 мкФ или более при развязке его от источника питания диодом с малым прямым сопротивлением (диодом Шоттки).

Обратно данные автоматически передаются параллельно при нормализации питания для всех элементов. Восстановление данных осуществляется за несколько сотен микросекунд после достижения напряжением питания заданного уровня. При необходимости процесс восстановления данных может быть инициирован программой. Некоторые микросхемы NV-SRAM имеют специальный вывод для аппаратной инициализации процесса сохранения данных путем подачи на этот вывод соответствующего напряжения. В рабочем режиме элементы ПЗУ отключены от оперативного ЗУ, и память работает как обычная SRAM.

§ 5.11. Статические ЗУ типа БиКМОП

В БиКМОП-схемотехнике стремятся объединить достоинства схем на основе биполярных приборов и МОП-структур. В ЗУ триггеры реализуются на схемах КМОП, а цепи выдачи данных, имеющие большую емкостную нагрузку, — на биполярной схемотехнике (ЭСЛ или ТТЛШ). Повышенная сложность изготовления БиКМОП-схем и их удорожание могут быть скомпенсированы их более высоким быстродействием, эффективной работой на длинные линии и другими факторами.

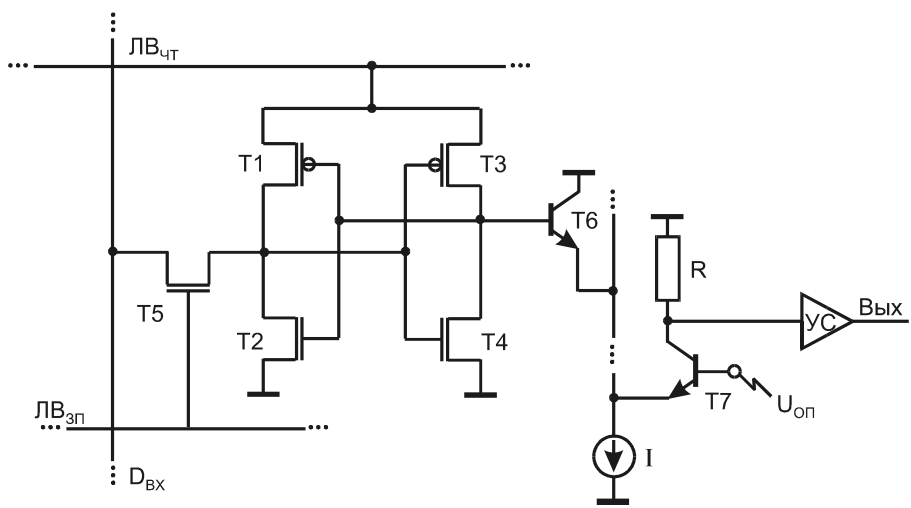


Рис. 5.45. Схема ячейки статического ЗУ в схемотехнике БиКМОП

В БиКМОП-ячейке двухпортового ЗУ (рис. 5.45) триггер построен на транзисторах T1—T4. Его выход подключен к базе биполярного транзистора T6, который совместно с опорным транзистором T7, общим для всех ячеек столбца, образует схему токового переключателя ЭСЛ, способного с большой скоростью коммутировать ток из одного плеча в другое (T6↔T7). Возможность быстро переключать нагруженные линии записи-считывания позволяет сохранить быстродействие на уровне, соответствующем внутренним схемам ЗУ, в которых КМОП-схемы работают в условиях малых нагрузок.

Ячейка имеет две линии выборки — для чтения ($ЛВ_{чт}$) и для записи ($ЛВ_{зп}$). Это позволяет записывать данные в невыбранные для чтения элементы одновременно со считыванием из других элементов (двухпортовая память).

Питанием ячейки служит потенциал линии $ЛВ_{чт}$. В отсутствие выборки для чтения этот потенциал невысок и любые переключения триггера не могут открыть транзистор Т6. Запись производится сигналом $D_{вх}$ при выборке ячейки по линии $ЛВ_{зп}$. Транзистор Т5 изготовляется как низкоомный, что позволяет ему диктовать состояние триггера.

Для чтения напряжение на линии $ЛВ_{чт}$ повышают на 0,55 В. Если триггер хранит единицу, то Т3 открыт, а Т4 заперт. При этом перепад напряжения на $ЛВ_{чт}$ передается на базу Т6, он открывается, и ток I переключается из транзистора Т7 в транзистор Т6. Напряжение на коллекторе Т7 повышается, что и служит сигналом чтения единицы для последующих каскадов усилителя считывания УС. Если триггер хранит логический нуль, то Т3 заперт и Т4 открыт. Ясно, что в этом случае перепад напряжения на линии $ЛВ_{чт}$ никак не повлияет на потенциал базы Т6, переключения тока I не возникнет и перепада выходного напряжения схемы не будет.

§ 5.12. Динамические запоминающие устройства — базовая структура

В динамических ЗУ (DRAM) данные хранятся в виде зарядов емкостей МОП-структур. Запоминающий элемент DRAM значительно проще триггерного, что позволяет разместить на кристалле намного больше ЗЭ (в 4...5 раз) и обеспечивает динамическим ЗУ максимальную информационную емкость. Конденсатор из-за токов утечки неизбежно теряет со временем свой заряд, и хранение данных требует их периодической регенерации (через каждые несколько миллисекунд).

Запоминающие элементы

В последнее время практически всегда применяют однотранзисторные ЗЭ — лидеры компактности (рис. 5.46).

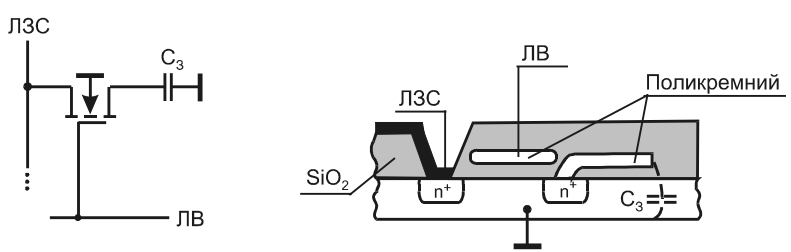


Рис. 5.46. Схема и конструкция запоминающего элемента динамического ЗУ

Ключевой транзистор отключает запоминающий конденсатор $C_З$ от линии записи-считывания ЛЗС или подключает его к ней. Сток транзистора не имеет внешнего вывода и образует одну из обкладок конденсатора. Другой обкладкой служит под-

ложка. Между обкладками расположен тонкий слой диэлектрика — оксида кремния SiO_2.

Примечание

Изображения плавающего затвора и затвора транзистора ЗЭ динамической памяти внешне сходны, но по существу они принципиально различны. Плавающие затворы не имеют связей с внешними цепями, а в динамическом ЗЭ такие связи есть, на затворы транзисторов подаются напряжения выборки строки, но это на рисунке просто не видно т. к. в разрезе показано только торцевое сечение длинной шины выборки, конец которой подключен к управляющему напряжению.

В режиме хранения транзистор заперт. При выборке ЗЭ на затвор подается напряжение, отпирающее транзистор. При этом запоминающая емкость подключается к линии записи-считывания ЛЗС и в зависимости от заряженного или разряженного состояния емкости различно влияет на потенциал линии. При записи потенциал ЛЗС передается на конденсатор, определяя его состояние.

Чтение состояния запоминающего элемента. Фрагмент ЗУ (рис. 5.47) показывает ЗЭ, усилитель считывания УС а также ключи К1 и К0 записи единицы и нуля. К линии записи-считывания ЛЗС подключено столько ЗЭ, сколько строк имеется в запоминающей матрице. Емкость ЛЗС $C_л$, в силу большой протяженности линии и большого числа подключенных к ней ЗЭ многократно превышает запоминающую емкость.

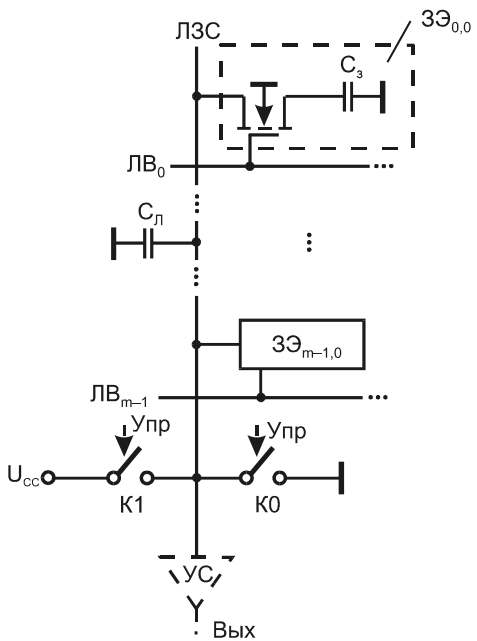


Рис. 5.47. Фрагмент схемы динамического ЗУ

Перед считыванием производится предзаряд ЛЗС. Имеются варианты ЗУ с предзарядом ЛЗС до уровня напряжения питания и до уровня его половины. Рассмотрим последний вариант. Итак, перед считыванием емкость $C_л$ заряжается до уровня $U_{cc}/2$. Будем считать, что хранение единицы соответствует заряженной емкости $C_з$, а хранение нуля — разряженной.

При считывании нуля к ЛЗС подключается емкость $C_з$, имевшая нулевой заряд. Эта емкость подключается параллельно $C_л$. Часть заряда емкости $C_л$ перетекает в емкость $C_з$, и напряжения на них уравниваются. Потенциал ЛЗС снижается на величину ΔU, которая и является сигналом, поступающим на усилитель считывания. При считывании единицы, напротив, напряжение на $C_з$ составляло вначале величину U_{cc} и превышало напряжение на ЛЗС. При подключении $C_з$ к ЛЗС часть заряда стекает с запоминающей емкости в $C_л$ и напряжение на ЛЗС увеличивается на ΔU. Графики сигналов при считывании нуля и единицы показаны на рис. 5.48.

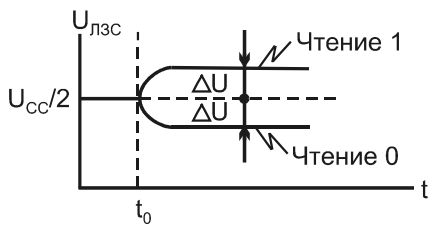


Рис. 5.48. Временные диаграммы сигналов при считывании данных в динамических ЗУ

Значение ΔU можно вычислить на основе анализа любого из процессов — считывания нуля или единицы. Рассмотрим считывание нуля. До выборки ЗЭ емкость ЛЗС имела заряд

$$Q = C_л U_{cc}/2.$$

После выборки ЗЭ этот же заряд имеет суммарная емкость $C_л + C_з$ и можно записать следующее соотношение:

$$Q = (C_л + C_з)(U_{cc}/2 - \Delta U).$$

Приравнивая выражения для одного и того же значения заряда Q, получим соотношение

$$C_л U_{cc}/2 = (C_л + C_з)(U_{cc}/2 - \Delta U),$$

из которого следует выражение

$$\Delta U = U_{cc} C_з/[2(C_з + C_л)] \approx U_{cc} C_з/2C_л.$$

В силу неравенства $C_з << C_л$ сигнал ΔU оказывается *слабым*. Кроме того, считывание является *разрушающим* — подключение запоминающей емкости к ЛЗС изменяет ее заряд.

Мерами преодоления отмеченных недостатков служат увеличение емкости $C_з$ (желательно без увеличения площади ЗЭ), уменьшение емкости ЛЗС и применение усилителей-регенераторов для считывания данных. Для увеличения $C_з$ разрабатывают новые диэлектрики, имеющие большую диэлектрическую постоянную, вводят в ЗЭ токоусиливающие структуры, что эквивалентно увеличению емкости ЗЭ и т. д.

Уменьшения емкости ЛЗС можно достичь "разрезанием" этой линии на две половины с включением дифференциального усилителя считывания в разрыв между половинами ЛЗС (рис. 5.49, *а*). Очевидно, что такой прием вдвое уменьшает емкость линий, к которым подключаются запоминающие емкости, т. е. вдвое увеличивает сигнал ΔU.

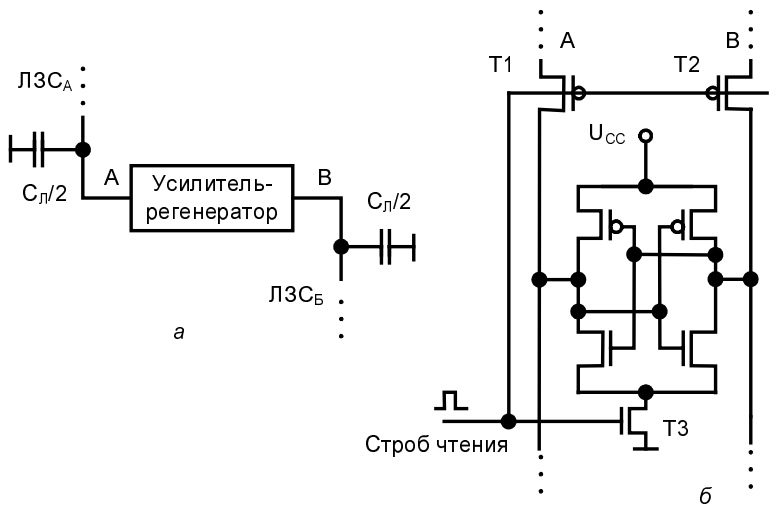


Рис. 5.49. Схема включения усилителя-регенератора в разрыв линии записи-считывания динамического ЗУ (*а*) и вариант схемной реализации усилителя-регенератора (*б*)

Усилители-регенераторы

Большинство усилителей-регенераторов строится на основе триггеров. Один из возможных вариантов (рис. 5.49, *б*) представляет собой триггер, дополненный ключевыми транзисторами Т1, Т2, Т3. При выборке ЗЭ в одной половине ЛЗС потенциалы в точках А и В становятся неидентичными, отличаясь на малую величину ΔU. Действительно, одна из половин ЛЗС, к которой не подключается $C_з$, сохраняет напряжение предзаряда $U_{cc}/2$, напряжение на другой половине, к которой подключается выбранный ЗЭ, отклоняется от напряжения предзаряда на ΔU в ту или иную сторону в зависимости от того, считывается единица или ноль. Транзисторы Т1 и Т2 включены, и на триггер поступают напряжения, действующие в точках А и В. Однако триггер не имеет питания (транзистор Т3 заперт), т. е. фактически схема триггера не функционирует.

Далее активизируется сигнал строб чтения, который открывает транзистор Т3, подавая тем самым питание на схему триггера, и запирает транзисторы Т1 и Т2, временно отключая усилитель-регенератор от линии считывания для предотвращения его влияния на выбранный ЗЭ. Оказавшийся под напряжением питания триггер должен перейти в одно из стабильных состояний. При этом состояние, в которое перейдет триггер, определяется несимметрией его режима, предварительно созданной при выборке ЗЭ, причем триггер, находящийся в неустойчивом состоянии, имеет высокую чувствительность к "перекосу", т. е. надежно воспринимает слабые сигналы считывания с линий ЛЗС, которые и определят переход в нужное стабильное состояние. На выходах триггера сформируются напряжения высокого и низкого уровней. Так как одни и те же точки А и В являются одновременно и входами и выходами усилителя-регенератора, после своего срабатывания и окончания строба чтения усилитель подключается к ЗЭ и восстанавливает на емкости C_3 полное значение считанного сигнала. Тем самым данные в ЗЭ автоматически регенерируются, иными словами *чтение данных сопровождается их регенерацией*. Состояние триггера определяет также сигналы, выводимые во внешние цепи в качестве считанной информации.

Мультиплексирование шины адреса

Особенность почти всех динамических ЗУ — мультиплексирование шины адреса. Адрес делится на две части — адрес строки и адрес столбца матрицы ЗЭ. Эти адреса подаются на одни и те же выводы ИС поочередно. Адрес строки сопровождается стробом RAS (Row Address Strobe), а адрес столбца — стробом CAS (Column Address Strobe). Мультиплексирование шины адреса уменьшает число выводов микросхемы на половину разрядности адреса и тем самым удешевляет ее. Сокращение числа выводов корпуса для динамических ЗУ особенно актуально, т. к. они имеют максимальную емкость и, следовательно, большую разрядность адресов. Мультиплексированию адресов способствует и то, что адреса строки и столбца в некоторых режимах используются различно (например, в режимах регенерации адрес столбца вообще не нужен).

Внешняя организация и временные диаграммы

На рис. 5.50 показаны внешняя организация и временные диаграммы динамического ОЗУ.

Циклы обращения к ЗУ начинаются сигналом $\overline{\text{RAS}}$ и запаздывающим относительно него сигналом $\overline{\text{CAS}}$. Отрицательным фронтам этих сигналов соответствуют моменты восприятия адресов строки и столбца соответственно. Согласно указанию выполняемой операции (сигналу R/W) либо вырабатываются выходные данные DO, либо принимаются входные данные DI. В циклах регенерации подаются только импульсные сигналы $\overline{\text{RAS}}$ и адреса строк. Области безразличных значений сигналов на рисунке затемнены.

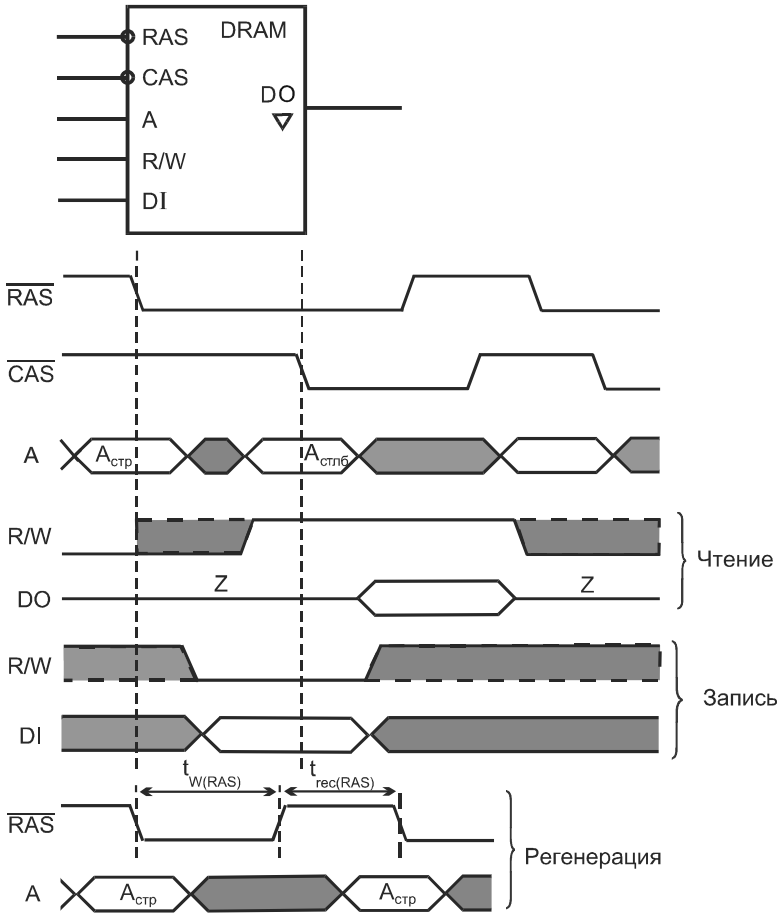


Рис. 5.50. Внешняя организация и временные диаграммы динамического ЗУ

Схема динамического ЗУ

Схема динамического ЗУ представлена на рис. 5.51 двумя частями. На рис. 5.51, *а* приведена схема выработки сигналов, поступающих далее в матрицу, изображенную на рис. 5.51, *б*. В схеме (*а*) выполняются следующие действия. При обращении к ЗУ активизируется строб $\overline{\text{RAS}}$, запускающий формирователь ФТС 1, последовательно вырабатывающий пару сигналов Ф1 и Ф2. Сигнал Ф1 разрешает загрузку регистра Рг Х адресом строки и работу дешифратора ДШ Х, одна из выходных линий которого (линий выборки строки ЛВ) соответственно адресу возбуждается и выбирает все ЗЭ строки, адрес которой содержится в регистре Рг. Второй тактирующий сигнал Ф2 подает строб чтения на усилители-регенераторы, которые под его воздействием активизируются и срабатывают, считывая выбранную строку.

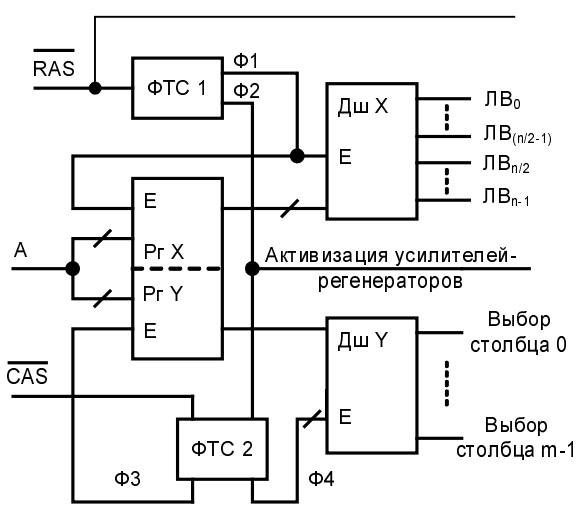
Для последующих операций чтения или записи требуется сигнал $\overline{CAS}$, разрешающий формирователю ФТС 2 выдачу второй пары тактирующих сигналов Ф3 и Ф4. Сигнал Ф3 загружает в Рг Y адрес столбца, а Ф4 активизирует дешифратор ДШ Y, вследствие чего возбуждается линия выбора столбца, соответствующая принятому адресу, открываются ключи 2 выбранного столбца.

В матрице ЗУ (рис. 5.51, *б)* в исходном состоянии (до обращения к ЗУ) сигнал $\overline{RAS}$ пассивен, т. е. имеет высокий уровень, который замыкает ключи К1 и подает напряжение $U_{cc}/2$ на полушины записи-считывания $ЛЗС_а$ и $ЛЗС_в$ для их предзаряда. При обращении к ЗУ одновременно с подачей адреса строки активизируется сигнал $\overline{RAS}$, ключи К1 размыкаются и линии записи-считывания изолируются от источника напряжения $U_{cc}/2$, становясь готовыми к процессам чтения или записи данных.

В разрывы между секциями $ЛЗС_а$ и $ЛЗС_в$ включены усилители-регенераторы У-Р, для которых подключение ЗЭ создает дисбаланс входных сигналов. Под воздействием "перекоса" после активизации каждый У-Р придет в состояние, диктуемое хранимой в подключаемом ЗЭ информацией, формируя в своих точках входов-выходов полные уровни сигналов, что восстанавливает состояния ЗЭ выбранной строки, поскольку эти ЗЭ подключены к выводам усилителей-регенераторов.

Из содержащейся в усилителях-регенераторах строки линия выборки столбца, открывая свои ключи К2, пропускает далее на ключи К3 адресованный бит.

В зависимости от сигнала R/W, линии ЛЗС подключаются либо к выходной шине данных (через ключи К4 при R/W = 1, т. е. при чтении), либо к линии входных данных (через ключи К3 при R/W = 0, т. е. при записи). Данные, записываемые в разные секции линий ЛЗС, задаются инверсными значениями, поскольку воздействуют на разные входы дифференциального усилителя-регенератора.



а

Рис. 5.51, *а*. Схема динамического ЗУ *(начало)*

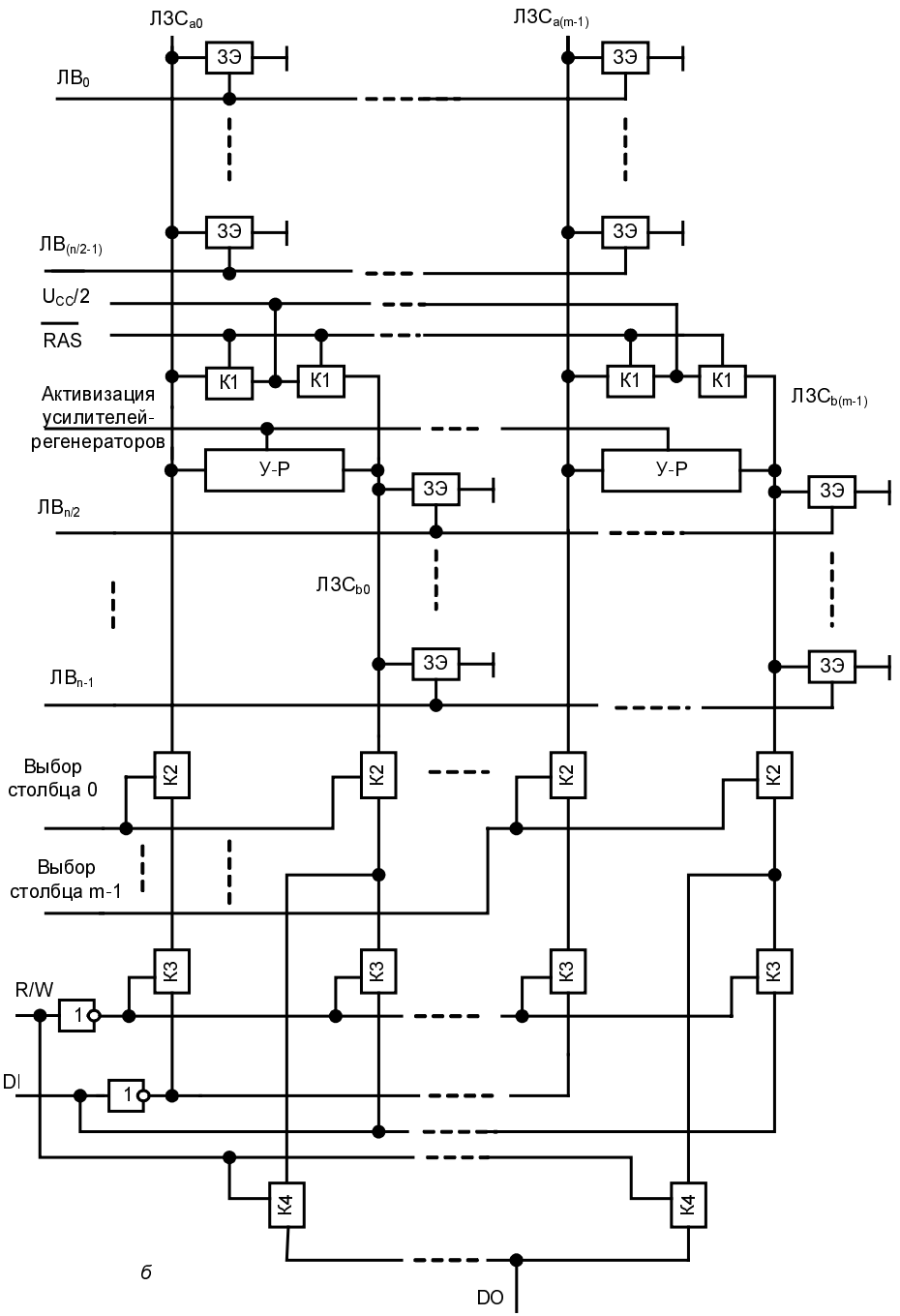


Рис. 5.51, *б*. Схема динамического ЗУ *(окончание)*

Для операции регенерации, целиком проходящей внутри ЗУ, связь с внешними выводами не требуется, поэтому для нее достаточно подачи только сигнала $\overline{\text{RAS}}$ (со-

вместно с адресами регенерируемых строк) и выработки только тактирующих сигналов Ф1 и Ф2.

Кроме режимов записи и считывания, в динамических ЗУ иногда организуют дополнительные режимы, в частности, *режим "считывание-модификация-запись"*. В этом режиме в одном цикле слово считывается и вновь записывается по тому же адресу, но может быть изменено (модифицировано). Такой режим используется в ЗУ с коррекцией ошибок, например, с применением кодов Хемминга. В этом случае слово с контрольными разрядами считывается, проверяется контрольной схемой и при необходимости исправляется и вновь записывается по старому адресу. Длительность цикла режима "считывание-модификация-запись" больше циклов записи и считывания, но меньше их суммы, поэтому время на коррекцию содержимого ЗУ сокращается.

§ 5.13. Динамические запоминающие устройства повышенного быстродействия

Высокое быстродействие процессоров требует и высокой скорости работы ОЗУ. Предложен ряд вариантов динамических ОЗУ повышенного быстродействия. Методы ускорения работы, использованные в этих ОЗУ, можно разделить на две группы. Методы первой группы *основаны на предположении о кучности адресов обращения к ОЗУ*, справедливом, когда адреса последующих обращений вероятнее всего расположены близко к адресу текущего обращения. Если такое предположение не оправдывается, то методы первой группы становятся неэффективными и основанные на них ЗУ работают с той же скоростью, что и обычные стандартные архитектуры. Методы второй группы *дают ускорение работы памяти даже при произвольном доступе к ее ячейкам*, когда адреса предыдущих и последующих обращений к памяти никак не взаимосвязаны.

Разработка DRAM повышенного быстродействия началась с освоения методов первой группы и лишь затем появились ЗУ, эффективные при произвольном доступе к хранимым данным, а также микросхемы с элементами комбинирования методов обеих групп, в которых реализуются в основном методы первой группы, но одновременно с этим приняты меры к сокращению времени доступа к первому слову пакета, так что схемы стали более эффективными и при произвольном доступе.

FPM, EDORAM, BEDORAM

FPM

FPM (Fast Page Mode) — это *быстрый страничный доступ*, эффективный, если после текущего обращения к памяти следующее будет к ЗЭ той же строки матрицы. Сравним такую ситуацию с обращением по произвольному адресу. При чтении по произвольному адресу сначала в матрице выбирается строка, затем — столбец.

При этом сначала требуется перезарядить шину выборки строки, а затем шину выборки столбца, что сопровождается соответствующими задержками.

При выборке строки (страницы) происходят процессы, связанные с заданием исходного состояния схемы и затем активизацией строба RAS — предзаряд линий считывания, подключение к ним запоминающих элементов строки и срабатывание усилителей-регенераторов (см. описание работы базовой схемы ЗУ). После завершения перечисленных процессов ЗУ готово к выполнению очередных фаз, связанных с активизацией строба CAS и выборкой столбца.

При обращении к данным в пределах одной страницы изменяются только адреса столбцов в сопровождении строба CAS. Изменяет состояние фактически только группа ключей 3 и 4 (см. рис. 5.51, *б*). Пока не изменился номер страницы, в циклах обмена исключены первые этапы, что сокращает длительность циклов.

Из временных диаграмм для FPM (рис. 5.52) видно, что время доступа к данным при неизменности адреса строки RA сокращается в сравнении со временем доступа при полном цикле. Пропорциональность времен первого и последующих обращений к ЗУ можно записать следующим образом: 5-3-3-... .

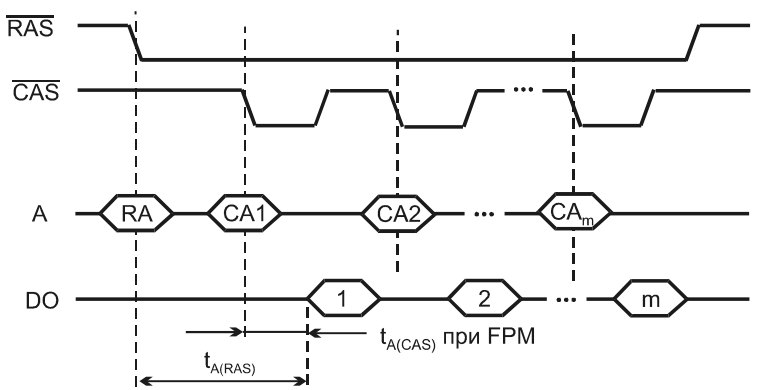


Рис. 5.52. Временные диаграммы режима FPM динамических ОЗУ

В компьютерной технике FPM DRAM пришли на смену стандартным в середине 90-х гг. XX в. FPM — начало развития методов повышения быстродействия DRAM, возможности которого сейчас намного превышены более поздними вариантами.

Дополнительные средства для организации FPM просты: требуется лишь проверять принадлежность очередного адреса текущей странице (строке), позволяющую выполнять цикл страничного режима. В противном случае выполняется полный цикл. ОЗУ типа FPM обеспечивали времена обращения к ЗУ 30...40 нс, что допускало их работу с процессорными шинами на тактовой частоте до 33 МГц.

EDORAM

EDORAM (Enhanced Data Out RAM) — это ОЗУ с усовершенствованным выводом данных, близкие к структурам FPM и эффективные при кучности адресов. В EDORAM

данные в усилителях-регенераторах не сбрасываются по окончании строба $\overline{\text{CAS}}$. При этом как бы появляется статический регистр, хранящий строку. При обращениях в пределах строки (страницы) используется чтение данных из регистра, т. е. из быстродействующей статической памяти. По-прежнему используется только сигнал $\overline{\text{CAS}}$, но длительность его может быть сокращена в сравнении с режимом FPM. Это увеличивает быстродействие ЗУ. Для EDORAM пропорциональность времен первого и последующих обращений к словам в пределах страницы характеризуется цифрами 5-2-2-... . EDORAM допускали работу на частотах до 50 МГц и получили в свое время широкое распространение, в частности, из-за тесной преемственности с FPM, замена которых на EDORAM требует лишь небольших изменений в схеме и синхросигналах ЗУ.

BEDORAM

В структуру BEDORAM (Burst EDORAM) с пакетным доступом, эффективную при кучности адресов, вводится счетчик адресов столбцов. При обращении к группе слов (пакету) адрес столбца формируется обычным способом только в начале пакетного цикла. Для последующих слов адреса образуются быстро инкрементированием счетчика. Характерный для пакета из 4 слов тайминг: 5-1-1-1. Память BEDORAM не получила широкого распространения из-за появления сильного конкурента — синхронных DRAM (SDRAM), в которых достигается не только доступ к словам внутри пакета с единичным интервалом времени (т. е. в каждом такте), но и сами длительности такта также сокращаются.

SDRAM и DDR SDRAM

Хотя переход от базовой структуры DRAM к архитектурам FPM, EDORAM и BEDORAM повысил быстродействие памяти, это оказалось недостаточным для компьютеров, графических систем и т. д. Более высокую производительность обеспечили микросхемы синхронной памяти *SDRAM* (Synchronous DRAM) и *RDRAM* (Rambus DRAM) в том числе и с интерфейсами DDR, DDR2, DDR3.

В SDRAM применяются:

- тесная увязка операций с тактирующими сигналами процессора;
- буферизация (защелкивание) адресов и данных;
- многобанковые структуры;
- пакетные режимы;
- конвейеризация тракта продвижения информации.

Первые две из перечисленных черт определяют принадлежность памяти к классу SDRAM.

Достоинства синхронной памяти. При работе с асинхронными ОЗУ после выдачи адреса и управляющих сигналов процессор *ждет выполнения памятью внутрен-*

них операций, сохраняя на системных шинах адреса и сигналы управления (и данные, если речь идет о записи). Ожидание снижает производительность системы. В синхронных ЗУ ввод и вывод данных контролируются системными тактами, а выдаваемые процессором адреса, сигналы управления и данные *фиксируются* (защелкиваются) в регистрах. После этого (при выполнении внутренних операций) системные шины памятью не используются. Заранее известно, через какое число тактов память ответит своей готовностью. Во время работы памяти процессор может выполнять другие операции. Пассивное ожидание исключается.

К достоинствам SDRAM относится и отсутствие *для системотехника* проблем по временному согласованию входных сигналов, что в иных случаях может быть сложным. Здесь же это согласование уже выполнено при разработке ЗУ.

Эффективность многобанковости, пакетного обмена, конвейеризации трактов продвижения информации и применения интерфейсов типа DDR рассмотрена ранее.

Командные слова. Характерная черта SDRAM — использование *наборов управляющих сигналов в качестве команд.* Внешние управляющие сигналы фиксируются фронтами тактовых импульсов в *регистре управления*. Сочетание нескольких сигналов (RAS, CAS, WE, CS,…) трактуется как *командное слово*, которое декодируется внутренним автоматом и задает набор и последовательность внутренних сигналов управления.

Регистр программирования. Регистр программирования и связанная с ним логика управления позволяют специализировать параметры памяти применительно к требованиям конкретной системы (изменять длину пакета, регулировать латентность и др.).

Интерфейсы DDR. Память DDR SDRAM (Double Data Rate SDRAM) отличается от SDRAM лишь характером тактирования процессов ввода/вывода данных. Ввод/вывод тактируется обоими фронтами синхроимпульсов, и это удваивает пропускную способность интерфейса. Во внутренних блоках микросхем DDR SDRAM тактирование ведется обычным способом (по одному фронту), а пропускная способность внутренних блоков и ввода/вывода согласуются с помощью разной ширины шин. Синхронизация по технологии DDR при высоких тактовых частотах предъявляет жесткие требования к точности временных диаграмм, для поддержания которой принимаются специальные меры:

- тактовые сигналы генерируются и передаются в дифференциальной форме (имеются одновременно сигналы CLK и $\overline{\text{CLK}}$);
- в интерфейс вводится специальный сигнал DQS (Data Queue Strobe), вырабатываемый самим источником данных (памятью при чтении, контроллером памяти при записи), фронты которого центрируются различным образом (по фронту или центру импульсов) для чтения и записи данных;
- для автоподстройки задержек между синхросигналами в состав микросхем вводятся блоки DLL.

Структура DDR SDRAM. На рис. 5.53 приведена структура DDR SDRAM фирмы Samsung емкостью 128 Мбит, в варианте организации 16M × 8.

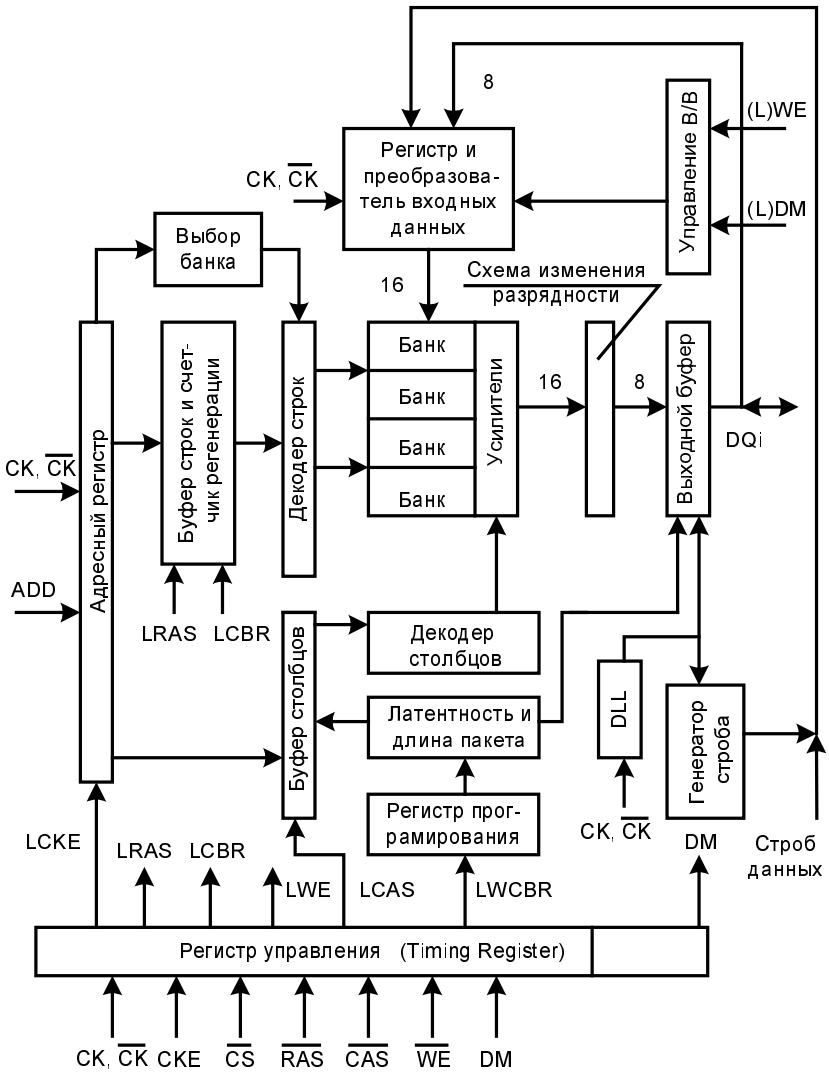


Рис. 5.53. Структура микросхемы DDR SDRAM

Микросхема имеет 4 банка. Емкость банка 32 Мбита при организации 2M × 16. В адресе выделяются три части — двухразрядное поле выбора банка и поля адресов строки и столбца, поступающие на декодеры через буферные схемы. В буфере адреса строк размещен также счетчик адресов, используемый при регенерации данных для перебора строк в режиме чтения. Входные данные принимаются регистром со схемой изменения разрядностей, которая передает на блок банков данные удвоенной разрядности для согласования пропускной способности DDR и внутренних областей памяти

с обычным тактированием, при котором информация воспринимается только по положительным фронтам тактовых импульсов. На выходе блока банков данные, снимаемые с усилителей, подвергаются обратному изменению разрядности в связи с переходом из области обычного тактирования в область DDR. Передачи данных стробируются сигналом DQS (на схеме обозначен как "Строб данных"). Для повышения качества тактовых сигналов использован блок DLL.

Внешние управляющие сигналы поступают на регистр управления (Timing Register), который правильнее было бы называть автоматом управления. Набор внешних сигналов определяет выполняемую команду. Содержащаяся в наборе сигналов информация используется для выработки определенной последовательности внутренних сигналов управления (их обозначения начинаются с буквы L), обеспечивающих выполнение требуемых действий.

Примеры команд: установка регистра программирования, активизация банка и адрес строки, чтение и адрес столбца, запись и адрес столбца, саморегенерация, предзаряд, режим снижения мощности и т. д. Всего микросхема использует 18 команд.

Интерфейсные сигналы имеют следующий смысл:

- CK и $\overline{CK}$ — (Clock) дифференциальные тактовые сигналы. Адреса и сигналы управления воспринимаются блоками микросхемы по положительным фронтам CK и, соответственно, отрицательным фронтам $\overline{CK}$. Данные — по обоим фронтам. Внутренние синхросигналы вырабатываются из CK и $\overline{CK}$;
- CKE — (Clock Enable) разрешает или запрещает внутренне тактирование и функционирование входных и выходных буферов. Снятие тактирования обеспечивает проведение операций предзаряда (Precharge), снижения мощности (Power-Down) и саморегенерации (Self Refresh) или некоторых других действий;
- $\overline{CS}$ — разрешает или запрещает декодирование команд, при высоком уровне этого сигнала все команды маскируются, сигнал является частью командного кода;
- $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ вместе с $\overline{CS}$ определяют вводимую команду;
- DM — (Data Masked) маскирует входные сигналы при записи данных, при чтении его состояние безразлично;
- ADD ($A_{11}...A_0$) — адресные входы, задающие адрес строки при командах ACTIVE и адрес столбца и бит предзаряда при командах READ/WRITE, чтобы выбрать ячейку памяти в матрице соответствующего банка. Один из разрядов адреса используется при командах предзаряда для определения того, относится эта команда к одному или всем банкам. При выполнении команды Mode Register Set на адресные входы поступает код операции;
- DQ_i — шина ввода/вывода данных;
- Строб данных (DQS) — выходной сигнал при чтении данных и входной при их записи. При чтении согласован с данными по фронтам, при записи отмечает центры битовых интервалов. Служит для захвата записываемых данных.

Кроме сигнальных выводов микросхемы имеют по три вывода для общего питания 2,5 В и схемной земли, по пять выводов для питания 2,5 В и схемной земли цепей передачи данных и для опорного напряжения 1,25 В, соответствующего стандарту SSTL-2 (Stub Series Terminated Logic).

Рабочие частоты микросхем семейства лежат в диапазоне от 100 до 166 МГц. Латентность при чтении — 2 или 2,5 периода тактовых сигналов, длины пакетов 2, 4 или 8, типы пакетов — последовательный или с перемежением адресов (Interleave).

При работе с тактовыми частотами 100, 133 и 166 МГц микросхемы DDR маркируются как DDR200, DDR266 и DDR330.

RDRAM

Микросхемы RDRAM (Rambus DRAM) названы по имени фирмы Rambus. Основное новшество в их структуре — *специальный интерфейс Rambus Channel* с 16-разрядной шиной данных (в первых вариантах с 8-разрядной). Сужение шин отвечает современной тенденции реализации трактов высокого быстродействия.

Среди высокопроизводительных динамических ЗУ сейчас доминируют SDRAM, поскольку более широкий круг потребителей выбирает их в силу меньшей стоимости. Память RDRAM лучше всего приспособлена к графическим и мультимедийным задачам с типичным для них процессом — быстрой выдачей длинной последовательности слов (для формирования изображения на экране и т. п.). Именно эти приложения и составляют наиболее перспективную область применения RDRAM.

В микросхемах RDRAM, как и в SDRAM, используется технология DDR, многобанковость, конвейеризация и др. В сущности, в RDRAM сочетаются "ядро", сходное с применяемыми в других DRAM, и *специфичный интерфейсный блок*, позволяющий передавать в это ядро информацию с очень высокой производительностью.

Число линий в Rambus Channel значительно меньше, чем у традиционных интерфейсов памяти. В нем нет специализированных адресных линий. По интерфейсу *посылаются пакеты*, включающие в себя команды и адреса. Вначале посылается пакет запросов, на который память отвечает пакетом подтверждения, после чего идет пакет данных. Из-за такого процесса первый доступ к данным оказывается сильно запаздывающим, поэтому при чтении малых пакетов RDRAM неэффективна.

Сейчас RDRAM работают преимущественно на тактовой частоте 400 МГц, что при использовании DDR соответствует "эффективной частоте" 800 МГц. Поскольку ширина шины равна 16 разрядам, пропускная способность канала составляет 1600 Мбайт/с. В одной подсистеме памяти может быть до 4 каналов (таковы возможности современных контроллеров RDRAM). Таким образом, общая пропускная способность может быть доведена до 6,4 Гбайт/с. Хотя в сравнении с первыми образцами современные RDRAM имеют сниженную латентность, их эффективность при произвольном доступе к малым объемам данных все же значительно снижается.

Если, например, при темпе передач байтов 800 МГц (1,25 нс на байт) запаздывание составляет 480 нс, то при обмене пакетами по 256 байт средняя частота передач составит около 320 МГц (к 1,25 нс добавляется приблизительно 1,9 нс на байт), при пакетах по 64 байта — около 110 МГц и т. д.

Состав микросхем памяти RDRAM

В состав памяти RDRAM (рис. 5.54, *а*) входят:

- контроллер;
- каналы;
- микросхемы, содержащие матрицы запоминающих элементов.

Со стороны системной шины контроллер принимает обычные для работы с памятью сигналы, а со стороны канала реализуется интерфейс Rambus Channel, так что учет специфики этого интерфейса контроллер берет на себя.

Структура канала

Канал — экономная по топологии последовательно-параллельная шина с небольшим числом линий, ориентированная на упрощение разводки проводников для получения максимальных частот передачи сигналов. Структура канала показана на рис. 5.54, *б*. В шине применяются дорогостоящие сверхбыстродействующие интерфейсные схемы и технология DDR.

В канале имеются 30 основных линий в стандарте *RSL* (Rambus Signaling Level) и 4 вспомогательных в стандарте КМОП. Двухпроводные линии CTM (Clock To Master) и CFM (Clock From Master) передают дифференциальные синхросигналы, к которым "привязана" информация, поступающая от контроллера к микросхемам или наоборот. Группы линий ROW, COL и DQA, DQB рассматриваются как шины строк, столбцов и данных (двухбайтная шина данных DQA, DQB может иметь 16 или 18 разрядов с учетом разряда контроля по четности). По этим шинам информация передается пакетами, занимающими по 8 интервалов синхронизации (по 4 такта), так что объем пакета строк 24 бита, столбцов — 40 бит и данных 16 байт (восьми- или девятиразрядных). Инициализация схемы выполняется с помощью последовательного КМОП-интерфейса с линиями данных SIO_0 и SIO_1 (Serial Input/Output), линией синхронизации SCK (Serial Clock) и линией команд CMD (Command). При инициализации каждой микросхеме канала (их может быть до 32) присваивается свой адрес DEVID. По основным связям микросхемы канала соединены параллельно, по линии последовательного интерфейса — в цепочку.

В высокочастотном интерфейсе RSL амплитуда сигнала 0,8 В, волновые сопротивления в концах линий, удаленных от контроллера, согласованы резисторами-терминаторами, в началах (у контроллера) линии практически разомкнуты, и коэффициент отражения близок к единице. Сигналы, посылаемые контроллером, попадают в согласованную линию, не создающую отраженных волн. Микросхемы,

посылая сигналы контроллеру, формируют сигналы амплитудой 0,4 В. На выводах контроллера сигналы благодаря отраженной волне удваиваются и достигают требуемой амплитуды 0,8 В. Повторных отражений не будет, поскольку на дальнем конце линий сопротивления согласованы.

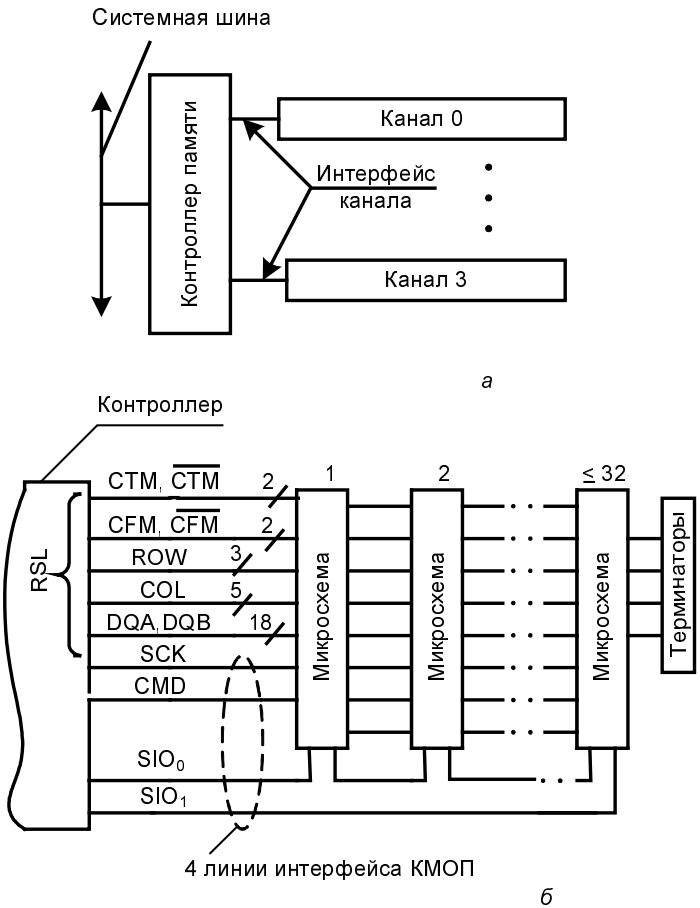


Рис. 5.54. Состав памяти RDRAM (*а*) и структура одного канала (*б*)

Связь канала с микросхемами и их совместная работа

Стыковка высокочастотного канала с внутренними цепями схемы показана упрощенно на рис. 5.55.

Через линии $SIO_{1\text{-}0}$ последовательного интерфейса, тактируемые сигналами SCK и CMD, информация для инициализации схемы, управления потребляемой ею мощностью и некоторых других целей вводится в регистры управления, из которых

микросхемы получают пятиразрядные назначаемые адреса, сведения о регенерации строк и режимах понижения мощности. Необходимые сведения поступают также в блоки пакетных декодеров и схем выработки команд.

Три линии управления доступом к строке ROW_{2-0} и пять линий управления доступом к столбцу COL_{4-0} образуют шину для передачи в микросхему пакетов управляющей и адресной информации, необходимой для осуществления транзакций. Блоки формируют команды ACT, PRER, PREX, PREC, RD, WR, организующие работу ядра.

Для внутренней синхронизации передач и приема данных из дифференциальных синхросигналов CTM и CFM формируются однополюсные сигналы TCLK и RCLK. Две 9-разрядные шины служат для приема/выдачи в канал байтов A и B (в байте восемь информационных разрядов плюс контрольный, который можно использовать и как информационный).

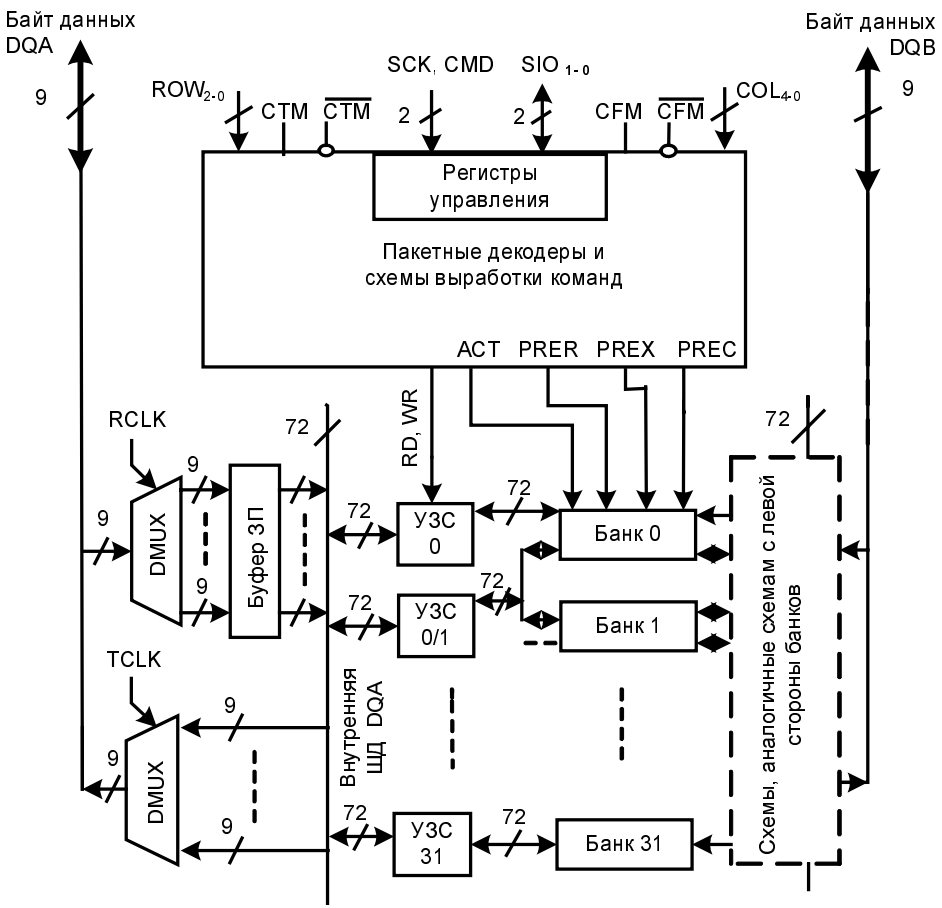


Рис. 5.55. Упрощенная схема согласования ядра и канала в RDRAM

Для упрощения на рис. 5.55 изображена лишь часть ядра, а именно банки и левая сторона, обслуживающая передачи байта A в ядро и из него. Для обслуживания передач байта B справа от банков имеются схемы, зеркально повторяющие схемы левой стороны ядра (на рисунке не раскрыты).

В ядро микросхемы входят 32 банка памяти, 34 усилителя записи/считывания УЗС и две 72-разрядные внутренние шины данных для байтов A и B. УЗС состоят из двух частей — слева от банка и справа от него (на рисунке видна только левая часть).

Банк имеет емкость 1 Мбайт, содержит 512 строк, состоящих из 128 16-байтовых наименьших адресуемых единиц информации (Dualocts). Каждый УЗС имеет 1 Кбайт быстродействующей регистровой памяти (по 512 байт для левой и правой частей УЗС). Поскольку емкость строки равна 2 Кбайтам, УЗС может принимать данные половины строки (любую из 1024 полустрок банка, связанного с этой строкой). Равным образом из УЗС можно передавать данные в любую полустроку ассоциированного со строкой банка. Почти все УЗС (кроме 0, 15, 16 и 31-го) находятся в совместном пользовании двух соседних банков.

Команда ACT (Activate) загружает одну из 512 строк выбранного банка в УЗС, связанные с этим банком. Команда предзаряда PRER заставляет выбранный банк освободить связанную с ним пару УЗС, позволяя активизироваться другой строке этого или смежного банка. Команды PREC и PREX реализуют другие механизмы операции предзаряда. Команда RD передает на выводы DQA/DQB из УЗС одну из минимально адресуемых единиц информации. Команда WR вызывает загрузку полученной из канала по шинам DQA/DQB минимально адресуемой единицы информации в буфер записи (в буфере имеется место и для некоторых сведений об адресах банка и столбца). Наличие буфера записи уменьшает задержку при реверсе внутренней шины данных DQA/DQB.

Взаимодействие ядра микросхемы с быстродействующими шинами DQA/DQB происходит следующим образом. Ядро работает на частоте в 1/8 от частоты шин DQA/DQB. При этом данные в УЗС или из них поступают порциями по 72 разряда на каждую из двух внутренних шин данных или от них (по 8 девятиразрядных байтов). В интервале между передачами 72-разрядные данные с помощью мультиплексоров разбиваются на 8 байтов, которые на частоте, в восемь раз более высокой, успевают последовательно пройти по малоразрядным шинам DQA/DQB. При другом направлении передач с помощью демультиплексоров из байтов собираются 72-разрядные данные, которыми на частоте работы ядра УЗС обмениваются с банками.

CDRAM

В структурах CDRAM (Cached DRAM, кэшированная DRAM) на одном кристалле с DRAM размещена статическая кэш-память (кэш первого уровня). При этом кэш обеспечивает быстрый обмен с процессором, если информация находится в кэше, а

также быстрое обновление своего содержимого. Последняя возможность связана с тем, что *размещение кэша на одном кристалле с DRAM* делает связи между ними внутренними (реализуемыми внутри кристалла), а в этом случае разрядность шин может быть очень большой и обмен может производиться большими блоками данных. Как синоним обозначения CDRAM иногда используется обозначение EDRAM (Enhanced DRAM). Кэширование, как и всегда, эффективно при выполнении программ, для которых промахи относительно кэша достаточно редки.

Ускорение произвольного доступа

Вторая группа методов повышает быстродействие динамических ЗУ *даже при произвольном доступе к ее ячейкам*, когда адреса предыдущих и последующих обращений к памяти произвольны. Добиться высокого быстродействия при произвольном доступе труднее, чем при кучности последовательно поступающих адресов. Так как в компьютерных приложениях кучность адресов проявляется достаточно четко, для них целесообразно применять SDRAM, DDR SDRAM, RDRAM. Для алгоритмов, реализуемых в сетевых приложениях и многих мультимедийных задачах, важно быстродействие при произвольном доступе. Динамические ЗУ с ускоренным произвольным доступом, разработанные в последнее время, образуют как бы параллельную линию развития микросхем памяти — линию "некомпьютерных" ЗУ. Среди таких ЗУ значительная доля предназначена для портативной аппаратуры с батарейным питанием, например, для мобильных телефонов. Поэтому при их разработке одновременно с методами повышения быстродействия особенно активно применяют разнообразные методы снижения потребляемой мощности.

ЗУ с блочной структурой

В *§ 5.2* показано, что фрагментирование ЗУ большой емкости дает возможность повысить его быстродействие и уменьшить потребляемую им мощность. Этот подход успешно реализован и в технике DRAM с целью повышения их быстродействия при произвольном доступе.

В DRAM с блочной структурой (Embedded DRAM) цикл произвольного доступа существенно короче, чем в DRAM базовой структуры с плотно упакованными матрицами, ориентированных на минимизацию площади кристалла и стоимости микросхем. Число ячеек, подключаемых к каждой разрядной линии (в современных DRAM обычно до 512), обычно ограничивается возможностью получить амплитуды сигналов чтения, достаточные для их надежного восприятия. Длина словарных линий в обычных DRAM также велика, поскольку приемлема с точки зрения нагрузки на источники сигналов управления этими линиями (WL-драйверы).

В блочных DRAM матрица разбивается на части, при этом словарные и разрядные линии укорачиваются, уменьшаются их паразитные емкости и сопротивления, а

следовательно, и задержки переключений. Таким способом удается существенно повысить быстродействие DRAM, не требуя при этом кучности адресов обращения к ним. Платой за это является усложнение схемы ЗУ и соответствующее ему увеличение стоимости. Если в базовой структуре непосредственно под запоминающие элементы удается отвести около 50% площади кристалла, то для блочных DRAM этот процент снижается приблизительно до 35.

Укорачивание разрядных линий ведет и к увеличению амплитуды считываемых сигналов, линейно зависящих от отношения емкости запоминающего конденсатора к емкости разрядной линии. Рост амплитуды считываемых сигналов в свою очередь способствует ускорению работы усилителей считывания. В настоящее время реализованы блочные конвейеризованные DRAM с произвольным доступом, уступающие по быстродействию лишь лучшим статическим ЗУ. В то же время уровень интеграции фрагментированных DRAM более чем в 5 раз превышает возможности статических ЗУ.

RLDRAM

RLDRAM (Reduced Latency DRAM) — это DRAM с уменьшенной длительностью *полного цикла* обращения к памяти, что и требуется для ускорения произвольного доступа. Быстрый произвольный доступ в RLDRAM был достигнут благодаря *ускорению реверса шины* (см. *§ 5.9*) и *передаче сразу всего адреса одним тактирующим фронтом* (отказу от последовательных передач в память полуадресов по стробам RAS и CAS, т. е. от мультиплексирования адреса, применяемого для упрощения и удешевления ЗУ). В RLDRAM первоначально был реализован другой выбор — усложнение ради ускорения. Получение сразу всего адреса позволяет организовать процессы чтения/записи с более выраженным параллелизмом действий, чем при последовательном поступлении полуадресов. Однако отказ от мультиплексирования адреса не получил продолжения и во втором поколении RLDRAM II (2005 г.) используется адресная схема с мультиплексированием и интерфейсом DDR 2.

Разработчик RLDRAM, фирма Infinion, получила время произвольного доступа при чтении 25 нс (приблизительно вдвое меньшее, чем у быстродействующей компьютерной памяти типов DDR SDRAM и DRDRAM). Информационная емкость кристаллов памяти составляла 256 Мбит, их разрядности — 16 и 32. В структуре применена многобанковость (8 или 16 банков). Напряжение питания 1,8 В. Число выводов корпуса 144 (шариковые выводы тонкого корпуса BGA, Ball Grid Array). Микросхемы RLDRAM II дают полный цикл записи/чтения около 15…20 нс при частотах до 576 МГц.

Многобанковость позволяет ускорить темп передачи данных в пакетных режимах. Таким образом, в RLDRAM не только уменьшена латентность, но и сохранено повышение быстродействия ЗУ при кучности адресов последовательных обращений. Возможно, этот вариант ЗУ будет успешно конкурировать с представителями как "компьютерных", так и "некомпьютерных" микросхем памяти.

FCRAM

FCRAM (Fast Cycle RAM) т. е. ОЗУ с быстрым циклом — один из вариантов динамических ЗУ повышенного быстродействия. Фирмы Fujitsu и Toshiba, выпустившие микросхемы FCRAM, благодаря ряду приемов получили ЗУ с информационными емкостями, характерными для DRAM, и временами произвольного доступа, конкурирующими с возможностями быстродействующих SRAM. Микросхемы FCRAM имеют много общего с микросхемами SDRAM. Сравнивая варианты DDR SDRAM и DDR FCRAM, можно отметить такие отличия последних, как *сегментация ядра микросхемы*, разделенного на подматрицы, и введение во внешние относительно ядра схемы дополнительной логики для реализации *конвейерной структуры*, позволяющей обрабатывать одновременно до трех адресов. Поскольку основной целью создания FCRAM было ускорение произвольного доступа, режим страничного доступа этими микросхемами не поддерживается. После обращения строка автоматически закрывается и выполняется предзаряд в данном банке. Поддерживается *многобанковая работа* (Bank-Interleave Mode). Для FCRAM была получена длительность полного цикла доступа около 25 нс, а позднее и 20 нс, что, как и у микросхем типа RLDRAM, приблизительно вдвое меньше, чем у микросхем типов DDR SDRAM и DRDRAM. Повышение быстродействия микросхем FCRAM сопровождается их экономичностью по потребляемой мощности (благодаря *активизации только выбранного слова*, а не всей строки, так называемому автопредзаряду и т. д.). Таким образом, структуры FCRAM выступают в качестве еще одного конкурента на роль быстродействующих ЗУ, пригодных и для некомпьютерных применений.

Для микросхем FCRAM с малой потребляемой мощностью, применяемых в мобильных телефонах, при емкостях до 256 Мбит получено время доступа 60…70 нс. При этом ток, потребляемый в активном режиме, не превышает 20 мА, в режиме покоя составляет 70…100 мкА, а в режиме глубокого понижения мощности — 10 мкА.

§ 5.14. Регенерация данных в динамических ЗУ

Во избежание потери информации динамические ЗУ нуждаются в периодической регенерации. Без обновления заряды конденсаторов могут сохраняться только в течение миллисекунд (до 50…60 мс). Традиционный режим строчной регенерации — проведение циклов чтения по всем строкам матрицы ЗЭ. При этом процесс не сопровождается выдачей данных на выходные буферы, а целиком проходит внутри ЗУ. Используются только адреса строк, а адреса столбцов не требуются.

Если длительность цикла чтения t_{CY}, а число строк матрицы $N_{стр}$, то на регенерацию данных потребуется время $t_{рег} = t_{CY}N_{стр}$.

Относительные потери времени на регенерацию составят величину

$$\tau_{per} = (t_{per}/T_{per}),$$

где T_{per} — период повторения операции регенерации.

Например, в ЗУ с емкостью 4 Мбита, для которого длительность цикла чтения равна 50 нс, а период регенерации составляет 50 мс, потери времени на регенерацию составят $\tau_{per} = (50 \cdot 10^{-9} \cdot 2^{11}/50 \cdot 10^{-3}) \cdot 100\% = 0{,}2\%$ ($2^{11} = 2048$ — число строк в квадратной матрице, содержащей 4М запоминающих элементов).

Пример схемы контроллера регенерации приведен на рис. 5.56. Модуль памяти составлен из одноразрядных микросхем, число которых равно разрядности хранимых в ЗУ слов. Относительно входных сигналов все микросхемы включены параллельно. В рабочем режиме модулем управляет процессор, в режиме регенерации — контроллер.

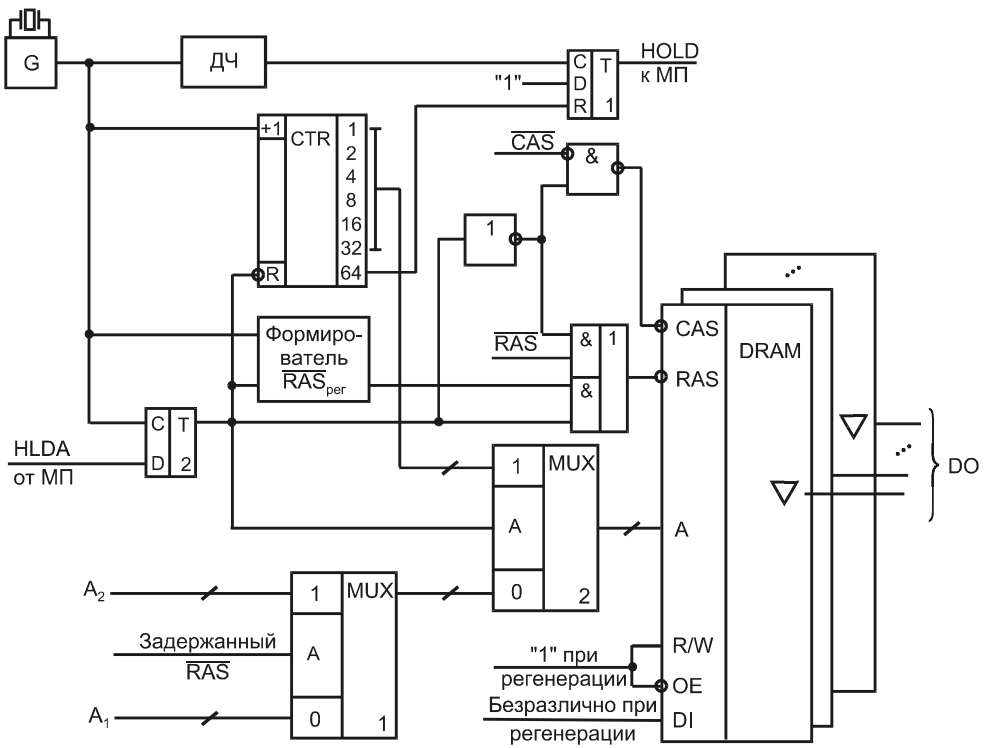


Рис. 5.56. Схема контроллера динамического ОЗУ

Рабочий режим

В рабочем режиме триггеры T1 и T2 сброшены. Нулевое значение выхода T2 сбрасывает счетчик CTR и удерживает его в нулевом состоянии, блокирует передачу через элемент И-ИЛИ строба RAS_{per} и по адресному входу A мультиплексора

MUX2 обеспечивает передачу на выход этого мультиплексора адресов от мультиплексора MUX1.

При этом модуль памяти получает стробы $\overline{RAS}$ и $\overline{CAS}$, соответствующие рабочему режиму, адреса A1 и A2 строк и столбцов, выдаваемые процессором в сопровождении этих стробов, а также сигналы управления R/W и $\overline{OE}$. При записи модулем воспринимаются входные данные DI, при чтении выдаются выходные данные DO. Так реализуется рабочий режим.

Переход к режиму регенерации

Генератор G непрерывно генерирует последовательность импульсов, период повторения которых равен длительности цикла чтения ЗУ. Делитель частоты ДЧ понижает частоту импульсов генератора так, что на его выходе период повторения импульсов будет равен периоду регенерации $T_{рег}$ (составит несколько миллисекунд). Таким образом, с периодом $T_{рег}$ на выходе ДЧ появляется импульс, заставляющий триггер T1 принять единичное состояние и инициировать режим регенерации. Единичное значение сигнала HOLD является сигналом запроса на управление памятью со стороны контроллера. Этот сигнал поступает на соответствующий вход процессора. Процессор не может остановиться мгновенно, т. к. для прерывания выполняемой им программы требуются определенные операции. Произведя эти операции, процессор вырабатывает сигнал HLDA, разрешающий переход к операции регенерации ЗУ.

Режим регенерации

Сигнал HLDA устанавливает триггер T2, в результате чего блокируется передача стробов $\overline{RAS}$ и $\overline{CAS}$ на модуль памяти, разрешается передача на модуль строба $\overline{RAS_{рег}}$, вырабатываемого формирователем контроллера, мультиплексор MUX 2 переключается на передачу адресов со счетчика CTR на адресный вход ЗУ. Одновременно с этим триггер T2 снимает сигнал асинхронного сброса со входа $\overline{R}$ счетчика, и он начинает перебирать адреса строк от нулевого до максимального (конкретно в показанной схеме таких адресов 64). Появление импульса переполнения счетчика сбрасывает триггер T1, обозначая этим окончание операции регенерации и снимая сигнал HOLD. В ответ процессор снимает сигнал HLDA, после чего очередной импульс генератора сбрасывает T2, возвращая схему в рабочий режим.

В последнее время разработаны совмещенные контроллеры кэш-памяти и динамических ЗУ.

Квазистатические ЗУ

В некоторых ЗУ схемы регенерации реализованы на самом кристалле памяти, и от разработчика не требуется специальных мер по организации этого процесса. Такие

ЗУ называют *квазистатическими*. Пример квазистатических ЗУ — микросхемы 1T-SRAM фирмы Mosys (Monolithic Systems), трактуемые как статические ЗУ с однотранзисторными запоминающими элементами, что по существу неправильно, т. к. запоминающим элементом является все-таки конденсатор.

§ 5.15. Перспективные запоминающие устройства

Даже впечатляющие успехи полупроводниковой памяти не снимают проблемы дальнейшего совершенствования ЗУ. Чтобы приблизиться к идеалу, желательно к таким свойствам, как большая емкость, высокое быстродействие и малая потребляемая мощность, добавить и энергонезависимость, которой современные ОЗУ не обладают. Если к такому комплексу качеств прибавить и низкую стоимость, то получатся ЗУ, близкие к идеалу. На пути совершенствования ЗУ используют несколько новых физических явлений — ферроэлектрических, магниторезистивных, связанных с изменением фазовых состояний материалов и др.

FRAM (ферроэлектрические ЗУ)

Ферроэлектрические ЗУ — *FRAM* (Ferroelectric RAM) основаны на применении материала, в кристаллической структуре которого имеется *бистабильный атом*. Занимая одно из двух возможных пространственных положений ("верхнее" или "нижнее"), этот атом создает в материале внутренние диполи того или иного знака (*спонтанная поляризация*). Придать внутреннему диполю тот или иной знак можно с помощью электрического поля. Под воздействием внешнего электрического поля и при температуре не выше определенной (связанной с точкой Кюри) *материал поляризуется*, диполи выстраиваются упорядоченно и могут отображать двоичные данные 0 и 1. Материал имеет петлю гистерезиса, показанную на рис. 5.57, *а*.

Через U_C на рис. 5.57, *а* обозначены коэрцитивные напряжения, через P_R — остаточные поляризации, которые сохраняются после снятия электрических полей, создаваемых напряжением U.

Если бы вместо ферроэлектрического конденсатора был включен обычный, соединенный не с Plate-линией, а с общей точкой схемы ("землей"), то получился бы запоминающий элемент обычного динамического ЗУ, и подключение конденсатора через транзистор к линии записи/считывания ЛЗС позволяло бы считывать хранимую элементом информацию (ЛВ — линия выборки), так как в зависимости от заряженности или разряженности конденсатора по-разному изменялось бы напряжение на линии ЛЗС. Здесь же нужно выявить не наличие или отсутствие заряда конденсатора, а знак поляризации запоминающего элемента. Простым подключением ферроэлектрического конденсатора к линии ЛЗС этого не определить. Поэтому после отпирания транзистора выборки на Plate-линию подается импульс длительностью около 10 нс. Если этот импульс вызовет переполяризацию элемента, то через него пройдет большой

ток, который сможет ощутимо изменить напряжение на паразитной емкости ЛЗС, изображенной на рис. 5.57, *б* штриховыми линиями. Если же знак поляризации был иным и переполяризации элемента не будет, то ток через него будет малым и не сможет заметно повлиять на потенциал линии ЛЗС.

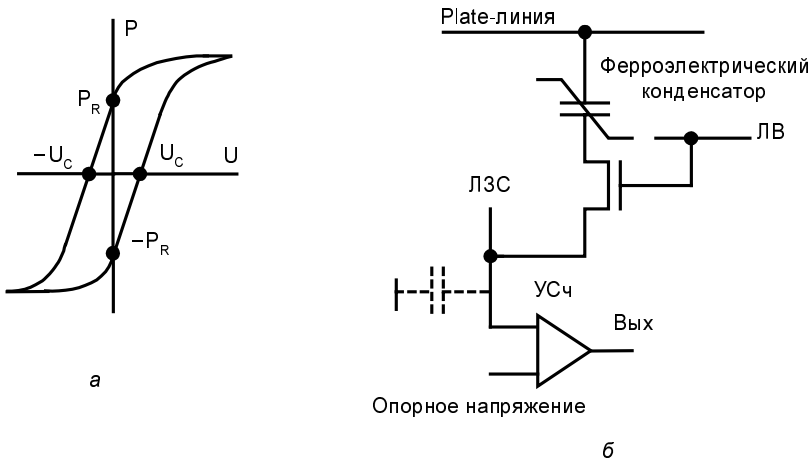


Рис. 5.57. Петля гистерезиса ферроэлектрического материала (*а*) и схема запоминающего элемента FRAM (*б*)

После пропускания импульса от Plate-линии подается питание на усилитель считывания УСч, логическое состояние которого определится тем, смогла ли линия ЛЗС зарядиться выше или ниже опорного напряжения, т. е. знаком поляризации запоминающего элемента. При этом выходной сигнал усилителя фиксируется (защелкивается) для обратной подачи на разрядную линию и возвращения ферроэлектрического конденсатора в его первоначальное состояние после проведенной разрушающей операции чтения. Такая обратная перезапись информации обеспечивает сохранение считанных данных и занимает время около 10…20 нс. Процессы в ЗУ синхронизированы с фронтами управляющих импульсов.

Рассмотренный запоминающий элемент называют элементом 1Т/1С. Существуют также элементы типа 2Т/2С, похожие на сдвоенный элемент 1Т/1С. В таких элементах две ячейки 1Т/1С программируются в противоположных направлениях и в элементе имеются две разрядные линии с взаимноинверсными сигналами. Используется дифференциальный канал для восприятия сигналов, а это повышает помехоустойчивость ЗУ и улучшает некоторые другие параметры.

Достоинства FRAM: *быстрые запись и чтение, практически неограниченное число циклов чтения/записи, малые напряжения питания и потребляемая мощность, компактность запоминающего элемента (площадь его соизмерима с площадью обычного запоминающего элемента DRAM), высокая радиационная стойкость, энергонезависимость.*

Такой набор достоинств позволяет FRAM выступить в роли конкурента как по отношению к динамическим ОЗУ, не обладающим энергонезависимостью, так и по отношению к EEPROM и Flash, не обеспечивающим быструю запись данных.

Среди первых выпущенных FRAM можно отметить микросхему емкостью 64 Кбит фирмы Samsung и работы фирмы NEC, создавшей ячейку с вчетверо большей компактностью относительно предыдущих технологий, совместимую с КМОП-логикой. В 2006 г. выпущен первый микроконтроллер с FRAM на кристалле.

PFRAM (полимерно-ферроэлектрические ЗУ)

PFRAM (Polimeric Ferroelectric RAM) — разновидность ферроэлектрических ЗУ. Они построены на основе полимерных ферроэлектрических материалов, в которых образуются полимерные цепи. Звенья этих цепей обладают дипольными моментами. Диполи служат запоминающими элементами, хранящими двоичные данные при изменении поляризации. Расположенный в полимерной пленке запоминающий элемент размещается между двумя взаимно перпендикулярными металлическими дорожками, на которые задаются определенные напряжения (рис. 5.58).

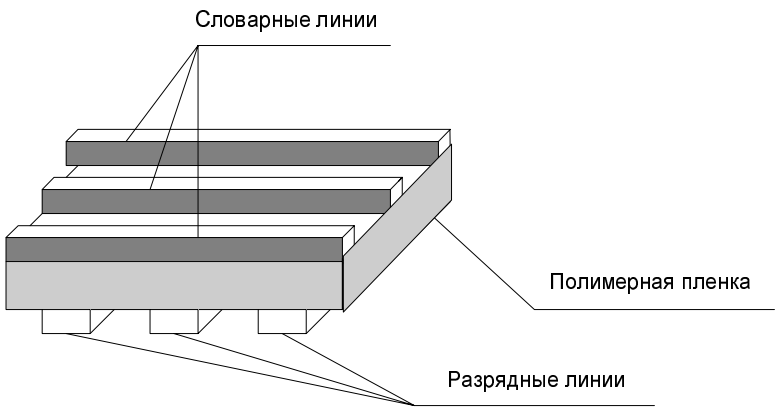


Рис. 5.58. Схематическая конструкция полимерно-ферроэлектрического ЗУ

ЗУ типа PFRAM отличаются крайне простыми запоминающими элементами — в них вообще нет транзисторов, не говоря уже о более сложных элементах. Показанный на рисунке "сэндвич" может служить частью многослойной конструкции, в которой подобные "сэндвичи" собираются в "этажерку" (стек). В стеках можно хранить очень большой объем данных. Технология изготовления ЗУ проста и хорошо сочетается со стандартными процессами изготовления интегральных схем. Обеспечивается очень малая стоимость/бит. При хранении данных не требуются какие-либо затраты мощности. Считывание является неразрушающим. Индивидуальные биты активизируются возбуждением словарной и разрядной линий, на пересечении которых они находятся. Наличие созданных диполей себя проявляет, и набор чувствительных усилителей в разрядных линиях воспринимает значения битов данных.

Процессы записи и чтения идентичны по быстродействию — и тот и другой занимают приблизительно по 50 мкс. Эта цифра исключает какой-либо разговор о быстродействии, она на три порядка превышает времена доступа обычных DRAM. Поэтому PFRAM перспективны не в качестве ОЗУ, а для замены дисковой памяти.

MRAM (магниторезистивные ЗУ)

В ЗУ типа *MRAM* (Magnetoresistive RAM) битам двоичных данных соответствуют участки с остаточной намагниченностью, образующие "микромагнитики", положение полюсов которых задается при записи информации. Магнитные ЗУ обладают естественной энергонезависимостью. Магнитные поля отдельных магнитиков обнаруживаются расположенными у их краев элементами с магниторезистивными свойствами, электрическое сопротивление которых зависит от магнитного поля, окружающего эти элементы. Чтение не является разрушающим.

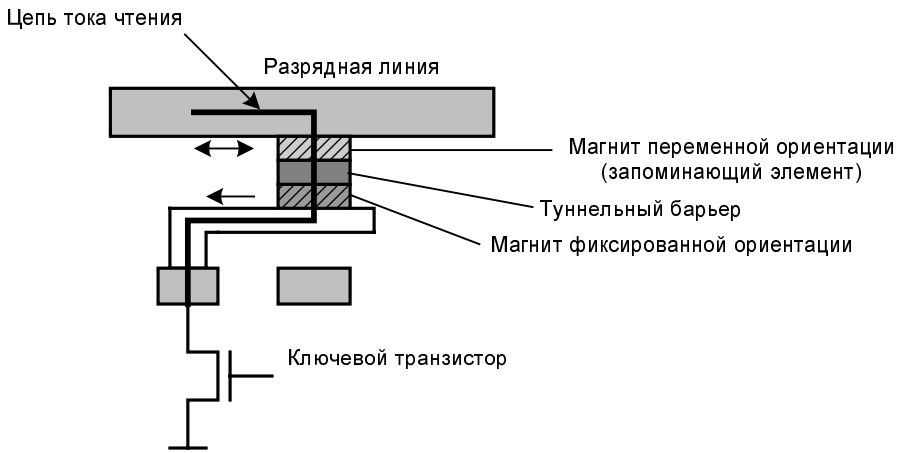


Рис. 5.59. Схематическая конструкция запоминающего элемента типа MTJ

Для создания MRAM используются два типа эффектов — так называемый *гигантский магнитный эффект* (в памяти типа *GMRAM*, Giant Magnetic-resistive Random Access Memory) или *туннелирование носителей заряда* через тонкий слой, управляемое магнитным полем (MTJ, Magnetic Tunnel Junction). Остановимся кратко на втором направлении.

Конструкция запоминающего элемента типа MTJ включает в себя два ферромагнитных слоя, разделенных тонким слоем диэлектрика, действующим как туннельный барьер (рис. 5.59). Электрическое сопротивление такого элемента зависит от магнитного поля, создаваемого в тонком слое. Поле зависит от окружающих диэлектрик двух ферромагнитных слоев — если их магнитные моменты параллельны, то сопротивление MTJ минимально, если антипараллельны, то максимально. Чтение осуществляется измерением туннельного тока между магнитными слоями. В этой конст-

рукции разница между сопротивлениями элементов, находящихся в состояниях 0 и 1, достигает 50%.

Над магнитной памятью работают фирмы Motorola, Intel и др.

ЗУ типа OUM (с фазовыми переходами вещества)

ЗУ типа OUM (Ovonics Unified Memory, по названию фирмы Ovonics) построено на основе физических эффектов, которые уже успешно использовались в памяти на компакт-дисках. В OUM эти эффекты реализуются по технологии изготовления интегральных схем. Как и в дисках CD и DVD с перезаписью данных, в ЗУ типа OUM применены *халкогенидные сплавы*. Халкогенид — сплав GeSbTe, который может иметь кристаллическое проводящее или аморфное непроводящее состояния. Эти состояния материал может сохранять, а выявлять их можно измерением сопротивления запоминающего элемента. Состояния взаимно обратимы, их изменения происходят быстро. В конструкции запоминающего элемента небольшой объем халкогенида играет роль резистора с программируемым сопротивлением при динамическом диапазоне между значениями низкого и высокого сопротивлений свыше 40.

Фазовое состояние халкогенида ("кристаллическое — аморфное") программируется пропусканием импульсов тока, имеющих разную величину. Управление током производится с помощью МОП-транзистора. Чтение бита осуществляется путем измерения сопротивления.

В элементе памяти с халкогенидом можно программировать сопротивление не только для двух его значений (максимального и минимального), но и для промежуточных, а это означает принципиальную возможность использовать в памяти многоуровневые сигналы, и, следовательно, хранить в одном элементе более одного бита данных.

Запоминающие элементы OUM просты по конструкции, потребляют малую мощность, энергонезависимы, имеют неразрушающее чтение, допускают до 10^{12} циклов записи/стирания, время записи равно приблизительно 100 нс. Особо можно отметить предполагаемую высокую надежность памяти OUM, что существенно для военной и аэрокосмической аппаратуры. Фирма Ovonics совместно с фирмой Intel разработала тестовый кристалл памяти OUM с топологической нормой 0,18 мкм и емкостью 1 Мбит. Начало производства модулей OUM планируется на конец 2003 г.

§ 5.16. Заключительные замечания

Архитектура, схемотехника и технология ЗУ постоянно развиваются. Поколения динамических ЗУ сменяются через 4—5 лет. Уровень интеграции и быстродействие памяти растут с уменьшением топологических норм проектирования, которые в настоящее время достигли 45 нм, а по прогнозу на ближайшие годы должны вновь существенно снизиться. Это создает перспективы совершенствования уже

сложившихся разновидностей полупроводниковых ЗУ, что и подтверждается регулярными сообщениями фирм-разработчиков ЗУ о новых успехах.

Производство современных ИС ЗУ требует крупных инвестиций (миллиардов долларов). Для разработки новых микросхем высокой емкости и быстродействия объединяют усилия даже самые известные и мощные фирмы.

Информационные емкости основных разновидностей современных микросхем памяти достигли внушительных уровней (приводятся данные для микросхем фирмы Samsung — лидера в области производства разнообразных видов ЗУ):

- для флэш-памяти с NOR-ячейками от 16 до 51 Мбит;
- для флэш-памяти с NAND-ячейками от 256 Мбит до 32 Гбит (микросхемы SLC Large Block), причем в ближайшие годы ожидается снижение цены до 1 доллара США за мегабайт;
- для SDRAM от 2 до 512 Мбит;
- для SRAM малой мощности от 2 до16 Мбит.

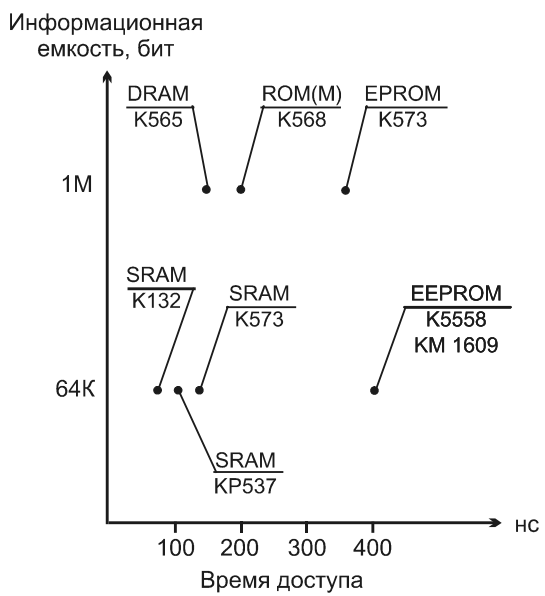


Рис. 5.60. Параметры емкости и быстродействия отечественных запоминающих устройств

Латентность динамических ЗУ с фрагментированными матрицами снижена до 7…15 нс. Быстродействие статических ОЗУ характеризуется тактовыми частотами в 250 МГц, в ближайшие годы планируется достичь 600 МГц и далее 1 ГГц (в следующие 5…10 лет).

На горизонте появляются новые смелые идеи создания устройств памяти на новых материалах и физических явлениях, обещающие трудно вообразимые сегодня пара-

метры емкости, быстродействия, энергетической экономичности и надежности ЗУ. В конечном счете речь идет об отображении двоичных данных с помощью определенного состояния всего одного электрона.

Параметры отечественных микросхем памяти за последние 10…20 лет практически не изменились (рис. 5.60) и являются более чем скромными по сравнению с уровнем мировой техники.

Контрольные вопросы и упражнения

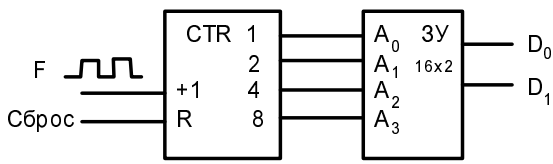
1. Чем объясняется сложный иерархический характер современных систем компьютерной памяти?
2. Что называется организацией микросхем памяти?
3. Нарисуйте УГО микросхемы памяти с организацией 256К×8, L-активными входами разрешения работы и разрешения вывода и общими выводами входов-выходов.
4. Подсчитайте общее число выводов корпуса микросхемы памяти с организацией 1М×8, входами разрешения работы и разрешения вывода и общими выводами входов-выходов.
5. Перечислите параметры, характеризующие быстродействие запоминающих устройств.
6. Почему времена считывания и записи и длительности соответствующих циклов могут отличаться друг от друга?
7. Что означает термин "энергонезависимость ЗУ"?
8. Какие ЗУ называют "многопортовыми"?
9. Укажите основные недостатки и достоинства структур 2D и 3D адресных ЗУ.
10. Структура 2DM адресных ЗУ имеет преимущества относительно структур 2D и 3D. В чем состоят эти преимущества?
11. Экран монитора состоит из 1280×1024 пикселов и во избежание мерцания изображения пикселы подсвечиваются с частотой 70 Гц. Для каждого из трех основных цветов предусмотрены 256 градаций яркости. Какова емкость видеопамяти, обслуживающей монитор, и с какой частотой она выдает выходные коды?
12. Почему фрагментирование ЗУ большой емкости (разбиение матрицы запоминающих элементов на блоки) позволяет существенно повысить их быстродействие?
13. Что называют "гнездовым характером обращений" к ЗУ или "кучностью адресов"? При реализации каких задач проявляются эти свойства?

14. Термин "кэш" (Cache) в переводе означает "скрытый". Чем объясняется применение этого термина к памяти соответствующего типа?

15. Какая память называется ассоциативной? Что такое "тег"? В чем состоят недостатки кэш-памяти с полной ассоциацией и какими путями они преодолеваются в альтернативных структурах кэш-памяти?

16. Требуется реализовать устройство для перемножения двух четырехразрядных двоичных чисел табличным методом. Нарисуйте УГО запоминающего устройства, которое выполняет эту задачу. Покажите на примере нескольких строк таблицу для программирования ЗУ.

17. На рисунке изображена схема формирования логических функций времени, вид которых зависит от программирования ЗУ. Начальное состояние счетчика нулевое. От генератора поступают импульсы частоты F. Каждый импульс увеличивает число в счетчике и адрес ЗУ на единицу, так что адресный код, линейно изменяясь во времени, отображает течение времени в цифровой форме. Через 16 импульсов начнется новый (повторный) цикл, т. к. после переполнения счетчика в нем снова возникнет нулевое состояние. На каждом интервале T = 1/F значения выходов D_0 и D_1 (0 или 1) определяются хранимой в ЗУ информацией. Постройте временные диаграммы для сигналов D_0 и D_1 для данного в таблице варианта программирования ЗУ:



$A_3A_2A_1A_0$	D_0D_1	$A_3A_2A_1A_0$	D_0D_1
0 0 0 0	0 0	1 0 0 0	1 1
0 0 0 1	1 1	1 0 0 1	0 0
0 0 1 0	0 0	1 0 1 0	1 0
0 0 1 1	1 0	1 0 1 1	1 0
0 1 0 0	1 1	1 1 0 0	1 0
0 1 0 1	0 0	1 1 0 1	1 1
0 1 1 0	1 0	1 1 1 0	0 0
0 1 1 1	1 0	1 1 1 1	1 1

18. Требуется воспроизвести функцию Sin X ($0 \le X \le 90°$) при дискретности задания аргумента 0,5° и функции 0,1%. ЗУ какой организации потребуется для решения этой задачи табличным методом?

19. Дайте определение следующим разновидностям ЗУ:
PROM, EPROM, EEPROM, Flash, SRAM, SSRAM, DRAM, SDRAM.

20. Что называется "страницей" применительно к структурам динамических ОЗУ? Почему страничный доступ является более быстрым, чем обычный?

21. Что называется пакетным доступом к данным? Почему пакетный доступ является более быстрым, чем обычный?

22. Какие структуры ЗУ называются многобанковыми? Почему и в каких условиях они являются более быстрыми, чем однобанковые структуры?

23. Почему понадобилось перейти от обычного восприятия данных по фронтам одного знака к восприятию по фронтам обоих знаков (к технологии DDR)?

24. Укажите достоинства и недостатки запоминающих элементов ОЗУ статического и динамического типов.

25. Какими способами можно предотвратить потерю данных в статических ЗУ при отключении питания?

26. Правильно ли утверждение "при считывании всего одного бита из динамической памяти с организацией $2^n \times 1$ регенерируется целая строка матрицы запоминающих элементов"?

27. Какие места в иерархии компьютерной памяти занимают статические и динамические ОЗУ?

28. Сколько выводов корпуса экономится в ИС динамической памяти с организацией $4M \times 1$ благодаря применению мультиплексирования адреса?

29. Подсчитайте общее число выводов корпуса для микросхемы DRAM с организацией $4M \times 1$.

30. Почему характерной особенностью синхронных ЗУ (статических и динамических) являются "защелки" на их входах и выходах? Как их наличие способствует повышению быстродействия систем с синхронными ЗУ?

31. Какая разновидность перспективных ЗУ скорее всего составит сильную конкуренцию современным видам запоминающих устройств?

Литература к главе: [19], [27], [28], [31], [38], [48], [50], [55], [IV], [VI], [VIII], [X], [XII], [XV], [XX], [XXVI], [XXVII], [XXXI], [XXXII].

Глава 6

Простые микропроцессоры и микропроцессорные системы. Микроконтроллеры

§ 6.1. Общие сведения. Структура и функционирование микропроцессорной системы

Микропроцессор (МП) — построенное на одном кристалле (реже на нескольких) программно-управляемое устройство, осуществляющее процесс обработки информации и управление им.

Микропроцессоры появились, когда микросхемы приобрели уровень интеграции, при котором набор блоков, необходимых для программной реализации алгоритмов, удалось разместить на одном кристалле. Первый микропроцессор (фирма Intel, USA, 1971 г.) содержал 2300 транзисторов, работал на тактовой частоте 100 кГц и выполнял команду в среднем за 10 тактов.

В последующие годы микропроцессоры стали широко применяться в аппаратуре самого разного назначения. Необычайная популярность микропроцессоров объясняется их *универсальностью* (с помощью одной и той же микросхемы можно решать разные задачи в зависимости от заданной программы). Внедрение в самые разнообразные сферы жизни и, соответственно, массовость производства, снижает стоимость МП. Удешевление МП, в свою очередь, расширяет круг их потребителей. Так возникает своеобразная положительная обратная связь, интенсифицирующая развитие микропроцессоров, рост их производства и применения. Современные МП содержат до миллиардов транзисторов, работают на гигагерцовых частотах и выполняют команды за один такт или даже быстрее.

Программа реализуется *микропроцессорной системой* (МПС), основными частями которой являются:

- микропроцессор (центральный элемент МПС);
- память;
- интерфейсные схемы и контроллеры для взаимодействия с внешними устройствами.

Диапазон характеристик современных МП чрезвычайно широк. Соответственно областям применения целесообразно выделить классы:

- простые МП;
- сложные МП высокой производительности.

> ***Примечание***
>
> В современных условиях уточнение "микро" применительно к термину "процессор", как правило, излишне, поскольку об иных вариантах реализации процессоров речь заведомо не идет. Поэтому дальше будем пользоваться и термином "процессор", имея в виду его реализацию на кристалле.

Первый класс ориентирован на применение в системах управления техническими объектами и технологическими процессами, причем чаще всего в качестве встроенных блоков. В подобных применениях требования к вычислительной мощности МП обычно относительно скромны, разрядность МП невелика и развитие идет в направлении интеграции функций в пределах одного кристалла. Если простой МП занимает уже не весь кристалл, то целесообразно добавить к нему другие блоки МПС (энергонезависимую и оперативную память, порты ввода/вывода, аналого-цифровые и цифроаналоговые преобразователи и т. д.). Так возникли несложные однокристальные МПС — *микроконтроллеры* (МК). В БИС/СБИС с программируемой структурой реализуются микроконтроллеры, структуру которых можно изменять по желанию проектировщика. В таких *программируемых системах на кристалле* МП играют роль отдельных "ядер", а кристалл в целом представляет собой программируемую МПС.

Современные средства обработки информации в значительной мере становятся портативными, питаемыми от автономных источников (батарей). В связи с этим для простых МП и МК задачей особой важности стало *уменьшение потребляемой мощности*, что достигается не только режимами работы схем (снижением питающего напряжения и др.), но и введением в структуры МПС внутренних подсистем управления питанием.

Сложные МП высокой производительности с разрядностью 32 или 64 разрядов характерны для универсальных компьютеров (персональных, серверов и т. п.). На универсальных компьютерах решаются разнообразные задачи, в том числе требующие большой вычислительной мощности. Это ведет к усложнению архитектур процессоров, введению параллелизма вычислений, кэшированию памяти, усложнению шинных структур и т. д. Система в целом реализуется на нескольких кристаллах. В частности, основной процессор стали дополнять *сопроцессором* (в первую очередь для арифметики с плавающей точкой). Так как в многокристальной системе возникают трудности передачи между кристаллами интенсивных потоков информации, "собирание" функций на одном кристалле остается желательным. Поэтому и для этого класса МПС стремятся к *"системам на кристалле"*, построенным на основе БИС/СБИС высшей сложности и стоимости и существенно отличающимся от своих "младших братьев" невысокой сложности.

В данном учебном пособии рассматриваются простые МП и МПС, поскольку знание основ их функционирования необходимо для понимания работы разнообразных цифровых устройств. Типовые устройства работают под управлением процессора и их параметры и характеристики подчинены системным требованиям. Изучение МП во всей полноте этой темы является предметом специальных курсов.

Структура простой МПС

Типичная простая МПС имеет *магистрально-модульную* структуру. В такой структуре имеется группа магистралей (шин), к которым подключаются различные модули (блоки), обменивающиеся между собой информацией по одним и тем же шинам поочередно, в режиме разделения времени.

Типична *трехшинная структура* МПС с шинами адресов ША, данных ШД и управления ШУ. Соответствующие английские термины: AB (Address Bus), DB (Data Bus) и CB (Control Bus) или просто шины A, D, C. Трехшинная структура в чистом виде характерна для простых МПС, в более производительных системах шинная структура образует более сложную иерархию, но в своей основе разделение на перечисленные виды шин остается справедливым.

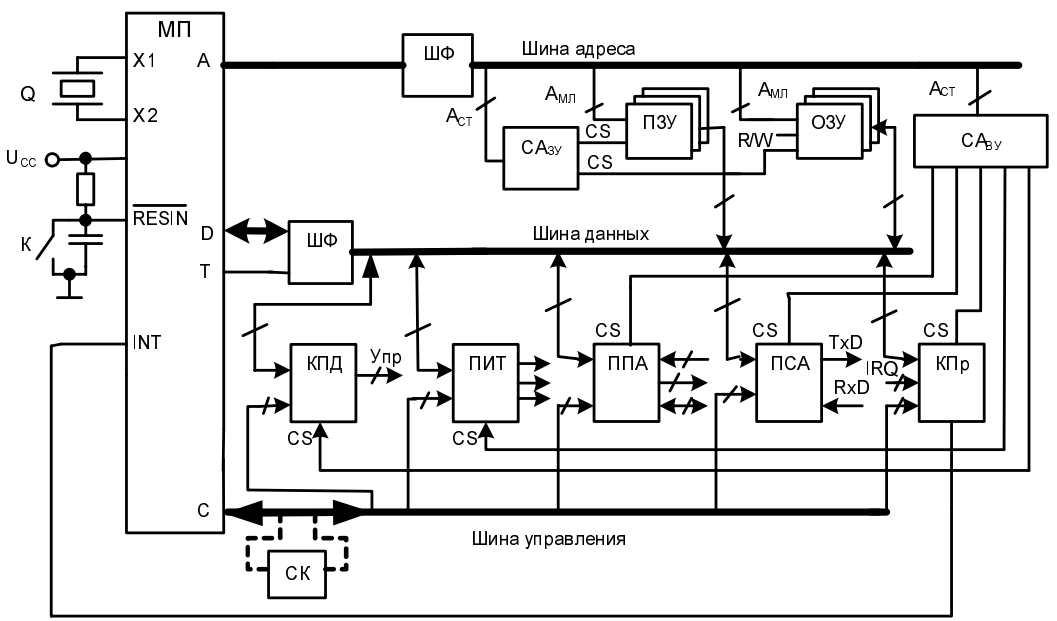


Рис. 6.1. Структура микропроцессорной системы

На рис. 6.1 показана микропроцессорная система с *Принстонской архитектурой* (архитектурой *фон Неймана*). По адресной шине A в систему передаются адреса модулей, к которым обращается процессор МП. В эту шину включен *шинный формирователь* ШФ, обеспечивающий работу на нагрузку, образуемую внешними це-

пями. Собственной нагрузочной способности для работы на шину у выводов МП, как правило, не хватает.

Адреса используются блоками постоянной и оперативной памяти ПЗУ и ОЗУ, а также адаптерами и контроллерами, посредством которых процессор общается с внешними устройствами (ВУ). При адресации памяти старшие и младшие разряды адресов используются по-разному. Старшие разряды поступают на селектор адреса $CA_{ЗУ}$. Соответственно старшим разрядам селектор разрешает работу тех или иных блоков с помощью сигналов выбора кристаллов CS (Chip Select). Младшие разряды адресов используются для адресации слов внутри выбранного блока. Оперативная память получает от процессора управляющий сигнал записи/чтения R/W (Read/Write), указывающий на требуемое направление передачи данных между памятью и процессором. Постоянная память в подобном сигнале не нуждается, так как в рабочем режиме используется только для чтения. Селектор адресов внешних устройств $CA_{ВУ}$ декодирует адреса, присвоенные адаптерам и контроллерам, и вырабатывает сигналы CS разрешения работы выбранным модулям.

С внешними устройствами, работающими с параллельными кодами, процессор общается с помощью программируемого *параллельного адаптера* ППА, который имеет три канала связи с ВУ. Для каналов ППА могут быть выбраны различные режимы ввода, вывода и двунаправленных передач и разные способы обмена с процессором. С ВУ или каналами связи, работающими с последовательными кодами, процессор общается с помощью программируемого *связного адаптера* ПСА. ПСА преобразует параллельные коды, получаемые от процессора, в последовательные либо наоборот, получая последовательные коды, преобразует их в параллельные для передачи в процессор. Программирование адаптера позволяет настраивать его на разные протоколы и режимы обмена. Последовательные выходные сигналы адаптера поступают в линию передатчика TxD (Tranceiver Data), последовательные входные — в линию приемника RxD (Receiver Data).

Контроллеры прерываний КПр обеспечивают обмен с внешними устройствами в режиме прерывания (временной остановки) выполняемой программы для обслуживания запроса от ВУ. Контроллер принимает запросы прерываний IRQ (Interrupt Requests) от нескольких внешних источников и обрабатывает их с учетом приоритетов и маскирований. *Векторные прерывания* реализуются следующим образом. Определив подлежащий обслуживанию запрос, контроллер запрашивает прерывание у процессора, выставляя для него сигнал INT (Interrupt). Если процессор признает запрос, он производит необходимую подготовку для перехода от основной программы к подпрограмме обслуживания прерывания, после чего отвечает контроллеру сигналом подтверждения прерывания INTA (Interrupt Acknowledge). Сигнал INTA (на рисунке не показан) служит стробом чтения сформированного контроллером начального адреса подпрограммы (вектора). Вектор поступает в процессор, затем выполняется подпрограмма обслуживания прерывания, после чего возобновляется выполнение основной программы.

Контроллер прямого доступа к памяти КПД (*DMA* controller — Direct Memory Access controller) обслуживает процесс прямого обмена данными между внешними уст-

ройствами и памятью. Обычная передача данных между ВУ и памятью выполняется так: сначала слово читается процессором из устройства-источника данных, потом оно записывается процессором в устройство-приемник, что требует двух машинных циклов. При *прямом доступе к памяти* (ПДП) процессор отключается от шин системы и передает управление шинами предварительно запрограммированному КПД, который реализует более быструю (за один цикл) передачу данных непосредственно между источником и приемником. Особенно эффективен ПДП при блочных передачах.

Программируемые интервальные таймеры ПИТ (*PIT*, Programmable Interval Timer) выполняют операции над временными интервалами (часы реального времени, генерация звуковых сигналов, сторожевые таймеры, генерация временных меток в многозадачных процессах с разделением времени между задачами, выработка широтно-модулированных импульсов и т. д.). На рисунке показан таймер с тремя выходными каналами.

Кроме сигналов, обозначенных на рис. 6.1, адаптеры и контроллеры имеют и другие управляющие сигналы, поступающие от шины управления. В число таких сигналов входят стробы записи (для загрузки в программируемые модули управляющих слов), стробы чтения (для чтения программой слов состояния модулей), адресные коды для адресации внутренних регистров модулей, сигналы сброса, тактирования и др.

Передачи данных в МПС осуществляются по системной ШД, разрядность которой определяет и понятие "*разрядность процессора*". Эти передачи двунаправлены, направление задается *шинным формирователем* ШФ (*BD*, Bus Driver) в зависимости от сигнала T (Transit). При активном состоянии формируемого процессором сигнала чтения RD (Read) данные передаются через ШФ справа налево, при пассивном — в обратном направлении. К шине данных подключены информационные выводы всех модулей МПС.

Выводы X1 и X2 служат для подключения кварцевого резонатора или иных контуров, задающих частоту тактовому генератору, расположенному в МП. Вход $\overline{\text{RESIN}}$ (Reset Input) является входом L-активного асинхронного сброса, приводящим процессор в исходное состояния. Сброс может быть осуществлен замыканием ключа К и автоматически происходит при включении питания U_{cc}. В этом случае благодаря цепочке RC напряжение на входе $\overline{\text{RESIN}}$ после включения питания нарастает постепенно, и в течение некоторого времени остается низким (ниже порогового), что равноценно подаче на этом интервале времени сигнала $\overline{\text{RESIN}}$.

Выполняя программу, МП обрабатывает команду за командой. Команда задает выполняемую операцию и содержит сведения об участвующих в ней операндах. После приема команды происходит ее расшифровка и выполнение, в ходе которого МП получает необходимые данные из памяти или внешних устройств. Ячейки памяти и внешние устройства (порты) имеют номера-адреса, которыми они обозначаются в программе.

Таким образом, по однонаправленной адресной шине МП посылает адреса, определяя объект, с которым будет осуществляться обмен, по шине данных (двуна-

правленной) обменивается данными с модулями (блоками) системы, по шине управления в разных направлениях передаются управляющие сигналы.

ПЗУ (ROM) предназначено для хранения фиксированных программ и данных, оно является энергонезависимым и при выключении питания информацию не теряет.

ОЗУ (RAM) хранит оперативные данные (изменяемые программы, промежуточные результаты вычислений и др.), является энергозависимым и теряет информацию при выключении питания (если не применяются специальные методы, например, автоматическое подключение резервного питания на время отсутствия основного). Для приведения системы в работоспособное состояние после включения питания ОЗУ следует загрузить необходимой информацией.

Устройства ввода-вывода (УВВ) или *внешние устройства* (ВУ) — технические средства для передачи данных из внешней среды в МП или память либо наоборот. Для подключения ВУ необходимо привести их сигналы, форматы слов, скорость передачи и т. п. к стандартному виду, воспринимаемому данным МП. Это и выполняется адаптерами и другими интерфейсными блоками.

Мультиплексирование шины адресов/данных

Многие микропроцессоры имеют *мультиплексируемую шину* адресов/данных (рис. 6.2). Применение мультиплексируемых шин позволяет уменьшить число внешних выводов кристалла. В этом случае разрядность адресной шины сокращается вдвое, и по ней в систему передается только старший полуадрес. Младший полуадрес в начале машинного цикла выдается процессором по мультиплексируемой шине адресов/данных $AD_{7\text{-}0}$ Управляющим сигналом ALE (Address Latch Enable) этот полуадрес загружается в специальный внешний регистр (регистр-защелку адреса), где и сохраняется на все время машинного цикла. Выход регистра совместно с шиной $A_{15\text{-}8}$ образуют полный адрес $A_{15\text{-}0}$. После передачи младшего полуадреса в регистр шина $AD_{7\text{-}0}$ отдается для передач данных. Направление передач задается шинному формирователю сигналом на его входе T.

Принстонская и Гарвардская архитектуры процессоров

В процессе развития архитектура МПС претерпела существенные изменения. Первые МПС строились по уже рассмотренной *Принстонской архитектуре* (архитектуре *фон Неймана*), в которой *память для команд и данных является общей.* Эта архитектура имеет свои достоинства — простоту, возможность оперативного перераспределения памяти между областями команд и данных и др. Недостаток — последовательная выборка из памяти команд и данных, передаваемых по одной и той же системной шине, что ограничивает производительность МПС. Тем не менее в силу своих достоинств Принстонская архитектура не только длительное время доминировала в микропроцессорной технике, но и сохранилась до настоящего времени.

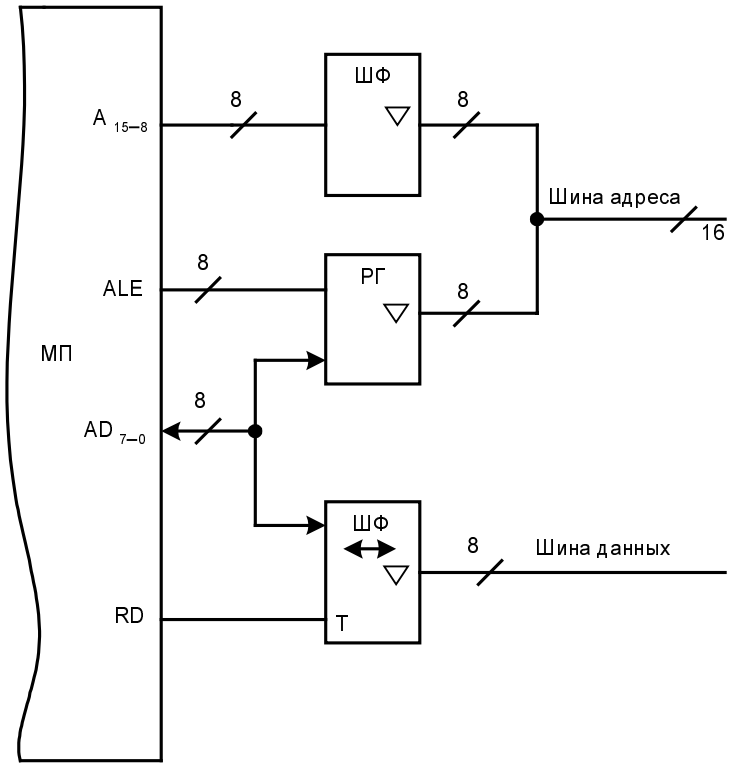


Рис. 6.2. Мультиплексирование шины адресов/данных

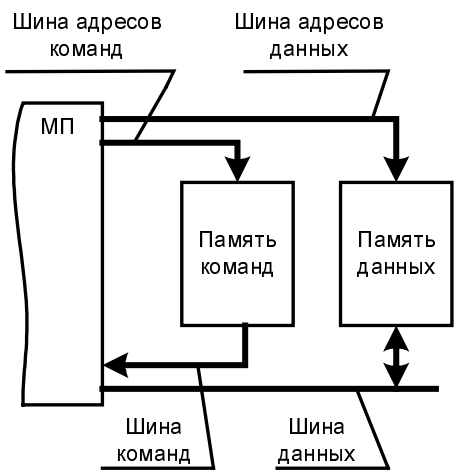


Рис. 6.3. Гарвардская архитектура МПС

В *Гарвардской архитектуре память команд и память данных разделены*, причем каждая из них имеет собственную шину для общения с процессором (рис. 6.3). При этом во время передач данных для выполнения текущей команды можно произво-

дить выборку и расшифровку следующей, что повышает производительность МПС. Реализация системы по сравнению с Принстонской архитектурой усложняется (в системе больше шин), ниже коэффициент использования памяти. Но в МПС высокой производительности и внутренних структурах высокопроизводительных МП Гарвардская архитектура находит широкое применение.

§ 6.2. Структура микропроцессора

Во всем мире широко применяются процессоры фирмы Intel и их аналоги. Эта фирма разработала первый МП, затем целый ряд их семейств и в настоящее время по разным оценкам производит 80...90% от общего мирового объема выпуска микропроцессоров.

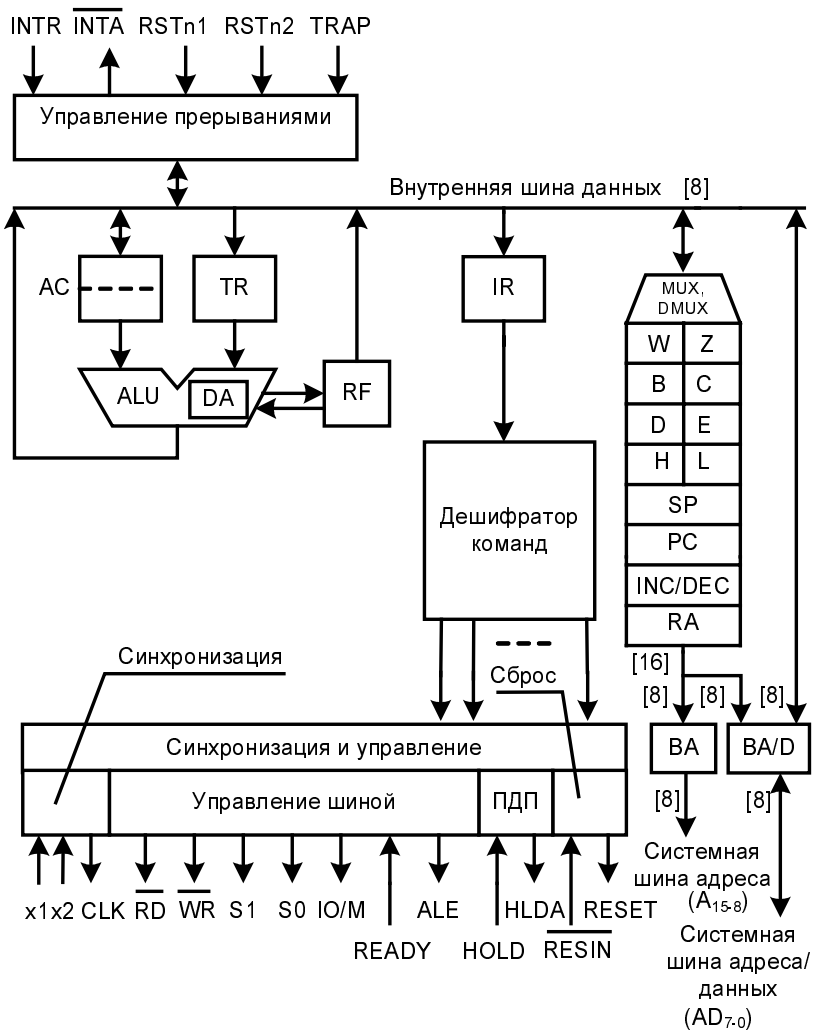


Рис. 6.4. Структура микропроцессора

Рассматриваемая далее модель процессора близка к структурам простых процессоров Intel, с которых начиналось развитие микропроцессорной техники. Модель имеет Принстонскую архитектуру и мультиплексируемую шину адресов/данных.

- Процессор (рис. 6.4) имеет восьмиразрядную внутреннюю шину данных, через которую его блоки обмениваются информацией. Эта шина передает байты между внутренними регистрами МП или обменивается с модулями МПС через мультиплексируемую шину адресов/данных AD_{7-0}. При таком обмене адрес ячейки памяти или ВУ поступает в регистр адреса RA. Буфер адреса BA с тремя состояниями выдает старший байт адреса на адресную шину $A_{15—8}$. Буфер шины адресов/данных BA/D с тремя состояниями передает на шину A/D младший байт адреса для его фиксации во внешнем регистре. Из двух байтов формируется 16-разрядный адрес памяти. Адреса ВУ являются восьмиразрядными. При адресации ВУ адресная информация на линиях A_{15-8} и AD_{7-0} дублируется. Затем через шину A/D передается байт данных.

Таким образом, в системные шины адресов и адресов/данных входят линии:

- $A_{15–8}$ — выходные линии для выдачи старшего байта адреса памяти или полного адреса ВУ. Переходят в третье состояние в режимах HOLD, HALT и RESET;
- $AD_{7–0}$ — двунаправленные мультиплексированные линии с тремя состояниями для выдачи вначале адреса (младшего байта адреса памяти или полного адреса ВУ), а затем данных.

Операционный блок

В операционный блок входят:

- AC (Accumulator) — регистр-*аккумулятор* (см. рис. 6.4), выполненный на двухступенчатых триггерах и способный хранить одновременно два слова (один из операндов и результат операции);
- TR (Temporary Register) — регистр временного хранения одного из операндов;
- ALU (Arithmetic-Logic Unit) — арифметико-логическое устройство (АЛУ), выполняющее действия над двумя словами-операндами. Аккумулятор служит источником операнда A и приемником результата, TR — источником второго операнда B. АЛУ функционирует согласно соотношению A:= A * B, где звездочкой обозначен обобщенный символ операции. Непосредственно выполняются лишь операции сложения, вычитания, сдвига, сравнения слов, поразрядные логические операции (конъюнкцию, дизъюнкцию, сложение по модулю 2). Более сложные операции (умножение, деление и др.) выполняются по подпрограммам. В АЛУ имеется схема перевода двоичных чисел в двоично-десятичные (DA, Decimal Adjust);
- RF (Register Flags) — регистр флажков, т. е. битов, указывающих признаки результатов операций, выполненных в АЛУ. Указываются пять признаков: Z (Zero) — нулевой результат, C (Carry) — перенос, AC (Auxiliary Carry) — вспомогательный перенос, S (Sign) — знак, P (Parity) — четность веса слова.

Признак вспомогательного переноса (переноса между младшей и старшей тетрадами восьмиразрядного слова) нужен при выполнении операций в двоично-десятичном коде. Смысл остальных признаков ясен из их наименований. Признаки служат для управления ходом процесса обработки информации.

Блок регистров

С внутренней шиной данных через мультиплексор/демультиплексор связан блок регистров, часть которых специализирована, другая часть (регистры общего назначения, РОН) программно доступна и может быть использована по усмотрению программиста. В блоке размещены следующие регистры:

- **Регистры W и Z.** Эти регистры предназначены для временного хранения адресной части команды при ее выборке из памяти. Они недоступны для программиста и используются только блоком внутреннего управления. Второй и третий байты команды, если они имеются, автоматически поступают в регистры W и Z.
- **Регистры B, C, D, E.** Восьмиразрядные регистры B, C, D, E относятся к регистрам *общего назначения*, т. к. адресуются программой и могут быть использованы по усмотрению программиста. Эти регистры могут применяться либо по отдельности, либо в виде пар B-C, D-E, играющих роль 16-разрядных регистров и именуемых как пары B, D (rp B, rp D).
- **Регистры H и L.** Регистры H, L (от High и Low) и rpH хотя и могут использоваться произвольно, т. е. играть роль РОН, как правило, используются для размещения адресов при косвенной адресации.
- **Регистр SP.** 16-разрядный регистр SP (Stack Pointer) — *указатель стека*. Стек (магазинная память, буфер LIFO) удобен для запоминания массива слов, т. к. при этом не требуется адресовать каждое слово отдельно. Слова загружаются в стек и считываются из него в определенном порядке. В частности, стек удобен при запоминании состояний регистров в момент прерывания программы. *Стек* имеет дно и верхушку, направление возрастания номеров ячеек в нем может быть различным (обычный и перевернутый стеки). Операции со стеком — `PUSH` (запись слова) и `POP` (считывание слова). Аппаратно стек реализуется в ОЗУ, где для него выделяется определенная область. Указатель стека SP содержит адрес последней занятой ячейки (рис. 6.5). При выполнении операций `PUSH` и `POP` значение SP уменьшается или увеличивается. Задавая в SP начальное значение, можно размещать стек в той или иной области ОЗУ, следя при этом за тем, чтобы эта область не использовалась для других целей.
- При байтовой организации памяти и занесении в стек содержимого регистровой пары старший байт запоминается по адресу SP – 1, а младший — по адресу SP – 2, содержимое SP уменьшается на 2. При выборке содержимое двух верхних ячеек стека помещается в соответствующие регистры, а содержимое SP увеличивается на 2.

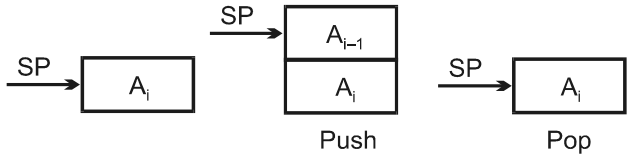


Рис. 6.5. Реализация стека в микропроцессорной системе

- **Регистр PC.** 16-разрядный *программный счетчик* PC (Program Counter) содержит адрес команды, посредством которого микропроцессор может обращаться в любую ячейку памяти. При сбросе МП регистр PC принимает нулевое состояние, которое, таким образом, является адресом первой исполняемой команды, иначе говоря, выполнение программы начинается с нулевой ячейки. Длина команды составляет 1...3 байта. Содержимое программного счетчика после выборки очередного байта из памяти автоматически инкрементируется, так что в PC появляется адрес следующей команды, если текущая команда была однобайтовой, или следующего байта текущей команды в противном случае.
- **Схема INC/DEC** (Increment/Decrement). Эта схема изменяет передаваемые через нее слова на +1 или –1.
- **Адресный регистр RA.** Через этот регистр адреса выдаются на буферы адресов (BA) и адресов/данных (BA/D) и далее на системные шины МПС. Шина адресов/данных мультиплексируется.
- **Регистр команд IR** (Instruction Register). Этот регистр принимает первый байт выбираемой из памяти команды, содержащий код операции.

Дешифрация команд

Команда содержит операционную и адресную части. Первая часть (код операции, КОП) указывает на операцию, которая должна быть выполнена. Вторая (код адреса, КАД) определяет операнды, над которыми выполняется операция. Сведения о подлежащей выполнению операции находятся в первом байте команды, который при ее выборке поступает на регистр команд IR. Содержимое регистра IR декодируется блоком дешифрации команд и формирования машинных циклов, вырабатывающим набор сигналов, настраивающих блоки МП на выполнение заданной операции (реализацию машинного цикла нужного типа). Этот набор сигналов используется блоком синхронизации и управления и другими блоками МП. Сигналы управления, настраивающие блоки МП на работу в соответствии с выполняемой операцией, поступают практически на все блоки.

Известны многие способы дешифрации команд. Примером может служить схема, показанная на рис. 6.6. Работа схемы тактируется синхрогенератором ГТИ, подключенным к распределителю тактов РТС. Выполняемые операции разбиваются на ряд этапов, число m выходов РТС определяется максимальным числом этапов в самой длинной операции. Переход активного состояния с одной выходной линии РТС на другую означает переход к очередному этапу. Набор формируемых на каж-

дом этапе сигналов определяется кодом операции КОП. Выходами схемы являются сигналы управления блоками, входами — осведомительные сигналы о состоянии МП, от которых также зависят сигналы управления.

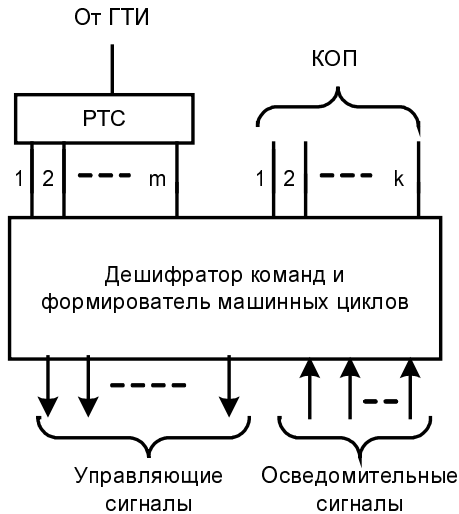


Рис. 6.6. Схема блока дешифрации команд и формирования машинных циклов

Блок синхронизации и управления

Блок служит для синхронизации процессов, генерации сигналов состояния и управления шиной (внешними устройствами микропроцессорной системы). Функции выводов и сигналов блока синхронизации и управления (см. рис. 6.4):

- X1, X2 — к этим выводам присоединяется кварцевый резонатор или другие цепи, задающие частоту внутреннего синхрогенератора МП (LC-контур, RC-цепочка). Возможна также синхронизация через эти выводы от внешнего генератора ГТИ;
- CLK — выход синхроимпульсов для микропроцессорной системы;
- $\overline{RD}$, $\overline{WR}$ — стробы чтения и записи. Низкий уровень соответствующего сигнала свидетельствует о том, что адресованная ячейка памяти или внешнее устройство должны выполнить операцию чтения или записи. Выводы переходят в третье состояние в режимах HOLD, HALT и RESET;
- S1, S0 — сигналы состояния МП, сообщаемые внешней среде. Формируются в начале и сохраняются во время всего машинного цикла;
- IO/M — сигнал выбора памяти или внешнего устройства. При высоком уровне происходит обращение к ВУ, при низком — к памяти. Сигналы S_1, S_0 и IO/M совместно идентифицируют тип машинного цикла;

- READY — входной сигнал, показывающий, что память или ВУ готовы к обмену с МП. Если готовности памяти или ВУ нет, то МП входит в состояние ожидания, которое может длиться любое число тактов вплоть до появления единичного уровня сигнала READY;
- ALE — строб разрешения загрузки младшего байта адреса памяти во внешний регистр для его хранения в течение машинного цикла. Появляется в первом такте машинного цикла. Регистр загружается задним фронтом сигнала ALE;
- HOLD — сигнал запроса захвата шин. Формируется внешним устройством;
- HLDA — сигнал подтверждения захвата (Hold Acknowledge) шин. Является ответом на сигнал HOLD, формируемым в конце текущего машинного цикла. Свидетельствует об отключении МП от системных шин. При этом шины и линии управляющих сигналов $\overline{\text{RD}}$, $\overline{\text{WR}}$, IO/M и ALE переводятся в третье состояние;
- $\overline{\text{RESIN}}$ ($\overline{\text{RESET IN}}$) — вход сигнала сброса МП в начальное состояние. Сигнал может поступить в любое время по команде оператора. Автоматически формируется при включении питания. Под его воздействием сбрасываются регистры PC и IR, триггеры разрешения прерывания, подтверждения захвата и др.;
- RESET — выходной сигнал сброса для внешних модулей системы, привязанный к тактовым импульсам CLK, т. е. отличающийся от сигнала $\overline{\text{RESIN}}$ по фазе.

Исключения и прерывания

При работе МПС в ней или вне ее могут произойти события, требующие немедленной реакции — прерывания выполняемой программы и перехода к обслуживанию возникшей ситуации. Внутри системы особые ситуации возникают при сбоях в работе, переполнении разрядной сетки, попытке деления на нуль и т. д. Ситуации подобного типа, связанные с ошибками в работе процессора, называются *исключениями*. *Аппаратными прерываниями* называют штатные ситуации, в которых прерывания запрашиваются внешними сигналами. Если же запрос формируется командами программы, то говорят о *программных прерываниях*.

Аппаратные прерывания возникают, в частности, при требованиях обслуживания от внешних устройств. Извне могут поступать также сигналы об аварийных ситуациях в управляемых объектах, неисправностях источников питания и др.

Аппаратные прерывания увеличивают производительность системы, позволяя внешним устройствам занимать время процессора только при их готовности к обмену. Когда ВУ нуждается в обслуживании, оно устанавливает триггер запроса прерывания, и сигнал запроса сохраняется, пока не будет воспринят и обработан процессором. В ответ на принятый запрос завершается выполнение текущей команды, запоминается состояние МП, выполняется подпрограмма обслуживания прерывания, восстанавливается состояние МП, и затем возвращается управление очередной команде основной программы.

При организации прерываний решаются задачи маскирования запросов и определяются их уровни приоритета при конфликтах из-за одновременного поступления нескольких запросов.

Маскирование состоит в запрещении действия соответствующего входа. Входы запросов прерывания могут быть маскируемыми или не маскируемыми, т. е. принимаемыми всегда. Маскируемые прерывания могут быть разрешены или запрещены командами `EI` (Enable Interrupt) и `DI` (Disable Interrupt), действующими на все маскируемые входы одновременно. Начальный сброс МП запрещает обслуживание маскируемых запросов, для их последующего разрешения следует ввести команду `EI`. Зачастую имеется также возможность раздельного маскирования запросов с помощью специальной команды.

Приоритеты входов вводятся для решения вопроса о том, какому запросу предоставить обслуживание при наличии одновременно нескольких. Простейший вариант введения приоритетности — присвоение входам фиксированных неизменных приоритетов.

Различают *радиальные* и *векторные* прерывания. При обслуживании радиальных прерываний источник выставляет запрос, обслуживание которого не требует передачи процессору сведений о начальном адресе подпрограммы, поскольку ее адрес известен (адрес присвоен данному входу, для запросов типа RSTn он равен 8n). При векторных прерываниях адрес подпрограммы обслуживания должен быть сообщен процессору (процессору передается *вектор* прерываний — информация, необходимая для перехода к соответствующей подпрограмме обслуживания, в простейшем случае это просто начальный адрес прерывающей подпрограммы).

Входы запросов могут быть *статическими* или *динамическими*. Статические реагируют на уровень сигнала и, следовательно, запросы автоматически снимаются при исчезновении активного уровня входа. Динамические реагируют на фронт сигнала запроса, запоминая запрос в триггере с динамическим входом. Такой запрос сохраняется, пока не будет обработано прерывание или до специальной команды или же до сигнала RESET.

Прерывание должно ввести в действие команду `CALL`, согласно которой текущее состояние программного счетчика PC передается в стек, а в PC загружается адрес подпрограммы, подлежащей выполнению. Инициатива ввода команды `CALL` принадлежит аппаратным средствам МПС. Если прерывания разрешены, то они осуществляются процессором в конце выполнения текущей команды.

Блок управления прерываниями

Рассматриваемый МП имеет четыре входа прерывания (см. рис. 6.4), а именно:

- TRAP — вход запроса немаскируемого прерывания, имеющий максимальный приоритет. Он не может быть запрещен командами программы. К этому входу подключают сигналы, оповещающие о наиболее важных событиях в МПС, появление которых требует безусловной реакции (например, сигнал, оповещаю-

щий об аварии питания). Прерывание радиальное, начальный адрес подпрограммы его обслуживания размещен в фиксированной ячейке памяти. Таким образом, появление запроса прерывания по входу TRAP независимо ни от чего вызовет соответствующее прерывание после выполнения текущей команды.

- RSTn1; RSTn2 — входы запросов радиального прерывания типа RSTn. Начальные адреса подпрограмм обслуживания известны (равны 8n). Приоритеты фиксированы. Приоритеты этих запросов выше приоритета запроса INTR. Запросы маскируемые, в том числе и независимо друг от друга.
- INTR (Interrupt Request) — вход запроса векторного прерывания, вызывающий генерацию строба $\overline{\text{INTA}}$, если прерывание разрешено программой. При поступлении запроса по входу INTR вектор прерывания должен быть передан в МП извне. К этому входу, в частности, подключают контроллер прерываний — блок, который воспринимает несколько запросов от внешних устройств, решает задачи учета их приоритетности и маскирования, вырабатывает для МП единственный сигнал запроса INTR и пересылает в МП соответствующий вектор прерывания по сигналу процессора $\overline{\text{INTA}}$.
- $\overline{\text{INTA}}$ (Interrupt Acknowledge) — выход строба подтверждения векторного прерывания, формируется после завершения процессором текущего командного цикла. Используется для чтения вектора прерывания. Сигнал реализуется как последовательность трех импульсов, если вводимый вектор прерывания содержит три байта.

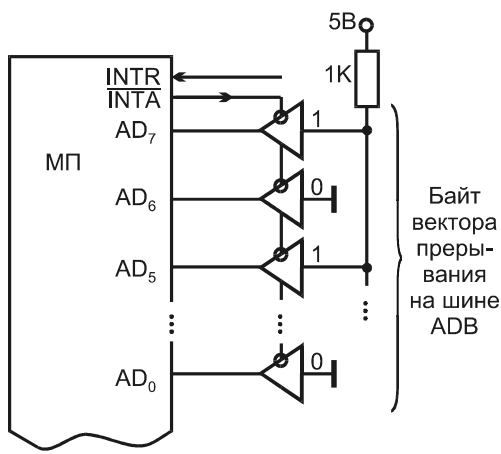


Рис. 6.7. Аппаратная реализация пересылки байта при выполнении операции рестарта

- Аппаратный ввод байта может быть реализован, например, согласно рис. 6.7. Появление запроса INTR при разрешенных прерываниях ведет к ответу микропроцессора сигналом $\overline{\text{INTA}}$, во время действия которого на шине AD появляется вводимый байт. Сигнал $\overline{\text{INTA}}$ поступает на входы разрешения выхода буферных усилителей $\overline{\text{OE}}$.

§ 6.3. Функционирование микропроцессора

При естественном следовании команд МП, начав работу, выбирает из памяти и выполняет одну команду за другой, пока не дойдет до команды "Останов" (HALT). Процессы выполнения команд подчиняются строгой последовательности действий.

Синхронизация и последовательность действий МП

Циклы и такты. Выборка и выполнение одной команды образуют *командный цикл* КЦ. Командный цикл состоит из одного или нескольких *машинных циклов* МЦ. В свою очередь машинный цикл делится на то или иное число *тактов* Т, число которых зависит от типа машинного цикла (рис. 6.8, *а*).

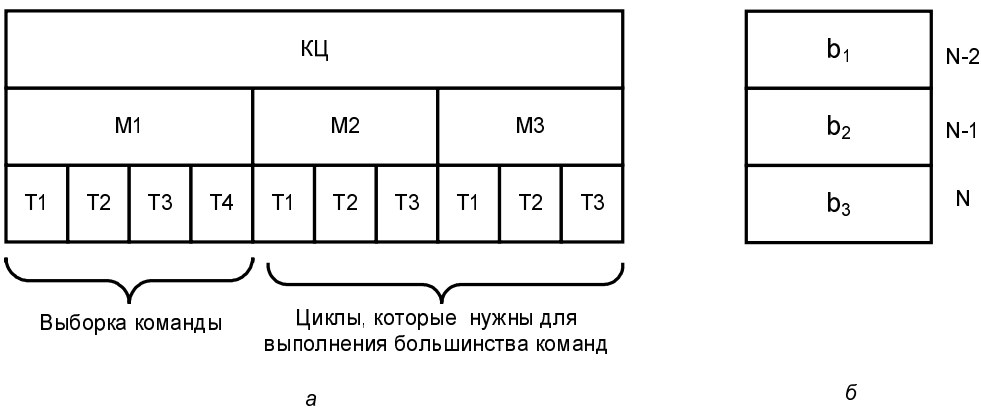


Рис. 6.8. Циклы и такты микропроцессора (*а*) и пример размещения команды в памяти микропроцессорной системы (*б*)

Типы машинных циклов. Микропроцессор имеет следующие типы машинных циклов:

1. Выборки кода команды (OF, Opcode Fetch).
2. Чтения из памяти (MR, Memory Read).
3. Записи в память (MW, Memory Write).
4. Чтения из ВУ (IOR, Input-Output Read).
5. Записи в ВУ (IOW, Input-Output Write).
6. Подтверждения прерывания (INA, Interrupt Acknowledge).
7. Освобождения шин (BI, Bus Idle).
8. Останов (HALT).

Каждое обращение к памяти или ВУ требует машинного цикла для передачи байта в МП или из него. Обращает на себя внимание отсутствие машинных циклов, связанных с выполнениями таких операций, как суммирование, вычитание, сдвиг и т. п. Это объясняется тем, что подобные операции, будучи внутренними, т. е. не требующими обращений к модулям МПС по системным шинам, выполняются быстро, и для них достаточно добавлений отдельных тактов в машинные циклы, предназначенные для передач данных.

В начале каждого машинного цикла генерируются сигналы состояния, идентифицирующие тип цикла и действующие в течение всего цикла. Сигналы, реализующие тот или иной МЦ, генерируются блоком управления МП на основании информации, содержащейся в первом байте команды.

Командный цикл КЦ начинается с выборки команды. Первый машинный цикл M1 всегда цикл выборки команды (OF) из памяти, в нем МП получает первый байт команды. После этого могут быть еще один или два машинных цикла типа MR (Memory Read), поскольку команда может быть однобайтной, двухбайтной или трехбайтной. Если команда трехбайтная, то она хранится в памяти так, как показано на рис. 6.8, *б*.

После выборки и декодирования команды могут понадобиться дополнительные машинные циклы для ее выполнения. Всего в командном цикле может быть от одного до пяти машинных циклов.

Машинный цикл состоит из тактов, в которых выполняются типовые действия. Число тактов в различных машинных циклах — 3...6. Большинство машинных циклов содержат три такта. В командном цикле может содержаться от 4 до 18 тактов.

Такты (состояния) — интервалы между одноименными фронтами тактовых импульсов.

Работа автомата управления. Типовые действия, выполняемые в тактах. Автомат управления МП меняет состояние от такта к такту. Такты машинных циклов предназначены для выполнения типовых действий. Каждый из них посвящен определенным операциям.

Машинный цикл всегда содержит такты T_1....T_3, иногда имеет большее число тактов, но для чтения или записи требуется только три такта.

Упрощенный граф переходов автомата управления МП показан на рис. 6.9. Упрощение состоит в неполном отображении реакций автомата на запросы захвата шин (HOLD), прерываний и признака команды останова `HALT`.

Состояние TR. Сигнал сброса RESET от кнопки или включения питания приводит автомат в исходное состояние сброса T_R, в котором регистры и триггеры-флажки принимают предписанные им состояния. В частности, сбрасывается программный счетчик и регистр команд (код 00 в регистре команд соответствует операции `NOP`, т. е. "нет операции"), устанавливаются маски запросов прерывания, сбрасываются триггеры-флажки запросов прерываний и захвата шин и т. д.

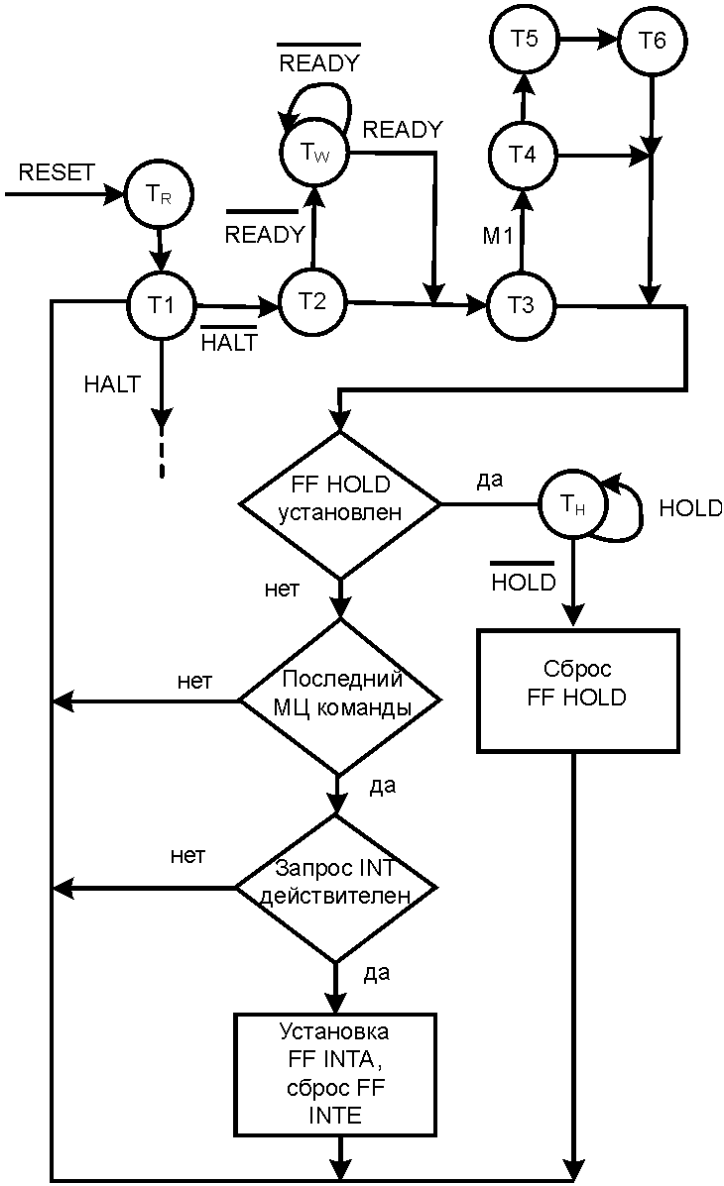


Рис. 6.9. Упрощенный граф переходов автомата управления МП

Состояние T1. Из состояния T_R автомат переходит в состояние T1, в котором происходят следующие процессы:

- адрес памяти или ВУ выставляется на шинах A_{15-8} и AD_{7-0};
- генерируется сигнал ALE для фиксации битов AD_{7-0} во внешнем регистре;
- на линиях IO/M, S1 и S0 выставляется информация о типе цикла;
- проверяется флаг HALT.

Состояние T2. Из состояния T1 автомат переходит в состояние T2 (рис. 6.9 упрощен, выполнение команды останова на нем не показано, поэтому переход из состояния T1 осуществляется по условию $\overline{\text{HALT}}$). Действия в такте T2:

- проверяется вход READY готовности партнера к обмену;
- программный счетчик инкрементируется, если данный машинный цикл выбирает команду;
- во всех машинных циклах кроме цикла освобождения шин один из управляющих стробов $\overline{\text{RD}}$, $\overline{\text{WR}}$ или $\overline{\text{INTA}}$ переходит в нулевое состояние.

Состояние T_W. Отсутствие сигнала READY вводит такты ожидания T_W. Сигнал READY проверяется в каждом такте ожидания. Такты ожидания вставляются до появления сигнала READY. Состояния линий адресов, данных и управления остаются теми же, что и в конце такта T_2.

Состояние T3. В такте T3 байт команды или данных передается в микропроцессор или из него задним фронтом управляющего строба ($\overline{\text{RD}}$, $\overline{\text{WR}}$ или $\overline{\text{INTA}}$). Управляющий строб возвращается в пассивное состояние (на высокий уровень напряжения).

Состояние T4. Если данный машинный цикл является циклом M1, то за состоянием T3 следует состояние T4, необходимое для декодирования принятой процессором команды. Системные шины в этом такте не используются.

Далее возможны два варианта — МЦ из четырех или из шести тактов.

Состояния T5 и T6. Пятый и шестой такты вводятся при необходимости для завершения некоторых команд, в которых после их декодирования нужно произвести какие-либо внутренние действия. Системные шины не используются. Выполняются операции, внутренние для МП.

Затем проверяется флажок захвата шин, и если он установлен, то реализуется переход к состоянию T_H на время существования сигнала HOLD. При сброшенном флажке захвата шин определяется, является ли данный МЦ последним машинным циклом команды. Если этого нет, то начинается следующий машинный цикл, для чего автомат возвращается в состояние T1. Если же данный МЦ завершает команду, то анализируется наличие или отсутствие подлежащих выполнению запросов прерывания INT. Если такие запросы присутствуют, то устанавливается флажок подтверждения прерывания $\overline{\text{INTA}}$ и сбрасывается триггер разрешения прерывания INTE (Interrupt Enable). После обработки запроса прерывания идет возврат к состоянию T1.

Пример временных диаграмм машинного цикла. Временные диаграммы цикла чтения с тактом ожидания приведены на рис. 6.10.

В первом такте цикла показаны сигналы, обеспечивающие формирование адреса ячейки памяти или внешнего устройства. Во втором такте обнаруживается отсутствие готовности партнера к обмену, приводящее к введению такта ожидания. В такте ожидания появляется сигнал готовности, разрешающий переход к такту T3, в котором положительным фронтом строба чтения $\overline{\text{RD}}$ данные вводятся в процессор.

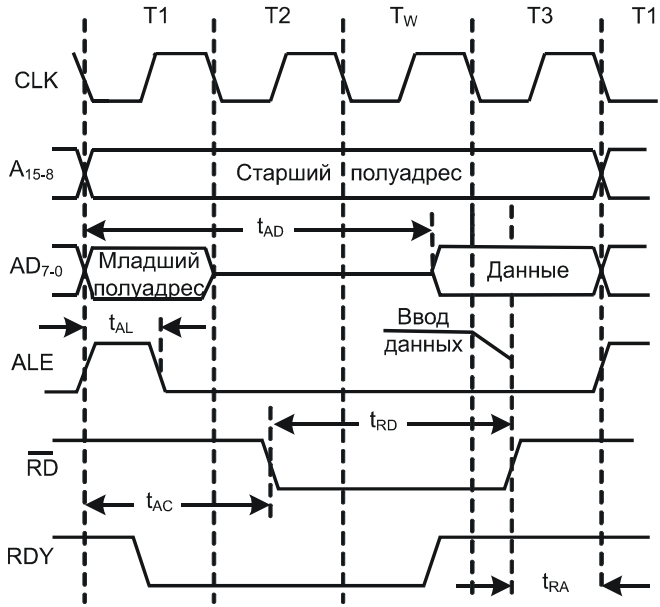


Рис. 6.10. Временные диаграммы цикла чтения микропроцессора

Интервалы действия тех или иных сигналов жестко определяются временными диаграммами машинных циклов. На рис. 6.10 отмечены некоторые из этих интервалов:

- интервал t_{AD} определяет время от момента выставления адреса до наиболее позднего допустимого момента появления действительных данных на шине $AD_{7\text{-}0}$. Величина t_{AD} задается соотношением вида

  $t_{AD}(N) = (5/2 + N)T - k$,

  где T — длительность такта, N — число тактов ожидания в цикле чтения и k — постоянная величина, составляющая долю такта. Если неравенство удовлетворяется при N = 0, то возможна работа без тактов ожидания. Иначе требуется ввести столько тактов ожидания, сколько потребуется для удовлетворения неравенства. При отсутствии тактов ожидания, как видно из формулы, данные должны быть готовы несколько раньше (на время k), чем наступит середина такта T3 (момент их чтения);
- интервал t_{AL} отмечает момент загрузки байта адреса во внешний регистр;
- интервал t_{AC} определяет время от момента выставления адреса до появления активного уровня управляющего строба $\overline{RD}$;
- интервал t_{RD} задает длительность строба $\overline{RD}$;
- интервал t_{RA} определяет время от момента чтения данных до появления нового значения адреса (за это время старые данные должны быть сняты с линий $AD_{7\text{-}0}$).

Адресные пространства, способы адресации, форматы команд

Адресные пространства. Совокупность адресов, которые могут быть использованы процессором, образует *адресное пространство* МПС. Адреса памяти могут занимать все адресное пространство (АП) или его часть, а линейно организованная память независимо от ее технической реализации может быть условно представлена набором регистров (ячеек), число которых M, а разрядность — N (рис. 6.11).

Диапазон адресов, к которым может обращаться процессор (т. е. емкость АП) связан с разрядностью шины адреса m соотношением $АП = 2^m$. Например, с помощью 16-разрядной шины адреса можно адресовать $2^{16} = 64К$ объектов, с помощью 20-разрядной 1М объектов и т. д.

АП используется блоками ОЗУ, ПЗУ и ВУ, к которым обращается процессор. Распределение АП между указанными претендентами производится проектировщиком системы, имеющим известную свободу действий, хотя у конкретных процессоров могут быть особенности, заставляющие отдавать определенную область АП для адресации определенных объектов.

Для оценки емкостей АП используются обычно единицы измерения $К = 2^{10} = 1024$, $М = 2^{20}$, $Г = 2^{30}$. Адреса в АП обычно выражают в шестнадцатеричной системе счисления. Так, например, диапазон адресов в АП с емкостью 64К будет записан как 0000h…FFFFh.

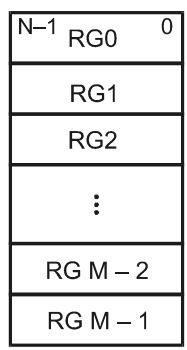


Рис. 6.11. Условное представление линейной организации памяти

Адреса внешних устройств могут размещаться в отдельном адресном пространстве или же занимать часть общего. Выбор адресованной ячейки памяти или ВУ осуществляется схемами декодирования адреса. При управлении памятью и ВУ процессор должен вначале сформировать нужный адрес, который затем декодируется аппаратными средствами.

Способы адресации. В МПС применяют несколько способов формирования адресов. Для систем с простыми процессорами характерны способы адресации, рассмотренные далее (рис. 6.12, *а*).

При *прямой адресации* адрес операнда содержится в команде, подлежащей выполнению. Прямая адресация удобна, но удлиняет команды (увеличивает их разрядности), причем существенно, т. к. при значительных емкостях памяти разрядности адресов достаточно велики.

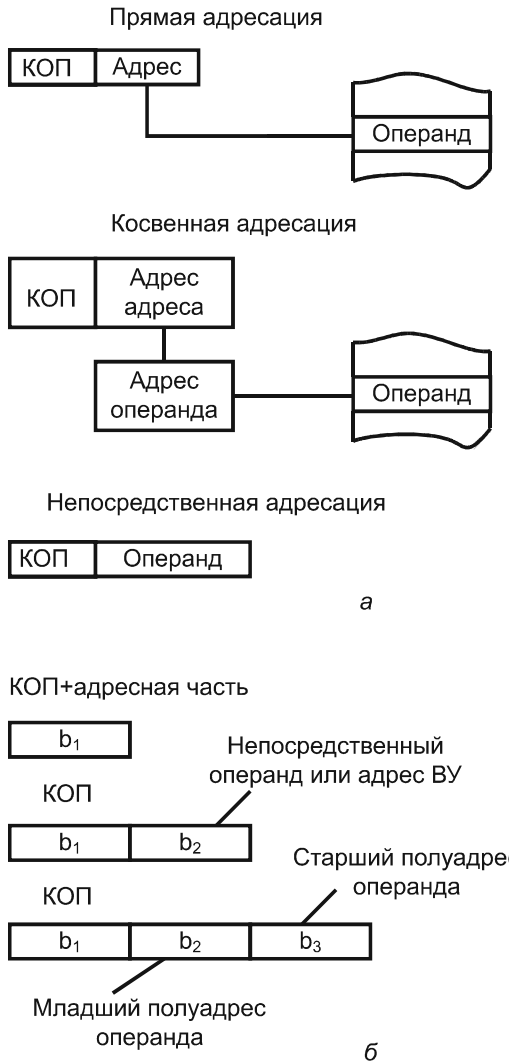


Рис. 6.12. Способы адресации в системах с простыми процессорами (*а*) и форматы команд МП (*б*)

При *прямой регистровой* адресации, когда операнд находится в одном из внутренних регистров процессора, адрес является малоразрядным, поскольку число внутренних регистров мало. В этом случае прямая адресация сочетается с компактностью команд и проявляет все свои достоинства. В частности, для адресации 8

регистров общего назначения достаточны трехразрядные адреса, а для адресации 4-х регистровых пар даже двухразрядные.

При *косвенной адресации* в команде явно или неявно указывается регистр процессора, содержащий адрес операнда (можно сказать, что в команде указывается или подразумевается короткий адрес адреса). Команда компактна, но для ее выполнения требуется предварительная настройка — загрузка регистра косвенного адреса (индексного регистра). Косвенная адресация особенно удобна при обработке списков, когда настройка производится однократно, а очередной адрес получается модификацией предыдущего (изменением его на единицу). В зависимости от действий, которые производятся над содержимым регистра при формировании адреса, различают несколько видов косвенной адресации. В частности, при *простой косвенной адресации* обращение к ячейке памяти производится по адресу, находящемуся в индексном регистре, т. е. никаких действий над его содержимым не производится. При *относительной косвенной адресации* адрес ячейки памяти получается суммированием содержимого индексного регистра и числа, задаваемого в команде.

При *непосредственной адресации* в команде содержится сам операнд.

Использование различных видов адресации сокращает объем и время выполнения программ.

Форматы команд. Команда может быть однобайтной, двухбайтной или трехбайтной (рис. 6.12, *б*). Первый байт содержит код операции КОП, сведения о способе адресации, а если команда однобайтная, то и адрес операнда. Размещение адреса в однобайтной команде возможно для операций типа "регистр-регистр" с короткими адресами. Второй байт двухбайтной команды содержит непосредственный операнд либо адрес ВУ, а второй байт трехбайтной команды — младший полуадрес операнда. Третий байт команды содержит старший полуадрес операнда или байт непосредственных данных при загрузке пары регистров.

Тип команды может неявно подразумевать способ адресации. Так, например, в прототипе рассматриваемого МП ссылка на память (участие в выполняемой операции операнда, находящегося в ячейке памяти) предусматривает косвенную адресацию с чтением адреса из регистровой пары H.

О системе команд

Команды, выполняемые процессором и применяемые при написании программ, разбиваются на группы. Основные группы составляют следующие команды:

- пересылки;
- арифметических и логических операций;
- управления;
- специальные.

При описании команд даются их мнемокоды, машинные коды, длины команд в байтах, числа машинных циклов и тактов, необходимые для выполнения команды.

В мнемокодах команд регистры обозначаются буквой r, пары регистров через r_p, ячейки памяти как M, байты команды как b_i (i = 1, 2, 3).

> ***Примечание***
>
> Далее представлены лишь краткие сведения о командах МП. Подробнее раскрываются только некоторые команды, которые встретятся дальше при рассмотрении примеров функционирования процессора.

Команды пересылки предусматривают передачи данных между регистрами; между регистрами и ячейками памяти; передачу непосредственных данных (находящихся в команде) в регистр, пару регистров или ячейку памяти; обмен между парами регистров.

Арифметические и логические операции выполняются командами сложения, вычитания, инкрементирования и декрементирования, преобразования данных в двоично-десятичный код, логических операций, сравнения и инвертирования данных, циклических сдвигов содержимого аккумулятора.

Команды управления включают в свой состав команды переходов, вызовов подпрограмм, возвратов в основную программу, передачи содержимого пары регистров в стек.

Специальные команды связаны с операциями со стеком, загрузкой программного счетчика, вводом/выводом данных, прерываниями, остановом МП и введением пустых циклов `NOP`. Примеры команд приведены в табл. 6.1.

Команды, в машинных кодах которых имеются буквенные обозначения источников и приемников данных, являются обобщенными. Подставляя вместо буквенных обозначений определенные адреса, получают коды конкретных вариантов команды. Например, для команды пересылки из регистра в память `MOV M, r` при подстановке вместо буквенного обозначения источника данных ИИИ адреса 001, принадлежащего регистру С, получим конкретный вариант команды с кодом 01110001, определяющий пересылку содержимого регистра С в ячейку памяти при косвенной адресации, т. е. по адресу, содержащемуся в регистровой паре Н. В кодах обобщенных команд с индексом "усл" встречаются буквенные обозначения УУУ условий, при выполнении которых осуществляется указанная в команде операция. Эти условия также определяются в описании системы команд (например, 000 — неравенство нулю, что обозначается как NZ, 001 — равенство нулю, что обозначается как Z, и т. д.). Так, в частности, команда условного перехода из обобщенной формы `Jусл b3b2` переводится в вариант `JZ b3b2` с адресом b_3b_2, если признак результата свидетельствует о его нулевом значении. Признаки формируются в регистре флажков в виде полного либо частичного набора при выполнении команд арифметических и логических операций.

Таблица 6.1

Мнемокод	Первый байт	Число			Содержание
		байтов	тактов	циклов	
Примеры команд пересылки (всего около 15 команд)					
MOV M,r	01110ИИИ	1	7	2	Пересылка из регистра в память
MOV r,M	01ППП110	1	7	2	Пересылка из памяти в регистр
LXI rpb3b2	00ПР0001	3	10	3	Загрузка непосредственных данных в пару регистров
STA b3b2	00110010	3	13	4	Прямая запись аккумулятора в память
Примеры команд арифметических и логических операций (всего около 40 команд)					
ADD r	10000ИИИ	1	4	1	Сложение регистра и аккумулятора
ADI b2	11000110	2	7	2	Сложение непосредственных данных и аккумулятора
ANI b2	11100110	2	7	2	Логическое И непосредственных данных и аккумулятора
INX rp	00ПР0011	1	6	1	Инкрементирование пары регистров
Примеры команд управления (всего около 10 команд)					
JMP b3b2	11000011	3	10	3	Безусловный переход
Jусл b3b2	11УУУ010	3	10	3	Условный переход
Примеры специальных команд (всего около 15 команд)					
PUSH rp	11РП0101	1	11	3	Пересылка пары регистров в стек
POP rp	11РП0001	1	10	3	Загрузка регистровой пары из стека
IN port	11011011	2	10	3	Ввод данных в порт
OUT port	11010011	2	10	3	Вывод данных из порта
EI	11111011	1	4	1	Разрешение прерываний
DI	11110011	1	4	1	Запрещение прерываний
HLT	01110110	1	7	2	Останов
NOP	00000000	1	4	1	Нет операции

Пример выполнения команды и фрагмента программы

Проиллюстрируем работу процессора примерами выполнения отдельной команды и фрагмента программы.

Пример выполнения команды. Рассмотрим выполнение команды `ADI b2` (Add Immediate b_2) сложения содержимого аккумулятора с непосредственным операндом. Команда двухбайтная, для ее передачи в МП требуются два машинных цикла, в первом из которых байт b_1, содержащий код операции, передается в регистр команд IR, а во втором байт b_2 передается в регистр временного хранения TR. После получения всей команды МП выполняет ее, суммируя b_2 с содержимым аккумулятора и размещая результат в аккумуляторе. Таким образом, цикл команды составится из двух машинных циклов в следующем порядке OF-MR.

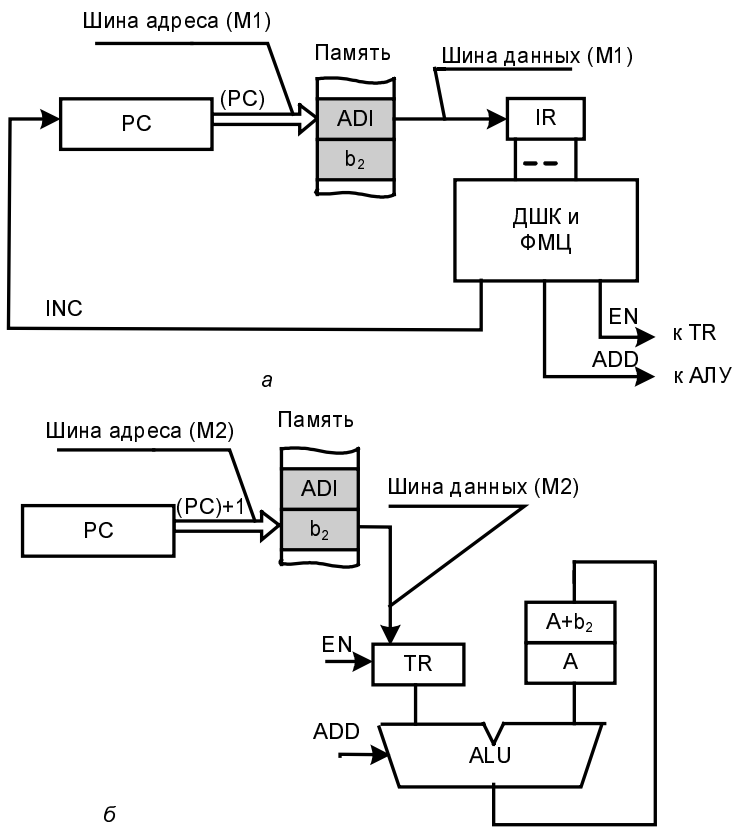


Рис. 6.13. Последовательность действий МП при выполнении команды ADI b_2

Функционирование МП при выполнении команды `ADI b2` иллюстрируется на рис. 6.13. Первый машинный цикл M1 (рис. 6.13, *а*) начинается с обращения по

шине адреса в память за первым байтом команды. Его адрес (PC) находится в программном счетчике PC (величины, заключенные в скобки, означают содержимое соответствующих регистров). Из адресованной ячейки памяти по шине данных первый байт команды поступает в регистр команд IR. Код операции передается на дешифратор команд и формирователь машинных циклов ДШК и ФМЦ, который выдает сигнал INC, увеличивающий содержимое программного счетчика на единицу, и настраивает АЛУ на выполнение операции сложения `ADD`, а регистр временного хранения TR на прием данных (сигнал EN).

Код операции указывает на то, что она выполняется с непосредственной адресацией, и, следовательно, операнд расположен во втором байте команды, т. е. по адресу (PC) + 1. В следующем машинном цикле M2 (рис. 6.13, *б*) по шине адреса идет обращение к ячейке памяти, из которой по шине данных операнд передается в регистр TR. По окончании операции сложения результат фиксируется в аккумуляторе.

Пример выполнения фрагмента программы. Управление модулями реализуется в МПС сигналами, формируемыми процессором в шинах адресов, данных и управления. Процессор расшифровывает команды и задает на шинах такие последовательности потенциалов, которые заставляют управляемые им блоки выполнять требуемые действия.

Для иллюстрации рассмотрим выполнение короткого фрагмента программы — передачу байта из одной ячейки памяти в другую. Пусть численное значение байта будет 10h, а его передача производится из ячейки 0100h в ячейку 0101h. Пусть также фрагмент программы размещается в памяти, начиная с ячейки 2000h (рис. 6.14).

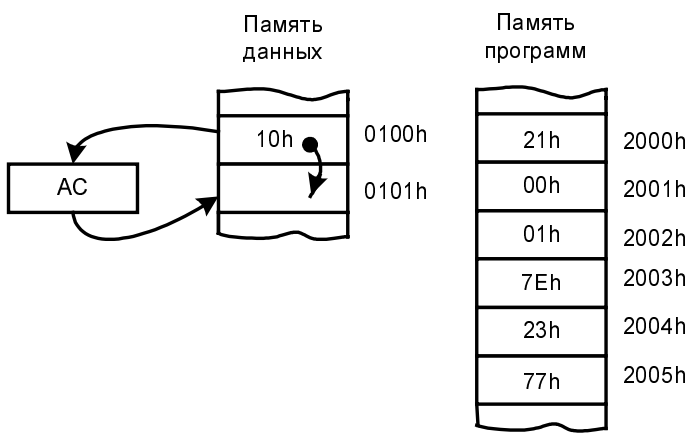


Рис. 6.14. Иллюстрации к процессу выполнения фрагмента программы

Для выполнения фрагмента сначала нужно переслать байт из памяти в аккумулятор, а затем из аккумулятора в другую ячейку памяти. Так как обращение к памяти подразумевает косвенную адресацию, вначале требуется загрузка пары регистров H адре-

сом ячейки, к которой идет обращение. С учетом сказанного фрагмент программы в мнемокодах (на ассемблере МП) будет иметь следующий вид:

```
LXI rpb3b2      (команда загрузки непосредственных данных в пару регистров)
MOV r, M  (пересылка из памяти в регистр)
INX rp    (инкрементирование пары регистров)
MOV M, r  (пересылка из регистра в память)
```

В полученном фрагменте использованы обобщенные команды из числа приведенных в табл. 6.1. В нашем случае регистром r является аккумулятор A, а парой r_p — пара H, поэтому фрагмент программы конкретизируется следующим образом:

```
LXI H, b3b2
MOV A, M
INX H
MOV M, A
```

Чтобы проиллюстрировать выполнение фрагмента временными диаграммами сигналов в шинах системы, запишем его в машинных кодах. Команда загрузки непосредственных данных в пару регистров имеет код 00ПР0001, где ПР — адрес пары регистров. Пара регистров H имеет адрес 10. Подставив это значение в код команды, получаем код 00100001, т. е. 21h. Загружаемый адрес согласно условиям задачи равен 0100h, причем после кода операции из памяти извлекаются сначала младший (00), а затем старший (01) байты адреса. Команда трехбайтная и занимает ячейки памяти 2000h...2002h.

Однобайтная команда `MOV A, M` пересылки из памяти в аккумулятор записывается в виде 01ПППП110, где ППП — адрес аккумулятора. Подставив в этот код адрес аккумулятора 111, получаем код команды 01111110, т. е. 7Eh.

Команда `INX H` прибавляет единицу к содержимому регистровой пары и получается из кода 00ПР0011 при подстановке адреса пары регистров 10, что дает код 00100011, т. е. 23h.

Последняя команда фрагмента (пересылка из аккумулятора в память) `MOV M, A` передает в ячейку памяти, адрес которой находится в регистровой паре H, содержимое аккумулятора. Эта команда имеет код 01110ИИИ. Подстановка адреса аккумулятора, являющегося в данном случае источником данных, дает код команды 01110111, т. е. 77h. Эта команда завершает выполнение фрагмента программы.

Команда в кодах имеет вид:

```
2000 21 00 01
2003 7E
2004 23
2005 77
```

Фрагмент программы, записанный в машинных кодах, наглядно указывает информацию, появляющуюся на шинах МПС. Последовательность четырех рассмотрен-

ных команд сгенерирует временные диаграммы (рис. 6.15), посредством которых программа будет выполнена.

Сведения о длительностях выполнения команд позволяют оценивать и времена выполнения линейных участков программ.

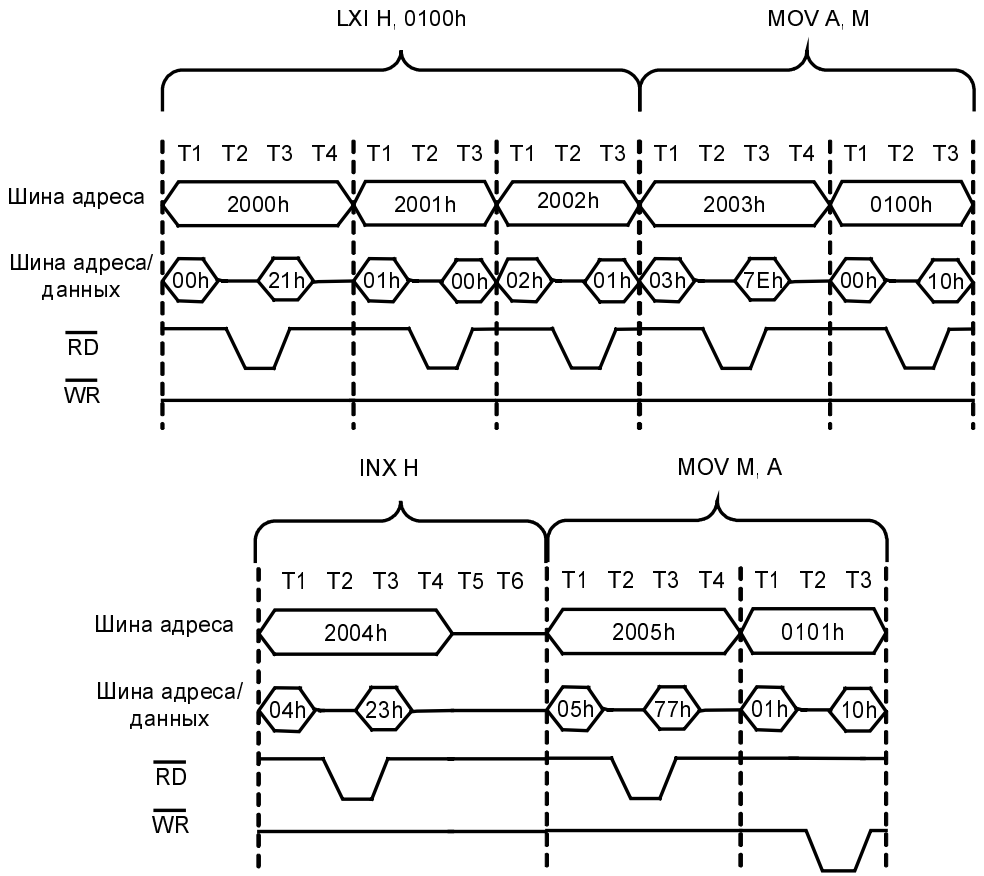


Рис. 6.15. Реализация программы сигналами шин микропроцессорной системы

§ 6.4. О развитии микропроцессорной техники

Первый микропроцессор Intel 4004 появился в 1971 году, что дало толчок бурному развитию микропроцессорной техники. С тех пор производительность и некоторые другие параметры микропроцессоров улучшились на шесть порядков. Движущими силами столь впечатляющего развития служат все возрастающие потребности практики. Обеспечивает совершенствование микропроцессоров ряд архитектурных, схемотехнических и технологических достижений.

По принципу общности или раздельности памяти программ и данных, как уже говорилось, различают МП Принстонской и Гарвардской архитектуры. По характеру системы команд различают *CISC*-, *RISC*- и *VLIW-процессоры*.

CISC-процессоры

Процессоры CISC имеют так называемую сложную систему команд (CISC — Complex Instruction Set Computer), т. е. большой набор разноформатных команд при использовании многих способов адресации. Архитектура CISC присуща классическим процессорам, она в силу многообразия команд позволяет применять эффективные алгоритмы решения задач, но, в то же время, усложняет схему процессора и его стоимость и в общем случае не обеспечивает его максимального быстродействия.

RISC-процессоры

RISC-процессоры имеют сокращенную систему команд (RISC — Reduced Instruction Set Computer), из которой исключены редко применяемые команды. Форматы команд (по крайней мере подавляющего их большинства) идентичны (например, все команды содержат по 4 байта), резко снижено число используемых способов адресации. Данные, как правило, обрабатываются только с регистровой или непосредственной адресацией. Значительно увеличенное число регистров процессора, т. е. его емкая внутренняя память, позволяет редко обращаться по системным шинам к модулю памяти, а это повышает быстродействие системы. Идентичность временных циклов выполнения команд отвечает потребностям конвейерных схем. В результате может быть достигнуто упрощение схемы процессора при увеличении его быстродействия.

VLIW-процессоры

Последними по времени появления стали VLIW-процессоры (VLIW — Very Long Instruction Word), особенность которых состоит в использовании очень длинных команд (16 и более байт). Отдельные поля длинной команды определяют несколько подлежащих реализации операций, которые могут выполняться параллельно во времени в нескольких операционных устройствах процессора. Таким образом, одна длинная команда определяет сразу группу операций. VLIW-процессоры считаются перспективными для высокопроизводительных процессоров.

Направления развития МП

Эволюционное развитие микропроцессоров, направленное главным образом на повышение их производительности, обеспечивалось и обеспечивается несколькими факторами, в первую очередь следующими:

- применением конвейеров, т. е. разбиением выполняемых операций на отдельные этапы, которые позволяют переходить к выполнению этапов над следую-

щими операндами сразу же после освобождения ступени конвейера от обработки предыдущих, иначе говоря, не дожидаясь конца всей операции над предыдущими операндами;

- размещением кэш-памяти первого уровня на одном кристалле с процессором, что позволяет ускорить обмен процессора с кэшем, а также кэшированием как памяти данных, так и памяти команд;
- введением в структуру процессора предсказателей ветвления программы, повышающих эффективность кэш-памяти путем снижения процента промахов кэша;
- реализацией на одном кристалле с основным процессором вспомогательных специализированных процессоров (сопроцессоров), прежде всего для обработки данных с плавающей запятой, введением в структуру процессора дополнительных аппаратных средств реализации важных для решаемых задач специальных функций;
- выполнением команд не в порядке их следования в программе, а по мере готовности данных и ресурсов для данной команды с последующим учетом фактической последовательности выполнения команд;
- делением исполнительного ядра процессора на области (кластеры) с концентрацией по возможности операций внутри одного и того же кластера и минимизацией распространения сигналов на большие расстояния по кристаллу, поскольку времена распространения сигналов в пространстве уже становится причиной ограничения быстродействия микропроцессоров;
- применением для внутренних исполнительных устройств процессора повышенной частоты тактирования с целью исключить возможные интервалы ожидания со стороны других устройств;
- применением многоядерных структур.

Перечисленные и новые намечающиеся способы повышения производительности микропроцессоров ведут к усложнению их структур. Изучение микропроцессоров в полном объеме является предметом специальных курсов, которым посвящен ряд зарубежных и отечественных работ (см. *перечень литературы в конце главы*).

По мере развития микропроцессорной техники происходит естественный процесс специализации МП соответственно областям их применения. Наряду с микропроцессорами общего назначения стал интенсивно развиваться класс проблемно ориентированных МП — *процессоры цифровой обработки сигналов*, которые находят применение в современных системах связи, обработки графических изображений, медицине и многих других областях. Особое место среди микропроцессоров занимают *микроконтроллеры* (см. *главу 8*). Микроконтроллеры также постепенно специализируются, так, например, среди них выделился класс *коммуникационных* контроллеров.

§ 6.5. Управление памятью и внешними устройствами

Для обращения к памяти или ВУ на шине адреса формируется адресный код, соответственно которому схема адресации вырабатывает сигналы, обеспечивающие доступ к ячейке памяти или ВУ.

Абсолютная и неабсолютная адресации

Адресация может быть *абсолютной* или *неабсолютной*. При абсолютной адресации обратиться к ячейке памяти или ВУ можно только по одному-единственному адресу. При этом разрядность адреса равна $\log_2 N$, где N — объем адресного пространства (например, для АП, содержащего 64К = 2^{16} адресов, разрядность адреса составит 16).

При неабсолютной адресации для ячейки памяти или ВУ выделяется некоторая зона адресов. Число таких зон в пределах адресного пространства будет меньше, чем число отдельных адресов, поэтому для указания зоны потребуется меньшая разрядность адреса (например, выделив для каждой ячейки памяти в АП емкостью 64К зону из 16 адресов, получим 2^{12} зон, адресация которых требует 12-разрядных адресов). Иными словами, абсолютная адресация требует полного декодирования адреса, а неабсолютная — частичного, что упрощает схемы декодирования. Возможность использования неабсолютной адресации связана с наличием в АП "лишнего" пространства.

Интерфейсы с общей и раздельной шиной

С точки зрения использования адресного пространства (АП) памятью и ВУ различают концепции *интерфейса с общей шиной и раздельной шиной*. В рамках первой концепции для адресов памяти и ВУ выделяются части общего АП. К ВУ обращение происходит так же, как и к ячейкам памяти, т. е. с помощью тех же команд и той же шины. Недостатком этой концепции является сужение АП для памяти, поскольку его часть занимается адресами внешних устройств. Достоинство состоит в том, что над данными, получаемыми от ВУ, можно производить все те операции, которые имеются в системе команд процессора для данных, находящихся в ячейках памяти. Таких операций много и это способствует улучшению параметров программ и упрощению программирования. Концепцию "с общей шиной" называют также *вводом/выводом, отображенным на память*.

В концепции "с раздельной шиной" ячейки памяти и ВУ имеют свои АП. При этом требуется наличие управляющих сигналов, определяющих, с каким типом объектов ведется обмен. Например, вводится сигнал IO/M (Input-Output/Memory), указывающий, адресуется память или ВУ. При этом память может использовать все АП. Для обмена с ВУ обычно имеются только операции ввода `IN port` и вывода `OUT port`, и

теряется возможность применять к данным от ВУ широкий набор команд, имеющихся для работы с данными, хранимыми в памяти.

Построение модуля памяти

Модуль памяти обычно состоит из нескольких *субмодулей*, которыми служат отдельные микросхемы или их группы с параллельным включением по входам. При организации модулей адресный код обычно рассматривается как состоящий из двух частей. Одна часть указывает на субмодуль, в котором расположена искомая ячейка памяти, другая является адресом ячейки в субмодуле. Иногда при этом используется терминология, аналогичная применяемой для описания систем виртуальной памяти со страничной организацией, — субмодули называют *страницами*, так что старшие разряды адреса указывают страницу, а младшие — адрес слова на странице.

Микросхемам памяти свойственна организация $2^k \times \ell$, где k — четное число; 2^k — число хранимых слов; ℓ — их разрядность. Если требуется модуль памяти с организацией $2^m \times n$, где $m > k$ и $n > \ell$, то при страничной организации модуля его состав и структура определятся следующими соображениями.

Для наращивания разрядности хранимых слов включаются параллельно несколько микросхем (а именно n/ℓ ИС). Это образует субмодуль, который хранит 2^k слов.

Для увеличения числа хранимых слов до 2^m требуется взять 2^{m-k} субмодулей. Слово в пределах субмодуля адресуется k младшими разрядами адреса, поступающими непосредственно на адресные входы микросхем, а старшие разряды адреса используются для формирования сигнала разрешения работы того или иного субмодуля (рис. 6.16).

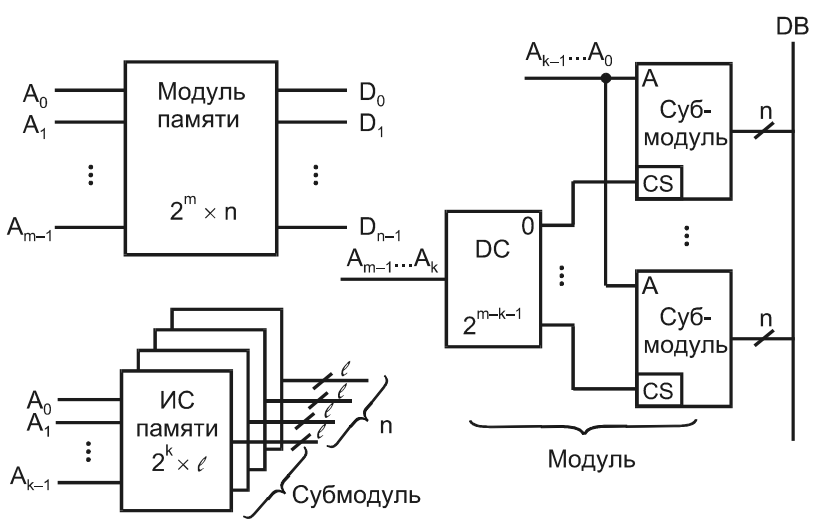


Рис. 6.16. Структуры модуля памяти

Схемы подключения памяти к шинам МПС

Рассмотрим примеры конкретных решений, ориентированных на работу процессора с асинхронными ЗУ. Увязка параметров синхронных микросхем памяти и процессора падает на изготовителя микросхем, тогда как взаимодействие процессора с асинхронной памятью проектируется самим разработчиком.

При проектировании схем подключения микросхем памяти к процессору решаются следующие задачи:

- разработка схем адресации памяти с размещением адресов постоянных и оперативных ЗУ в заданных областях адресного пространства и разработка схем формирования управляющих сигналов;
- анализ нагрузочных условий в полученной схеме и устранение при необходимости перегрузок элементов, обеспечение рабочих режимов для выходов с открытым электродом;
- согласование временных диаграмм процессора и микросхем памяти.

Примечание

Для краткости адреса в АП обычно выражают в шестнадцатеричной системе счисления, а информационные емкости памяти, как правило, оцениваются единицами измерения (байтами, словами и т. д.) с коэффициентами К = 2^{10}, М = 2^{20} или Г = 2^{30}. При работе приходится переходить от одних единиц к другим. Переведем для примера 64К в шестнадцатеричное число. Этой величине соответствует число $2^6 \times 2^{10} = 2^{16}$. Поскольку 16 = 2^4, полученное число можно записать как 16^4 = 10000h. Еще один пример — перевод в шестнадцатеричное число величины 192К. Имеем соотношения: 192К = 3×64К = 3×2^{16} = 3×16^4 = 30000h. Пример перевода шестнадцатеричного числа в единицы К: E000h = 14×16^3 = 14×2^{12} = 56×2^{10} = 56К.

Пример 1. Абсолютная адресация

Типовой элемент схем адресации — дешифратор с информационными и разрешающими входами. На рис. 6.17 приведена схема адресации ПЗУ, составленного из трех субмодулей с организацией 4К×8. Адреса занимают 12К в верхней части АП, т. е. зону от 0000h до 2FFFh.

Сигнал разрешения работы дешифратора $E = \overline{E}_1\overline{E}_2E_3$. Двенадцать младших разрядов адреса выбирают ячейку в субмодуле. Старшие разряды адреса декодируются для формирования сигналов выбора кристалла $\overline{CS}$. Стробирующим сигналом $\overline{RD}$ определяется интервал выполнения операции чтения. Одним из условий разрешения работы дешифратора является низкий уровень сигнала IO/M.

Таким образом, младшая часть адреса, которая "декодируется на кристалле", определяет протяженность зоны адресов, занимаемой данным субмодулем, а другая часть адреса (его старшие разряды) служит для выработки сигнала $\overline{CS}$ и определяет положение зоны адресов в АП.

Пример 2. Неабсолютная адресация

С целью упрощения схем декодирования и при наличии "лишнего" адресного пространства можно применить неабсолютную адресацию, при которой каждому объекту адресации присваивается не один-единственный адрес, а группа адресов (некоторая зона АП).

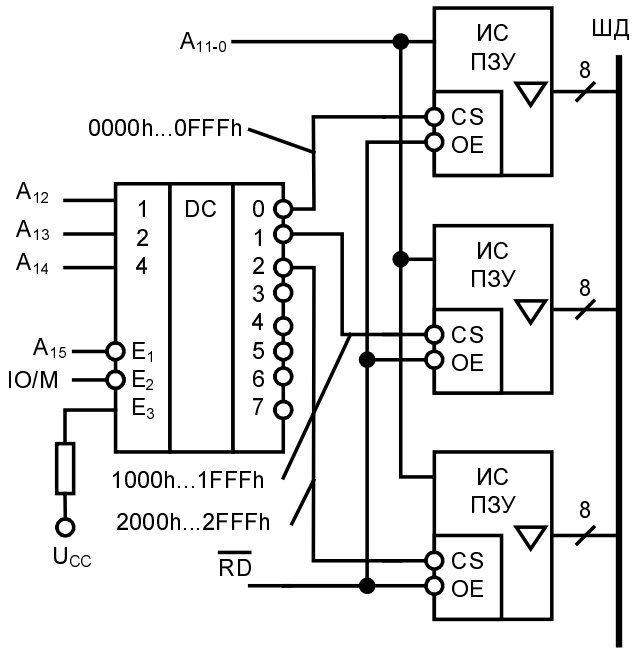


Рис. 6.17. Пример адресации модуля памяти

Пусть, например, в АП емкостью 64К требуется разместить всего два субмодуля памяти по 8К адресов в каждом. При абсолютной адресации (рис. 6.18, *а*) на адресные входы самих ИС памяти поступают 13 младших разрядов адреса для адресации 8К ячеек субмодуля. Оставшиеся разряды $A_{15–13}$ поступают на логические схемы, вырабатывающие сигналы разрешения работы субмодулей СМ1 и СМ2. В зоне АП остаются свободными 48К.

При неабсолютной адресации линии адреса $A_{12–0}$ по-прежнему подаются на адресные входы ИС, а для выбора одной из них используется линия A_{15}. Линии A_{14} и A_{13} не используются вообще, их состояния безразличны. Схема адресации (рис. 6.18, *б*) упрощается, вместо логической схемы нужен только инвертор. Платой за это является занятие двумя субмодулями по 8К всего АП. Действительно, все адреса вида 0XXdd...d принадлежат субмодулю СМ 1, а это соответствует верхним 32К АП. Все адреса вида 1XXdd...d принадлежат субмодулю СМ 2 и занимают 32К в нижней части АП.

Неабсолютная адресация — гибкий подход к построению схем декодирования адреса. Для адресации объекта можно использовать более или менее широкую зону АП, выбирая при необходимости компромисс между крайними решениями, показанными на рис. 6.18.

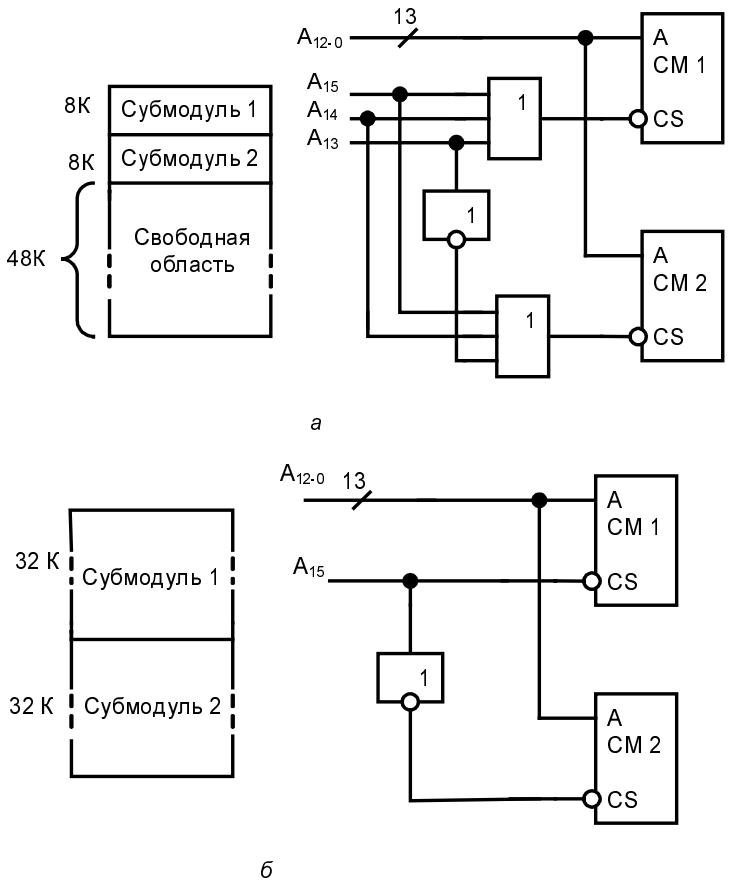


Рис. 6.18. Примеры абсолютной (*а*) и неабсолютной (*б*) адресации субмодулей памяти

Так, в частности, если для адресации субмодулей СМ 1 и СМ 2 использовать не 14 разрядов адреса, как в последнем примере, а 15 при назначении адресов 00Xdd...d для первого субмодуля и 01Xdd...d для второго, то первая четверть АП будет занята адресами СМ 1, вторая — адресами СМ 2, а третья и четвертая свободны. Иначе говоря, "расходование" областей АП окажется вдвое меньшим, чем для схемы (рис. 6.18, *б*), но, в то же время, в схеме декодирования адреса вместо инвертора потребуется небольшая логическая схема. В итоге получится вариант, промежуточный между схемами абсолютной адресации и неабсолютной с минимальным числом адресных разрядов.

Пример 3. Декодирование адресов при совмещенном вводе/выводе

Рассмотрим пример размещения в общем АП адресов ПЗУ, ОЗУ и ВУ, т. е. совмещенный ввод/вывод. Для памяти используем абсолютную адресацию, а для ВУ — линейную селекцию.

> *Линейная селекция* — частный случай неабсолютной адресации кодом "1 из N", когда обращение к ВУ задается тем, что в группе адресных линий возбуждена лишь одна, выделенная для данного ВУ. При этом адресация ВУ реализуется простым подключением входа разрешения его работы к линии, отведенной для данного ВУ, и не требует никаких схем декодирования адреса.

Пусть для ПЗУ отведено 16К адресов в начале АП, адреса ВУ занимают третью четверть АП, а адреса ОЗУ — последние 8К адресного пространства. Примем, что в системе имеется 5 ВУ, каждое из которых имеет 4 внутренних регистра со своими адресами. Распределение АП показано на рис. 6.19.

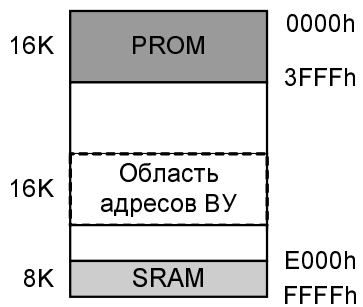


Рис. 6.19. Пример распределения адресного пространства между модулями памяти и внешними устройствами

Пусть ПЗУ строится на микросхемах с организацией 8К×8, а ОЗУ на микросхемах 2К×8. Имея в виду байтовую организацию модуля памяти, видим, что каждая микросхема играет роль субмодуля (не нуждается в наращивании разрядности хранимых слов). Для адресации ВУ используем младшие разряды шины адреса, число которых определяется как N + 2, где N — число ВУ, а две самые младшие линии нужны для адресации их внутренних регистров.

Таблица 6.2

Вид объекта	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
PROM 0	0	0	0	d	D	d	d	d	d	D	d	D	d	d	d	d
PROM 1	0	0	1	d	D	d	d	d	d	D	d	D	d	d	d	d

Таблица 6.2 *(окончание)*

Вид объекта	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
ВУ 0	1	0	X	X	X	X	X	X	X	0	0	0	0	1	d	d
ВУ 1	1	0	X	X	X	X	X	X	X	0	0	0	1	0	d	d
ВУ 2	1	0	X	X	X	X	X	X	X	0	0	1	0	0	d	d
ВУ 3	1	0	X	X	X	X	X	X	X	0	1	0	0	0	d	d
ВУ 4	1	0	X	X	X	X	X	X	X	1	0	0	0	0	d	d
SRAM 0	1	1	1	0	0	d	d	d	d	D	d	D	d	d	d	d
SRAM 1	1	1	1	0	1	d	d	d	d	D	d	D	d	d	d	d
SRAM 2	1	1	1	1	0	d	d	d	d	D	d	D	d	d	d	d
SRAM 3	1	1	1	1	1	d	d	d	d	D	d	D	d	d	d	d

Схема адресации, соответствующая табл. 6.2, приведена на рис. 6.20. Дешифратор DC 1 делит АП на четыре части, его выходы разрешают работу тем объектам адресации, которые расположены в соответствующей четверти АП. Линия A_{13} разрешает работу микросхемы PROM 0 в первой половине первой четверти АП при нулевом состоянии и работу микросхемы PROM 1 — при единичном. Линии A_2...A_6 использованы для линейной селекции внешних устройств, а линии A_{12} и A_{11} декодируются дешифратором DC 2, для разрешения работы микросхемам SRAM 3...SRAM 0 в их зонах адресов.

Символом "X" обозначены безразличные состояния адресных разрядов, а буквой d — разряды, "декодируемые на кристалле", т. е. входящие в состав адресных входов самих микросхем памяти или адресных линий внутренних регистров ВУ.

Выработка сигналов управления

Адресация — это только часть процесса обращения к памяти или ВУ. Кроме адресации требуется и управление другими сигналами, к которым прежде всего относятся сигналы, определяющие направление обмена (чтение или запись) и признаки обращения к ВУ или памяти. Эти сведения можно передавать с помощью разных сочетаний управляющих сигналов.

Сигналы управления чтением/записью. Сигналы, вырабатываемые процессором, и сигналы, предусматриваемые системным интерфейсом, могут отличаться друг от друга. В таком случае один из наборов сигналов преобразуется в другой.

Рассмотренный ранее процессор вырабатывает для памяти и ВУ минимальную группу из трех сигналов управления ($\overline{RD}$, $\overline{WR}$ и IO/M), тогда как в стандартном систем-

ном интерфейсе предусмотрены четыре сигнала: чтения из памяти $\overline{\text{MEMR}}$, записи в память $\overline{\text{MEMW}}$, чтения из ВУ $\overline{\text{IOW}}$ и записи в ВУ $\overline{\text{IOR}}$.

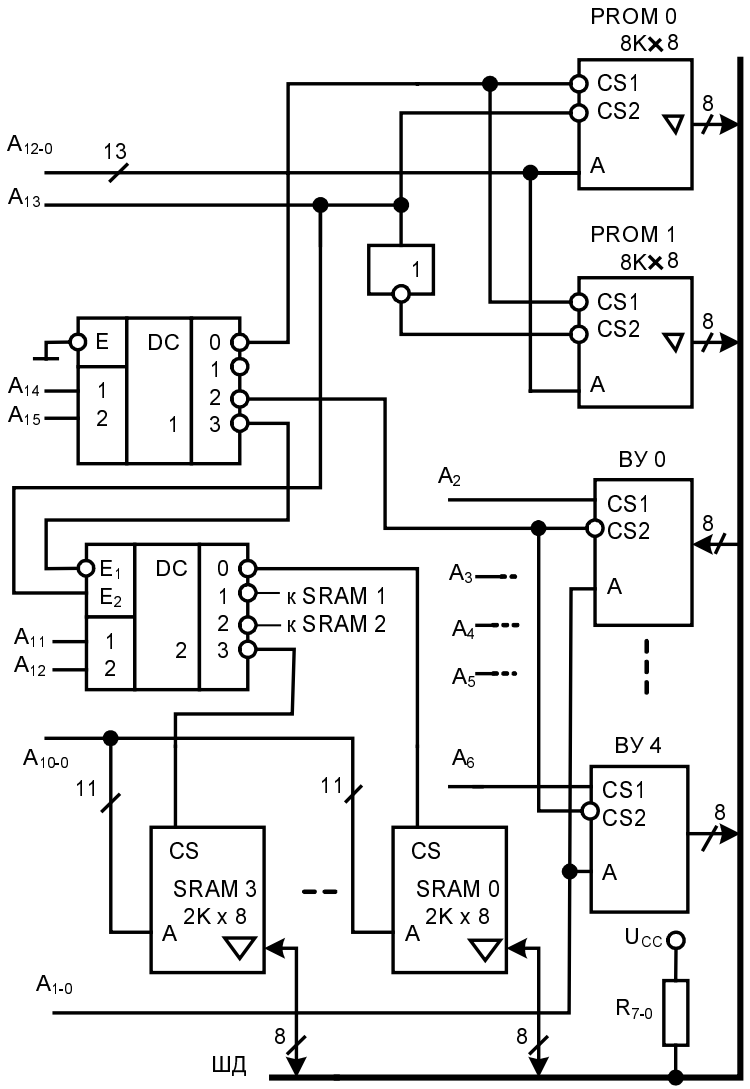


Рис. 6.20. Схема адресации памяти и внешних устройств при отображении системы ввода/вывода на память

К четверке сигналов можно перейти по следующим соотношениям:

$$\overline{\text{MEMR}} = \overline{\text{RD} \cdot \overline{\text{IO}/\text{M}}} = \overline{\text{RD}} \vee \text{IO}/\text{M},$$

$$\overline{\text{MEMW}} = \overline{\text{WR} \cdot \overline{\text{IO}/\text{M}}} = \overline{\text{WR}} \vee \text{IO}/\text{M},$$

$$\overline{\text{IOR}} = \overline{\text{RD} \cdot \text{IO}/\text{M}} = \overline{\text{RD}} \vee \overline{\text{IO}/\text{M}},$$

$$\overline{\text{IOW}} = \overline{\text{WR} \cdot \text{IO}/\text{M}} = \overline{\text{WR}} \vee \overline{\text{IO}/\text{M}}.$$

Придание сигналам импульсного характера. Для некоторых ЗУ (например, тактируемых статических асинхронных ОЗУ) нужен импульсный характер какого-либо сигнала управления (обычно сигнала $\overline{CS}$). В этом случае для разрешения работы памяти при каждом обращении к ней нужно предварительно вернуть сигнал в пассивное состояние. Для придания сигналу импульсного характера можно применить, например, соотношение

$$\overline{CS} = \overline{MEMR} \cdot \overline{MEMW},$$

обеспечивающее пассивное состояние сигнала $\overline{CS}$ на интервалах, на которых не действуют ни сигнал чтения, ни сигнал записи.

На рис. 6.21 показан пример выработки сигналов управления для тактируемого SRAM с импульсным сигналом $\overline{CS}$. Для возвращения $\overline{CS}$ к пассивному (высокому) уровню между циклами обращения к ЗУ используется конъюнкция сигналов $\overline{MEMR}$ и $\overline{MEMW}$ как сигнал разрешения работы дешифратора. В этом случае между циклами обращения к ЗУ и сигнал чтения из памяти $\overline{MEMR}$ и сигнал записи в память $\overline{MEMR}$ пассивны, т. е. имеют единичные уровни. При этом на выходе конъюнктора вырабатывается единичный сигнал, запрещающий работу дешифратора, на выходе которого формируются пассивные (единичные) уровни, т. е. снимаются сигналы $\overline{CS}$. При любом обращении к памяти активизируется (становится нулевым) сигнал $\overline{MEMR}$ или $\overline{MEMR}$, что создает нулевой сигнал на выходе конъюнктора и разрешает работу дешифратора.

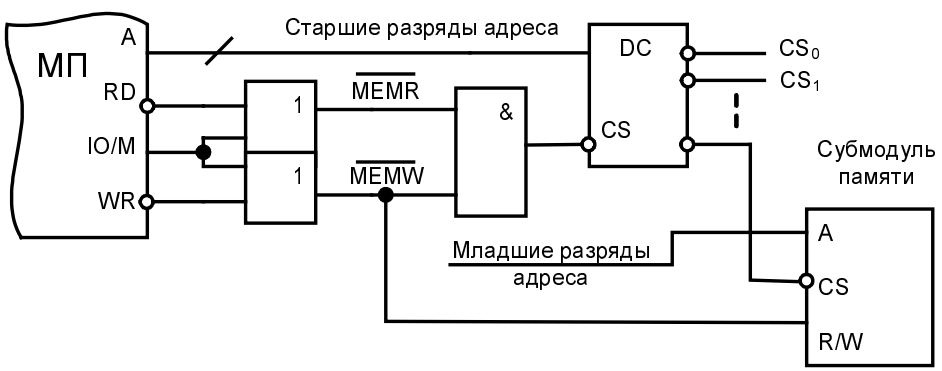


Рис. 6.21. Схемы выработки сигналов управления для асинхронного тактируемого статического ЗУ

Выработка сигналов готовности. Иногда условием обмена является *готовность* к нему памяти или ВУ. Для выявления готовности применяют такой метод: появление адреса медленного устройства ведет к запуску генератора одиночного импульса необходимой длительности, на время существования которого сигнал готовности READY снимается. Длительность интервала неготовности рассчитывается согласно требованиям медленного устройства. Процессор ждет появления сигнала

готовности и только после его появления выполняет операцию обмена. Чтобы избежать потерь времени, желательно генерировать интервал неготовности с привязкой его к синхроимпульсам МПС.

На рис. 6.22, *а* показана выработка сигнала готовности, подаваемого на вход READY микропроцессора. Когда микропроцессор обращается к медленному субмодулю (или другому медленному устройству), соответствующий сигнал $\overline{CS}$ становится нулевым. Это говорит о необходимости снятия сигнала готовности на время, которое задает проектировщик, знающий характеристики медленного устройства. Сигнал разрешения работы медленного устройства запускает генератор одиночных импульсов G1, выходной уровень логического нуля которого ("отрицательный" импульс) имеет длительность t_0, назначенную проектировщиком. На время существования этого импульса сигнал готовности снимается, и МП вводит в цикл обращения к устройству такты ожидания. По окончании импульса появляется сигнал READY, МП выходит из состояния ожидания и реализует операцию обмена. Длительность импульса подбирается соответственно требованиям медленного устройства.

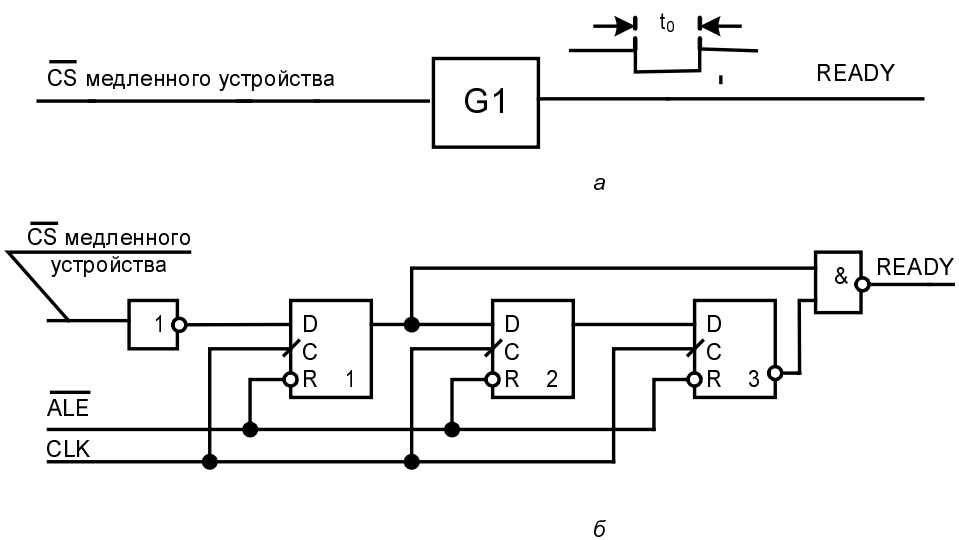


Рис. 6.22. Пример схемы генерации сигнала неготовности при работе процессора с памятью малого быстродействия или другим медленным устройством

Показанный на рис. 6.22, *б* генератор вырабатывает одиночный импульс, синхронизированный с тактовыми импульсами системы. В каждом машинном цикле сигнал $\overline{ALE}$ сбрасывает триггеры, на вход элемента И-НЕ действует нулевой сигнал с выхода триггера Т1 и, следовательно, сигнал RDY = 1. Появление сигнала разрешения работы $\overline{CS}$ медленного устройства приводит к установке триггера Т1 первым же тактовым импульсом, при этом на входах элемента И-НЕ оба сигнала становятся единичными и выход READY принимает нулевое значение. Это состояние

неготовности продлится до тех пор, пока единичное состояние не продвинется по цепочке триггеров до конца. Установка триггера T_n создает на нижнем входе элемента И-НЕ нулевой сигнал, и сигнал READY станет единичным, что позволит МП перейти к операции обмена. Вводимое число тактов ожидания здесь соответствует n – 1, т. е. числу триггеров в цепочке, начиная с триггера Т2.

Анализ нагрузочных условий

После разработки функционально-логической схемы управления памятью следует провести анализ токовой нагрузки в обоих статических состояниях и емкостной нагрузки всех ее элементов. Для обоих состояний выхода элемента суммарный ток нагрузки не должен превышать указанного в технических условиях (ТУ) допустимого выходного тока. Суммарная емкость входов, подключенных к выходу данного элемента в сумме с емкостью монтажа также не должна превышать допустимой для данного элемента. Перегрузки элементов должны быть устранены введением буферных каскадов или другими способами (*см. § 1.6*).

Для выходов с открытым коллектором определяются сопротивления резисторов внешней цепи U_{cc} – R (*см. § 1.2*).

Согласование временных диаграмм МП и ЗУ

Наличие законченной схемы подключения памяти к системным шинам позволяет рассмотреть задачу согласования временных диаграмм МП (системной шины) и асинхронных ЗУ. Такое согласование — необходимое условие работоспособности системы. Исходными данными при этом являются:

- временные диаграммы циклов чтения и записи МП;
- временные диаграммы циклов чтения и записи ЗУ;
- схема адресации и формирования управляющих сигналов ЗУ;
- сведения о задержках сигналов в элементах схемы и связях между ними.

При определении задержек следует иметь в виду их зависимость от емкостных нагрузок на выходах элементов (*см. § 1.1*).

Перечень режимных параметров ЗУ (необходимых длительностей сигналов, их предустановок, времен удержания и сохранения) достаточно велик. Методика их обеспечения будет показана на примере некоторых параметров. Анализ для других параметров выполняется аналогичным способом.

Рассмотрим процесс чтения для тактируемой SRAM. Выясним вопросы, связанные с временем доступа по адресу (t_A), временем доступа по сигналу выбора (t_{CS}) и с временем $t_{SU(A-CS)}$ предустановки адреса относительно сигнала $\overline{CS}$. Для этого воспользуемся схемой, на которой показаны тракты прохождения интересующих нас сигналов (рис. 6.23).

Началом отсчета считаем момент выставления адреса на выходах МП. После этого происходят следующие процессы:

- часть старшего полуадреса поступает на ЗУ через время t_{BUF}, т. е. время задержки буфера (другая часть старшего полуадреса поступает на дешифратор выработки сигналов $\overline{CS_A}$);
- младший полуадрес появляется на входах ЗУ позднее, чем старший, т. к. только через время t_{AL} сигнал ALE задним фронтом загружает регистр, после чего через время задержки регистра t_{RG} сформируется адрес на входах ЗУ;
- через время t_A после поступления адреса ЗУ вырабатывает выходные данные (t_A — характеристика по ТУ);

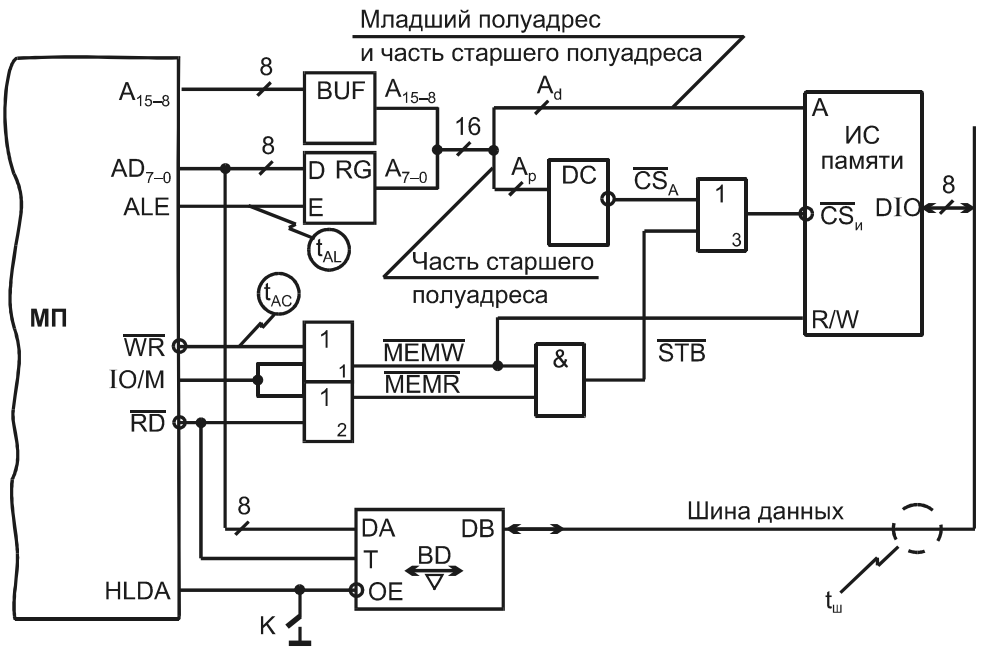


Рис. 6.23. Схема трактов передачи сигналов при управлении памятью

- по истечении времени задержек шины и буфера данных ($t_Ш$ и t_{BD}) данные появятся на линиях $AD_{7–0}$ микропроцессора, и это должно произойти не позднее, чем в момент t_{AD}, определяемый временной диаграммой МП.

Параметр $t_Ш$ здесь — задержка шины, существенная, если ЗУ имеет выходы с открытым коллектором, которые медленно формируют положительные фронты. Как правило, подобных задержек избегают (например, предварительно приводят линии шины в единичные состояния, так что при появлении данных могут происходить только переключения в ноль). В дальнейшем задержку $t_Ш$ учитывать не будем.

Таким образом, на основании сказанного должно соблюдаться соотношение:

$$t_{AL} + t_{RG} + t_A + t_{BD} \le t_{AD}.$$

Из полученного неравенства следует условие, предъявляемое к параметру памяти t_A:

$$t_A \le t_{AD}(N) - (t_{AL} + t_{RG} + t_{BD}).$$

Напомним, что интервал t_{AD} задается формулой (приводилась ранее), в которую входит число вводимых циклов ожидания, за счет которых удовлетворяется требуемое неравенство, если в системе используется память, которая не успевает подготовить данные в течение обычного цикла. В связи со сказанным в последнем неравенстве величина t_{AD} записана как функция $t_{AD}(N)$.

Рассмотрим теперь требования к параметру памяти t_{CS}. Из отмеченного выше следует, что сигнал $\overline{CS}_A$ (сигнал выбора субмодуля, получаемый декодированием нескольких старших разрядов адреса) появляется на входе элемента $ИЛИ_3$ через время $t_{BUF} + t_{DC}$. Нулевой сигнал на нижнем входе этого элемента ИЛИ появится позднее и определит тем самым момент поступления сигнала $\overline{CS}_C$ на вход ИС памяти. Этот сигнал, обозначенный как $\overline{STB}$, появится в момент времени $t_{AC} + t_{ИЛИ} + t_И$, где t_{AC} — параметр временной диаграммы МП (интервал между моментами выставления адреса и строба чтения). По истечении времени задержки элемента ИЛИ на входе $\overline{CS}_И$ сформируется сигнал выбора субмодуля. После этого памяти потребуется время t_{CS} для подготовки выходных данных, которые после задержки в буфере данных появятся на линиях AD_{7-0} микропроцессора, что должно произойти не позднее, чем в момент времени $t_{AD}(N)$.

Из сказанного следует условие:

$$t_{AC} + 2t_{ИЛИ} + t_И + t_{CS} + t_{BD} \le t_{AD}(N),$$

на основании которого предъявляется требование к величине t_{CS}:

$$t_{CS} \le t_{AD}(N) - (t_{AC} + 2t_{ИЛИ} + t_И + t_{BD}).$$

Разность времен появления сигналов $\overline{CS}_C$ и адреса на входах ИС памяти определит их предустановку в схеме:

$$t_{SU(A-CS).CX} = t_{AC} + 2t_{ИЛИ} + t_И - (t_{AL} + t_{RG}).$$

Требуется соблюдение условия:

$$t_{SU(A-CS).CX} \ge t_{SU(A-CS).ТУ},$$

где $t_{SU(A-CS).ТУ}$ — параметр памяти.

К завершению цикла чтения тоже предъявляются определенные требования. Необходимо, чтобы "старые" данные были сняты с шины AD_{7-0} раньше, чем появится новое значение адреса. Временные диаграммы МП определяют интервал от конца строба чтения до появления нового адреса t_{RA} (для нашего примера это T/2 – 10 нс). В паспортных данных ИС памяти имеется параметр $t_{DIS(CS)}$ — время запрещения данных после снятия сигнала CS. Временные диаграммы для завершения чтения (рис. 6.24) учитывают, что сигнал $\overline{CS}_И$ будет снят после окончания

(рис. 6.24) учитывают, что сигнал $\overline{CS}_И$ будет снят после окончания сигнала $\overline{RD}$ через время, равное суммарной задержке элементов $ИЛИ_2$, И и $ИЛИ_3$. На основании рисунка можно записать соотношение:

$$t_{DIS(CS)} \le t_{RA} - (2t_{ИЛИ} + t_И).$$

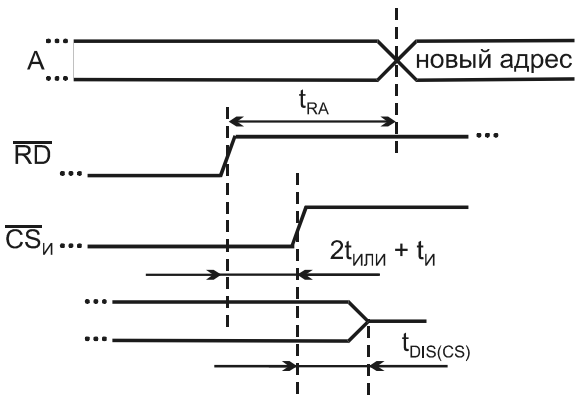


Рис. 6.24. Временные диаграммы сигналов для завершения цикла чтения из памяти

В цикле записи следует обеспечить доступ к ячейке только после ее четкой адресации и предотвратить обращение к иным, кроме выбранной, ячейкам. Первое условие требует определенной предустановки адреса относительно строба записи на входах ИС памяти, второе — полного отключения от трактов записи предыдущей ячейки до начала нового цикла.

Согласование временных диаграмм памяти и МП для быстродействующих систем может оказаться сложной задачей. В таких системах временные интервалы малы, и их сдвиги из-за паразитных задержек сильно усложняют ситуацию. Для синхронных ЗУ эти задачи решаются разработчиками кристаллов.

Разновидности операций ввода/вывода

Передача данных между устройствами требует управления этим процессом. В МПС обмен данными может проходить в режимах *программно-управляемого обмена* или *прямого доступа к памяти (ПДП)* и, соответственно, сигналы управления генерируются *процессором* или *контроллером прямого доступа к памяти*. В свою очередь программно-управляемый обмен может *инициироваться программой* (процессором) или же *запросом прерывания* от внешнего устройства

Обмен по инициативе программы

В этом случае возможно взаимодействие с устройством, всегда готовым к обмену *(безусловный ввод/вывод)* или с ожиданием готовности устройства *(условный ввод/вывод)*. В последнем случае вырабатываются сигналы, сообщающие о состоя-

нии устройства. Процессор анализирует эти сигналы и при готовности устройства реализует программу его обслуживания. Такой обмен может быть сопряжен с большими потерями времени. Быстродействие внешних устройств, с которыми идет обмен, зачастую очень мало в сравнении с быстродействием процессора. Ожидая готовности устройства, процессор не выполняет полезной работы, а занят в каждом цикле проверкой состояния внешнего устройства и простаивает в течение больших интервалов времени. Кроме того, опасны ситуации, когда по тем или иным причинам сигнал готовности от ВУ вообще не формируется, поскольку это приводит к прекращению работы системы. Для предотвращения подобных ситуаций применяют специальные меры.

Обмен по прерываниям

При обменах по прерываниям, ожидание исключается, т. к. *инициатива обмена исходит от внешнего устройства* (ВУ). При своей готовности ВУ сигнализируют процессору, запрашивая у него прерывания основной программы и обслуживания обмена. Процессор завершает выполнение текущей команды основной программы и переходит к подпрограмме обслуживания прерывания. Отсутствие длительных интервалов ожидания существенно увеличивает производительность МПС. В то же время обмен по прерываниям требует применения дополнительных аппаратных и программных средств его обеспечения.

Прямой доступ к памяти

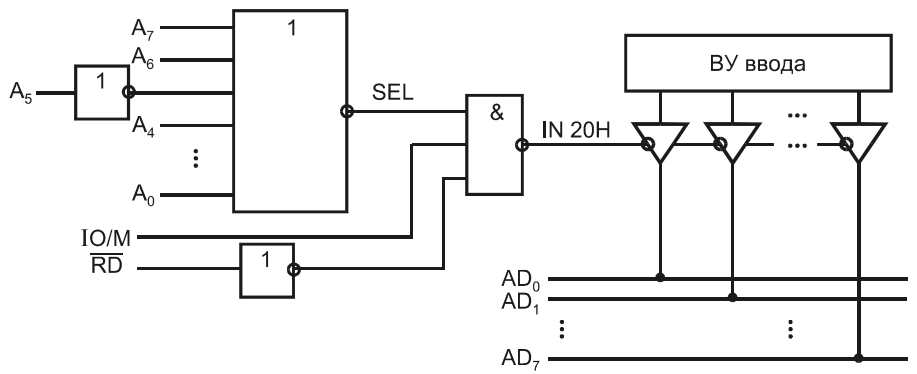
Для обмена между памятью и ВУ без участия процессора используется режим ПДП. В обычном режиме пересылка данных между памятью и ВУ требует вначале приема данных в процессор, а затем выдачи их приемнику, что снижает темп передачи. В режиме ПДП организуется прямой тракт передач между памятью и ВУ. При этом процессор отключается от системных шин и передает управление обменом специальному контроллеру ПДП, что увеличивает темп передачи данных. Кроме того, при кэшированной памяти не исключается параллельное выполнение процессором основной программы и передачи данных между основной памятью и внешним устройством в режиме ПДП. Наличие ПДП повышает эффективность МПС.

Безусловный программный ввод/вывод

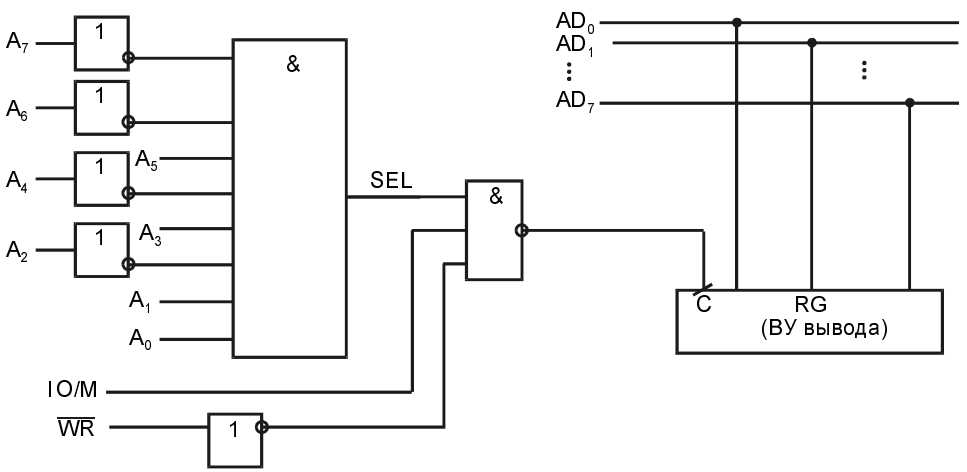
Для подключения внешних устройств к шинам МПС при малом числе ВУ можно применить линейную селекцию (пример рассмотрен ранее), не требующую для своей реализации каких-либо аппаратурных затрат. Возможностями подключения большого числа ВУ, равного 2^m устройств ввода и такого же числа устройств вывода, где m — разрядность адресов ВУ, обладает вариант *адресуемых портов*. При восьмиразрядных адресах через адресуемые порты можно подключить до 256 ВУ ввода и 256 ВУ вывода.

Схемная реализация безусловного ввода. При вводе данных микропроцессор вырабатывает адрес ВУ A_{7-0}, сигнал IO/M = 1 и строб чтения $\overline{RD}$. В интерфейсе при вводе требуется адрес ВУ и строб чтения $\overline{IOR}$.

При использовании управляющих сигналов МП схема адресного порта безусловного ввода имеет вид рис. 6.25, *а*, где адрес ВУ принят равным 20h = 00100000. В этом случае сигнал выбора ВУ должен появиться при подаче с адресных шин на входы конъюнктора набора $\overline{A}_7\overline{A}_6A_5\overline{A}_4\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ или, что то же самое, при подаче на входы элемента ИЛИ-НЕ набора $A_7A_6\overline{A}_5A_4A_3A_2A_1A_0$. Последний вариант лучше, т. к. требует инвертировать всего один бит адресного кода, тогда как при реализации первого потребуются семь инверторов.



а



б

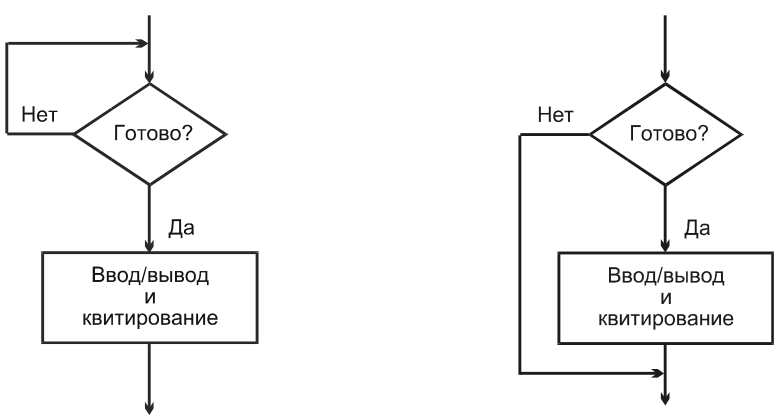
Рис. 6.25. Схемы реализации команд ввода (*а*) и вывода (*б*) при безусловном программном обмене

Совпадение сигналов ввода дает нулевой сигнал на выходе элемента И-НЕ, открывающий линейку буферов с тремя состояниями выхода и передающий таким образом байт данных от ВУ на шину данных, откуда они и считываются процессором.

Схемная реализация безусловного вывода. На рис. 6.25, *б* показана реализация команды `OUT 2Bh`. Вывод осуществляется при поступлении от процессора адреса 00101011, сигналов IO/M = 1 и $\overline{WR}$ = 0. Поступая на конъюнктор с инверторами в соответствующих входных линиях, адрес формирует единичный сигнал SEL, что в совокупности с единичными сигналами на двух других входах элемента И-НЕ дает отрицательный перепад напряжения на его выходе. Задний фронт строба $\overline{WR}$ дает положительный перепад выходного напряжения элемента И-НЕ, загружающий данные с шины данных AD_{7-0} в регистр RG порта вывода.

Условный программный ввод/вывод

При условном обмене могут возникать потери времени на ожидание готовности ВУ. Алгоритм обмена с ожиданием готовности (рис. 6.26, *а*) таков, что МП может задерживаться в цикле ожидания, причем при работе с ВУ малого быстродействия время ожидания может оказаться большим. Возможен также другой алгоритм условного ввода/вывода (с пропуском цикла), когда при неготовности ВУ процессор отказывается от операции обмена и продолжает основную программу (рис. 6.26, *б*). В последующем может быть выполнена новая попытка обмена.



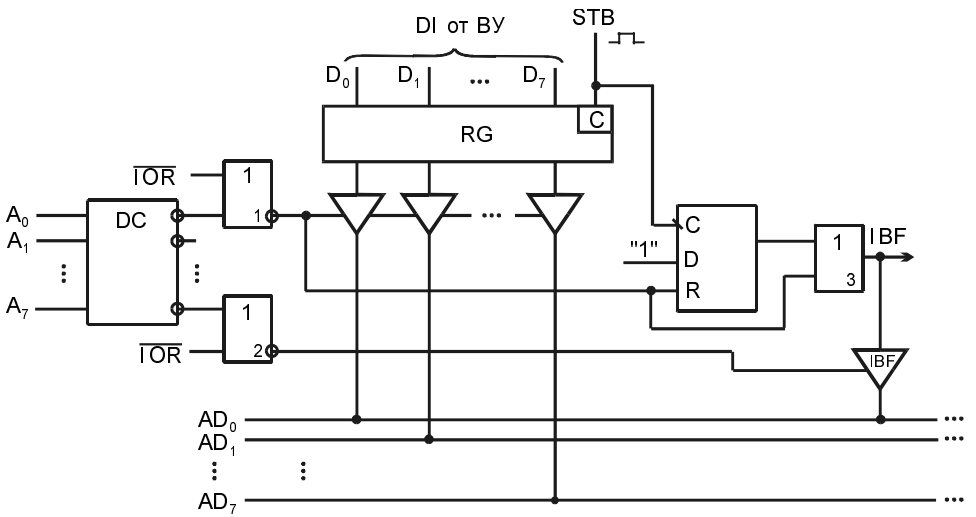
а *б*

Рис. 6.26. Алгоритмы условного программного обмена с занятием цикла (*а*) и совмещенного (*б*)

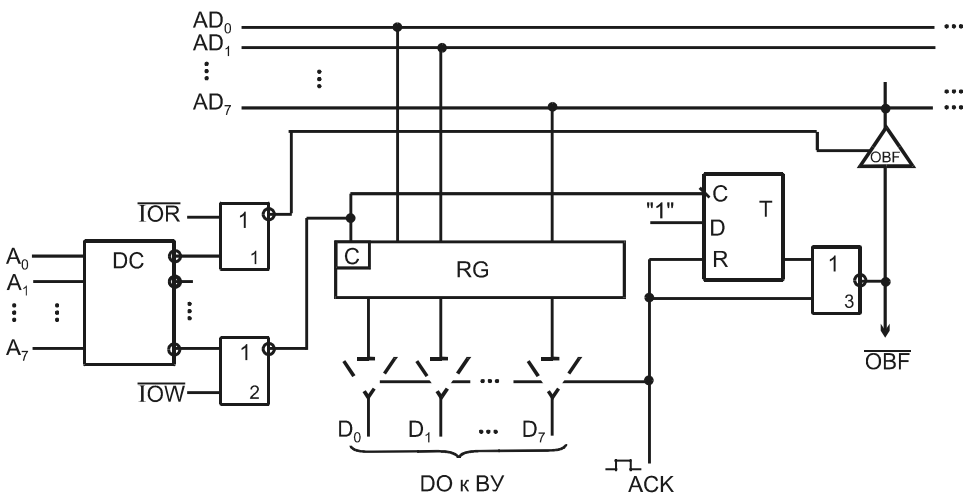
При условном обмене необходимо удостовериться в готовности ВУ, прежде чем начать обмен, т. е. операции ввода/вывода сопровождаются специальными сигналами готовности, генерируемыми ВУ и вводимыми в МП. После операции ввода/вывода сигнал готовности снимается и выставляется снова при новой готовно-

сти к обмену. Такой протокол называют обменом *с квитированием*. Обмен происходит со скоростью, определяемой внешним устройством.

Схема условного ввода. Вариант порта ввода с квитированием и "четверкой" управляющих сигналов (рис. 6.27, *а*) предусматривает наличие регистра-защелки RG. Блок DC — схема декодирования адреса. ВУ готовит данные на линиях $D_{7\text{-}0}$ и оповещает об их наличии сигналом STB, загружающим регистр, и своим задним фронтом, устанавливающим триггер, формируя этим сигнал IBF = 1 (IBF, Input Buffer Full). Этим фиксируется готовность ко вводу со стороны порта.



а



б

Рис. 6.27. Схемы реализации команд ввода (*а*) и вывода (*б*) при условном программном обмене

Микропроцессор начинает обращение к порту чтением по адресу, присвоенному буферному каскаду IBF (для определенности принято, что этот адрес соответствует возбуждению нижней выходной линии DC). По стробу $\overline{\text{IOR}}$ на выходе элемента ИЛИ_2 возникает единичный сигнал, открывающий буферный каскад, через который сигнал IBF поступает на линию AD_0. Этот сигнал есть бит слова состояния, которое считывается процессором. Если значение этого бита 1, то далее осуществляется ввод по адресу порта (здесь этот адрес принят нулевым, и ему соответствует возбуждение верхней выходной линии схемы DC). Строб $\overline{\text{IOR}}$ устанавливает логическую 1 на выходе элемента ИЛИ_1, т. е. открывает линейку буферных каскадов, через которые байт D_{7-0} поступает на линии AD_{7-0} и далее вводится в процессор. По окончании строба $\overline{\text{IOR}}$ сигнал IBF становится нулевым. Это снимает готовность к обмену до новой загрузки входного буферного регистра RG от ВУ с установкой после этого триггера и приведения IBF в состояние логической 1, т. е. в состояние новой готовности к обмену. Подача на входы вентиля 3 самого сигнала ввода и выхода сбрасываемого триггера не дает признаку IBF измениться до окончания импульса ввода (при использовании только выхода триггера сигнал запрещения новой загрузки снимался бы слишком рано, в начале процесса использования предыдущих данных).

В программе описанный процесс ввода отображается следующей процедурой:

```
IA: IN  00h       ; Ввод слова состояния
    ANI 01h       ; Выделение бита
    JZ IA         ; Если IBF = 0, то ждать
    IN OFFh       ; Иначе ввод данных
```

Схема условного вывода. При условном выводе данных (рис. 6.27, *б*) сначала вводится слово состояния с шины AD_{7-0}, на одной из линий которой действует сигнал готовности $\overline{\text{OBF}}$ (Output Buffer Full). Данные будут выводиться процессором в RG, готовность к выводу выражается в том, что данные регистра уже приняты ВУ. Об этом сигнализирует подтверждение ACK, сбрасывающее триггер и формирующее после своего окончания сигнал $\overline{\text{OBF}} = 1$ (буфер пуст). Подача на входы вентиля 3 самого сигнала сброса и выхода сбрасываемого триггера не дает признаку $\overline{\text{OBF}}$ измениться до окончания импульса ACK. Появление готовности именно после завершения указанных действий требуется для надежной работы схемы.

Из введенного слова состояния выделяется бит $\overline{\text{OBF}}$ (как и ранее командой `ANI`, т. е. выполнением поразрядной конъюнкции со словом, содержащим единицу только в разряде, принадлежащем сигналу $\overline{\text{OBF}}$). При $\overline{\text{OBF}} = 1$ идет обмен, иначе — переход к циклам ожидания. Появление готовности вызывает запись в RG данных и установку триггера, дающего $\overline{\text{OBF}} = 0$, как сигнал для ВУ к использованию выводимых данных, после чего ВУ дает сигнал ACK, сбрасывающий триггер и выставляющий сигнал готовности порта к новому выводу.

§ 6.6. Микроконтроллеры. Основные сведения

Микроконтроллеры (МК) — разновидность микропроцессорных систем, ориентированная на реализацию алгоритмов управления техническими устройствами и технологическими процессами. Микроконтроллеры проще, чем универсальные компьютеры, и уже около 25 лет тому назад оказалось возможным размещать их на одном кристалле.

Что отличает МК от универсального компьютера? Прежде всего, это малый объем памяти и менее разнообразный состав внешних устройств. В состав универсального компьютера входят модули памяти большого объема и высокого быстродействия, имеется сложная иерархия ЗУ, поскольку многие задачи (автоматизированное проектирование, компьютерная графика, мультимедийные приложения и др.) без этого решить невозможно. Для МК ситуация иная, они реализуют несложные алгоритмы, и для размещения программ им требуются емкости памяти, на несколько порядков меньшие, чем у универсальных микроЭВМ. Для хранения в процессе вычислений промежуточных данных достаточна оперативная память небольшой емкости. Набор внешних устройств также существенно конкретизируется и сужается, а сами они значительно проще. В результате модули МК размещаются на одном кристалле, тогда как модули универсального компьютера (процессор, память, интерфейсные схемы) выполняются как конструктивно самостоятельные.

Сравнивая микропроцессоры с микроконтроллерами по масштабам производства, можно видеть преобладание МК. Число пользователей МК в несколько раз превышает число пользователей МП. Применение МК поддерживается такими областями массового производства, как бытовая аппаратура, станкостроение, автомобильная промышленность, военное оборудование и т. д. Годовой мировой выпуск микроконтроллеров сейчас составляет несколько миллиардов, а их номенклатура насчитывает тысячи типов.

Первые МК были выпущены фирмой Intel в 1976 г. (восьмиразрядный МК 8048). В настоящее время многими поставщиками выпускаются 8-, 16- и 32-разрядные МК с емкостью памяти программ до десятков Кбайт, небольшими ОЗУ данных и набором таких интерфейсных и периферийных схем, как параллельные и последовательные порты ввода/вывода, таймеры, аналого-цифровые и цифроаналоговые преобразователи, широтно-импульсные модуляторы и др.

Среди выпускаемых МК широко известно семейство восьмиразрядных контроллеров MCS-51/151/251 и 16-разрядных MCS-96/196/296 (фирма Intel). Очень многие производители выпускают аналоги этих семейств или совместимые с ними МК. В отечественной номенклатуре это восьмиразрядные МК К1816ВЕ51, К1830ВЕ51. Со временем фирма Intel сосредоточила усилия на разработке сложных МП для компьютеров и уступила сектор рынка простых МК другим фирмам. Признанные авторитеты в области создания и производства МК, наиболее востребованные на российском рынке — фирмы Motorola (позднее Freescale, созданная на основе фирмы

Motorola), Microchip, Atmel, Philips (позднее NXT, преобразованная фирма Philips), Zilog и др. Микроконтроллеры широко применяют в СБИС программируемой логики типа "система на кристалле".

Наибольший успех на рынке долго оставался за 8-разрядными МК благодаря их компактности и дешевизне, что и сейчас продолжает привлекать потребителей. В 2004 г. производство 8-разрядных МК в 1,5…2 раза превышало производство 16- и 32-разрядных. В 2008 г. ожидается приблизительно равный объем производства 8-, 16- и 32-разрядных МК.

На отечественном рынке восьмиразрядных микроконтроллеров доминирует следующая тройка: семейство AVR (фирма Atmel), семейство PIC (фирма Microchip), семейство 8051 (фирма Intel и другие). Остальные микроконтроллеры по объему продаж значительно отстают от лидеров.

> Напомним, что по одному из классификационных признаков МП или МК могут принадлежать к *CISC- или RISC-процессорам*. Процессоры CISC имеют сложную систему команд (CISC — Complex Instruction Set Computer), т. е. большой набор разноформатных команд и используют многие способы адресации. Архитектура CISC присуща классическим (традиционным) процессорам, она в силу многообразия команд позволяет применять эффективные алгоритмы решения задач, но, в то же время, усложняет схему процессора и его стоимость и в общем случае не обеспечивает его максимального быстродействия.
>
> Процессоры типа RISC имеют сокращенную систему команд (RISC — Reduced Instruction Set Computer), из которой исключены редко применяемые команды. Форматы команд, по крайней мере подавляющего их большинства, идентичны (например, все команды содержат по 4 байта), резко снижено число используемых способов адресации. Данные, как правило, обрабатываются только с регистровой или непосредственной адресацией. Значительно увеличенное число регистров процессора, т. е. его емкая внутренняя память, позволяет редко обращаться к внешнему модулю памяти микропроцессорной системы, а это повышает быстродействие контроллера. Идентичность временных циклов выполнения команд отвечает потребностям конвейерных схем обработки информации. В результате может быть достигнуто упрощение схемы процессора при увеличении его быстродействия.

Микроконтроллеры семейства 8051 с архитектурой CISC многократно описаны в литературе, начиная с работ пятнадцатилетней давности, например, [53] и кончая современными, например, [39], [37], [11].

Микроконтроллеры AVR. В последнее время в структуры микроконтроллеров интенсивно внедряются RISC-процессоры. Микроконтроллеры семейства AVR с RISC-процессором обладают хорошо продуманной архитектурой и высоким быстродействием. Они подразделяются на три семейства, причем основную линию развития МК AVR представляет семейство Classic, младшие модели входят в семейство Tiny AVR, а старшие — в семейство Mega AVR. Далее рассматривается типичный представитель семейства Classic, называемый просто МК AVR (рассматриваемый МК близок к модели Classic 8515, обладающей повышенной функциональной полнотой и поддерживающей большую часть возможностей, характерных для всего семейства в целом).

Контроллеры AVR семейства Classic характеризуются следующим:

- почти все команды выполняются за один машинный такт, что при тактовой частоте 1 МГц дает производительность в 1 MIPS;
- флэш-память программ емкостью 1…8 Кбайт имеет допустимое число репрограммирований 10^3;
- статическая память программ (SRAM) имеет емкость до 512 байт;
- память данных типа EEPROM с допустимым числом репрограммирований 10^5 имеет емкость 64…512 байт;
- многоуровневая система прерываний обслуживает от 3 до 16 источников запросов прерываний;
- имеется достаточно обширный набор периферийных устройств;
- имеется последовательный отладочный интерфейс.

§ 6.7. Структура микроконтроллера

МК AVR — 8-разрядный RISC МК с Гарвардской архитектурой и пониженным энергопотреблением. Набор команд, ограниченность которого свойственна RISC-архитектурам, в данном случае необычно широк (120 команд), однако при этом сохранено основное преимущество RISC-архитектур — повышенное быстродействие и сокращенное число операций обмена с памятью программ. Почти все команды занимают одну ячейку программной памяти и выполняются за один такт синхросигнала. Частота синхронизации лежит в пределах 0…8 МГц. Доступ к памяти программ и памяти данных осуществляется через их собственные шины, поэтому можно реализовать параллелизм операций в процессах выполнения текущей команды и выборки и дешифрации следующей.

МК AVR (рис. 6.28) имеет 8-разрядную шину данных, посредством которой его модули обмениваются информацией.

> С целью упрощения рисунка разрядности шин на нем не указаны, но они легко могут быть определены на основе достаточно простых соображений: все шины, выходящие на шину данных, имеют по 8 разрядов, такова же разрядность блока регистров и АЛУ, счетчик команд соответственно емкости памяти программ (2048 слов, т. е. 4096 байт) является 12-разрядным, флэш-память имеет 16-разрядную организацию, определяющую разрядность регистра команд IR. Линии шины управления RESET, ALE, ICP, входы и выходы последовательных каналов блоков SPI и UART, линия канала последовательного программирования флэш-памяти и EEPROM, линии передачи аналоговых сигналов и др. являются однонаправленными.

Ряд блоков AVR аналогичен блокам микропроцессора, рассмотренным ранее в этой же главе, и имеет те же самые обозначения. Программный счетчик PC содержит адрес подлежащей выполнению команды и адресует флэш-память программ. Считанная из флэш-памяти команда поступает в регистр команд IR, ее КОП (код операции)

декодируется дешифратором команд для выработки сигналов управления блоками МК соответственно заданной операции, а КАД (код адреса) адресует данные в блоке регистров или в памяти данных SRAM.

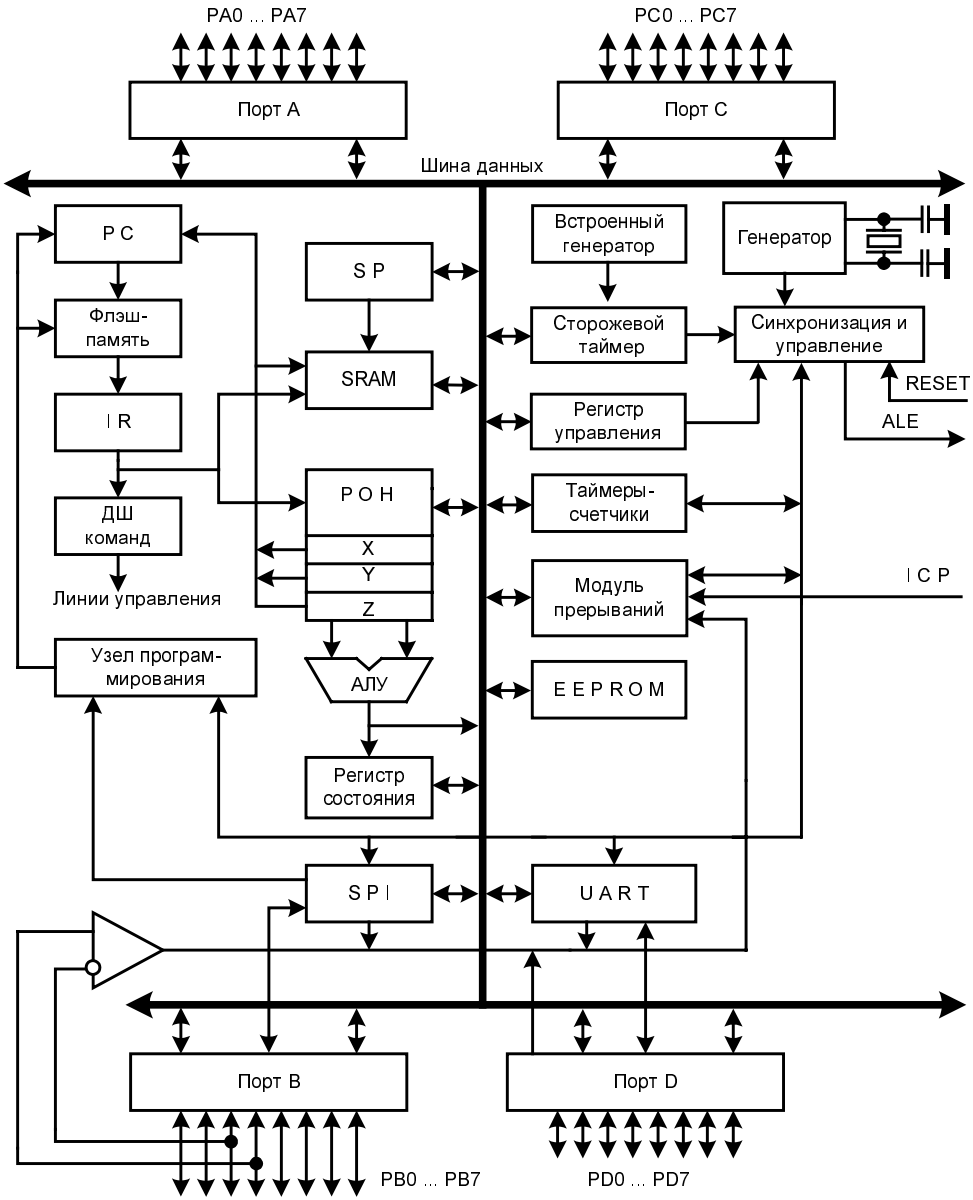


Рис. 6.28. Структура микроконтроллера AVR

В памяти EEPROM хранятся редко изменяемые данные (калибровочные константы и т. п.). Указатель стека SP используется для организации стека в некоторой облас-

ти SRAM, глубина стека ограничивается только наличием свободной области в этой памяти. Регистры общего назначения (РОН) объединены в регистровый файл.

Арифметико-логическое устройство АЛУ способно выполнять операции над содержимым любой пары регистров блока и направлять результат в любой регистр, т. е. *все регистры РОН непосредственно доступны для АЛУ* (у многих МК и МП рабочим регистром для АЛУ служит только один регистр — аккумулятор). Наличие у АЛУ многих рабочих регистров упрощает выполнение операций за один такт. Три регистровые пары (X, Y, Z), получаемые объединением двух 8-разрядных регистров в один 16-разрядный, используются для косвенной адресации. Регистр состояния функционально аналогичен регистру флажков RF в рассмотренном ранее МП, он содержит признаки результатов при выполнении некоторых команд (нуль, знак, перенос, половинный перенос и т. д.).

Генератор синхросигналов имеет внешние выводы для подключения кварцевого или иного резонатора либо внешнего тактирующего сигнала. Кроме основного синхрогенератора МК имеет и дополнительный встроенный RC-генератор с фиксированной частотой 1 МГц (при напряжении питания 5 В) для тактирования сторожевого таймера. L-активный вход RESET служит для сброса МК (приведения его в исходное состояние), а также перевода его в режим программирования при подаче на этот вход специального повышенного напряжения 12 В. Выход ALE имеет то же назначение, что и одноименный выход рассмотренного ранее микропроцессора и используется при подключении к микроконтроллеру внешнего ЗУ. В этом случае реализуется режим мультиплексируемой шины: старший полуадрес выводится в течение всего цикла через один из 8-разрядных портов, а младший — через второй 8-разрядный порт загружается в начале цикла во внешний регистр-защелку, где сохраняется на все время цикла. После загрузки внешнего регистра-защелки этот же порт используется для передачи данных и сигналов управления.

Модуль прерываний служит для приема и обработки запросов прерывания основной программы, как внутренних, так и внешних. Предусмотрено наличие 10 внутренних и двух внешних запросов.

Блоки, относящиеся к центральному процессорному элементу, рассматривались при описании МП. Порты ввода/вывода, таймеры, интерфейс SPI, блоки UART рассмотрены в главе 7.

Структура одного из четырех портов ввода/вывода МК AVR приведена на рис. 6.29. Порт представляет собою набор из 8 линий. Каждая из 8 линий любого порта конфигурируется как входная или выходная индивидуально с помощью управляющего слова, загружаемого в регистр направления передачи. Каждый бит этого слова задает конфигурацию своей линии. Выводимые или вводимые данные поступают в регистр данных. Входные и выходные сигналы проходят через буферные каскады (драйверы).

Для уяснения процессов работы основных цепей в линиях порта следует обратиться к рис. 7.3 и тексту к нему, а затем к более полному рис. 7.4. В состав МК AVR входят таймеры (таймер 0, таймер 1 и сторожевой таймер), структуры которых см. на рис. 7.36, 7.37, 7.39.

Синхронный интерфейс *SPI* (Serial Peripheral Interface) (см. *главу 7*) применяется как для программирования микроконтроллера в последовательном режиме, так и для обмена данными с периферийными устройствами или между микроконтроллерами. Протокол обмена предусматривает работу МК в режиме либо ведущего (Master), либо ведомого (Slave). Скорости передачи задаются синхросигналами, получаемыми делением тактовой частоты МК, и имеют четыре программируемых значения: 1/4, 1/16, 1/64 и 1/128 от этой частоты.

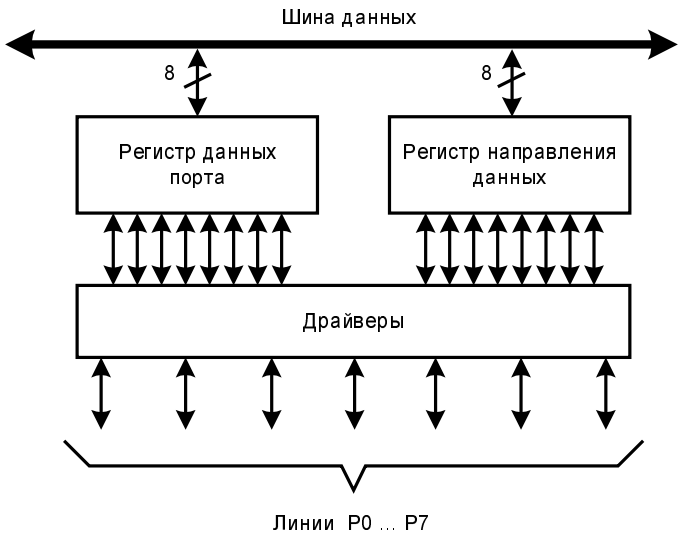


Рис. 6.29. Структура порта ввода/вывода микроконтроллера AVR

Асинхронный последовательный интерфейс между МК и внешними устройствами обеспечивается блоком *UART* (Universal Asynchronous Receiver/Transmitter) (см. *главу 7*), преобразующим параллельные данные от микроконтроллера в последовательные для внешнего устройства и наоборот. Блок UART работает по асинхронному протоколу обмена с квитированием для учета готовности блоков к пересылке очередного байта.

Аналоговый компаратор сравнивает напряжения на двух специально выделенных линиях порта B. Одно из этих напряжений сигнальное, другое опорное. Выходной сигнал компаратора имеет логический характер, показывает, какое из двух напряжений больше, доступен программе и поступает на блок прерываний. С помощью аналогового компаратора можно, например, измерять длительность входных импульсов, поскольку начало и конец импульса могут быть отмечены изменением состояния компаратора. Сигналы компаратора могут служить источниками команды захвата для таймера, что позволит зафиксировать цифровые значения моментов начала и конца импульса. Можно возразить, что эта задача решается и самим таймером без аналогового компаратора, но это так только для импульсов, у которых уровни напряжения соответствуют уровням логических сигналов. При использовании аналогового компаратора таких ограничений нет и можно измерять длительно-

сти импульсов с произвольными значениями высокого и низкого уровней, установив величину опорного напряжения между ними.

§ 6.8. Организация памяти и функционирование МК

Распределение памяти в МК AVR

Согласно Гарвардской архитектуре МК имеет отдельные адресные пространства (АП) для программ и данных (рис. 6.30). Память данных организована линейно и имеет два АП. В первом находятся адреса регистровой памяти (РОН и РВВ — регистров ввода/вывода) и статического ЗУ (SRAM). Во втором АП размещены адреса энергонезависимой репрограммируемой памяти EEPROM. Кроме того, возможно подключение к МК внешнего ОЗУ, для которого шина адреса и мультиплексируемая шина адресов/данных организуются с помощью портов ввода/вывода PA и PC. Линии этих портов формируют 16-разрядные адреса для работы с внешней памятью емкостью до 64 Кбайт. При отсутствии внешней памяти достаточны 9-разрядные адреса.

В регистровой области памяти данных размещены адреса регистров общего назначения РОН (32 адреса для байтов) и регистров ввода/вывода РВВ (64 адреса для байтов). Для косвенной адресации служат регистры X, Y и Z, представляющие собой 16-разрядные пары байтовых регистров. Общая емкость регистровой памяти составляет 96 байтов. Для адресов статической памяти SRAM отведены следующие 512 адресов. Подключение внешнего ОЗУ, как уже отмечалось, может довести емкость памяти до 64 Кбайт. Обращение к внешнему ОЗУ увеличивает время выполнения команды на 1—2 такта для каждого обрабатываемого байта.

При обращении к разным областям памяти данных используются команды с различными способами адресации. Адреса области РВВ являются операндами команды (прямая адресация).

В пространстве РВВ размещены служебные регистры МК и регистры, относящиеся к внешним устройствам. В их числе 12 регистров для работы с портами ввода/вывода (таких портов 4, для каждого предусматриваются регистр данных, регистр направления данных и регистр выводов, функции которых рассмотрены в *главе 7*). Имеется регистр — указатель стека. Глубина стека определяется наличием свободной области памяти программ. Многочисленные регистры обеспечивают работу модулей SPI, UART, таймеров, стека, имеются регистры состояния и управления микроконтроллером, регистр флагов и масок запросов прерывания, регистр управления памятью EEPROM. Для управления микроконтроллером служат регистры разрешения внешнего ОЗУ, разрешения перехода к режиму пониженного энергопотребления и выбора конкретного варианта из таких режимов, условий генерации запросов внешних прерываний (по фронту сигнала, по уровню и др.).

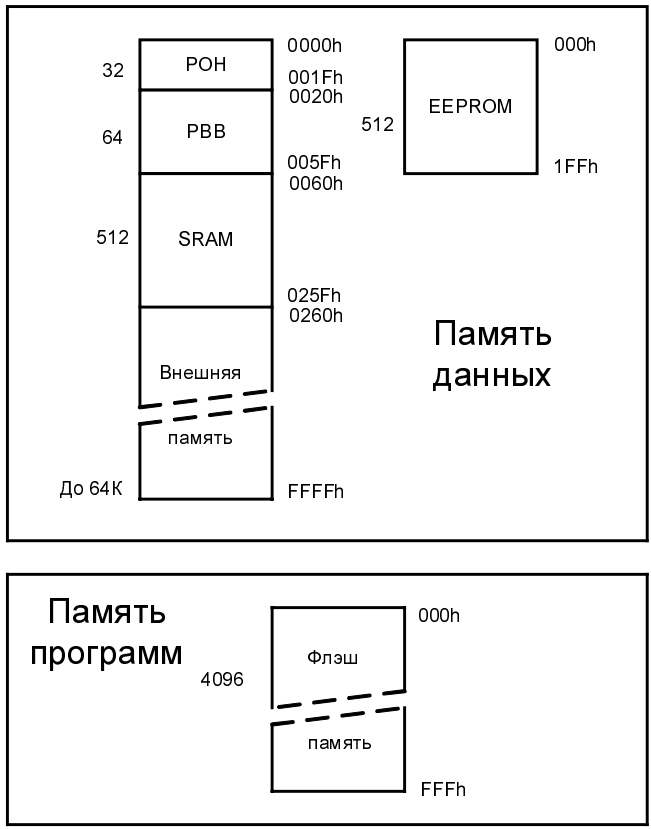


Рис. 6.30. Адресные пространства микроконтроллера AVR

К РОН и РВВ возможны обращения как по командам ввода и вывода `IN` и `OUT`, так и к ячейкам ЗУ. При этом для разных способов обращения диапазоны адресов РВВ несколько смещены (на 32 адреса).

Энергонезависимая память EEPROM отличается особой организацией. Доступ к ней происходит с использованием трех регистров области РВВ. Это регистр адреса (составленный из пары регистров, т. к. емкости памяти в 512 байт соответствует девятиразрядный адрес), регистр данных и регистр управления. Регистр данных служит для размещения данных, подлежащих записи в память или считываемых из нее. В регистре управления задействованы три бита: один определяет наличие или отсутствие разрешения записи, второй разрешает саму запись, третий разрешает чтение. Установка второго и третьего битов вызывает выполнение соответствующих процессов. Длительность цикла записи при напряжении питания 5 В составляет 2 мс, при 2,7 В — 4 мс. Чтение выполняется за один такт синхронизации.

Программы хранятся во флэш-памяти, разрядность которой соответствует формату команд и равна 16. По начальному адресу 000h расположен вектор сброса, конечный адрес равен FFFh, т. е. емкость памяти составляет 4096 слов или 8192 байта (организация флэш-памяти 4К×16).

Способы адресации

В микроконтроллере применяются прямая и косвенная адресации.

Прямая адресация используется в следующих случаях:

- при адресации одного РОН, занимающей в команде 5 бит, или при адресации двух РОН с занятием 10 бит (по 5 бит для адресов источника и приемника данных);
- при пересылке между регистрами РВВ и РОН с занятием 11 бит в слове команды (6 на адрес РВВ и 5 на адрес РОН);
- при обращении к всему адресному пространству ОЗУ с использованием двух слов команды, из которых второе целиком отдается под адрес ячейки памяти.

Косвенная адресация с использованием индексных регистров X, Y, Z применяется в следующих вариантах:

- простая (адрес находится в индексном регистре);
- относительная (адрес вычисляется как сумма содержимого индексного регистра и константы, содержащейся в команде);
- с преддекрементом (до обращения к ячейке памяти адрес в индексном регистре уменьшается на единицу);
- с постинкрементом (после обращения по адресу в индексном регистре его содержимое увеличивается на единицу).

Два последних варианта косвенной адресации позволяют эффективно обрабатывать массивы данных.

Выполнение команд

Повышению производительности МК способствует *выполнение почти всех команд за один такт машинного времени*. Такая возможность — следствие конвейерной обработки информации и непосредственного подключения АЛУ ко всем РОН. При работе конвейера параллельно выполнению текущей команды производится выборка и декодирование следующей. При нарушении естественного следования команд (выполнении команд переходов) в работе конвейера возникает разрыв и уже выбранная для очередной операции команда не может быть использована и приходится возвращаться к выбору новой, нарушая нормальный ритм работы конвейера. Время выполнения команды увеличивается на 2…4 такта.

Начало выполнения программы инициируется *сигналом сброса*, после которого МК обращается к адресу стартовой команды. Сброс вызывается несколькими причинами — включением питания, внешним сигналом, сигналом от сторожевого таймера, снижением питающего напряжения ниже допустимого уровня. После события, вызывающего сброс, запускается *таймер задержки сброса* и лишь после отработки таймером интервала задержки формируется внутренний сигнал сброса. Задержка позволяет микроконтроллеру принять исходное состояние, необходимое для нор-

мального начала работы. Таймер задержки сброса работает от внутреннего RC-генератора, основной генератор за время задержки выходит на стационарный режим. Относительно большое время выхода на стационарный режим характерно, в частности, для генератора с кварцевой стабилизацией частоты.

В ходе выполнения программы контроллер выполняет команду за командой, пока не дойдет до команды останова. Все *множество команд*, выполняемых контроллером, можно разбить на следующие группы:

- команды пересылки данных;
- команды арифметических операций и сдвигов;
- команды логических операций;
- команды операций с битами;
- команды передачи управления;
- команды управления системой.

Сопоставляя перечисленные группы команд с группами команд микропроцессора, рассмотренного ранее в этой главе, можно видеть их большое сходство. Специфичная группа команд операций с битами содержит команды установки или сброса заданного разряда в регистре РОН или РВВ. Для удобства программирования задействованным разрядам регистров РВВ присваиваются символические имена, определение которых дается в специальном файле, подключаемом в начале программы. Детальное описание команд приводится в работах [12], [20], [37] и в справочных данных фирмы Atmel.

Режимы потребления мощности

Микроконтроллер может находиться в нескольких *режимах потребления мощности.* В активном режиме потребляется полная мощность, кроме того имеются режимы покоя (Idle) и существенного понижения мощности (Power Down). В режимы пониженного потребления мощности ("спящие") МК переходит по команде `SLEEP`, которая устанавливает в регистре управления контроллером два флажка — флажок перехода к режиму пониженного потребления мощности и флажок, задающий конкретный тип такого режима.

В *режиме покоя* останавливается работа процессора, а периферийные устройства и система прерывания функционируют. Поэтому выход из режима покоя возможен по запросам прерывания от внутренних и внешних источников, в частности, от таймера. "Пробуждение" микроконтроллера из состояния покоя является быстрым.

В *режиме существенного понижения мощности* прекращается работа всех блоков контроллера кроме сторожевого таймера и подсистемы обработки внешних прерываний. Выход из этого режима возможен по внешнему сигналу сброса, сигналу сброса от сторожевого таймера или внешним прерываниям. "Пробуждение" из режима существенного понижения мощности является медленным, для него требуется интервал времени, соответствующий задержке таймера сброса.

"Спящие" режимы позволяют резко уменьшить потребляемые токи в периоды бездействия микроконтроллера. Это особенно полезно при использовании МК в устройствах с автономным питанием. Если выход из "спящего" режима происходит по событию прерывания, МК переходит в рабочий режим выполнения соответствующей подпрограммы и затем возобновляет выполнение основной программы с той команды, которая стояла после команды `SLEEP`. Если же "пробуждение" является результатом процесса сброса, то происходит реинициализация контроллера.

Система прерываний

Микроконтроллер имеет *многоуровневую систему приоритетных прерываний*. Под таблицу векторов прерываний отводятся младшие адреса памяти программ, начиная с адреса 001h. Каждое прерывание имеет свой адрес в таблице, который загружается в программный счетчик для обслуживания прерывания. Приоритетность запросов прерываний непосредственно связана с их адресами, чем меньше адрес, тем выше приоритет запроса. Предусмотрены следующие прерывания: два внешних, четыре связанных с таймером 1 (при захвате, совпадениях A и B, переполнении), одно, связанное с таймером 0 (при переполнении), прерывание при завершении передачи по интерфейсу SPI, три связанных с блоком UART (при завершениях приема и передачи и при пустом регистре данных), прерывания от аналогового компаратора.

Программирование МК

Программирование микроконтроллера предусматривает запись машинных кодов в память программ и необходимых данных в энергонезависимую память EEPROM. Существуют два варианта программирования: параллельное при высоком напряжении и по последовательному каналу. Первый вариант требует применения специального программатора и характерен для условий массового производства. *Программирование по последовательному каналу* осуществляется через интерфейс SPI, не требует использования высокого напряжения, производится непосредственно в системе и в силу своих достоинств наиболее удобно для модернизации программ пользователями. Схема включения микроконтроллера в режим программирования по последовательному каналу показана на рис. 6.31.

Тактовый сигнал может подаваться извне, создаваться подключением резонатора к выводам XTAL1 и XTAL2 корпуса МК или восприниматься от его внутреннего RC-генератора. Тактирующий сигнал интерфейса SPI (сигнал SCK) должен иметь длительность обоих уровней более двух периодов тактового сигнала. Программирование состоит в посылке 4-байтовых команд на вывод MOSI. Результаты чтения снимаются с вывода MISO. Программирование памяти команд и памяти данных ведется побайтно посылкой команд "Запись флэш-памяти" и "Запись EEPROM". В этих командах содержатся адреса изменяемых ячеек и записываемое в них содержимое.

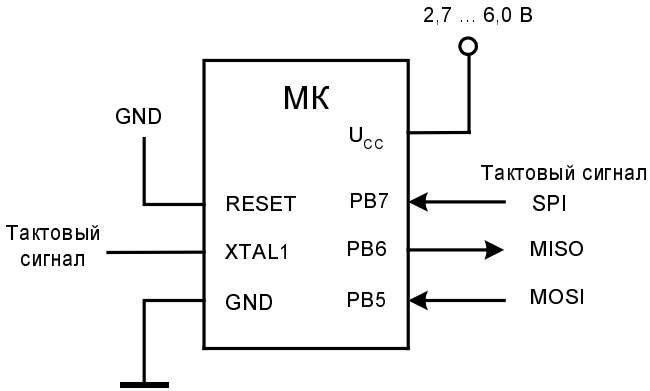


Рис. 6.31. Включение МК в режим программирования по последовательному каналу

Контрольные вопросы и упражнения

1. Какие основные архитектурные разновидности процессоров используются в микропроцессорных системах?
2. Какие запоминающие устройства применяются в микропроцессорных системах? Каково их назначение?
3. Какие сигналы микропроцессора связаны со следующими задачами:
   - синхронизацией;
   - прерыванием;
   - прямым доступом к памяти;
   - введением тактов ожидания.
4. Чем объясняется широкая распространенность магистрально-модульных структур в микропроцессорных системах?
5. Каковы особенности и назначение регистра-аккумулятора в схеме микропроцессора?
6. Какие регистры микропроцессора называются регистрами общего назначения (РОН)? В чем состоит роль этих регистров?
7. В чем состоят функции регистра команд и программного счетчика в схеме микропроцессора?
8. Каково основное назначение стека в структуре МПС? В чем состоит назначение регистра указателя стека?
9. В какие блоки микропроцессора поступают первый, второй и третий байты команды и каково содержимое этих байтов для разных форматов команд (однобайтной, двухбайтной, трехбайтной)?

10. Какие действия выполняет автомат управления микропроцессора в первых трех тактах машинного цикла?

11. Какие изменения в работу МП вносит отсутствие сигнала READY на соответствующем входе?

12. Каков применяемый в МПС метод формирования сигнала неготовности устройств к обмену с процессором?

13. В чем состоит различие реакций МП на запросы радиальных и векторных прерываний?

14. Прямая регистровая адресация имеет большие достоинства. Почему этого нельзя сказать о простой прямой адресации?

15. Какую адресацию называют "абсолютной" и какую "не абсолютной"? Чем они отличаются друг от друга?

16. Какой процесс называют "прямым доступом к памяти"? Какие преимущества дает использование этого процесса в МПС?

17. Адресное пространство МПС составляет 4Г. Запишите для этого АП максимальное значение адреса в 16-ричной системе счисления.

18. Требуется составить модуль памяти с организацией $2^m \times n$ с применением микросхем памяти с организацией $2^{m/4} \times n/4$. Сколько микросхем понадобится для построения модуля?

19. Составьте схему подключения к микропроцессору модулей ПЗУ с организацией 256К× 16 и ОЗУ с организацией 64К× 8 при использовании микросхем постоянной памяти с организацией 64К× 8 и микросхем ОЗУ с организацией 32К× 8. Адреса модулей (сначала ПЗУ, затем ОЗУ) следует разместить в начале второй четверти адресного пространства емкостью 4Г.

20. Какие сведения о схеме МПС и какие параметры микропроцессора и ЗУ потребуются для анализа пригодности ЗУ к работе в данной системе?

21. Как расшифровываются обозначения сигналов IBF и OBF в схемах условного ввода/вывода? Как они используются в этих схемах (назовите два варианта)?

22. В чем состоят основные отличия микроконтроллеров от универсальных компьютеров?

23. Какие архитектуры характерны для современных микроконтроллеров? В чем состоят особенности этих архитектур?

24. Какие схемотехнические виды памяти применяются в современных микроконтроллерах?

25. Какие факторы способствуют возможностям выполнения команд за один такт синхронизации микроконтроллера?

26. Каков типичный состав периферийных устройств в составе микроконтроллеров?
27. В чем состоит назначение сторожевого таймера? Каким образом сторожевой таймер выполняет возложенные на него функции?
28. Какую роль в схеме микроконтроллера играет таймер задержки сброса?
29. Какие способы адресации типичны для микроконтроллеров?
30. Какие приемы используются для снижения мощности, потребляемой микроконтроллерами?

Литература к главе: [11], [12], [14], [18], [19], [20], [27], [28], [29], [30], [37], [39], [53], [IV], [VI], [XV], [XIX], [XXXI], [XXXV].

Глава 7

Интерфейсные схемы, адаптеры, контроллеры

§ 7.1. Общие сведения

С обменом информацией между цифровыми устройствами связано понятие стандартного интерфейса. *Интерфейс* — совокупность аппаратных и программных средств, обеспечивающих функциональную, электрическую и механическую совместимость устройств, обменивающихся информацией. К основным элементам интерфейса относят протоколы обмена (совокупность правил, регламентирующих способ выполнения заданных функций), аппаратную часть (физическую реализацию устройств) и программное обеспечение.

Интерфейсы классифицируются по конфигурации связей между объектами (магистральные, радиальные, кольцевые и др.), типу передаваемой информации (параллельные, последовательные, параллельно-последовательные), режиму передачи данных (дуплексный, полудуплексный, симплексный), способу обмена (асинхронные, синхронные).

Согласно областям применения выделяют несколько классов интерфейсов. Интерфейс межмодульного обмена в микропроцессорных системах называют *системным (внутренним)*.

Для МПС первого поколения были разработаны интерфейсные схемы семейства Intel 82XX (цифра 82 обозначает принадлежность к интерфейсным схемам, а цифры на позициях, отмеченных крестиками, определяют конкретный тип схемы). Базовые структуры семейства 82XX получили широкое распространение как в виде отдельных микросхем, так и в виде макрофункций (мегафункций, IP-блоков), т. е. в виде данных для настройки программируемых структур на выполнение требуемых интерфейсных функций.

Интерфейсы микропроцессорных систем

Системные интерфейсы появились в конце 70-х годов для МПС на основе 8-разрядных микропроцессоров Intel 8080, Motorola 6800 и др. Первый магистральный параллельный интерфейс *Microbus* был рассчитан на объединение в систему не более 10 модулей, расположенных в непосредственной близости друг от друга. Ин-

терфейс имел 16-разрядную шину адреса, 8-разрядную шину данных и линии управляющих сигналов CLK, RESET, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$, INT, $\overline{\text{INTA}}$, HOLD, $\overline{\text{HLDA}}$, RDY, $\overline{\text{BUSEN}}$, которые уже рассматривались при описании микропроцессора. В интерфейсе адресные пространства памяти и ВУ разделены, выполняются протоколы программного обмена, обмена по прерываниям и прямого доступа к памяти.

Позднее в связи с ростом разрядности и быстродействия процессоров был разработан и ряд других интерфейсов. Для персональных компьютеров PC/AT это интерфейс (шина) *ISA* (Industrial Standard Architecture). Появление 32-разрядных процессоров (80386 и далее) ассоциируется с шиной *EISA* (Extended ISA).

На уровне локальных шин компьютеров с 1992 г. стала широко применяться шина *PCI* (Peripheral Component Interconnect) — интерфейс с новыми качествами, не зависящий от платформы (т. е. способный работать с разными процессорами), более производительный и недорогой в производстве. Концептуальное достоинство шины — способность выполнять некоторые действия без обращения к процессору, тем самым уменьшая его загрузку; применение для связи с другими компонентами системы так называемых мостов; реализация принципов ведущей и ведомой шин и др. Шина приспособлена к распознаванию аппаратных средств системы и анализу ее конфигурации согласно стандарту Plug&Play. Малопроизводительный вариант шины имел частоту тактирования 33 МГц и разрядность 32, более производительный — 66 МГц и 64 разряда. Важную роль, особенно для военно-промышленных применений, сыграла и шина *VMEbus*. Тактовая частота системных шин стала составлять 66...133 МГц, появились и шины с тактовой частотой 166 МГц.

Благодаря постоянным модернизациям (варианты PCI-X и ряд других) параллельная шина PCI сохранилась и до нашего времени, причем ее тактовая частота для варианта *PCI-X 2,0* доведена до 533 МГц, что при разрядности 64 соответствует производительности шины 4,26 Гбайт/с. Однако дальнейшее повышение производительности стало даваться с таким трудом, что на смену модернизации параллельной шины пришел новый подход — применение высокоскоростных структур с последовательной передачей данных (шина *PCI Express*, первые устройства с которой появились в 2004 г., и др.). Основа передач в шине PCI Express — две пары дифференциальных линий стандарта LVDS, использующих протокол дуплексных передач типа "точка-точка" при тактовой частоте 2,5 ГГц и в перспективе 10 ГГц. Шина ориентирована на передачи ЦПУ-ВУ, память-ВУ, ВУ-ВУ, межкристальные и межплатные передачи.

Средства передач в *параллельных* интерфейсах организуются в виде шин данных, адреса и управления. Шина данных передает слова между регистрами источника и приемника данных. Передатчики магистрально-модульных структур, подключенные к шине данных, имеют выходы с третьим состоянием, что дает возможность поочередно работать на одной шине нескольким передатчикам. На шине адреса выставляется информация, декодирование которой формирует сигнал разрешения работы тому или иному устройству (порту). Шина управления формирует управляющие сигналы, причем зачастую это выполняется упрощенными схемами с от-

крытым выходом (стоком или коллектором). При этом исполнительными уровнями обычно делаются нулевые, которые формируются быстро, и, кроме того, появляется возможность применения монтажной логики. Параллельные интерфейсы предназначаются для передач сигналов на малые расстояния (не более единиц метров).

В *последовательных* интерфейсах шины данных, адресов и управления отсутствуют и вся необходимая информация (данные и служебная информация) передается по одному каналу. Число линий связи в канале может быть различным. Минимальное число — одна сигнальная линия и одна линия схемной "земли". Полезная информация (данные) передается только часть времени, т. к. определенное время линия занята передачей служебной информации (адреса, сигналы управления). Увеличение числа линий в канале увеличивает пропускную способность интерфейса, но одновременно увеличивает число выводов микросхемы и потребляемую ею мощность.

Если устройства обработки данных и каналы передачи оперируют со словами разной разрядности, то требуются схемы преобразования разрядностей слов — блоки SERDES. Благодаря преобразованиям разрядностей слов можно строить интерфейсы с гибким соотношением разрядностей обрабатываемых и передаваемых слов.

В системах высокой производительности для увеличения пропускной способности связей между блоками, кристаллами или печатными платами используют новые дополнительные к шинным структурам средства — технологии PCI Express, Hyper Transport, Serial RapidIO и др., имеющие радиальную структуру, т. е. обеспечивающие соединения типа "точка-точка" (point-to-point). В современных условиях реализация перечисленных технологий успешно переводится на внутрикристальные ресурсы. В этом отношении ведущую роль играет применение FPGA — микросхем программируемой логики, которые разрабатываются с учетом возможностей создания в них эффективных интерфейсных средств нового типа (быстродействующих блоков SERDES, линий передач с частотами в несколько гигагерц и др.).

Для разных областей применения в интерфейсах используются различные *среды передачи* (проводная, оптическая, каналы с инфракрасным излучением, радиоканалы). В системных интерфейсах передаются электрические сигналы по проводным связям с наличием или отсутствием согласования волновых сопротивлений в линиях.

Для любой среды максимальная достижимая частота передачи уменьшается с увеличением расстояния, на которое передаются сигналы. Соответствующая зависимость имеет вид рис. 7.1, меняются для разных сред лишь количественные данные. Для близко расположенных источника и приемника сигналов максимальная частота передачи ограничивается возможностями приемо-передающей аппаратуры, быстродействие которой конечно. При росте расстояния (длины линий передачи) максимально возможная частота падает, поскольку при удлинении линии растут помехи и ослабляется сигнал. Существует и предельное для данной среды расстояние передачи, при котором отношение сигнал/шум становится таким, что надежное восприятие сигнала уже невозможно. В качестве примера приведем параметры последовательного интерфейса RS-422A: при длине связи 12 м частота передачи достигает 10 Мбит/с, при 120 м — 1 Мбит/с, а при 1200 м — 0,1 Мбит/с.

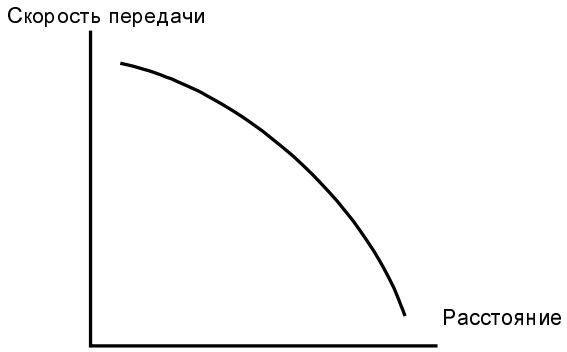


Рис. 7.1. График зависимости максимальной частоты передачи от длины линии

Обмен последовательными данными в микроконтроллерах и других устройствах управления чаще всего осуществляется с помощью интерфейсов *SPI* (Serial Peripheral Interface) и I^2C (Inter Integrated Circuits), для подключения периферийных устройств популярна шина *USB* (Universal Serial Bus), отличающаяся удобством подключения к ней дополнительных устройств. Для более дальних связей (с другими системами и т. п.) традиционны последовательные интерфейсы стандартов RS-232c, RS-485, RS-422, а для более сложных задач шина *CAN* (Control Area Network).

К наиболее употребительным интерфейсным схемам относятся шинные формирователи, буферные регистры, параллельные и последовательные порты и адаптеры, контроллеры прерываний, контроллеры прямого доступа к памяти, интервальные таймеры.

Примечание

Адаптеры и контроллеры рассматриваются в этой главе сокращенно. Цель их описания — показать существующие средства, их возможности и области применения. Не приводятся форматы управляющих слов и другие данные справочного характера. Подробное описание адаптеров и контроллеров имеется в предыдущем издании этой книги, а также в ряде других источников ([30], [39], [53] и др.).

§ 7.2. Шинные формирователи и буферные регистры

Шинные формирователи

Шинные формирователи ШФ (BD, *Bus Driver*), называемые также *приемопередатчиками*, *шинными драйверами* или *магистральными вентиль-буферами*, включаются между источником информации и системной шиной. Они усиливают сигналы по мощности и отключают источник информации от шины, когда он не участвует в обмене. Двунаправленные ШФ позволяют передавать сигналы в шину или, напротив, принимать их с шины и передавать приемнику данных.

Шинные формирователи отличаются разрядностью, передачей сигналов в прямом (для ШФ) или инвертированном (для ШФИ) виде, полярностью сигналов разрешения работы и электрическими характеристиками. В сериях КР1533, КР1554 и др. имеются ШФ и ШФИ — функциональные аналоги микросхем Intel 8286 и 8287, схема такого ШФ показана на рис. 7.2, *а*.

Шина A_{0-7} принимает данные от МП или передает их ему, шина B_{0-7} связана с магистралью, на которую передает информацию или с которой принимает ее. Сигнал $\overline{OE}$ переводит выходы усилителей в третье состояние (при его высоком уровне) либо разрешает их работу (при низком уровне). При разрешении работы направление передачи зависит от сигнала T (Transmit). Функционирование ШФ подчиняется условиям, указанным в табл. 7.1.

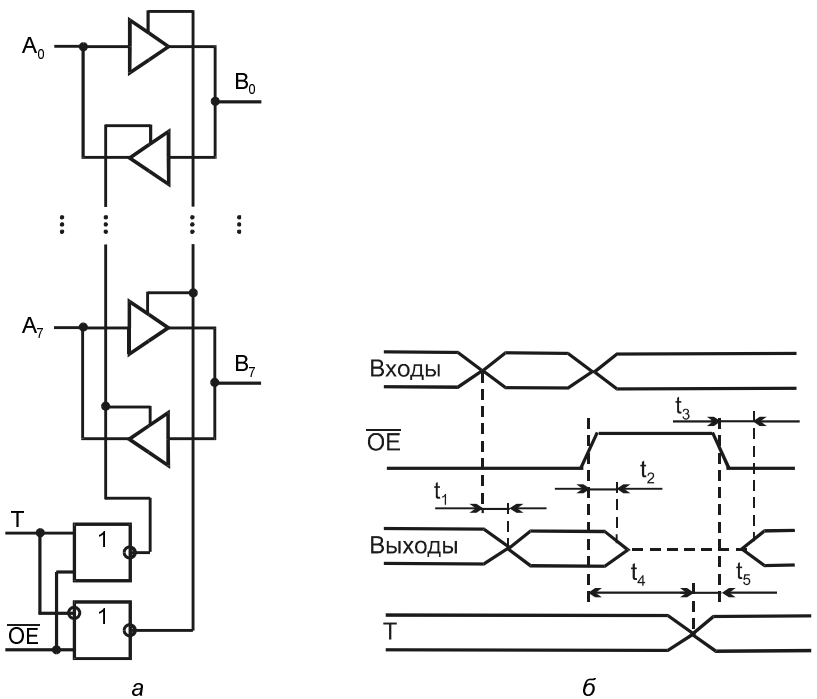


Рис. 7.2. Схема двунаправленного шинного формирователя (*а*) и временные диаграммы его работы (*б*)

Таблица 7.1

$\overline{OE}$	T	**Режим**
1	0	Нет передачи
1	1	Нет передачи
0	1	Передача от A к B
0	0	Передача от B к A

Так как шина А связана с МП, а шина В — с магистралью, то для них предусмотрена разная нагрузочная способность: выходы В обеспечивают токи большей величины, чем выходы А.

На временных диаграммах (см. рис. 7.2, *б*) показаны задержки сигналов при их распространении через открытые ШФ (задержка t_1) и задержки относительно изменений управляющих сигналов (задержка t_2 перехода в состояние "отключено", задержка t_3 выхода из состояния "отключено"). Интервалы t_4 и t_5 — времена выдержки и предустановки сигнала Т относительно моментов изменения сигнала $\overline{OE}$. Временные параметры ШФ даются для определенных нагрузочных токов (обычно максимальных) и емкостей.

Восьмиразрядный ШФ серии КР1533 (технология ТТЛШ) характеризуется следующими параметрами: выходной ток 30...112 мА; задержка распространения сигнала ≤ 10 нс; время выхода из ТС в активное состояние ≤ 20 нс; время перехода из активного состояния в ТС 25...40 нс.

Для ШФ серии КР1554 (технология КМОП) параметры таковы: выходные токи 86 мА и 75 мА для низкого и высокого уровней выходного напряжения соответственно при условии протекания не дольше 20 мс и 24 мА без ограничения времени; при напряжении питания 4,5 В задержка распространения сигнала ≤ 6 нс, задержка выхода из ТС в активное состояние ≤ 6,5 нс, а задержка перехода из активного состояния в ТС будет ≤ 8,5 нс.

Буферные регистры

Буферные регистры также работают на магистраль, но, в отличие от ШФ, способны хранить данные, т. е. выполнять их *временную* буферизацию, что составляет важную функцию портов. В сериях элементов имеются восьмиразрядные буферные регистры ИР82 и ИР83 (инвертирующий) — аналоги ИС Intel 8282 и 8283. Регистр ИР82 (рис. 7.3, *а*) принимает данные по шине A_{0-7}. Сигнал $\overline{OE}$ низким уровнем разрешает работу вентиль-буферов и тем самым передает содержимое регистра на выходную шину, а высоким уровнем переводит выходы вентиль-буферов в состояние "отключено". Прием данных в регистр разрешается сигналом строба STB.

Временные диаграммы работы буферного регистра (рис. 7.3, *б*) показывают задержку t_1 сигналов от входа к выходу при STB = 1, задержку t_2 от моментов изменения сигнала $\overline{OE}$ до перехода к режиму "отключено" и задержку t_3 до выхода из этого режима. Задержка t_4 — интервал от момента изменения строба до изменения выхода схемы. Интервал t_5 — время предустановки сигнала на входе относительно спада строба (часто не лимитировано), t_6 — время выдержки входного сигнала относительно спада строба ≥ 25 нс.

Буферные регистры широко представлены в сериях ИС. В серии КР1533 они обеспечивают выходные токи 15...70 мА при максимальных задержках от тактирующего входа около 15 нс, временах выхода из ТС около 20 нс и входа в ТС около

20...30 нс. В серии КР1554 выходные токи буферных регистров те же, что и для ШФ, задержки при U_{cc} = 4,5 В имеют порядок 10 нс.

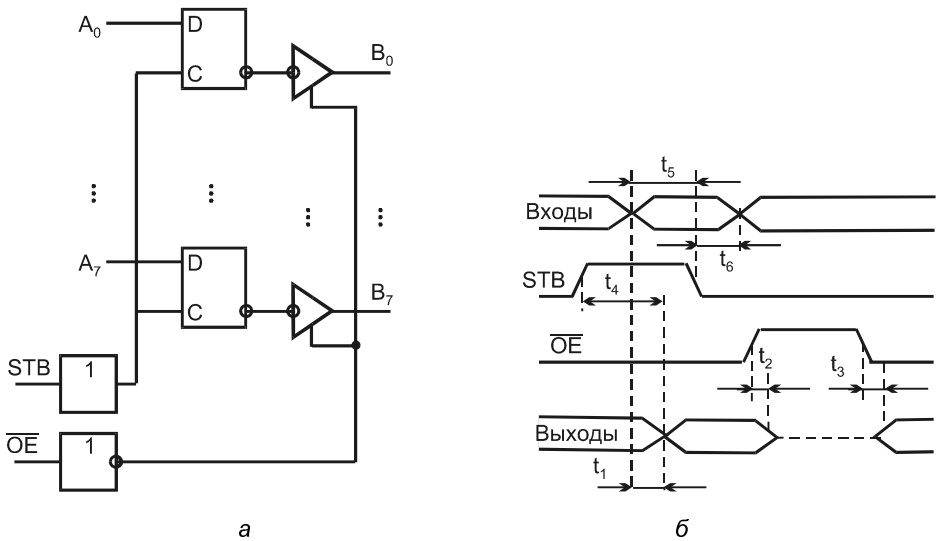


Рис. 7.3. Схема буферного регистра (*а*) и временные диаграммы его работы (*б*)

§ 7.3. Параллельные порты

Параллельные порты — основные средства обмена информацией между цифровыми устройствами в пределах компактной системы. Через порты ввода данные от ВУ поступают в магистраль, а через порты вывода данные с магистрали передаются тому или иному модулю. Порты ввода-вывода могут выполнять обе указанные операции.

Схемотехника параллельных портов разнообразна, они имеют как общие черты, так и свои особенности. В старой схемотехнике линии портов рассматривались обычно как группы *с единым управлением*, и можно было говорить о всей группе как о порте ввода либо порте вывода. Затем схемотехника портов стала более гибкой, в частности, у некоторых портов каждая линия стала *индивидуально конфигурироваться* как вход или выход, так что в составе одного и того же порта могут быть и линии ввода, и линии вывода. Кроме того, на линии портов стали возлагать и дополнительные задачи индивидуального характера.

Основные цепи конфигурируемой линии порта. В двунаправленной линии с конфигурированием на ввод или вывод (рис. 7.4) направленность линии задается битом направления передачи, загружаемым в регистр (триггер) направления передачи, тактируемый синхросигналом "Запись направления передачи".

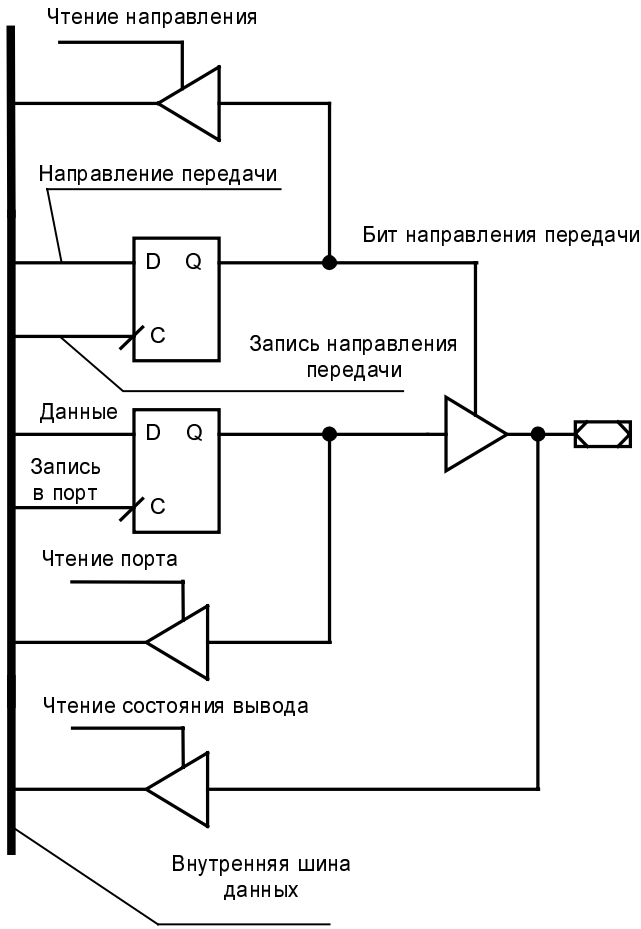


Рис. 7.4. Упрощенная схема программируемой линии порта ввода/вывода

При единичном значении этого бита работа выходного драйвера разрешена, и контакт является выходом. Состояние регистра может быть прочитано по сигналу "Чтение направления", разрешающему работу драйвера направления передачи. Выводимый бит своим синхросигналом записывается в регистр данных, состояние которого может быть прочитано по сигналу "Чтение порта". Кроме того, можно читать непосредственно состояние контакта по сигналу "Чтение состояния вывода", разрешающему работу связанного с контактом входного драйвера.

Если бит направления передачи имеет нулевое значение, выходной драйвер отключен, а вывод становится входом, передающим сигналы с контакта на шину данных под управлением сигнала "Чтение состояния вывода".

При работе с портом используются три адреса: при обращении по одному из адресов возможны ввод или вывод сведений о *направлении* передач, при обращении по второму записываются или читаются *данные* порта, по третьему производится обращение для чтения фактического *состояния внешнего контакта.*

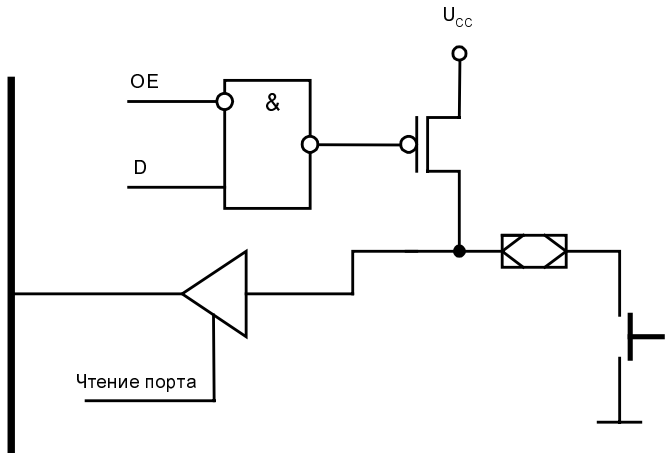


Рис. 7.5. Схема чтения состояния кнопки, подключенной к выводу порта

В линии порта с дополнительными возможностями (по сравнению со схемой рис. 7.4) вводят цепи для выполнения специальных функций. В схему линии с "подтягиванием" потенциала выходного контакта к высокому уровню вводят цепь, содержащую резистор или транзистор и источник питания. Для повышения помехоустойчивости ввода перед входным драйвером помещают триггер Шмитта, имеющий передаточную характеристику с петлей гистерезиса и т. д. Иногда в связи с тем, что на линии порта возлагаются дополнительно новые функции, схемы линий значительно усложняются.

Схема опроса состояния кнопки. Вариант подтягивающей цепочки может быть использован при вводе в систему сведений о состоянии кнопки, подключенной к внешнему контакту (рис. 7.5). Поставив линию в режим ввода и записав 1 в регистр порта, можно воспринять состояние кнопки. Если она замкнута, то введется нуль, а если разомкнута, то 1 (сопоставляя эту схему со схемой, содержащей подтягивающую цепочку с резистором, подключенным к источнику питания, видим, что в данном случае роль резистора играет сопротивление открытого транзистора).

§ 7.4. Параллельные адаптеры

Шинные формирователи осуществляют непосредственную, а порты буферизованную во времени передачу данных между МП и системной шиной. Более сложные операции выполняются периферийными адаптерами. *Программируемость адаптеров* обеспечивает им широкую область применения вследствие изменяемости процедур обмена с помощью команд программы, в том числе и во время работы микропроцессорной системы.

При обмене параллельными данными обычно используется базовая структура *параллельного периферийного адаптера* (ППА, *PPI*) Intel 8255A (отечественные аналоги ВВ55А). ППА — это устройства ввода/вывода, обеспечивающие двунаправленный об-

мен (непосредственный или с квитированием) при программном обмене или по запросам прерывания. С помощью ППА внешние устройства, работающие с параллельными кодами, связываются с магистралью системы.

Структура адаптера

Адаптер 55А (рис. 7.6) имеет три двунаправленных 8-разрядных порта (канала) PA, PB и PC, причем порт PC разделен на два четырехразрядных канала: старший PC_H и младший PC_L. Обмен информацией между каналами и системной шиной данных МПС производится через буфер данных BD.

Блок Чт/Зп управления чтением/записью получает сигналы чтения и записи $\overline{RD}$ и $\overline{WR}$ ($\overline{IOR}$ и $\overline{IOW}$ в стандартном интерфейсе), сигналы сброса RESET и выбора адаптера $\overline{CS}$ и два младших разряда адреса (A_1 и A_0) для адресации внутренних регистров. Внутренних адресуемых объектов 5: три порта, регистр управляющего слова РУС и регистр команды BSR установки/сброса битов порта С (Bit Set/Reset). Адресу $A_1A_0 = 11$ соответствует запись управляющих слов УС1 в РУС или УС2 в BSR. Запись двух разных слов при одном и том же адресе возможна потому, что признаком того или иного УС служит значение старшего бита передаваемого слова (D_7), который, таким образом, выполняет дополнительную адресацию управляющих слов.

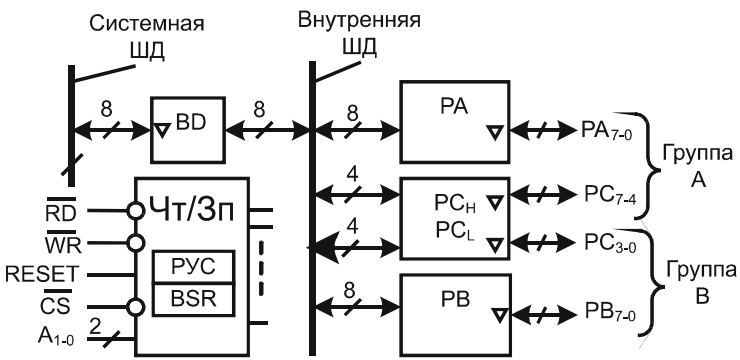


Рис. 7.6. Структура параллельного периферийного адаптера

Сигнал RESET сбрасывает регистр РУС и устанавливает для всех портов режим простого ввода. Работа адаптера начинается после загрузки с ШД в РУС управляющего слова, задающего портам адаптера один из трех возможных режимов и направленность порта (ввод или вывод).

Режимы работы портов

Возможны *три режима работы портов*: 0, 1 и 2, причем порт А может работать в любом режиме, порт В только в двух (0 и 1), а порт С работает в режиме 0 или об-

служивает другие порты, передавая для них управляющие сигналы. Такая возможность обеспечивается тем, что разряды порта С могут *программироваться и использоваться поодиночке.* Любой из восьми разрядов порта С может быть установлен или сброшен программным способом (командой `BSR`). Это нужно для передач сигналов квитирования при обмене через порты А и В в режимах 1 и 2.

Режимы работы портов:

- *режим 0* — однонаправленный ввод/вывод без квитирования, в этом режиме могут работать порты А и В, а также свободные (не занятые передачей служебных сигналов для портов А и В) линии порта С;
- *режим 1* — однонаправленный ввод/вывод с квитированием;
- *режим 2* — двунаправленный ввод/вывод с квитированием.

При работе порта в режиме 1 для него под сигналы управления требуются три линии, в режиме 2 — пять. Квитирование позволяет вести асинхронный обмен с учетом готовности абонента к передаче.

Разряды управляющего слова УС1 определяют режимы (ввод или вывод) и направление передач портов РА и РВ и свободных от служебных сигналов линий порта С. Управляющее слово УС2 записывает 0 или 1 в один из разрядов порта С. Для приведения в определенное состояние нескольких разрядов нужно подать в адаптер соответствующее число слов УС2. В этом слове размещается двоичный код номера разряда, приводимого данным УС2 в состояние D, и само значение D (0 или 1).

Режим 0

Режим 0 — безусловный однонаправленный ввод-вывод. Каждый из 4-х портов может быть использован для ввода или вывода независимо от других, так что возможны 16 вариантов режима 0. При вводе поступающая из ВУ информация адаптером не фиксируется и должна присутствовать на входе порта. При выводе данные фиксируются в регистрах портов и сохраняются до нового цикла вывода или смены режима работы порта. При вводе информация выдается на ШД при выполнении микропроцессором команды `IN port`, а при выводе — при выполнении команды `OUT port`. Такой вариант соответствует работе "с раздельной шиной", но не исключается и организация интерфейса "с общей шиной". Режим 0 не требует особых пояснений — это режим простой передачи, в котором порты адаптера подобны шинным формирователям.

Режим 1

В режиме 1 каждая из двух 12-разрядных групп (А и В) может быть запрограммирована на однонаправленный ввод или вывод с квитированием. Входные и выходные данные фиксируются адаптером. По линиям портов C_H и C_L, которым придается определенное функциональное назначение, передаются управляющие сигналы. Оставшиеся свободными линии могут быть использованы в режиме 0.

Режим 2

В режиме 2, возможном для порта A, осуществляются двунаправленные передачи с квитированием. При этом 5 линий порта C передают управляющие сигналы.

Работа адаптера в режиме 1

При *вводе* используются следующие управляющие сигналы:

- $\overline{\text{STB}}$ — строб загрузки данных в регистр (по заднему фронту), для приема этого сигнала используется разряд PC4 порта C;
- IBF (Input Buffer Full) — входной буфер полон, сигнал подтверждения загрузки данных, формируется в разряде PC5 порта C;
- INT — запрос прерывания, вырабатывается на линии PC3 порта C.

Временные диаграммы процесса ввода в режиме 1 показаны на рис. 7.7.

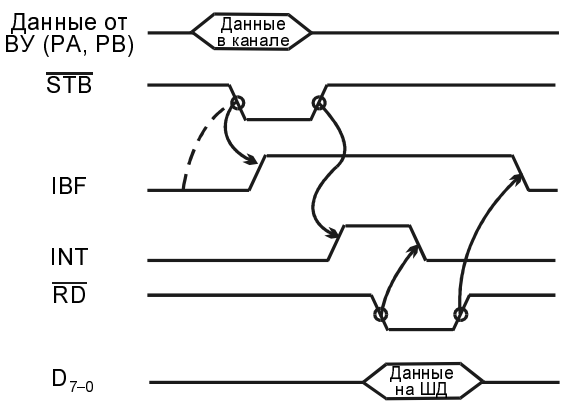


Рис. 7.7. Временные диаграммы процесса ввода и вывода для режима 1 ППА

Имея данные для ввода в порт, ВУ при условии IBF = 0 (показано на рисунке штриховой линией) вырабатывает сигнал $\overline{\text{STB}}$, загружающий данные в адаптер. Передний фронт этого сигнала устанавливает сигнал IBF, запрещающий внешнему устройству ввод следующего слова до освобождения порта. К моменту окончания $\overline{\text{STB}}$ данные введены в порт, и если прерывания разрешены (внутренний триггер разрешения прерываний INTE установлен командой программы), то адаптер формирует для МП запрос прерывания INT. Пока прерывания не разрешены, данные хранятся в адаптере. Подпрограмма обслуживания запроса содержит команду `IN port`. В ходе выполнения этой команды передний фронт строба $\overline{\text{RD}}$ отмечает начало считывания слова микропроцессором и снимает запрос на прерывание INT. Задний фронт $\overline{\text{RD}}$ отмечает завершение считывания слова и снимает сигнал IBF, допуская новую загрузку в адаптер слова со стороны ВУ.

Управляющие сигналы (STB, INT, IBF и т. д.) принимаются и вырабатываются выделенными для этого разрядами порта С. Слово состояния также формируется в разрядах порта С. Оно имеет разряд INTE (Interrupt Enable), установкой/сбросом которого программа может разрешать или запрещать выработку сигнала INT. Сигнал INT имеет два варианта использования: как запрос прерывания для МП и как содержимое опрашиваемого разряда слова состояния, когда процессор сам проявляет инициативу к обмену и проверяет готовность адаптера.

Сигнал IBF устанавливается адаптером, отображается в слове состояния и выдается для ВУ. Наличие этого сигнала и сигнала разрешения прерывания INTE ведет к выработке запроса на прерывание, который отображается в слове состояния и в разряде порта С. Флажок INTE создает возможность маскирования запросов прерывания, позволяющего запрещать или разрешать работу ВУ. Свободные линии порта С могут быть использованы для простого ввода/вывода. В слове состояния имеются независимые сигналы разрешения прерываний для ввода и вывода. Контроль текущего состояния портов в режимах 1 и 2 путем считывания состояния порта позволяет анализировать процесс обмена, которым можно оперативно управлять.

Пример программы. Покажем фрагмент программы инициализации адаптера и ввода через порт А в режиме 1, написанный на языке ассемблера рассмотренного ранее МП. Пусть портам адаптера присвоены адреса: порту А адрес F0h, порту В — F1h, порту С — F2h и РУС/BSR — F3h. В этих адресах значения младших разрядов A_1A_0 соответствуют требованиям к адресации портов адаптера, а старшие разряды выбраны произвольно. Формат управляющего слова УС1 определится тем, что порт А работает как порт ввода в режиме 1, что задает следующие значения битам УС1: $D_7 = 1$ (признак УС1), $D_6D_5 = 01$ (режим 1 порта А), $D_4 = 1$ (ввод для порта А). Остальные разряды не используются, их состояния безразличны и для определенности приняты нулевыми. Таким образом, УС1 = 10110000 = B0h.

Рассмотрим режим с разрешенной выработкой сигналов прерывания. Для его реализации в данном случае следует установить в единицу разряд PC4, содержащий величину INTA, для чего потребуется выполнить команду `BSR` формата 0XXX1001 ($D_3D_2D_1$ — номер устанавливаемого разряда, D_0 — его значение), которую запишем как 00001001 = 09h.

Ввод в адаптер управляющих слов (УС1 и УС2) производится с помощью последовательности двух команд: непосредственной загрузки аккумулятора и вывода данных в адресованный порт

```
MVI A, b2
OUT port,
```

где загружаемый в аккумулятор байт b_2 представляет собой вводимое в адаптер слово УС1 или УС2, а `port` — адрес регистров управления, шесть старших разрядов которого дают номер (адрес) адаптера, а два младших содержат единицы. Указанный фрагмент программы повторяется столько раз, сколько необходимо для задания адаптеру режима и функций, а выходам порта С нужных значений.

Фрагмент программы имеет вид:

```
MVI  A, 0B0h        ; Пересылка УС1 в аккумулятор
OUT  0F3h           ; Загрузка УС1 в РУС адаптера
```

```
MVI  A, 09h        ; Пересылка BSR в аккумулятор
OUT  0F3h          ; Установка PC4 в единицу
……
M1:    IN  0F2h    ; Ввод содержимого PC в аккумулятор
       ANI  08h    ; Выделение PC3, содержащего запрос INT
       JZ  M1      ; Переход на M1, если данные не готовы
       IN  0F0h    ; Иначе ввод из порта PA
```

При *выводе* используются следующие управляющие сигналы:

- $\overline{\text{OBF}}$ (Output Buffer Full) — выходной буфер полон, строб вывода новых данных;
- $\overline{\text{ACK}}$ (Acknowledge) — подтверждение приема внешним устройством;
- INT — запрос прерывания.

Временные диаграммы вывода показаны на рис. 7.8. При выводе выполняется команда `OUT port`, и процессор устанавливает адрес порта и данные на ШД. При разрешенных прерываниях далее вырабатывается сигнал $\overline{\text{WR}}$, загружающий данные с ШД в буфер адаптера и сбрасывающий запрос прерывания INT. После окончания записи в адаптер формируется сигнал $\overline{\text{OBF}}$, указывающий на готовность данных для ВУ. Приняв данные, ВУ выдает сигнал подтверждения приема $\overline{\text{ACK}}$, снимающий $\overline{\text{OBF}}$, а по окончании сигнала $\overline{\text{ACK}}$ восстанавливается запрос прерывания (если триггер INTE установлен), что вызывает обслуживание следующего цикла вывода.

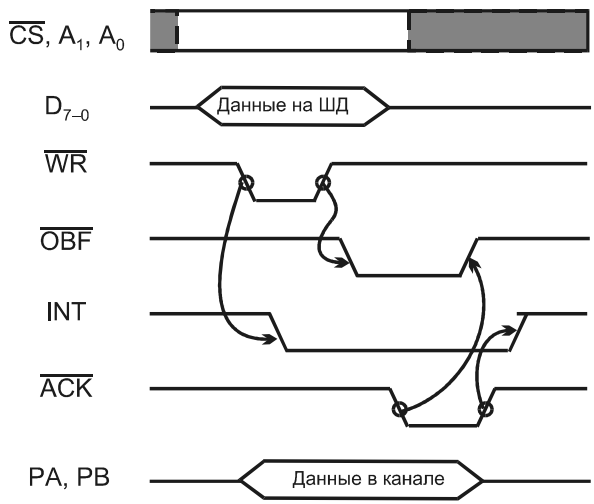


Рис. 7.8. Временные диаграммы процесса вывода для режима 1 ППА

Работа адаптера в режиме 2

В режиме 2 двунаправленный асинхронный обмен через порт A выполняется как последовательность нескольких независимых этапов (записи с ШД в адаптер, ввода

в адаптер из ВУ, чтения на ШД, вывода в ВУ), некоторые из них могут совмещаться во времени. Используются те же сигналы управления, что и для режима 1 ($\overline{\text{STB}}$, IBF, $\overline{\text{OBF}}$, $\overline{\text{ACK}}$, INT). Организация работы адаптера в режиме 2 основана на тех же принципах, что и рассмотренная выше организация работы в режиме 1.

С помощью адаптера ВВ55А могут быть выполнены требования построения интерфейсов для связи с периферийными устройствами, не слишком удаленными от МПС.

§ 7.5. Передачи последовательных данных

Передачи последовательных данных применяются по нескольким причинам:

- при увеличении расстояний, на которые передаются данные. В этом случае не удается обеспечить помехоустойчивую работу параллельных связей, кроме того, они становятся неприемлемо дорогими;
- в коротких быстродействующих связях (например, между микросхемами на одной печатной плате) с целью упрощения интерфейса, причем возможна реализация интерфейсов высшей производительности (см. *главу 1*);
- для взаимодействия с устройствами, оперирующими с последовательными кодами.

Тракты передачи последовательных данных

В варианте с использованием телефонных линий для связи удаленных объектов тракт (рис. 7.9, *а*) включает в себя источник и приемник данных (процессор и внешнее устройство), преобразователи параллельных данных в последовательные и наоборот (в этой роли выступают *программируемые связные адаптеры* ПСА) и модемы. Такой тракт соответствует взаимодействию процессора с ВУ, оперирующим параллельными кодами, но находящимся на большом расстоянии от процессора.

Модемы (модуляторы-демодуляторы) преобразуют двоичные импульсные сигналы (последовательности нулей и единиц) в аналоговый модулированный сигнал, приспособленный к передаче по узкополосным телефонным линиям (полоса пропускания около 3 кГц). Узкополосность телефонных линий ограничивает их производительность.

> Число различных состояний канала в секунду измеряют в *бодах*. Бодовая скорость в узкополосной линии невелика, и если состояния будут просто двоичными, битовая скорость передачи, измеряемая в бит/с, будет мала (будет совпадать с бодовой). С помощью разных видов модуляции (фазовой, частотной, амплитудной) и их сочетаний получают на каждом бодовом интервале сигнал с несколькими состояниями, при этом один бодовый интервал содержит несколько бит, так

что битовая скорость выше бодовой. Например, если при фазовой модуляции синусоидального сигнала на бодовом интервале задавать четыре фазы сигнала (–90°, 0°, 90° и 180°), то он будет отображаться двумя двоичными разрядами, и это означает удвоение битовой скорости относительно бодовой.

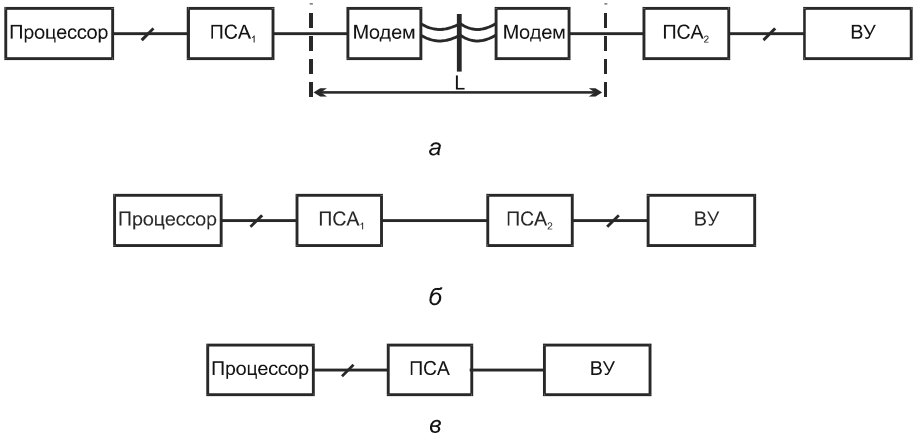


Рис. 7.9. Структура трактов передачи данных

Если часть тракта с модемами не нужна (рис. 7.9, *б*), то останется $ПСА_1$, преобразующий параллельные данные в последовательные, и $ПСА_2$, преобразующий последовательные данные в параллельные. Если требуется взаимодействие процессора с относительно недалеко расположенным ВУ, оперирующим с последовательными кодами, то тракт передачи будет иметь вид, показанный на рис. 7.9, *в*.

Передачи могут быть *симплексными*, *полудуплексными* или *дуплексными*. В первом случае данные передаются только в одну сторону, во втором — в обе, но с разделением во времени, в третьем — в обоих направлениях одновременно.

Характер передаваемой информации

При обмене последовательными данными передается *символьная информация* (буквы, цифры и другие знаки). Символы кодируются группой битов, число которых обычно лежит в пределах от 5 до 8. Если разрядность группы 5, то можно отображать до 32 различных символов, если 8, то 256.

Международное признание получил американский стандартный код обмена информацией *ASCII* (American Standard Code for Information Interchange), в котором символы кодируются 7 двоичными разрядами. Этот код позволяет передавать цифры, прописные и строчные буквы латинского алфавита и другие символы (всего 96 символов, т. к. 32 кодовые комбинации выделены для представления команд). На основе этого кода построен отечественный код КОИ-7 (код обмена информацией семиразрядный). Применяется также восьмиразрядный код *ДКОИ-8*.

Асинхронные и синхронные передачи

Важнейшее требование правильного приема — определение приемником моментов времени, в которые следует воспринять очередной бит данных. Иными словами, речь идет о синхронизации процессов в передатчике и приемнике. Можно связать передатчик с приемником специальной линией для синхронизации. Можно обойтись и без такой линии, применив *синхронизацию приемника самим передаваемым сигналом*, для чего сигналу придается *определенная структура*.

Протоколы последовательного обмена задают два его вида: *асинхронный и синхронный*. При асинхронном обмене символы передаются поодиночке по мере их готовности. Интервал между символами может быть различным, хотя интервалы между битами в одном символе фиксированы. При отсутствии готовых данных линия простаивает.

При синхронной передаче символы следуют один за другим слитно, поэтому можно говорить о передаче массива символов — текста. Если очередной символ не готов, передача не останавливается, передатчик посылает в линию специальные слова-синхросимволы, до тех пор пока не сможет передать следующий символ данных. Синхронный обмен повышает скорость передачи данных.

Скорость передачи оценивается числом передаваемых в секунду битов и соответствует типовому периферийному оборудованию. Принят ряд стандартных значений скорости, к которому принадлежат, в частности, 300, 1200, 2400, 4800, 9600, 14400, 19200, 33600, 56000 бит/с.

Структура кадра при последовательной асинхронной передаче

При асинхронных передачах *посылка (кадр)*, т. е. группа битов, отображающих символ, имеет следующий формат: до начала кадра линия находится под высоким потенциалом (состояние, называемое *маркой*) начало посылки отмечается нулевым старт-битом, за ним следуют 5...8 информационных (младшим разрядом вперед), затем идет необязательный бит контроля по модулю 2 (бит четности/нечетности) и заканчивается посылка 1; 1,5 или 2 единичными стоп-битами (рис. 7.10, *а*).

В отсутствие передачи линия находится под высоким потенциалом (активная пауза, марка). Появление низкого уровня означает поступление старт-бита, свидетельствующего о последующей передаче известного заранее числа информационных битов. Далее может идти контрольный бит четности (нечетности). Стоп-бит также используется для проверки правильности передачи, но уже по другому критерию. Длительность стоп-бита определяет минимальный промежуток между окончанием данного символа и началом следующего. Этот промежуток составляет 1...2 битовых интервала.

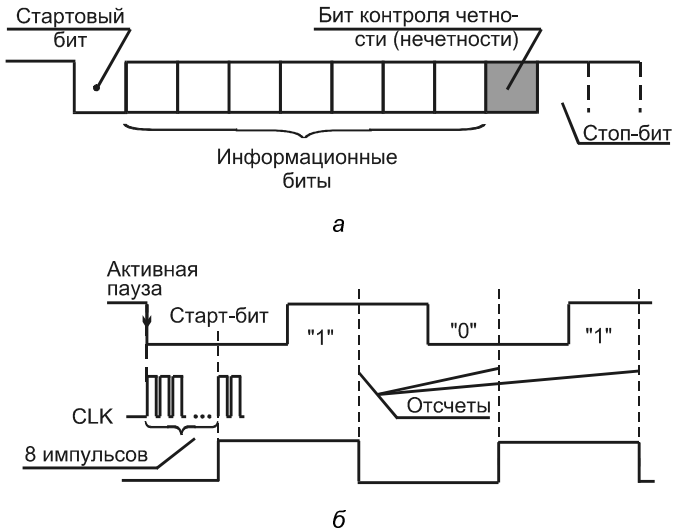


Рис. 7.10. Структура кадра для асинхронных передач (*а*) и временные диаграммы формирования временных меток в середине бита (*б*)

Работа приемника при асинхронных передачах

В этом режиме приемник синхронизируется самим сигналом. Приемник должен считывать значения битов в серединах их интервалов, где искажения импульсов меньше всего влияют на величину считываемого уровня. Это достигается следующим образом. *Передатчик и приемник имеют свои генераторы тактовых импульсов, работающие на одинаковой частоте*. При отсутствии передачи передатчик устанавливает в линии высокий уровень напряжения (марку). Появление нуля (старт-бита) отмечает начало передачи, которое, таким образом, фиксируется перепадом "1→0". От этого фронта начинает работать генератор приемника. Приемник выдерживает интервал в половину длительности бита, проверяет, есть ли еще нуль на входе (контролирует истинность старт-бита с целью исключить реакцию на кратковременную помеху), и затем начинает воспринимать данные с интервалом в длительность бита (если старт-бит не подтвердился, то приемник возвращается в исходное состояние).

Частоты генераторов передатчика и приемника реально отличаются, поэтому отсчеты постепенно "сползают" с середины битов и смещаются к тому или другому краю импульсов. Однако за время короткой посылки (не более 10...11 битов) смещение отсчетов с середины битов легко сделать пренебрежимо малым.

Проверка истинности старт-бита и выборка отсчетов в середине битовых интервалов производятся благодаря наличию в ПСА частоты, более высокой, чем частота следования битов (обычно в 16 раз). После пуска генератора высокой частоты CLK с помощью счетчика отсчитывается 8 импульсов, что и отмечает середину старт-бита. Затем отсчеты повторяются с интервалом τ, получаемым от деления частоты CLK на 16 (рис. 7.10, *б*).

Фиксируемые ошибки передачи

Контролируются три вида ошибок передачи. В конце кадра проверяется стоп-бит. Отсутствие на позиции стоп-бита высокого уровня напряжения свидетельствует об ошибке формата (кадра, обрамления) и устанавливает триггер-флажок *ошибки формата.*

Если это запрограммировано, то проверяется и четность веса посылки с учетом контрольного разряда. Для фиксации результата этой проверки также имеется специальный триггер *контроля четности*.

Основная часть приемника — два регистра, последовательный и параллельный. Принимаемый символ бит за битом вдвигается в последовательный (сдвигающий) регистр. Принятый символ передается в регистр хранения (параллельный) для последующей выдачи в виде параллельного кода. После этого приемник ищет следующий символ и вдвигает его в регистр сдвига. В регистр хранения второе слово не идет, пока не считано первое. В это время может пойти третий символ, который будет потерян, поскольку хранить его негде. Это *ошибка пропуска* (переполнения), которая тоже фиксируется установкой соответствующего триггера-флажка. Ошибка пропуска не возникает, если процессор считывает слово за интервал, меньший, чем интервал ввода символа в сдвигающий регистр.

Синхронные передачи

Различают *две разновидности синхронных передач* — с внутренней и внешней синхронизацией.

При *внутренней синхронизации* перед массивом данных передаются *слова-синхросимволы* (одно или два). При отсутствии передачи передатчик не перестает работать, а посылает в линию синхросимволы, пока не возобновится передача данных. Приемник при этом находится в режиме активного ожидания (в режиме Hunt — охоты). Он сравнивает каждое принятое слово с символом синхронизации. Если результат сравнения отрицательный, то обращения к данному приемнику нет (по описанному протоколу к одному передатчику можно подключить несколько приемников, имеющих индивидуальные синхросимволы). Если же опознается синхросимвол данного приемника, то это означает, что передатчик обращается к нему и первое же слово, не являющееся синхросимволом, принимается как начинающее информационный массив. После начала массива приемник считает передаваемые символы или же сопоставляет их с символами синхронизации, определяя одним из этих способов конец передачи. Во втором варианте после информационного массива идет синхросимвол, отмечающий конец передачи. Приемник далее переходит к режиму Hunt нового поиска синхросимвола. Символы данных не разделяются старт- и стоп-битами. Возможен контроль массива по модулю 2.

При *внешней синхронизации* в канал связи вводится дополнительная линия, по которой передается строб-сигнал, отмечающий интервал времени, соответствующий передаче данных. Фронты строба отмечают начало и конец передачи массива, в котором символы по-прежнему передаются слитно (без старт- и стоп-битов).

§ 7.6. Связные адаптеры

На рис. 7.11 показана структура *программируемого связного адаптера* ПСА (*PCI* — Programmable Communication Interface) Intel 8251A, аналогом которого являются отечественные микросхемы ВВ51А. Согласно реализуемым протоколам ПСА является *универсальным синхронно-асинхронным приемопередатчиком* УСАПП (USART — Universal Synchronous/Asynchronous Receiver/Transmitter). Адаптер, реализующий только асинхронные протоколы, называют *универсальным асинхронным приемопередатчиком* УАПП (*UAR* — Universal Asynchronous Receiver/Transmitter).

В МПС адаптер программируется процессором для работы с различной аппаратурой, принимает от процессора символы в параллельной форме и преобразует их в последовательную для передачи или получает последовательные данные и преобразует их в параллельные символы для процессора. Кроме того, адаптер сигнализирует процессору о готовности принять новый символ для передачи или о том, что получил символ для него и обеспечивает обмен сигналами квитирования с терминалом (модемом). В любое время процессор может читать слово состояния адаптера.

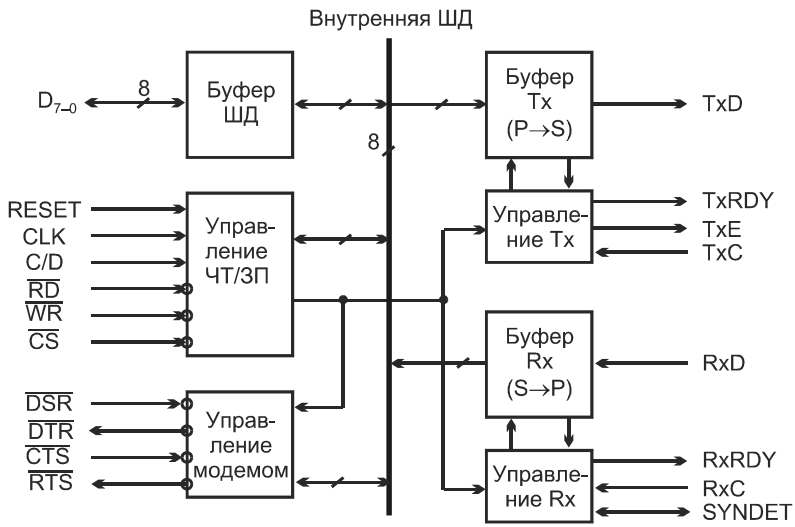


Рис. 7.11. Структура программируемого связного адаптера

Буфер ШД — двунаправленный, восьмиразрядный, с тремя состояниями. Он связывает адаптер с системной шиной данных (процессором) и принимает данные по командам процессора `OUT port`, выдает — по командам `IN port`. Через буфер передаются также управляющие и командные слова, а также слово состояния адаптера. В буфере имеются регистры данных (входной и выходной), команд и состояния.

Блок управления чтением/записью принимает сигналы от системной шины данных и генерирует сигналы управления работой всех блоков адаптера.

Выводы и сигналы ПСА имеют следующее назначение:

- RESET — установка адаптера в исходное состояние, после которой адаптер находится в бездействии до записи управляющих слов, определяющих задаваемые ему функции. В состояние бездействия адаптер вводится также программой по команде сброса;
- CLK — вход процессорной тактовой частоты. Частота передачи данных не привязана к частоте CLK, но должна быть ниже ее не менее чем в 30 раз;
- $\overline{RD}$, $\overline{WR}$ и $\overline{CS}$ — стробы чтения и записи и сигнал выбора микросхемы;
- C/D (Control/Data) — указывает на тип передаваемой информации, при единичном значении этого сигнала вводятся управляющие слова или выводится слово состояния адаптера, при нулевом — передаются данные. Вместе с сигналами $\overline{RD}$ и $\overline{WR}$ сигнал C/D определяет характер передачи. Обычно на этот вход подключается младший разряд адреса A_0.

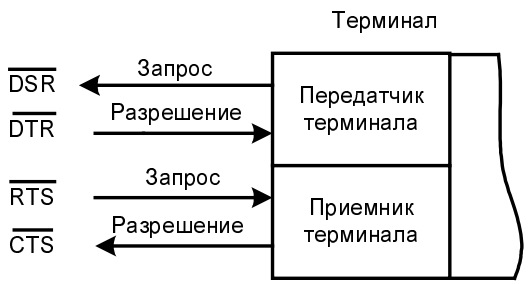


Рис. 7.12. Сигналы взаимодействия ПСА с терминалом

Устройство, с которым взаимодействует ПСА, называют *терминалом*. Типичный терминал — модем. Адаптер имеет две пары сигналов квитирования для управления терминалом (рис. 7.12):

- $\overline{DSR}$ (Data Set Ready) — входной сигнал готовности передатчика терминала. Низкий уровень этого сигнала говорит о том, что терминал имеет информацию для передачи. Сигнал может быть проверен процессором чтением слова состояния;
- $\overline{DTR}$ (Data Terminal Ready) — выходной сигнал, является реакцией на сигнал $\overline{DSR}$. Разрешает терминалу посылку данных на вход приемника адаптера. Активизируется соответствующим битом командного слова, если процессором разрешен обмен с терминалом;
- $\overline{RTS}$ (Request to Send) — запрос от адаптера готовности приемника терминала принять данные. Задается программированием соответствующего бита в командном слове, когда процессором разрешен обмен с терминалом;
- $\overline{CTS}$ (Clear to Send) — сигнал готовности приемника терминала принять данные. Низкий уровень этого сигнала разрешает адаптеру передачу последова-

тельных данных, если установлен бит TxEN в командном слове. При снятии бита TxEN или сигнала $\overline{CTS}$ во время работы передатчика он будет передавать все данные, записанные до запрещения передачи, прежде чем остановится.

Передатчик ПСА

Буфер передатчика (буфер Tx) (см. рис. 7.11) принимает от буфера ШД параллельные данные, преобразует их в поток последовательных битов, вводит в этот поток служебные символы или биты и выдает составленный поток на вывод TxD (по отрицательным фронтам импульсов TxC). Передача начинается после ее разрешения и при условии $\overline{CTS} = 0$. Вывод TxD принимает высокий уровень после сброса, запретов по условиям TxEN или $\overline{CTS}$ либо при условии "передатчик пуст".

Схема управления передатчиком (управление Tx) вырабатывает следующие внутренние и внешние сигналы:

- TxRDY — выходной сигнал, указывающий процессору на готовность передатчика адаптера принять символ. Сигнал может проверяться чтением слова состояния или служить запросом прерывания (он может маскироваться битом TxEN командного слова). Автоматически сбрасывается передним фронтом строба записи $\overline{WR}$, когда символ загружается из процессора;
- TxE (Tx Empty) — сигнал устанавливается, когда адаптер не имеет символа для передачи и после освобождения сдвигающего регистра передатчика этот регистр будет нечем загрузить. Сбрасывается после получения символа от процессора, если передача разрешена, и сохраняется, если передача запрещена битом командного слова. В полудуплексном режиме работы может быть использован для индикации конца передачи и оповещения процессора о моменте переключения линии на другое направление. В синхронном режиме показывает, что данные не были загружены и в выходной поток данных следует вводить синхросимволы. Пока передаются синхросимволы, высокий уровень сигнала сохраняется;
- TxC — сигнал синхронизации передатчика, задающий скорость следования последовательных битов. Скорость синхронных передач равна частоте сигнала TxC, при асинхронных передачах она составляет 1 или 1/16 или 1/64 от TxC. Синхронности TxC с сигналом CLK не требуется.

Схемная реализация передатчика. Без учета контроля по четности/нечетности и для структуры кадра с восемью информационными битами иллюстрируется рис. 7.13.

> ***Примечание***
>
> Далее (под названиями "Схемная реализация передатчика" и "Схемная реализация приемника") даются детализированные и, соответственно, более сложные для восприятия описания схем передатчика и приемника, изучение которых можно считать не обязательным при первом знакомстве с ПСА. В то же время ознакомление с этими разделами полезно с точки зрения иллюстрации типичных приемов, применяемых при разработке цифровых схем.

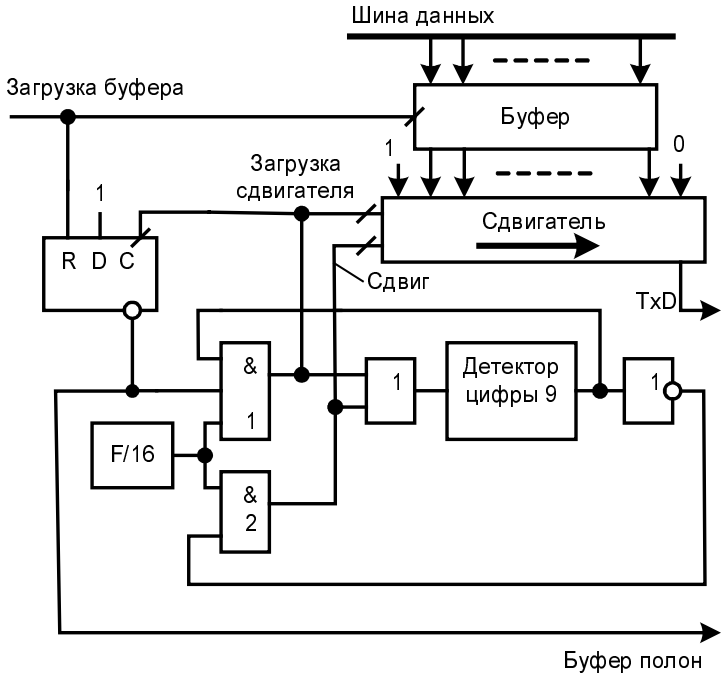


Рис. 7.13. Схемная реализация передатчика ПСА

Обозначение F/16 относится к блоку деления частоты на 16. Для работы передатчика и приемника, как видно из изложенного ранее, требуются не только частоты передачи битов TxC и RxC, но и более высокая частота. Такая частота F вырабатывается генератором (на рисунке не показан), ее деление на 16 дает частоту передачи битов.

Параллельный регистр, который далее будем называть буфером, принимает с шины данных информационные разряды кадра по внешнему стробу загрузки "Загрузка буфера". К информационным разрядам добавляются нулевой старт-бит в начале и единичный стоп-бит в конце кадра. Блок "Детектор цифры 9" имеет нулевое значение выхода при всех состояниях, кроме 9 (состояния с кодом 1001 в счетчике этого блока). Пока счет в детекторе не дошел до 9, верхний вход конъюнктора 2 получает сигнал логической единицы и импульсы сдвига проходят на "Сдвигатель", а также и на вход блока "Детектор цифры 9". Появление 9 в счетчике детектора блокирует конъюнктор 2 и останавливает как сдвиги, так и подачу импульсов на вход детектора. Если триггер D установлен, то его инверсный выход блокирует конъюнктор 1, так что вход загрузки последовательного регистра, который будем называть сдвигателем, также пассивен. Передатчик, закончив предыдущий кадр, бездействует.

Внешний сигнал загрузки буфера сбрасывает триггер, блокировка конъюнктора 1 снимается и через него на сдвигатель проходит импульс загрузки, поступающий и на детектор, в котором состояние 9 изменяется на нулевое. Это блокирует конъ-

юнктор 1 и открывает конъюнктор 2 для прохождения на сдвигатель пачки из 10 импульсов (до появления в детекторе нового состояния 9). Десять импульсов сдвига выводят из сдвигателя десять битов — 8 информационных и два обрамляющих. Импульс загрузки сдвигателя одновременно устанавливает триггер, что блокирует сигнал загрузки сдвигателя до появления в буфере нового информационного слова.

Приемник ПСА

Буфер приемника (буфер Rx) (см. рис. 7.11) принимает последовательные данные, преобразует их в параллельные, проверяет биты или символы, специфичные для посылок данного типа, и посылает принятый символ в процессор. Вывод RxD служит входом последовательных данных.

Блок управления приемником Rx обеспечивает управление всеми действиями, связанными с приемом информации. Схемы этого блока предотвращают восприятие неиспользуемой линии данных как L-активной в режиме паузы. Для начала приема требуется появление высокого уровня (марки) на входе RxD после сброса системы. Если это выполняется, то разрешается поиск старт-бита и проверка его истинности. Ошибки работы адаптера (*четности, формата* или *переполнения*, если новая информация замещает старую раньше, чем она была использована) устанавливают соответствующие биты в слове состояния, но не останавливают работу адаптера. Выводы и сигналы блока управления приемником Rx имеют следующее назначение:

- RxRDY — выходной сигнал, показывающий, что адаптер имеет символ, готовый к выводу в процессор. Может проверяться чтением слова состояния или служить запросом прерывания для процессора. Если команда разрешения приема RxEN отсутствует, то сигнал RxRDY находится в состоянии сброса. Отсутствие чтения принятого символа из выходного регистра адаптера до появления следующего ведет к загрузке нового символа и потере старого. Устанавливается ошибка переполнения;
- RxC — сигнал синхронизации приемника, задающий скорость следования последовательных битов. При синхронных передачах скорость равна частоте RxC, при асинхронных она является частью частоты RxC (это 1 или 1/16 или 1/64 от RxC). Синхронности RxC с сигналом CLK не требуется. Частоты TxC и RxC обычно идентичны;
- SYNDET (SYNC Detect/Break Detect) — в синхронном режиме может быть входом или выходом. При внутренней синхронизации это выход (признак выявления синхросимвола при приеме). Если запрограммированы два синхросимвола, сигнал SYNDET установится в середине последнего бита второго синхросимвола. Сигнал автоматически сбрасывается после операции чтения состояния. При внешней синхронизации это вход, его появление заставляет адаптер начать прием данных. В асинхронном режиме это сигнал Break Detect, который устанавливается при низком уровне на интервалах стоп-битов в двух последовательных кадрах. Сигнал может быть выявлен чтением слова состоя-

ния. Сбрасывается при сбросе адаптера или возвращении входного сигнала к нормальному состоянию (появлению единиц на интервалах стоп-битов).

Схемная реализация приемника. Схемная реализация приемника, работающего с описанным ранее передатчиком, показана на рис. 7.14. Последовательные данные поступают в сдвигатель, причем импульсы сдвига должны вырабатываться в середине битовых интервалов (в некоторых приемниках в этом интервале берутся три отсчета и результат определяется мажоритарным голосованием). Для этого битовые интервалы заполняются импульсами частоты 16ТхС, а сдвиговые импульсы генерируются в середине пачки из 16 импульсов. Во время паузы счетчик по модулю 16 (CTR mod 16) сброшен по входу R сочетанием единичных сигналов входной линии (состояние марки) и выхода старшего разряда счетчика по модулю 9 (CTR mod 9) на входе конъюнктора 1. Появление нулевого старт-бита снимает сигнал R и счетчик начинает работать, формируя на седьмом импульсе (при состоянии 0111, т. е. вблизи середины битового интервала) сигнал сдвига, если низкий уровень старт-бита сохранился до этого времени. Так проверяется истинность старт-бита и предотвращается реакция на случайную помеху. Далее через каждые 16 импульсов, т. е. через битовый интервал, генерируется очередной импульс сдвига.

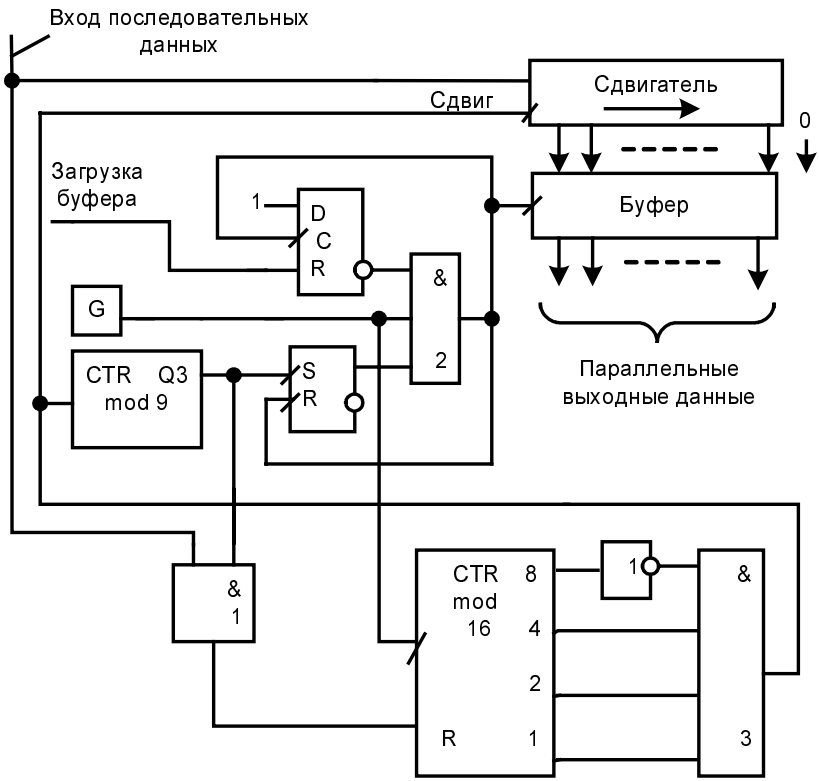


Рис. 7.14. Схемная реализация приемника ПСА

До начала приема очередного кадра счетчик по модулю 9 (CTR mod 9) находится в состоянии 8 (содержит максимальное число). Первый же импульс сдвига переведет его в нулевое состояние, в конце кадра он опять окажется в состоянии 8, на выходе Q3 старшего разряда появится логическая единица, и счетчик по модулю 16 (CTR mod 16) остановится единичными значениями сигнала Q3 и стоп-бита. Импульсы сдвига прекратятся, прием кадра закончен. В сдвигатель будут введены 9 битов, старт-бит пройдет через весь сдвигатель и будет потерян на его правом конце.

Содержимое сдвигателя загружается в буфер, если он свободен, а сдвигатель полон. Соответствующие признаки хранятся в двух триггерах. Для сдвигателя это триггер RS специального типа, имеющий динамические входы. Единичное состояние триггера устанавливается старшим разрядом счетчика по модулю 9, когда в сдвигателе сформировалось новое слово. Сброс происходит при передаче содержимого сдвигателя в буфер. Триггер признака "Буфер полон" — обычный D-триггер, который сбрасывается внешним стробом считывания содержимого буфера и устанавливается, когда в буфер загружается содержимое сдвигателя.

Программирование ПСА

Адаптер программируется загрузкой в него по командам `OUT port` *начального и текущего управляющих слов*. Начальное управляющее слово (инструкция режима MI — Mode Instruction) вводится после сброса и задает режим работы адаптера (синхронный, асинхронный), формат передаваемых символов, скорость передачи/приема, характеристику контроля, тип синхронизации. При синхронном обмене и внутренней синхронизации после инструкции режима MI в адаптер вводятся один или два синхросимвола, для хранения которых в схеме управления приемником имеются два специальных регистра.

После синхросимволов или непосредственно после MI, если задан режим асинхронного обмена или синхронного обмена с внешней синхронизацией, в адаптер загружается командное слово CI (Command Instruction), называемое также текущим управляющим словом. Новое командное слово может быть загружено в адаптер в любое время, что позволяет оперативно влиять на процесс обмена. Правильная загрузка нескольких регистров без индивидуальных адресов у них обеспечивается жестким порядком записи управляющих слов в программируемый адаптер. В адаптере формируется слово состояния.

Адаптер может работать в одном режиме или комбинации совместимых режимов, осуществляя программный условный обмен процессора с ВУ или обмен по прерываниям. Первый вид обмена предусматривает программное чтение слова состояния адаптера и при его готовности выполнение подпрограммы обмена. При обмене по прерываниям сигналы готовности адаптера TxRDY и RxRDY используются как запросы прерывания для процессора. Появление ошибок не останавливает работу адаптера. Ошибки выявляются установкой триггеров-флажков.

Рассмотрим для примера условия передачи в асинхронном старт-стопном режиме. В этом режиме после записи в адаптер параллельных данных они автоматически обрамляются старт- и стоп-битами, а при соответствующем программировании и битом контроля. Если командным словом CI дано разрешение режима передач

($D_0 = 1$) и от терминала получено условие готовности $\overline{CTS} = 0$, то на выход TxD начнет поступать поток битов с частотой, равной TxC или 1/16, или 1/64 этой частоты в зависимости от программирования адаптера. При отсутствии передачи на выходе TxD действует высокий уровень напряжения (марка). Если командным словом CI задана пауза, то уровень TxD становится низким.

При программном условном обмене (рис. 7.15) процессор осуществляет регулярный опрос состояния адаптера чтением слова состояния SW (Status Word). При готовности адаптера (TxRDY = 1, т. е. входной буфер пуст) выдается строб записи $\overline{WR}$, который передним фронтом снимает сигнал готовности (буфер уже занят), а задним, когда символ уже получен адаптером, начинает процесс передачи кадра (выталкивания символа из регистра сдвига). Начало выдачи кадра говорит о том, что буфер шины данных освободился (символ уже в регистре передатчика) и нужно вернуть сигнал TxRDY в состояние 1. Вторая запись снимает готовность буфера, и его неготовность продлится до конца передачи первого кадра, за которой произойдет перегрузка символа из входного буфера адаптера в регистр передатчика, освобождение входного буфера и восстановление единичного уровня сигнала TxRDY. После чтения SW на интервале неготовности строб записи не вырабатывается. После появления готовности повторятся уже описанные действия.

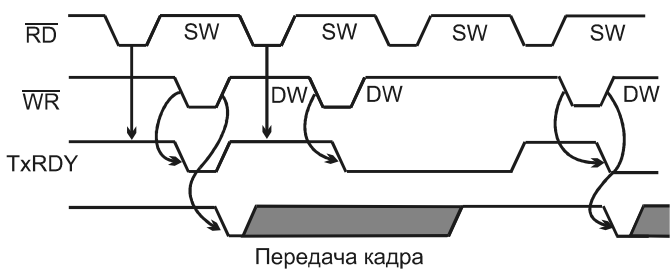


Рис. 7.15. Временные диаграммы программного условного обмена с помощью программируемого связного адаптера

Пример подключения ПСА к МП и терминалу

Шина данных может подключаться к выводам адаптера через буфер или непосредственно в зависимости от нагрузочных условий. Селектор адреса CA (рис. 7.16) выдает на выходе низкий логический уровень, разрешающий работу адаптера, в ответ на одну-единственную комбинацию входных сигналов A_{7-1}. На вход CLK поданы синхроимпульсы от МП, а частоты передачи и приема (в данном случае равные) получены из этой частоты с помощью делителя частоты ДЧ. Как требуется условиями работоспособности адаптера, коэффициент деления должен быть ≥ 30. Делитель частоты имеет 4 выхода с разными частотами. С помощью ключа K можно изменять скорость передачи-приема данных. Остальные соединения понятны без дополнительных пояснений.

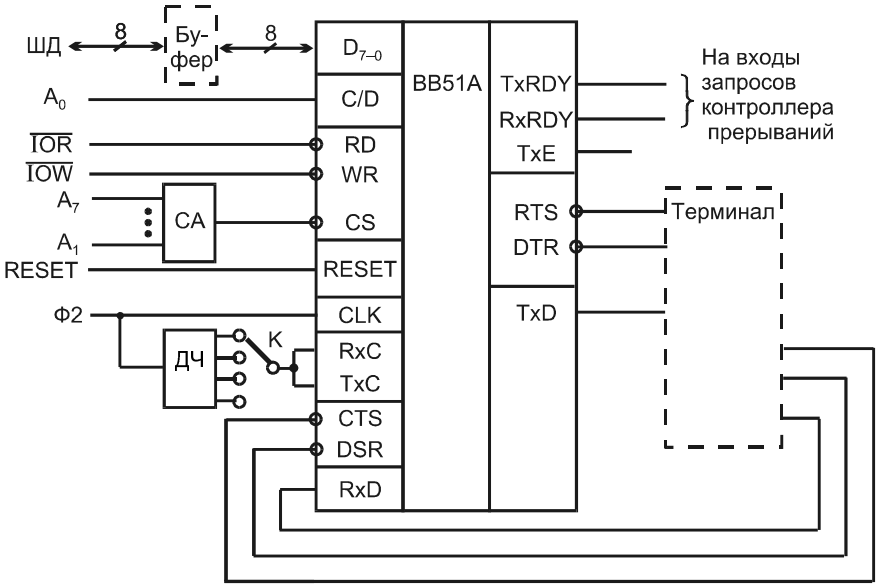


Рис. 7.16. Схема подключения программируемого связного адаптера к микропроцессору и терминалу

Последовательные порты часто строятся на основе адаптеров UART (типа 16550 и 16550A фирмы Texas Instruments и др.), которые во многом подобны адаптеру BB51A, но имеют 16-символьные буферы FIFO для приема и передачи данных.

Модули UART. Как следует из названия, модули UART работают в асинхронных режимах, но на самом деле многие из них способны реализовать и простые режимы синхронных передач. В режиме асинхронных передач модули UART поддерживают протоколы обмена RS-232c, RS-422, RS-485. Модули аналогичного типа фирмы Motorola называются *SCI* (Serial Communication Interface). Типичные UART содержат каналы передатчика и приемника, принципы работы которых не отличаются от рассмотренных ранее.

§ 7.7. Интерфейсы SPI и I²C

В устройствах и системах, не требующих таких больших функциональных возможностей, которыми обладает USART типа 51A или подобные ему, используют более простые средства последовательных интерфейсов, самыми распространенными из которых являются интерфейсы SPI и I^2C.

Интерфейс SPI

SPI (Serial Peripheral Interface) — синхронный дуплексный трехпроводной интерфейс обмена последовательными данными, отличающийся структурной и схемо-

технической простотой. В этом интерфейсе отсутствуют пересылки адресной информации и, следовательно, время передачи эффективно используется для обмена полезными данными. С помощью интерфейса SPI удобно подключать к микроконтроллерам датчики, исполнительные устройства и т. п. Он может быть использован и для связи между процессорами в многопроцессорных системах. По протяженности связей интерфейс можно отнести к внутриплатным. Из подключенных к линиям интерфейса модулей SPI выделяется ведущий, остальные являются ведомыми.

В тракте передач обязательны три линии: две сигнальные (MOSI — Master Output – Slave Input и MISO — Master Input – Slave Output) и одна для синхронизации (SCK). Четвертая линия (L-активная линия $\overline{SS}$, Slave Select) используется не всегда, что и объясняет термин "трехпроводной интерфейс". В схеме с одним ведущим и одним ведомым модулями при неизменности их предназначений линия $\overline{SS}$ не требуется, ее сигналы заменяются подачей на вывод $\overline{SS}$ ведущего модуля высокого уровня напряжения, а на вывод $\overline{SS}$ ведомого — низкого. Вообще же линия $\overline{SS}$ служит для выбора активного ведомого модуля, с которым взаимодействует ведущий. В схемах с несколькими ведомыми только модуль с низким уровнем сигнала на входе $\overline{SS}$ может реагировать на сигналы интерфейса и генерировать данные на линии MISO. При высоком уровне сигнала $\overline{SS}$ выход MISO переходит в третье состояние.

Процесс обмена в схеме с одним ведущим и одним ведомым модулями SPI иллюстрируется на рис. 7.17, *а*. На входы $\overline{SS}$ ведущего и ведомого модулей поданы высокий и низкий уровни напряжения соответственно, что и определяет их функции. В более общем случае ведущая роль модуля задается соответствующим битом регистра управления, а выводы $\overline{SS}$ ведомых модулей получают сигналы активизации от линии $\overline{SS}$ ведущего модуля. Структура порта SPI приведена на рис. 7.17, *б*.

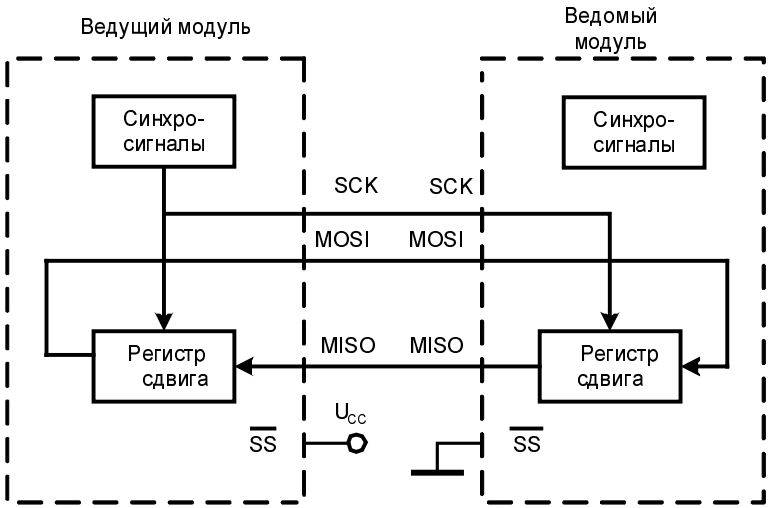
Регистр управления загружается по шине данных управляющим словом, имеющим следующие разряды:

- разрешения/запрещения прерываний от SPI;
- разрешения/запрещения работы SPI;
- порядка передачи данных (передач старшими или младшими разрядами вперед);
- выбора режима (ведущий/ведомый);
- полярности тактирующего сигнала;
- фазы тактирующего сигнала;
- два разряда выбора частоты тактирования (подаются на адресные входы мультиплексора).

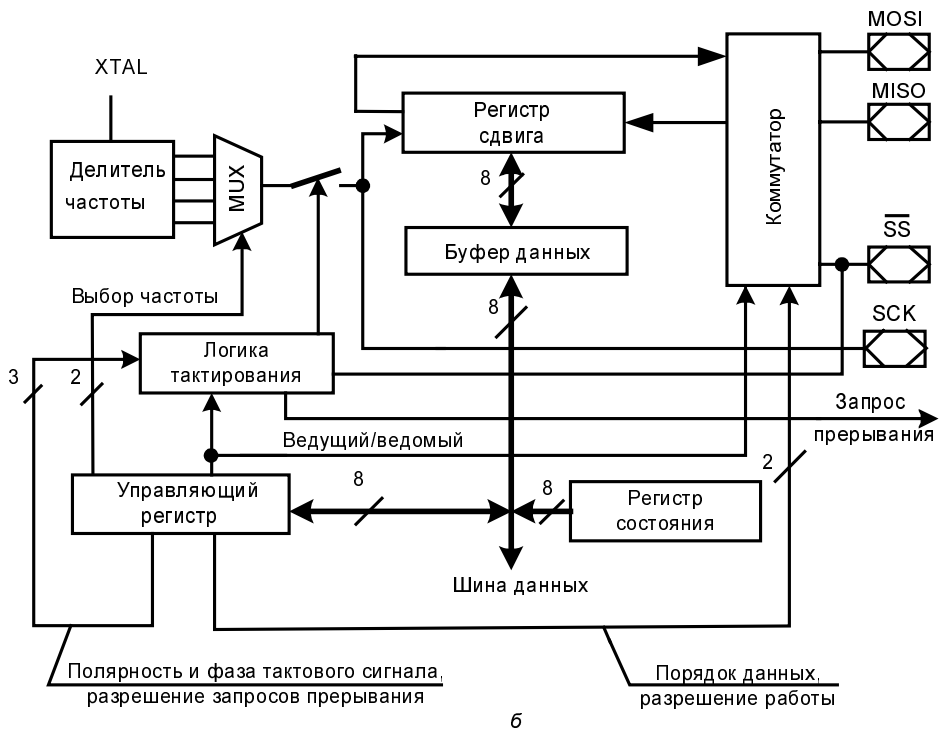
В *регистре состояния* используются два разряда (остальные разряды зарезервированы):

- конца передачи, который устанавливается по завершении транзакции, и если соответствующим битом слова управления прерывания разрешены, то формируется запрос прерывания;

- флага ошибки записи в регистр в ходе последовательных передач, сбрасываемый при чтении регистра состояния.



а



б

Рис. 7.17. Схема обмена данными между модулями SPI (*а*) и порт SPI (*б*)

К линиям передачи данных регистр сдвига подключен через коммутатор, сигналы управления которым показаны на рисунке. Частота синхросигналов, задаваемая линией XTAL, поступает на делитель частоты с четырьмя выходами, которым соответствуют коэффициенты деления 4, 16, 64 и 128. Выбор частоты определяется значениями адресных входов мультиплексора, заданных в слове управления. Возможность выбора частоты передачи позволяет, в частности, приспосабливать скорость передачи к длине связи, поскольку с удлинением линий связи частотные возможности канала падают. Тактирование осуществляется ведущим модулем, выдающим синхросигналы на линию SCK. У ведомых модулей средства тактирования отключаются.

Ведущий модуль начинает цикл передачи, устанавливая низкий уровень напряжения на выводе $\overline{SS}$ выбранного для взаимодействия ведомого. Передачу инициирует запись байта в регистр данных. Ведущий и ведомый модули готовят данные, загружают их в свои регистры сдвига и ведущий генерирует тактирующие импульсы в линии SCK, чтобы осуществить обмен приготовленными байтами. Сдвигающие регистры обменивающихся модулей соединены в кольцо, так что после поступления восьми сдвигающих импульсов байт передатчика переходит в приемник и одновременно байт приемника переходит в передатчик. Передача байта останавливает генератор и устанавливает флаг конца передачи. Если прерывания от SPI разрешены, то генерируется запрос прерывания. До завершения цикла сдвигов новая загрузка регистра данных невозможна. Принятый байт должен считываться из регистра данных до завершения приема следующего байта, иначе он будет потерян.

В интерфейсах SPI с несколькими ведомыми модулями могут быть организованы разные системы обмена (стандартная последовательная, стандартная параллельная, со сравнением адресов). На рис. 7.18 приведен пример стандартной параллельной системы, в которой для данных и синхросигнала применена магистральная структура, а для сигналов $\overline{SS}$ — отдельные линии для каждого из ведомых модулей. В такой системе ведущий модуль должен раздельно формировать сигналы $\overline{SS}$ для каждого из ведомых.

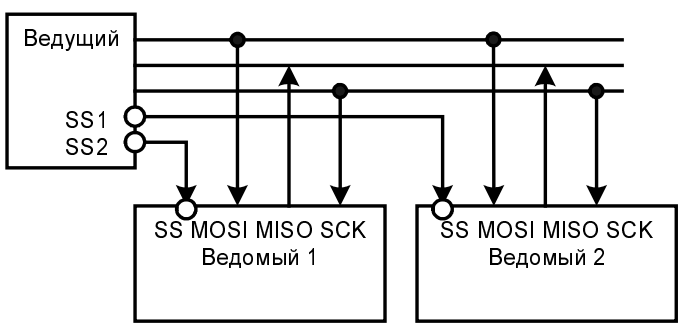


Рис. 7.18. Стандартная параллельная система для SPI

Традиционно интерфейсы SPI применялись для частот передачи до 4…5 МГц, т. е. с генераторами частот до 16…20 МГц. Позднее появились более быстродействующие варианты. Например, наиболее быстродействующая последовательная память фирмы Atmel (2006 г.) имеет интерфейс *Rapid S*, представляющий собою расширение интерфейса SPI с частотой передачи более 50 МГц.

Интерфейс I^2C

Двухпроводной интерфейс I^2C (фирма Philips) разработан для синхронной последовательной связи между микросхемами, расположенными на одной печатной плате. Его достоинство — максимальная простота. Обмен ведется с использованием всего двух линий: SDA (Serial Data Line) — линии передачи данных и SCL (Serial Clock Line) — линии, на которую ведущий модуль выдает синхросигналы, стробирующие передаваемые данные. Поскольку для передачи информации имеется всего одна линия, адреса передаются в составе посылки, т. е. сокращение числа линий окупается за счет снижения доли времени, отдаваемого на передачу полезной информации, иначе говоря, скорости обмена. Обе линии являются двунаправленными.

Выходные цепи источников сигналов построены по схеме "с открытым коллектором", так что их объединение на линиях интерфейса образует элемент И монтажной логики (см. *§ 1.2*). Вследствие этого среди объединяемых сигналов нуль оказывается доминирующим — если какой-либо источник формирует нулевой сигнал, то единичные сигналы других источников не могут этому помешать. Использование монтажной логики И позволяет эффективно разрешать конфликты на шинах и решать задачи арбитража, когда два или более ведущих устройств пытаются одновременно управлять шинами. Организация канала по интерфейсу I^2C показана на рис. 7.19.

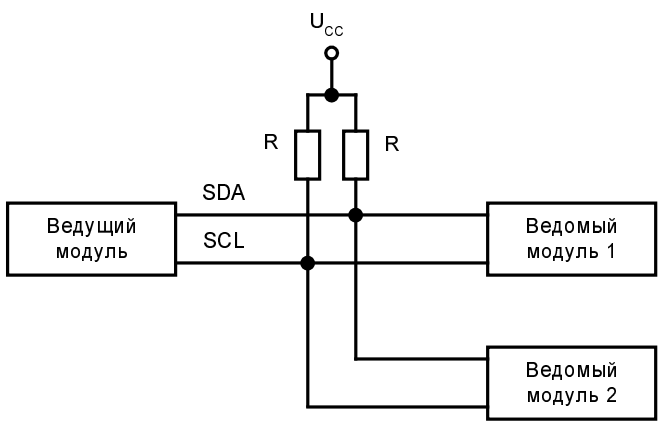


Рис. 7.19. Организация последовательного канала с интерфейсом I^2C

Для начала передачи ведущее устройство, если линия свободна, формирует на линии SDA перепад 1→0, что при единичном состоянии линии SCL идентифицируется как стартовый бит посылки. Формирование стартового бита отмечает занятость

линии. После стартового бита ведущий модуль посылает адресующе-управляющий байт (возможен и десятиразрядный формат первоначального слова), содержащий адрес ведомого модуля (семь разрядов), и разряд R/W, задающий направление передачи (чтение или запись). Все ведомые модули сравнивают посланный ведущим модулем адрес со своими адресами, и модуль, нашедший свой адрес, посылает в линию SDA сигнал подтверждения — нулевой уровень напряжения во время девятого импульса синхронизации. Подтверждение является условием выполнения операции и посылается ведомым модулем в конце каждого пересылаемого байта в течение всего времени обмена. Отсутствие сигнала подтверждения заставляет ведущий модуль сформировать сигнал STOP в виде положительного перепада напряжения на линии SDA при единичном состоянии линии SCL. Такой же сигнал формируется в конце обмена.

В интерфейсе легко организуется асинхронный обмен с учетом готовности ведомого устройства. Если оно не готово к началу обмена, то фиксирует уровень 0 на линии SCL, что реализуется через монтажную логику И. Такая же ситуация может применяться и при приеме каждого бита данных.

Арбитраж также использует монтажную логику. Ведущее устройство, сформировавшее сигнал начала передачи, захватывает управление интерфейсом и выполняет требуемые операции обмена. Если же несколько ведущих устройств одновременно пытаются захватить управление шинами, то происходит следующее. Претенденты совместно формируют стартовый бит и затем передают адреса, начиная со старшего бита. Пока значения разрядов адреса совпадают, претенденты работают согласованно. Как только дело дойдет до различающихся разрядов (0 и 1), проявятся доминирующие свойства нуля и устройство, передающее 1, зафиксирует отсутствие ее передачи (передается 1, а наблюдается 0). Это означает проигрыш арбитража, и проигравшее устройство немедленно прекращает передачу. Как видно, приоритетными являются передачи по численно меньшим адресам. Если адреса, передаваемые претендентами, совпадают, арбитраж продолжается и далее, распространяясь на байты данных, следующие за адресующе-управляющим байтом.

Скорости передачи в интерфейсе I^2C невысоки — 100 кГц в стандартном варианте и 400 кГц в повышенном. В связи с малым числом линий и невысокими требованиями к параметрам буферных каскадов интерфейс отличается дешевизной, компактностью, малой потребляемой мощностью и низким уровнем создаваемых помех. Такой комплекс свойств определяет для интерфейса и ряд областей применения — связь с датчиками, схемы программирования аналоговых устройств, схемы тестировании и инициализации БИС и др.

§ 7.8. Схемы обслуживания прерываний

При работе МПС отклик на события, требующие немедленной реакции, обеспечивается прерыванием выполняемых программ и переходом к обслуживанию запросов прерывания.

Программный опрос

Обработка запросов прерывания может выполняться *программным способом* без применения дополнительных аппаратных средств. В этом случае процессор осуществляет последовательный опрос — *поллинг* (polling) источников запросов, начиная с источника, обладающего высшим приоритетом, т. е. последовательно перебирает адреса источников запросов. Место источника в последовательности проверок определяет его приоритет. Обнаружив по какому-либо адресу сигнал запроса, процессор выполняет подпрограмму его обслуживания. Подобная стратегия сопряжена с большими потерями времени. Даже при отсутствии запросов, нужно периодически перебирать все адреса источников, выполняя по ним циклы чтения. Объединив сигналы запросов по ИЛИ, можно получить признак наличия хотя бы одного запроса, инициируя тем самым процесс последовательного опроса источников. Но и в этом случае прерывания реализуются с поочередной проверкой источников на наличие запроса. При обнаружении источника, выставившего запрос, процессор читает его вектор прерывания, после чего выполняется подпрограмма обслуживания.

Аппаратный опрос источников прерываний

При аппаратном опросе последовательный перебор адресов заменяется процессом распространения сигнала опроса по так называемой *дейзи-цепочке* (рис. 7.20).

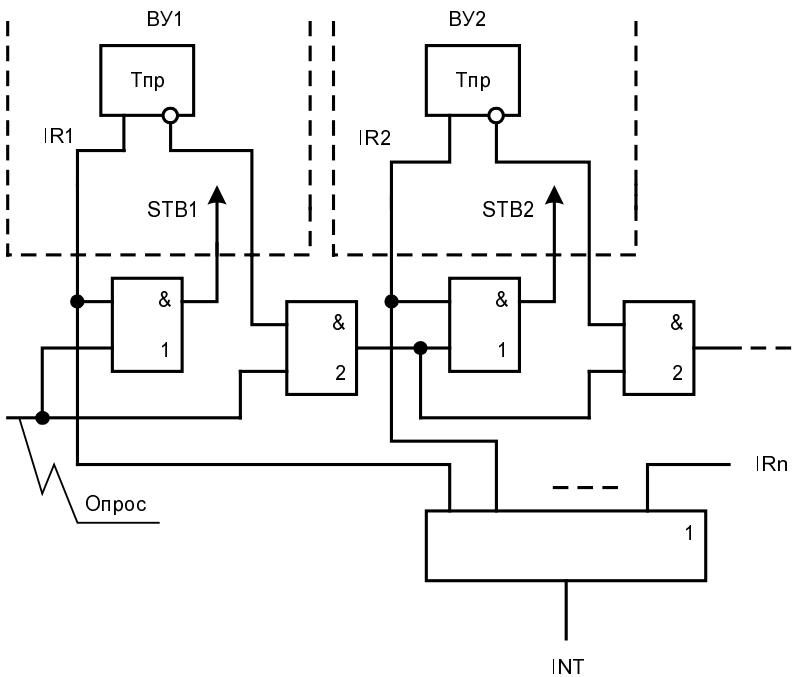


Рис. 7.20. Схема аппаратного опроса (поллинга) источников прерываний

В дейзи-цепочке каждому ВУ (источнику запроса прерывания) соответствуют два вентиля. Наличие или отсутствие запроса отображается состоянием триггера прерывания $T_{ПР}$ данного ВУ. Все линии запросов IR1…IRn объединяются по ИЛИ, так что наличие хотя бы одного запроса порождает для процессора сигнал INT. Если процессор подтверждает запрос, то он выдает на линию опроса сигнал, который начинает свое распространение со старшего ВУ1. Если, например, это ВУ имеет запрос, то на линии строба STB1 появится сигнал выдачи процессору вектора прерывания от данного ВУ, а дальнейшее распространение сигнала опроса по цепочке будет заблокировано вентилем 2, на который поступает инверсное значение запроса, т. е. нулевой сигнал. Если же запроса у старшего ВУ нет, сигнал опроса пройдет на следующее ВУ и т. д., анализируя состояния ВУ. Первое же ВУ, имеющее запрос, получит сигнал строба STB и прекратит распространение сигнала опроса к следующим младшим устройствам.

Контроллеры прерываний

Аппаратные прерывания обслуживаются как простыми специализированными ИС (блоки приоритетного прерывания Intel 8214 и др.), так и программируемыми контроллерами прерываний ПКП (PIC, Programmable Interrupt Controller), обеспечивающими эффективную обработку запросов прерываний с широким набором режимов.

Базовая структура ПКП Intel 8259A (отечественный аналог ВН59А) обрабатывает до 8 запросов. С помощью нескольких ПКП легко организуется обработка до 64 запросов. Задача ПКП — выдать процессору команду перехода к подпрограмме обслуживания признанного запроса. При этом могут выполняться как различные виды прерываний, так и режим опроса.

Вложенные прерывания с фиксированными приоритетами. Из имеющихся у ПКП 8 входов запросов прерывания IR_7...IR_0 (IR от Interrupt Request) высший приоритет имеет вход IR_0, низший — IR_7. *Вложенность* — возможность прерывания подпрограммы обслуживания запроса другой подпрограммой с более высоким приоритетом, которая, в свою очередь, также может быть прервана более приоритетной подпрограммой и т. д. Вложенность прерываний обеспечивается введением команды `EI` (Enable Interrupt) в подпрограммы их обслуживания. Прерывания с фиксированными приоритетами просты в реализации, но *запросы неравноправны* и при интенсивном поступлении запросов с высокими приоритетами запросы с низкими приоритетами могут вообще не получить обслуживания, т. е. возможно их "грубое оттеснение" более приоритетными.

Прерывания с циклическими приоритетами. В этом случае у каждого входа тоже есть свой приоритет, но после обслуживания он изменяется так, что обслуженный вход получает низший приоритет, который по мере обслуживания других запросов начинает возрастать, вновь доходя до высшего. Такая дисциплина обслуживания характерна для ситуации с равноправными источниками, *не имеющими преимуществ друг перед другом*. Устройство, запрашивающее обслужива-

ния, будет ждать в худшем случае до того, как 7 других источников будут обслужены по одному разу. Работу с круговым приоритетом можно иллюстрировать примером (рис. 7.21) с регистром запросов, содержащим вначале 6-й и 4-й запросы, где наивысший приоритет имеет 4-й запрос, который и будет обслужен. Строка "уровни приоритетов" рассматривается как *кольцо*, в котором позиции 7 и 0 являются соседними. После обслуживания приоритетность входов изменяется вращением кольца, причем номер 7 с низшим приоритетом становится на 4-ю позицию только что обслуженного запроса. Позицию низшего приоритета называют *дном приоритетного кольца*. Работу с круговым (циклическим) приоритетом можно выразить так: *после обслуживания дно приоритетного кольца устанавливается на позицию обслуженного запроса.*

Кроме рассмотренного, имеется также режим адресуемого циклического приоритета, при котором дно кольца после обслуживания ставится в положение, определяемое программно, а не устанавливается автоматически на позицию обслуженного запроса.

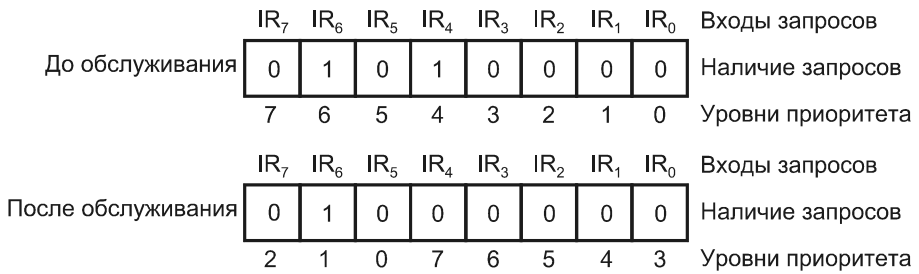


Рис. 7.21. Пример обслуживания запросов прерывания с круговым приоритетом

Маскирование запросов. Маскирование запросов запрещает их восприятие с помощью соответствующих битов регистра маски. При этом возможны разные ситуации. При *обычном маскировании* запрещение какого-либо запроса ведет и к маскированию других запросов с меньшими приоритетами. *Специальное маскирование* заключается в блокировании восприятия только одного запроса при отсутствии маскирования младших по приоритету. После снятия маски обслуживание запросов становится возможным.

Включение контроллера прерываний в систему. Включение контроллера прерываний в систему показано на рис. 7.22. Контроллер принимает запросы от внешних устройств, определяет, какой из незамаскированных запросов имеет наивысший приоритет, сравнивает его с приоритетом текущей программы и при соответствующих условиях выдает запрос прерывания INT для МП. После подтверждения запроса МП должен получить от контроллера информацию, которая укажет на подпрограмму, соответствующую данному ВУ, иными словами МП должен получить вектор прерывания.

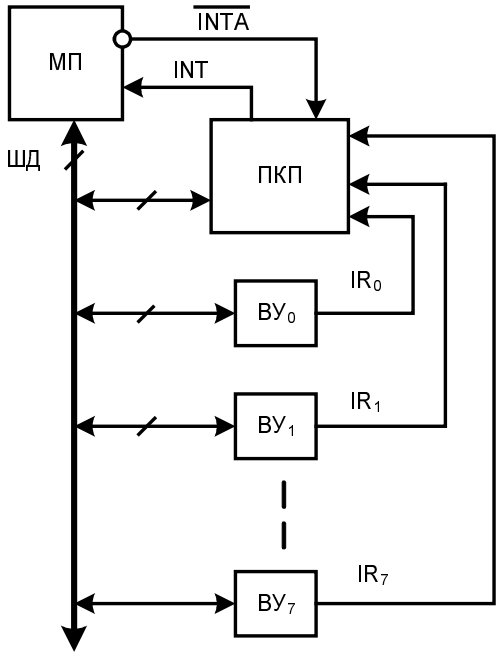


Рис. 7.22. Схема включения контроллера прерываний в микропроцессорную систему

Структура ПКП

Структура ПКП (PIC, Programmable Interrupt Controller) представлена на рис. 7.23.

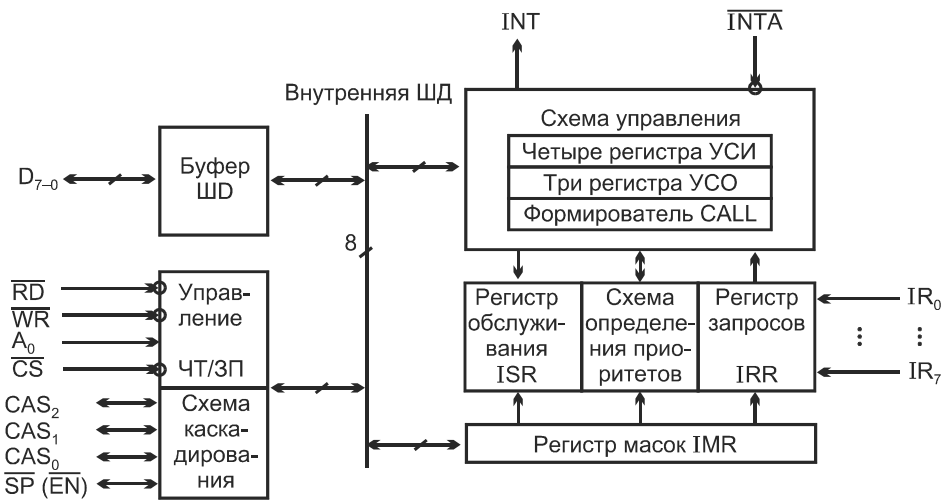


Рис. 7.23. Структура программируемого контроллера прерываний

Запросы прерываний от ВУ поступают на регистр запросов IRR (Interrupt Request Register), сохраняющий их до принятия на обслуживание. Биты запросов сопоставляются с битами регистра масок IMR (Interrupt Mask Register) как на стадии приема запросов, так и на более поздних стадиях их обработки (в схеме определения приоритетов и регистре обслуживания ISR). Маскирование осуществимо во всех перечисленных блоках.

Если приоритет запроса выше текущего, то при вложенных прерываниях формируется сигнал INT для процессора. При поступлении от процессора сигнала подтверждения прерывания $\overline{\text{INTA}}$ принятый запрос переходит в регистр обслуживания ISR (Interrupt Servicing Register) и сбрасывается в регистре запросов IRR. Установка бита ISR запрещает прерывания от всех других запросов с меньшими приоритетами. Подпрограмма обслуживания прерывания завершается сбросом бита регистра ISR.

Буфер ШД восьмиразрядный, двунаправленный, с третьим состоянием. При программировании контроллера через буфер передаются управляющие слова и считывается состояние регистров. При обслуживании прерывания по сигналам $\overline{\text{INTA}}$ через буфер ШД в шину данных системы выдается трехбайтная команда вызова подпрограммы.

Смысл сигналов $\overline{\text{RD}}$, $\overline{\text{WR}}$ и $\overline{\text{CS}}$ ясен (совпадает со смыслом этих сигналов в описанных выше устройствах). Остальные сигналы имеют следующее назначение:

- $\overline{\text{INTA}}$ — поступает от процессора в виде трех последовательных импульсов, для выдачи контроллером кода команды `CALL`, а затем младшего и старшего байтов адреса начала подпрограммы;
- $IR_0 ... IR_7$ — входы запросов прерывания;
- A_0 — младший разряд адреса, вместе с другими признаками показывает, к какому регистру управляющих слов (УСИ или УСО) обращается процессор;
- CAS_{2-0} — сигналы связаны с работой контроллера в групповой схеме, образуют выходную шину для ведущего контроллера и входную для ведомых;
- $\overline{\text{SP}}$ ($\overline{\text{EN}}$) — как $\overline{\text{SP}}$ сигнал определяет, является ли контроллер ведущим или ведомым в групповой схеме, как $\overline{\text{EN}}$ используется в так называемом буферизованном режиме для разрешения/запрещения выходов модулей на шину системы, т. е. для управления выходными буферами участников обмена.

Программирование контроллера

Перед работой контроллер программируют загрузкой управляющих слов инициализации (УСИ) и управляющих слов операций (УСО).

УСИ (от 2 до…4 в зависимости от назначаемого режима) записываются стробами $\overline{\text{WR}}$ и приводят контроллер в исходное состояние. С их помощью задаются адреса подпрограмм обслуживания запросов (интервал между начальными адресами подпрограмм принимается равным 4 или 8).

Управляющее слово УСИ1 определяет младшие разряды адресов подпрограмм; задает способ восприятия входных запросов IR_7...IR_0 (по фронтам или уровням); адресный интервал (4 или 8); определяет, является ли контроллер единственным или он работает в групповой схеме и отвечает на вопрос, понадобится или нет управляющее слово УСИ4, т. е. вводится или нет буферизованный режим. УСИ2 содержит старший байт начального адреса зоны подпрограмм обслуживания прерываний. Если несколько контроллеров работают в групповой структуре, то загружается и УСИ3. Ведущему контроллеру это слово сообщает, какие его входы подключены к ведомым, а каждому из ведомых — к какому входу ведущего подключен его выход INT. Таким образом, УСИ3 отражает физическую схему соединения контроллеров. Для буферизованного режима вводится УСИ4.

В результате инициализации сбрасываются триггеры приема запросов и регистры контроллера. Входу IR_7 присваивается низший приоритет, снимается специальное маскирование, чтение состояния устанавливается на регистре IRR, если не предусматривается буферизованный режим, то все функции, задаваемые УСИ4, обнуляются.

После инициализации контроллер может работать в базовом режиме. Для выбора других режимов загружаются управляющие слова операций УСО.

С помощью УСО1 можно в любое время программно установить или сбросить отдельные биты регистра масок. Каждый вход может быть замаскирован словом, содержащим логическую 1 в соответствующем разряде. Регистр масок воздействует и на регистр IRR, и на регистр ISR.

Слово УСО2 может задать пять вариантов завершения прерывания. Каждая подпрограмма обслуживания прерывания должна сообщать контроллеру о своем завершении, передавая ему одно из УСО2, в котором задан характер конца прерывания из числа следующих:

- КП — конец прерываний, т. е. неадресуемый конец прерываний, заключающийся в сбросе бита ISR с максимальным приоритетом, который был обслужен;
- СКП — специальный (адресуемый) конец прерываний, т. е. сброс бита, определяемого полем из трех разрядов слова УСО2;
- КПЦ — конец прерываний с циклическим сдвигом приоритета, т. е. сброс в регистре ISR бита, соответствующего последнему обслуженному запросу, и перевод дна приоритетного кольца на позицию этого бита;
- СКПЦ — специальный конец прерываний с циклическим сдвигом приоритетов, т. е. присвоение позиции дна тому входу, который указан полем слова УСО2 с одновременным выполнением обычного конца прерываний;
- УПЦ — установка приоритетов, т. е. присвоение позиции дна указанному в поле УС биту без выполнения операции обычного конца прерываний (без изменения регистра ISR).

Команды типа УСО3 применяются в режиме чтения и при установке/снятии режима специального маскирования.

Чтение состояния заключается в чтении регистров контроллера (IMR, IRR и ISR) или кода старшего из поступивших запросов. Чтение регистра масок по условиям $\overline{RD} = 0$ и $A_0 = 1$ не требует предварительной загрузки УСО3. Остальные регистры

считываются по команде `IN port` или при подаче низкого уровня напряжения на вывод $\overline{RD}$ после загрузки соответствующего УСО3.

В режиме опроса (поллинга) программа сама запрашивает информацию о запросах прерывания, обращаясь к источникам запросов поочередно в порядке понижения приоритета. Обнаружив запрос, программа осуществляет переход на подпрограмму обслуживания прерывания, которая и выполняет обмен данными. Режим опроса инициируется выдачей в контроллер УСО 3 с единичным значением бита P (Polling).

Каскадное включение контроллеров. Каскадирование контроллеров расширяет число обрабатываемых запросов. Возможно подключение к ведущему контроллеру не более восьми ведомых. Подключение одного ведомого контроллера (рис. 7.24) дает схему с 15 входами (остальные 7 контроллеров могут быть подключены аналогичным способом). По данной методике можно получить схему с 64 входами.

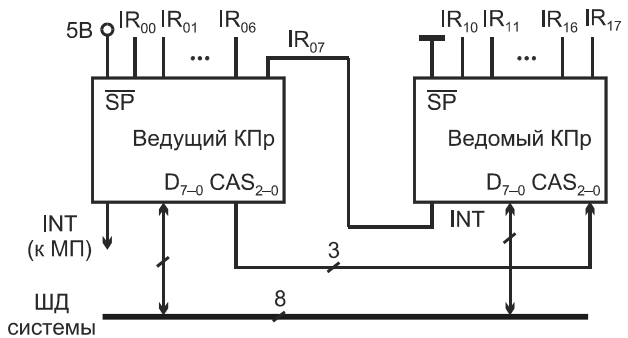


Рис. 7.24. Схема каскадного включения контроллеров прерываний

Функции ведущего и ведомого контроллеров определяются сигналами на входе $\overline{SP}$. Предварительно каждый ведомый контроллер получает номер, соответствующий номеру входа ведущего, к которому он подключен (это осуществляется загрузкой соответствующего УСИ3). На запросы по своим входам ведущий контроллер реагирует обычным способом, формируя команду `CALL`, как уже было описано. Для запросов от входов ведомого по первому импульсу $\overline{INTA}$, поступающему от МП, ведущий выдает на ШД команду `CALL`, а на шине CAS номер ведомого. По сигналам $\overline{INTA2}$ и $\overline{INTA3}$ адресованный ведомый контроллер выдает на ШД код адреса подпрограммы обслуживания.

По сигналу $\overline{INTA}$ устанавливаются соответствующие разряды в регистрах ISR обоих контроллеров. Поэтому подпрограмма обслуживания должна завершаться выдачей двух управляющих слов "Конец прерывания" для ведущего и ведомого контроллеров.

Подключение ПКП к шине микропроцессорной системы не требует специальных пояснений, поскольку ранее уже были определены все цепи, с которыми связаны выводы контроллера.

§ 7.9. Контроллеры прямого доступа к памяти

Прямой доступ к памяти ПДП (DMA — Direct Memory Access) — прямая передача данных от ВУ к памяти или от памяти к ВУ. При обычном обмене (без ПДП) передачи "Память ↔ ВУ" выполняются за два цикла: вначале данные от источника принимаются в процессор, а затем выдаются из процессора приемнику. При ПДП данные не проходят через процессор, и передача по тракту "Память ↔ ВУ" производится напрямую за один цикл. В режиме ПДП процессор отключается от системных шин и передает управление ими контроллеру прямого доступа к памяти (КПД). Программируемые КПД обслуживают прямой доступ с учетом требований различных систем. ПДП особенно удобен при передачах блоков данных, например, при обмене данными между внешней памятью и ОЗУ. Возможности контроллера позволяют организовать и обмен типа "память-память", т. е. решать задачу перемещения блока данных в адресном пространстве системы.

Взаимодействие блоков МПС при подготовке и реализации ПДП показано на рис. 7.25. Процессор программирует КПД, настраивая его на определенный режим работы, и может читать состояние контроллера. Соответствующие связи показаны штриховыми линиями. При ПДП процессор отключен, а контроллер вырабатывает сигналы управления для ВУ и ОЗУ. Тракт передачи данных непосредственно связывает ВУ с ОЗУ.

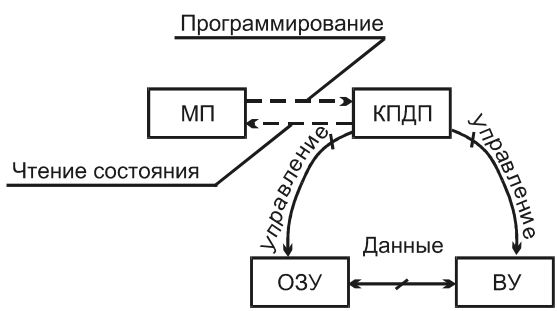


Рис. 7.25. Схема взаимодействия блоков микропроцессорной системы при ПДП

Возможны два вида *ПДП* — с *блочными* или *одиночными* передачами. В первом работа процессора останавливается на время передачи блока данных, во втором передачи слов в режиме ПДП чередуются с выполнением программы, и для ПДП выделяются отдельные такты машинных циклов, в которых процессор не использует системные шины. Например, каждый командный цикл процессора начинается с машинного цикла М1 — выборки кода команды. В цикле М1 есть такт декодирования принятой команды, в котором системные шины не используются. На это время их можно отдать для ПДП и передать одно слово. Производительность системы может возрасти благодаря тому, что ПДП будет для процессора "невидимым". Сам обмен с ПДП будет не быстрым, темп обмена нерегулярен,

т. к. длительности циклов различных команд различны, и, кроме того, ПДП может и замедлить выполнение программы, если цикл ПДП не уложится в интервал, соответствующий такту процессора.

В отличие от прерываний ПДП выполняется без участия программы и *не влияет на содержимое рабочих регистров процессора.* Поэтому при вхождении в режим ПДП не требуются затраты времени на передачу в стек содержимого рабочих регистров МП. ПДП предоставляется процессором по завершении текущего машинного цикла.

Структура и функции КПД

Базовая структура КПД — Intel 8237A, ее отечественные аналоги — микросхемы ВТ57 (рис. 7.26).

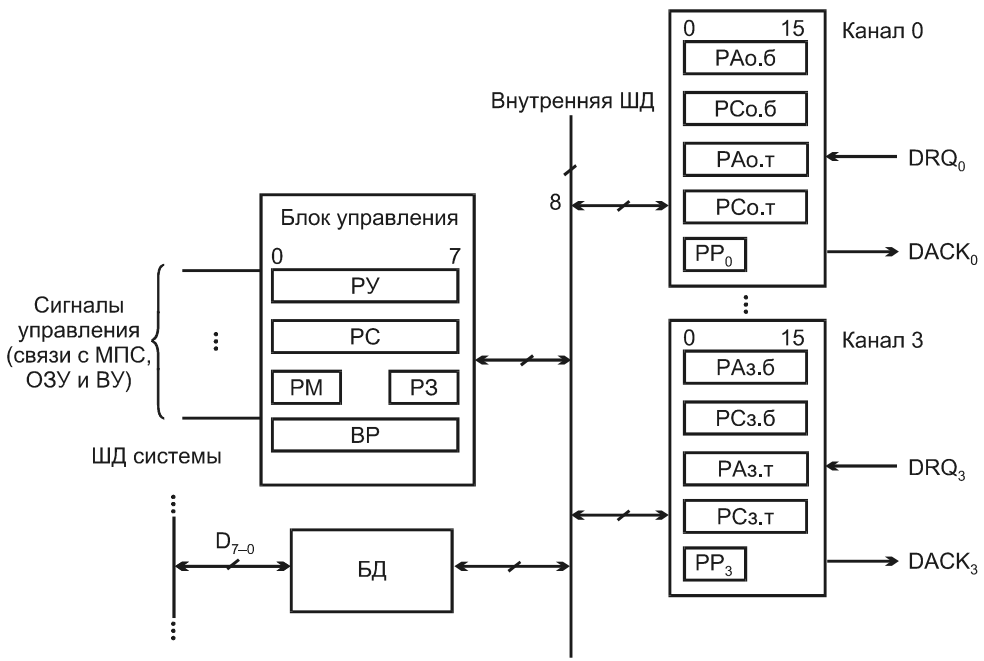


Рис. 7.26. Структура контроллера прямого доступа к памяти

Действия, выполняемые контроллером в режиме *блочных передач*:

- прием от процессора сведений о размещении в памяти передаваемого блока (начальный адрес и размер блока);
- трансляция запроса ПДП, исходящего от ВУ, т. е. запроса захвата шин у процессора с учетом маскирования и приоритетности поступающих запросов;
- прием от процессора сигнала подтверждения ПДП, свидетельствующего о том, что процессор отключился от системных шин;

- генерация адресов и сигналов управления для ЗУ и ВУ при передаче данных;
- фиксация завершения ПДП;
- снятие запроса захвата шин с соответствующего входа процессора и возвращение тем самым управления основной программе.

Контроллер имеет 4 независимых канала, возможно каскадирование контроллеров до любого числа каналов. В каждом из каналов размещено по пять регистров, а именно: два регистра адреса (базовый $РА_{i.б}$ и текущий $РА_{i.т}$, где i — номер канала), два регистра счета слов (базовый $РС_{i.б}$ и текущий $РС_{i.т}$) и регистр режима $РР_i$. Адресные регистры и регистры счета слов шестнадцатиразрядные, следовательно, начальный адрес блока данных может располагаться в любом месте адресного пространства емкостью 64К, а максимальный размер блока составляет 64 Кбайт.

> ***Примечание***
>
> У некоторых контроллеров для указания режима канала отдельный регистр не используется, а выделяются два разряда регистра-счетчика. При этом разрядность счетчика уменьшается до 14, а максимальный размер блока сокращается до 16К.

При программировании в оба адресных регистра загружается один и тот же адрес, а в оба регистра счета слов — одно и то же значение размера блока. При ПДП меняются состояния *текущих* регистров адреса и счета слов. Оба они *работают в режиме счетчиков* и при передаче очередного байта регистр адреса инкрементируется или декрементируется (в зависимости от программирования контроллера), а регистр счета слов декрементируется. Когда регистр-счетчик $РС_{i.т}$ дойдет до нуля и перейдет от состояния 0000h к состоянию FFFFh, выработается сигнал конца счета (т. е. в качестве начального значения в $РС_{i.т}$ следует загружать число, на единицу меньшее размера блока). Этим заканчивается передача блока.

Базовые регистры адреса и счета слов обслуживают *режим автоинициализации* канала. В них начальные адреса и размеры блоков сохраняются неизменными, что при необходимости позволяет в конце ПДП вновь загрузить текущие регистры теми же кодами и вновь повторить вывод того же блока данных, что и в предыдущем ПДП. Такой режим нужен, например, при управлении дисплеем, который для поддержания на экране какого-либо изображения нуждается в повторении блока данных с частотой в несколько десятков герц.

Загружаемое в *регистр режима* РР (см. рис. 7.26) слово направляется в тот или иной канал (для выбора канала используются два разряда слова) и определяет его режим (блочных или одиночных передач, обмена по требованию, каскадированная схема, запись, чтение, контроль, инкрементирование, декрементирование). В режиме "Контроль" нет передач, но сигналы доступа к данным формируются, что позволяет выполнять операции контроля информации, например, проверить данные, принятые внешним устройством. В режиме обмена по требованию передачи выполняются до выработки контроллером признака конца счета или поступления внешнего сигнала $\overline{EOP}$ (End of Process) или до перехода сигнала DRQ в пассивное состояние, т. е. до истощения ВУ в смысле исчерпания его дан-

ных. При этом возобновление данных в ВУ позволяет продолжить ПДП с помощью изменения сигнала DRQ. В периоды между ПДП, когда позволено работать процессору, промежуточные значения адресов и счетчика слов хранятся в текущих регистрах.

Регистр управления РУ загружается процессором, сбрасывается сигналом RESET или командой Master Clear. Определяет полярность сигналов, вид приоритета, разрешение или запрещение работы контроллера и др.

Регистр состояния РС содержит информацию о текущем состоянии контроллера и может читаться процессором. Четыре младших бита этого регистра устанавливаются каждый раз при конце счета или появлении внешнего сигнала $\overline{EOP}$ в соответствующем канале и сбрасываются сигналом RESET и при каждом чтении состояния. Четыре старших бита устанавливаются, если соответствующие каналы запрашивают обслуживание. Таким образом, РС позволяет определить, какие каналы закончили ПДП и какие требуют его.

Четырехразрядный *регистр масок* РМ имеет биты, соответствующие четырем каналам. Установка бита запрещает действие входного запроса DRQ. Если канал не запрограммирован на автоинициализацию, то по окончании ПДП он вырабатывает сигнал $\overline{EOP}$, при этом устанавливается бит маски этого канала. Этот бит устанавливается или сбрасывается также программно. Весь регистр устанавливается сигналом RESET, что запрещает запросы до поступления команды сброса регистра Clear Mask Rg, разрешающей начать прием запросов.

Регистр запросов РЗ позволяет контроллеру реагировать на запросы ПДП, исходящие от программы. Каждый канал имеет свой бит в этом четырехразрядном регистре, биты немаскируемы, но подчиняются требованиям приоритетности. Биты устанавливаются и сбрасываются индивидуально программой или сбрасываются после генерации контроллером признака конца счета или внешним сигналом $\overline{EOP}$. Весь регистр одновременно сбрасывается сигналом RESET.

Временный регистр ВР используется при передачах типа "память-память" для временного хранения данных и всегда содержит последний байт, переданный в предыдущей операции, если не сброшен сигналом RESET.

Для выбора внутренних регистров используются четыре линии адреса A_{3-0}, но число загружаемых байтов (16-разрядные регистры могут загружаться только двумя передачами по 8 разрядов) превышает 16. Поэтому в каналах имеются счетные триггеры, переключающиеся при последовательных передачах байтов по признаку первый/последний для направления байтов в соответствующую часть регистров. Эти триггеры должны быть сброшены до записи новых значений адреса и счета слов для правильного распознавания младших и старших байтов 16-разрядных слов. Для этого имеется команда Clear First/Last. Как и две упомянутые ранее команды Master Clear и Clear Mask Rg, эта команда выполняется при программировании контроллера.

В контроллере применено мультиплексирование шин для передачи старшего байта адреса через шину данных во внешний регистр, где этот байт фиксируется стробом ADSTB, после чего шина данных используется для других передач.

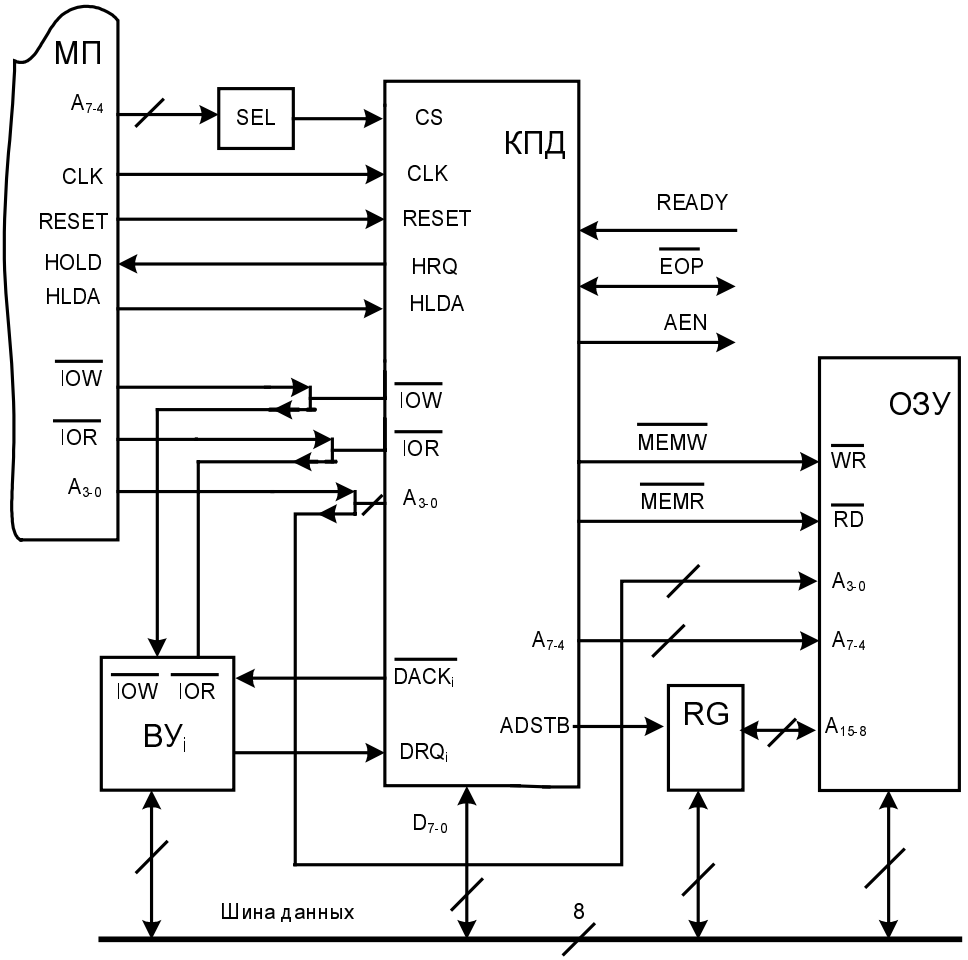


Рис. 7.27. Схема взаимодействия процессора, памяти и внешних устройств с контроллером прямого доступа к памяти

В работе контроллера можно выделить *две фазы* — простоя и активную. В фазе простоя контроллер находится, когда на его входах нет запросов ПДП. В этой фазе контроллер может быть запрограммирован процессором. Состояние программируемости продолжается и после начала действий ПДП, до момента получения от процессора ответа (сигнала HLDA) о предоставлении ПДП. При простое контроллер постоянно (в каждом такте) проверяет линии DRQ_i (не поступил ли запрос от ВУ) и линию $\overline{CS}$ (не обращается ли процессор к регистрам контроллера). В последнем случае линии адреса A_{3-0} выбирают регистры, а по стробам $\overline{IOR}$ и $\overline{IOW}$ производятся чтение или запись.

Запрос незамаскированного канала порождает запрос прерывания HRQ для процессора, реализующего один из вариантов ПДП, указанный в регистре режима. После поступления сигнала HLDA, когда процессор освободил шины системы, начи-

наются рабочие состояния контроллера. Они проходят в активной фазе и, если требуется, вводятся такты ожидания с учетом сигнала готовности/неготовности READY участников обмена (входного сигнала контроллера). Для передач "ВУ → память" генерируются пары одновременных сигналов $\overline{IOR}$ и $\overline{MEMW}$ для передач "память → ВУ" — пары $\overline{IOW}$ и $\overline{MEMR}$.

Подробная схема взаимодействия блоков в микропроцессорной системе с ПДП дана на рис. 7.27.

Выводы и сигналы контроллера

Выводы и сигналы контроллера имеют следующий смысл:

D_{7-0}	двунаправленная ШД с тремя состояниями;
RESET	сброс внутренних регистров контроллера и выходных управляющих сигналов. После сброса контроллер перестает реагировать на запросы ПДП до выполнения программы инициализации;
CLK	тактовый сигнал синхронизации;
$\overline{IOW}$, $\overline{IOR}$	в режиме программирования это входные стробы записи или чтения для адресованного регистра контроллера, в режиме ПДП — выходные сигналы, стробирующие запись или чтение байта для ВУ;
A_{3-0}	линии адреса, в режиме программирования входные, адресуют регистры контроллера, в режиме ПДП — выходные, дают 4 младших разряда адреса, формируемого контроллером;
$\overline{CS}$	в режиме программирования разрешает работу контроллера, в режиме ПДП отключен;
A_{7-4}	линии адреса, формируемого контроллером при ПДП, в других режимах отключены;
READY	сигнал готовности, отсутствие которого переводит контроллер в состояние ожидания и удлиняет циклы обращения к памяти;
$\overline{MEMW}$, $\overline{MEMR}$	сигналы записи и чтения памяти в режиме ПДП. В циклах чтения и записи контроллер вырабатывает парные сигналы $\overline{IOW}$ и $\overline{MEMR}$ или $\overline{IOR}$ и $\overline{MEMW}$;
HRQ	(Hold Request) запрос захвата шин, выдается контроллером для МП на его вход HOLD;
HLDA	подтверждение захвата, поступает от МП и разрешает переход в режим ПДП;
AEN	указывает на режим ПДП, может быть использован для блокировки шин адреса в других (невыбранных) устройствах с целью запрета их ошибочной работы;
ADSTB	сигнал выдачи адреса, разрешающий запись старшего байта адреса, вырабатываемого контроллером, по ШД во внешний регистр-защелку, хранящий адрес до конца цикла;

DRQ_i (i = 0...3)	запрос ПДП, формируемый ВУ. Для передачи одного байта должен быть снят внешним устройством по получении сигнала $DACK_i$ от контроллера. Для передачи массива данных должен удерживаться ВУ до получения от контроллера сигнала конца счета. Входы DRQ_i имеют фиксированные приоритеты (у DRQ_0 высший) или круговой приоритет;
$\overline{DACK_i}$ (i = 0...3)	подтверждение (разрешение) ПДП. Информирует ВУ о предоставлении ему ПДП. Вырабатывается согласно приоритетам входов DRQ_i;
$\overline{EOP}$	двунаправленный, информирует о завершении ПДП, генерируется в конце счета самим контроллером, может поступать извне. Появление внутреннего или внешнего $\overline{EOP}$ заставляет контроллер прекратить ПДП, сбросить запрос, и если разрешена автоинициализация, то загрузить текущие регистры канала от базовых.

Передачи "память-память"

Возможность организации передач *"память-память"* позволяет простыми средствами перемещать блок данных из одной области адресного пространства в другую. Установка младшего бита регистра управления определяет использование каналов 0 и 1 для передач "память-память". Передачи начинаются программной установкой запроса DRQ в канале 0. После подтверждения ПДП сигналом HLDA контроллер читает данные из области памяти соответственно адресам в регистре текущего адреса канала 0, которые формируются обычным способом (инкрементированием или декрементированием регистра адреса в зависимости от программирования). Прочитанный байт помещается в регистр временного хранения ВР. Затем канал 1 выполняет операцию передачи из регистра ВР в память соответственно текущим адресам в своем адресном регистре. Регистр счета слов канала 1 декрементируется. Когда в ходе передач текущий регистр счета слов канала 1 обнулится (точнее перейдет в состояние FFFFH), генерируется признак конца счета, обуславливающий появление сигнала $\overline{EOP}$, прекращающего процедуру ПДП.

Наращивание числа каналов ПДП

Для увеличения числа каналов ПДП несколько контроллеров могут объединяться в многоуровневую схему с подключением линий HRQ и HLDA контроллеров следующего яруса к входам DRQ и DACK предыдущего.

На рис. 7.28 приведен пример двухуровневой схемы с двумя контроллерами во втором ярусе, что позволяет получить 8 каналов ПДП. Если использовать показанным образом все четыре канала контроллера первого яруса, получится схема с 16 каналами.

Запросы контроллеров второго яруса распространяются через схемы приоритета первого яруса. Контроллер первого яруса именно для этого и используется, он не оперирует адресами и другими управляющими сигналами. Его роль состоит в при-

нятии сигналов DRQ и DACK, выработке HRQ и принятии HLDA. Другие функции не требуются.

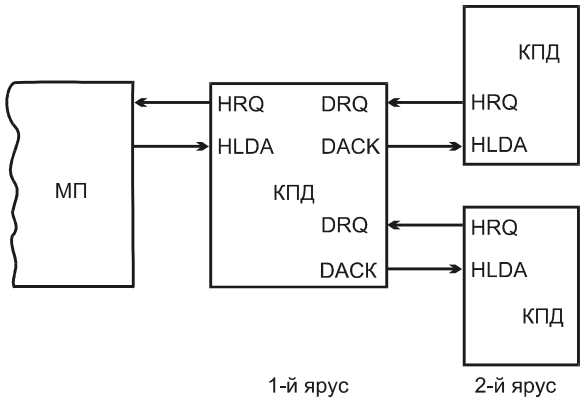


Рис. 7.28. Пример схемы наращивания числа каналов прямого доступа к памяти

§ 7.10. Таймеры

Таймеры выполняют операции, связанные с временами, частотами и интервалами. В МПС они используются для решения многих задач — реализации часов реального времени, подсчета числа внешних событий, генерации звуковых сигналов, ввода/вывода широтно-модулированных сигналов, организации мультизадачных режимов для операционных систем реального времени и др.

Термин "*таймер-счетчик*" отражает существование двух типовых режимов работы таймеров. Режим таймера — формирование интервалов, при этом на вход таймера подаются тактирующие импульсы постоянной частоты. Режим счетчика — подсчет поступающих сигналов, при этом на вход подаются внешние сигналы.

Простые таймеры

Простые таймеры входят, в частности, в состав микроконтроллеров. Типичные представители их — таймеры МК AVR (таймер 0, таймер 1 и сторожевой таймер). Структура таймера 1 показана на рис. 7.29, таймер 0 входит в нее в виде части, ограниченной штриховыми линиями (о конкретных значениях разрядностей счетчиков говорить не будем).

Таймер 0

Этот таймер-счетчик выполняет лишь простые операции формирования временных интервалов и подсчета числа внешних событий. Основной блок таймера — счетчик, на вход которого поступают сигналы с выхода мультиплексора. Мультиплек-

сор в зависимости от адресующего кода $CS_2 \ldots CS_0$ (CS — Clock Select) выбирает для подачи на вход счетчика один из восьми сигналов.

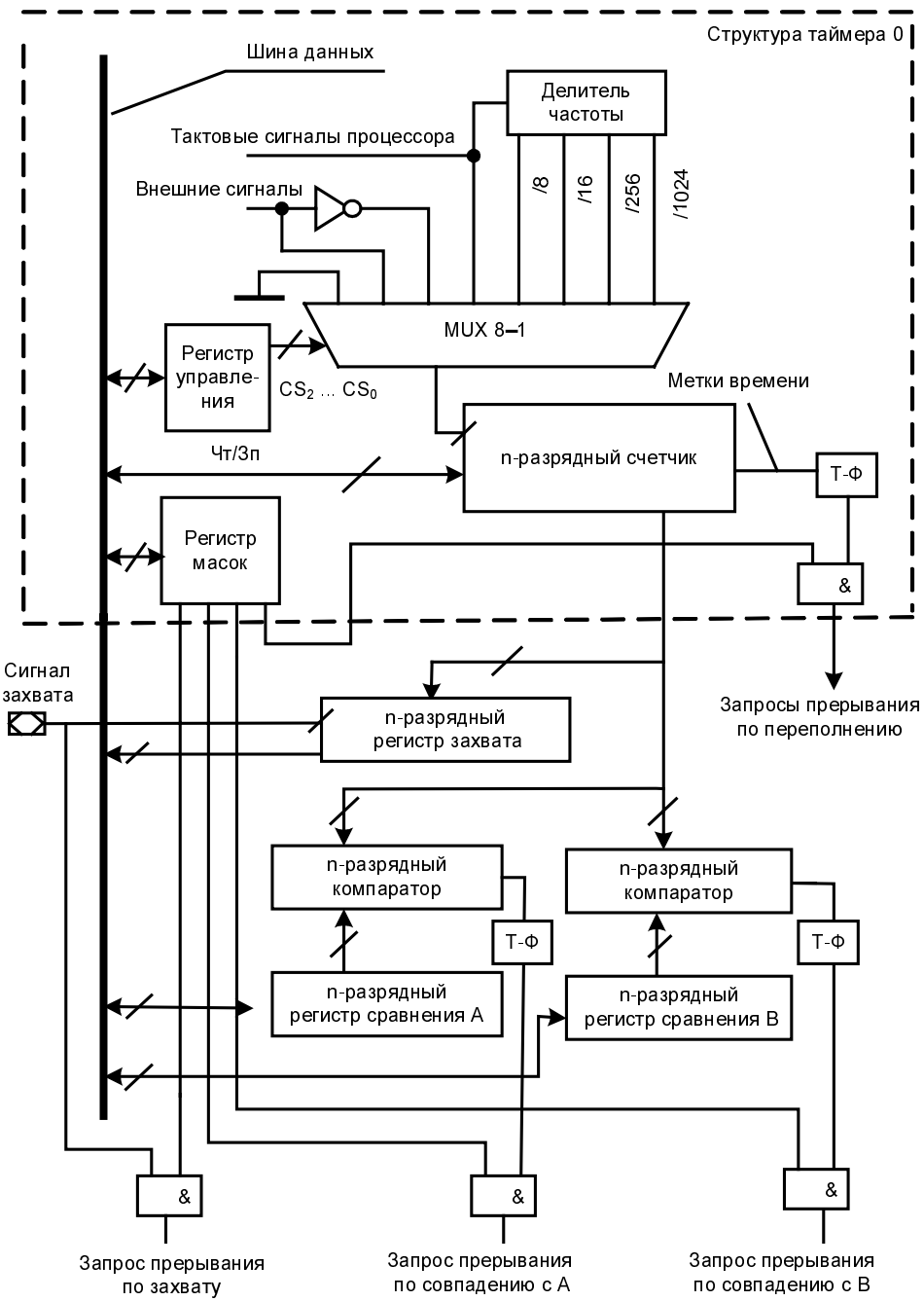


Рис. 7.29. Структура таймера

Выбор "заземленного" входа мультиплексора отключает счетчик, внешний сигнал может быть подан на счетчик в двух вариантах — как прямой или как инвертированный, а тактовые сигналы процессора — непосредственно или через делитель частоты с коэффициентами деления 8, 16, 256, 1024.

При переполнении (переходе от максимального значения кода к нулевому) счетчик устанавливает флаг запроса на прерывание (Т-Ф — триггер-флажок). Если прерывания по этому признаку разрешены (соответствующий разряд регистра масок установлен), то будет сгенерирован запрос прерывания по переполнению. Таким образом, таймер Т0 генерирует метки времени, формируя временные интервалы. Интервалы между метками зависят от выбора коэффициента деления частоты.

> Частным случаем формирования временных интервалов является работа с "часовым" кварцем, резонансная частота которого равна 32768 Гц, т. е. 2^{15} Гц. Подавая сигнал с генератора, стабилизированного таким кварцем, на 15-разрядный делитель частоты, получают метки времени с частотой 1 Гц, т. е. секундные интервалы. Затем включается цепочка трех делителей частоты с коэффициентами деления 60-60-24. После первого делителя получают метки с минутными интервалами, после второго — с часовыми, после третьего — с суточными. Снабдив делители частоты визуальными индикаторами их состояний, получают табло электронных часов.

В режиме подсчета внешних событий содержимое счетчика инкрементируется от фронтов сигналов, выражающих факт наступления события. Счетчик доступен для записи и чтения в любое время. Регистр управления содержит адресующие сигналы мультиплексора.

Таймер 1

В этой структуре помимо схем, входящих в состав таймера 0, присутствуют новые блоки: регистр захвата и два регистра совпадения (А и В). Реализуются и новые режимы:

- таймера событий;
- генератора ШИМ-сигналов.

В первом из этих режимов используется регистр захвата, который в произвольный момент времени, задаваемый сигналом загрузки (захвата), запоминает текущее состояние счетчика (захватывает его). Сигнал загрузки, если он поступает от внешнего входа, должен быть защищен от действия помех специальными приемами подавления "дребезга контактов".

Благодаря части схемы, содержащей компараторы, таймер может отмечать моменты равенства содержимого счетчика заданным значениям (А и В). Таким образом, эти моменты могут быть использованы для выполнения определенных действий.

Формирование ШИМ-сигналов

Вывод ШИМ-сигналов возможен в двух вариантах. В первом варианте запросы прерывания генерируются при совпадении текущего значения счетчика с величи-

нами A и B. Обработка запросов позволяет сформировать ШИМ-сигнал. В частности, если по совпадению с A устанавливается высокий уровень, а по совпадению с B он сбрасывается, сгенерируется импульс с длительностью, пропорциональной разности B – A.

Во втором варианте широтно-импульсная модуляция заключается в выработке импульсных сигналов с программируемыми частотой и скважностью. В этом режиме счетчик работает как реверсивный, а его содержимое изменяется от нулевого до некоторого максимального (TOP-значения), затем направление счета изменяется и содержимое счетчика, уменьшаясь, возвращается к нулевому значению, после чего такой цикл вновь повторяется. Пренебрегая ступенчатым характером изменения кода в счетчике, можно считать, что содержимое счетчика изменяется по пилообразному закону. Частота и амплитуда пилообразного сигнала зависят от входной частоты счетчика и от TOP-значения. Код в регистре совпадения можно изменять в моменты достижения счетчиком TOP-значения. Компаратор сравнивает содержимое счетчика с величиной модулирующего воздействия, отображаемого кодом в регистре совпадения, выделяя во времени интервалы по признаку "больше" или "меньше". В результате генерируется ШИМ-сигнал (рис. 7.30).

Признаки состояний счетчика (переполнение, совпадение, захват) отображаются в регистре прерываний от таймера соответствующими флажками, а разрешение или запрещение прерываний задается битами регистра маскирования прерываний. Счетный регистр в любое время доступен для записи и чтения.

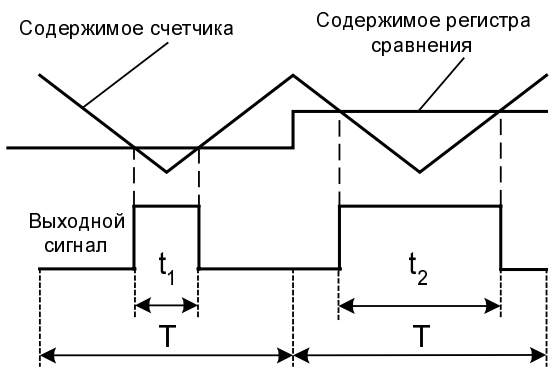


Рис. 7.30. К пояснению процесса генерации ШИМ-сигнала таймером T1

Сторожевой таймер

Сторожевой таймер (WDT, Watch Dog Timer) предназначен для отслеживания ситуаций, в которых МК неправильно выполняет программу, что может быть следствием воздействия помех, особенно опасных при работе МК в окружении мощного силового оборудования. Сторожевой таймер, переполняясь, вызывает сброс МК (его перезапуск), после которого начинается новое выполнение программы. Если

программа выполняется правильно, то она должна периодически сбрасывать сторожевой таймер через промежуток времени, меньший его периода, так что переполнения таймера не возникнет. Если таймер не будет сброшен в течение заданного промежутка времени, то переполнение таймера квалифицирует выполнение программы как неправильное, и программа начинает выполняться заново. Таким образом, сторожевой таймер защищает работу системы от сбоев.

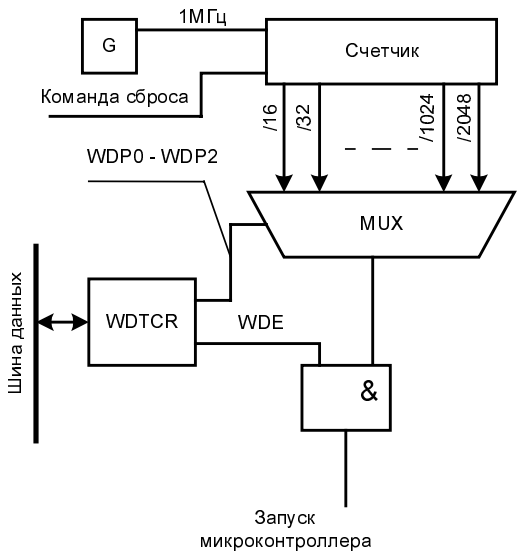


Рис. 7.31. Структура сторожевого таймера

Сторожевой таймер (рис. 7.31) имеет независимый генератор тактовых сигналов, который сохраняет рабочее состояние таймера даже в режиме существенного понижения мощности МК, так что этот таймер может быть использован для "пробуждения" микроконтроллера, т. е. вывода его в активное состояние. Сбросы таймера происходят под воздействием команды, а импульсы сброса/запуска контроллера выбираются мультиплексором с одного из 8 выходов счетчика. Коэффициенты деления входной частоты счетчика для этих выходов различны, а периоды тайм-аута в зависимости от выбора того или иного выхода составляют 16, 32, 64, 128, 256, 512, 1024 или 2048 мкс. Величина тайм-аута выбирается трехразрядным кодом WDP0…WDP2, загруженным в регистр WDTCR (WDT Control Register). Включение сторожевого таймера производится установкой бита WDE (Watch Dog Enable) в том же управляющем регистре.

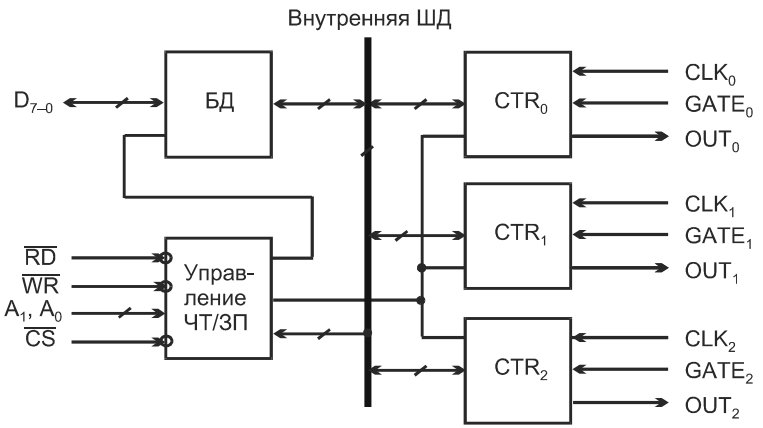
Программируемый интервальный таймер

Базовая структура *программируемого интервального таймера* (ПИТ, PIT) — микросхема Intel 8254 (отечественный аналог ВИ54). Таймер трехканальный, содержит три 16-разрядных счетчика с независимыми режимами работы при входных часто-

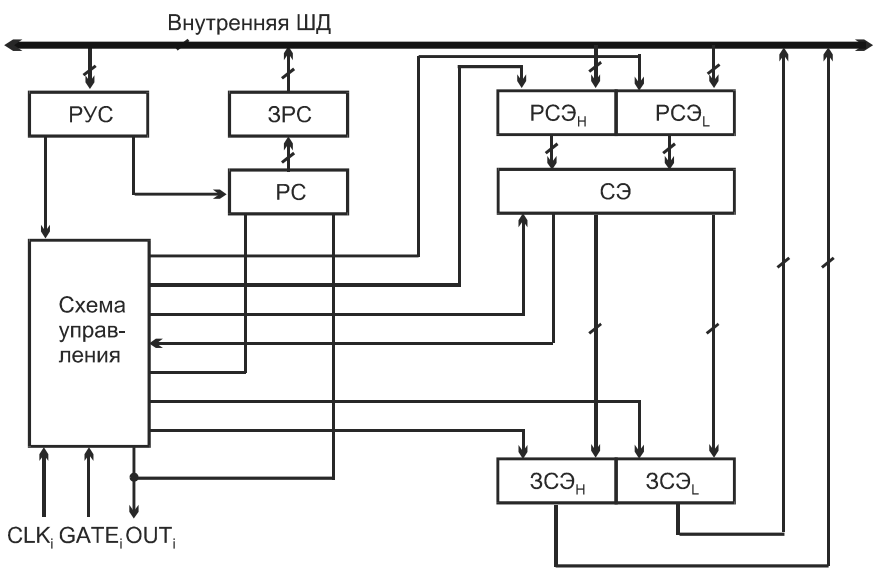
тах от нулевой до 5; 8 или 10 МГц (для разных модификаций). Таймер может работать в шести режимах в двоичной или двоично-десятичной системах счисления.

Структура таймера

Структура таймера представлена на рис. 7.32, *а*. Двунаправленный буфер данных БД с тремя состояниями связывает ПИТ с шиной данных системы.



а



б

Рис. 7.32. Структура интервального таймера (*а*) и одного из его каналов (*б*)

Блок управления чтением-записью принимает от шин МПС сигналы $\overline{RD}$ ($\overline{IOR}$) или $\overline{RW}$ ($\overline{IOW}$), первый из которых передает содержимое адресуемого счетчика или регистра ПИТ на шину данных, а второй загружает байт с этой шины в адресуемый счетчик или регистр. Сигнал $\overline{CS}$ разрешает или запрещает работу ПИТ. Сигналы младших линий адреса A1 и A0 выбирают конкретный счетчик CTRi комбинациями 00, 01 и 10, или регистр управляющего слова (комбинацией 11).

Структура счетчика CTR_i приведена на рис. 7.32, *б*. Регистр управляющего слова РУС хранит загруженные в таймер сведения о назначенных режимах работы счетчиков. Регистр состояния РС вместе с защелкой ЗРС (защелкой регистра состояния) отображает часть управляющего слова, состояние выхода OUT и флажок нуля счета. Собственно счетчик, обозначенный как СЭ (счетный элемент) — 16-разрядный синхронный вычитающий, со сбросом. Его состояние отслеживается двумя 8-разрядными защелками счетного элемента ЗСЭ для старшего (H) и младшего (L) байтов числа, формируемого в счетчике. Имеется команда Counter Latch (защелкивание), по которой текущее число СЭ фиксируется в защелках до его считывания процессором, после чего защелки вновь возвращаются в режим слежения за числом в СЭ. Управляющая логика обеспечивает поочередный вывод содержимого защелок на внутреннюю шину данных для вывода 16-разрядных слов по 8-разрядным шинам. Состояние СЭ может читаться только через защелки.

СЭ имеет на входах 8-разрядные регистры счетного элемента РСЭ, хранящие старший и младший байты нового числа, подлежащего записи в СЭ. Управляющая логика обеспечивает загрузку регистров с внутренней шины данных для побайтной передачи 16-разрядных чисел. В СЭ оба байта загружаются одновременно. При программировании счетчика регистры РСЭ сбрасываются.

Работа таймера. После включения питания необходимо запрограммировать каждый счетчик, который будет использоваться, записывая *управляющее слово*, а затем *начальное число*.

Управляющее слово, адресуемое комбинацией A_1A_0 = 11, двумя старшими разрядами (D_7, D_6) определяет номер счетчика, для которого предназначено, или же команду Read Back. Двумя следующими разрядами (D_5, D_4) определяются чтение-запись однобайтных или двухбайтных чисел или же команда Counter Latch. Далее задается режим работы счетчика (разряды D_3, D_2, D_1) и используемая система счисления (разряд D_0).

После управляющего слова в счетчик записывается начальное число в формате, определенном управляющим словом. Новые начальные числа могут быть записаны в счетчик в любое время.

Шесть разрядов слова состояния счетчика отображают содержимое разрядов D5…D0 управляющего слова, кроме того, в слове состояния отображается значение выходного сигнала OUT и наличие в счетчике числа или нуля.

Чтение содержимого счетчика без нарушения процесса счета организуется тремя способами: простым чтением, чтением по команде Counter Latch и чтением по команде Read Back.

Первый вариант осуществляется с помощью остановки счетчика по входу GATE путем запрещения его тактовых сигналов.

При втором варианте в РУС записывается команда Counter Latch , действие которой ведет к защелкиванию соответствующих ЗСЭ, из которых процессор читает число, после чего защелки возвращаются в режим слежения. Таким образом, содержимое счетчика читается "на лету", без влияния на его работу.

Третий вариант реализуется командой Read Back, записываемой в РУС. Два старших бита при этом идентифицируют команду, два следующих определяют кодом "1 из N" объект чтения (защелки счетного элемента или регистра состояния), три следующих выбирают счетчик (кодом "1 из N"), последний бит должен быть нулевым. Таким образом, можно проверить число в счетчике и, согласно формату слова состояния, наблюдать уровень выходного сигнала OUT, наличие в счетчике числа или нуля, запрограммированные режимы работы счетчиков, зафиксированные в управляющем слове.

При описании режимов работы ПИТ используются термины: *тактовые импульсы* — импульсы сигнала CLK, *запуск* — положительный фронт сигнала GATE, *загрузка счетчика* — передача числа из РСЭ в СЭ.

Режим 0

Режим 0 — прерывание по окончании счета. Здесь после записи управляющего слова CW выход OUT имеет низкий уровень L и остается таким до обнуления счетчика, приводящего выход к высокому уровню H до записи нового числа или управляющего слова режима 0.

Сигнал GATE разрешает (при единичном значении) или запрещает (при нулевом) процесс счета. При загрузке начального числа N переход сигнала OUT на высокий уровень происходит на N + 1 импульсе после записи начального числа. На рис. 7.33 показаны процессы в счетчике для режима 0 при постоянно разрешенном счете (*а*), наличии интервалов запрещения счета (*б*), когда GATE = var и при поступлении нового начального числа в процессе счета, т. е. перезаписи начального числа DW (Data Word) (*в*).

Режим 1

Режим 1 — аппаратно-перезапускаемый одновибратор. В этом режиме выход OUT первоначально имеет высокий уровень, после сигнала запуска формируется его отрицательный фронт и начинается счет, а при достижении счетчиком нулевого состояния выход OUT возвращается в исходное состояние H до поступления нового сигнала запуска. Начальное число N дает импульс длительностью в N тактов.

Одновибратор назван перезапускаемым, т. к. выход OUT остается на низком уровне в течение N тактов после любого запуска, в том числе поступившего во время существования выходного импульса.

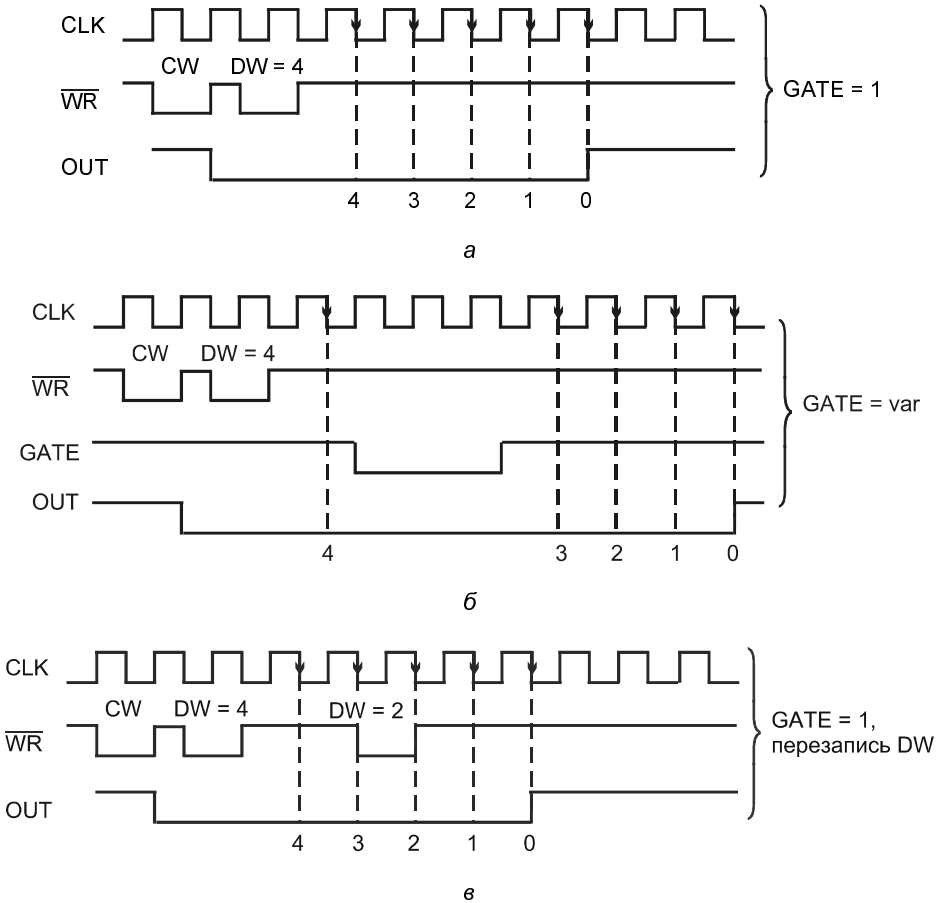


Рис. 7.33. Временные диаграммы режима 0 интервального таймера

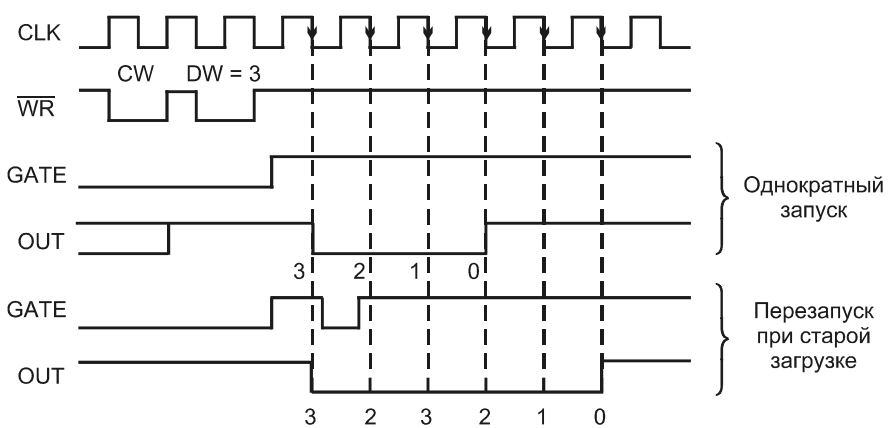


Рис. 7.34. Временные диаграммы режима 1 интервального таймера

Если новое число записывается в счетчик во время импульса, то текущий импульс не изменяется, если нет перезапуска. Перезапуск продлевает импульс на время, соответствующее новому загруженному числу. Временные диаграммы режима 1 показаны на рис. 7.34.

Режим 2

Режим 2 — генератор частоты. В этом режиме счетчик делит частоту входных тактовых импульсов на N. Начальный уровень выхода OUT является высоким. Когда число в счетчике равно единице, выход снижается до низкого уровня L на время, равное одному периоду тактовых импульсов CLK, после чего вновь возвращается на высокий уровень H, счетчик перезагружается начальным числом, и процесс повторяется. Режим периодический, начальное число N определяет период выходных импульсов (рис. 7.35).

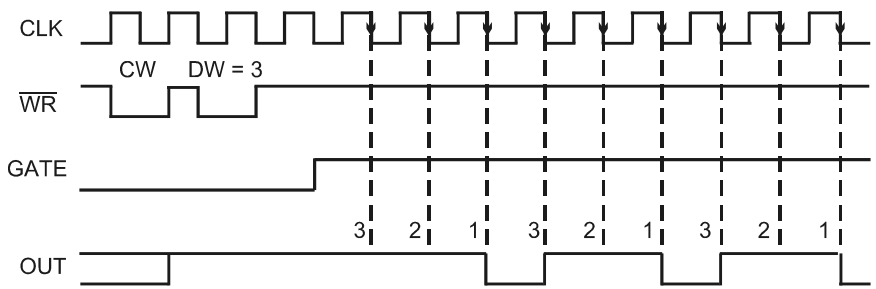


Рис. 7.35. Временные диаграммы режима 2 интервального таймера

Запрещение счета по входу GATE ведет к немедленному переходу выхода OUT на высокий уровень. Сигнал запуска перезагружает счетчик начальным числом. Запись нового числа во время счета не влияет на текущую последовательность счета.

Режим 3

Режим 3 — генерация меандра, т. е. последовательности импульсов с приблизительно одинаковыми длительностями уровней H и L (импульса и паузы). Такие сигналы часто используются для тактирования бодовой скорости при последовательных передачах данных. Этот режим близок к режиму 2. Начальный уровень выхода OUT высокий. После уменьшения начального числа до его половины сигнал OUT переходит на низкий уровень на оставшееся время счета. Начальное число N определяет период выходных импульсов. Запрещающий уровень сигнала GATE немедленно переводит выход OUT на высокий уровень. Запуск перезагружает счетчик.

Запись нового числа во время счета не влияет на текущую последовательность. Если после записи нового слова происходит перезапуск до конца текущего полупе-

риода, счетчик будет загружен новым числом в конце текущего полупериода и счет будет продолжен. При нечетном начальном числе N длительности импульса и паузы составляют соответственно (N + 1)/2 и (N – 1)/2 периодов частоты CLK.

Режим 4

Режим 4 — генератор одиночного программно-запускаемого строба. Начальное состояние выходного сигнала OUT — высокий уровень. При полном списывании начального числа выход OUT переходит на низкий уровень на время одного тактового импульса частоты CLK и затем возвращается на высокий уровень. Счетная последовательность запускается самой записью начального числа. Перезагрузка счетчика во время счета дает следующее:

- загрузка младшего байта не влияет на счет;
- загрузка второго (старшего) байта позволяет новому числу записаться в счетчик по следующему импульсу частоты CLK (рис. 7.36).

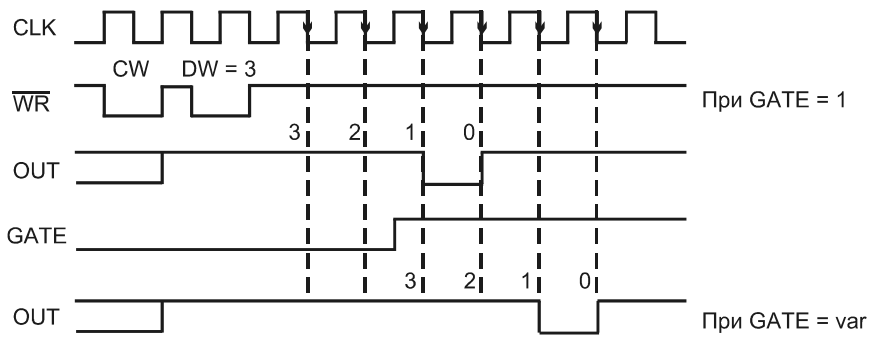


Рис. 7.36. Временные диаграммы режима 4 интервального таймера

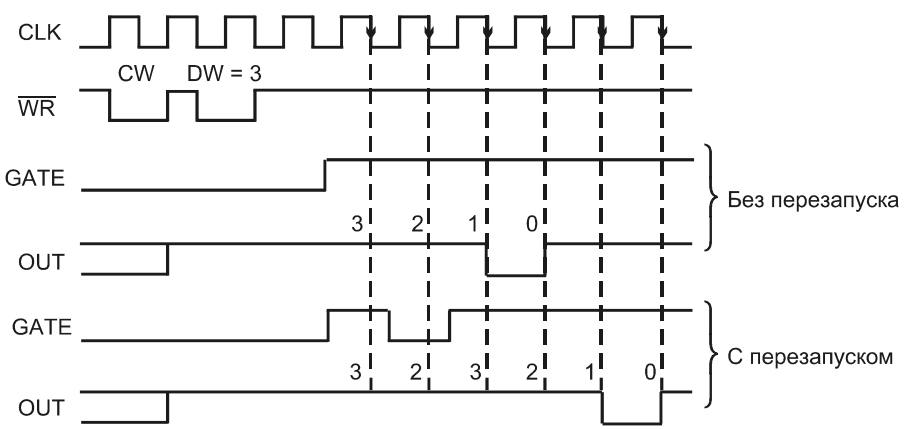


Рис. 7.37. Временные диаграммы режима 5 интервального таймера

Это позволяет последовательности запускаться от программного воздействия. Выход OUT перейдет на низкий уровень во время N + 1 импульса частоты CLK после записи нового числа N.

Режим 5

Режим 5 — генератор одиночного аппаратно-запускаемого строба. Вначале выход имеет высокий уровень. Счет запускается фронтом сигнала GATE. Списывание начального числа ведет к переходу сигнала OUT на низкий уровень на время одного импульса частоты CLK, после чего OUT возвращается на высокий уровень (рис. 7.37).

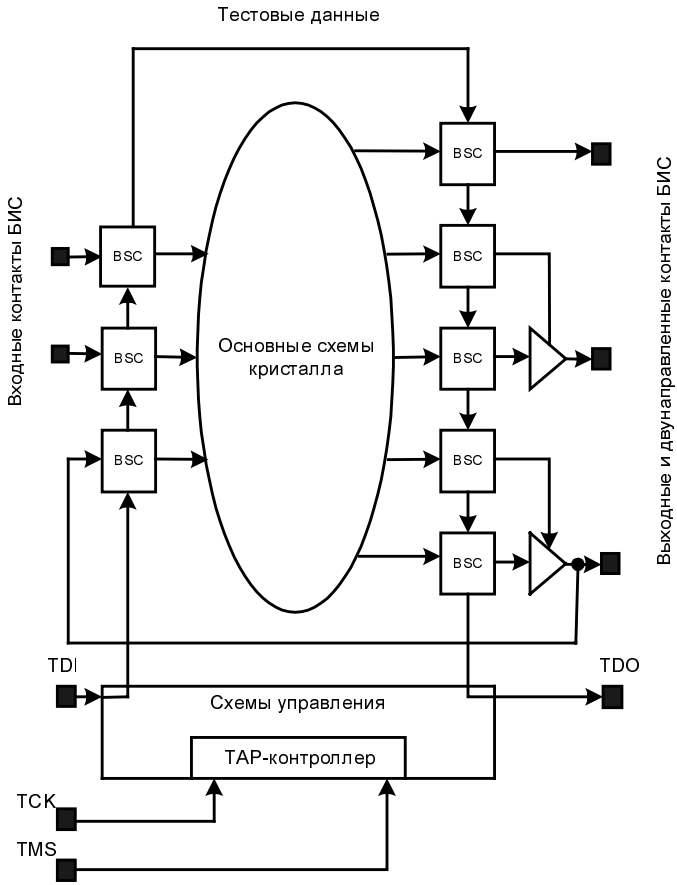
§ 7.11. Схемотехника интерфейса JTAG

Интерфейс JTAG и граничное сканирование

JTAG-интерфейс решает специфичные задачи тестирования БИС и конфигурирования программируемых структур. Термин "JTAG-интерфейс" обозначает совокупность средств и операций, позволяющих тестировать БИС без физического доступа к каждому их выводу. Аббревиатура JTAG возникла по наименованию разработчика — объединенной группы по тестам Joint Test Action Group. Термином "граничное сканирование" (ГС) или, иногда, "периферийное сканирование" (Boundary Scan Testing или BST) назвали тестирование по JTAG-стандарту (IEEE Std 1149.1). Такое тестирование возможно только для микросхем, внутри которых имеется набор специальных элементов — ячеек граничного сканирования BSC (Boundary Scan Cells) и схем управления их работой. Позднее функции интерфейса JTAG были расширены и он нашел широкое применение для конфигурирования микросхем с программируемой структурой. В настоящее время все большее число БИС/СБИС поддерживает транспортный механизм интерфейса JTAG, а многие и весь стандарт 1149.1.

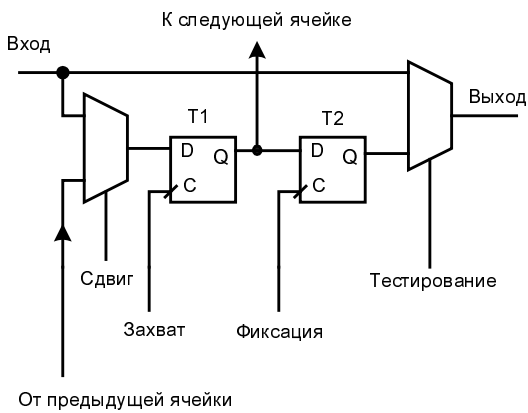
Основная концепция граничного сканирования иллюстрируется рис. 7.38, *а.* Ячейки BSC размещены между каждым внешним выводом микросхемы и схемами кристалла, образующими само проверяемое устройство, и могут работать в разных режимах. *В рабочем режиме* они просто пропускают сигналы через себя слева направо и не изменяют функционирования устройства. Входные сигналы проходят через ячейки BSC прямо к соответствующим точкам основных схем кристалла. При этом выводы обычного логического типа снабжаются одной BSC, для выходов с третьим состоянием нужны две (вторая для выработки сигнала управления буфером), для двунаправленных выводов — три, что и показано на рис. 7.38, *а.*

В режиме активного тестирования пропуск сигналов через ячейки прекращается, а в сдвигающий регистр, образуемый цепочкой ячеек, можно ввести последовательно тестовый код и переписать его в статический регистр-защелку для подачи на входные точки основной схемы кристалла.



а

Рис. 7.38. Структура аппаратных средств интерфейса JTAG (*а*)



б

Рис. 7.38. Схема ячейки граничного сканирования (*б*)

Оба регистра (сдвигающий и статический) создаются ресурсами самих ячеек. Результат, который выработает основная схема, можно передать в выходные ячейки BSC и затем вывести из БИС последовательно в режиме сдвигающего регистра для сравнения с ожидаемым правильным результатом (анализ результата, как и подготовку входных данных для тестирования, осуществляет внешний тестирующий прибор).

Ячейка BSC

Ячейка BSC (см. рис. 7.38, *б*) содержит два мультиплексора и два D-триггера. В зависимости от адресного входа "Тестирование" выходного мультиплексора, ячейка либо свободно пропускает сигнал со входа на выход, либо передает на выход состояние триггера T2, образующего разряд статического регистра. Адресный сигнал входного мультиплексора "Сдвиг" управляет подачей на первый триггер входного сигнала (от логических входов микросхемы) или же сигнала от предыдущей ячейки. Таким образом, совокупность триггеров T1 при подаче на них сигналов от нижнего входа мультиплексора образует сдвигающий регистр, а при подаче сигналов от верхнего входа мультиплексора эти триггеры загружаются параллельно данными от входных контактов БИС. При передаче через входной мультиплексор сигнала от предыдущей ячейки тактовый сигнал "Захват" производит сдвиг на один разряд в регистре, образованном последовательным соединением триггеров T1. Параллельная загрузка триггеров T1 также тактируется этим сигналом.

По синхросигналу "Фиксация" текущее содержимое регистра, составленного из триггеров T1, переписывается в статический регистр, составленный из триггеров T2. Сдвиги в регистре на триггерах T1 не будут влиять на содержимое статического регистра.

Ресурсы ячеек делают возможным и так называемое *пассивное тестирование*, поскольку триггеры T1, соединяясь последовательно, образуют сдвигающий регистр, в котором могут быть зафиксированы и затем прочтены тестирующим устройством значения пропускаемых сигналов.

Интерфейс JTAG

При разработке интерфейса JTAG основным требованием была минимизация числа контактов БИС, используемых при выполнении тестирования. Таких контактов 4 (реже 5), их совокупность образует порт доступа TAP (Test Access Port). Назначение контактов:

- TCK — для сигнала синхронизации передач данных и команд;
- TMS — для выбора режима передач;
- TDI — для ввода данных и команд:
- TDO — для вывода данных, команд или состояния;
- TRST (если используется) — для сброса в исходное состояние контроллера TAP.

Столь малое число контактов оказывается достаточным вследствие последовательного характера передач команд и данных.

Транспортный механизм

Если в тестируемом устройстве установлено несколько БИС с интерфейсом JTAG, то они могут быть объединены в JTAG-цепочку (рис. 7.39).

При выполнении любых действий команды и данные передаются в цепочке последовательно. Используя только контакты TMS и TCK, устройство управления JTAG-цепочкой, входящее в состав тестирующего прибора, может устанавливать автоматы TAP-контроллеров всех БИС цепочки в любое требуемое состояние (исходное, загрузки команд или данных в регистры, чтения из них). Совокупность регистров всех БИС цепочки является как бы одним регистром на пути от выходного контакта TDO тестирующего прибора до его же входа TDI. Поэтому при настройке одной из БИС цепочки необходимо составить и ввести в цепочку последовательность битов с длиной, соответствующей всей цепочке (это относится как к цепочке регистров команд, так и к цепочке регистров данных). В режимах передач данных и команд каждый импульс TCK сдвигает на один разряд код в цепочке регистров.

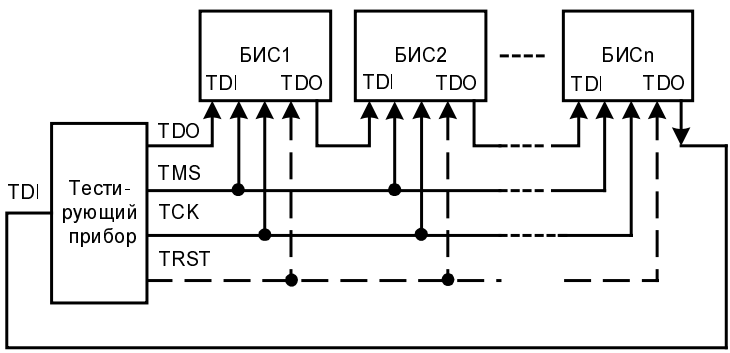


Рис. 7.39. Структура JTAG-цепочки

Устройство управления граничным сканированием

В это устройство (рис. 7.40) входят регистры команд (IR), данных (DR) и пропуска (BYPASS), а также выходной мультиплексор и контроллер управления (TAP Controller). Регистр данных (сканирующий регистр) образован последовательным соединением триггеров T1 ячеек BSC и принимает или выдает данные при выполнении в JTAG-цепочке любых команд.

Параллельно регистру данных включен регистр команд (параллельное включение означает общность последовательных входа и выхода TDI и TDO). Устройство управления принимает тактирующие сигналы TCK и интерпретирует команды на

входе TMS, выбирая тот или иной регистр для записи или чтения. Зафиксированные в регистрах-защелках команды дешифрируются и формируют сигналы выбора режиме граничного сканирования или регистров данных, подключаемых через мультиплексор к выходу TDO.

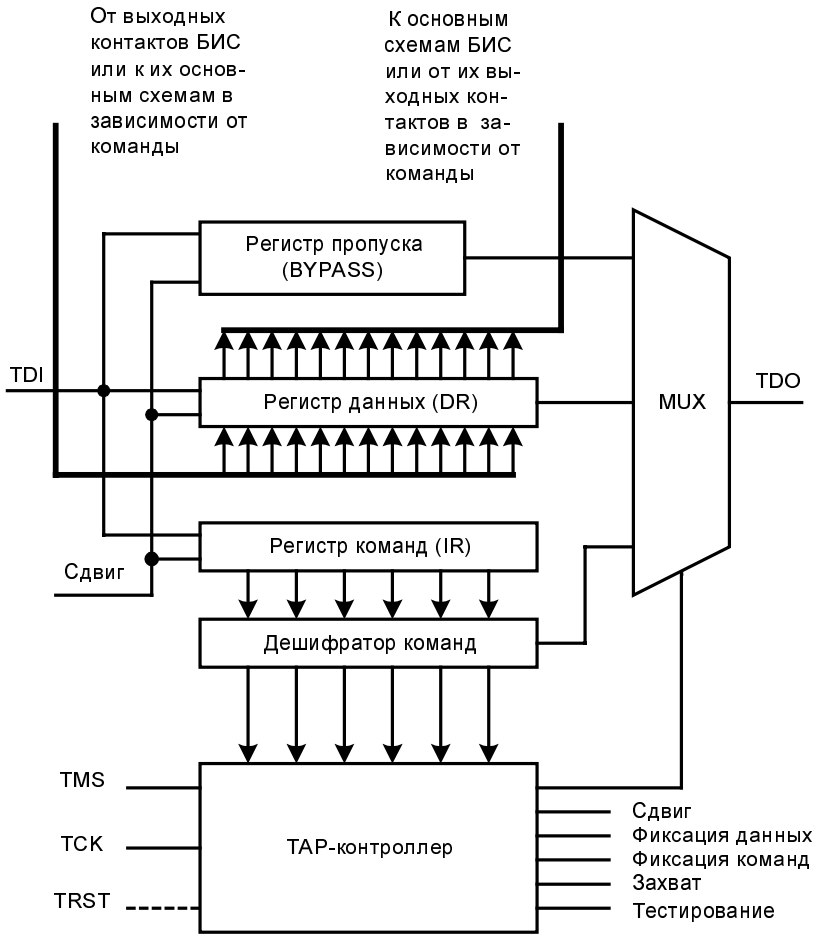


Рис. 7.40. Структура устройства управления граничным сканированием

Одноразрядный регистр пропуска (обхода) BYPASS используется в режимах загрузки /выгрузки данных как обходной путь для сдвигаемых многоразрядных данных, не относящихся к данной БИС. Прохождение в некоторых БИС JTAG-цепочки данных от входа TDI к выходу TDO через одноразрядный регистр BYPASS сокращает длину цепочки и ускоряет процессы тестирования.

Механизм граничного сканирования

Граничное сканирование выполняется в такой последовательности: загрузка команды, загрузка данных, исполнение команды, чтение результата.

Функциональные возможности ячеек BSC позволяют организовать следующие режимы граничного сканирования:

- *Самотестирования БИС или записи/чтения внутрисхемных ЗУ*. В этом режиме выводы БИС изолируются от ее внутренних схем, на которые подается информация от триггеров T2 BSC-ячеек (JTAG-цепочки). Дальнейшие действия определяются поступившей командой. Результирующая информация (реакция на введенные входные данные) может быть зафиксирована в триггерах T1 ячеек BSC и передана для анализа в тестирующий прибор, что и требуется для самотестирования БИС или записи/чтения внутрисхемных ЗУ;
- *Тестирования межсоединений БИС*. В этом режиме внешние контакты БИС также отключаются от ее внутренних схем. В выходные ячейки JTAG-цепочки одной БИС загружаются данные, которые затем передаются во входные ячейки JTAG-цепочки второй БИС. При исправности межсоединений БИС данные должны совпадать, что и проверяется тестирующим прибором;
- *Тестирования штатной работы БИС*. Здесь внутренние схемы кристалла соединяются с внешними контактами БИС, что соответствует ее рабочим режимам. В ячейках BSC в заданный момент времени фиксируется состояние всех контактов. Передача полученных данных в тестирующий прибор позволяет оценить правильность работы БИС. При этом внутренние сигналы схемы становятся известными без физического доступа к ее контактам;
- *Смешанные режимы*. В этих режимах часть БИС находится в штатном режиме, а другая часть — в тестовом.

Команды граничного сканирования

Выполняемые в ходе граничного сканирования тестовые процедуры определяются командами, поступающими из тестирующего прибора (обычно это персональный компьютер). Согласно стандарту минимальным является набор из четырех команд — `INTEST`, `BYPASS`, `EXTEST`, `SAMPLE/PRELOAD`. Можно использовать и дополнительные команды.

Команда `INTEST` проверяет функционирование БИС при подаче данных от тестирующего прибора в сканирующие регистры DR. Запуск процедуры сканирования ведет к проверке основных схем кристалла. Затем в регистре DR фиксируются результаты тестирования.

Команда `RUNBIST` тестирует кристалл без привлечения каких-либо внешних данных, активизируя встроенную в кристалл тестирующую схему. Результаты тестирования заносятся в регистр DR и позволяют оценить исправность БИС.

Команда `EXTEST` используется для тестирования внешних соединений БИС в устройстве (на печатной плате) и для задания входных воздействий другим БИС. При ее выполнении данные, загруженные в BSC-ячейки выходов БИС, считываются ячейками входов другой БИС, отражая исправность их межсоединений. Внутренние схемы кристалла при этом отключаются от внешних выводов.

Команда SAMPLE/PRELOAD в зависимости от состояния управляющего автомата позволяет либо "захватить" в сдвигающий регистр граничного сканирования данные с параллельных входов ячеек BSC, либо загрузить данные из сдвигающего регистра в статический регистр-защелку (зафиксировать данные, т. е. обновить содержимое статического регистра). Эта команда используется и при проверке поведения БИС в рабочих режимах, фиксируя данные на ее внешних контактах. Внутренние схемы кристалла от внешних контактов при этом не отключаются. Наличие "снимка" режимов нескольких взаимосвязанных БИС и возможность задания различных конфигураций тестируемой системы (одна группа БИС формирует тестовые сигналы, а другая находится в рабочем режиме) позволяет организовать совместное тестирование межсоединений и функционирования внутренних схем БИС.

Команды BYPASS, CLAMP и HIGHZ помещают между выводами TDI и TDO одноразрядный регистр BYPASS при различных состояниях выводов БИС (первая не влияет на работу схем на кристалле, вторая устанавливает на выводах данные от статического регистра, третья — Z-состояния).

С помощью интерфейса JTAG, как уже отмечалось, можно производить также реконфигурацию микросхем программируемой логики непосредственно в системе, без извлечения их из устройства.

Расширения интерфейса JTAG

Для классического JTAG тестирование ограничивалось цепями с исключительно цифровыми сигналами. Многообразие типов сигналов, характерное для современных печатных плат, заставило расширить перечень сигналов, тестируемых расширенными версиями интерфейса JTAG. Все большее распространение получают аналоговые сигналы, для которых анализ самих сигналов и тестирование соединений требуют новых подходов. Аналогична и ситуация с быстродействующей передачей дифференциальных пар сигналов через цепи постоянного или переменного тока (цепи с последовательно включенными конденсаторами). Естественно, подобные расширения требуют включения новых средств во все составляющие стандарта IEEE Std 1149.1. Изменения касаются как тестируемых ИС, так и тестирующего оборудования (персональных компьютеров, их программного обеспечения и аппаратурных добавок).

Тестирование аналоговых цепей и сигналов. Интегральные схемы, предназначенные для тестирования входных или выходных аналоговых сигналов, содержат помимо традиционных контактов интерфейса JTAG (TCK, TMS, TDI и TDO) два дополнительных контакта AT1 и AT2. Эти контакты всех ИС поступают на двухразрядную шину, подключенную к тестирующему оборудованию.

Каждая ИС цепочки содержит коммутирующий блок, позволяющий подключать специальные (аналоговые) сканирующие ячейки ИС к двухразрядной шине. Аналоговые сканирующие ячейки функционально близки к традиционным цифровым сканирующим ячейкам, но вместо соединения с цепочкой данных позволяют под-

ключать входные аналоговые контакты к коммутирующему блоку, а через него к тестирующему оборудованию. Подобная организация приводит к тому, что, в отличие от стандартного JTAG-тестирования, в каждый момент времени доступной является информация на двух произвольных аналоговых контактах ИС, а не "снимок" всех контактов одновременно. Конструкция коммутирующих цепей внутри ИС позволяет производить в тестирующем оборудовании не только проверку наличия или отсутствия соединений между определенными выводами ИС на печатной плате, но даже параметрические измерения (уровни напряжений, величины токов, значения омических или даже комплексных сопротивлений).

Конечно, проведение подобных измерений требует соответствующей контрольно-измерительной аппаратуры, управляемой ведущей тестовой системой. Точность проведения экспериментов определяется не только свойствами этой аппаратуры, но и качеством коммутирующих цепей внутри ИС. Несомненным достоинством разработанного подхода к аналоговому тестированию является совместимость со стандартом интерфейса JTAG. Более подробную информацию о тестировании аналоговых сигналов можно найти в [15].

Тестирование дифференциальных соединений по постоянному и переменному току. Для быстродействующей (десятки мегабит в секунду) качественной передачи информации широкое распространение получили соединения при помощи дифференциальных пар сигналов постоянного или переменного тока. Поэтому актуальна задача тестирования подобных соединений (стандарт IEEE Std 1149.6).

Изменения потребовались в организации как аппаратной части ИС, так и программной части стандарта для ведущей системы. Необходимость изменений диктовалась двумя основными обстоятельствами. Во-первых, контроль динамических характеристик цепей обмена потребовал добавления аппаратно-программных средств, связанных с временными параметрами обмена. Во-вторых, понадобились схемы, корректирующие форму или параметры передаваемых сигналов (дифференцирование сигналов или появление постоянной составляющей).

Выходные усилители ИС, предназначенные для передачи дифференциальной пары сигналов по цепям постоянного или переменного тока, могут формировать тестовые импульсы с длительностью или периодом, управляемым контроллером тестирующего устройства. В ИС добавлены схемы, фиксирующие значения на входных и выходных контактах ИС в регистрах граничного сканирования. Для контроля параметров, получаемых в приемниках дифференцированных сигналов, требуется наличие схем, восстанавливающих тестовые импульсы.

Создание импульсов с требуемыми свойствами в тестируемых устройствах потребовало и определенных изменений в командах TAP контроллеров. Стандарт вводит всего две новые команды для тестирования контактов переменного тока: EXTEST_PULSE и EXTEST_TRAIN. При создании нового стандарта разработчики стремились обеспечить совместимость с традиционным IEEE Std 1149.1. При тестировании устройств по стандарту 1149.1 можно использовать типовое ПО JTAG интерфейса, но для их исследования по стандарту 1149.6 включая определение параметри-

ческих характеристик интерфейса (таких как границы уровней срабатывания напряжений, допустимые временные параметры передаваемых сигналов) необходимо использовать специализированную аппаратуру и программное обеспечение. Более подробную информацию о тестировании дифференциальных пар сигналов можно найти в [16].

Контрольные вопросы и упражнения

1. Как следует понимать выражение "условный программный обмен"?
2. Чем радиальные прерывания отличаются от векторных?
3. Какие передачи называются симплексными, полудуплексными, дуплексными?
4. Какими тремя основными задержками оценивается быстродействие шинных формирователей?
5. Какими пятью задержками оценивается быстродействие буферных регистров?
6. Какие три адреса используются при работе с конфигурируемыми параллельными портами, не имеющими дополнительных функций, кроме функции ввода/вывода?
7. В чем состоит назначение программируемого параллельного адаптера (ППА)? В каких трех режимах может работать ППА?
8. Каким образом с помощью двухразрядного адреса в ППА адресуются 5 объектов?
9. Какую последовательность действий выполняет процессор для программирования ППА?
10. Составьте управляющее слово РУС, обеспечивающее работу ППА в режиме 1 для ввода в порт A и вывода из порта B при безразличном состоянии свободных линий порта C.
11. Укажите временную последовательность и источники сигналов $\overline{STB}$, IBF, INT, $\overline{OBF}$, $\overline{ACK}$, $\overline{RD}$, $\overline{WR}$, $D_{7\text{-}0}$ при вводе и выводе с квитированием с помощью ППА.
12. Объясните логический смысл сигналов INT, INTE, $\overline{INTA}$.
13. Благодаря чему битовая скорость передач через модем может превышать бодовую скорость?
14. Перечислите биты, обрамляющие информационные биты в кадре асинхронных передач.
15. Чем различаются синхронные передачи через программируемый связной адаптер (ПСА) с внутренней и внешней синхронизацией?

16. Какова роль синхросимволов, обрамляющих массив данных при синхронных передачах через ПСА с внутренней синхронизацией?

17. Какие три вида ошибок фиксируются при работе ПСА?

18. Что выражают сигналы TxRDY и RxRDY программируемого связного адаптера и какими двумя способами их можно использовать?

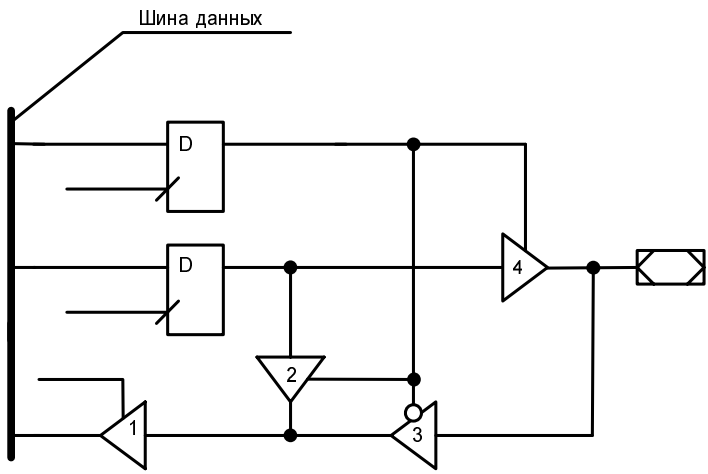
19. Какую последовательность действий выполняет процессор для программирования ПСА?

20. Каким образом ПСА, имеющий одноразрядный адресный вход C/D, может принимать несколько управляющих слов, определяющих его работу?

21. Составьте начальное управляющее слово MI (управляющее слово режима) ПСА, для его программирования на асинхронный режим с контролем по четности при передачах символов с 7 информационными битами, двумя стоп-битами и частотой передач в 1/16 от частоты CLK.

22. Составьте программу для обеспечения работы ПСА в режимах приема и передач данных с указанием видов синхронизации, числа информационных битов, условий контроля и т. д.

23. На рисунке показан возможный вариант реализации двунаправленной линии параллельного порта. Укажите назначение всех входов, получающих сигналы от шины данных, системы тактирования, управления буферным каскадом.



24. Нарисуйте схему, реализующую аппаратный опрос четырех источников запросов прерываний.

25. Чем специальное маскирование, осуществляемое контроллером прерываний, отличается от обычного?

26. Дайте определение прерываний с фиксированными и кольцевыми приоритетами и укажите их достоинства и недостатки.

27. Какую последовательность действий выполняет процессор для программирования контроллера прерываний?
28. Что означает термин "прямой доступ к памяти" (ПДП)? Зачем применяется ПДП?
29. Какую последовательность действий выполняет процессор для программирования контроллера прямого доступа к памяти (ПДП)?
30. Какие сигналы генерирует контроллер ПДП в режиме передачи блока данных?
31. Зачем в каналах ПДП загрузка текущих регистров дублируется загрузкой базовых?
32. Чем объясняется "двойной" термин "таймер-счетчик"?
33. Какая операция среди реализуемых микроконтроллерами понимается под названием "широтно-импульсная модуляция"?
34. Что такое "сторожевой таймер" и как он функционирует?
35. Какими тремя способами организуется чтение содержимого счетчиков в схемах программируемого интервального таймера?
36. Режим 0 программируемого интервального таймера называется "прерывание по окончании счета". Какие сигналы вырабатываются в этом режиме на выходе счетчика и какую роль в этих процессах играет сигнал GATE?
37. Какой процесс называют "граничным сканированием"? Какие аппаратные средства на кристалле требуются для его реализации?
38. Как функционируют ячейки граничного сканирования BSC в рабочем режиме БИС и в режиме ее активного тестирования?
39. Какие контакты БИС образуют порт доступа TAP при тестировании по стандарту JTAG?

Литература к главе: [11], [18], [19], [24], [27], [28], [30], [37], [39], [53], [XV].

Глава 8

SPLD и CPLD — простые и сложные программируемые логические устройства

§ 8.1. Микросхемы с программируемой структурой. Вводные замечания

В предыдущих главах рассмотрены типовые узлы и устройства — набор "строительных блоков" для создания электронной аппаратуры. В этой и следующей главах речь пойдет о *средствах реализации* узлов и устройств на кристаллах повышенного и высокого уровня интеграции. Эти средства — микросхемы *с программируемой структурой,* создание которых оценивается специалистами как очередная революция в микроэлектронике, не уступающая по значению так называемой микропроцессорной революции.

Цифровые системы содержат как стандартные части (процессор, память и др.), так и нестандартные, специфичные для данного проекта. Это прежде всего схемы управления модулями системы и обеспечения их взаимодействия. Реализация нестандартных частей системы исторически была связана с применением микросхем малого и среднего уровней интеграции (МИС и СИС), поскольку изготовление по заказу специализированных БИС связано с очень большими затратами средств и времени. Использование МИС и СИС сопровождается резким ростом числа корпусов ИС, усложнением монтажа, снижением надежности системы и ее быстродействия.

Возникшие трудности нашли разрешение на путях разработки микросхем с программируемыми структурами. Одни из таких микросхем изготовляются как законченные *стандартные* изделия и затем программируются пользователями в соответствии с требованиями конкретных проектов. Потребитель в этом случае избавляется от необходимости заказывать для себя дорогостоящие специализированные микросхемы. Другие разновидности изготовляются как *полуфабрикаты* и далее специализируются с помощью сокращенного числа технологических операций. В этом случае потребитель существенно уменьшает расходы на создание требуемой продукции в сравнении с расходами на изготовление полностью заказной схемы.

Первыми представителями микросхем с программируемой структурой явились:

- *программируемые логические матрицы* ПЛМ (*PLA*, Programmable Logic Array);
- *программируемая матричная логика* ПМЛ (*PAL*, Programmable Array Logic);
- *вентильные матрицы* ВМ (*GA*, Gate Array), чаще называемые в отечественной литературе *базовыми матричными кристаллами* (БМК).

Микросхемы PLA (ПЛМ) и PAL (ПМЛ) объединяются термином *SPLD*, Simple Programmable Logic Devices (простые программируемые логические устройства).

Появление ПЛМ, ПМЛ и БМК ознаменовало собою начало важнейшего направления развития цифровой компонентной базы, в рамках которого стало *экономически возможным* применение микросхем высокого уровня интеграции и в проектах умеренной тиражности. Разработка БИС/СБИС с программируемой и репрограммируемой структурой оказалась чрезвычайно перспективным направлением и привела к новым эффективным средствам создания специализированных ИС, таким как: *CPLD* (Complex Programmable Logic Devices), *FPGA* (Field Programmable Gate Arrays), *SGA* (Structured Gate Arrays), *SOPC* (System On Programmable Chip) и др.

Целесообразность применения того или иного типа специализированных ИС зависит от конкретных условий, в большой степени от объема выпуска проектируемой аппаратуры. Эти вопросы рассмотрены в *главе 12* и иллюстрируются рис. 12.4. Имеющийся в *главе 12* материал базируется на полученных из предыдущих глав знаниях, что позволяет полнее классифицировать методы специализации ИС. Предварительная классификация с целью пояснения структурирования материалов этой главы, посвященных специализированным ИС, приведена на рис. 8.1.

Все специализированные, т. е. приспосабливаемые к требованием конкретного потребителя, ИС делятся на программируемые пользователем и программируемые изготовителем. В обоих случаях речь идет о программировании структуры, т. е. изменении схемы согласно требованиям проекта. Как уже отмечалось, программирование пользователем или изготовителем существенно меняет характер разработки ЦУ.

Среди программируемых пользователем логических (цифровых) схем выделены простые (SPLD), представляющие первое поколение схем с программируемой структурой в число которых входят микросхемы ПЛМ и ПМЛ, более сложные программируемые логические устройства (CPLD) и программируемые пользователем вентильные матрицы (FPGA), отличающиеся максимальной сложностью структуры и максимальными функциональными возможностями. К микросхемам, программируемым изготовителем (точнее с его участием), относятся полузаказные и заказные ИС. В число полузаказных ИС входят так называемые базовые матричные кристаллы БМК, среди которых выделены стандартные и появившиеся в последнее время структурированные варианты. Полностью заказные схемы отличаются наилучшими техническими характеристиками, но и самым дорогостоящим проектированием. Они делятся на схемы, разрабатываемые методом стандартных ячеек, т. е. с широким использованием готовых библиотечных фрагментов схем, и полностью заказные, проектируемые индивидуально вплоть до транзисторного уровня.

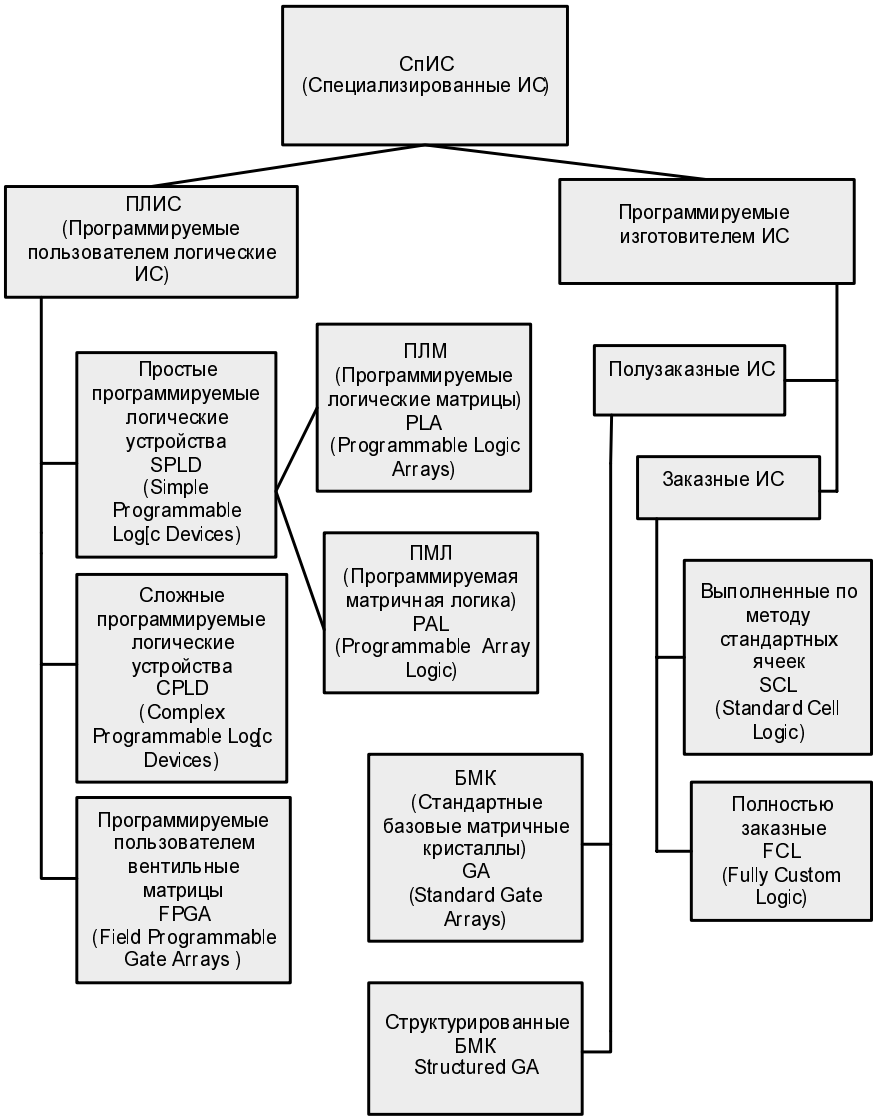


Рис. 8.1. Классификация специализированных интегральных схем

Важным видом БИС/СБИС сейчас стали и *"системы на кристалле"*, которые не указаны в классификации на рис. 8.1, поскольку они в нее непосредственно не встраиваются. К "системам на кристалле" относятся схемы, объединяющие в себе все основные функциональные элементы конечного продукта (процессор, память, аппаратные быстродействующие блоки разного функционального назначения, интерфейсные схемы и т. д.). Разработка "систем на кристалле" требует новых подходов к задачам их проектирования. Что касается средств реализации "систем на кристалле", то они могут быть различными. "Жесткие" системы SoC (Systems on Chip) могут быть реализованы на полностью заказных или полузаказных схемах, про-

граммируемые системы SoPC (Systems on Programmable Chip) в качестве своей основы обычно используют FPGA.

§ 8.2. Программируемые логические матрицы и программируемая матричная логика (ПЛМ и ПМЛ)

Структура ПЛМ

Программируемые логические матрицы появились в середине 70-х годов. Основой их служат последовательно включенные программируемые матрицы (наборы) элементов И и ИЛИ (рис. 8.2, *а*). В их состав входят также блоки входных и выходных буферных каскадов (БВх и БВых).

Примечания

1. Трактовка ПЛМ и ПМЛ как сочетания матриц И и ИЛИ обязана своим происхождением раннему этапу их развития, на котором они выполнялись по биполярной технологии. Позднее (для схем на МОП-транзисторах) в ПЛМ и ПМЛ стали применяться матрицы других логических элементов. Однако представление ПЛМ и ПМЛ в показанном на рис. 8.2 виде остается самым удобным, поскольку схемы, реализованные в булевском базисе, наглядны и легко понимаются. Общность рассмотрения при сохранении булевского базиса не страдает, так как в конечном счете функциональные возможности разных вариантов оказываются идентичными.

2. Термин "матрица" здесь обозначает не более чем набор логических элементов, которые обычно расположены по строкам и столбцам.

Входные буферы БВх, если не выполняют более сложных действий, преобразуют однофазные входные сигналы в парафазные и формируют сигналы необходимой мощности для питания матрицы элементов И ($М_{И}$ или М1). Переменные $x_1...x_m$ подаются через входные буферы на входы элементов И (конъюнкторов), и в матрице И формируются λ термов ($t_1 \ldots t_l$). *Под термом здесь понимается конъюнкция, связывающая входные переменные, представленные в прямой или инверсной форме.* Число формируемых термов равно числу конъюнкторов или, что то же самое, числу выходов матрицы И. Термы подаются далее на входы матрицы ИЛИ ($М_{ИЛИ}$ или М2), т. е. на входы дизъюнкторов, формирующих выходные функции. Число дизъюнкторов равно числу вырабатываемых функций n. Выходные буферы обеспечивают необходимую нагрузочную способность выходов, разрешают или запрещают выход ПЛМ на внешние шины с помощью сигнала OE, а нередко выполняют и более сложные действия, в частности, программируются на передачу сигналов в прямом или инвертированном виде.

В схеме ПЛМ на вентильном уровне (рис. 8.2, *б*) крестиками в пересечениях горизонтальных и вертикальных линий обозначены программируемые точки связей

(ПТС). После программирования в зависимости от наличия или отсутствия соединений в тех или иных ПТС формируются необходимые термы, из которых далее составляются требуемые функции.

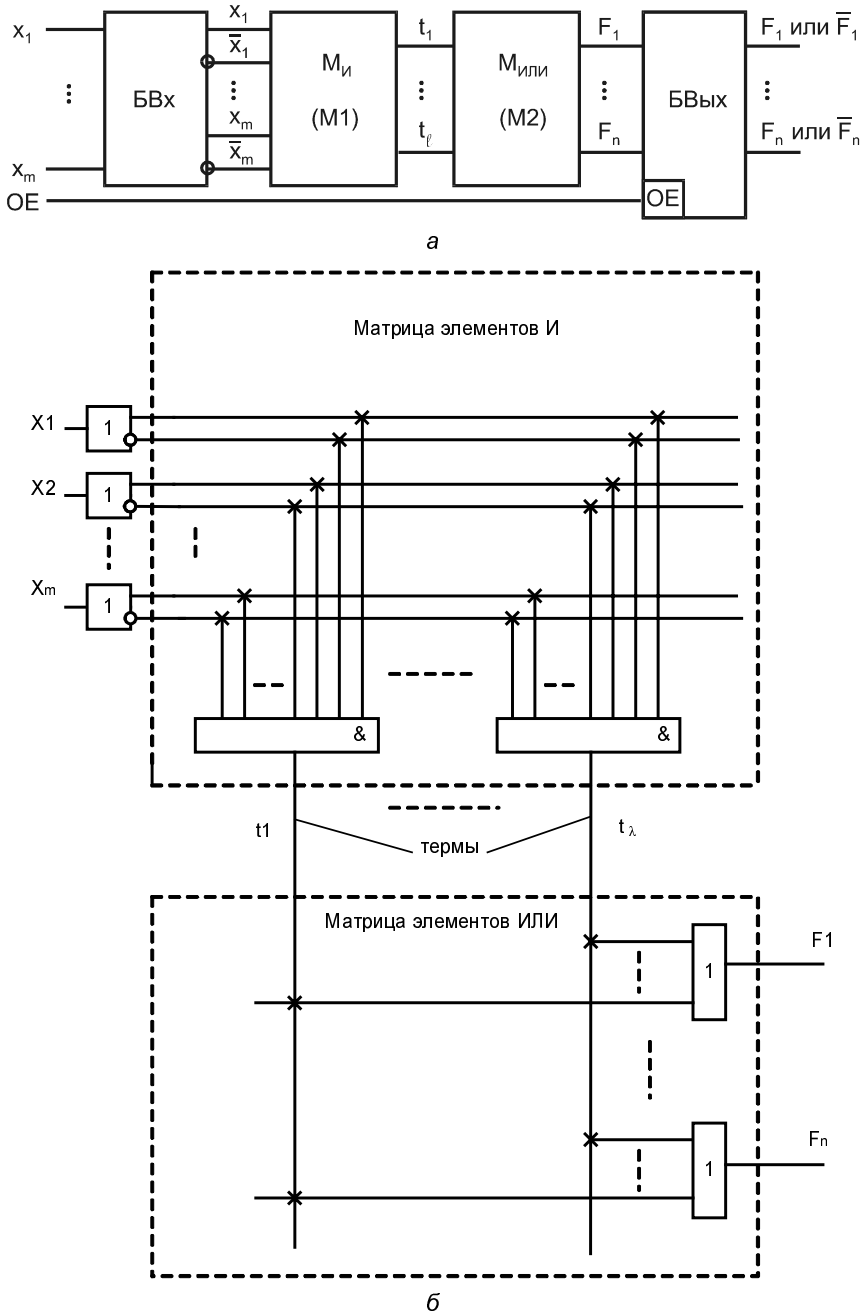


Рис. 8.2. Базовая структура ПЛМ (*а*) и схема ПЛМ на вентильном уровне (*б*)

Таким образом, ПЛМ реализует *дизъюнктивную нормальную форму* (ДНФ) воспроизводимых функций, т. е. представляет их в виде логической суммы логических произведений. *ПЛМ способна реализовать систему n логических функций от m аргументов, содержащую не более λ термов*. Воспроизводимые функции — комбинации из любого числа термов, формируемых матрицей И. Какие именно термы будут выработаны и какие комбинации этих термов составят выходные функции, определяется программированием. Основными параметрами ПЛМ являются:

- число входов m;
- число термов ℓ;
- число функций n.

Упрощенное изображение схем ПЛМ

Схемы ПЛМ достаточно громоздки, поэтому изображать их желательно с максимально возможным упрощением.

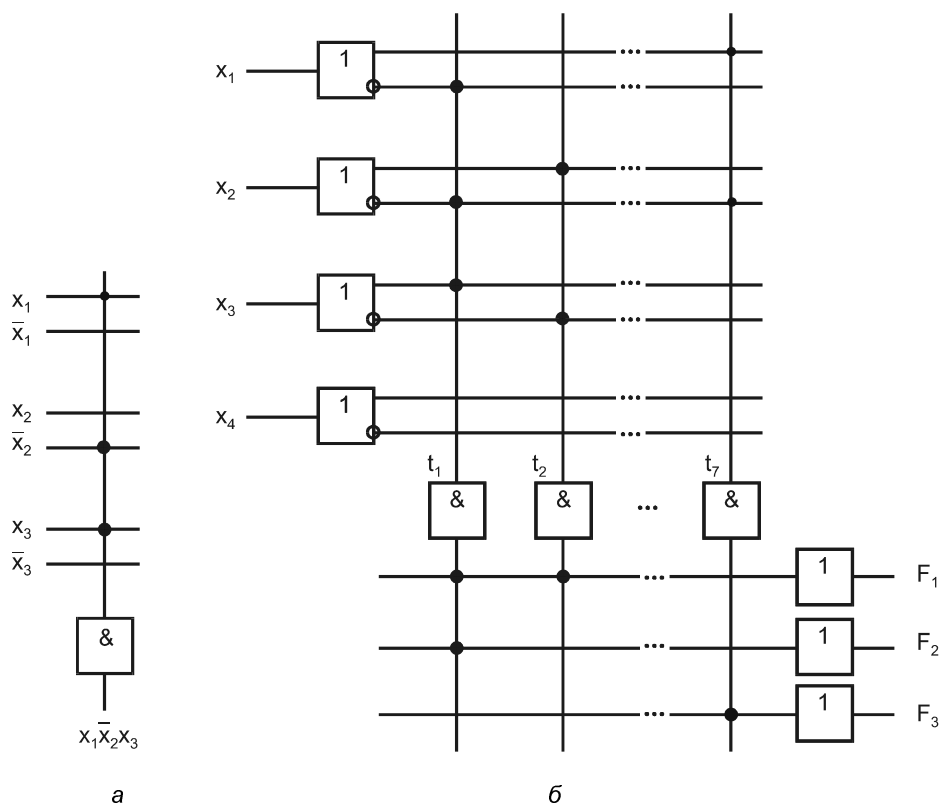


Рис. 8.3. Упрощенное изображение схемы многовходового логического элемента (*а*) и ПЛМ (*б*)

Для этого используются изображения, в которых многовходовые элементы И, ИЛИ условно заменяются одновходовыми.

Единственная линия входа таких элементов пересекается с несколькими линиями входных переменных. Если пересечение отмечено точкой, данная переменная подается на вход изображаемого элемента, если точки нет, то переменная на элемент не подается. Пример многовходового конъюнктора с входами $x_1\bar{x}_2x_3$ показан на рис. 8.3, *а*. Матрицы ПЛМ в упрощенном изображении имеют вид, приведенный на рис. 8.3, *б*. Показан фрагмент для воспроизведения системы функций

$$F_1 = \bar{x}_1\bar{x}_2x_3 \vee x_2\bar{x}_3 \vee x_1\bar{x}_4 = t_1 \vee t_2; t_3,$$

$$F_2 = \bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2x_4 \vee x_2\bar{x}_3x_4 = t_1 \vee t_4 \vee t_5; t_6,$$

$$F_3 = x_1\bar{x}_4 \vee x_1\bar{x}_2 = t_1 \vee t_7$$

размерностью m = 4, ℓ =7, n = 3.

Воспроизведение скобочных форм логических функций

С помощью ПЛМ, как и с помощью рассматриваемых далее ПМЛ, можно воспроизводить не только нормальные дизъюнктивные, но и скобочные формы логических функций, которые могут оказаться более экономичными для реализации. В этом случае сначала получают выражения в скобках, которые затем рассматриваются *как аргументы* для вычисления окончательного результата. В схеме появляются обратные связи — промежуточные результаты с выхода вновь подаются на входы, при этом логическая глубина схемы увеличивается и задержка выработки результата растет.

Например, для получения функции

$$F = x_1x_2 \vee (x_1x_2 \vee \bar{x}_2\bar{x}_1)x_3$$

следует применить включение ПЛМ по схеме (рис. 8.4).

Схемотехника ПЛМ

ПЛМ может быть реализована на основе как биполярной, так и МОП-технологии. Во всех случаях в матрицах имеются горизонтальные и вертикальные линии, в узлах пересечения которых при программировании создаются или удаляются элементы связи. Первые ПЛМ строились на биполярных приборах и имели программируемые элементы в виде прожигаемых (плавких) перемычек. Биполярные ПЛМ выпускались в течение многих лет, в немалой степени потому, что обладали более высоким быстродействием в сравнении с ПЛМ на МОП-транзисторах.

Со временем быстродействие МОП-схем не только сравнялось с быстродействием биполярных, но и превысило его, причем сохранились такие достоинства МОП-схем как пониженное потребление мощности и компактность. В результате в продукции ведущих фирм сейчас представлены почти исключительно ПЛМ и ПМЛ на основе МОП-транзисторов с двумя затворами (плавающим и управляющим). В этом случае программирование матриц состоит в задании плавающим затворам транзисторов того или иного состояния (заряжен или нет).

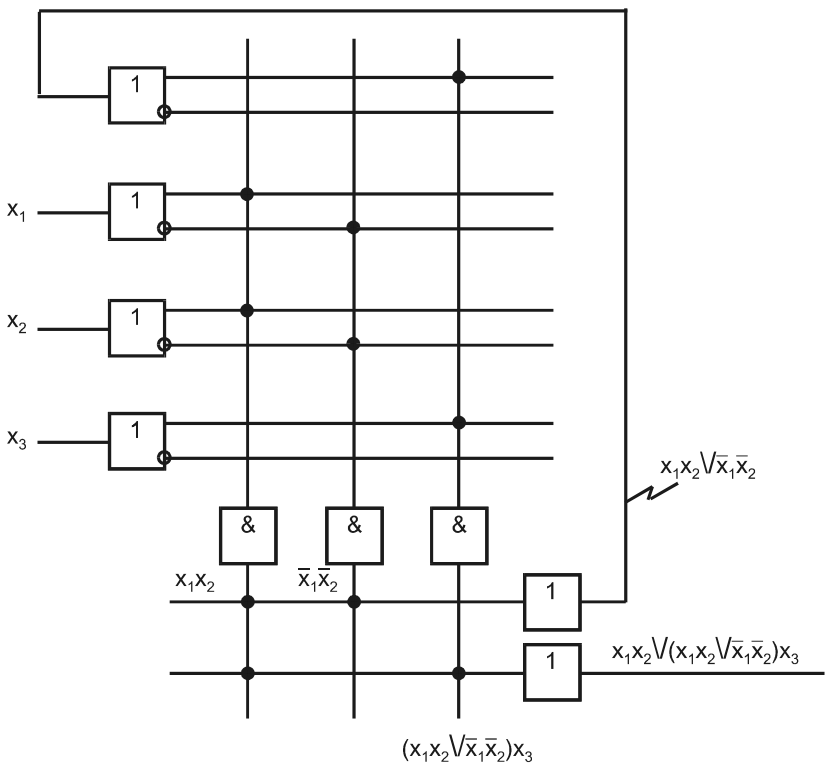


Рис. 8.4. Воспроизведение на ПЛМ скобочных форм переключательных функций

ПЛМ на биполярных приборах, в том числе микросхемы отечественного производства, рассмотрены в предыдущем издании этой книги.

ПЛМ с матрицами элементов ИЛИ-НЕ. В схемах на МОП-транзисторах базовыми являются логические ячейки ИЛИ-НЕ и И-НЕ. Хотя с точки зрения функциональных возможностей они равноценны, многовходовые ячейки ИЛИ-НЕ предпочтительны по быстродействию. В ПЛМ с ячейками ИЛИ-НЕ в обеих программируемых матрицах (рис. 8.5) первая матрица служит для выработки термов, вторая — для выработки выходных функций.

Сравним функциональные характеристики ПЛМ с матрицами И, ИЛИ и матрицами ИЛИ-НЕ. Начнем с получения термов. Пусть, например, требуется получить терм $\bar{a}\bar{b}c$, который будет использован далее в воспроизводимых функциях. Подавая в матрицу элементов ИЛИ-НЕ инвертированные значения переменных $a, b, \bar{c}$, получим нужный терм согласно соотношению $\overline{a \vee b \vee \bar{c}} = \bar{a}\bar{b}c$. Формирование функции из термов будет выполняться согласно формуле $\overline{t_1 \vee t_2 \vee \ldots t_l}$ вместо формулы $t_1 \vee t_2 \vee t_3$ для операции ИЛИ, и отличие от результатов, вырабатываемых ПЛМ с матрицами И и ИЛИ, будет состоять только в инверсии значения функции.

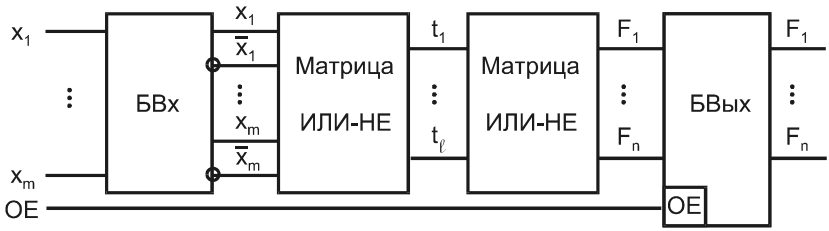


Рис. 8.5. Структура ПЛМ, реализованная на элементах ИЛИ-НЕ

Схема выработки терма $\bar{a}\bar{b}c$ и его введения в воспроизводимую функцию для ПЛМ на элементах ИЛИ-НЕ показана на рис. 8.6. Столбец образует элемент ИЛИ-НЕ выработки терма. Для наглядности заряженные плавающие затворы показаны светлыми прямоугольниками, а разряженные — обычным способом. Транзистор с заряженным электронами плавающим затвором всегда будет запертым, и соответствующий вход будет отключен. Транзисторы с незаряженными затворами функционируют как обычно, и их параллельное соединение, подключенное к цепочке "источник-резистор", образует элемент ИЛИ-НЕ (достаточно хотя бы одного единичного входа, т. е. включенного транзистора, чтобы выходное напряжение приобрело низкий уровень логического нуля).

В строке, тоже представляющей собой элемент ИЛИ-НЕ, вырабатывающий инверсию логической суммы термов, показан один из транзисторов элемента (транзистор, воспринимающий терм первого столбца).

Полученные выражения свидетельствуют о *функциональной равноценности ПЛМ с матрицами И, ИЛИ и ПЛМ с матрицами ИЛИ-НЕ*: если на входы последней подавать аргументы, инвертированные относительно аргументов первой ПЛМ, то на ее выходе получим результаты, отличающиеся от выходов первой ПЛМ только инверсией. Иными словами, ПЛМ с матрицами элементов ИЛИ-НЕ выполняет для инвертированных переменных те же операции, что и ПЛМ с матрицами И и ИЛИ для прямых значений переменных. В дальнейшем работа ПЛМ и ПМЛ иллюстрируется структурами с матрицами И и ИЛИ.

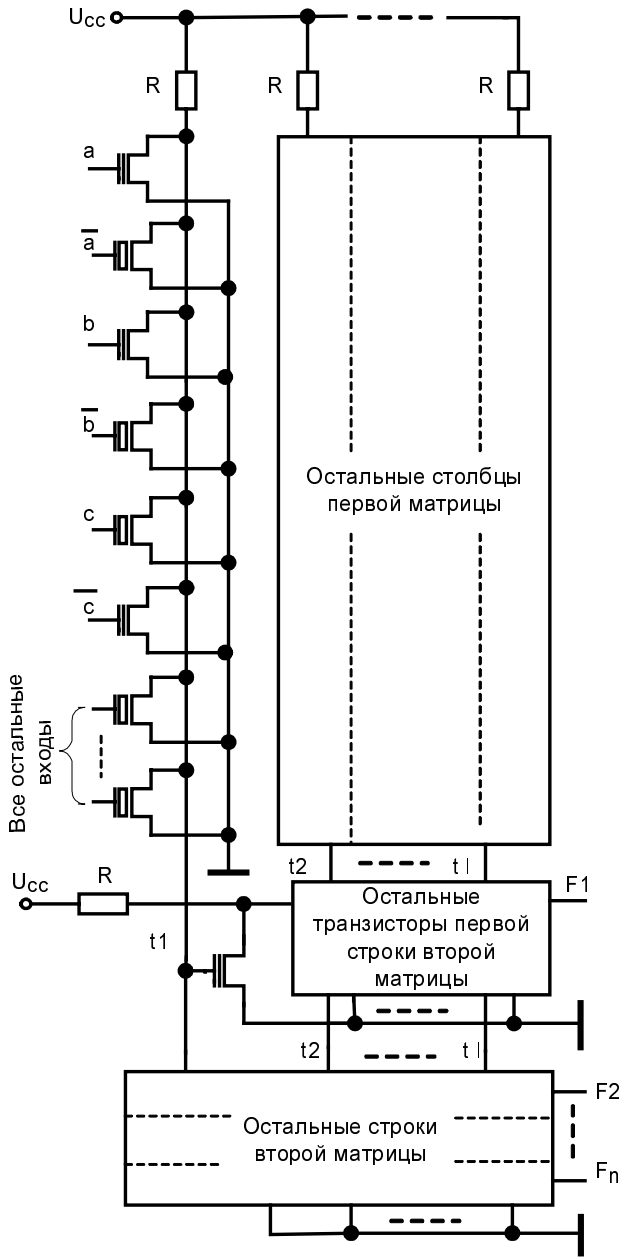


Рис. 8.6. Схемная реализация ПЛМ с матрицами ИЛИ-НЕ

Подготовка задачи к решению на ПЛМ

При проектировании ПЛМ минимизация числа термов ведет к упрощению ее схемы. При работе с готовой (стандартной) ПЛМ следует пытаться уменьшить число

термов в данной системе функций, если оно превышает λ — параметр имеющихся ПЛМ. Если размерность имеющихся ПЛМ обеспечивает решение задачи в ее исходной форме, то минимизация не требуется, т. к. не ведет к сокращению оборудования. Содержание минимизации — поиск *кратчайших дизъюнктивных форм*.

Программирование ПЛМ, выполняемое пользователем, проводится с помощью специальных программаторов и сведения для них должны иметь определенную стандартизированную форму. Для программирования микросхем той или иной фирмы обычно приемлемы программаторы нескольких производителей, перечень которых вместе с необходимыми инструкциями дается в технической документации на ПЛМ или ПМЛ. Отечественные ПЛМ — серия К556 (микросхемы РТ1, РТ2 схемотехнологии ТТЛШ с программированием прожиганием перемычек). Их размерность 16 входов, 48 термов, 8 выходов, задержка около 50 нс. Микросхема РТ1 имеет выходы с открытым коллектором, а РТ2 — с тремя состояниями.

Структура ПМЛ

Одно из важных применений микросхем программируемой логики — замена ИС малого и среднего уровня интеграции при реализации так называемой произвольной логики. В этих применениях логическая мощность ПЛМ зачастую избыточна. Это проявляется, в частности, при воспроизведении типичных для практики систем логических функций, не имеющих больших пересечений друг с другом по термам. В таких случаях возможность использования выходов любых конъюнкторов любыми дизъюнкторами (как предусмотрено в ПЛМ) становится излишним усложнением. Отказ от этой возможности означает отказ от программирования матрицы ИЛИ и приводит к структуре ПМЛ.

В ПМЛ (рис. 8.7) выходы элементов И (выходы первой матрицы) жестко распределены между элементами ИЛИ (входами матрицы ИЛИ). В показанной ПМЛ m входов, n выходов и 4n элементов И, поскольку каждому элементу ИЛИ придается по четыре конъюнктора. ПМЛ, как и ПЛМ, воспроизводят дизъюнктивные нормальные формы логических функций, но с более жесткими ограничениями.

В сравнении с ПЛМ схемы ПМЛ имеют меньшую функциональную гибкость, т. к. в них матрица ИЛИ фиксирована, но их изготовление и использование проще. Преимущества ПМЛ особенно проявляются при проектировании несложных устройств. Продукция класса SPLD, выпускаемая несколькими крупными фирмами, почти исключительно представлена микросхемами ПМЛ (PAL, GAL). Структуры ПЛМ (PLA) типичны для схем формирования управляющих сигналов в машинных циклах микропроцессорных систем и других подобных применений, где они проектируются как часть систем, а не являются автономными микросхемами.

Подготовка задач к решению на ПМЛ имеет много общего с подходом к решению задач на ПЛМ, но есть и различия. Для задач, решаемых на ПМЛ, важно уменьшить число элементов И в каждом выходном канале, но если для ПЛМ стремятся искать представления функций с наибольшим числом общих термов, то для ПМЛ

это не требуется, поскольку элементы И одного выхода не могут быть использованы другими выходами (т. е. для других функций).

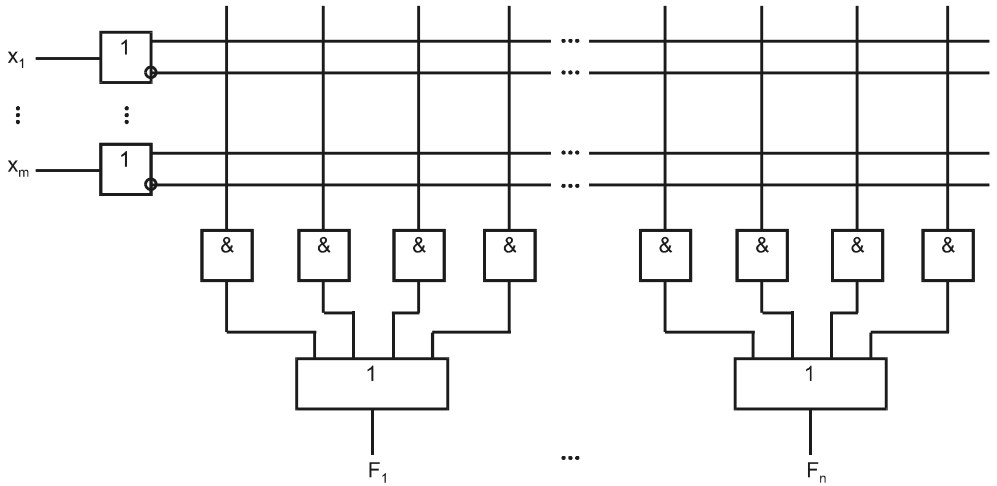


Рис. 8.7. Базовая структура ПМЛ

Наряду с классическими ПМЛ разработаны и применяются такие, структуры которых в некоторой мере приближаются к ПЛМ, поскольку жесткость распределения термов по отдельным функциям в них снижена, например, для соседних каналов выработки функций имеются возможности "делиться" друг с другом термами. Подобные схемы рассмотрены далее.

Обогащение функциональных возможностей ПЛМ и ПМЛ

Рассмотренные структуры ПЛМ и ПМЛ — базовые, с них началось развитие этих направлений. В дальнейшем происходило обогащение функциональных возможностей ПЛМ и ПМЛ с помощью ряда приемов, в первую очередь следующих.

Программирование выходных буферов

В схемах с программируемым выходным буфером возможна выдача функций в прямом или инверсном виде. В такой схеме (рис. 8.8) выработанные матрицами функции $F_1^*...F_n^*$ проходят через выходные буферы, содержащие сумматоры по модулю 2 (М2).

В показанной на рисунке схеме вторые входы сумматоров получают сигналы, величина которых зависит от состояний транзисторов с плавающим затвором. Если в плавающий затвор введен заряд электронов, транзистор заперт, и второй вход получает сигнал логической единицы от источника питания через резистор R. Складываясь по

модулю 2 с единицей, функции F_i^* инвертируются. Если в плавающем затворе заряда нет, то транзистор открыт, и его малое сопротивление делает напряжение на втором входе сумматора по модулю 2 низким, отображающим логический нуль. При этом $F_i^* = F_i$ и функции с выхода матриц передаются через буфер без изменений.

Программируемый буфер дает дополнительные возможности для минимизации числа термов в реализуемой системе. В исходной системе можно заменять функции их инверсиями, если это приводит к уменьшению числа термов. Никаких последствий в смысле введения дополнительных схем это не вызовет — возврат к исходной системе будет обеспечен просто программированием буфера.

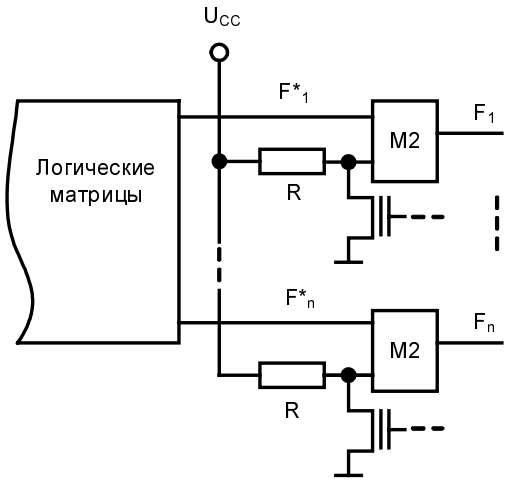


Рис. 8.8. Схема программируемого выходного буфера

Пример

Пусть нужно воспроизвести систему из двух функций:

$$F_1 = \bar{x}_3\bar{x}_2\bar{x}_0 \vee \bar{x}_3x_2x_0 \vee x_3\bar{x}_1x_0 \vee x_3x_1\bar{x}_0;$$
$$F_2 = \bar{x}_3\bar{x}_2x_0 \vee \bar{x}_3x_2\bar{x}_1 \vee x_3\bar{x}_1\bar{x}_0 \vee x_3x_1x_0.$$

Карты Карно для этих функций (рис. 8.9) показывают контуры, соответствующие 8 различным термам системы:

$$t_1 = \bar{x}_3\bar{x}_2\bar{x}_0,\ t_2 = \bar{x}_3x_2x_0,\ t_3 = x_3\bar{x}_1x_0,\ t_4 = x_3x_1\bar{x}_0;$$
$$t_5 = \bar{x}_3\bar{x}_2x_0,\ t_6 = \bar{x}_3x_2\bar{x}_1,\ t_7 = x_3\bar{x}_1\bar{x}_0,\ t_8 = x_3x_1x_0.$$

При инвертировании функции единицы занимают в карте Карно те позиции, которые были нулями. Видно, что при инвертировании одной из функций получим карты Карно с меньшим суммарным количеством различных термов. При инвертировании, например, функции F_2 получим карту с контурами, показанными штриховыми линиями, и систему функций:

$$F_1 = \bar{x}_3\bar{x}_2\bar{x}_0 \vee \bar{x}_3x_2x_0 \vee x_3\bar{x}_1x_0 \vee x_3x_1\bar{x}_0 = t_1 \vee t_2 \vee t_3 \vee t_4,$$
$$\overline{F}_2 = \bar{x}_3\bar{x}_2\bar{x}_0 \vee \bar{x}_3x_2x_1 \vee x_3\bar{x}_1x_0 \vee x_3x_1\bar{x}_0 = t_1 \vee t_5 \vee t_3 \vee t_4,$$

в которой всего пять различных термов. Возврат от функции $\overline{F}_2$ к функции F_2 осуществляется введением заряда в плавающий затвор транзистора в линии F_2.

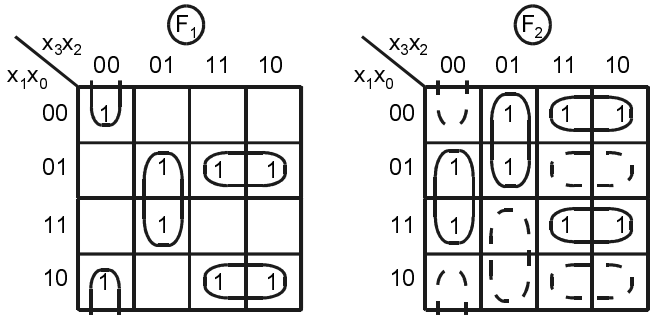


Рис. 8.9. Карты Карно для примера воспроизведения функций в ПЛМ с программируемым выходным буфером

Применение двунаправленных выводов

Используя элементы с тремя состояниями, можно приспосабливать выводы для работы в качестве входов или выходов в зависимости от их программирования. В такой схеме один из конъюнкторов предназначен для управления элементом с тремя состояниями выхода (рис. 8.10). Выход элемента одновременно связан с матрицей И как вход.

Возможны четыре режима вывода Вх/Вых в зависимости от того, как запрограммированы входы конъюнктора К:

1. На входы конъюнктора К хотя бы одна переменная подана одновременно в прямом и инверсном виде (и x_i и $\overline{x}_i$). В этом режиме на выходе конъюнктора будет нуль, буфер имеет третье состояние (отключен) и вывод функционирует как вход.
2. Все переменные отключены, на выходе конъюнктора единица, буфер активен, вывод работает как простой выход (его сигналы не используются в матрице И).
3. Выход с обратной связью. Этот режим отличается от предыдущего только тем, что сигналы вывода используются в матрице И.
4. Управляемый выход. Здесь входы конъюнктора программируются. При заданной комбинации входных сигналов конъюнктор приобретает единичный выход, и вывод срабатывает как выход.

В схеме с некоторым числом двунаправленных выводов можно изменять соотношение числа входов-выходов. Если число входов равно m, число выходов n и число двусторонних выводов p, то можно иметь число входов от m до m + p и число выходов от n до n + p при условии, что сумма числа входов и выходов не превосходит m + n + p.

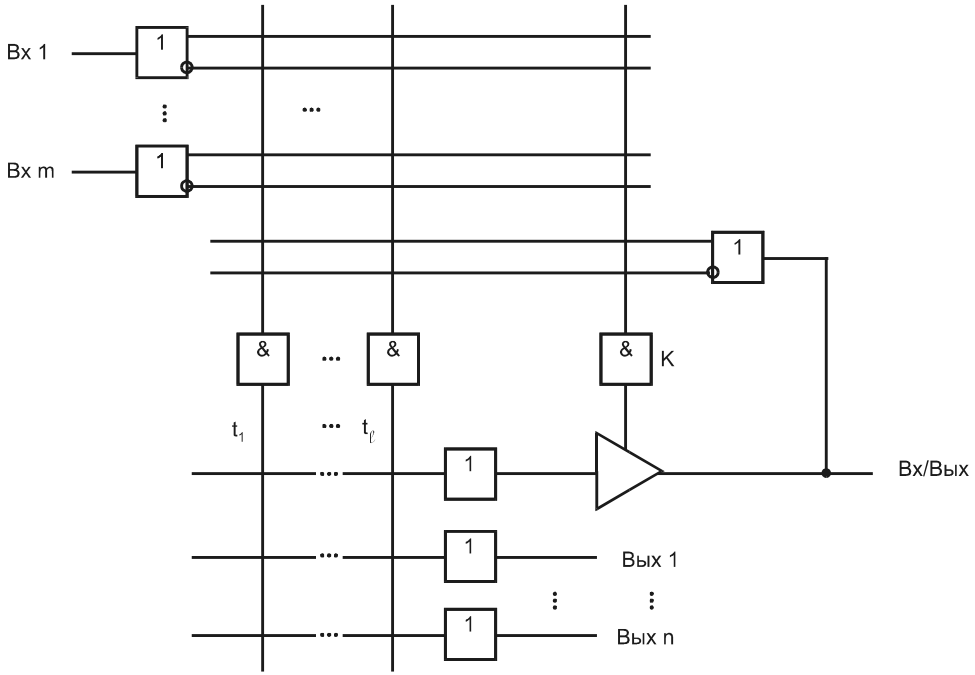


Рис. 8.10. Схема с двунаправленным буфером

Введение элементов памяти

Для построения автоматов нужны элементы памяти (триггеры). Поскольку триггер составлен из логических ячеек, образующих схему с обратными связями, автомат можно реализовать и на ПЛМ или ПМЛ чисто логического типа. Однако схемы, содержащие готовые триггеры, более удобны для создания автоматов. Поэтому появились ПЛМ и ПМЛ, которые помимо комбинационной части содержат на кристалле триггеры (регистры), обычно типа D. Включение триггера в выходную цепь ПЛМ или ПМЛ показано на рис. 8.11.

Триггер D включается в выходную цепь матрицы ИЛИ. Через управляемый буфер с третьим состоянием выход триггера соединен с выходом соответствующего канала ПЛМ или ПМЛ. Выходы триггера подаются обратно в матрицу И в виде парафазных сигналов. При показанном включении триггера реализуется типовое функционирование элемента памяти в схеме автомата. Функция возбуждения триггера вырабатывается логическими ресурсами комбинационной части ПЛМ или ПМЛ, а выход триггера подается в эту часть, поскольку функции возбуждения триггеров зависят не только от входных переменных, но и от внутреннего состояния автомата. Сигнал OE разрешает или запрещает передачу выходного сигнала триггера на выход канала ПЛМ или ПМЛ.

ПЛМ с памятью характеризуется четырьмя параметрами. Кроме трех обычных параметров, она имеет и параметр r — число элементов памяти. Максимальное число

внутренних состояний автомата 2^r. Автомат рассматривается как синхронный — петля обратной связи активизируется только по разрешению тактовых сигналов.

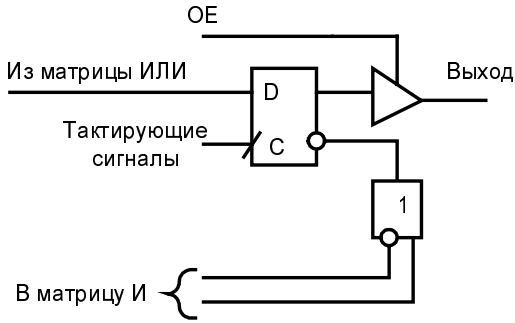


Рис. 8.11. Включение элементов памяти в схемы ПЛМ или ПМЛ

Использование разделяемых конъюнкторов в схемах ПМЛ

Наряду с пригодными как для ПЛМ, так и для ПМЛ методами обогащения функциональных возможностей, рассмотренными ранее, существуют и специфические модификации, относящиеся только к ПМЛ. К ним относится вариант с так называемыми *разделяемыми конъюнкторами*. "Разделяемость" здесь означает *"возможность использования как одним, так и другим каналом"*, при которой одни и те же конъюнкторы могут быть отданы тому или иному выходу схемы. В таких схемах сложные функции с большим числом термов могут воспроизводиться за счет заимствования термов от каналов выработки простых функций, где имеются избыточные термы.

Вариант с разделяемыми конъюнкторами смягчает наиболее очевидное ограничение функциональных возможностей простых (жестких) ПМЛ — фиксированное число элементов И на входах элементов ИЛИ, которого может не хватить при воспроизведении сложных функций. Имея ПМЛ с разделяемыми конъюнкторами и размещая сложную функцию рядом с простой, можно позаимствовать часть общего набора конъюнкторов у простой функции в пользу сложной. Полного программирования матрицы ИЛИ здесь не возникает, но все же эта модификация является *шагом в направлении к ПЛМ.* Реализация разделяемости конъюнкторов иллюстрируется на рис. 8.12.

Единая матрица И принимает входные переменные и сигналы обратной связи. Конъюнкторы вырабатывают термы, которые подаются на элементы ИЛИ (по четыре терма на каждый из двух элементов ИЛИ). Столбец из четырех программируемых мультиплексоров реализует разделяемость термов, позволяя данному макроэлементу использовать не только термы от своих конъюнкторов, но и получать термы от соседних каналов (при программировании мультиплексоров 1 и 4 на передачу данных от верхних входов) и отдавать свои термы соседям с выходов дизъюнкторов 1 и 2. Программирование мультиплексора на передачу данных от нижне-

го входа исключает поступающие на него термы из формируемого набора. Окончательный набор термов формируется дизъюнктором 3, на входы которого поступают выходные сигналы мультиплексоров. Выработанная функция F может содержать до 16 термов.

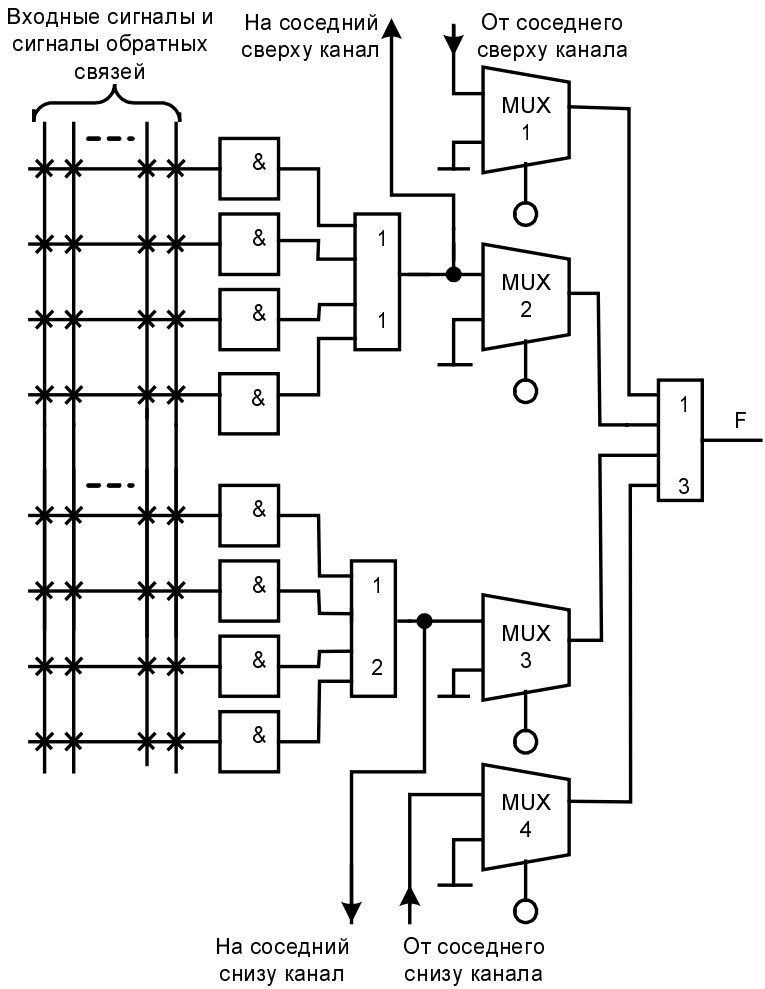


Рис. 8.12. Разделение термов в ПМЛ

Примеры отечественных ПМЛ

В серию КР1556 отечественных ПМЛ входят микросхемы ХЛ8, ХП4, ХП6, ХП8. Буквой Л отмечаются ПМЛ чисто логического типа (без элементов памяти), а буквой П — ПМЛ с триггерами.

ПМЛ без элементов памяти

В микросхеме ХЛ8 (рис. 8.13) число входов может изменяться от 10 (входы, показанные с левой стороны матрицы) до 16, если все двунаправленные выводы В2...В7 запрограммированы как входы. Число выходов изменяется от 2 до 8. Суммарное число входов и выходов не может превышать 18.

Выходные буферы ПМЛ получают разрешение или запрещение работы от матрицы И. Набор элементов ИЛИ состоит из 8 элементов с семью входами, т. е. на каждый элемент ИЛИ приходится по 7 конъюнкторов с числом входов от 10 до 16. Исходя из сказанного, можно оценить и размерность матрицы И, содержащей 2048 узлов (64×32).

ПМЛ с элементами памяти

В микросхемах серии К1556 типа ХП имеются элементы памяти — триггеры типа D, число которых совпадает с цифрой в обозначении ИС (4, 6 или 8).

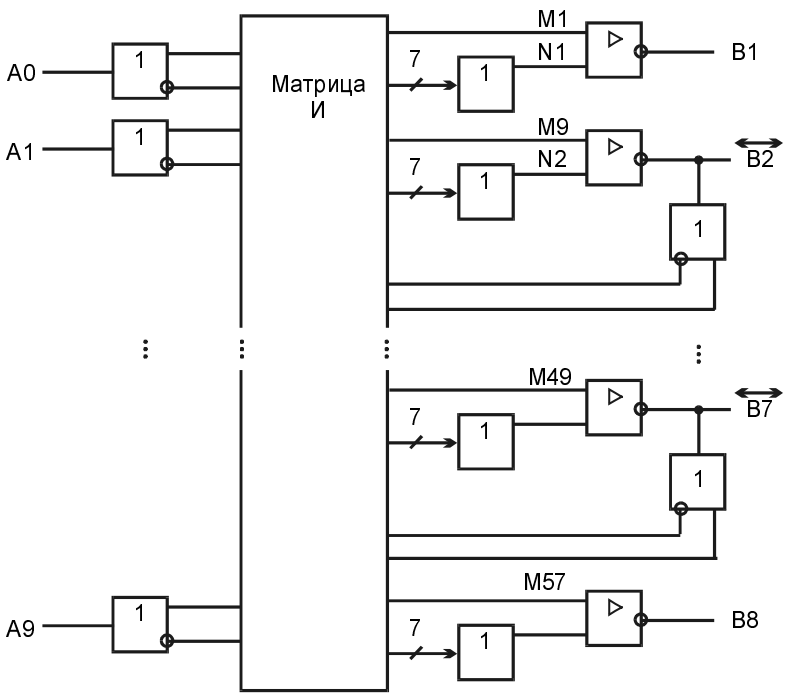


Рис. 8.13. Структура ПМЛ КР1556ХЛ8

Структура ИС ХП4 (рис. 8.14) имеет первый уровень логики, на котором образуются термы входных переменных, второй уровень — матрица ИЛИ, состоящая из 8 дизъюнкторов (четырех 7-входовых и четырех 8-входовых). Выходные усилители выполнены по схеме с тремя состояниями. Четыре D-триггера имеют управление от поло-

жительного фронта внешнего синхросигнала C. Сигнал OE управляет буферами, подключенными к выходам триггеров.

Число входов у ПМЛ типа ХП — восемь, число выходов 8(4), 8(2) и 8 для ХП4, ХП6 и ХП8 соответственно (в скобках указано число выводов, не содержащих триггеров), задержка между выводами вход-выход не более 40 нс, а между тактовым сигналом и выходом не более 25 нс. Потребление тока — 180 мА.

Пример подготовки задачи к решению с помощью ПМЛ

Пусть на ПМЛ КР1556ХП4 требуется реализовать 4-разрядный синхронный счетчик, выполняющий помимо счета также операцию параллельной асинхронной загрузки.

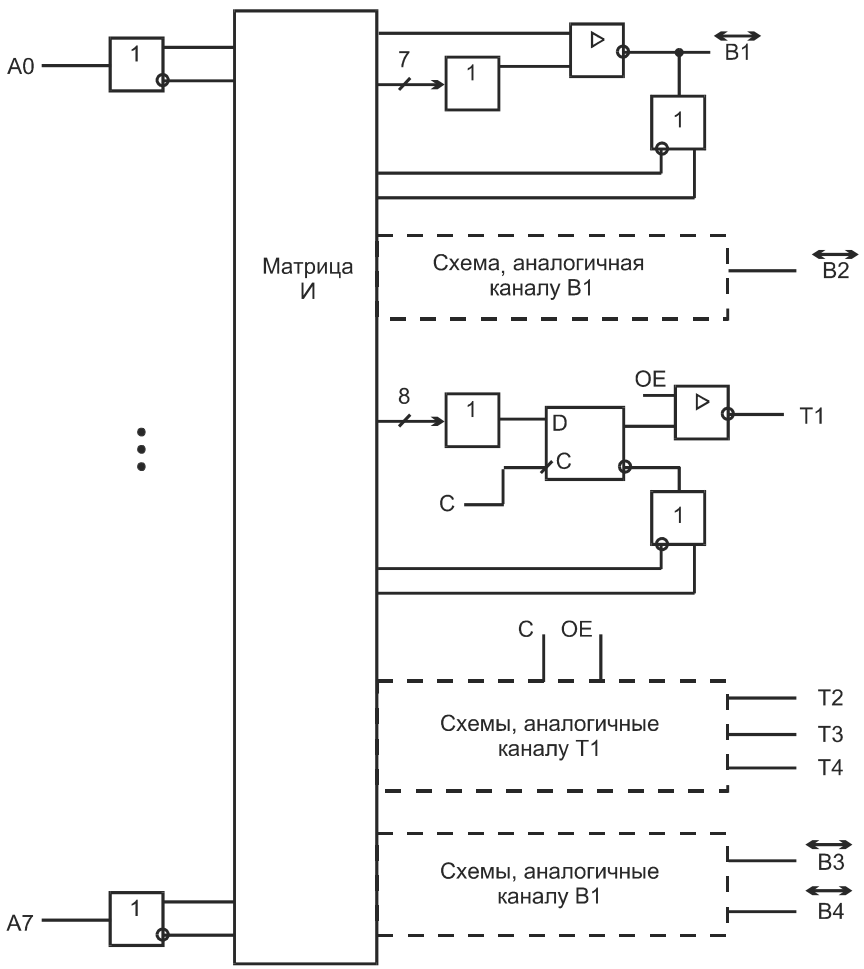


Рис. 8.14. Структура ПМЛ КР1556ХП4

Для реализации счетчика нужно определить функции возбуждения триггеров всех его разрядов. В нашем распоряжении имеются триггеры типа D, для которых функции возбуждения совпадают с состояниями, которые должны принять триггеры в следующем такте. Найдем выражения для этих состояний.

Обозначим выходы разрядов счетчика, начиная с младшего, через Q_0, Q_1, Q_2, Q_3. Сигнал асинхронной загрузки обозначим как LE (Load Enable). Загружаемое слово — $A_3A_2A_1A_0$.

Триггер младшего разряда счетчика переключается от каждого входного сигнала при отсутствии сигнала загрузки и принимает значение A_0 при загрузке. Следовательно, для его выхода в новом состоянии можно записать

$$Q_{0H} = \overline{LE}\ \overline{Q}_0 \vee LE\ A_0,$$

где первое слагаемое отображает процесс переключения триггера, а второе — параллельную загрузку.

Следующий разряд переключается только при условиях отсутствия сигнала загрузки и единичном состоянии триггера младшего разряда. При $Q_0 = 0$ этот триггер сохраняет свое состояние. Для его выхода можно записать:

$$Q_{1H} = \overline{LE}\ \overline{Q}_1 Q_0 \vee \overline{LE}\ Q_1 \overline{Q}_0 \vee LE\ A_1,$$

где первое слагаемое отображает переключение триггера, второе — сохранение его состояния при $Q_0 = 0$, третье — загрузку.

Продолжая аналогичные рассуждения, для последующих разрядов счетчика можно получить соотношения:

$$Q_{2H} = \overline{LE}\ \overline{Q}_2 Q_1 Q_0 \vee \overline{LE}\ Q_2 \overline{Q}_1 \vee \overline{LE}\ Q_2 \overline{Q}_0 \vee LE; A_2,$$
$$Q_{3H} = \overline{LE}\ \overline{Q}_3 Q_2 Q_1 Q_0 \vee \overline{LE}\ Q_3 \overline{Q}_2 \vee \overline{LE}\ Q_3 \overline{Q}_1 \vee \overline{LE}\ Q_3 \overline{Q}_0 \vee LE\ A_3,$$

где первые слагаемые отображают процесс переключения разряда, последние — параллельную загрузку, а промежуточные — сохранение состояния при отсутствии условий переключения.

Поскольку искомые функции содержат не более пяти конъюнкций, то возможна их непосредственная реализация на микросхеме ХП4 (в этой микросхеме число элементов И на входах элементов ИЛИ составляет 7 или 8 для разных выходов).

При проектировании устройств на основе ПЛМ и ПМЛ пользуются подсистемами автоматизации проектирования, т. к. ручная подготовка задачи как правило неприемлемо громоздка. Для подсистемы автоматизированного проектирования подготовка данных проводится с использованием стандартных средств (таблиц, систем булевых уравнений, текстовых описаний на языках описания аппаратуры). В подобных подсистемах имеются также режимы входного контроля ИС, ввода данных с эталона, т. е. уже запрограммированной ИС, установленной в специальную соединительную розетку, а также режимы сравнения задания на программирование с состоянием запрограммированной ИС и др.

ПМЛ типа PAL 22V10

Пример популярной ПМЛ — микросхема PAL 22V10 (V от Versatile — гибкий).

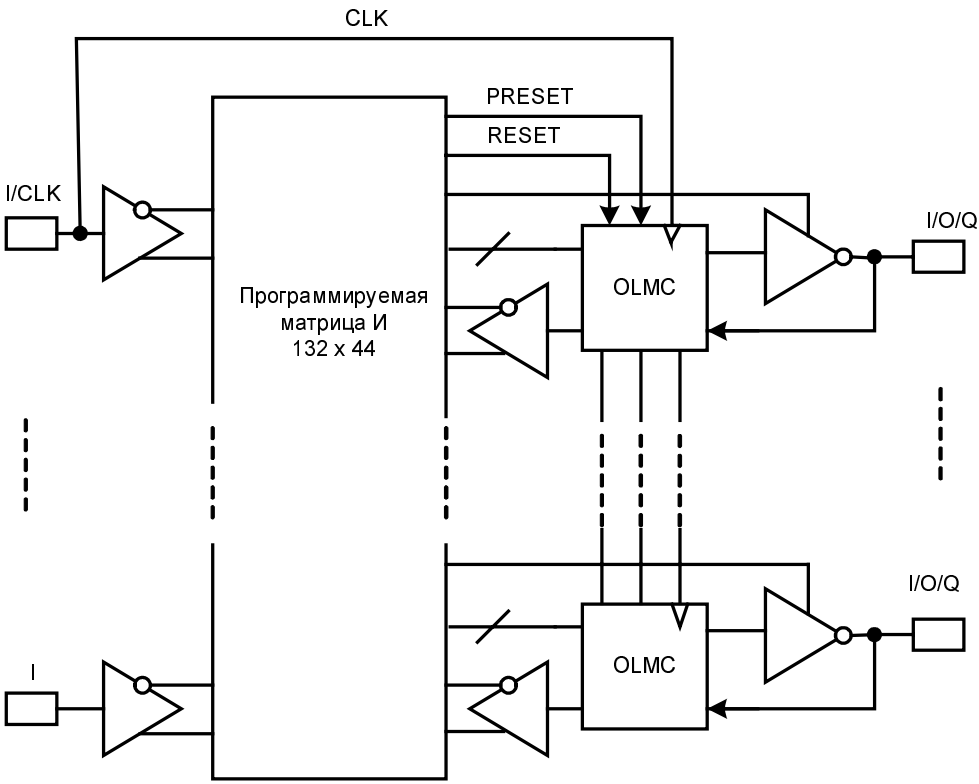


Рис. 8.15. Структура микросхемы PAL22V10

У этой микросхемы (рис. 8.15) 11 входов и 10 выходов. Выходы различаются числом подключенных к ним конъюнкторов. Разные выходы имеют от 8 до 16 конъюнкторов (в порядке расположения выходы имеют 8-10-12-14-16-16-14-12-10-8 термов). Выходные величины вырабатываются не просто дизъюнкторами, а более сложными схемами, называемыми *макроэлементами* или *макроячейками* (*OLMC*, Output Logic Macrocell). Кроме термов для формирования логических функций макроячейки получают дополнительно терм управления третьим состоянием выходного буфера. Сигналы сброса (RESET), предустановки (PRESET) и тактирования (CLK) являются общими для всех макроячеек.

Макроячейка (рис. 8.16) содержит D-триггер с цепями тактирования, асинхронного сброса AR и синхронной установки SP. Сигналы AR и SP вырабатываются специальными термами матрицы И. Мультиплексор "4—1" работает на выходной буфер, мультиплексор "2—1" передает сигналы обратной связи в матрицу И. Цепи с транзисторами, имеющими плавающие затворы, программируют мультип-

лексоры, задавая им сигналы на адресных входах S1 и S0. На вход мультиплексора "4—1" подаются прямой и инверсный сигналы от логической части ПМЛ (т. е. значения выработанной функции), а также выход с триггера (регистровый) и его инверсия. Сигнал обратной связи можно взять с выхода ПМЛ или с выхода триггера. При установке выходного буфера в третье состояние внешний вывод может быть использован как вход. Сигналы разрешения для буферов вырабатываются индивидуальными термами для каждой макроячейки и, следовательно, могут задаваться не только как "разрешено" или "запрещено", но и определяться логическими формулами.

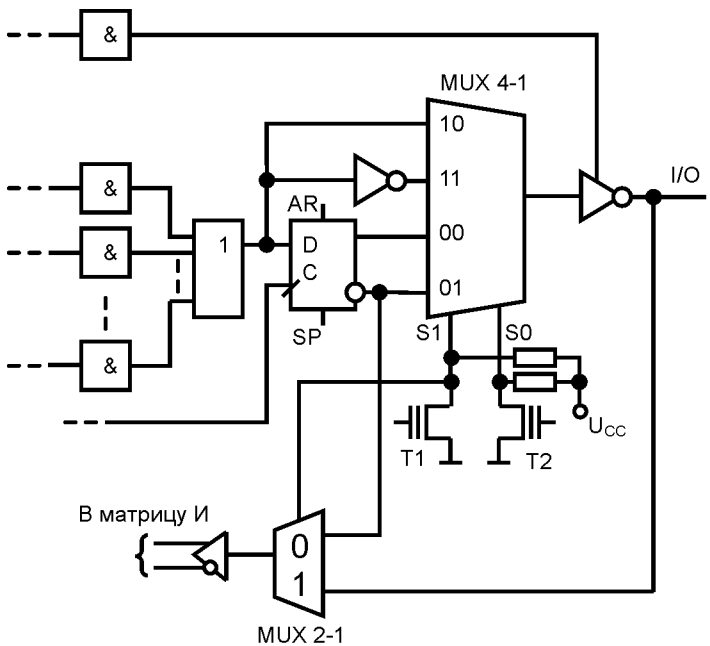


Рис. 8.16. Схема макроячейки ПМЛ PAL22V10

В зависимости от состояний программируемых транзисторов, задающих для мультиплексоров значения адресных сигналов S1 и S0, реализуются режимы комбинационного и регистрового выходов с прямой или инверсной передачей данных при различных видах обратной связи или режимы входа. Схемы режимов макроячейки при всех сочетаниях величин S1 и S0 приведены на рис. 8.17.

Микросхемы PAL 22V10 снабжены некоторыми полезными для их применения свойствами, в частности, следующими. Имеется возможность засекретить содержимое функциональных блоков, предотвратив их чтение. Команда засекречивания может быть стерта только операцией репрограммирования, так что засекреченная конфигурация схемы никак не может быть выявлена, если была активизирована ячейка засекречивания (Security Cell). Имеется также возможность требуемой загрузки триггеров выходных цепей. При тестировании автомата с памятью должны

быть проверены все его состояния, а не только состояния рабочих циклов, поскольку при включении питания или из-за помех автомат может попадать и в непредусмотренные состояния. Для проверки таких состояний должна быть предусмотрена возможность разрыва цепей обратных связей и принудительного создания любого состояния элементов памяти автомата. Затем схема автомата восстанавливается и его выходы тестируются для исследования последующих состояний. Микросхема PAL 22V10 имеет средства для синхронной установки каждого триггера в единичное или нулевое состояние, обеспечивая тем самым принудительное создание любого состояния автомата для его тестирования.

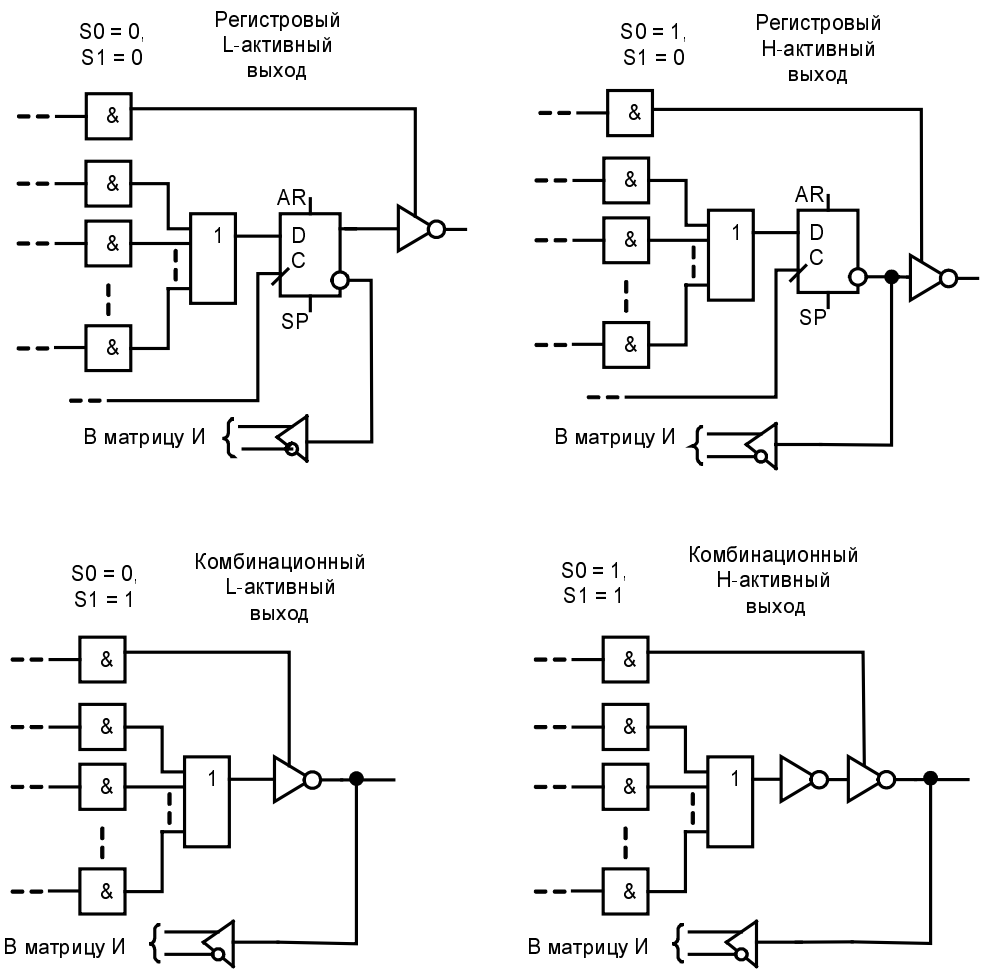


Рис. 8.17. Режимы макроячейки ПМЛ PAL 22V10

При включении питания все триггеры микросхемы получают сигнал сброса и приходят в нулевое состояние. Таким образом, регистровые выходы окажутся в определенных состояниях (единичных или нулевых) в зависимости от запрограммиро-

ванной полярности регистровых выходов. Это обстоятельство также облегчает разработку автоматов с памятью. Тактирование схем не начинается, пока не пройдет время предустановки для всех входных сигналов и сигналов обратных связей.

§ 8.3. CPLD — сложные программируемые логические устройства

Структура CPLD

CPLD (Complex Programmable Logic Device) представляет собой несколько функциональных блоков — ФБ, объединенных в единую структуру программируемой матрицей соединений — ПМС (*PIA*, Programmable Interconnect Array) (рис. 8.18). Каждый ФБ подобен ПМЛ (PAL) и содержит несколько *макроячеек* — МЯ. С внешней средой CPLD связывают блоки ввода/вывода — БВВ. Кроме основных блоков CPLD может содержать контроллеры интерфейсов JTAG и ISP, используемые для конфигурирования и тестирования создаваемых структур, и другие блоки.

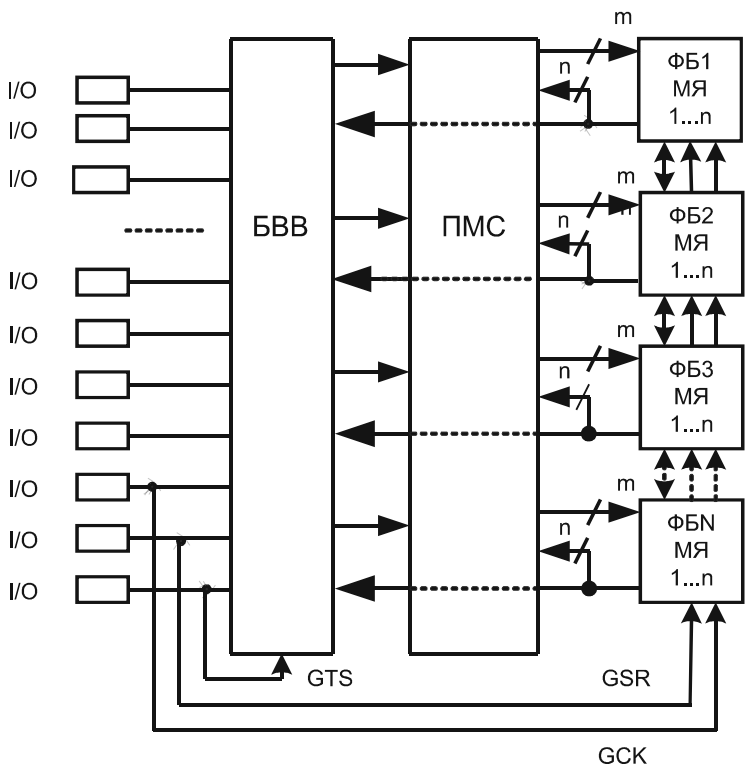


Рис. 8.18. Архитектура CPLD

Число ФБ в составе CPLD изменяется в зависимости от ее сложности. Каждый ФБ получает *m* сигналов от ПМС, а *n* его выходов подключены как к ПМС, так и к блокам ввода/вывода БВВ (Input/Output Block, IOB), связанным с внешними двунаправленными выводами. Три вывода специализированы и предназначены для глобальных сигналов тактирования GCK (Global Clock), сброса/установки GSR (Global Set/Reset), управления третьим состоянием GTS (Global Tri-State). Возможно и иное использование специализированных выводов, если они не применяются по назначению.

> ***Примечание***
>
> Здесь и далее термин "глобальный" применяется для сигналов, общих для всей схемы.

Число контактов ввода/вывода может быть меньше числа выводов всех ФБ. В этом случае часть макроячеек может быть использована только для выработки внутренних сигналов (сигналов обратных связей), потребность в которых типична для многих видов устройств.

Функциональные блоки CPLD

Функциональные блоки CPLD (рис. 8.19) подобны ПМЛ (PAL) и содержат:

- многовходовую программируемую матрицу элементов И ($М_И$), вырабатывающую конъюнктивные термы из поступающих на ее входы переменных $x_1 \ldots x_m$;
- матрицы распределения термов МРТ;
- группу из N макроячеек, между которыми с помощью МРТ распределяются выработанные матрицей $М_И$ термы.

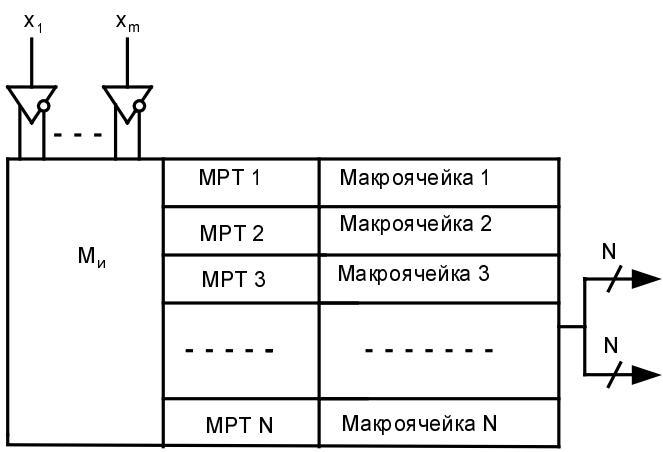


Рис. 8.19. Обобщенная структура функционального блока CPLD

Функциональные блоки реализуют двухуровневую логику типа ДНФ с вариантами формируемых выходных сигналов (прямой или инверсный, комбинационный или регистровый).

В классических ПМЛ термы жестко распределяются между макроячейками, формирующими выходные функции. Матрицы распределения термов МРТ позволяют варьировать число термов в вырабатываемой макроячейкой функции F_i. При этом термы заимствуются у каналов выработки других функций или отдаются им. Если термы используются не только дизъюнкторами формирования выходных функций, но и другими элементами (например, для управления триггерами, входящими в состав макроячеек), то и для них МРТ играет роль "раздатчика термов".

Логические расширители

Для обогащения выходных функций макроячеек дополнительными термами служат *параллельные и последовательные логические расширители* (рис. 8.20).

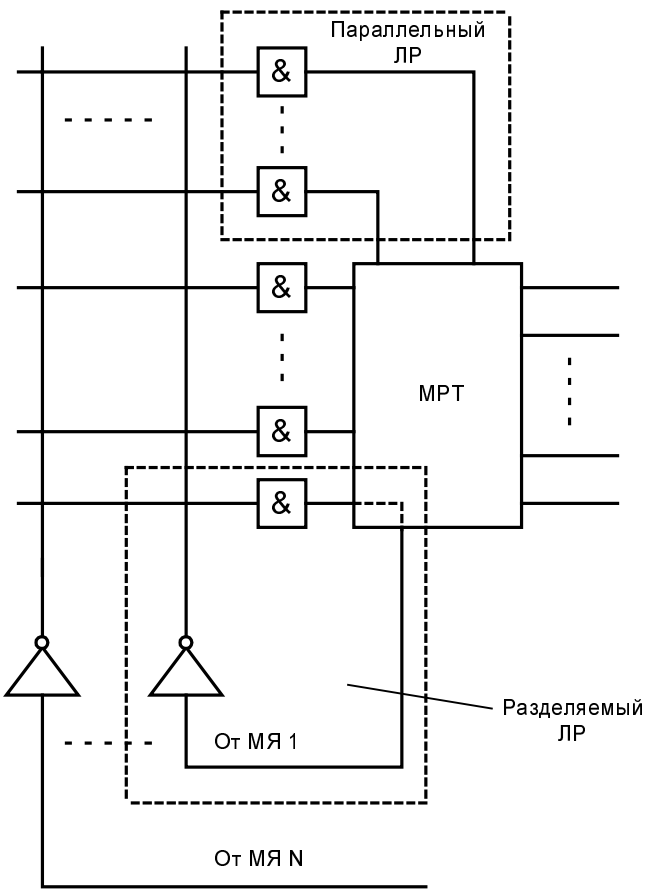


Рис. 8.20. Логические расширители параллельного и последовательного типов

Последовательные (разделяемые, общие) логические расширители создаются подачей инвертированного значения терма из МРТ данного канала обратно на один из входов матрицы $М_и$. Переданный в матрицу $М_и$ терм становится доступным для использования во всех каналах данного ФБ. Если, например, это терм $q = x_1\bar{x}_2x_5\bar{x}_{10}$, т. е. его инверсия $\bar{q} = \bar{x}_1 \vee x_2 \vee \bar{x}_5 \vee x_{10}$, то в том канале, где он будет использован вместе с входными термами канала q1, q2, q3, будет получена функция

$$F = q1 \vee q2 \vee q3 \vee \overline{x_1\bar{x}_2x_5\bar{x}_{10}} = q1 \vee q2 \vee q3 \vee \bar{x}_1 \vee x_2 \vee \bar{x}_5 \vee x_{10}.$$

Параллельный расширитель позволяет передавать термы одного канала другому. Возможность приема в свой канал термов от соседнего канала обычно означает и возможность приема через него и термов от более далеких каналов с образованием цепочки для сбора термов от нескольких каналов (например, в пределах функционального блока). Термы от МРТ поступают в макроячейку (МЯ).

Макроячейки

Макроячейка содержит элемент ИЛИ, программируемые мультиплексоры, триггер (или триггеры) и формирует выходные сигналы в нескольких вариантах. На рис. 8.21, *а* показана схема макроячейки, прообразом которой является макроячейка CPLD фирмы Xilinx.

> ***Примечание***
>
> Программируемость мультиплексоров в этой схеме и всех дальнейших не отображается, так как присуща всем без исключения имеющимся в схеме мультиплексорам, если не оговорено противоположное. Пример программирования мультиплексора с помощью цепи, содержащей транзистор с плавающим затвором и электрическим стиранием заряда, приведен на рис. 8.21, *б*.

В зависимости от программирования каждый мультиплексор передает на выход сигнал с того или иного входа. Триггер (*регистр*) может программироваться на режимы D или T, тактируется положительными фронтами синхросигналов и имеет входы установки S и сброса R. Выходные сигналы ФБ передаются в ПМС и в блоки ввода/вывода БВВ.

Аргументы реализуемой функции поступают в матрицу $М_и$ из ПМС. Аргументами могут быть как входные сигналы, поступающие извне через БВВ, так и сигналы обратных связей, подаваемые в матрицу И с выходов макроячеек. Пять термов из матрицы И поступают на элемент ИЛИ для образования логической функции. Для управления триггером вырабатываются также термы PTSet, PTClock, PTReset (PT от Product Term), которые могут быть использованы как сигналы установки, синхронизации и сброса триггера. Терм PTOE — программируемый терм управления третьим состоянием буфера БВВ (OE — Output Enable).

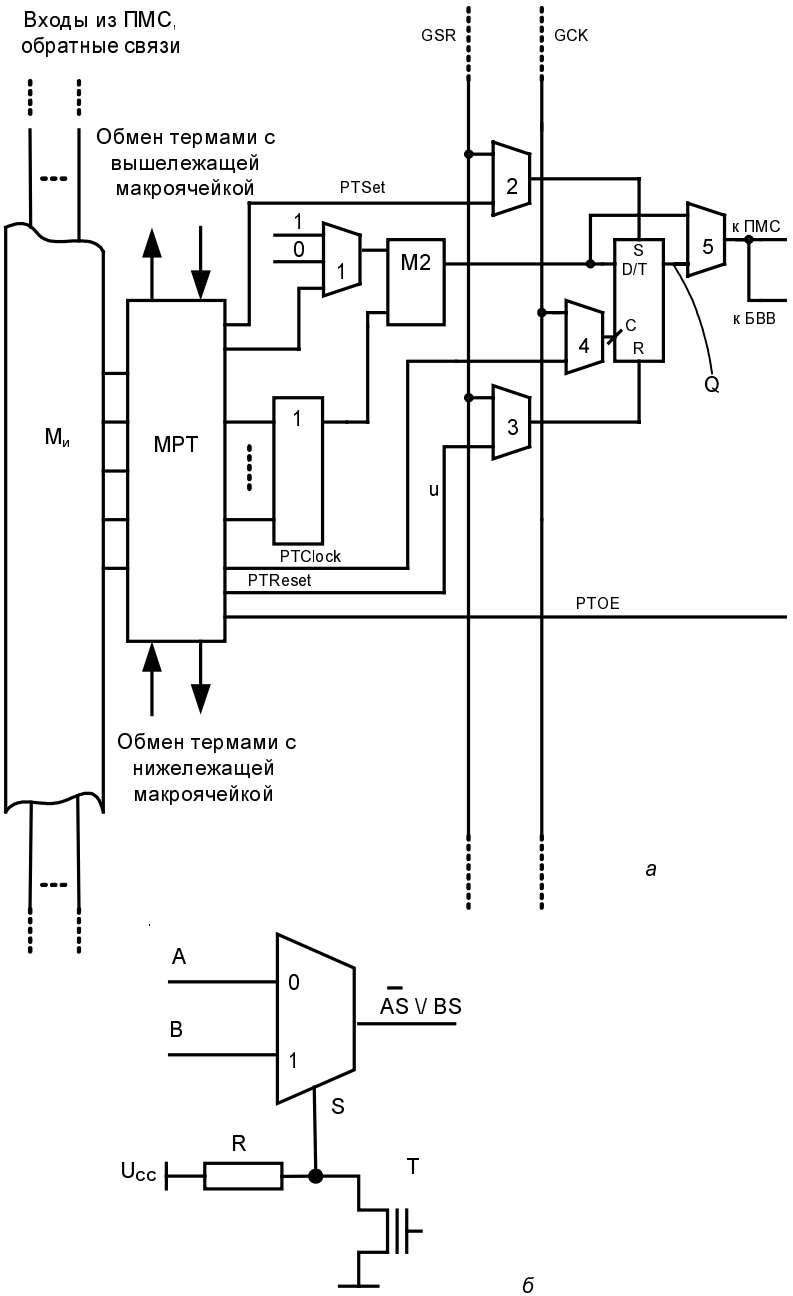


Рис. 8.21. Пример схемы макроячейки функционального блока CPLD (*а*) и пример программирования мультиплексора (*б*)

На выходе элемента ИЛИ вырабатывается ДНФ логической функции. Ее значение передается дальше на один из входов элемента сложения по модулю 2, на второй вход которого в зависимости от программирования мультиплексора 1 может быть

подан логический нуль, логическая единица или терм PT1. В первом случае функция передается без изменений (F = F*), во втором инвертируется (F = $\overline{F}$*), в третьем передается в прямом виде во всех ситуациях за исключением такой, в которой PT1 = 1.

Мультиплексор MUX5 программируется для передачи на выход МЯ либо непосредственно значения функции F (*комбинационный выход*), либо состояния триггера (*регистровый выход*). Тактирование триггера определяется программированием мультиплексора MUX4, при этом возможно использование глобального синхросигнала (GCK, Global Clock) или сигнала, порождаемого термом PTClock. Асинхронные установка и сброс триггера производятся либо глобальным сигналом (GSR, Global Set/Reset), либо термами PTSet и PTReset, что определяется программированием мультиплексоров MUX2 и MUX3.

Выходной сигнал макроячейки поступает как в ПМС, которая может направлять его по любому требуемому маршруту, так и в блоки ввода/вывода.

Системы коммутации CPLD

В CPLD используется система *одномерно непрерывных* связей, реализуемая в виде *программируемой матрицы соединений ПМС*. В этом случае все связи идентичны по конфигурации, что дает хорошую предсказуемость задержек сигналов — важное достоинство, облегчающее проектирование и изготовление схем высокого быстродействия.

В ПМС (рис. 8.22) выходы функциональных блоков ФБ подключаются к вертикальным непрерывным (не сегментированным) линиям, причем каждому выходу соответствует своя линия. Входы ФБ связаны с горизонтальными линиями, пересекающими все вертикальные линии. На пересечениях горизонтальных и вертикальных линий имеются программируемые точки связи. Замкнув одну из этих точек, можно подключить вход к соответствующему выходу. Таким образом любой вход ФБ может быть подключен к любому выходу, а каждый из выходов может быть подключен ко многим входам, чем обеспечивается *полная коммутируемость* блоков.

Для каждого соединения образуется идентичный всем другим канал связи с малым числом программируемых ключей в линиях передачи сигналов. Замкнутые транзисторные ключи имеют в первом приближении схему замещения в виде инерционной RC-цепи и вносят основные задержки в процесс распространения сигнала. При этом задержка сигнала в цепочке пассивных RC-звеньев зависит от их числа по квадратичному закону.

Программируемые ключи в линиях передачи сигналов из ПМС в ФБ могут даже отсутствовать, если эти передачи организованы так, как показано на рис. 8.23. В этом случае программируются только напряжения на нижних входах конъюнкторов, и ФБ получит сигнал от i-ой вертикальной линии ПМС (i = 1…m), если транзистор T_i будет заперт, и на нижнем входе i-го конъюнктора будет действовать высокий потенциал логической единицы. Открытый транзистор T_i подключает нижний вход конъюнкто-

ра к нулевому потенциалу, создавая на нем и на выходе конъюнктора сигнал логического нуля. Таким образом, задавая триггеру $Тг_i$ состояние логического нуля, а остальным триггерам состояние логической единицы, можно обеспечить запертое состояние транзистора $Т_i$ и открытое состояние всех других транзисторов, что означает подключение выхода ФБ к i-ой вертикальной линии ПМС с образованием так называемого непрерывного соединения. Разрешив прохождение нескольких сигналов, можно получить на выходе схемы их дизъюнкцию.

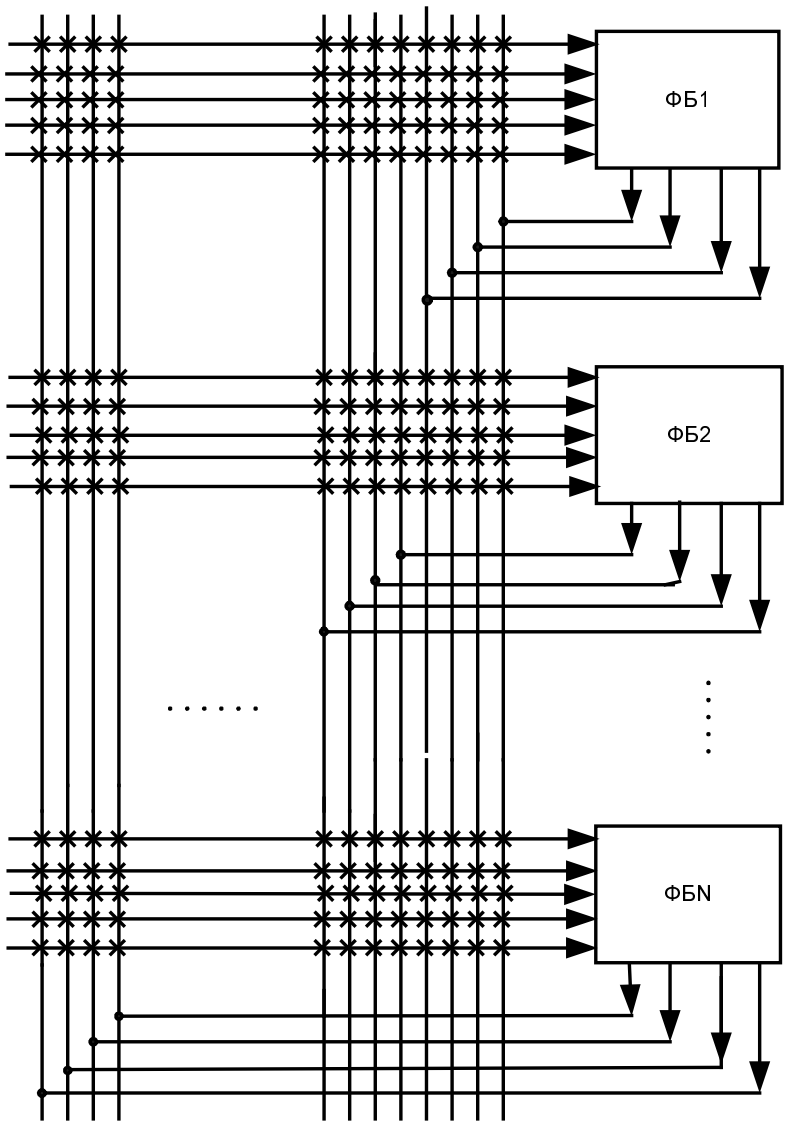


Рис. 8.22. Схема программируемой матрицы соединений

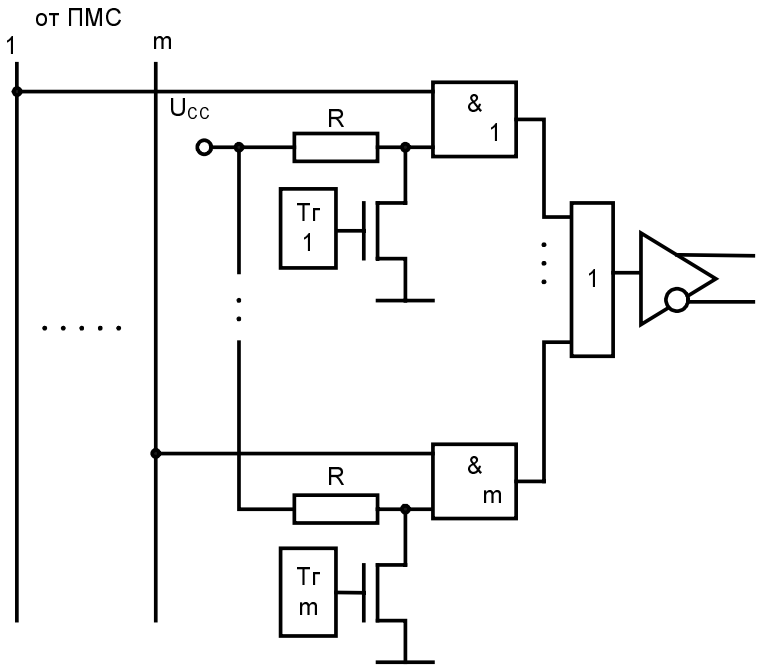


Рис. 8.23. Вариант схемы, передающей сигналы из матрицы соединений в функциональный блок

ПМС эффективны в схемах с относительно небольшим числом коммутируемых блоков. При большом их числе, характерном для FPGA, подобные ПМС чрезмерно сложны. Поэтому в FPGA системы коммутации строятся иначе — с помощью сегментированных линий связи.

Блоки ввода/вывода CPLD

Блоки ввода/вывода соединяют внешние контакты микросхемы с ее внутренними цепями. Примером такого блока может служить БВВ одной из микросхем фирмы Xilinx.

Основой БВВ (рис. 8.24) служат два буфера — входной (1) и выходной (2). Чтобы обеспечить постоянство уровней напряжения, поступающих на входной буфер, и их независимость от амплитуды входных сигналов, в схеме вырабатывается внутреннее напряжение питания U_{CCINT} и вводится цепь из двух фиксирующих диодов.

Схема программируемой общей точки ПрОТ позволяет пользователю при необходимости получать дополнительный "заземленный" вывод. Дополнительные выводы для системы "заземления" повышают ее качество и тем самым снижают уровень помех в микросхеме.

Схема ПрР программирования подключения резистора введена для исключения плавающих потенциалов на контактах ввода, когда они не используются в рабочем режиме. В этом случае цепь U_{CCINT} –R задает контакту высокий потенциал.

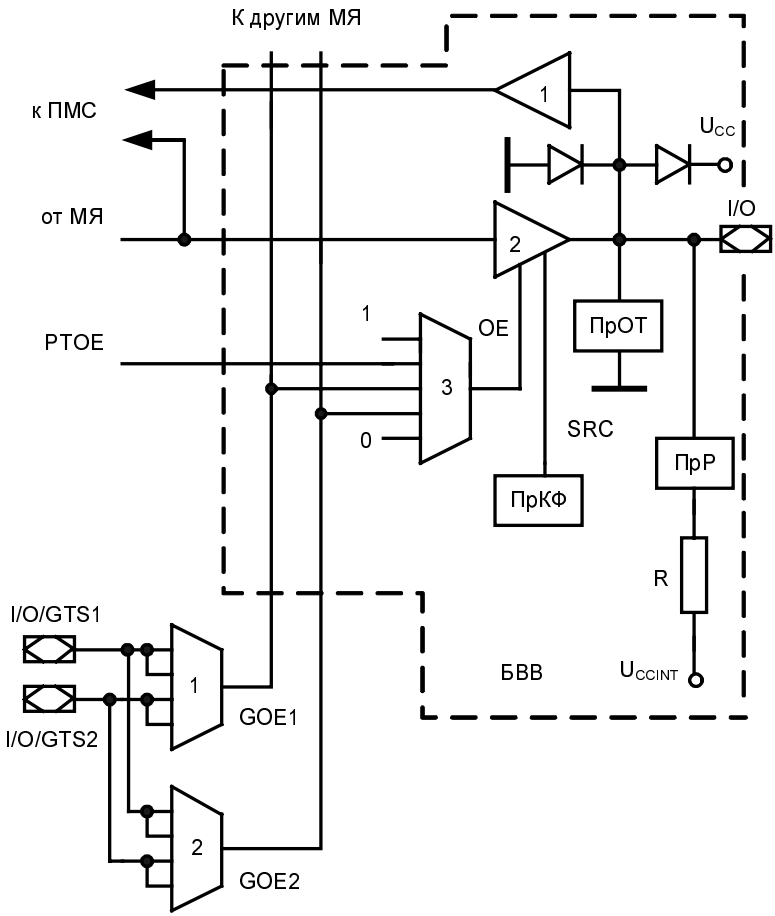


Рис. 8.24. Схема блока ввода/вывода CPLD

Выходной буфер 2 получает сигналы разрешения работы OE и управления крутизной фронта выходного напряжения SRC (Slew Rate Control). Сигнал OE с помощью программируемого мультиплексора 3 вырабатывается в нескольких вариантах: от терма PTOE, получаемого от макроячейки, от любого из глобальных сигналов управления третьим состоянием (GOE1, GOE2), от константы 1 и от константы 0. Глобальные сигналы управления третьим состоянием образуются с возможностью выбора любой полярности исходных сигналов GTS1 и GTS2.

Пример типичной CPLD

Микросхемы CPLD выпускаются фирмами Altera, Xilinx, Lattice Semiconductor, Cypress Semiconductor и др.

Микросхемы фирмы Altera. Altera — пионер в разработке CPLD, впервые реализовавшая их в конце 1980-х гг. (семейство Classic). CPLD имеют энергонезависи-

мую память конфигурации с электрическим стиранием. Основой семейств CPLD стала архитектура MAX7000.

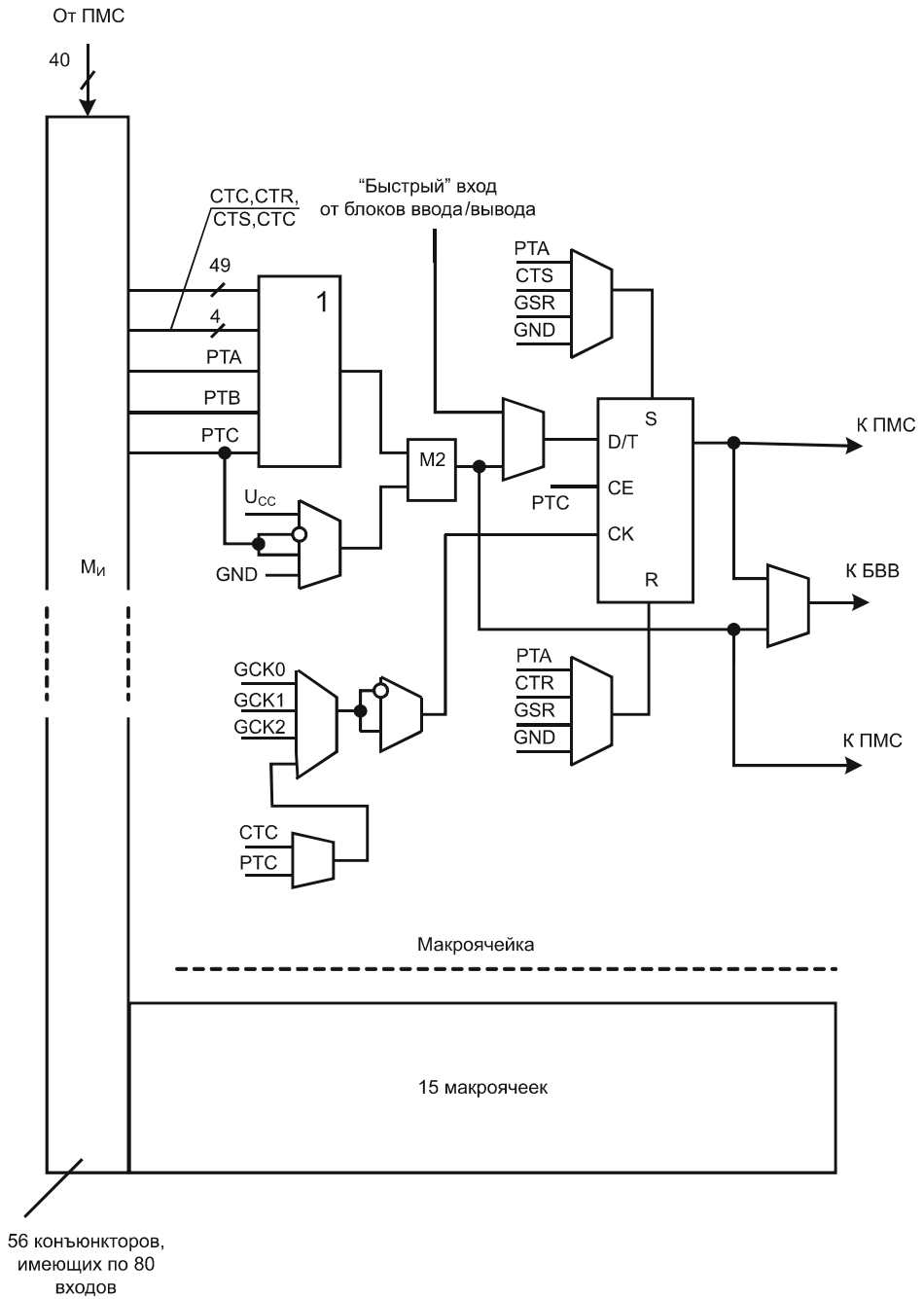


Рис. 8.25. Структура функционального блока CPLD CoolRunner-II

Семейство MAX3000A отличается от базового MAX7000 снижением требований к некоторым параметрам схемы, прежде всего, к потребляемой ими мощности. Микросхемы MAX3000A тестируются по пониженным нормам, уменьшен выбор типов их корпусов, реализованы не все возможности семейства MAX7000. Следствием стало снижение стоимости почти вдвое. Семейство MAX9000 имеет повышенный уровень интеграции. Однако вследствие этого пришлось еще дальше отойти от непрерывности межсоединений и предсказуемости и постоянства задержек сигналов независимо от их конкретного пути.

Семейство MAX-II отнесено компанией Altera к CPLD, хотя фактически это FPGA и о нем будет сказано позже.

Начиная с семейства MAX7000S, микросхемы снабжаются средствами программирования в системе (ISP). Появляется также большое разнообразие в выборе стандартов для сигналов ввода-вывода (Multivolt I/O Technology).

Микросхемы фирмы Xilinx. Фирмой Xilinx разработаны серии XC9500 и CoolRunner. Последнее по времени выпуска семейство (CoolRunner-II) обладает особенно ценными свойствами. Семейство CoolRunner-II состоит из шести представителей и выполняется по технологии с топологическими нормами 0,18 мкм. Память конфигурации типа флэш с допустимым числом циклов перепрограммирования 10^3. Время распространения сигналов от входных до выходных контактов составляет 3,5…6 нс, системная частота около 300 МГц. Семейство отличается *ультранизкими значениями потребляемой мощности*.

Структура функционального блока приведена на рис. 8.25. Его логическая часть в отличие от других CPLD выполнена не как ПМЛ, а как ПЛМ, в которой все термы, выработанные матрицей элементов И, доступны всем элементам ИЛИ, т. е. всем макроячейкам. Это 56 термов, часть которых может быть использована в качестве сигналов управления макроячеек (для тактирования, сброса/установки триггеров и т. д.). Блоки имеют как комбинационные, так и регистровые выходы. Триггер программируется как D- или T-триггер и может тактироваться сигналами разной полярности. Сброс и установка триггера могут быть асинхронными от термов матрицы PTA, управляющих термов CTR и CTS и глобального сигнала сброса GSR. Сигналы CTR и CTS — общие для блока, а GSR — общий для всех блоков. "Быстрый" вход позволяет использовать триггер как входной или защелку с сохранением комбинационной функции в данной макроячейке.

Основные параметры популярных CPLD фирм Altera и Xilinx приведены в табл. 8.1.

Таблица 8.1

Семейство	Число макроячеек, шт	Число макроячеек в блоке/ число входов блока, шт.	Время распространения сигнала от вывода к выводу, нс	Число пользовательских выводов, шт.	Ориентировочная стоимость, долларов США*
MAX 3000A	32…512	16/36	4,5…6,0	34…208	1…15

Таблица 8.1 *(окончание)*

Семейство	Число макроячеек, шт	Число макроячеек в блоке/ число входов блока, шт.	Время распространения сигнала от вывода к выводу, нс	Число пользовательских выводов, шт.	Ориентировочная стоимость, долларов США*
MAX 7000B	32…512	16/36	3,5…5,5	36…212	2…30
CoolRunner-II	32…512	16/40	3,5…6,0	44…124	1…25
XC9500	36…288	18/36	5,0…15,0	44…352	2…20

Примечание к таблице

* — указанная здесь и в последующих таблицах книги стоимость микросхем является приблизительной, поскольку меняется со временем, зависит от поставщика, размера лота и т. д. Тем не менее представляется полезным ее указывать, т. к. это важный показатель, в значительной мере определяющий выбор микросхем для реализации проекта.

Контрольные вопросы и упражнения

1. Какие причины вызвали к жизни структуры ПЛМ и ПМЛ?
2. В чем состоит отличие ПЛМ от ПМЛ? В чем ПЛМ и ПМЛ выигрывают и проигрывают относительно друг друга?
3. Во второй главе была рассмотрена схема воспроизведения логических функций с помощью дешифратора и элементов ИЛИ. Эта схема имеет сходство со структурой ПЛМ, поскольку дешифратор, как и первая матрица ПЛМ, тоже представляет собой набор элементов И. В чем состоит существенное отличие схемы с дешифратором и элементами ИЛИ от схемы ПЛМ (оцените число термов, формируемых сравниваемыми структурами, и потребность в термах при воспроизведении практически применяемых функций)?
4. Целесообразность применения конкретного варианта программируемых матриц (ПЛМ или ПМЛ) зависит от характера решаемых задач. От чего именно?
5. Укажите достоинства и недостатки реализации скобочных форм логических функций на ПЛМ и ПМЛ в сравнении с их реализациями в ДНФ.
6. В этой главе была показана функциональная равноценность структур, составленных из программируемых матриц И, ИЛИ и из матриц ИЛИ-НЕ. Покажите возможность получения тех же результатов с помощью структуры, составленной из матриц элементов И-НЕ.

7. В этой главе использовался термин "кратчайшие дизъюнктивные формы". Как следует понимать этот термин?

8. Каким образом программирование выходных буферов на передачу сигналов в прямом или инверсном виде связано с расширением функциональных возможностей ПЛМ и ПМЛ?

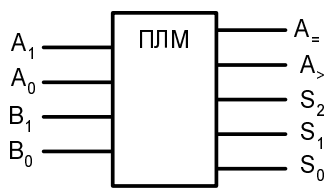
9. Почти у всех микросхем ПЛМ и ПМЛ по крайней мере часть выводов выполняется в виде двунаправленных. Почему этот вариант так популярен?

10. Какой вариант обогащения функциональных возможностей специфичен только для ПМЛ? Как он реализуется?

11. Жесткость структуры классических ПМЛ (функциональных блоков CPLD) смягчается введением в их структуру блоков МРТ. Каково назначение этих блоков?

12. Составьте и изобразите упрощенным способом схему запрограммированной ПЛМ, реализующую компаратор на "равно" для двух трехразрядных чисел $A_2A_1A_0$ и $B_2B_1B_0$.

13. Разработайте и изобразите упрощенным способом реализованный на ПЛМ функциональный узел, показанный на рисунке. На вход узла поступают два двухразрядных числа A_1A_0 и B_1B_0. На выходах вырабатываются сигналы, отображающие равенство этих чисел ($A_=$), условие A > B ($A_>$) и сумму A + B ($S_2S_1S_0$).



14. Разработайте и изобразите упрощенным способом запрограммированную ПМЛ, выполняющую преобразование двоичных кодов для чисел от 0 до 9 в код управления семисегментным индикатором.

15. Разработайте и изобразите упрощенным способом запрограммированную ПМЛ, реализующую трехразрядный синхронный двоичный счетчик с параллельным переносом. В выходных цепях ПМЛ включены триггеры типа D. ПМЛ имеет три выхода, 4 входа и по 4 терма на каждый выход.

16. Основными частями микросхем с программируемыми структурами являются функциональные блоки, системы коммутации и блоки ввода/вывода. Какие функциональные блоки и системы коммутации свойственны микросхемам CPLD?

17. В чем состоят достоинства систем коммутации, используемых в CPLD? Какие причины ограничивают возможности повсеместного применения таких систем коммутации?

Литература к главе: [17], [23], [45], [46], [48], [III], [IV], [XVI], [XXXIV].

Глава 9

FPGA — программируемые пользователями вентильные матрицы

§ 9.1. Общие сведения

FPGA (Field Programmable Gate Arrays) — программируемые пользователем вентильные матрицы — наиболее обширный класс программируемых схем, обладающих максимальными функциональными возможностями. На их основе созданы системы на программируемом кристалле СнПК (в английском оригинале SoPC, Systems on Programmable Chip).

С учетом архитектурных особенностей и областей применения выделим следующие подклассы FPGA и систем на их основе:

- FPGA невысокой и средней сложности;
- FPGA высокой сложности и системы на кристалле;
- микроконтроллерные программируемые системы.

В разработке FPGA участвуют десятки фирм, ведущие среди них — Xilinx (пионер в создании FPGA), Altera, Actel, Atmel, Lattice Semiconductor, Cypress Semiconductor (все USA) и др. Этими фирмами выпускаются семейства FPGA, которые по мере освоения новых технологических процессов (с интервалом в год-два) подвергаются модификациям и образуют *серии*, состоящие из родственных *семейств*.

Свойства и возможности FPGA

FPGA, как и другие программируемые пользователем микросхемы (ППМС), выпускаются как полностью готовые, т. е. относятся к *стандартной продукции*, что сопровождается известными преимуществами — массовостью производства и снижением стоимости. Потребитель использует их, не обращаясь к изготовителю (выполняет программирование FPGA самостоятельно).

Благодаря регулярной структуре FPGA *реализуются с уровнем интеграции, близким к максимальному*. Вместе с тем, поскольку для средств программирования межсоединений требуются значительные затраты дополнительной площади кристалла, по количеству логических элементов, *предоставляемых для реализации проекта*, FPGA, как и другие ППМС, уступают полузаказным и, тем более, заказным схемам.

Примечание

Отмеченные далее свойства и возможности присущи не только FPGA, но в некоторой мере и другим ПЛИС. Однако для FPGA эти возможности выражены наиболее ярко.

Отметим две *области применения FPGA*:

- отработка прототипов блоков и систем при их проектировании, даже если их конечная реализация рассчитана на другие средства;
- создание конечной продукции для изделий не слишком большой тиражности быстрыми и эффективными способами.

Вначале развитие ППМС было направлено на замену схем малого и среднего уровней интеграции и на перенос концепции вентильных матриц в область малотиражной аппаратуры, но в дальнейшем в связи с появлением репрограммируемости стало ясно, что это нечто значительно большее. Это, в частности, видно из приведенных далее примеров.

Построение реконфигурируемых систем. При использовании аппаратуры встречаются ситуации, в которых те или иные блоки работают поочередно. Например, средства помехоустойчивого кодирования и декодирования при передаче и приеме данных. Обе функции (кодирование и декодирование) никогда не выполняются одновременно. Поэтому не обязательно иметь два устройства (кодер и декодер), а можно иметь одну репрограммируемую схему с двумя разными конфигурациями, хранимыми в ПЗУ и поочередно загружаемыми в ПЛИС (программируемую логическую интегральную схему). В такой реконфигурируемой системе одна и та же аппаратная часть может выполнять различные преобразования после соответствующей перестройки.

Задачи логической эмуляции. При отладке устройств традиционно пользовались как изготовлением прототипа, так и программными моделями. Изготовление прототипа — сложная и дорогостоящая задача, но с его помощью можно вести тестирование с реальными сигналами и на высоких скоростях, наблюдая фактические возможности устройства. Программное моделирование лишено указанных достоинств, но проще и дешевле. Модели легко модифицируются для удаления ошибок в проекте и в них обеспечивается хорошая наблюдаемость процессов в объекте исследования.

Применение ПЛИС в задачах логической эмуляции дает сочетание достоинств обоих классических подходов. Система из микросхем программируемой логики легко создается и модифицируется, но может работать с реальными сигналами и частотами. В то же время программные модели и макетирование не зачеркиваются появлением репрограммируемых микросхем. Создание программной модели может остаться наиболее быстрым и дешевым вариантом эмуляции, а макетирование отличается наиболее полным отображением свойств объекта. Таким образом, применение ПЛИС хорошо дополняет прежние методы разработки и тестирования схем.

Построение динамически реконфигурируемых систем. От простых реконфигурируемых систем системы с динамической реконфигурацией (Run-Time Reconfiguration) отличаются тем, что в них требуется *быстрая смена настроек*. Обычная настройка с введением в микросхему последовательного потока битов или байтов занимает достаточно большое время. В динамически реконфигурируемых системах уже имеется (хранится) набор предварительно загруженных настроек, быстро сменяющих друг друга соответственно требованиям реализуемого алгоритма.

Динамически реконфигурируемая микросхема может иметь практически любое число настроек, которое ограничивается лишь емкостью памяти для их хранения. Устройства с динамической реконфигурацией уже используются практически и дают ожидаемый положительный эффект. Проблемы построения систем на микросхемах ПЛ с динамической реконфигурацией в настоящее время активно исследуются.

FPGA-процессоры. В современной литературе ставится вопрос о построении *FPGA-процессоров* с иными в сравнении с микропроцессорами свойствами. Алгоритмы работы процессора загружаются в FPGA принципиально подобно загрузке в память микропроцессорной системы выполняемой программы. Но в противоположность микропроцессорной системе, возникает сильно выраженный параллелизм на уровне логических блоков с простейшими операциями (типа воспроизведения функции от данного числа аргументов). Такие FPGA-процессоры могут давать хорошие результаты при параллельной обработке данных, где большое число переменных преобразуется сходным образом.

Эффективность схем с программируемой структурой стимулирует быстрый рост соответствующей отрасли промышленности и объемов их производства, а также научных исследований по развитию их архитектур, схемотехники и алгоритмов решения практических задач.

Программируемые элементы

Программируемость пользователем, т. е. реализуемость индивидуального проекта на основе стандартной микросхемы, обеспечивается наличием в схеме множества двухполюсников, проводимость которых может быть задана либо очень малой (это соответствует разомкнутому ключу), либо достаточно большой (это соответствует замкнутому ключу). *Состояния ключей задают конфигурацию схеме, формируемой на кристалле*. Число программируемых ключей (программируемых точек связи) в схеме зависит от ее сложности и может доходить до сотен миллионов и более. Для FPGA характерны следующие виды программируемых ключей:

- перемычки типа antifuse (общепринятый русский термин отсутствует);
- ключевые транзисторы, управляемые триггерами;
- флэш-ключи.

Перемычки типа antifuse. Программирование перемычками antifuse (фирмы QuickLogic и Actel) является однократным. В этом заключается основное ограни-

чение на область применения схем с перемычками. В остальном перемычки дают коммутацию отличного качества.

Перемычка второго поколения фирмы Actel (рис. 9.1, *а*) компактна, имеет очень малые токи в исходном непроводящем состоянии и очень малые паразитные емкости (единицы фемтоампер и фемтофарад, фемто = 10^{-15}). Перемычка образована слоями "поликремний —диэлектрик" и помещена между проводящими дорожками металлизации с включением специального вольфрамового электрода. На рис. 9.1, *б* для сравнения изображено обычное (не программируемое) соединение.

Программирующий импульс напряжения пробивает перемычку и создает в ней проводящий канал. Сопротивление запрограммированной перемычки мало (приблизительно 25 Ом). Малые сопротивления и емкости перемычек положительно влияют на скорость распространения сигналов в программируемых связях. Схемы с перемычками обладают компактностью, повышенной радиационной стойкостью, надежностью и невысокой стоимостью. Поскольку перемычки включаются между металлическими проводниками, их размещают не в одной плоскости с логическими схемами, а над ними, экономя таким образом площадь кристалла.

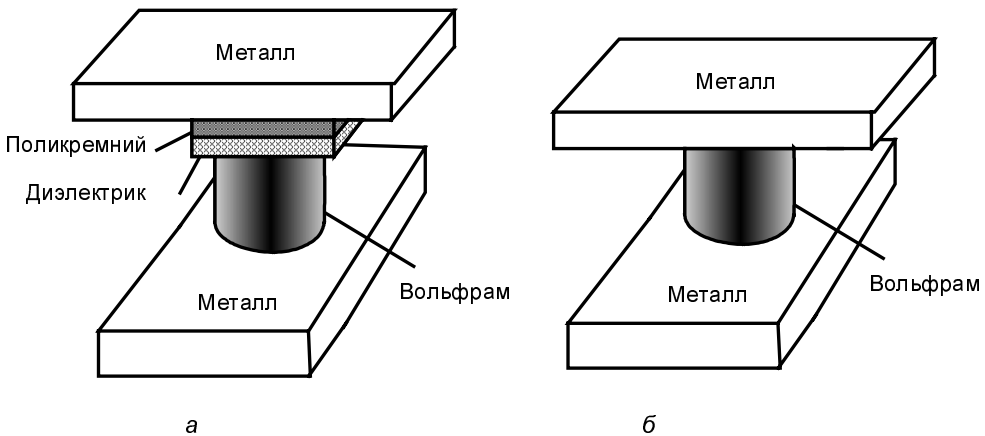


Рис. 9.1. Программируемые перемычки antifuse (*а*) и обычные перемычки "металл-металл" (*б*)

Ключевые транзисторы, управляемые триггерами. В этой схеме (рис. 9.2) транзисторный ключ замыкает или размыкает участок "a – b" в зависимости от состояния триггера.

При программировании на линию выборки подается высокий потенциал, и транзистор T1 включается. С линии записи-чтения подается сигнал, устанавливающий триггер в состояние логической "1", что замкнет ключ, или "0", что разомкнет ключ. В рабочем режиме транзистор T1 заперт и триггер сохраняет неизменное состояние. Совокупность триггеров памяти конфигурации называют *теневым ЗУ*. Поскольку триггерная память имеет название SRAM, микросхемы с такой памятью конфигурации называют схемами *SRAM-based*).

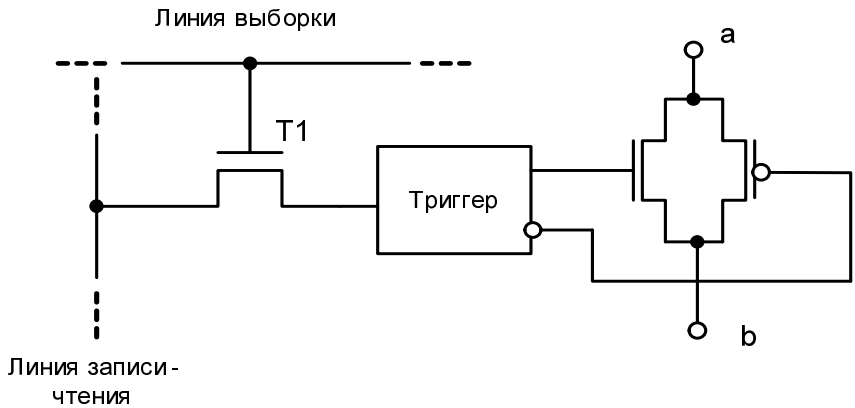


Рис. 9.2. Схема ключа,
управляемого триггером памяти конфигурации

Триггеры памяти конфигурации распределены по кристаллу вперемежку с элементами схемы, которые они конфигурируют. Режим программирования по своему характеру не отличается от рабочего, поскольку сводится к простой записи кодовой последовательности в цепочку триггеров. Стирание информации — просто сброс триггеров.

Так как от триггера памяти конфигурации не требуется высокое быстродействие, он оптимизируется по параметрам компактности и максимальной устойчивости стабильных состояний.

Схема "триггер плюс ключ" значительно сложнее элементов antifuse и транзисторов с плавающим затвором. Тем не менее, именно такие схемы доминируют в FPGA. Причина этого состоит как в рассматриваемых далее достоинствах этих схем, так и в том, что их сложность компенсируется *технологической однородностью* с другими схемами кристалла (логикой, регистрами), чего не имеют запоминающие элементы других типов.

Программирование (загрузка памяти конфигурации) может производиться *неограниченное число раз*. Конфигурация, хранимая в триггерной памяти, разрушается при каждом выключении питания. При включении питания необходим *процесс программирования (инициализации, конфигурирования) схемы* — загрузка данных конфигурации из какой-либо энергонезависимой памяти.

Флэш-ключи. На рис. 9.3 показана схема флэш-ключа, главным элементом которой служит пара транзисторов с общим плавающим затвором. Левый транзистор используется только для программирования ключа и чтения его состояния. Правый транзистор программирует участок "a-b", поскольку является ключом, которому передается состояние левого транзистора в силу общности плавающего затвора для обоих транзисторов.

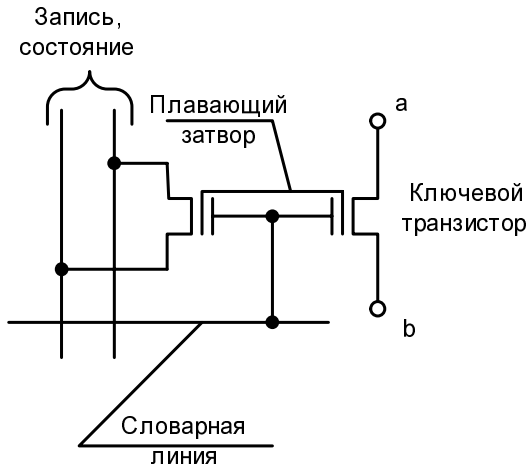


Рис. 9.3. Схема флэш-ключа

§ 9.2. Архитектура и основные блоки FPGA

Базовая архитектура

Во внутренней области FPGA базовой архитектуры по строкам и столбцам размещаются идентичные функциональные блоки ФБ (КЛБ, конфигурируемые логические блоки), между которыми проходят трассы межсоединений. На периферии кристалла расположены блоки ввода/вывода БВВ. Для первого поколения перечисленные части составляли по существу всю схему FPGA. В последующем архитектура FPGA усложнилась, и в ее составе появились дополнительные функциональные ресурсы, среди которых в первую очередь следует назвать *встроенные блоки памяти* (их стали вводить практически во все разрабатываемые микросхемы) и *умножители*. В составе FPGA появились также *специализированные* средства для автоподстройки задержек в системе тактирования (PLL, DLL, DCM), средства для поддержки интерфейса JTAG, для реализации высокопроизводительных шин, для генерации тактовых сигналов и т. д.

На рис. 9.4 укрупненно показаны фрагмент FPGA базовой архитектуры и состав ее функционального блока ФБ, в который входят:

- функциональный (логический) преобразователь (ФП);
- триггер (регистр RG);
- мультиплексоры (MUXs), играющие роль средств конфигурирования ФБ.

При конфигурировании FPGA функциональные (логические) блоки настраиваются на выполнение требуемых операций, а система соединений — на требуемые связи между элементами и блоками. В результате в FPGA реализуется схема необходимой

конфигурации. Блоки ввода/вывода связывают FPGA с внешней средой, и их, как правило, можно программировать на выполнение ряда стандартов передачи данных.

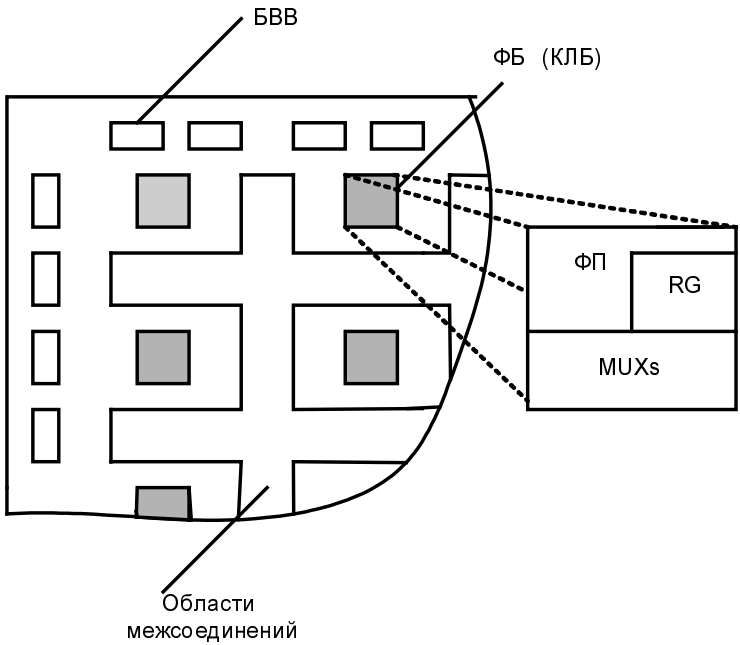


Рис. 9.4. Фрагмент FPGA базовой архитектуры и ее функциональный блок

Усложненные архитектуры

В схемах FPGA с усложненной архитектурой помимо уже перечисленных блоков применяются и дополнительные, к которым относятся:

- ресурсы памяти;
- умножители;
- схемы управления синхросигналами (PLL, DLL);
- блоки ЦОС (цифровой обработки сигналов);
- блоки обработки аналоговых и аналого-цифровых данных.

Фрагмент типичной структуры усложненной FPGA показан на рис. 9.5 (микросхема Stratix-3).

В число основных частей микросхемы входят адаптивные логические модули ALM, блоки быстродействующей памяти с изменяемой организацией, блоки цифровой обработки сигналов (ЦОС-блоки) и банки ввода/вывода, обеспечивающие интерфейс для взаимодействия с различными внешними устройствами. Применена трехуровневая иерархия блоков памяти. Имеются блоки памяти трех типов — MLAB, M9K и M144K с емкостями 640 бит, 9 Кбит и 144 Кбита соответственно.

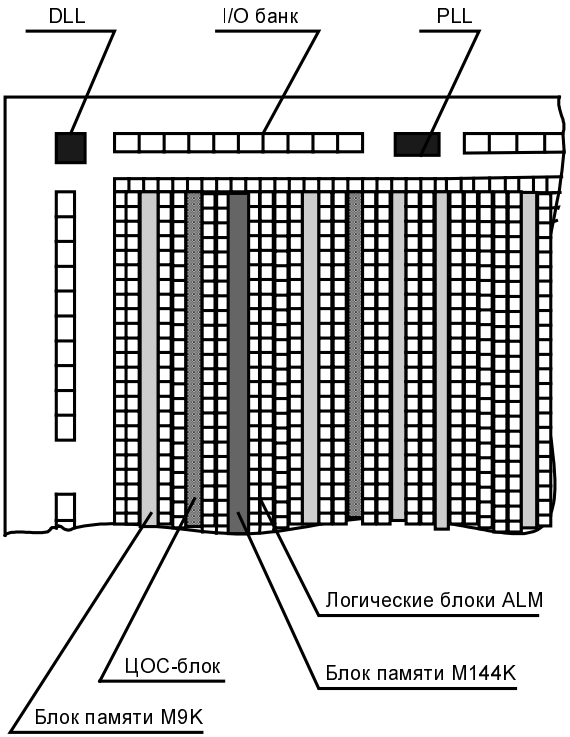


Рис. 9.5. Структура FPGA усложненной архитектуры

Логические блоки

В качестве КЛБ (конфигурируемых логических блоков FPGA или, для краткости, просто ЛБ) используются:

- SLC — Simple Logic Cells, простые логические вентили;
- логические модули на основе мультиплексоров;
- LUT-блоки — программируемые ЗУ (LUT — Look-Up Table).

LUT-блок — наиболее распространенная разновидность логического преобразователя для FPGA с триггерной памятью конфигурации. В однократно программируемых FPGA с пробиваемыми перемычками antifuse находят применение логические модули на основе мультиплексоров, а в схемах с флэш-ключами — преобразователи с простыми логическими вентилями.

Одной из характеристик логических блоков является их "*зернистость*" (Granularity), определяющая, насколько "мелкими" будут те части, из которых можно "собирать" нужные схемы.

Мелкозернистость ЛБ ведет к гибкости их использования, возможностям реализовать воспроизводимые функции разными способами, получая разные варианты в

координатах "площадь кристалла — быстродействие". В то же время она усложняет систему межсоединений в связи с большим числом программируемых точек связи.

Среднезернистые логические блоки. Пример *среднезернистого* ЛБ (фирма Actel) — блок, состоящий из трех мультиплексоров "2—1" и элемента ИЛИ (рис. 9.6).

Подключая ко входам определенные сочетания переменных и констант, можно получить все комбинационные функции двух переменных, все функции трех переменных с не менее, чем одним, положительно юнатным входом, многие функции четырех переменных и некоторые функции большего числа переменных, вплоть до восьми. В целом получаются 702 различных варианта. Например, подключая ко входам переменные и константы согласно рис. 9.6, получим функцию $F = ab \vee \bar{c}$.

Крупнозернистые логические блоки. В FPGA с триггерной памятью конфигурации обычно применяют крупнозернистые блоки, в которых реализуются более сложные функции. Это ведет к упрощению программируемой части межсоединений, но затрудняет полное использование логических элементов блоков, что может привести к излишним затратам площади кристалла и снижению быстродействия. Иными словами, меняя зернистость, можно выиграть в одном и проиграть в другом.

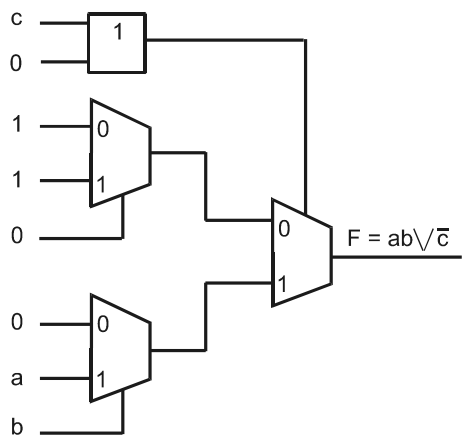


Рис. 9.6. Пример реализации функции с помощью мультиплексорного логического блока

Пример 1. Логический блок (фирма Xilinx, серия Spartan) в несколько упрощенном виде показан на рис. 9.7 (как и ранее, в обозначениях мультиплексоров не отражена их программируемость, поскольку все они без исключения обладают этим свойством).

Логические преобразования выполняются тремя LUT-блоками (G, F и H). Преобразователи G и F — программируемые ЗУ с организацией 16×1, способные воспроизводить любые функции четырех переменных, значения которых могут быть переданы на выходы Y и X через мультиплексоры 4 и 6 при соответствующем их программировании (через линии верхних входов).

Через верхний вход мультиплексора 1 и нижний вход мультиплексора 2 функции G и F могут быть поданы на ФП-Н (ЗУ с организацией 8×1) для образования "функции от функций" с целью получения результирующей функции, зависящей от более чем четырех аргументов. К третьему входу ФП-Н подключен входной сигнал H1, так что H = f (G, F, H1). Аргументами для ФП-Н, поступающими от мультиплексоров 1 и 2, в зависимости от их программирования может быть не только набор G, F, H1, но также наборы G, H1, DIN; SR, H1, DIN; SR, H1, F. Линии DIN и SR используются либо для передачи в триггер непосредственно входных данных и сигнала установки/сброса (Set/Reset), либо как входы ФП-Н.

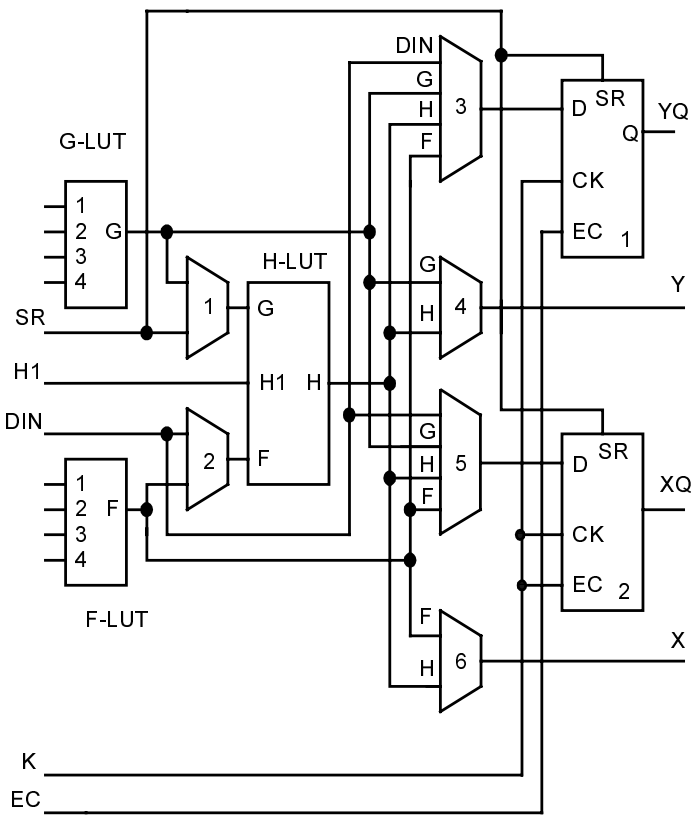


Рис. 9.7. Схема ЛБ семейства Spartan

Ресурсы логической части ФБ позволяют воспроизводить следующие функции:

- любую функцию с числом аргументов до 4 включительно плюс вторую такую же функцию плюс любую функцию с числом аргументов до трех;
- любую функцию 5 аргументов (одну);
- любую функцию 4 аргументов плюс некоторые функции 6 аргументов, некоторые функции с числом аргументов до 9.

Мультиплексоры 3 и 5 направляют сигналы на триггеры 1 и 2. Триггеры могут использоваться для фиксации и хранения выходных сигналов функциональных преобразователей или же работать независимо от них. Входной сигнал DIN для ФБ может быть прямым входом для любого триггера. Сигнал H1 тоже можно передавать любому триггеру, но через ФП-H, что вносит в цепь его передачи некоторую задержку.

Оба триггера имеют общие входы CK, тактируемые сигналом K, входы разрешения тактирования EC и установки/сброса SR. Внутренние программируемые цепи в схеме триггера позволяют индивидуально программировать полярность тактирующего сигнала, поступающего на вход CK. Сигнал EC синхронизирован с сигналом, подаваемым на вход CK, а сигнал SR асинхронный и для каждого триггера с использованием его внутренних цепей программируется как сигнал установки или сброса. Этот сигнал определяет состояние, в котором окажется триггер после процесса конфигурации микросхемы. Конфигурация определяет и характер воздействия на триггеры импульсов установки/сброса при работе схемы.

Воспроизведение функций многих переменных. Для получения функций с числом аргументов, превышающим возможности отдельных функциональных блоков, используются *цепочки каскадирования*. При каскадировании можно реализовать функции с большим числом аргументов, используя их декомпозицию. При этом блоки ФП соседних элементов параллельно во времени вычисляют частичные функции, а цепи каскадирования образуют последовательную цепочку для их объединения (рис. 9.8, *а*).

В цепочке каскадирования объединяющими могут быть элементы И либо ИЛИ. Каждый элемент цепочки добавляет для воспроизводимой функции 4 аргумента и вносит дополнительную задержку. Цепочки каскадирования, как и цепочки переносов, могут формироваться автоматически компилятором *системы автоматизированного проектирования* (*САПР*) или самим проектировщиком при вводе проекта в САПР.

Функции многих переменных можно получить и другим способом, применяя обратные связи. При этом сначала вырабатывается некоторая функция ограниченного числа аргументов (в данном случае четырех), затем она вводится в качестве одного из входов в другой ЛЭ и т. д. В результате вычисляется "функция от функций" с числом аргументов, превышающим 4 (рис. 9.8, *б*).

Пример 2. Логический блок семейства Cyclone (фирма Altera). В логический блок входят 16 логических элементов. Логические элементы имеют два типовых режима работы — нормальный и арифметический.

В *нормальном* режиме (рис. 9.9, *а*) Lut-блок имеет четырехвходовую конфигурацию, имеются также входы цепи переноса и цепи регистров. Регистр может конфигурироваться как D, T, JK или RS и имеет входы данных, тактирования, разрешения тактирования и сброса (сигналы управления регистром являются общими для всего логического блока). Сигналы от LUT-блока могут поступать на выходы как комбинационные (с обходом регистра) или регистровые.

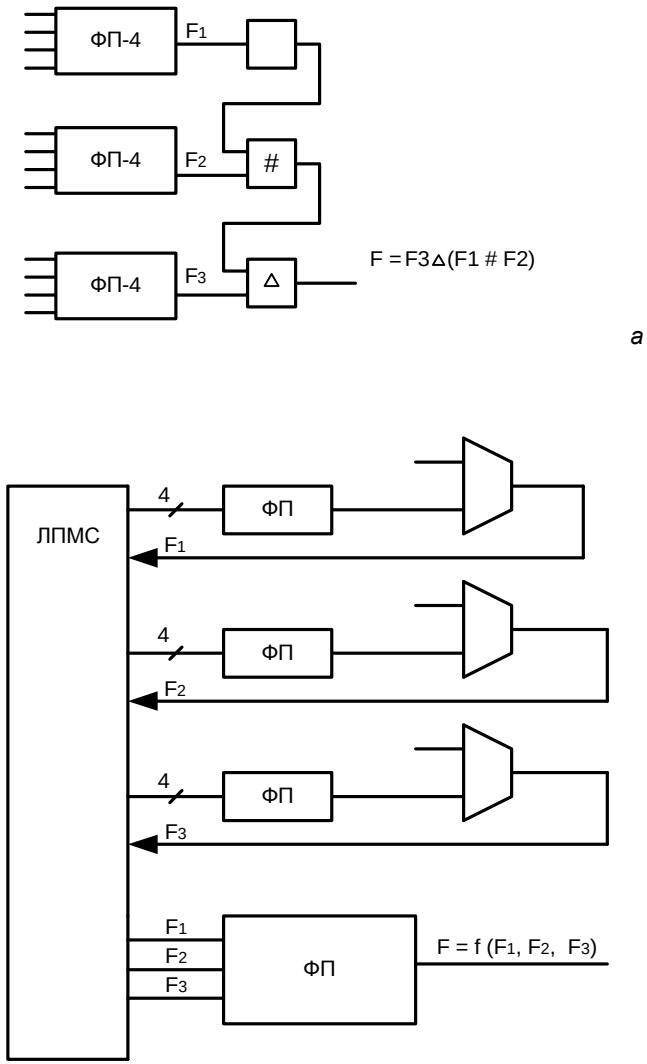


Рис. 9.8. Способы воспроизведения функций многих переменных методами каскадирования (*а*) и обратных связей (*б*)

Элемент имеет три выхода, на которые сигналы с LUT-блока и регистра могут поступать независимо, так что возможен вывод двух информационных сигналов — одного с регистра, а другого непосредственно с LUT-блока (свойство, называемое *Register Packing*). При соответствующем программировании мультиплексора 1 создается обратная связь с выхода регистра на вход LUT-блока. Вход и выход регистровой цепи позволяют каскадировать регистры одного логического блока с образованием регистра сдвига. При этом сохраняется и возможность выработки данным логическим элементом комбинационной функции.

Арифметический режим логического элемента (рис. 9.9, *б*) применяется при построении сумматоров, счетчиков, компараторов и других структур, состоящих из

разрядных схем. При этом LUT-блок разбивается на два блока с тремя входами каждый, один из блоков реализует функцию выхода данного разряда, другой — сигнал переноса. Цепь переноса может превышать по длине 16 элементов путем перехода в другой логический блок данного столбца. Выходы элемента, как и ранее, возможны в комбинационном и регистровом вариантах.

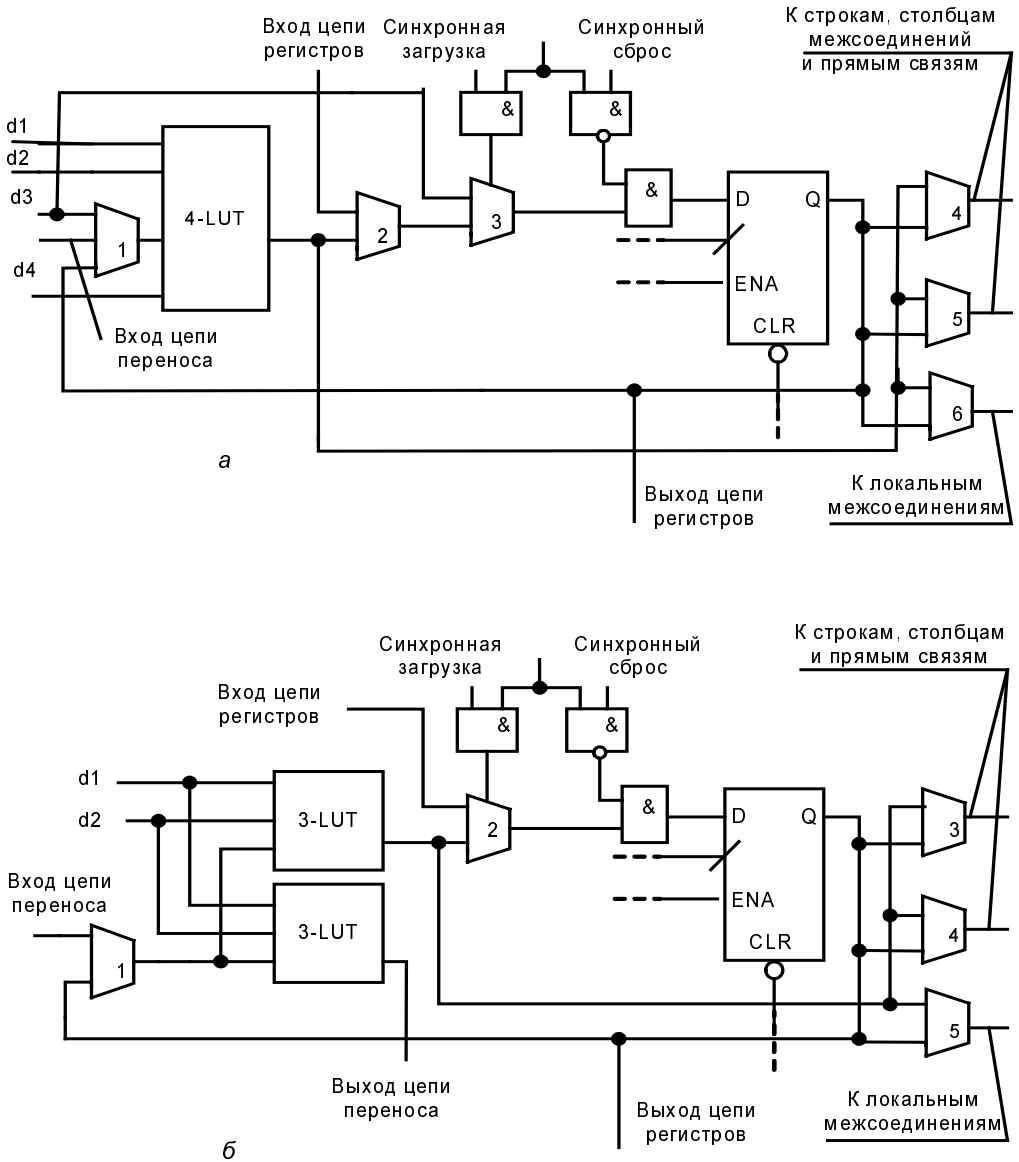


Рис. 9.9. Схемы логического элемента в нормальном (*а*) и арифметическом (*б*) режимах

В логический блок (рис. 9.10) входят 16 логических элементов и локальные межсоединения. Блок имеет общие для логических элементов сигналы управления, в нем организуются цепь переноса и цепь регистров. Локальные межсоединения коммутируют логические элементы в пределах блока, получая сигналы как от выходов логических элементов блока, так и от трассировочных столбцов и строк. Для связей с соседними схемами (логикой, памятью, умножителями) предусмотрены прямые связи.

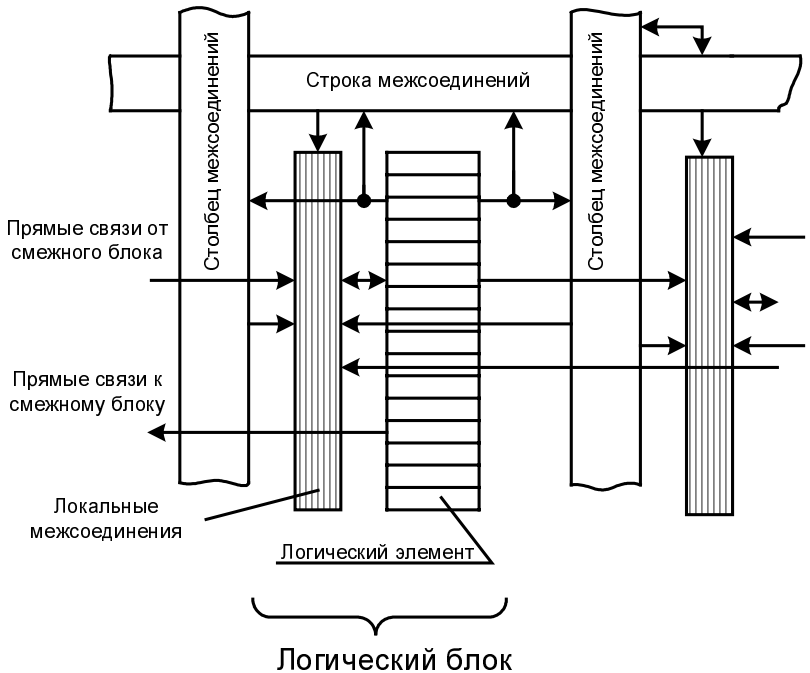


Рис. 9.10. Логический блок и его межсоединения

Системы межсоединений

Системы межсоединений (коммутации), как и логические блоки, реализуются в широком диапазоне архитектурных и технологических решений. Линии связей в FPGA обычно *сегментированы*, т. е. составлены из отдельных проводящих сегментов (участков, не содержащих ключей). Сегменты соединяются друг с другом *программируемым элементом связи* (ключом).

Так как короткие сегменты затрудняют реализацию длинных связей (в них появляется большое число ключей, увеличивающих затраты площади кристалла и задержки сигналов), а длинные — коротких, целесообразна *иерархическая система связей* с несколькими типами межсоединений для передач на разные расстояния. Поэтому в системе межсоединений применяются сегменты различной длины.

Цели построения системы связей — максимальная коммутируемость блоков при минимальном количестве ключей, а также, по возможности, предсказуемость задержек, облегчающая проектирование. Зависимость задержек от конкретного пути создает проблему возможных гонок сигналов и сбоев в работе схемы. *Трассировочная способность* межсоединений отражает возможность создания в FPGA множества схем типового применения (только с помощью программируемых ключей, т. к. сегментная часть соединений стандартна).

Быстродействие FPGA существенно зависит от задержек в связях, определяемых в основном ключами, и квадратично зависящих от числа ключей в последовательной цепочке. Цепи с большим числом ключей особенно нежелательны. Целесообразно разбивать длинные цепочки сегментов на несколько коротких с помощью промежуточных буферных каскадов.

Пример 1. Иерархическая система межсоединений FPGA базовой архитектуры, включающая в себя:

- связи общего назначения (General-Purpose Interconnects);
- длинные линии (Long Lines);
- прямые связи (Direct Interconnects);
- линии тактирования (Clock Lines).

Пример связей общего назначения (фирма Xilinx) показан на рис. 9.11, *а*. В них применяются связные и переключательные блоки (СБ и ПБ). Связные блоки — области пересечений выводов логического блока ЛБ и линий вертикальных и горизонтальных трасс, обозначенные прямоугольниками из штриховых линий. Крестики обозначают программируемые точки связей. Через СБ сигналы логических блоков связываются с линиями трасс. Переключательные блоки расположены на пересечении горизонтальных и вертикальных трассировочных каналов, каждый из которых имеет восемь линий.

Линии могут иметь одинарную длину (соединяют соседние ПБ, как на рис. 9.11, *а*) или двойную (соединяют ПБ через один). Могут применяться и связи общего назначения с линиями учетверенной длины. Структура ПБ (рис. 9.11, *б)* позволяет передавать сигналы влево-вправо или вверх-вниз между смежными линиями, а также изменять направление передачи сигнала. Схема, соответствующая зачерненному квадрату, показана отдельно справа (рис. 9.11, *в*). Для обеспечения перечисленных передач в эту схему входят 6 ключевых транзисторов.

Для ускорения и упрощения дальних передач приняты специальные меры. Для передач на большие расстояния с малой задержкой или для передач на разные приемники с малым расфазированием сигналов служат *длинные линии* (здесь термин "длинные линии" имеет прямой смысл, и его не следует путать с аналогичным термином, употребляемым при согласовании волновых сопротивлений в ином смысле). Длинные линии пересекают кристалл вдоль или поперек по всей его длине или ширине. Могут применяться несколько типов длинных линий: горизонтальные и вертикальные (по несколько на каждую строку и столбец логических блоков), линии для тактирования блоков ввода/вывода (вдоль блоков ввода/вывода), так назы-

ваемые глобальные линии с выходами на определенные БВВ и т. д. Для связей с соседними логическими блоками ЛБ часто применяются локальные *прямые связи*. На рис. 9.12 приведен пример трассировочных ресурсов FPGA.

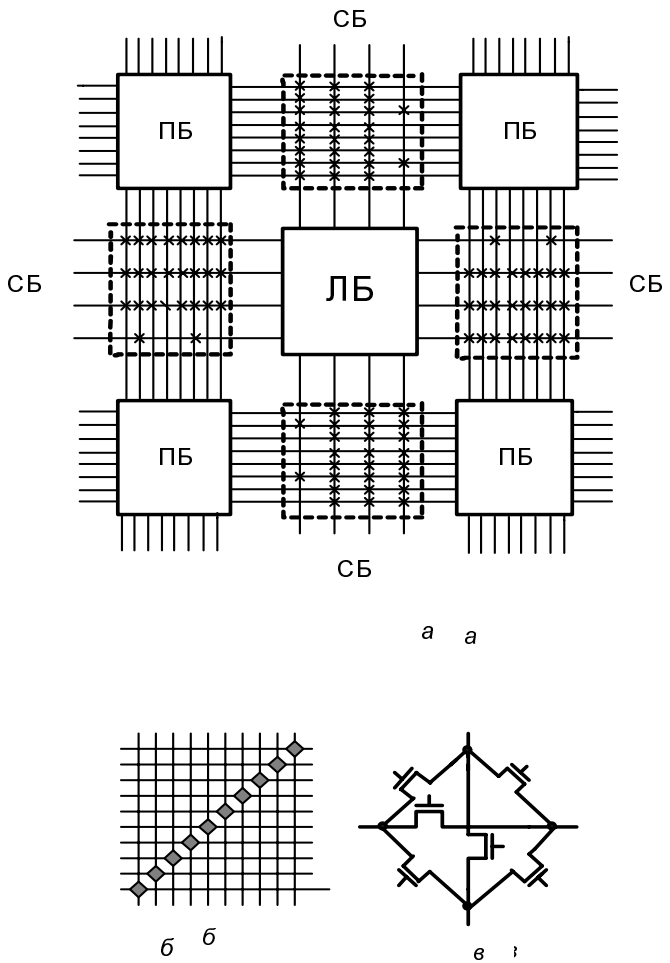


Рис. 9.11. Схема связей общего назначения с линиями одинарной длины (*а*) и схема переключательного блока (*б, в*)

Кроме системы коммутации для логических блоков некоторые FPGA могут иметь дополнительные трассировочные ресурсы VersaRing, расположенные в виде кольца между матрицей КЛБ и блоками ввода/вывода. Эти ресурсы позволяют с помощью конфигурирования изменять назначение вводов/выводов микросхемы (т. е. по разному соединять внутренние точки схемы с внешними выводами) и облегчают тем самым модификацию проекта, реализованного на FPGA, без влияния на разводку печатных плат, на которых монтируются микросхемы. Кроме того, указанные воз-

можности содействуют ускорению работ по выпуску новой продукции, так как программируемость соединений между внешними и внутренними выводами схемы позволяет вести разработку печатных плат, не дожидаясь полной отладки микросхем, входящих в состав проекта.

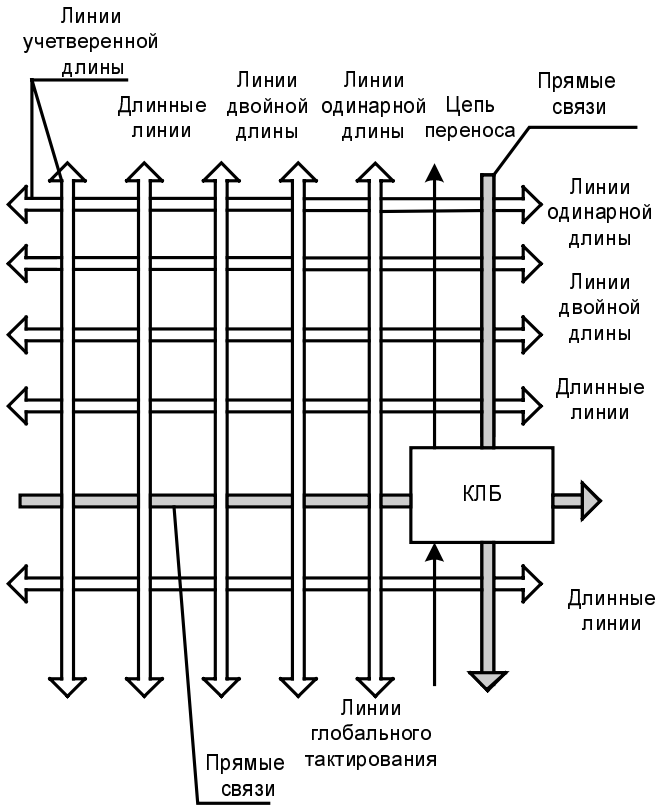


Рис. 9.12. Пример трассировочных ресурсов FPGA

Двухуровневая система межсоединений. Такой вариант системы межсоединений представлен главным образом в микросхемах фирмы Altera. В его основе лежит стремление обеспечить предсказуемость задержек сигналов в системе коммутации и их идентичность. Эти факторы существенно облегчают проектирование для сложных схем высокого быстродействия. Указанными свойствами обладают системы коммутации CPLD, но они неприемлемо сложны, если речь идет о межсоединениях большого числа схемных элементов. Компромисс в известной мере реализуется в двухуровневой системе.

Двухуровневая коммутация порождает и два уровня иерархии логических преобразователей (рис. 9.13). Наименьшая единица логических ресурсов — *логический элемент ЛЭ* (LE, Logic Element). Группа логических элементов совместно со средствами *локальных* межсоединений образует *логический блок ЛБ* [Y2](LAB, Logic Array Block).

Логические блоки содержат логические элементы и связи для их внутриблочной коммутации (локальную программируемую матрицу соединений). Межблочная коммутация реализуется строками и столбцами соединений второго уровня. К концам трассировочных строк и столбцов подключаются блоки ввода/вывода (БВВ). Каждый логический блок имеет связи с горизонтальными и вертикальными трассами. Кроме того, горизонтальные и вертикальные трассы имеют между собой двусторонние связи.

Подробнее двухуровневые средства коммутации изображены на рис. 9.14.

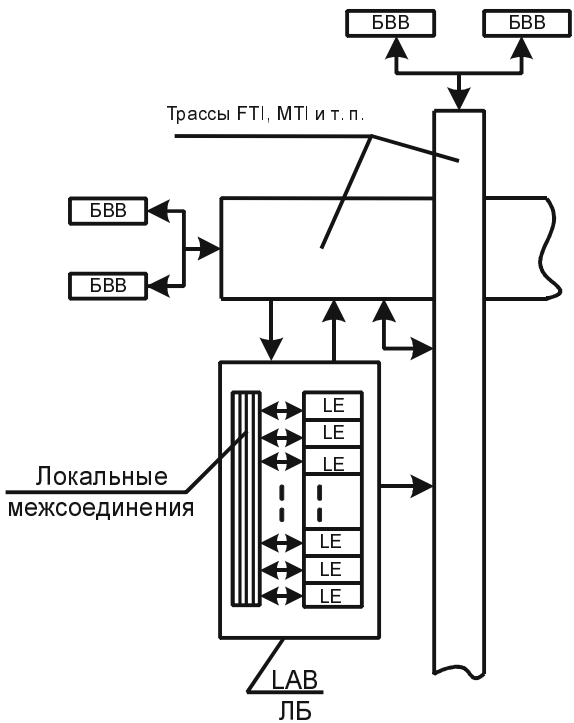


Рис. 9.13. Структура двухуровневых межсоединений в схемах FPGA

В логический блок входят n логических элементов LE1…LEn, схема передачи им сигналов управления (мультиплексор 3) и локальная программируемая матрица соединений ЛПМС. Остальные элементы относятся к системе межблочной коммутации, которая обеспечивается горизонтальными и вертикальными трассами (строками и столбцами), т. е. группами линий фиксированной, но разной длины, как проходящих вдоль всех логических блоков строки или столбца, так и распространяющихся на меньшие расстояния. Горизонтальные трассы передают сигналы между логическими блоками строки и связаны со своими блоками ввода/вывода БВВ. Вертикальные трассы передают сигналы между горизонтальными трассами и также связаны со своими БВВ. Все линии выходов на горизонтальные и вертикальные трассы буферированы, поскольку испытывают значительные нагрузки. Показанные

на рисунке средства межблочной коммутации относятся к одному логическому элементу — LE1, все другие элементы снабжены такими же средствами.

Линия строки получает сигналы от логического элемента или от одной из трех линий столбца. Эти сигналы поступают через два мультиплексора (1 и 2) и передаются на две линии строки. Такая схема допускает передачи от столбцовых линий на строчные даже в тех случаях, когда все логические элементы блока используют выходы в строку. Столбцовые линии могут передавать сигналы своим элементам ввода/вывода БВВ или в трассы других строк для передач другим блокам (логическим блокам или встроенным блокам с памятью).

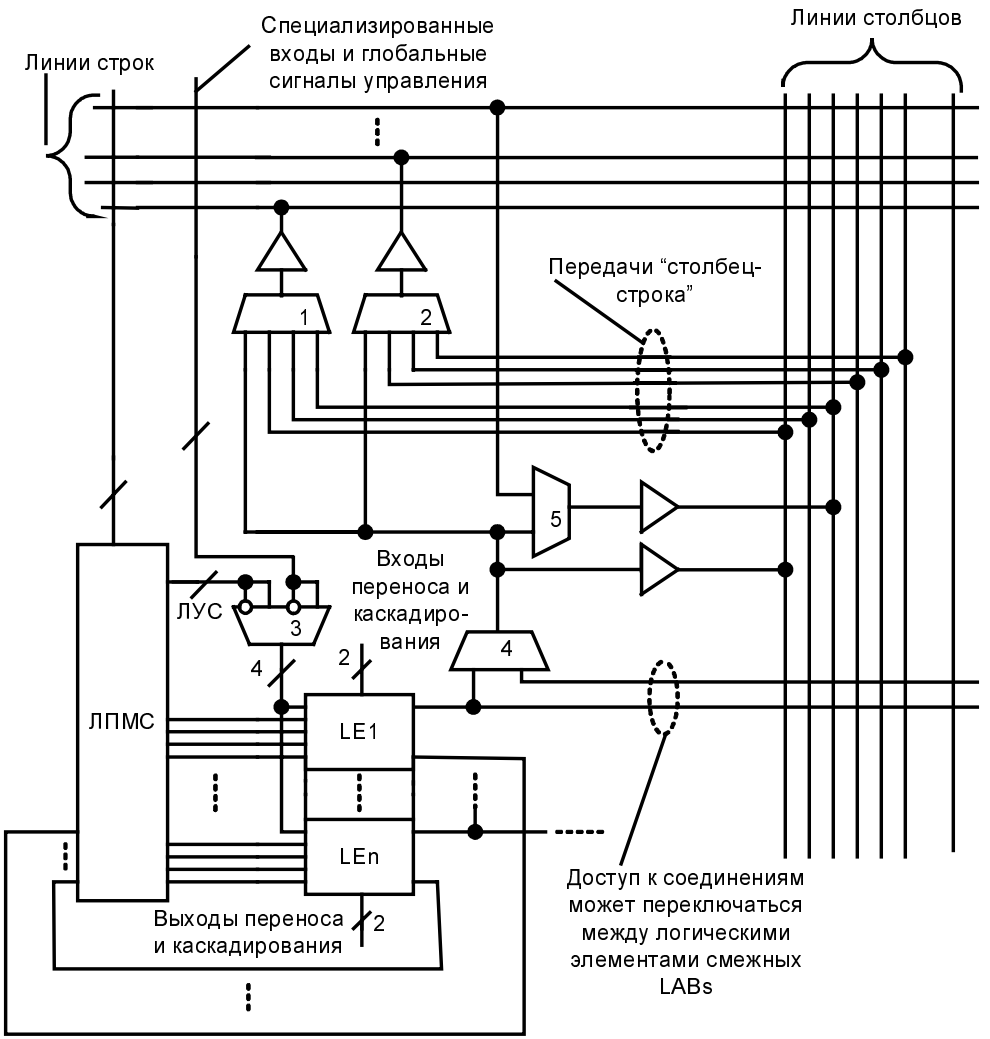


Рис. 9.14. Средства двухуровневых межсоединений в схемах FPGA

Доступ к линиям строк и столбцов, предусмотренный для логического элемента данного блока, может быть переключен для использования логическим элементом смежного блока и, наоборот, сигналы данного элемента могут получить доступ с ресурсам коммутации соседнего блока (мультиплексор 4). Такая возможность облегчает эффективное использование трассировочных ресурсов микросхемы.

В каждом блоке вырабатываются *локальные управляющие сигналы* ЛУС (тактирующие, сброса и установки регистра), которые могут быть использованы во всех логических элементах блока. Возможности тактирования велики — может быть использован сигнал от специального входного контакта, сигнал глобального тактирования, сигналы от контактов ввода/вывода или внутренние сигналы (через локальную систему межсоединений).

Блоки ввода/вывода

Каждому выводу корпуса микросхемы придается блок ввода/вывода, который может конфигурироваться как вход, выход или двунаправленный вывод.

Пример 1. Блок ввода/вывода семейства Spartan (рис. 9.15).

Режим вывода. Режим вывода обслуживается выходным буфером 1, триггером 1, мультиплексорами 1, 2, 5 и логической схемой ИЛИ. Сигнал O (Out) можно выводить в прямой или инверсной форме в зависимости от программирования мультиплексора 2. Этот сигнал может передаваться на выходной буфер непосредственно или с триггера при соответствующем программировании мультиплексора 5. Сигналы T и GTS (Global Tri State) согласно логике ИЛИ управляют переводом буфера в третье состояние, причем полярность активного уровня сигнала T программируется с помощью мультиплексора 1. Внутренние программируемые цепи триггера (на рисунке не показаны) позволяют изменять полярность тактирующего фронта. Сам буфер имеет программируемые крутизну фронта выходного сигнала и его уровни (КМОП/ТТЛ). Крутизна фронтов в некритичных к скорости передачи цепях снижается для уменьшения уровня помех на шинах питания и земли. Используется так называемый *мягкий старт* (Soft Start Up), снижающий помехи при конфигурировании схемы и ее переходе к рабочему режиму, когда одновременно активизируются многие буферы. Первая активизация автоматически происходит с пологими фронтами перепадов напряжения. Затем вступает в силу заданный выбор крутизны фронтов в зависимости от принятой конфигурации БВВ.

Режим ввода. Тракт ввода содержит входной буфер 2, триггер 2, программируемые мультиплексоры 3, 4, 6, элемент задержки DEL и программируемые схемы (Pull-Up/Pull-Down) задания определенных потенциалов выводу, к которому не подключен информационный сигнал.

Вводимый сигнал в зависимости от программирования мультиплексоров 3 и 4 поступает непосредственно в FPGA (по входным линиям I1 и I2) или же фиксируется триггером и с его выхода передается в эти линии. Триггеры могут конфигурироваться как тактируемые фронтом или как защелки. Выбор осуществляется при-

своением триггеру соответствующего библиотечного символа. В цепи передачи сигнала на триггер 2 могут быть включены элементы задержки DEL (при передаче сигнала через нижний вход мультиплексора 6). Включение задержки гарантирует необходимые временные соотношения между входными сигналами триггера D и глобальным сигналом тактирования.

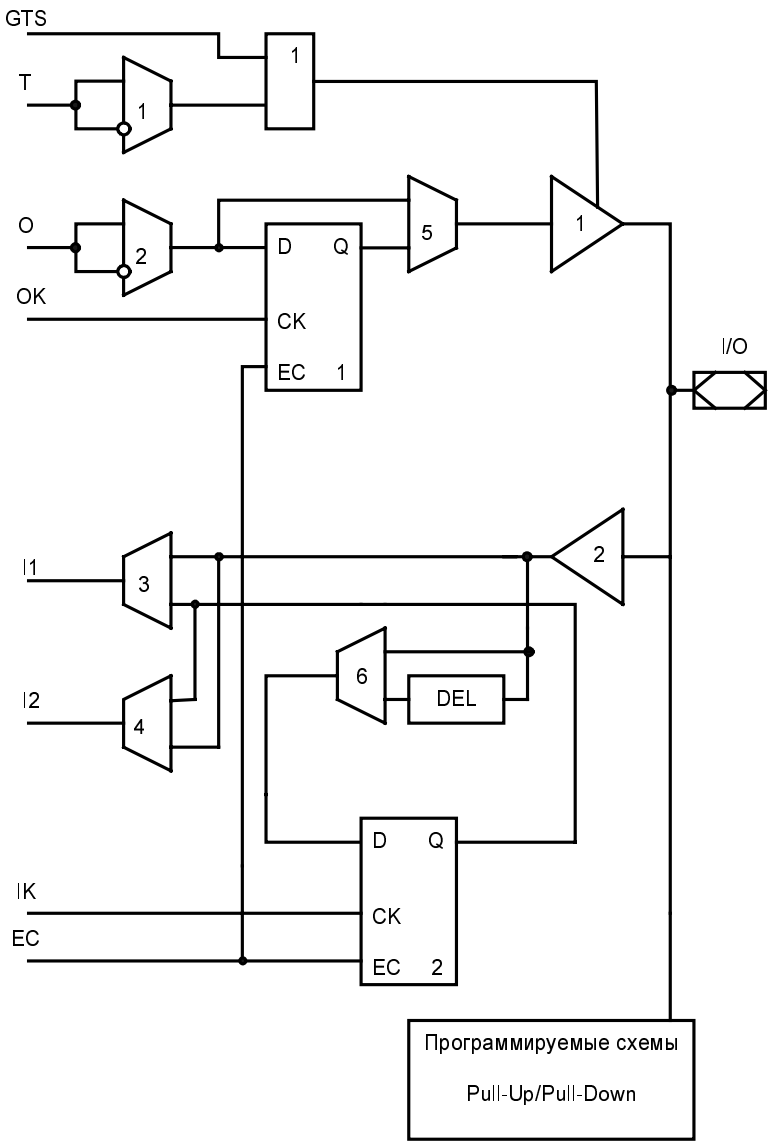


Рис. 9.15. Пример схемы блока ввода/вывода FPGA

Входной буфер 2 может конфигурироваться для восприятия входных сигналов с пороговым значением ТТЛ (1,2 В) или КМОП (0,5 U_{cc}). Выходные уровни тоже

конфигурируются, две глобальные регулировки входных порогов и выходных уровней независимы.

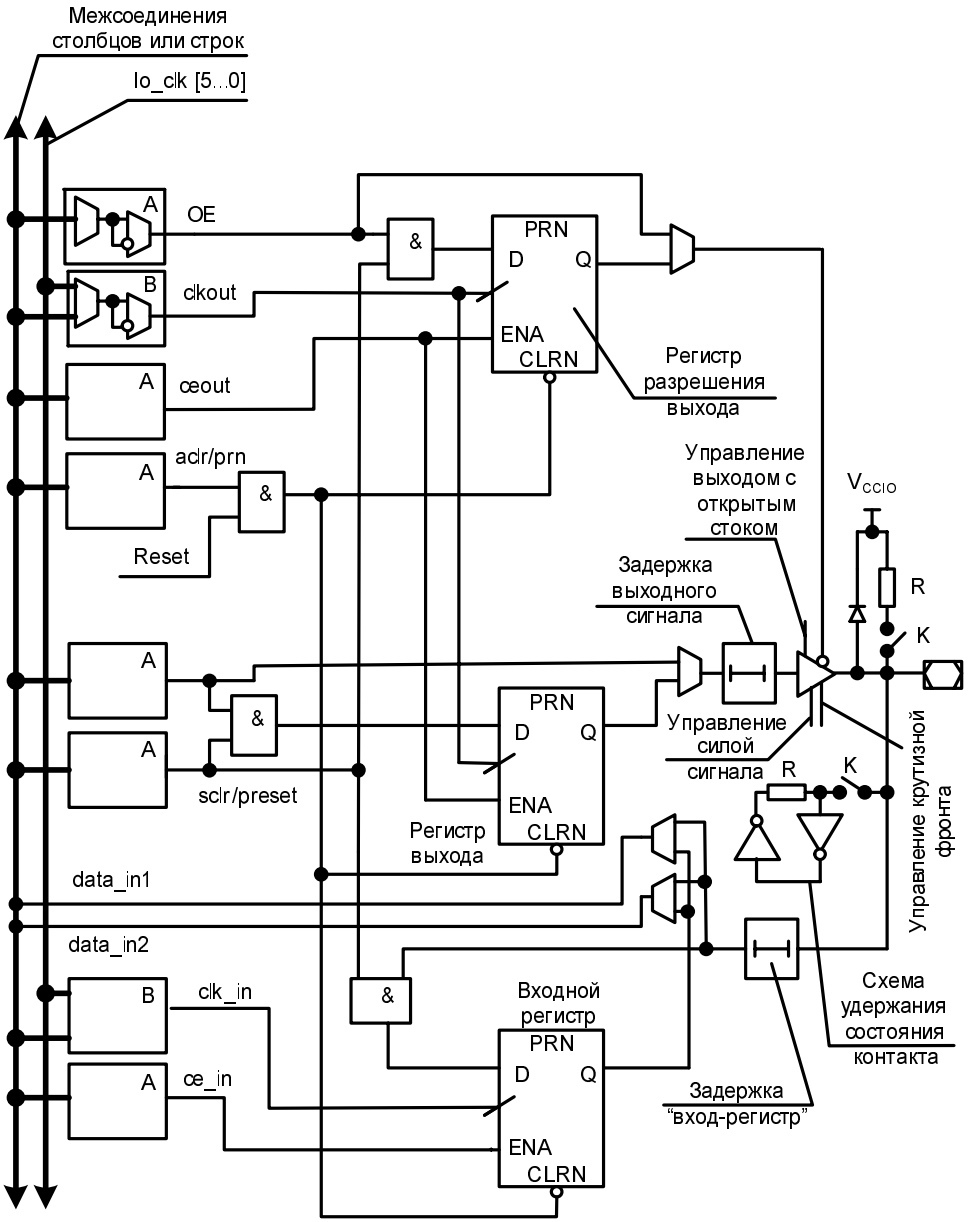


Рис. 9.16. Схема блока ввода/вывода

Пример 2. Блок ввода/вывода семейства Cyclone (рис. 9.16). Ввод/вывод сигналов может осуществляться для ряда стандартов однополюсных и дифференциаль-

ных передач. Поддерживается интерфейс JTAG. Реализуется подтягивание потенциалов выводов (Pull-Up) слабым сигналом при конфигурировании схемы и программируемое подтягивание в рабочем режиме. Кроме того, имеются схемы удержания контактов в прежних состояниях при снятии информационных сигналов. Выходы буферных каскадов могут конфигурироваться как выходы с открытым стоком или как выходы с третьим состоянием.

Блок имеет разнонаправленные буферы и три регистра (входной, выходной и разрешения выхода). Организуются передачи с одинарной скоростью (SDR, синхронизация фронтами одного знака). Схемы, обозначенные буквами A и B, позволяют выбирать используемые блоком сигналы управления и программировать их полярность. Назначение других схем указано на рисунке либо соответствует рассмотренным ранее вариантам блоков ввода/вывода.

§ 9.3. Ресурсы памяти

FPGA обладают ресурсами распределенной и встроенной памяти. В их схемах для реализации функций применяются LUT-блоки, пригодные и для обычного хранения данных. Это *распределенная* память. Малая емкость LUT-блоков (типично 16 бит, максимально 64 бита) не препятствует решению с их помощью локальных задач, таких, например, как буферизация данных в трактах передач, но построение на LUT-блоках емких модулей памяти затруднительно. По мере развития FPGA в них стали широко применяться *встроенные* блоки памяти, емкость которых на порядки превышает емкости блоков распределенной памяти.

Распределенная память

В блоках распределенной памяти запись обычно тактируется, а чтение производится асинхронно, хотя можно организовать и тактируемое чтение с использованием триггеров, не входящих в LUT-блоки. Каждый LUT-блок может быть организован как однопортовая память 16×1, а два блока — как двухпортовая память 16×1 с одним портом для чтения/записи и другим только для чтения. Потоки данных в блоках распределенной памяти показаны на рис. 9.17.

Любая операция по входу данных и выходу SPO (Single-Port Output) может производиться одновременно с чтением из порта DPO (Dual-Port Output) и независимо от него. При записи сигнал разрешения позволяет фронту синхросигнала записать данные в адресованную ячейку памяти. Операция чтения аналогична обращению к комбинационной схеме, она асинхронна, и время доступа определяется лишь задержками элементов LUT-блока. Имеются два адресных входа, верхний принимает адрес записи и чтения, нижний — независимый адрес только для чтения.

Упрощенная схема двухпортового блока показана на рис. 9.18.

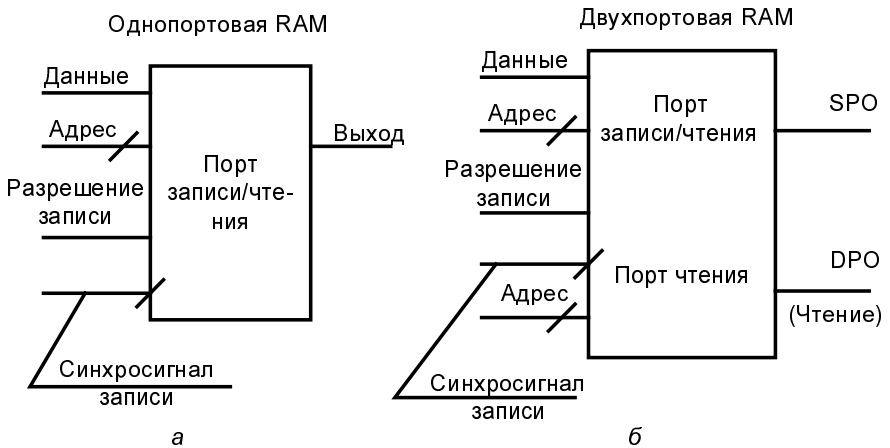


Рис. 9.17. Информационные потоки в однопортовых (*а*) и двухпортовых (*б*) блоках распределенной памяти

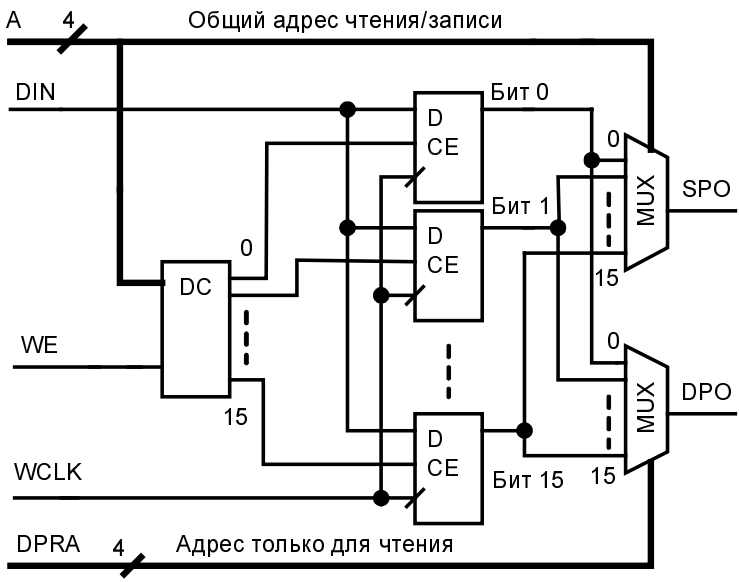


Рис. 9.18. Схема двухпортового блока распределенной памяти

Примеры применения двухпортовой памяти. При построении ЦУ встречается немало ситуаций, в которых недостаточно одного порта доступа к памяти. Яркий пример такой ситуации — *буфер FIFO*, имеющий два адреса (для чтения и записи). Применение однопортовой памяти в этом случае требует мультиплексирования обоих адресов на один порт, а это усложняет схему и исключает одновременность операций чтения и записи. Двухпортовая память дает более эффективное решение. По адресу A может производиться запись. Одновременно может выполняться и чтение по адресу DPRA с выводом данных через порт DPO. Эти операции независимы и могут быть одновременными.

Еще одно направление использования LUT-блоков — построение регистров. Покажем для примера реализацию *сдвигового регистра с динамической регулировкой его длины*. Представим LUT-блок в виде мультиплексора (рис. 9.19, *а*), функционирование которого соответствует таблице прошивки памяти LUT-блока (рис. 9.19, *б*). Регистр (рис. 9.19, *в*) составлен из двух LUT-блоков: 16 триггеров одного образуют регистр, а с помощью второго выполняется мультиплексор, адресный вход которого можно рассматривать как регулятор длины регистра. Установка того или иного значения адреса A выбирает разряд регистра, играющего роль выхода, т. е. и длину регистра.

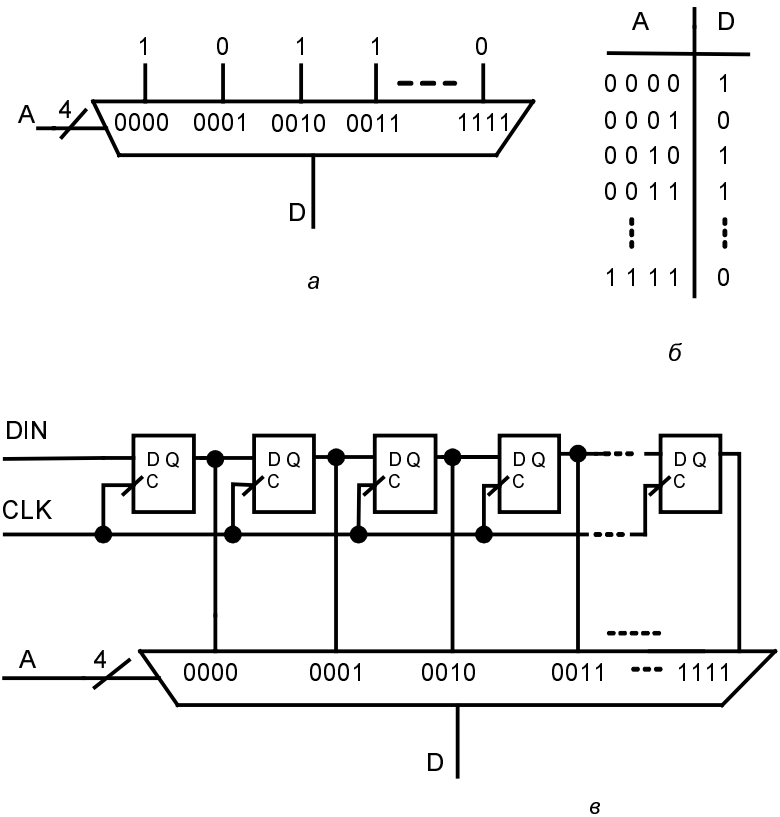


Рис. 9.19. Представление LUT-блока в виде мультиплексора (*а*, *б*) и схема сдвигового регистра с динамической регулировкой длины (*в*)

Наращивание разрядности и емкости блоков памяти. Организация памяти 16×1 непосредственно соответствует структуре LUT-блока. Разрядность памяти наращивается параллельным соединением нужного числа блоков по входам и трактовкой выходных линий блоков как разрядов выходного слова (рис. 9.20, *а*). Модули с расширением по числу хранимых слов строятся способом, аналогичным рассмотренному ранее способу увеличения числа аргументов функции — возникающий при удвоении числа хранимых слов дополнительный разряд адреса подается на мультиплексор, выбирающий выход одного из двух модулей мень-

шей размерности (рис. 9.20, *б*). Организация хранимых данных для этого способа показана на рис. 9.20, *в*.

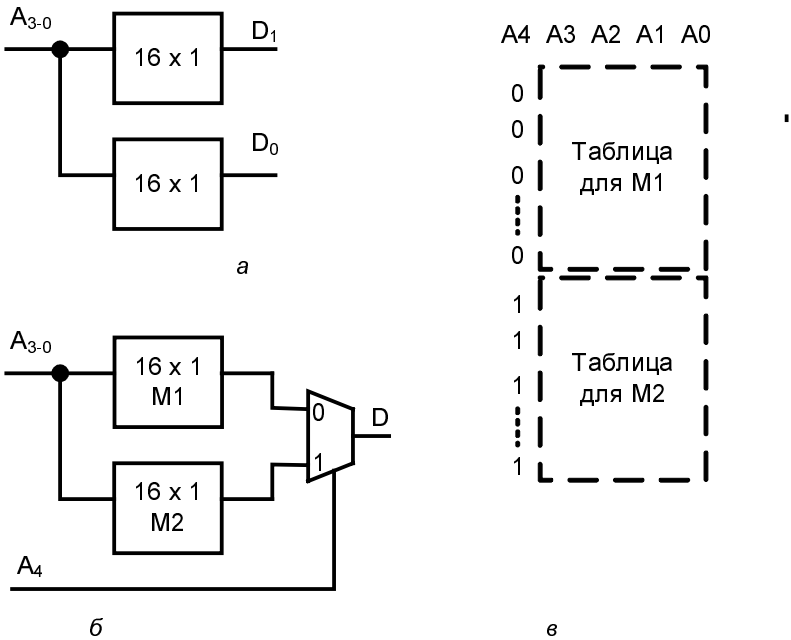


Рис. 9.20. Наращивание разрядности (*а*) и числа хранимых слов (*б*) в модулях памяти, организация хранимых данных при наращивании числа слов (*в*)

Встроенная память

Однопортовый режим. Пример организации встроенного блока памяти (ВБП) емкостью 2 Кбита для однопортового режима показан на рис. 9.21. В структуре ВБП кроме модуля RAM/ROM имеется несколько синхронных D-триггеров и программируемых мультиплексоров. Регистры 1 и 2 программируются для передачи в модуль данных и адресов разной разрядности в зависимости от заданной конфигурации памяти. В блоке с емкостью 2 Кбит разрядность данных может изменяться от 1 до 8, а разрядность адреса от 11 до 8. Запись в память в зависимости от программирования мультиплексоров 4…6 может быть синхронной (от регистров по сигналам тактирования) или асинхронной (непосредственно от входов данных).

Полярность сигналов управления регистрами 1—3 можно выбирать (мультиплексоры 1—3). Выходные сигналы блока могут выдаваться в тактируемом или асинхронном вариантах.

Двухпортовый режим. Блокам встроенной памяти обычно присущ и двухпортовый режим, внешняя организация которого показана на рис. 9.22.

Физически блок имеет два независимых структурно идентичных порта чтения и записи (A и B). Оба порта тактируются своими синхросигналами (CLKA и CLKB) и

имеют свои сигналы разрешения работы (ENA и ENB) и разрешения записи (WEA и WEB). Кроме того, порты имеют входы SSRA и SSRB синхронных сброса/установки, *действующие на регистры-защелки выходных данных*, адресные шины ADDRA и ADDRB, шины входных данных DIA и DIB, шины для битов паритета DIPA и DIPB, а также выходы данных DOA и DOB и битов паритета DOPA и DOPB. Для управляющих сигналов предусматривается выбор их полярностей.

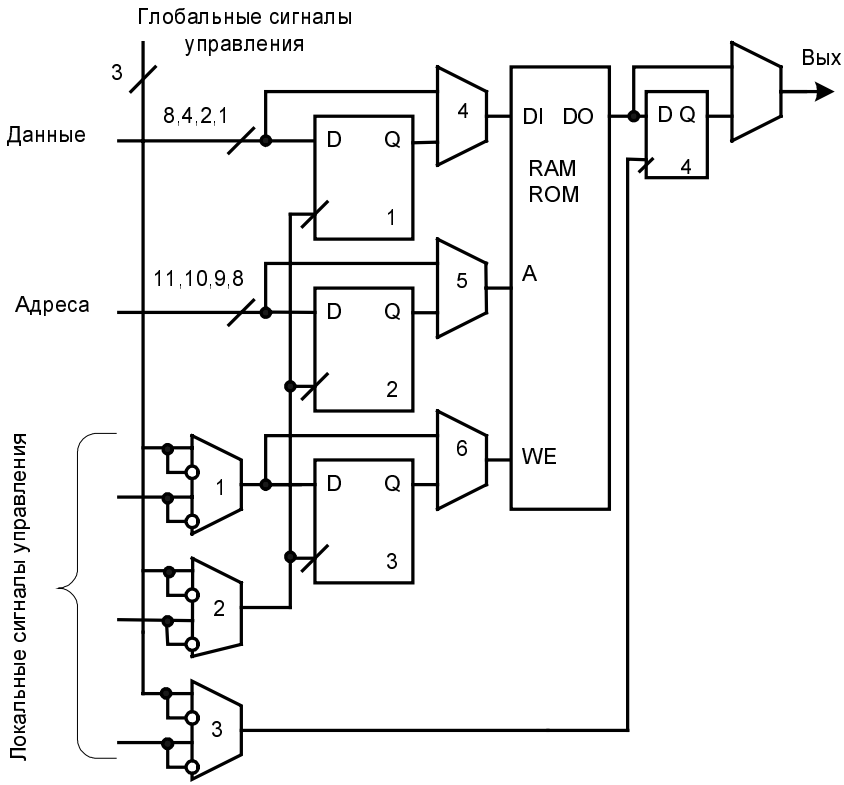


Рис. 9.21. Структура ВБП для однопортового режима

Разрядности шин данных, адресов и тракта битов паритета задаются конфигурированием схемы. В блоках с емкостью 18 Кбит, в частности, предусматриваются разрядности 1, 2, 4, 8, 9 (8 для данных плюс 1 для бита паритета), 16, 18 (16 + 2), 32, 36 (32 +4) и 72 (только для однопортовой организации). Разрядности шин адреса изменяются согласно числу хранимых в блоке слов, зависящему от принятой разрядности данных. Схемы адресации в двух портах могут быть различными в связи с разными разрядностями хранимых данных.

Физически двухпортовый блок моделирует и однопортовый с набором сигналов, идентичным набору каждого из двух портов, показанных на рис. 9.22, т. е. с сигналами WE, EN, SSR, CLK, ADDR, DI, DIP, DOP, DO.

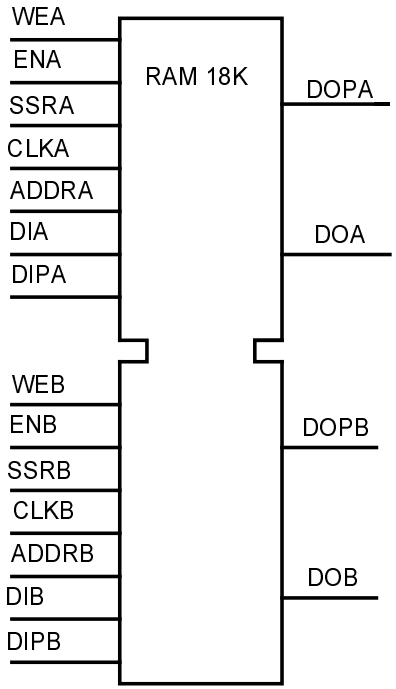


Рис. 9.22. Структура встроенного блока двухпортовой памяти

Оба порта имеют доступ к одному и тому же содержимому памяти, но могут иметь различные схемы адресации, зависящие от разрядностей данных порта. Реализуются следующие операции и потоки данных (рис. 9.23):

1. Операции 1 (2). Порт A (B) ведет себя как независимая однопортовая RAM, выполняющая операции чтения и записи с использованием одного набора адресных линий.
2. Операции 3 (4). Порт A (B[UE4]) является портом записи с отдельным адресом записи, а порт B (A) есть порт чтения с отдельным адресом чтения. Разрядность данных для портов может быть различной.

Записи соответствует сигнал WE, а при его отсутствии выполняется чтение. Однако существуют и такие режимы, в которых запись сопровождается некоторыми возможностями одновременного чтения. С помощью атрибутов системы проектирования можно задавать блокам ВБП различные варианты поведения (Write_Modes), определяющие информацию в адресованной ячейке памяти и регистре-защелке на выходе блока. В зависимости от поведения выходной защелки при записи данных различают варианты: WRITE_FIRST, READ_FIRST и NO_CHANGE (рис. 9.24).

В варианте WRITE_FIRST (режиме прозрачности) (рис. 9.24, *а*) данные со входов DI, DIP записываются в выбранную ячейку и одновременно появляются на выходах DO, DOP. При одновременном доступе по одному адресу выходные данные другого порта становятся недействительными.

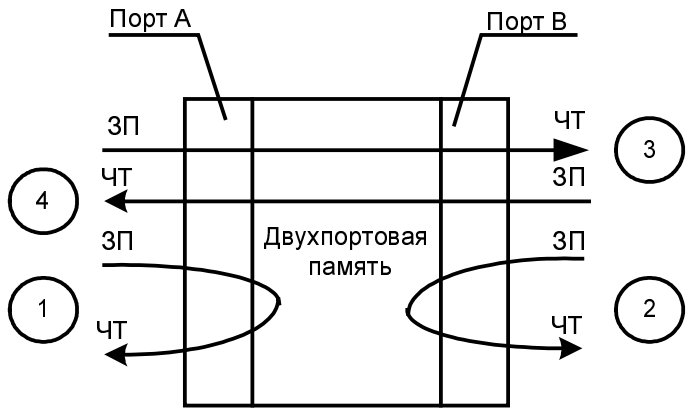


Рис. 9.23. Операции и потоки данных в двухпортовом ВБП

В варианте READ_FIRST (Read-before-Write) (рис. 9.24, *б*) данные, хранившиеся в ячейке по адресу записи, появляются в выходном регистре-защелке (на выходах DO, DOP) в то время как данные со входов DI, DIP записываются в адресованную ячейку. На выходах DO, DOP другого порта появляются те же данные из выбранной ячейки. Этот режим считается наиболее эффективным для увеличения полосы пропускания памяти и полезен для большинства применений памяти в проектах. Он поддерживает одновременность операций чтения и записи по одному и тому же адресу и не создает трудностей согласования задержек в цепях устройств. Режим хорошо соответствует задачам построения многоразрядных регистров сдвига, кольцевых буферов, фильтров с конечной импульсной характеристикой и т. д.

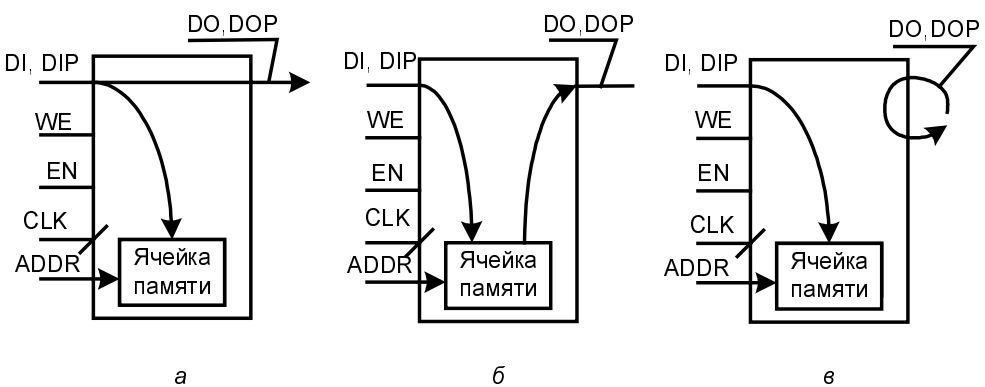


Рис. 9.24. Варианты поведения двухпортового ВБП при записи данных. Режимы WRITE_FIRST (*а*), READ_FIRST (*б*) и NO_CHANGE (*в*)

Режим NO_CHANGE (рис. 9.24, *в*) предусматривает запрещение изменения состояния выходной защелки порта при записи данных. При одновременном доступе по одному адресу данные на выходе другого порта становятся недействительными. Содержимое памяти может быть обновлено без влияния на выходы схемы. Режим

полезно применять в тех случаях, в которых память хранит функциональные таблицы, формы генерируемых колебаний и т. п.

Встроенные блоки памяти в микросхемах разных фирм в основных чертах сходны друг с другом. Они реализуют несколько вариантов операций двухпортовой памяти, поддерживают режимы ROM, FIFO, сдвигающего регистра, тактируемой записи и, при необходимости (например, для конвейерных структур), тактируемый выход и т. д. Организация памяти задается при конфигурации блоков, разрядность которых можно менять в широких пределах. Различаются следующие режимы:

- *однопортовый* (Single-port) — операции чтения и записи выполняются не одновременно;
- *простой двухпортовый* (Simple dual-port), допускающий одновременность операций чтения и записи;
- *истинно двухпортовый* вариант, выполняющий любую комбинацию операций — два чтения, две записи, одно чтение и одну запись. Каждый порт имеет свой синхросигнал. Две операции записи по одному и тому же адресу дают недействительный результат — содержимое адресованной ячейки неизвестно. Чтобы избежать этого, операции одновременной записи специально разделяют по времени с помощью внешних средств, разнося во времени фронты синхросигналов обоих портов.

Применение встроенных блоков памяти

Трудно перечислить все возможные случаи применения встроенных блоков памяти при построении цифровых устройств, т. к. память реализует табличные методы решения задач, обладающие широкими возможностями. Рассмотрим только некоторые примеры.

Построение кругового буфера. Модель кругового буфера приведена на рис. 9.25, *а*. Входные данные записываются в буфер и через *n* тактов появляются на выходе, причем в эту же ячейку записываются новые данные. Круговые буферы находят применение в цифровых фильтрах, корреляторах, схемах задержки данных и др.

Реализация кругового буфера с помощью блока памяти, работающего в режиме READ_FIRST, показана на рис. 9.25, *б*. Каждый синхроимпульс CLK записывает в блок новые данные. Задержанные данные также читаются по этим синхроимпульсам. Счетчик инкрементируется, увеличивая на единицу адрес обращения к памяти, что соответствует подаче следующей ячейки на позицию чтения-записи.

С помощью блока двухпортовой памяти просто реализуются круговые буферы с разными разрядностями данных на входе и выходе [XXXIV, Xilinx Application Notes № 463].

Построение многоразрядного счетчика. Счетчик под воздействием входного сигнала инкрементирует свое состояние. Следующее состояние зависит только от предыдущего. Такой процесс реализуется с помощью памяти ROM, в которую за-

писывается таблица состояний, далее память работает без режима записи. Счетчик начинает работать с нулевого состояния (если не предусматривается иное). Это состояние является адресом, по которому в памяти записана единица. Операция чтения даст на выходе единицу, которая явится адресом для следующего считывания. По единичному адресу записана двойка и т. д. Таким образом, при подаче очередного входного сигнала состояние счетчика изменяется на единицу. В конце счета вырабатывается сигнал, который возвращает счетчик в нулевое состояние.

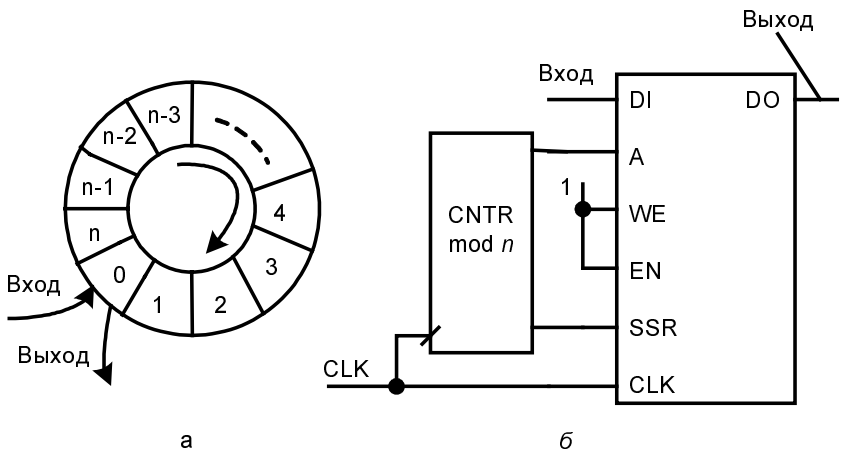


Рис. 9.25. Модель кругового буфера (*а*) и его реализация с помощью блока памяти (*б*)

Записав таблицу состояний в двухпортовую память, можно использовать ее как для формирования младших разрядов счетчика, так и для формирования старших, так что память с числом хранимых слов 2^n позволит построить счетчик с разрядностью 2n, а не n, как было бы при применении однопортовой памяти.

В схеме 20-разрядного счетчика на основе двухпортовой памяти с организацией 1К×18 (рис. 9.26) порт B образует счетчик для десяти младших разрядов выходного кода. Эти десять разрядов, отображающие текущее состояние счетчика, непосредственно соединяются с десятью адресными входами $A_B[9:0]$. Следующее состояние считывается по текущему данному, и представляет собой инкрементированное текущее. Одиннадцатый разряд $DO_B[10]$ фиксирует конец счета (сигнал TC) для счетчика младших разрядов, который сбрасывается. Одновременно сигнал TC на один такт разрешает работу порта A, т. е. счетчика для десяти старших разрядов выхода.

Работа счетчика, реализованного с помощью порта A, отличается от работы счетчика на основе порта B, только тем, что он инкрементируется с частотой в 1/1024 от частоты порта B, что и должно быть для выработки следующих по старшинству десяти разрядов. Разряды от 17-го до 11-го, имеющиеся в памяти, не используются. Десятый разряд порта A может быть использован для выработки сигнала TC старшего счетчика при дальнейшем наращивании разрядности счетчика.

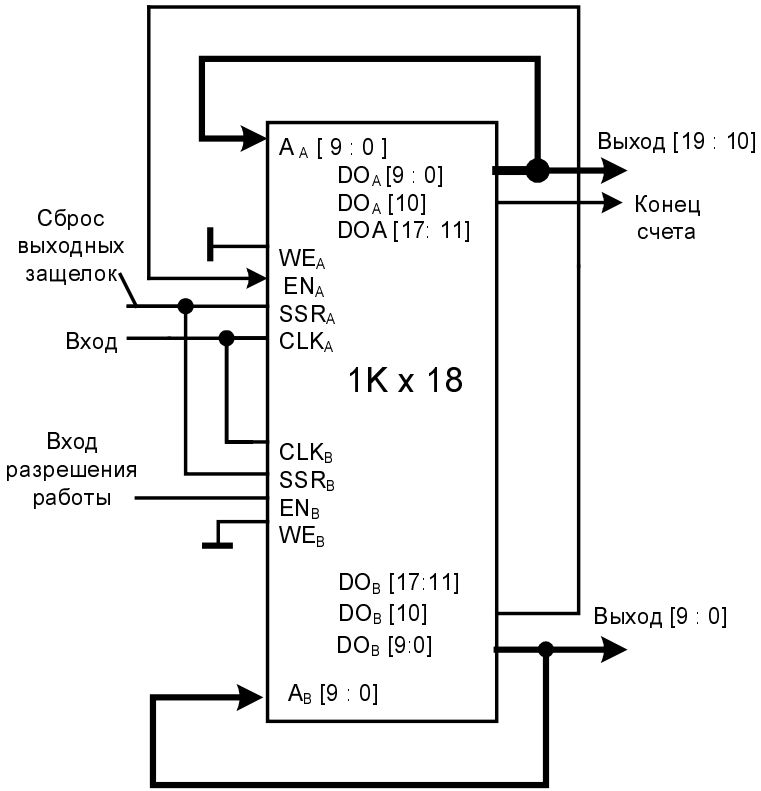


Рис. 9.26. Схема счетчика, реализованного в блоке памяти

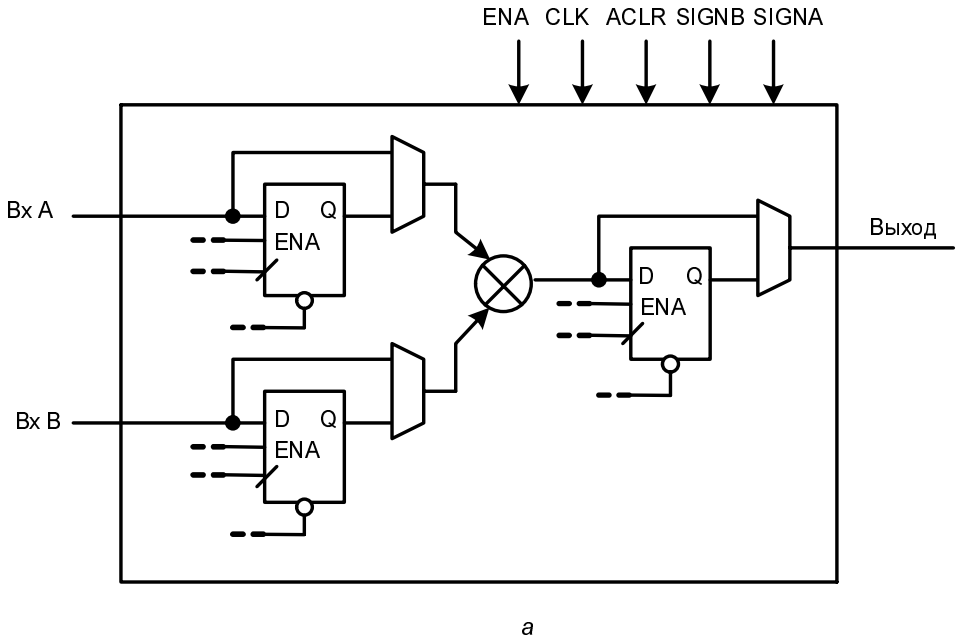
§ 9.4. Умножители и блоки ЦОС

Встроенные умножители стали вводить в состав FPGA со времени достижения ими уровня средней сложности. В FPGA высшей сложности, ориентированные на решение задач обработки сигналов, вводятся блоки цифровой обработки сигналов, выполняющие базовые операции ЦОС (умножение-накопление и др.).

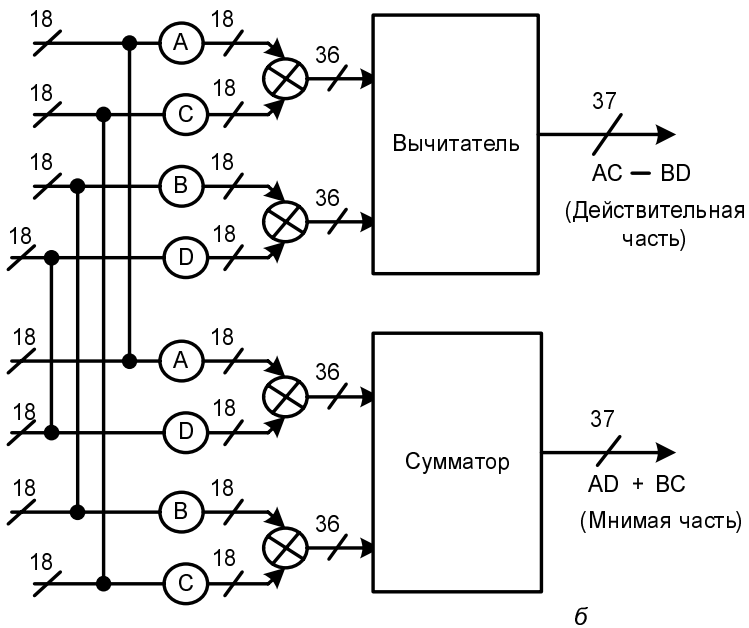
Умножители

Умножители ориентированы на решение задач цифровой обработки сигналов и обычно кроме блока умножения содержат также входные и выходные регистры (рис. 9.27, *а*).

Сигналы управления умножителями: тактовый сигнал CLK, сигнал разрешения тактирования ENA, сигнал асинхронного сброса ACLR и сигналы SIGNA и SIGNB, определяющие представление операндов как чисел со знаками или без, причем это определение индивидуально для обоих сомножителей.



а



б

Рис. 9.27. Структуры умножителей для действительных (*а*) и комплексных (*б*) чисел

В зависимости от программирования мультиплексоров входные переменные и результат операции могут быть представлены в комбинационном или регистровом вариантах. Вычисляется произведение полной или сокращенной точности.

Обработка сигналов в частотной области требует *действий над комплексными числами*. И сложение и умножение комплексных чисел сводятся к обычным сложению и умножению для действительных и мнимых составляющих комплексных величин. Блок умножения комплексных чисел показан на рис. 9.27, *б*. Умножение выполняется по соотношению

$$(a + jb)(c + jd) = [(ac - bd) + j(ad + bc)].$$

Для сложения и вычитания выходных сигналов умножителей применяются блоки сумматора и вычитателя.

Основные операции обработки сигналов

Многие FPGA высокой сложности имеют в своем составе блоки, выполняющие операции цифровой обработки сигналов — *ЦОС* (*DSP*, Digital Signal Processing). Блоки ЦОС содержат большое число умножителей, сумматоров/вычитателей и аккумуляторов, благодаря чему при обработке сигналов достигается высокий уровень параллелизма и, соответственно, производительности. Производительность специализированных блоков при решении задач ЦОС может многократно превышать производительность процессоров цифровой обработки сигналов.

К основным направлениям обработки сигналов относятся: фильтрация, спектральный анализ, нелинейные преобразования. Структуры блоков ЦОС подчинены решению перечисленных задач.

Типы сигналов. В теоретических исследованиях и технике применяются сигналы трех типов:

- *аналоговые*;
- *дискретные*;
- *цифровые*.

В системах ЦОС *аналоговые сигналы* являются входными, поступающими от физических объектов, или выходными, поступающими на такие объекты. Аналоговый сигнал характеризуется *непрерывностью существования*. В любой момент времени можно считывать значение сигнала, причем оно может принимать любые значения в пределах диапазона от минимума до максимума.

Дискретный сигнал, соответствующий аналоговому, отображается совокупностью отсчетов, взятых через равные интервалы времени (имеется в виду *равномерная дискретизация*, применяются и более сложные ее виды). Переход к представлению аналогового сигнала отдельными отсчетами называют его *дискретизацией*.

> Между точками отсчетов сигнал не определен, поэтому одному и тому же дискретному сигналу может соответствовать бесконечно большое число аналоговых сигналов. Иными словами, информация об исходной аналоговой функции при дискретизации частично теряется. Во избежание существенных потерь информации на соотношение частоты сигнала и частоты дискретизации согласно *теореме Найквиста (Котельникова)* накладываются определенные ограничения.

В *цифровом сигнале* отсчеты выражаются числами ограниченной разрядности и, следовательно, соответствуют аналоговым значениям лишь приближенно. Другими словами, при переходе от дискретного сигнала к цифровому значения отсчетов *квантуются по уровню*. При этом возникают погрешности отображения сигнала (*шум квантования*) и возможны нежелательные специфические явления (*предельные циклы* в рекурсивных фильтрах). Оба этих фактора влияют на выбор разрядностей чисел, отображающих отсчеты.

Дискретные сигналы занимают промежуточное место между цифровыми и аналоговыми. Хотя реально в системе присутствуют только аналоговые и цифровые сигналы, дискретные играют важную роль в качестве *полезной модели*, позволяющей анализировать процессы, отвлекаясь от факторов, второстепенных для определенного этапа описания системы.

Цифровая фильтрация. *Фильтрация* состоит в разделении сигнала на составляющие по частотному признаку. Фильтры низкой частоты (ФНЧ) пропускают сигналы в полосе частот от нулевой до граничной и подавляют сигналы более высоких частот. Фильтры высокой частоты (ФВЧ) выполняют обратную задачу, полосовые фильтры (ПФ) пропускают сигналы с частотами, лежащими в некотором диапазоне, и подавляют сигналы с частотами, расположенными вне этого диапазона.

Для нерекурсивного фильтра (термин *нерекурсивный* отражает отсутствие в схеме обратных связей) процесс линейной фильтрации описывается выражением

$$y(k) = \sum_{i=0}^{n} b_i x(k-i),$$

где $x(k-i)$ — отсчеты входного сигнала, y(k) — отсчеты выходного сигнала, b_i — постоянные коэффициенты. Иными словами, отсчеты выходного сигнала вычисляются в виде суммы *n* взвешенных отсчетов входного сигнала (текущего и предыдущих).

Операция умножения-накопления MAC (Multiply-Accumulation), дающая значение искомой величины $y(k)$, является важнейшей базовой операцией ЦОС.

В схеме *нерекурсивного фильтра* (рис. 9.28) символом Z^{-1} обозначены элементы задержки. Текущий входной отсчет $x(k)$ умножается на коэффициент b_0, предыдущий отсчет $x(k-1)$ умножается на b_1 и т. д. Все *n* произведений складываются сумматором.

Нерекурсивный фильтр называется также фильтром *с конечной импульсной характеристикой (КИХ-фильтром)*. Термин "с конечной импульсной характеристикой" говорит о том, что реакция фильтра на единичный входной импульс ограничена во времени (конечна). Это действительно так, поскольку после прохождения единичного входного воздействия через всю цепочку элементов задержки входы всех блоков $b_i (i = 0...n)$ окажутся нулевыми, и выходной сигнал также станет нулевым.

Для вычисления очередного выходного отсчета можно использовать не только то или иное число входных отсчетов, но и уже имеющиеся выходные отсчеты, определяя отсчет *y(k)* следующим образом

$$y(k) = \sum_{i=0}^{n} b_i x(k-i) + \sum_{j=1}^{m} a_j y(k-j),$$

где a_j — постоянные коэффициенты.

Эта формула описывает работу *рекурсивного* фильтра или *фильтра с бесконечной импульсной характеристикой (БИХ-фильтра)*. Первый вариант названия отражает наличие в схеме обратных связей (выходные отсчеты подаются обратно на входы схемы), а второй говорит о том, что для данной структуры возможны ситуации, когда реакция на входной единичный импульс может длиться сколь угодно долго. Это видно хотя бы из простого примера для фильтра с формулой $y(k) = x(k) + y(k-1)$. После подачи на вход такого фильтра единичного импульса на выходе получим неограниченно возрастающую последовательность отсчетов 1-2-3-4-… При ослаблении обратной связи вдвое, когда $y(k) = x(k) + 0{,}5y(k-1)$, амплитуды выходных отсчетов будут бесконечно приближаться к 2.

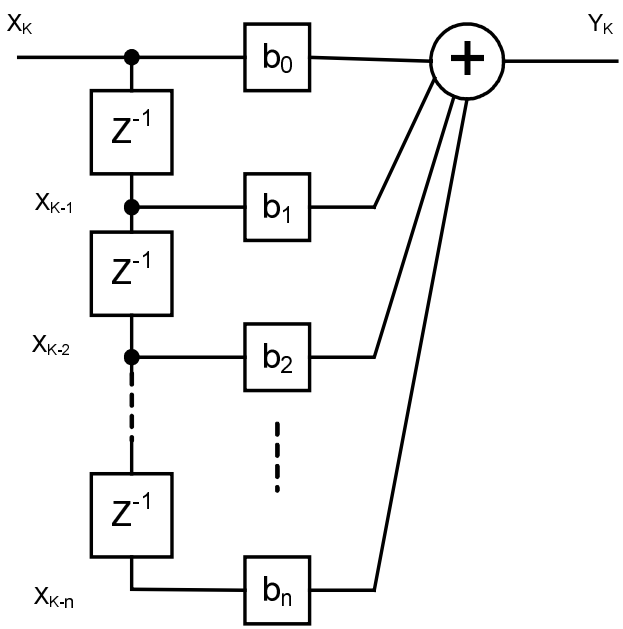


Рис. 9.28. Структура нерекурсивного фильтра (КИХ-фильтра)

После небольших преобразований, основанных на свойствах линейности и стационарности фильтра, схему БИХ-фильтра. можно привести к виду, показанному на рис. 9.29.

Преобразования Фурье. Сигналы могут быть представлены как во временной, так и в частотной области. Периодические аналоговые сигналы, удовлетворяющие условиям Дирихле, могут быть представлены бесконечным гармоническим рядом *(рядом Фурье)*, сумма которого совпадает со значениями сигнала для всех точек его непрерывности, а в точках разрывов первого рода дает среднее арифметическое левого и правого предельных значений.

Для спектрального анализа непериодических аналоговых сигналов пользуются преобразованием Фурье, в результате которого сигнал f(t) представляется непрерывной функцией частоты $\dot{F}(\omega)$, называемой *спектральной функцией* или *спектральной плотностью сигнала.*

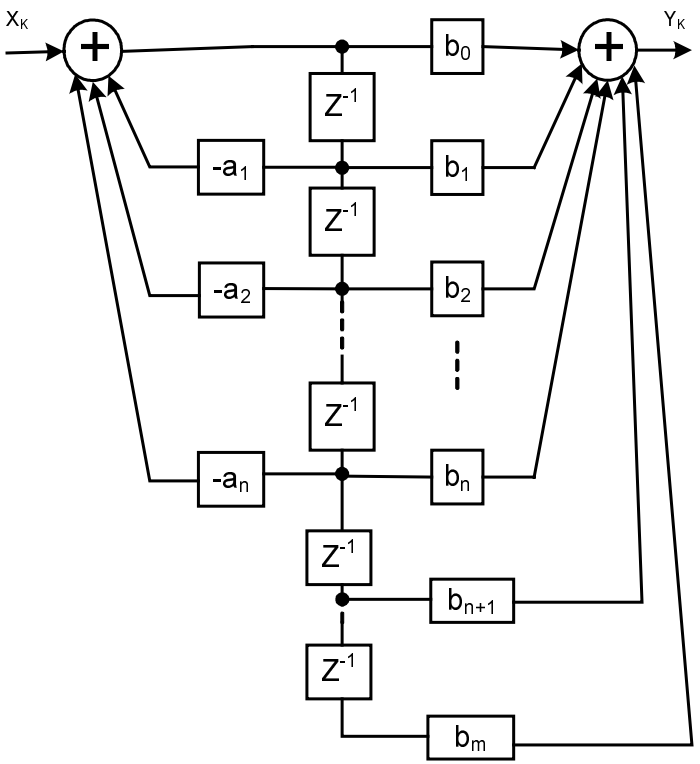


Рис. 9.29. Каноническая структура рекурсивного фильтра (БИХ-фильтра)

При обработке дискретных сигналов, представленных последовательностями отсчетов, применяются дискретные преобразования Фурье, преобразующие группу из N отсчетов сигнала в группу из N комплексных амплитуд его гармоник. *Прямое дискретное преобразование Фурье* (прямое ДПФ) имеет вид:

$$\dot{X}(n) = \sum_{k=0}^{N-1} x(k)\, \dot{W}_N^{nk}, \qquad 0 \le n \le N-1,$$

где k — номер отсчета, n — номер составляющей спектра, $W_N^{nk} = exp(-j\frac{2\pi nk}{N})$ — комплексная экспонента.

Для перехода от дискретного спектра к временным отсчетам сигнала служит *обратное ДПФ*:

$$x(k) = \sum_{n=0}^{N-1} X(n) W_N^{-nk}, \qquad 0 \le k \le N-1.$$

Модуль и аргумент ДПФ образуют амплитудный и фазовый спектры соответственно. ДПФ называют N-точечным по числу вычисляемых составляющих спектра. Рассмотренные преобразования можно применять к периодическим последовательностям и к последовательностям конечной длины, трактуя их как один период и условно дополняя аналогичными периодами для всей временной оси.

Быстрое преобразование Фурье. Расчет ДПФ требует большого объема вычислений (выполнения N^2 операций комплексного умножения и сложения). Для существенного сокращения объема вычислений применяют более рациональную схему вычислений — *быстрое преобразование Фурье (БПФ)*. При этом используется базовая операция БПФ, называемая "*бабочкой*", характер которой поясняется на рис. 9.30.

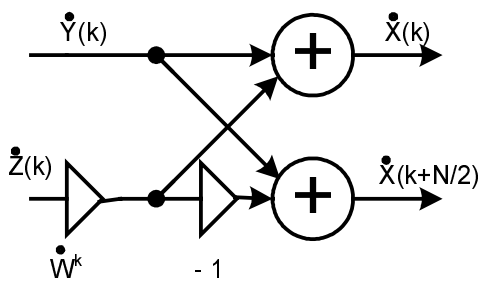


Рис. 9.30. Структура операции "бабочка"

Структура ЦОС-блока

Рассмотрим *ЦОС-блоки* (*DSP-blocks*) микросхемы Stratix фирмы Altera. Эти блоки сгруппированы на кристалле в два столбца. Каждый блок находится в окружении массива программируемой логики, которая может быть использована для дополнительных операций при построении устройств из нескольких блоков (рис. 9.31). У разных представителей семейства от 6 до 28 блоков.

Блоки ЦОС могут выполнять операции умножения с возможностью накопления (суммирования) или вычитания результатов при разрядностях операндов 9, 18 или 36. Вырабатываются произведения полной точности. При размерности 9×9 можно

реализовать в одном блоке до 8 умножителей, при 18×18 — четыре и при 36×36 — один. Суммарно в старшей микросхеме может быть создано до 224 умножителей размерностью 9×9. Блоки ЦОС дают по 2×10^9 операций умножения-накопления в секунду, и если использовать все 28 блоков старшего представителя семейства, то будет достигнута вычислительная мощность в 56×10^9 операций MAC в секунду. Блоки содержат также 18-разрядные сдвигающие регистры. Имеющийся в блоках набор устройств позволяет строить типичные для ЦОС структуры, в частности, цифровые фильтры с конечной или бесконечной импульсной характеристикой и схемы быстрого преобразования Фурье. Максимальная частота работы блоков составляет 250 МГц. Структура блока при конфигурировании умножителей в варианте 18×18 приведена на рис. 9.32.

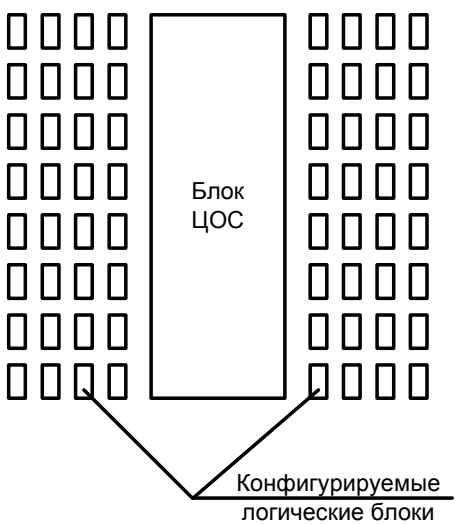


Рис. 9.31. Размещение блока ЦОС на кристалле

Как видно из структур КИХ- и БИХ-фильтров (см. рис. 9.28 и 9.29), важную роль в их схемах играет цепь из элементов Z^{-1} задержки отсчетов (многоразрядных слов). Эта цепь реализована как последовательность статических регистров (в данном случае 18-разрядных), каждый из которых по разрешению тактирующего сигнала принимает информацию от такого же регистра в соседнем верхнем канале.

Таким образом, в цепи из регистров, подключенных к верхним входам умножителей, будут продвигаться сверху-вниз значения отсчетов, как это и предусмотрено структурами КИХ- и БИХ-фильтров.

На вторые входы умножителей должны подаваться значения коэффициентов, взвешивающих отсчеты. Эти коэффициенты загружаются в нижние регистры каналов, причем загрузка может производиться последовательно (по такому же способу продвижения сверху-вниз, но через нижние регистры каналов) или параллельно, поскольку все регистры имеют и входы параллельной загрузки. В пределах одного блока при 18-разрядных операндах реализуются 4 умножителя и ассоциированные

с ними схемы. Для наращивания числа каналов несколько блоков могут соединяться по обозначенным на рисунке цепям входов и выходов. Умножители могут оперировать с числами разных типов (со знаками, без них и т. д.).

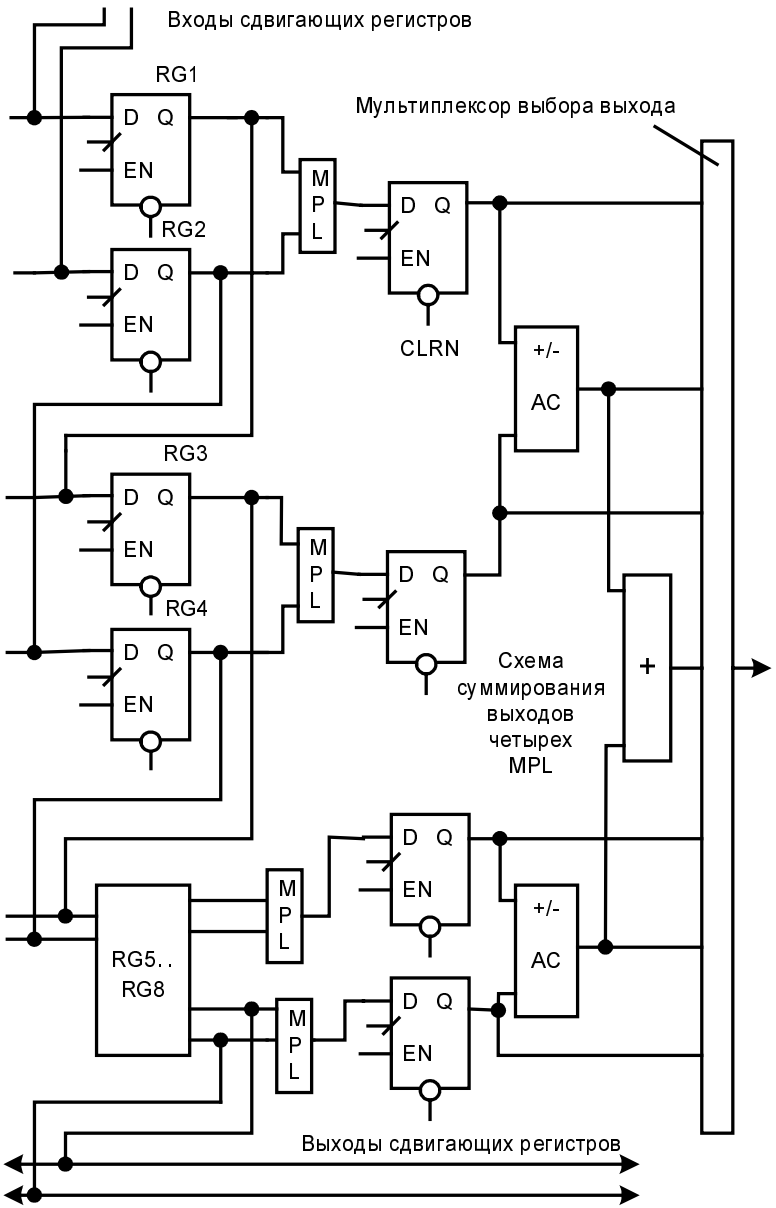


Рис. 9.32. Структура блока ЦОС

Результаты умножений поступают далее на блоки суммирования/вывода (Adder/Output Blocks), состоящие из сумматоров/вычитателей/аккумуляторов, сумми-

рующих схем, мультиплексоров выбора выхода и выходных регистров. Блок суммирования/вывода может направить результат умножения прямо на выход, или на аккумулятор, или для сложения двух произведений, или для сложения четырех произведений. В частности, как видим, блоком выполняется и базовая операция MAC.

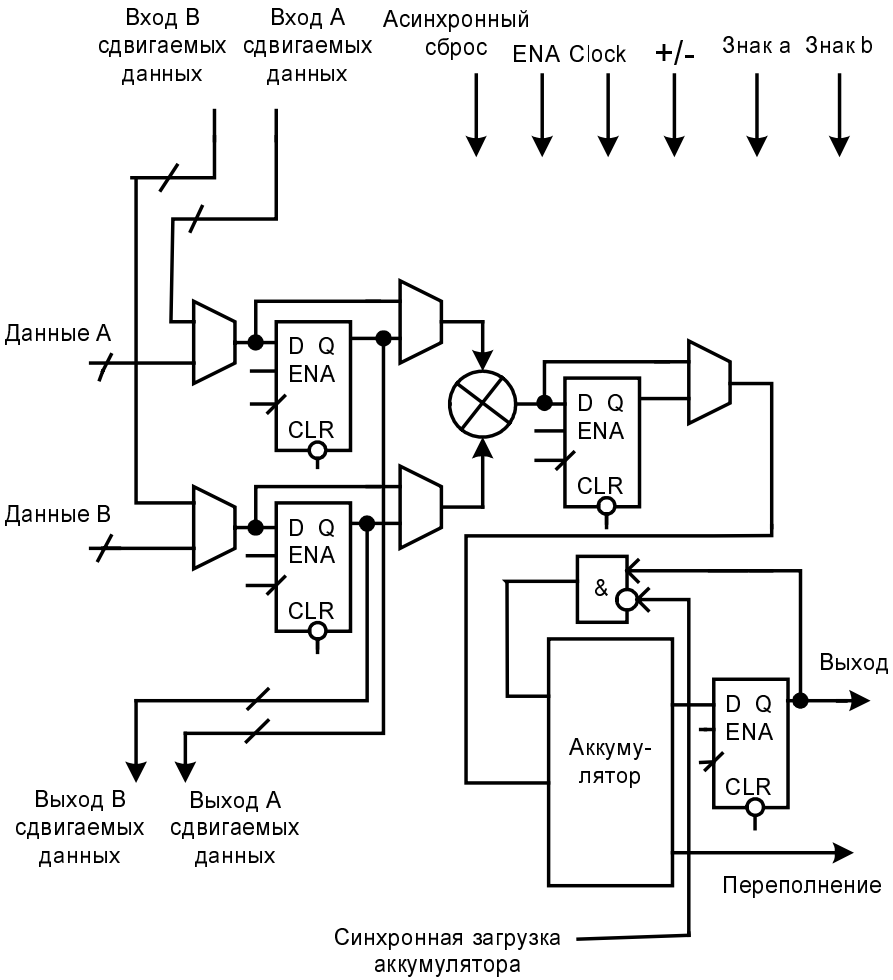


Рис. 9.33. Реализация операции MAC в ЦОС-блоке

Вариант реализации операции MAC в режиме Multiple-Accumulate иллюстрируется рис. 9.33.

В субблоке, реализующем операцию MAC, результат перемножения двух операндов поступает в схему "сумматор/вычитатель/аккумулятор", сконфигурированную как аккумулятор. Выход аккумулятора может иметь до 52-разрядов. Так как результат перемножения 18-разрядных слов дает 36 разрядов, на накопление остается 16 разрядов. Сигналы управления включают в себя асинхронный сброс, синхросигнал (Clock), сигнал разрешения тактирования регистров (ENA), сигналы, задающие

знаки операндов, и сигнал синхронной загрузки аккумулятора. На рис. 9.30 показан сокращенный набор управляющих сигналов (не учтены варианты временного управления сигналами, когда какой-либо один по логическому смыслу сигнал вырабатывается как комбинационный, синхронизированый одноступенчатым регистром, синхронизированный двухступенчатым регистром и т. п.).

§ 9.5. Программируемые аналоговые и аналого-цифровые схемы

Общие сведения. Многие управляющие системы оперируют с информацией, представленной не только в цифровой, но и в аналоговой форме. Аналоговые сигналы вырабатываются датчиками физических величин. Они же используются для воздействия на исполнительные механизмы. При вводе/выводе этих величин в/из ЦУ обычно необходимы операции, обозначаемые термином *End-Front Design* (нормирование, фильтрация, аналого-цифровое и цифроаналоговое преобразования и др.). В последние годы наблюдается устойчивая тенденция к увеличению аналоговой и цифроаналоговой частей в составе систем.

Аналоговые и аналого-цифровые устройства уже давно встраиваются в микропроцессорные системы в виде отдельных микросхем малого и среднего уровней интеграции, использующих дискретные (навесные) операционные элементы. Технология БМК (базовых матричных кристаллов) также применяется в аналого-цифровой технике, но программирование самим пользователем аналоговых и аналого-цифровых схем, целиком реализованных на кристаллах, до последнего времени практически не было освоено. Трудности освоения аналоговых интегральных схем в значительной мере объяснялись их пониженными точностными возможностями в сравнении со схемами на дискретных компонентах. Появление "систем на кристалле" сделало проблему интеграции программируемых аналоговых и аналого-цифровых схем и цифровой части системы особенно актуальной. На основе программируемых структур возможны быстрое проектирование и отладка аналоговых и аналого-цифровых подсистем обработки сигналов и создание промышленных образцов. Несколько крупных фирм (Lattice Semiconductor, Cypress Semiconductor, Anadigm Inc., Fast Analog Solutions) уже отреагировали на требования времени и уделили внимание разработкам программируемых аналоговых и аналого-цифровых структур, выполненных как на отдельном кристалле, так и совместно с цифровой частью системы.

Два варианта интегральных аналоговых схем

Двоичные цифровые сигналы принимают лишь два значения, одно из которых соответствует логической единице, а другое — логическому нулю. Проблема точного задания этих сигналов отсутствует — требуется лишь надежно отличать один из этих сигналов от другого. Совершенно иным является положение в аналоговой

технике, где *сигнал должен передавать точное значение величины с погрешностью в десятые или сотые доли процента*, т. е. требуется "дозирование" сигналов с разрешающей способностью в тысячи или даже более уровней. Традиционно (до конца 70-х начала 80-х гг.) роль дозирующих параметров в схемах нормирующих усилителей, сумматоров и т. п. играли *отношения сопротивлений точных резисторов*. Постоянные времени RC (масштабирующие параметры в схемах интеграторов, фильтров и др.) задавались совместно значениями сопротивления точного резистора и емкости операционного конденсатора. Так, например, в известной схеме масштабирующего усилителя, т. е. устройства умножения сигнала, заданного напряжением постоянного тока, на константу используются два точных резистора R_1 и R_2, от соотношения сопротивлений которых зависит функциональная характеристика схемы, в идеализированном виде имеющая вид

$$U_2 = (-R_2/R_1)U_1,$$

где U_1 и U_2 — входное и выходное напряжения соответственно. Интегратор имеет идеализированную функциональную характеристику вида

$$U_2 = (-1/RC)\int_0^t U_1(t)\,dt,$$

в которой роль масштабирующего коэффициента играет произведение сопротивления резистора входной цепи R на емкость конденсатора цепи обратной связи C.

В схемотехнике с дискретными (навесными) схемными элементами проблема реализации точных резисторов имеет удовлетворительное решение. Для технологии интегральных схем эта проблема намного сложнее, но существует *альтернативное решение*, благодаря которому резисторы имитируются цепями, содержащими коммутируемые (переключаемые) конденсаторы (рис. 9.34, *а*).

В цепь, имитирующую резистор, входят конденсатор C и ключевые транзисторы T1 и T2. Транзисторы T1 и T2 под воздействием тактирующих напряжений U_{t1} и U_{t2} замыкаются поочередно, и конденсатор C попеременно заряжается через замкнутый ключевой транзистор до напряжения U_1 или U_2. В момент коммутации ключевых транзисторов заряд конденсатора изменяется на величину $q = C(U_1 - U_2)$. Изменение заряда осуществляется короткими импульсами тока, протекающими через конденсатор при замыкании соответствующего ключевого транзистора.

Среднее значение тока в цепи между точками 1 и 2 составляет величину

$$i = q/T = (U_1 - U_2)C/T,$$

где T — период тактирующих импульсов.

Из полученного выражения видно, что для средних значений сигналов цепь ведет себя как резистор с сопротивлением

$$R = T/C = 1/Cf,$$

пропорциональным периоду тактирующих импульсов (т. е. обратно пропорциональным их частоте f) и обратно пропорциональным емкости. Разница состоит в том, что резистор регулирует передаваемый через него заряд плавно, а цепь с пере-

ключаемыми конденсаторами — импульсно. При этом резисторы и имитирующие их цепи функционально эквивалентны для *средних* за период T значений токов.

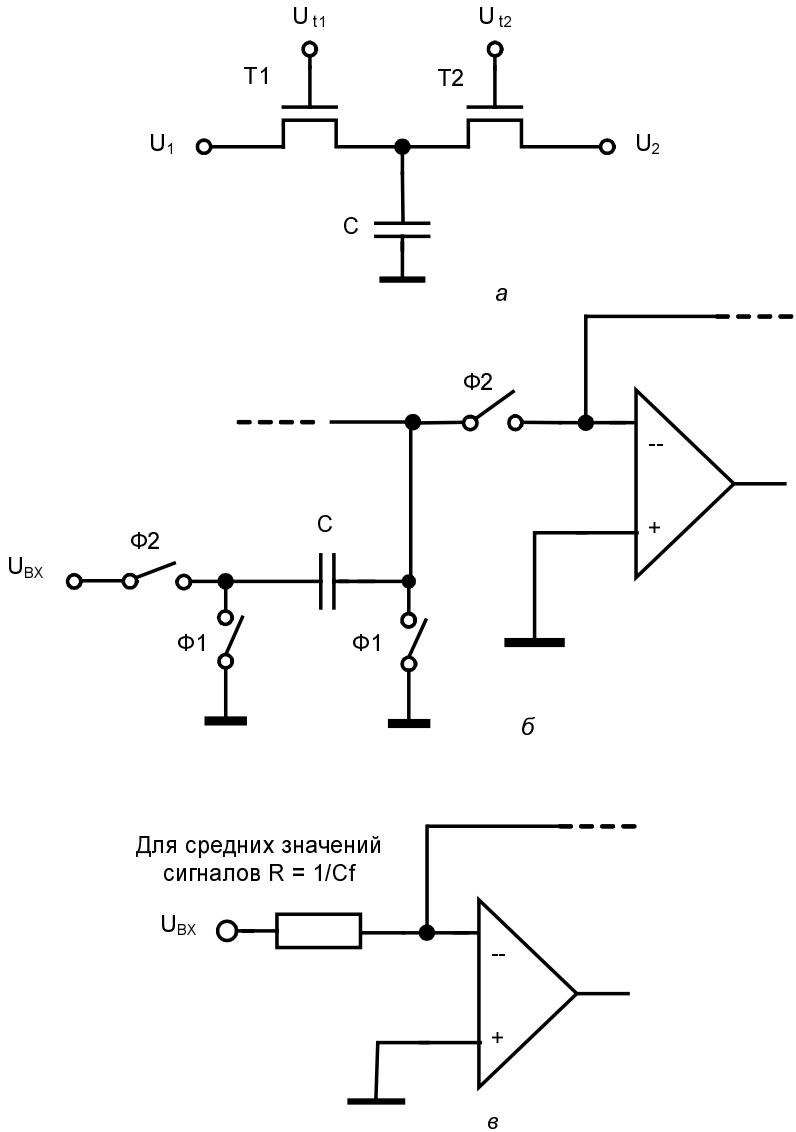


Рис. 9.34. Цепь с переключаемыми конденсаторами, моделирующая резистор (*а*), включение ее во входную цепь операционного усилителя (*б*) и эквивалент цепи для средних значений сигналов (*в*)

На основе схем с переключаемыми конденсаторами *можно строить операционные звенья, аналогичные известным из традиционной аналоговой схемотехники*,

заменяя резисторы эквивалентными им цепями. Сопротивления эквивалентных цепочек управляются значениями тактовой частоты f = 1/T, которую можно изменять и дистанционно. А это ставит аналоговые блоки в один ряд с цифровыми и с точки зрения возможностей программирования в системе на расстоянии.

В ряде схем с переключаемыми конденсаторами *функциональные характеристики зависят только от отношения емкостей,* которое может задаваться с высокой точностью. Параметры емкостей мало критичны к изменению температуры и старению. Резко (в сотни раз) снижается площадь, занимаемая цепями с переключаемыми конденсаторами в сравнении с цепями, содержащими точные резисторы. Таковы технологические достоинства схемотехники переключаемых конденсаторов. В то же время применение цепочек с переключаемыми конденсаторами имеет и свои недостатки. В цепях с непрерывными сигналами (без переключаемых конденсаторов) отсутствует проблема отделения полезной информации (среднего значения пульсирующей величины) от сопровождающих ее паразитных высокочастотных составляющих, что благоприятно влияет на динамические характеристики устройств. Кроме того, схемы с непрерывными сигналами имеют лучшие шумовые характеристики.

На рис. 9.31, *б* показана схема с переключаемыми конденсаторами в цепи входа операционного усилителя (ОУ), также моделирующая резистор для средних значений токов. По такту Ф1 конденсатор C полностью разряжается (обе его обкладки заземляются), а по такту Ф2 разряженный конденсатор включается между входным напряжением и потенциально заземленной точкой операционного усилителя, заряжаясь до напряжения $U_{ВХ}$ и передавая в цепь обратной связи ОУ заряд $CU_{ВХ}$. За секунду через цепи входа и обратной связи будет передан заряд $CU_{ВХ}f$, где f — частота коммутации ключей. Таким образом, цепь с переключаемым конденсатором для средних значений токов будет эквивалентна сопротивлению R = 1/Cf (рис. 9.31, *в*).

Практические разработки

В 1999 г. фирма Lattice Semiconductor выпустила внутрисхемно программируемые аналоговые схемы с масштабирующими резисторами (семейство ispPAC). Фирма Cypress Semiconductor выпустила микросхемы PSoC класса "программируемая система на кристалле" с реализацией цифровой и аналоговой частей на одном кристалле. В микросхемах PSoC применены оба вида схемотехнических решений, т. е. и схемы с масштабирующими резисторами и схемы с переключаемыми конденсаторами.

Микросхемы ispPAC архитектурно просты (имеют немного конфигурируемых ресурсов и контактов ввода/вывода). Память конфигурации EEPROM может загружаться через специально выделенные контакты JTAG интерфейса. Допустимое число циклов репрограммирования не менее 10000. Конфигурация может быть закрыта от несанкционированного доступа битом секретности. В состав конфигурируемых ресурсов включены не только аналоговые, но и цифроаналоговые средства (восьмиразрядные цифроаналоговые преобразователи).

Микросхемы ispPAC (рис. 9.35) содержат входные (инструментальные) усилители IA (Input Amplifiers), выходные (операционные) усилители OA (Output Amplifiers), умножающие цифроаналоговые преобразователи *MDAC* (Multiplying Digital-Analog Converters) и другие блоки.

Микросхема имеет единственное напряжение питания 5 В, режимы понижения мощности (в режиме Power Down потребляются микроваттные мощности), автокалибровку внутренних смещений и точное программирование коэффициентов усиления. Важная особенность — *возможность динамической реконфигурации* "на лету", причем быстрой и без ограничения числа циклов. Для этой цели микросхема снабжена триггерной памятью конфигурации, наряду с которой имеется и энергонезависимая память типа EEPROM, постоянно хранящая настройки, из которых пользователь может выбирать требуемую и вводить ее в триггерную память. Среди хранимых в EEPROM настроек есть такая (Preset), которая вводится при включении питания (или при необходимости вернуться к ней в процессе работы).

Усилители IA имеют входные импедансы в области гигаомных значений, что практически исключает нагрузку на источники сигналов (датчики физических величин и др.). Шкала входных напряжений у этих усилителей от 0 до 2,8 В, коэффициенты усиления программируются в диапазоне от ± 1 до ± 10. Два усилителя имеют на входах мультиплексоры размерности 2—1. Управляя мультиплексорами, можно ввести в схему 4 различных сигнала.

Два выходных усилителя с диапазоном выходных напряжений от 0 до 5 В (в однополюсном режиме) имеют произведение коэффициента усиления на полосу пропускания (Gain-Bandwidth Product) 15 МГц. В зависимости от программирования перемычек в цепях обратных связей выходные усилители могут работать в режимах:

- *усилителя* (отключается емкость, точнее, оставляется минимальная емкость для сохранения устойчивого режима работы);
- *интегратора* (отключается резистор);
- *фильтра низкой частоты первого порядка* (включены оба элемента обратных связей). Для фильтров с различной полосой пропускания можно использовать 7 различных значений емкости C;
- *компаратора* (отключены оба элемента обратных связей). Отсутствие у OA двух сигнальных входов заставляет выполнять компаратор в виде сумматора входного напряжения и опорного напряжения другого знака. При этом знак суммы определяет логический выход OA. Для компараторов возможно программирование работы с гистерезисом или без него.

Выходные сигналы MDAC с высокой точностью пропорциональны как входному цифровому коду, так и входному напряжению. Входное напряжение умножается при этом на коэффициент, не превышающий 1, так что на выход MDAC в зависимости от поданного кода передается часть входного сигнала (от 100% до части, соответствующей единице младшего разряда кода). Являясь регулируемым аттенюатором входного напряжения, MDAC обеспечивает необходимые коэффициенты передачи в соответствующих цепях схем. Сочетание MDAC, входных усилителей и

источников опорных напряжений V_{REF} предоставляет для этого широкие возможности (иллюстрируется далее).

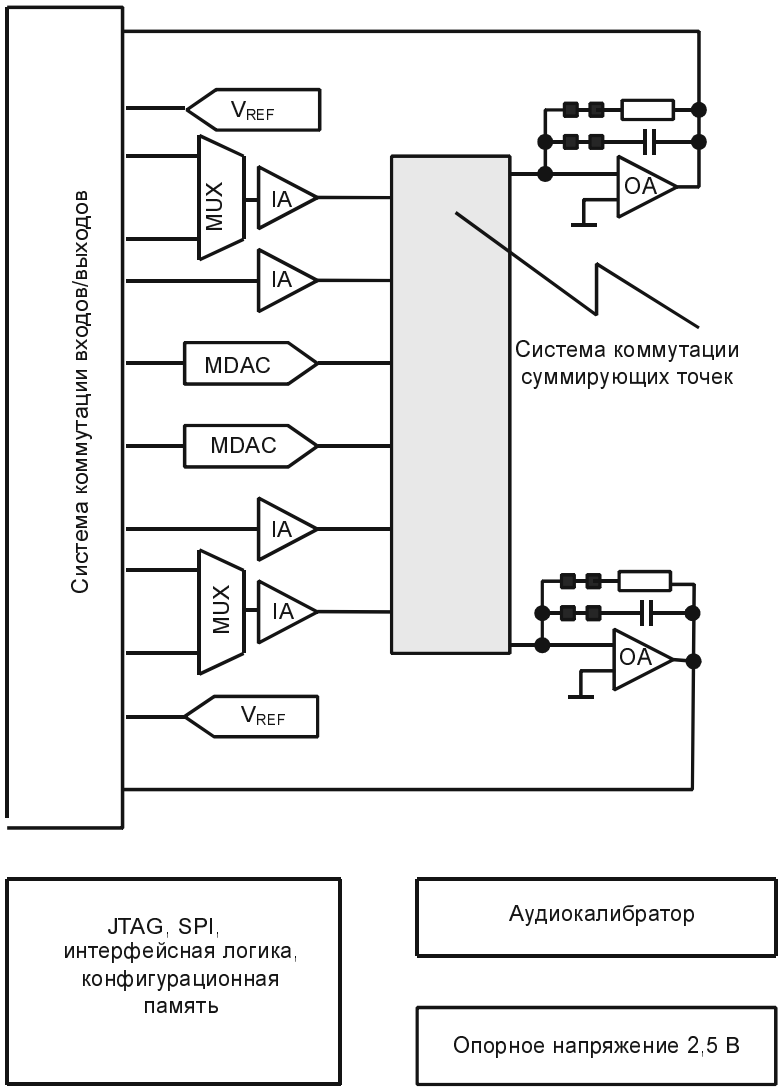


Рис. 9.35. Структура микросхем ispPAC30

Любые из блоков IA или MDAC могут подключаться к суммирующим точкам усилителей OA, придавая микросхеме максимальную гибкость конфигурирования.

Источники образцового напряжения V_{REF} = 2,5 В позволяют выводить рабочую точку усилителей в середину питающего напряжения для восприятия и выработки знакопеременных сигналов. Напряжения V_{REF} применяются и для более общих це-

лей формирования необходимых значений напряжений. От каждого источника V_{REF} можно получить 7 уровней напряжения (0,064; 0,128; … 2,048; 2,500 В). Напряжения могут суммироваться с другими сигналами или вычитаться из них, масштабироваться с коэффициентами от 1 до 10 с помощью усилителей IA и ослабляться до одного из 128 уровней с помощью MDAC , имеющих 7 информационных разрядов и один знаковый.

Комбинации способов формирования точных уровней напряжения дают очень большое число вариантов. Если, например, подать один и тот же сигнал U на входной усилитель и MDAC, а выходы этих блоков подключить к суммирующей точке усилителя OA, то от IA получим целую часть результата (от 1U до 10U), а от MDAC — дробную. В итоге можно сформировать любой уровень напряжения в пределах от −11U до + 11U с разрешением, лучшим, чем 0,01U, что дает приблизительно 2500 вариантов.

Выходные функции реализуются усилителями OA с элементами R и C в цепях обратных связей. Блоки IA, MDAC, V_{REF} предназначены для включения во входные цепи OA. Во входных цепях OA для режимов нормирующих усилителей, интеграторов, фильтров должны быть включены сопротивления определенных номиналов. Поэтому возникает вопрос о трактовке цепей с блоками IA и MDAC как некоторых эквивалентов сопротивлений. При этом последовательное включение усилителя IA и резистора R (для микросхем ispPAC30 R = 50 кОм) при единичном коэффициенте усиления трактуется как резистор с сопротивлением R, а при коэффициенте усиления 10 как резистор с сопротивлением 0,1R, т. е. для получения значения эквивалентного сопротивления фактическое его значение нужно разделить на коэффициент усиления IA. При последовательном включении резистора и MDAC для получения эквивалентного значения сопротивления нужно разделить R на коэффициент передачи MDAC. Например, если MDAC передает на выход 50% входного напряжения, то эквивалентное сопротивление составит 2R.

Схемы коммутации сигналов ввода/вывода и коммутации суммирующих точек позволяют точно передавать аналоговые значения напряжений и токов, передавать входные сигналы микросхемы к любому IA или MDAC (внешние контакты микросхемы на рис. 9.32 не показаны) и подключать любой IA или MDAC к суммирующей точке любого выходного усилителя. Распределение внешних контактов относительно точек внутренней схемы является гибким. Чувствительные к помехам цепи выполнены по дифференциальной схеме, защищенной от воздействий других сигналов, что делает качество передачи сигнала не зависящим от положения линии связи на кристалле и улучшает другие характеристики схем. Из нескольких схем семейства ispPAC можно строить перестраиваемые активные фильтры различных порядков. Микросхемы ispPAC80/81 специально предназначены для создания активных фильтров пятого порядка с программируемыми характеристиками. Могут строиться фильтры с аппроксимацией частотных характеристик по Баттерворту, Бесселю, Чебышеву, фильтры с аппроксимацией эллиптического типа.

Аналоговые блоки характеризуются совокупностью их функциональных и точностных характеристик с учетом статических и динамических погрешностей.

Укажем основные параметры микросхемы ispPAC30:

- Диапазон входных напряжений IA от 0 до 2,8 В, напряжение смещения, приведенное к дифференциальному входу при коэффициенте усиления 10, 60(200)[1] мкВ, его дрейф 50 мкВ/ °С. Входное сопротивление 10^9 Ом, входная емкость 2 пФ, ток смещения 1 пА при 25 °С и 200 пА при 85 °С. Приведенная ко входу при коэффициенте усиления 10 плотность напряжения входного шума 70 нВ/$\sqrt{\text{Гц}}$ (на частоте 10 кГц).
- Перепад выходного напряжения ОА 5 В, максимальный выходной ток ±30 мА.
- Диапазон программируемых коэффициентов усиления от 0 до 20 дБ, их погрешность 1(3)%, дрейф 35 ppm/ °С. Отношение сигнал/шум в полосе от 0,1 Гц до 114 кГц 83 дБ, коэффициент гармонических искажений на частоте 10 кГц составляет –85 дБ, на частоте 100 кГц –75 дБ. Малосигнальная полоса пропускания 1,57 МГц, крутизна фронтов 15 В/мкс. Время установления положительного перепада с погрешностью 0,1% составляет 2…4 мкс.
- Разрешающая способность MDAC — 128 уровней, интегральная нелинейность 0,25(0,5) цены младшего разряда LSB, дифференциальная нелинейность 1 LSB, напряжение смещения 3 мВ, погрешность коэффициента передачи 1(3)%, полоса пропускания (3 дБ) 1,6 МГц. Знаковый разряд определяет полярность выходного напряжения.
- Опорное напряжение V_{REF} = 2,500 В ± 0,2%, его дрейф 100 ppm/ °С, напряжение шумов при полосе 100 кГц составляет 40 мкВ.
- Времена переключения компаратора — 4 мкс при зоне установления 10 мВ и 2,5 мкс при зоне установления 100 мВ.
- Параметры фильтра — диапазон угловых частот 49…619 кГц, погрешности угловых частот 3…5%, дрейф 0,05%/ °С.

Относительно скромные точностные возможности микросхем ispPAC, тем не менее, приемлемы для построения на них ряда работоспособных функциональных узлов.

На рис. 9.36 приведена схема фильтра первого порядка, реализованная на кристалле ispPAC30.

Самый распространенный вариант реализации фильтра первого порядка — схема операционного усилителя с параллельной RC-цепочкой в обратной связи. Выходное напряжение такого фильтра с использованием преобразования Лапласа записывается в следующем виде

$$-U_{OUT}(p) = U_{IN}(p) /(1 + pT),$$

где p — комплексная переменная, используемая в преобразовании Лапласа, T — масштабный коэффициент, имеющий размерность времени, а $U_{OUT}(p)$ и $U_{IN}(p)$ — изображения по Лапласу выходного и входного напряжений.

[1] Если параметр указан двумя цифрами, то первая цифра относится к его типовому значению, а вторая (в скобках) — к предельному.

Это выражение можно преобразовать в следующее

$$(U_{OUT}(p) + U_{IN}(p))/pT = -U_{OUT}(p),$$

где pT — оператор интегрирования.

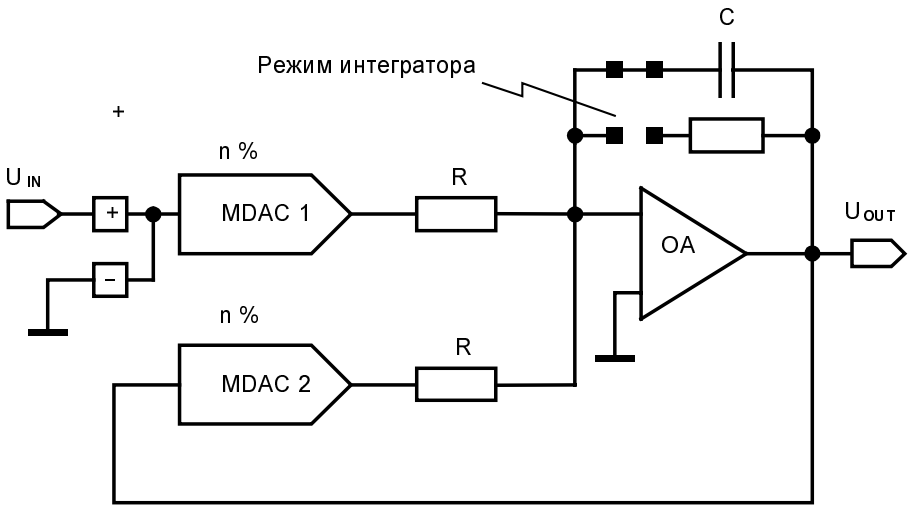


Рис. 9.36. Схема фильтра, реализованного на микросхеме ispPAC30

Именно по такому соотношению и реализован фильтр, показанный на рис. 9.33, в котором сумма входного и выходного напряжений интегрируется выходным усилителем (необходимые знаки напряжений и масштабные коэффициенты обеспечиваются программированием блоков MDAC).

Системы на кристалле, содержащие аналоговые программируемые блоки, рассмотрены в следующей главе.

§ 9.6. Способы оценки параметров ПЛИС

Разнообразие рынка *программируемых логических интегральных схем* (ПЛИС) требует от разработчиков аппаратуры глубокого знания особенностей и сравнительных характеристик существующих микросхем. Полное описание в фирменных справочниках более или менее сложной ПЛИС занимает десятки—сотни страниц. Для предварительной оценки ПЛИС нужно учесть хотя бы несколько основных параметров, в первую очередь логическую сложность, быстродействие, стоимость, тип корпуса, напряжение питания, потребляемую мощность. Оценки логической сложности и быстродействия ПЛИС неоднозначны и требуют пояснений.

Оценки логической сложности ПЛИС

О понятии "эквивалентный вентиль". Сложность базовых матричных кристаллов (БМК, предшественников ПЛИС) оценивалась числом *"эквивалентных венти-*

лей", причем под *эквивалентным вентилем* понимали обычно логический элемент 2И-НЕ. Подсчет числа таких вентилей, реализованных на кристалле, давал оценку его логической сложности. Однако для ПЛИС, более разнообразных по архитектуре и схемотехнике, стало невозможным просто подсчитывать число вентилей на кристалле, поскольку ПЛИС отнюдь не состояли из таких вентилей.

Для оценки сложности ПЛИС нельзя просто подсчитать число эквивалентных вентилей в ее схеме, но можно воспользоваться сопоставлением ее функциональных возможностей с возможностями *взятого в качестве образца БМК* с помощью *набора эталонных схем.* В качестве эталонных схем были выбраны типовые функциональные узлы (регистры, счетчики, дешифраторы и т. д.). Для кристалла ПЛИС определялось максимально возможное число реализаций в нем каждой схемы из этого набора. Например, определялось, что в данной ПЛИС можно реализовать N эталонных счетчиков. Из практики работы с БМК для каждой эталонной схемы была известна ее сложность, выраженная в числе эквивалентных вентилей. Пусть для эталонного счетчика это M вентилей. Сложность оцениваемого кристалла ПЛИС по данной эталонной схеме определялась как произведение сложности эталона на максимально возможное число его реализаций на кристалле, для нашего примера это MN. Усредненная по всем эталонным схемам величина характеризовала итоговый (окончательный) показатель сложности ПЛИС, выраженный числом эквивалентных вентилей. В течение некоторого времени такая оценка широко применялась производителями ПЛИС, встречается она и сейчас, хотя параметр "число эквивалентных вентилей" становился со временем все более размытым.

О понятии "системный вентиль". Ситуация с применением понятия "эквивалентный вентиль" осложнилась тем, что в ПЛИС в качестве функциональных блоков стали применять табличные преобразователи (LUT-блоки), допускающие *неоднозначное использование* — для воспроизведения логических функций или в качестве блоков памяти. Кроме того, на кристалле появились специализированные *области встроенной памяти* и другие *структурные неоднородности*. В ряде микросхем стали использоваться и аналоговые блоки, которые трудно "пересчитать" в эквивалентные вентили.

Специфическая ситуация создалась из-за неоднозначности использования LUT-блоков. Реализации разных режимов работы одного и того же LUT-блока средствами БМК (базовых матричных кристаллов) сильно различаются по сложности. Так, например, четырехвходовой LUT-блок имеет 16 бит памяти. При реализации памяти на БМК действует оценка "четыре вентиля на один бит", что для четырехвходового LUT-блока дает эквивалентную сложность, равную 64 вентилям. В то же время для режима воспроизведения функций эквивалентная сложность этого же блока оценивалась как 12 вентилей.

В ответ на эту ситуацию возникли понятия *логических и системных* вентилей. При подсчете числа логических вентилей предполагается работа LUT-блоков в режиме воспроизведения функций. При подсчете числа системных вентилей предполагается, что часть LUT-блоков работает в режиме блоков памяти и, следовательно, имеет другую, более высокую, эквивалентную сложность. Чтобы число системных

вентилей, характеризующих сложность кристалла, было правильно понятым, должно быть указано, какая часть LUT-блоков микросхемы используется в качестве функциональных блоков и какая часть в качестве блоков памяти.

О понятии "типичный диапазон числа вентилей". Для оценки *типичной емкости* привлекаются усредненные по множеству проанализированных проектов статистические данные о доле LUT-блоков, работающих в режиме воспроизведения функций. Для *типичного диапазона* числа вентилей нижняя граница часто соответствует числу логических вентилей (или определенной доле этого числа, например 2/3), а верхняя определяется при указании доли блоков, реализующих функции памяти (например, 20—30%).

Другие оценки логической сложности ПЛИС. Дифференциация понятия "число вентилей" не сняла перечисленные трудности оценки логической сложности ПЛИС одним интегральным параметром — числом тех или иных вентилей. В итоге в современных условиях параметр "число эквивалентных вентилей" приобретает *сугубо ориентировочный* характер. Наиболее достоверны сравнения микросхем с помощью этого параметра внутри продукции одной и той же фирмы.

В последнее время в качестве характеристик логических возможностей ПЛИС предпочитают указывать для CPLD число макроячеек и их параметры, а для FPGA число логических блоков (ячеек), данные о ресурсах памяти и др. Сведения об этих параметрах более репрезентативны, так как говорят о числе функций, которые могут быть воспроизведены данной микросхемой.

Во избежание неопределенности и для новых характеристик стали вводить понятия, близкие по смыслу к "эквивалентным вентилям". Так, например, для FPGA во многих разработках утвердился типовой состав логической ячейки, основа которого — четырехвходовой LUT-блок и триггер (регистр). Это вызвало к жизни термин *"стандартная логическая ячейка"*. Применение этого понятия как единицы измерения сложности ПЛИС удобно, но и здесь вскоре возникли трудности, поскольку в микросхемах стали применять LUT-блоки с 6 входами, "функциональность" которых увеличилась. Предложено (фирма Xilinx) по определенной методике подсчитывать "эквивалентные ресурсы" аппаратных средств (при этом ячейки с шестивходовыми LUT-блоками оцениваются как приблизительно 1,5 стандартные ячейки). Однако и здесь не все однозначно. В схемах используются различные функциональные расширители, цепи быстрых переносов и каскадирования и другие фрагменты, которые не отображаются как самостоятельная логика, но могут существенно влиять на функциональные возможности и производительность кристалла, причем степень такого влияния опять-таки зависит от характера конкретного проекта.

Оценки быстродействия

Быстродействие ПЛИС характеризуется либо задержками распространения сигналов по указанным путям, либо максимальными частотами их работы в указанных условиях.

Пути распространения сигналов в CPLD жестко предопределены их архитектурой, и *задержки по путям* от любого входа до любого выхода (Pin-to-pin, Corner-to-corner, Clock-to-pin) могут достаточно хорошо характеризовать быстродействие CPLD.

Пути распространения сигналов в FPGA разнообразны, и для них с целью оценки быстродействия целесообразно пользоваться максимально возможной частотой тактирования, причем эта частота может быть указана различным способом. Например, указывают максимально возможную *частоту работы счетчика* F_{CNT} или частоту работы (тактирования) схемы в целом, называемую *системной*. Обычно системная частота приблизительно вдвое ниже, чем частота работы счетчика.

Нередко в сведениях о ПЛИС отдельно приводятся тактирующие *частоты для синхронизации внутреннего ядра и цепей ввода/вывода*. Встречаются различные соотношения между этими частотами.

Оценка быстродействия ПЛИС дается также с применением понятия *градация быстродействия*. Этот параметр входит в состав кода обозначения схемы в виде цифры, перед которой ставится дефис (–3, –4, –5 и т. д.). Для архитектур типа CPLD с характерной для них независимостью задержек распространения сигналов от их конкретных путей цифра, приводимая на позиции "градация быстродействия", приближенно совпадает с округленным значением задержки распространения сигнала через данную ПЛИС. Для FPGA цифра, определяющая градацию быстродействия, имеет лишь относительное значение. Она ранжирует микросхемы по быстродействию, но не указывает абсолютные значения динамических параметров, являясь, таким образом, как бы указателем для их поиска в таблице. Указание градаций быстродействия целесообразно для сравнительной оценки ПЛИС в пределах одного и того же семейства. Для сравнения разнотипных ПЛИС лучше оценивать тактовые частоты работы типовых функциональных узлов, реализованных в данной микросхеме.

Факторы, влияющие на стоимость

На показатель стоимости, как и на другие параметры микросхемы, сильное влияние оказывает технологический уровень производства. Увеличение диаметра пластин (сейчас это около 300 мм) дает более экономичное использование кремния при разрезании пластины на кристаллы. Уменьшение *топологических норм* сокращает площадь кристалла, необходимую для размещения той же схемы, и тем самым уменьшает удельную стоимость ее ресурсов. Поскольку уменьшение топологических норм улучшает как экономические, так и важнейшие технические параметры микросхем, понятна упорная борьба их производителей за минимальность этих норм, успехи в которой обуславливают конкурентоспособность фирм. Например, переход от *технологии* с минимальными размерами 0,35/0,5 мкм (1997 г.) к технологии 0,09 мкм (2003 г.) для микросхем одного и того же семейства микросхем фирмы Xilinx улучшил показатель удельной стоимости "вентилей/доллар" в 8,5 раз.

Стоимость микросхем во многом определяется *типом их корпусов* — в зависимости от этого фактора цена одного и того же кристалла может измениться в несколько раз. Значительно (многократно) повышаются и цены на микросхемы, имеющие *специальную приемку* (военную и др.). Стоимость зависит от *размера приобретаемой партии* (лота) и ряда других факторов. Одним из экономических преимуществ ИСПС (интегральной схемы с перестраиваемой структурой) перед заказными микросхемами является *отсутствие минимального порога* при их заказе: допускается заказ любого количества микросхем.

§ 9.7. Конфигурирование программируемых микросхем

Способ *конфигурирования ПЛИС*, т. е. настройки микросхем на определенное функционирование, зависит от типа программируемых элементов (ПЭ). Микросхемы с необратимым изменением состояний перемычек (antifuse) и с энергонезависимой памятью конфигурации (EPROM, EEPROM, Flash) программируются электрическими сигналами, отличающимися от рабочих. Конфигурирование таких микросхем производится вне системы с помощью программаторов или же (для вариантов с EEPROM и Flash) в ее составе (т. е. при сохранении монтажа микросхемы на плате).

Загрузка триггерной памяти конфигурации не требует специальных режимов, и конфигурирование состоит в обычной передаче в микросхему информации по заданному протоколу и с фиксированными форматами данных. Эта информация обеспечивает создание требуемых связей в логических блоках, блоках ввода/вывода и системе межсоединений. Конфигурирование выполняется после каждого включения питания, причем, если установлены специальные загрузочные БИС, сам факт очередного включения питания автоматически инициирует процесс конфигурирования, который может повторяться неограниченное число раз. Память конфигурации можно загружать в работающей схеме, причем возможна и частичная реконфигурация схемы. При конфигурировании каждый бит настроечных данных задает состояние соответствующему триггеру, управляющему программируемым ключом в настраиваемой схеме.

Режимы конфигурирования

ИСПС обычно имеют *несколько возможных режимов конфигурирования*. Рассмотрим типичный пример.

Для конфигурирования микросхем используются как специализированные выводы, так и выводы, которые после завершения конфигурирования могут играть роль выводов общего назначения. К специализированным относятся выводы для задания кода того или иного режима, выводы для синхросигналов, выводы PROGRAM, DONE и выводы граничного сканирования. Вывод синхросигнала может быть вы-

ходом, когда этот сигнал генерируется микросхемой, или входом, когда сигнал поступает извне.

Возможные способы конфигурирования:

- *пассивный последовательный* (Slave-serial mode);
- *активный последовательный* (Master-serial mode);
- *байт-последовательный* (SelectMAP mode);
- *граничного сканирования*.

В *пассивном последовательном режиме* микросхема получает данные конфигурирования в виде потока битов из последовательной памяти при синхронизации от внешнего источника. Каждый фронт синхросигнала вводит бит данных, поступающий на вход DIN (Data Input). Несколько микросхем могут быть соединены в цепочку для конфигурирования в едином процессе от общего потока битов. В этом случае после завершения конфигурирования очередной микросхемы данные конфигурации для следующих микросхем появляются на выводе DOUT (Data Output) микросхемы, завершившей конфигурирование.

В *активном последовательном режиме* синхросигнал из микросхемы подается на последовательное ЗУ, с которого на вход DIN поступает последовательный поток битов конфигурации. Микросхема воспринимает каждый бит по фронту синхросигнала. После загрузки очередной микросхемы, входящей в цепочку, данные для следующей снимаются с выхода DOUT той микросхемы, которая закончила конфигурирование. Для синхронизации процесса можно выбирать частоту из некоторого диапазона значений. Устанавливаемые частоты, естественно, должны соответствовать возможностям используемой памяти и включенных в цепочку микросхем.

В *байт-последовательном* режиме время конфигурирования минимально. Байт-последовательный поток данных записывается в микросхему с учетом флажка ее готовности BUSY. Поток задается от внешнего источника, как и сигналы тактирования, разрешения работы (CS) и записи (WRITE). В этом режиме данные при пассивном сигнале WRITE могут читаться. Можно также конфигурировать несколько микросхем, но в этом случае они включаются параллельно по входам синхронизации, данных, WRITE и BUSY и загружаются поочередно путем соответствующего управления сигналами разрешения их работы CS.

В *режиме граничного сканирования* конфигурирование осуществляется исключительно через выводы порта тестирования TAP (Test Access Port) интерфейса JTAG. Используется специальная команда CFG_IN, позволяющая входным данным преобразовываться в пакеты данных для внутренней шины конфигурации микросхемы.

Этапы конфигурирования

Процесс конфигурирования содержит три этапа:

- очистка памяти конфигурации;
- загрузка в нее данных;
- активизация логических схем, участвующих в процессе.

Конфигурирование начинается автоматически после включения питания, но может быть и задержано пользователем с помощью сигнала PROGRAM, снятие которого запрещает конфигурирование. Завершение очистки памяти выявляется с помощью сигнала INIT, а завершение всего процесса — с помощью сигнала DONE.

Данные для загрузки памяти конфигурации формируются системой автоматизированного проектирования (САПР).

Реконфигурация в системе. Эта возможность (ISP, In-System Programmability) — одно из важнейших достоинств микросхем, позволяющее легко изменять логику их работы. Потребности в изменениях возникают для устранения не выявленных при первоначальном тестировании ошибок, для модернизации систем и в системах с многофункциональным использованием блоков. Наличие ISP облегчает работу с современными микросхемами, корпуса которых имеют большое число миниатюрных и легко повреждаемых выводов, что делает однократность установки микросхем на плату весьма желательной. Возможности ISP растут, если при проектировании часть функциональных ресурсов микросхемы оставлять свободной, имея запас по скорости, функциональным возможностям и ресурсам межсоединений. При реконфигурации в системе должно сохраняться назначение внешних выводов, иначе потребуется изменить монтаж печатных плат.

Конфигурирование без внешних источников данных. Среди ИСПС имеются и такие, в которых *реализованы одновременно триггерная и энергонезависимая память конфигурации.* В этом случае конфигурирование микросхемы можно производить без внешних источников данных путем автоматической загрузки триггерной памяти из энергонезависимой.

§ 9.8. Засекреченность проектов

Проблема защиты интеллектуальной собственности для программируемых микросхем приобретает *особую остроту*. Имея дело с программируемыми схемами, во многих случаях легко воспользоваться плодами чужого труда, т. к. для этого нужно лишь получить сведения о содержимом памяти конфигурации и затем загрузить их в стандартную микросхему.

Дублирование чужих проектов без раскрытия их внутреннего устройства называют *клонированием* проектов (Cloning). Более сложная задача расшифровки чужих проектов с раскрытием их архитектуры и деталей реализации (Reverse-engineering) — *реконструкция* проектов. Имея реконструированный проект, недобросовестный конкурент может внести в него какие-либо несущественные изменения и попытаться обойти вопросы лицензирования.

Уязвимость микросхем программируемой логики по отношению к клонированию или реконструкции проектов зависит от характера проекта и схемотехнологии микросхемы.

Логический анализ. Проекты с простыми комбинационными ПЛМ и ПМЛ можно реконструировать (по крайней мере принципиально) путем логического анализа, подавая на схему все возможные комбинации входных сигналов и фиксируя соответствующие им выходные комбинации сигналов. Из полученных сведений можно вывести булевы функции, воспроизводимые схемой, и далее реконструировать ее в соответствующем логическом базисе.

Более сложные схемы практически не поддаются логическому анализу. Они имеют большое число вводов/выводов, назначение которых заранее не известно (например, у двунаправленных выводов). Уже одно это создает *очень большие сложности для логического анализа проектов*, т. к. неизвестно, подавать ли на вывод входной сигнал, снимать ли с него выходной сигнал или же использовать его поочередно в обоих вариантах. Сложность внутренней структуры микросхем, наличие в них последовательностных фрагментов и встроенных функций чрезвычайно затрудняют логический анализ проектов.

Клонирование и реконструкция проектов

Для клонирования проектов нужно раздобыть сведения о содержимом памяти конфигурации микросхемы (битовом потоке конфигурирования). По возможностям защиты этой информации от несанкционированного доступа *микросхемы разных схемотехнологий существенно* различаются.

Схемы с пробиваемыми перемычками. *Однократно программируемые* схемы с пробиваемыми перемычками наиболее защищены от взлома. Для их эксплуатации битовый поток конфигурирования не нужен, поскольку программирование перемычек завершается на стадии изготовления микросхемы. В распоряжении взломщика находится *лишь сам кристалл*. Для раскрытия проекта требуется определить состояние всех перемычек, получив для каждой ответ на вопрос "замкнута-разомкнута". Это практически невозможно. Число перемычек очень велико, а наблюдением поверхности кристалла нельзя выявить не только состояние перемычек, но и их местоположение. Чтобы определить состояние перемычек, нужно сделать в каждой из них несколько поперечных срезов. Для выявления состояний всех перемычек потребовались бы неприемлемые усилия и затраты.

Схемы с программированием плавающих затворов. Репрограммируемые ИСПС с энергонезависимой памятью конфигурации (EPROM, EEPROM, Flash) в рабочих режимах также не используют файлы конфигурирования. Взломщик, как и в предыдущем случае, имеет в своем распоряжении сам кристалл, в котором скрыта информация о проекте. Чтение памяти конфигурации может быть запрещено битом секретности, сбросить который можно только при стирании всего содержимого этой памяти (имеются сведения о случаях проникновения в запертую память конфигурации с помощью специальных электрических режимов). Исследовать сам кристалл проще, чем кристалл с перемычками, но также очень нелегко. Если предположить, что расположение транзисторов с плавающими затворами, состояния которых программируются, известно или может быть визуально определено, то

задача раскрытия проекта сводится к исследованию состояний каждого из многих тысяч или даже миллионов транзисторов. Наличие или отсутствие заряда в плавающем затворе можно выявить без разрушения кристалла. Заряды создают электрические поля, которые можно обнаруживать специальными методами, можно использовать также электронный микроскоп или материалы, накладываемые на кристалл и изменяющие цвет под воздействием электрических полей. Трудоемкость и стоимость исследования состояний плавающих затворов остаются все же чрезвычайно высокими. Далее, даже если станут известны местоположение и состояния всех транзисторов, то для простого клонирования проекта нужно транслировать эти сведения в битовый поток конфигурирования схемы, а для реконструкции проекта еще и в саму схему. Решение этих задач очень сложно, поэтому схемы с энергонезависимой памятью конфигурации можно считать *практически (хотя и не абсолютно) защищенными от взлома.*

Схемы с триггерной памятью конфигурации. Самыми уязвимыми для взломщиков являются схемы *с триггерной памятью конфигурации*, которую нужно загружать при каждом включении питания от внешнего источника хранимых данных. Для клонирования проекта достаточно прочитать содержимое этой внешней памяти и использовать его для конфигурирования клонов. Установление соответствия недокументированного битового потока конфигурирования и внутренней структуры схемы (реконструкция проекта) является более сложным, но не считается невозможным. С целью повышения защищенности проектов с триггерной памятью конфигурации, принимается ряд мер: организационных, юридических, конструктивных и др.

К организационным мерам можно отнести поставку немаркированных кристаллов, что существенно затрудняет попытки взлома проектов; к юридическим — встраивание в проект некоторого недокументированного идентификатора; к конструктивным — покрытие кристалла и его связей с другими кристаллами непроницаемым слоем, например, эпоксидной смолой (это имеет и отрицательные последствия, не позволяя в дальнейшем дорабатывать проект и ухудшая тепловой режим кристалла). Возможно и размещение трассы передачи битового потока конфигурирования между энергонезависимой памятью и микросхемой в скрытых внутренних слоях печатной платы.

Можно вообще обойтись без энергонезависимой памяти конфигурирования, если разместить на плате автономный источник питания (литиевую батарею), сохраняющий запрограммированную конфигурацию в триггерной памяти. Однако в этой ситуации потребуются специальные схемы для изоляции источника автономного питания от всех цепей, кроме памяти конфигурации, и, кроме того, при истощении батареи схема все же разрушится. Разрушение может произойти и от мгновенной потери питания вследствие удара, действия помехи и т. д.

Большое распространение получило *шифрование данных конфигурации*. В этом случае на схему передается поток зашифрованных данных, который преобразуется в данные конфигурирования устройством дешифрации, находящимся на кристалле. Для этих целей используют генераторы и анализаторы *CRC* (Cyclic Redundance Code) и

шифрование ведется по стандартам, обеспечивающим высокую засекреченность данных.

Существуют и другие возможности затруднить процесс взлома проектов, реализованных в БИС/СБИС программируемой логики.

§ 9.9. Примеры типичных FPGA средней сложности

Рассмотрим примеры недорогих FPGA, которые в силу своей типичности дают достаточное представление об особенностях и возможностях класса FPGA средней сложности в целом.

FPGA с триггерной памятью конфигурации

Очевидные претенденты на роль типичного представителя недорогих FPGA средней сложности с триггерной памятью конфигурации — микросхемы популярных серий Cyclone фирмы Altera и Spartan фирмы Xilinx. Трудно отдать предпочтение рассмотрению одной из этих серий, но ради краткости приходится это делать. Так, в известной мере произвольно, выбор сделан в пользу микросхем Cyclone (о семействе Spartan приводятся только краткие сведения).

Семейство Cyclone-4E. Основные параметры микросхем семейства приведены на рис. 9.37.

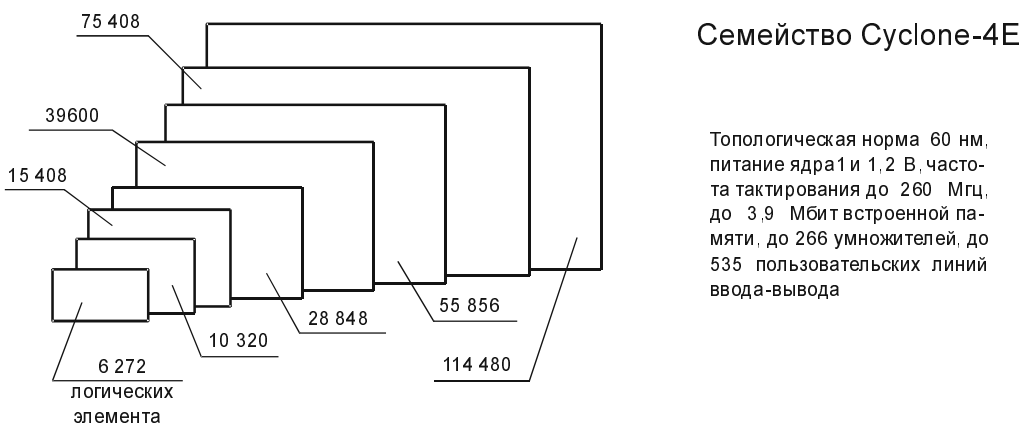


Рис. 9.37. Основные параметры семейства Cyclone-4E

Микросхемы имеют столбцово-строчную структуру как для логических блоков, так и для межблочных связей. Встроенные блоки памяти и умножители расположены в специальных столбцах, размещенных между столбцами логических блоков. Блоки PLL занимают углы кристалла (рис. 9.38).

Семейство MAX-2. Это ориентированное на несложные проекты популярное семейство микросхем невысокой стоимости общего назначения (фирма Altera). Сочетает признаки FPGA и CPLD. Частота работы внутренних блоков до 300 МГц, экономично по потребляемой мощности. Флэш-память на кристалле позволяет не терять конфигурацию схемы при снятии питания и обеспечивает схеме немедленную готовность к работе при ее включении.

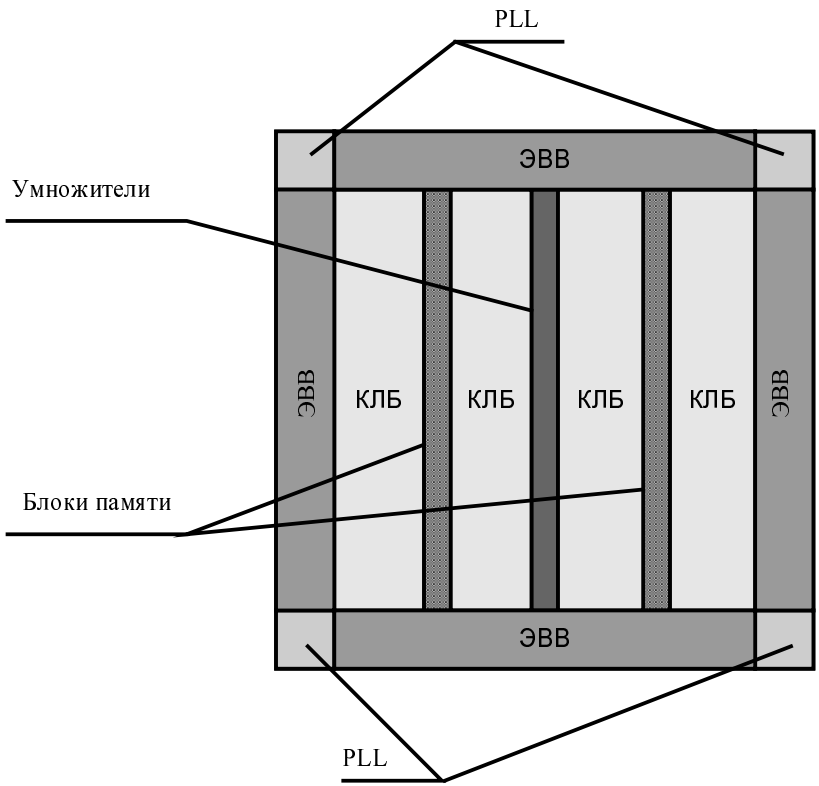


Рис. 9.38. Структура кристалла микросхем Cyclone

Семейство Spartan-6. К популярным и дешевым FPGA средней сложности относится и серия Spartan фирмы Xilinx. В настоящее время старшим представителем серии является семейство Spartan-6, основные параметры которого приведены на рис. 9.39. Разновидности Spartan-6X и Spartan-6LXT, ориентированы на оптимизацию по стоимости (LX) и высокую производительность последовательных передач (LXT). Напряжения питания 1,0 и 1,2 В, системная частота 320 МГц. Распределенная память находится в пределах от 75 до 1 533 Кбит, а встроенная — от 216 до 4 824 Кбит. Число блоков ЦОС составляет от 8 до 180, в каждом из них имеется умножитель 18 × 18, сумматор и аккумулятор. Число выводов пользователя 132…576.

Размещение ресурсов на кристалле типично и напоминает, в частности, размещение ресурсов в микросхемах семейства Cyclone.

В табл. 9.1 приведены основные характеристики рассмотренных FPGA.

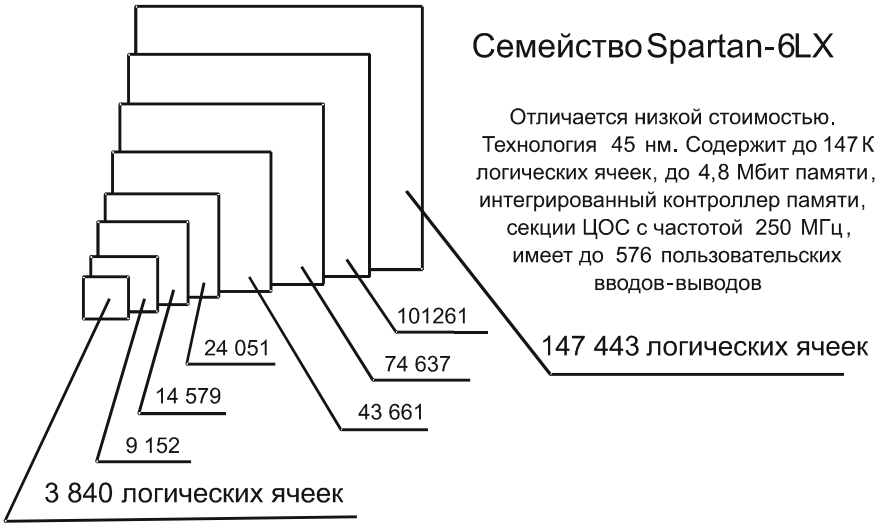


Рис. 9.39. Основные параметры семейства Spartan-6

Таблица 9.1

Семейство FPGA	Число блоков в/в	Число логических ячеек	Блоков встроенной памяти	Размер блока памяти, бит	Блоков ЦОС	Ориентировочная стоимость, USD
Cyclone-4	72…535	6К…150К	15…360	18К	0…360 MPL	15…790 (Cyclone-3)
MAX-2	80…272	240…2,2К	8 Кбит флэш-памяти		—	7…70
Spartan-6	132…576	3,8К…14К	12…268	18К	8…180	8…250

FPGA с программируемыми перемычками

Перемычки antifuse (фирмы Actel и QuickLogic) создают программируемые связи высокого качества (см. рис. 9.1 и текст к нему), но их преимущества покупаются ценою *однократности программирования*.

Логические модули и средства коммутации в FPGA с перемычками отличаются от традиционных для FPGA с триггерной памятью конфигурации.

Логические модули. Логические модули FPGA с перемычками реализованы на основе мультиплексоров и логических вентилей и составлены из ячеек двух типов — комбинационных C-ячеек (рис. 9.40, *а*) и регистровых R-ячеек (рис. 9.40, *б*). В ячейках первого типа нет элементов памяти, ячейки второго типа имеют триггеры с программируемыми функциями.

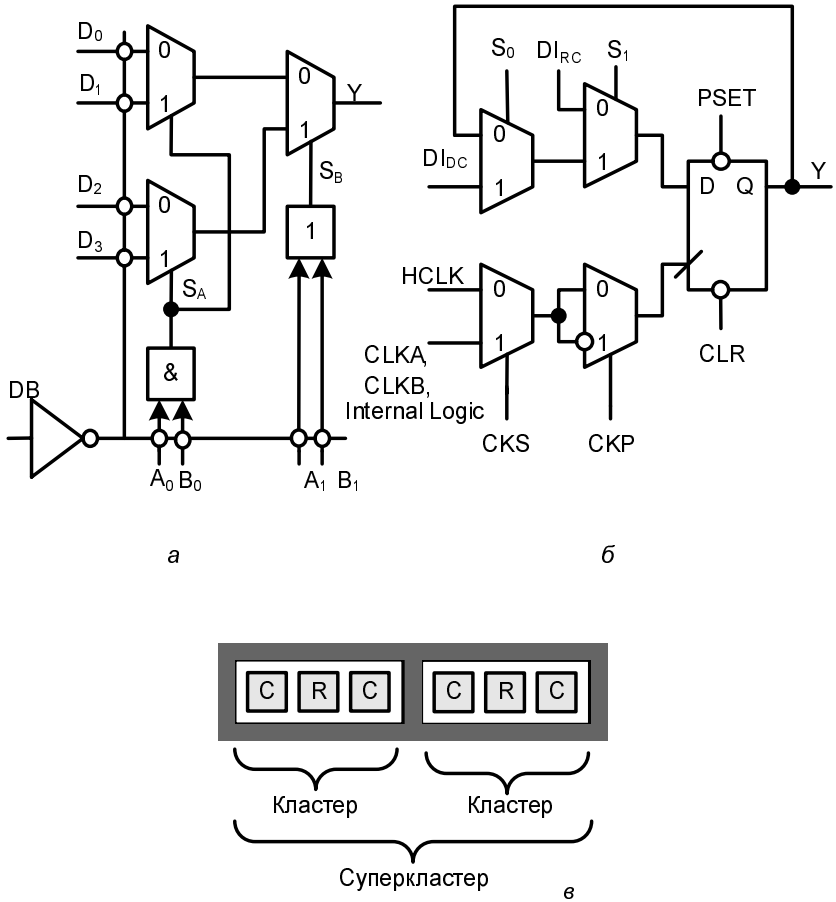


Рис. 9.40. Ячейки C (*а*), R (*б*) и суперкластер (*в*) логических модулей, программируемых перемычками antifuse

C-ячейка воспроизводит на выходе Y некоторую порождающую функцию, отличающуюся от рассмотренных ранее для мультиплексных блоков тем, что введение дополнительного входа DB, допускающего программированное инвертирование переменных (показано на рисунке условно), обогащает ее и позволяет получить более 4000 различных функций в одном логическом модуле. Две C-ячейки могут быть скомбинированы для получения схемы триггера (CC-макро).

R-ячейка содержит триггер с асинхронными сбросом и загрузкой, который в зависимости от сигналов S_0 и S_1 может получать входные сигналы DI от прямых связей (Direct Connect), от межсоединений типа Routing Connect или сигналы обратной связи. Источник сигналов тактирования выбирается программированием входа CKS, а их полярность — программированием входа CKP.

Чередование ячеек C-R-C образует кластер, а два кластера объединяются в суперкластер (рис. 9.40, *в*).

Система коммутации. В системе коммутации используются связи нескольких типов (рис. 9.41).

Прямые связи от C-ячеек к R-ячейкам внутри кластеров не включают в себя перемычек и обладают максимальным быстродействием (задержкой менее 0,1 нс). Быстрые связи (Fast Connects) соединяют любые два модуля в пределах суперкластера и создают вертикальные соединения с соседним снизу суперкластером. В них уже присутствуют перемычки и их задержки приблизительно втрое превышают задержки в прямых связях. Так называемые сегментные соединения (Segmented Routing) ориентированы на дальние передачи, составляются из отрезков разной длины и включают в себя до пяти перемычек. Специальные средства коммутации (High-Drive Routing) имеют повышенную нагрузочную способность.

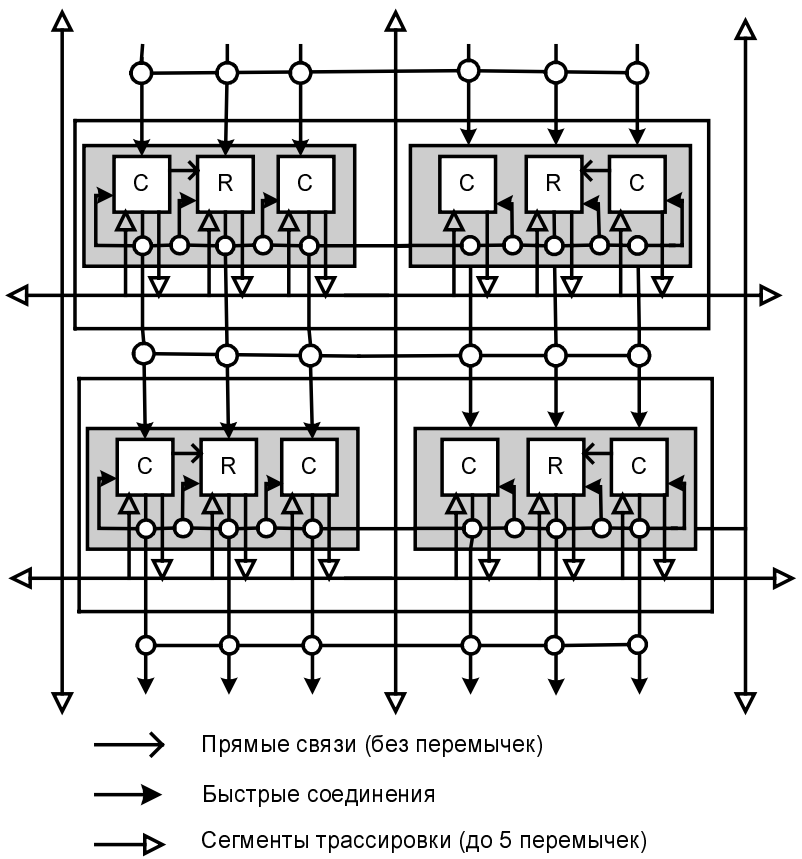


Рис. 9.41. Система коммутации FPGA с программируемыми перемычками

Элементы ввода/вывода. Каждый из элементов ввода/вывода может быть сконфигурирован как входной, выходной, выходной с третьим состоянием или двунаправленный. Элементы ввода/вывода не содержат регистров, при необходимости организуется их работа с регистрами, относящимися к логическим модулям. Все

выводы имеют резисторы Pull-Up и Pull-Down и при конфигурации выводы приводятся в определенное состояние. Затем эти резисторы отключаются, и элементы ввода/вывода управляются информационными сигналами.

Фирма Actel выпускает FPGA SX-A, eX и Axcelerator; микросхемы с повышенной радиационной стойкостью RTSX-S и RTAX-S (RT от Radiation Tolerant), а также системы на кристалле серии ProASIC с флэш-памятью конфигурации. В сериях для военной и аэрокосмической аппаратуры RTSX-S и RTAX-S используются тройное резервирование элементов памяти с мажоритарным голосованием, а также коды Хемминга для хранения слов.

Диапазон параметров некоторых микросхем фирмы Actel приведен в табл. 9.2.

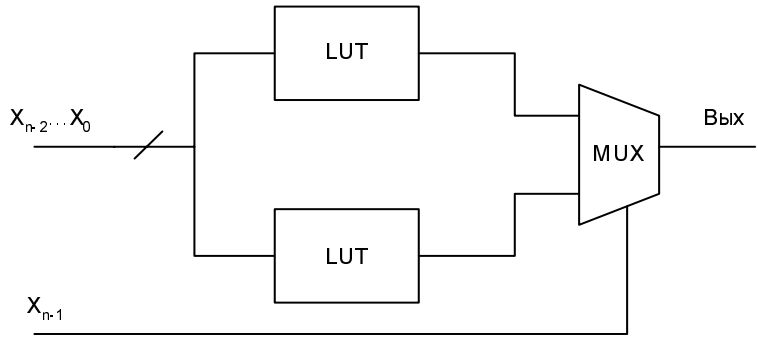
Таблица 9.2

Семейство FPGA	Число блоков в/в	Число логических ячеек	Встроенная память, бит	Размер блока памяти, бит	Ориентировочная стоимость, USD
eX	84…132	64…256 (R) 128…512 (C)	—	—	2…4
Axcelerator	168...684	672…10 К (R) 1344…21К (C)	18К…295К	4608	30…750
RTAX-S	248…840	1408…20К (R) 2816…43К (C)	54К…540К	4608	4000…10000

Контрольные вопросы

1. Дайте определение понятию FPGA.
2. CPLD и FPGA — две разновидности современных цифровых микросхем с программируемой структурой. Чем они отличаются по типу логических преобразователей и систем коммутации?
3. Как понимается термин "зернистость" применительно к функциональным блокам FPGA? Какие преимущества и недостатки связаны с применением мелкозернистых и крупнозернистых блоков в схемах FPGA?
4. Какие иерархические уровни типичны для системы межсоединений FPGA?
5. Почему в схемах FPGA широко распространены программируемые элементы в виде ключевого транзистора, управляемого триггером, несмотря на то, что такие элементы достаточно сложны?
6. Каким образом наличие комбинационного и регистрового выходов у логических блоков ПЛИС может увеличивать число воспроизводимых блоками функций?

7. Сколько разных функций может воспроизвести показанная на рисунке схема? Запишите аналитическое выражение для выходной функции.



8. К числу сигналов управления буферами блоков ввода/вывода ПЛИС относятся GOE, PTOE, SRS. Каков смысл этих сигналов?
9. Каково назначение цепей переноса и каскадирования в логических блоках FPGA?
10. Какая разновидность трассировочных ресурсов FPGA называется VersaRing? Какие дополнительные достоинства приобретает микросхема, имеющая такие ресурсы?
11. Почему в FPGA линии связи с большим числом последовательно включенных сегментов являются нежелательными? Как можно смягчить недостатки, свойственные таким линиям?
12. Что понимается под такими оценками сложности ПЛИС, как "число эквивалентных вентилей", "число системных вентилей", "число стандартных элементов"?
13. С какой целью разработаны и выпускаются FPGA, конфигурирование которых обслуживается памятью двух типов — энергонезависимой и триггерной?
14. Режимы конфигурирования делятся на активные и пассивные. Что понимается под этими определениями?
15. Какие ПЛИС обеспечивают защищенность реализованных в них проектов от клонирования и реконструкции? Какие ПЛИС наиболее уязвимы по отношению к этим действиям? Какой способ засекречивания проектов наиболее популярен для ПЛИС с триггерной памятью конфигурации?

Литература к главе: [17], [21], [23], [45], [III], [XVI], [XXXIV].

Глава 10

Программируемые системы на кристалле

§ 10.1. Основные сведения

Термин *"система на кристалле"* появился применительно к мегавентильным микросхемам, поскольку на них можно разместить целую систему (процессор, память, интерфейсные схемы и др.). Появление таких микросхем стирает грань между понятиями компонентной базы и аппаратуры. *"Программируемые системы на кристалле"* выпущены около десяти лет назад. Их современные варианты содержат уже десятки-сотни миллионов эквивалентных вентилей (миллиарды транзисторов), а их тактовые частоты составляют сотни мегагерц/ или даже превышают гигагерц.

Системы на одном кристалле имеют улучшенные технические и экономические качественные показатели. Интеграция функциональных блоков в одном кристалле повышает быстродействие и надежность систем, уменьшает потребляемую ими мощность и стоимость. Для некоторых сложных систем высшей производительности реализация на нескольких корпусах становится вообще невозможной.

> ***Примечание***
>
> Для обозначения программируемых систем на кристалле в англоязычной терминологии наряду с аббревиатурой *SoPC* (System on Programmable Chip) используется и ряд других, в частности, *PSoC* (Programmable System on Chip), *CSoC* (Configurable System on Chip), *FPSLIC* (Field-Programmable System-Level Integrated Circuit) и т. д. При описании конкретных микросхем далее сохранены наименования, применяемые фирмами-изготовителями. Для русского обозначения, как и ранее, принята аббревиатура СнПК — система на программируемом кристалле.

Большой значимости систем на кристалле отнюдь не препятствует то, что с точки зрения архитектуры (структуры) они не представляют собой чего-то принципиально нового. Если понятия CPLD или FPGA обозначают определенные архитектуры, то четкого понятия "система на кристалле" в этом аспекте не существует и такие системы архитектурно разнообразны. Термин "системы на кристалле" отражает специфику их проектирования, а также практические и рекламные факторы.

IP-ядра. Блочное и платформенное проектирование

Проектирование систем на кристалле требует больших затрат труда и времени. Подсчитано, что в рамках традиционного проектирования разработка сложной СБИС потребовала бы 500…1000 человеко-лет.

Снижение трудоемкости и стоимости проектирования — одна из важнейших задач микроэлектроники, одним из основных путей решения которой стало *создание библиотек* IP-блоков.

Библиотеки схемных решений получили широкое распространение уже в практике проектирования на основе базовых матричных кристаллов (БМК), предшественников ПЛИС. Производители БМК разрабатывали библиотеки проверенных и документированных схемных решений, чтобы облегчить использование своих кристаллов. Для СнПК проблема создания набора заранее отработанных схемных решений (IP-блоков) еще более важна. Созданием IP-блоков занимаются не только изготовители кристаллов, но и специализированные фирмы, разработки которых пригодны для кристаллов разных производителей. Диапазон сложностей синтезированных IP-блоков очень широк — от уровня функциональных узлов (счетчиков, регистров и т. п.) до уровня таких сложных устройств, как микропроцессоры, микроконтроллеры или ЦОС-блоки. В современных СнПК около 85% транзисторов, не входящих в схемы памяти, относятся к блокам IP.

IP-блоки для схем программируемой логики могут быть представлены в разной форме. Различают варианты:

- *soft-ядра* — это файлы, определяющие конфигурирование некоторой области кристалла таким образом, что в ней создается устройство заданного типа. Эти файлы можно интегрировать в описание проектируемого устройства на языках HDL (Hardware Description Language);
- *firm-ядра* — это вариант, близкий к предыдущему, его применяют для блоков, быстродействие которых особенно важно. В этих ядрах задание схемы является более жестким, чем в файле поведенческого описания, свойственного soft-ядрам, поскольку в описании firm-блока предопределены некоторые схемные межсоединения;
- *hard-ядра* — представляют собой реализованные на кристалле области с фиксированными функциями (устройства). Заказное проектирование hard-ядер позволяет оптимизировать соответствующие схемы, не содержащие средств программирования, но не позволяет как-либо репрограммировать эти ядра.

Между soft- и firm-ядрами с одной стороны и hard-ядрами с другой имеется существенная разница. Hard-ядра приобретаются вместе с микросхемой как ее части и не являются в этом смысле самостоятельным товаром. Soft- и firm-ядра приобретаются как *самостоятельные продукты* (причем обычно высокой стоимости).

Soft-блоки IP представляют собой не только описание конфигурируемого устройства на уровне регистровых передач, но и средства поддержки для пользователей (документацию с описанием работы блока и его интерфейса, модели тестирования

блока и т. п.). Однако, несмотря на тщательность отработки и высокую стоимость IP-блоков, всегда существует известный риск при интеграции их в разрабатываемый проект. Имеются сведения о том, что более половины новых проектов первоначально выполняются с ошибками функционирования, которые требуют доработок, иногда и серьезных.

> ***Примечание***
>
> Для обозначения ядер применяются термины IP, т. е. интеллектуальные собственности, и VC (Virtual Components), т. е. виртуальные компоненты. Между этими терминами существует некоторое различие. Под IP подразумевается разработка, которая может подойти и для повторного использования в других проектах, тогда как под VC подразумевается разработка, которая с самого начала проектирования рассчитана на многократное применение в проектах, причем условия применения в разных проектах могут и заметно отличаться друг от друга. Проектирование VC, способных надежно функционировать в разных проектах, является более трудоемким и дорогостоящим. Для дальнейшего изложения главы различие этих терминов несущественно и как правило применяется термин "IP-блок". В отечественных работах (НИИ "Прогресс") для обозначения IP или VC используется и русский термин СФ-блоки (сложные функциональные блоки).

В проектировании с применением IP-блоков сложились два стиля:

- *блочный*;
- *платформенный*.

В первом случае система проектируется как комбинация разнообразных IP-блоков, как вновь разработанных, так и привлеченных из библиотек поставщиков или других источников. В систему включаются только те блоки, которые требуются для данного проекта с учетом условий их работы. Это дает проектировщикам наибольшую гибкость в интеграции и оптимизации системы, но требует значительных затрат труда в процессе проектирования (разработка полной документации, сертификация блоков и т. д.).

Во втором случае *определенный набор блоков* организуется в многократно используемую платформенную архитектуру. Применение платформы повышает производительность проектирования и уменьшает его риски. Платформы могут быть разработаны под четко *определенные приложения* и содержать полный набор средств решения задач для конкретной прикладной области. Кроме того, платформы могут обладать существенными *возможностями реконфигурирования*, например, содержать hard-ядро процессора, массив программируемой логики, обеспеченный IP-библиотекой и средствами САПР и какие-либо фиксированные ядра. Платформа может быть *процессорно-ориентированной* (базироваться на определенном процессоре и шинной структуре и свойственных им наборе периферийных устройств).

В ходе эволюции стили проектирования проходят вначале этап блочного проектирования, а затем уже формируется платформенный подход. Блочный подход может оставаться оптимальным в ситуациях, когда требуется создать систему высшей производительности. Однако для очень многих приложений целесообразен платформенный подход с его большей экономичностью разработки системы.

Типы программируемых "систем на кристалле"

Все программируемые "системы на кристалле" подобны друг другу тем, что синтезируются с помощью ядер. Тип ядер (soft или hard) мог бы служить для СнПК классификационным признаком, однако этому мешает разнотипность ядер в одной и той же СнПК (такая разнотипность становится все более обычной). Из практических соображений целесообразно выделить два типа СнПК:

- *однородные*;
- *блочные*.

Системы однородного типа. Это системы на микросхемах высшей сложности и производительности, в которых применяются преимущественно *soft-ядра*, реализуемые схемотехнически однотипными ресурсами в тех или иных областях кристалла благодаря их программируемости. Такие системы ориентированы главным образом на телекоммуникации, сетевые приложения, обработку изображений и речи и другие мультимедийные задачи. Эти системы назовем *однородными*, имея в виду идентичность логических ресурсов, размещенных на кристалле и применяемых для построения требуемых устройств (в английской терминологии это системы типа generic).

> ***Примечание***
>
> Определение "однородные" следует относить лишь к схемотехническим ресурсам основной части микросхемы, в отдельных ее областях могут быть и "неоднородности" (встроенная память, блоки управления синхросигналами и т. д.).

Системы блочного типа. Это системы умеренной сложности и производительности, ориентированные на задачи управления техническими объектами и технологическими процессами. В таких системах наряду с синтезируемыми ядрами широко применяются *hard-ядра*, т. е. на кристалле выделяются области с фиксированными функциями, не содержащие средств программирования структуры. С учетом структуры эти системы можно назвать *блочными*. Кроме того, согласно главной области применения подобные системы можно назвать *микроконтроллерными*.

Исторически первыми были однородные СнПК (SoPC). В таких СнПК реализуемые блоки при проектировании можно *размещать в разных областях кристалла* с помощью их программирования (конфигурирования). Позднее в "системы на кристалле" стали вводить аппаратные ядра, которые значительно более компактны и обеспечивают работу на максимальных частотах. Характер и сложность аппаратных ядер со временем изменяются. Вначале аппаратные ядра были довольно простыми, сейчас ядрами блочных СнПК нередко служат *микропроцессоры или микроконтроллеры*. Программируемая часть блочных СнПК обычно имеет архитектуру FPGA.

Примеры систем на кристалле, имеющих однородную и блочную структуры, показаны на рис. 10.1.

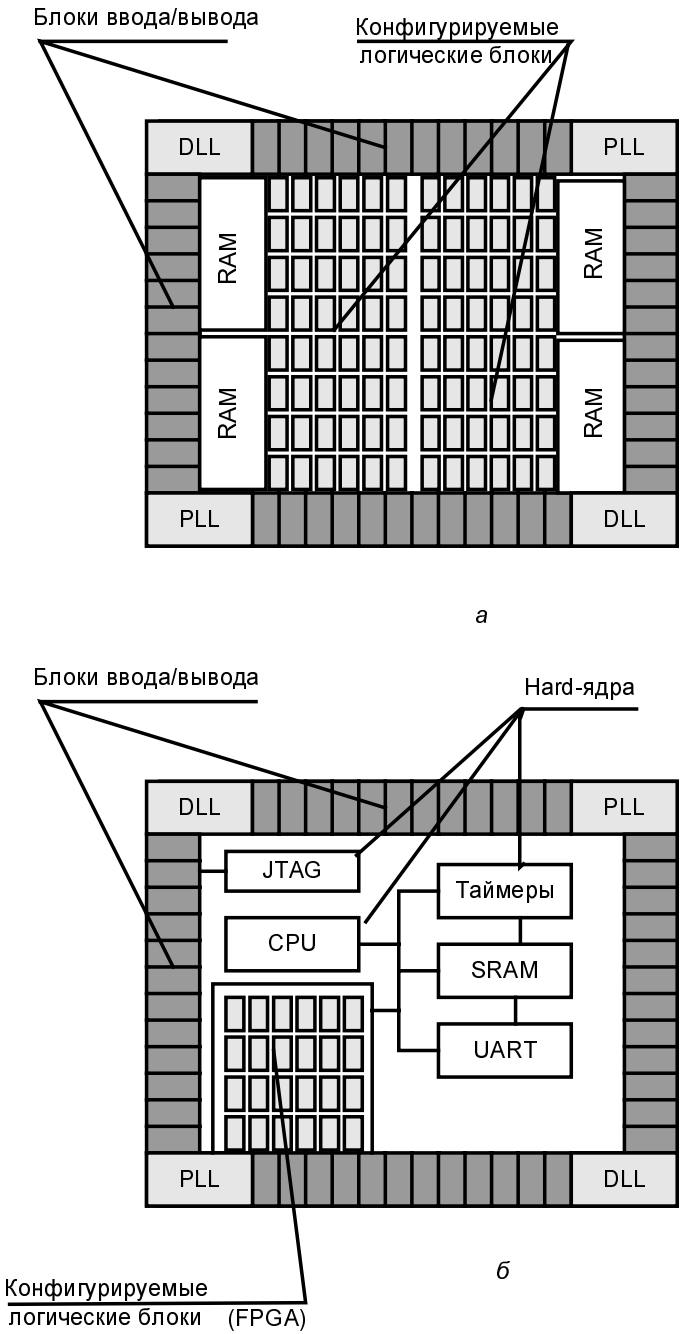


Рис. 10.1. Структуры однородной (*а*) и блочной (*б*) систем на кристалле

Сопоставление и возможности двух типов СнПК. В чем состоят преимущества и недостатки СнПК двух указанных типов? Современный уровень интеграции позво-

ляет разместить даже в однородных СнПК один или несколько процессоров, блоки памяти и многочисленные периферийные схемы, так что их состав можно сделать достаточно обширным.

Современные средства САПР с их приспособленностью к взаимозаменяемым и стандартным решениям позволяют объединять на одном кристалле виртуальные компоненты разных разработчиков. Состав системы имеет большую гибкость. Однако в однородных СнПК *не достигаются предельные быстродействия ядер и их компактность*.

Hard-ядра реализуют блоки, полученные методами проектирования заказных схем. Такие блоки в сравнении с их soft-аналогами занимают на кристалле значительно меньшую площадь (возможно, в несколько раз), поскольку они не содержат средств конфигурирования и оптимизированы для выполнения конкретной функции. По сравнению с реализацией на программируемых средствах площадь функционально аналогичных hard-ядер, выполненных по технологии МАБИС, сокращается в 5…10 раз, а при реализации по технологии стандартных ячеек в 10...20 раз. Существенно (на 20…50%) возрастает и быстродействие hard-ядер в сравнении с soft-ядрами.

В то же время предопределенность функций hard-ядер снижает универсальность микросхемы (уменьшает ее функциональную гибкость) и может сузить круг ее потребителей, что с точки зрения экономики является негативным фактором. Hard-ядра фиксированы на площади кристалла, что может затруднять решение задач размещения и трассировки для конфигурируемых областей микросхемы, препятствуя тем самым реализации максимальных показателей логической емкости и быстродействия для программируемых ресурсов схемы.

Во избежание больших потерь универсальности для hard-ядер блочных СнПК отбираются функциональные блоки, которые нужны многим потребителям. Это блоки памяти с возможностью изменять их организацию и режим работы, которые в той или иной мере нужны почти для всех систем, причем некоторые системы требуют очень больших объемов памяти. Далее можно назвать *аппаратные умножители,* широко применяемые при цифровой обработке сигналов. Успешно внедрились в СнПК *ядра интерфейса JTAG*, поскольку они выполняют важные функции и занимают небольшую площадь на кристалле. Свое место среди аппаратных ядер заняли *контроллеры интерфейсных схем.* И, наконец, в составе аппаратных ядер появились такие сложные устройства, как *микропроцессоры и микроконтроллеры*.

Соотношение программируемых и фиксированных областей в СнПК может быть самым разным. *Чем выше процент синтезируемой части микросхемы, тем больший контроль над реализацией получает разработчик проекта, но тем больше блоков при этом теряют оптимальность своих параметров.*

По поводу перспектив разновидностей СнПК высказываются разные мнения. Имеется мнение, согласно которому структуры однородного типа с их высокой степенью регулярности легче переводятся на новый технологический уровень, поэтому будут лидировать по времени выхода новой продукции на рынок. С другой стороны

отмечается, что аппаратные ядра не только гарантируют улучшенные параметры блоков, но и упрощают деловую часть разработки проекта, облегчая для проектировщика взаимоотношения с третьими лицами (поставщиками IP). Таким образом, введение аппаратных ядер в схемы СнПК — процесс противоречивый по результатам. Он сокращает площадь кристалла и ведет к достижению максимального быстродействия устройств, но и таит в себе возможность нежелательных последствий для изготовителя, т. к. может сузить рынок сбыта микросхем, а это ведет к росту цен и потере в какой-то мере конкурентоспособности продукции.

Что же будет преобладать? Ввиду больших возможностей и специфических особенностей обоих типов СнПК при решении вопроса о выборе того или иного типа учитывается целый комплекс показателей. Наиболее бесспорной областью использования блочных СнПК являются системы высшего быстродействия. В других, менее очевидных ситуациях, оценивают комплекс характеристик СнПК. В настоящее время *развиваются обе разновидности СнПК* и успехи видны в обоих направлениях.

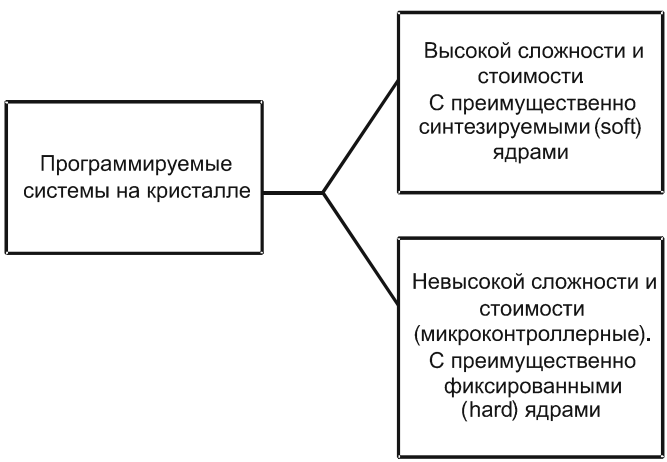


Рис. 10.2. Краткая классификация рассматриваемых далее программируемых систем на кристалле

Системы на кристалле создаются более чем десятком фирм в обстановке активной состязательности, которая выражается как в конкурентном развитии обоих рассмотренных ранее направлений, так и в разработке многих вариантов в рамках каждого из этих направлений. На (рис. 10.2) выделены типичные классы программируемых систем на кристалле, рассматриваемые в этой главе. При этом учтены признаки, связанные с характером применяемых ядер, и принадлежностью систем к продукции типа high-end (высокой сложности и стоимости) или low-end (небольшой сложности и стоимости).

Системы высокой сложности ориентированы главным образом на использование в телекоммуникации, цифровой обработке сигналов, сетевых приложениях и т. п. Системы невысокой сложности с умеренными требованиями к производительности применяются, как правило, в схемах управления техническими объектами и техно-

логическими процессами, в связи с чем названы системами *микроконтроллерного* типа. Соответственно времени их появления сначала рассматриваются системы однородного типа, а затем микроконтроллерные системы.

> ***Примечание***
>
> Для систем на кристалле с синтезируемыми ядрами (с однородной структурой) граница, за которой начинается классификационная область "системы на кристалле", не имеет четких признаков и *является условной.*

Soft-ядра процессоров

Как уже отмечалось, в проектировании "систем на кристалле" ведущую роль играет повторное использование IP-блоков или виртуальных компонентов. Им уделяется большое внимание, на международном уровне стандартизируется подход к их проектированию, решаются проблемы деловых взаимоотношений участников разработок, устанавливаются правила объединения IP или VC на основе тех или иных шин или стандартов коммутации на кристалле. Имеется и специальная литература, посвященная проектированию IP-блоков и VC — среди западной литературы прежде всего следует отметить работу [62], среди отечественной [32]. Основные (системообразующие) IP-блоки — процессорные.

Soft-ядра Nios. Для своих SoPC компания Altera разработала процессорные soft-ядра семейства Nios, второе поколение которых представлено ядром Nios-II. Основная задача процессоров того класса, к которому принадлежит семейство Nios — работа в режиме реального времени, требующая получения результата за ограниченный период и быстрого переключения между задачами в ходе их решения. Соответственно этому процессор должен иметь высокую производительность и быструю реакцию на прерывания. Этим требованиям отвечают особенности Гарвардской архитектуры и сокращенного набора простых команд (RISC-архитектуры). Именно такие решения выбраны для процессоров серии Nios (как, впрочем, и для процессорных ядер большинства других фирм).

Для soft-ядер процессоров специфична большая функциональная гибкость и *высокая степень перестраиваемости*. Функционирование встроенного процессорного soft-ядра благодаря погруженности в программируемую схемотехническую среду и собственным ресурсам перестройки можно в значительной степени варьировать. Возможности и производительность ядра можно увеличивать или уменьшать согласно требованиям проекта. Расширяемость ядра поясняется на рис. 10.3.

В операционном блоке (ALU) выполняются арифметические (блок "±"), логические (блок "&") и другие (блок "»") команды, входящие в систему команд процессора. Кроме того, входные переменные подаются на блок заказной логики, т. е. аппаратный блок собственной разработки проектировщика. С помощью этого блока можно ускорить выполнение критичных по времени фрагментов программы, добавив к системе команд свои *заказные команды*, соответствующие быстрой аппаратной реализации

выбранных фрагментов. Благодаря перестраиваемости можно не только добавлять для системы новые функции, но и удалять ненужные в данном проекте, достигая тем самым снижения стоимости кристалла, повышения его производительности и т. д. Можно создавать *новую конфигурацию ядра для каждого нового проекта.*

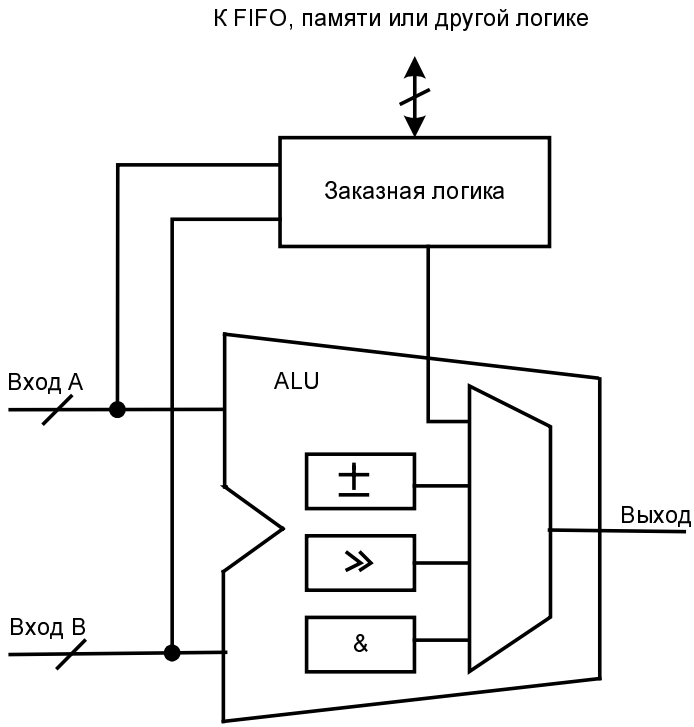


Рис. 10.3. Реализация заказных аппаратных команд для встроенного процессорного ядра

Перестраиваемость ядра проявляется также в возможности изменять назначение его выводов, что облегчает проектирование печатной платы. Например, в ядрах серии Nios выводы адреса и данных для работы с внешней памятью SDRAM можно размещать на любой стороне кристалла.

Изменяемая архитектура ядра придает SoPC высокую степень гибкости. Однако быстродействие soft-ядер не достигает максимальных значений и в сравнении с быстродействием аппаратных ядер остается умеренным.

Важная особенность конфигурируемых ядер — *изменяемый набор периферийных устройств*. Периферийные устройства встроенных процессорных ядер можно разделить на стандартные и заказные. К числу стандартных периферийных устройств, работающих с процессорным ядром, относятся UART, таймеры, порты параллельного ввода/вывода, интерфейс с различными видами памяти (SRAM, флэш), последовательный интерфейс SPI, схемы широтно-импульсных модуляторов, контроллеры дисковой памяти и др. Заказные периферийные устройства создаются проектиров-

щиком для выполнения специфических функций. Аппаратные средства реализации функций ускоряют решение задач и, кроме того, освобождают время процессора для обработки данных параллельно с работой аппаратных средств.

Процессорное ядро Nios-II. Выпускаются ядра типов Nios-II/e (Economy), Nios-II/s (Standard) и Nios-II/f (Fast). На реализацию перечисленных разновидностей ядра требуется соответственно не более 600, 1300 и 1800 логических элементов FPGA. Цифра 1800 логических элементов при размещении ядра в дешевых микросхемах соответствует стоимости 0,35 USD. Производительность процессора в микросхемах семейства Stratix достигает для перечисленных вариантов 0,16; 0,75 и 1,17 DMIPS/МГц, что при работе на максимальных частотах ядер (135…150 МГц) дает соответственно 28, 120 и 200 DMIPS. Разрядности команд, шины данных и РОН равны 32. Число РОН также равно 32. Обслуживаются 32 источника внешних прерываний. Объем внешнего адресного пространства 2 Гбайта. С помощью одной команды выполняется умножение 32-разрядных операндов с выработкой 32-разрядного результата. Имеется модуль отладки с аппаратной поддержкой, работающий под управлением интегрированных средств разработки процессорных систем Nios-II.

Ядра семейств Nios снабжены памятью и множеством периферийных устройств, так что они по существу эквивалентны микроконтроллерам. На рис. 10.4 показана структура системы с процессорным ядром Nios-II.

На рис. 10.5 показана структура процессорного ядра Nios-II.

Наряду с известными блоками в структуре ядра имеются и такие, которые требуют пояснений. Блоки памяти команд и данных, помеченные буквами TC (от Tightly Coupled), принадлежат специальному типу памяти, обладающему малыми задержками (Latency) и применяемому для критичных к быстродействию приложений. Такая память реализуется на кристалле, но вне ядра.

Наличие одновременно внутреннего и внешнего контроллеров прерываний не является противоречием, т. к. при работе внешнего контроллера внутренний отключается. Типовое использование теневых регистров состоит в ускорении контекстных переключений.

Процессорные ядра Microblaze и Picoblaze. Процессорные soft-ядра Microblaze и Picoblaze разработаны для FPGA фирмы Xilinx. Microblaze — это 32-разрядный RISC-процессор Гарвардской архитектуры, реализация которого требует затрат всего около 1000 логических ячеек при высоком среди soft-ядер своего поколения быстродействии. Процессор обеспечен обширным набором периферийных устройств и высокопроизводительными шинами для соединения ядра с другими блоками, расположенными на кристалле. Ориентирован на различные применения (сетевая аппаратура, телекоммуникации, общего назначения). Picoblaze — быстродействующий восьмиразрядный процессор. Процессоры сочетают высокое быстродействие с энергетической экономичностью. Для совместной работы с процессорами созданы soft-ядра периферийных устройств (арбитров, UART, контроллеров прерываний и др.).

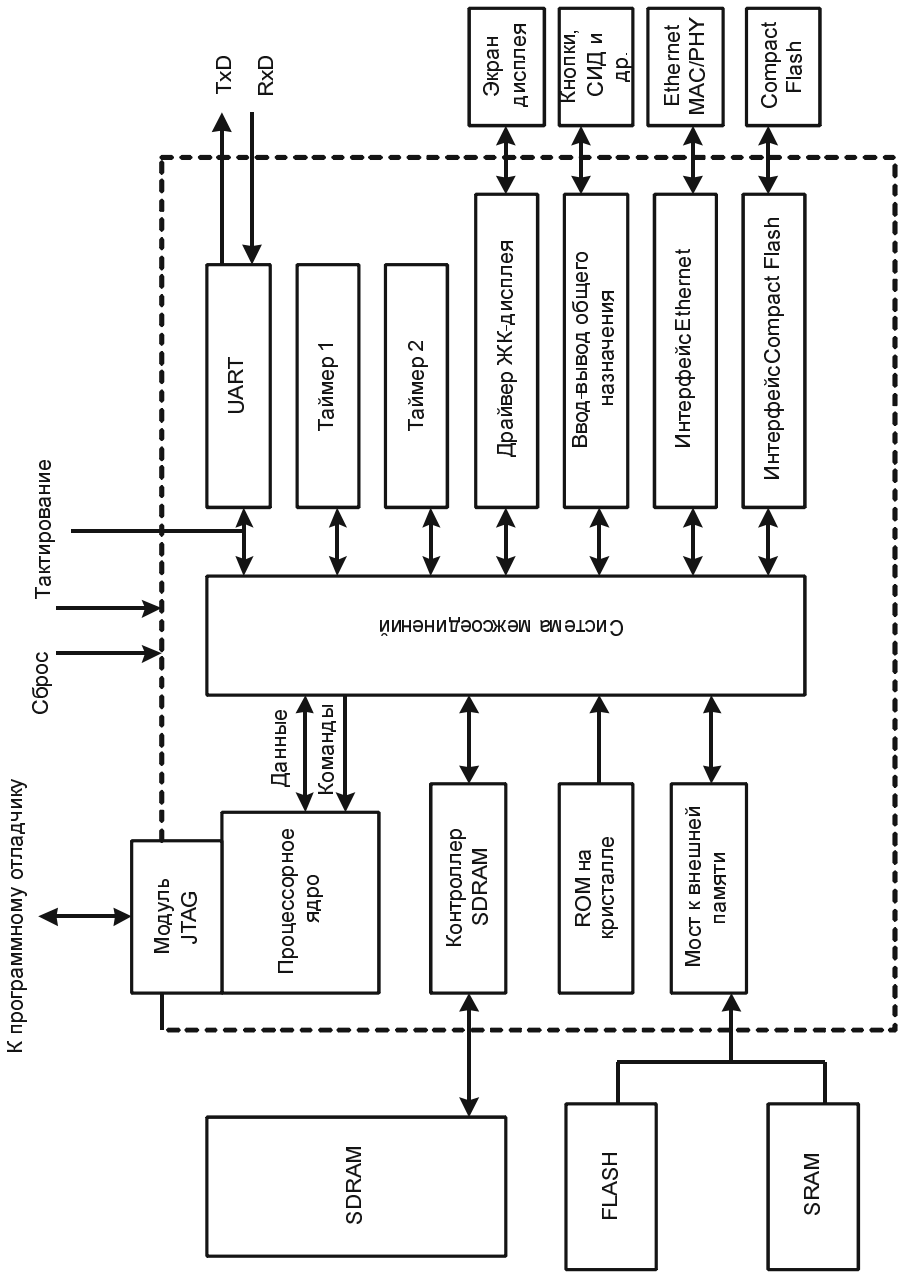


Рис. 10.4. Структура системы с процессорным ядром Nios-II

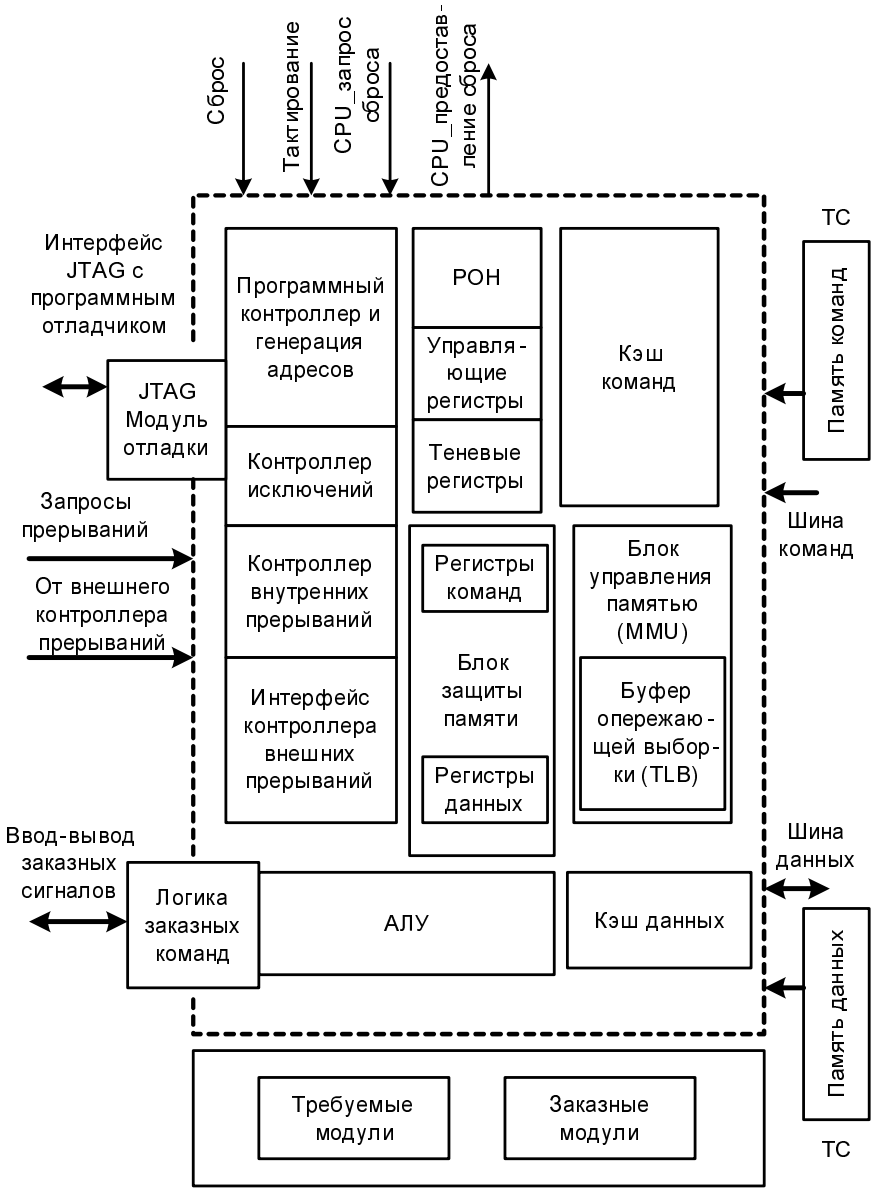


Рис. 10.5. Структура процессорного ядра Nios-II

Hard-ядра процессоров

В SoPC с hard-ядрами процессоров (фирмы Atmel, Altera, Xilinx, Cypress Semiconductor и др.) применялись преимущественно 8- и 32-разрядные процессорные ядра. В первых SoPC это были хорошо зарекомендовавшие себя 8-разрядные CISC-процессоры 8051 фирмы Intel и RISC-процессоры AVR фирмы Atmel. Позднее про-

ектировщики получили в свое распоряжение и 32-разрядные ядра (приспособленные к базовому технологическому процессу производства FPGA схемы RISC-процессоров ARM, MIPS и PowerPC).

Hard-ядра компактны (площадь 1,0…2,0 мм^2 или менее без учета кэш-памяти). Их рабочие частоты достигают уже 500 или более МГц, ядра имеют 5-ступенчатые конвейеры, 2…3 исполнительных устройства, ориентированы на малое потребление мощности. Большинство команд выполняется за один такт. Для повышения быстродействия ядра проектировались с учетом кэширования памяти (кэш L1), но некоторые при пониженном быстродействии могли работать и без кэширования.

Стандартность архитектур ядер позволяет применять для них имеющиеся инструменты и средства проектирования, что снижает трудоемкость разработок и уменьшает время выхода продукции на рынок.

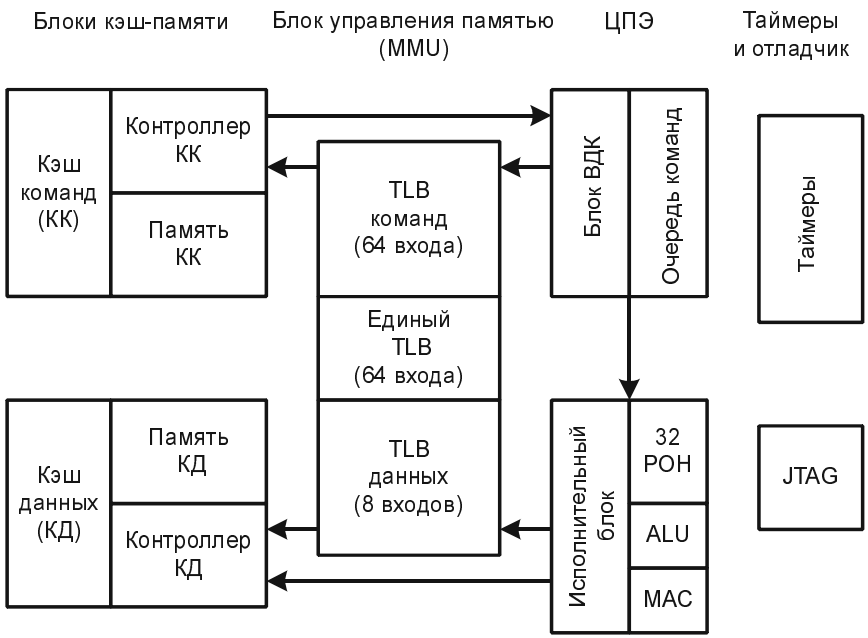
Первые процессорные hard-ядра появились в относительно несложных микроконтроллерных СнПК. Они отличались простотой и невысокой стоимостью. Затем сфера применения процессорных hard-ядер распространилась и на СнПК других классов. Сейчас hard-ядра реализуются в *широком диапазоне сложности, стоимости и других параметров*. Применяются как *специально разработанные ядра, так и заимствованные* из числа имевшихся микропроцессоров и микроконтроллеров. Хотя специальные разработки принципиально предпочтительны, поскольку в них полнее учитываются конкретные требования к ядрам, в практике большинство hard-ядер относится к заимствованным из завоевавших признание прежних решений, т. к. для успешного внедрения новых разработок важную роль играет фактор преемственности, позволяющий использовать имеющийся у проектировщиков опыт.

Среди 8-разрядных популярны ядра Intel 8051 (CISC-процессор с полным набором команд), AVR (фирма Atmel), M8C (фирма Cypress Microsystems). Среди 32-разрядных процессорных ядер наиболее известны ARM (ARM Holdings), PowerPC (IBM и др.), MIPS (MIPS Technology), ARC и Tensilica.

Ядра ARM9T. Ядра ARM9T — скалярные RISC-процессоры, ориентированные на архитектурное подмножество ARM9 Thumb системы команд ARM ISA (Instruction Set Architecture). Большинство команд процессора относится к обусловленным (Conditional), которые могут быть выполнены тогда и только тогда, когда удовлетворяется тестовое условие, определяемое самой командой. Этот подход минимизирует количество ветвлений при исполнении программы и тем самым повышает производительность процессора.

Некоторым недостатком архитектуры считается небольшой размер блока РОН (шестнадцать регистров, тогда как обычно в блоке РОН имеется 32 или более регистров). Процессор имеет пятиступенчатый конвейер, большинство операций выполняются за один такт, тактовая частота может изменяться в пределах от нуля (статическая схемотехника) до 200 МГц, кэши команд и данных имеют объемы 16 Кбайт, используется шинная система AMBA. При проектной норме 0,18 мкм процессор занимает площадь 11,8 мм^2, рассеивает мощность 160 мВт (без кэшей).

Ядра PowerPC 405C. Название происходит от Performance Optimization With Enhanced RISC. Ядро PowerPC 405C — скалярный RISC-процессор с кэшами команд и данных и тактовой частотой до 380 МГц (для версии с проектными нормами 0,18 мкм и напряжением питания 1,8 В). Совместная разработка фирм Apple, IBM, Motorola. Ядро не является малым RISC-процессором, характеризуется развитой системой команд (но все же в рамках концепции RISC) и возможностями, свойственными процессорам более высокого ранга. Процессор основан на ISA PowerPC и является представителем классических RISC с архитектурой Load/Store. В нем реализован пятиступенчатый конвейер и однотактное выполнение большинства команд (но не команд умножения и деления) в конвейерном режиме. Реализованы очереди команд для двух исполнительных блоков (ALU и MAC), статическое предсказание ветвлений, основанное на статистическом анализе стандартных программных блоков, конфигурируемые размеры кэшей (от их отсутствия до размеров 8, 16 или 32 Кбайта), механизмы виртуальной памяти, блок РОН из 32 32-разрядных регистров, два операционных устройства (ALU для сдвигов и арифметических операций и MAC для умножения, деления и умножения с накоплением), блок JTAG. Процессор занимает площадь 1,5 мм^2 и потребляет мощность от 0,5 до 1 Вт. Укрупненная блок-схема ядра приведена на рис. 10.6.



ВДК, Выборка и декодирование команд

MMU, Memory Manager Unit

TLB, Translation Look-aside Table

Рис. 10.6. Укрупненная блок-схема процессорного ядра PowerPC 405C

Через TLB обозначен буфер быстрого преобразования адресов, используемый в технике виртуальной адресации памяти. Это блок, в котором хранится таблица, используемая при преобразовании номеров виртуальных страниц в номера физических страниц.

Ядра MIPS32 4Kc. Ядро *MIPS* (Microprocessor without Interlocked Pipeline Stages, что означает "микропроцессор без задержек ожидания конвейера") фирмы MIPS Technologies лицензировано многими фирмами. Имеет кэши команд и данных по 16 Кбайт, 5-ступенчатый конвейер, аппаратные средства для выполнения операций умножения, деления, умножения с накоплением. Рабочая частота ядра 200 МГц (для технологии 0,25 мкм), рассеиваемая мощность 400 мВт (2 мВт/МГц). Используется с шинной системой AMBA.

Процессорное ядро MIPS32 4Kc — ориентированный на применение в СнПК (в качестве как hard-ядер, так и soft-ядер) вариант имевшихся ранее разработок RISC-процессоров фирмы MIPS Technology, особенно широко признанных в области решения мультимедийных задач с разрядностями данных как 32, так и 64. Предусмотрено добавление к нему кэшей команд и данных емкостью до 16 Кбайт и блока тестирования с интерфейсом JTAG. Имеется специальный блок умножения/деления, выполняющий операции быстро (операцию MAC за 1 или 2 цикла при разрядности до 32×32). Блок РОН содержит 32 регистра. Ядро имеет четыре исполнительных блока (для операций в АЛУ и сдвигателях, для операций умножения и деления, для управления ветвлениями и для управления процессором в части реализации привилегированных функций и исключений). Поддерживается подключение сопроцессора, блок MMU транслирует виртуальные адреса памяти в физические, реализованы механизмы защиты памяти. При топологических нормах 0,25 мкм ядро занимает на кристалле площадь 3 мм^2 (без кэшей).

Шинные системы

Для эффективного взаимодействия процессорных ядер с главной памятью, периферией и устройствами, реализованными в FPGA, нужны *шинные системы высокой производительности*. Этому вопросу при разработке ядер уделялось большое внимание. Была принята ориентация в первую очередь на две шинные системы: AMBA от фирмы ARM и CoreConnect от фирмы IBM.

В шинной системе AMBA определены 3 шины: две системные, называемые AHB (AMBA High-Speed Bus) и ASB (AMBA System Bus) и периферийная APB (AMBA Peripheral Bus). Системные шины является высокоскоростными, не мультиплексируются (имеют отдельные линии для адресов и данных). Более поздняя шина AHB имеет более изощренную архитектуру и наибольшее быстродействие, поддерживает пакетные передачи.

Шинная система CoreConnect состоит из локальной процессорной шины PLB (Processor Local Bus) и периферийной шины OPB (On-Chip Peripheral Bus). Шины не мультиплексируются, поддерживают одновременность чтения и записи (имеют

раздельные шины данных для этих режимов), настраиваются на версии разных разрядностей (от 16 до 128). Частота тактирования PLB для разных разрядностей составляет 66, 133 и 183 МГц.

§ 10.2. FPGA класса "система на кристалле"

Этот класс FPGA представлен микросхемами с триггерной памятью конфигурации и небольшим количеством схем с перемычками antifuse и флэш-ключами. В производстве микросхем высшей сложности с триггерной памятью конфигурации ведущие позиции занимают фирмы Altera и Xilinx, острая конкуренция между которыми сопровождается почти одновременными выпусками микросхем приблизительно равного научно-технического уровня. В производстве FPGA с перемычками и флэш-ключами более всего известна фирма Actel.

Серия Stratix

Фирма Altera начала выпуск микросхем класса "систем на кристалле" около 15 лет тому назад. Сейчас это направление представлено серией микросхем Stratix, современным семейством которых является Stratix-4. В составе семейства разработаны варианты, оптимизированные для разных областей применения.

Микросхемы Stratix-4 выпускаются в трех вариантах:

- *вариант 4E* — содержит расширенные ресурсы логики, памяти, умножителей и т. п.;
- *варианты 4GX* и *4GT* — отличаются от варианта *4E* несколько меньшими ресурсами логики и памяти, но зато имеют по 48 полнодуплексных передатчиков на основе CDR (Clock-Data Recovery), причем у варианта GX передатчики имеют быстродействие 8 Гбит/с, а у варианта GX —11,3 Гбит/с.

Основные сведения о варианте Stratix-4E даны на рис. 10.7.

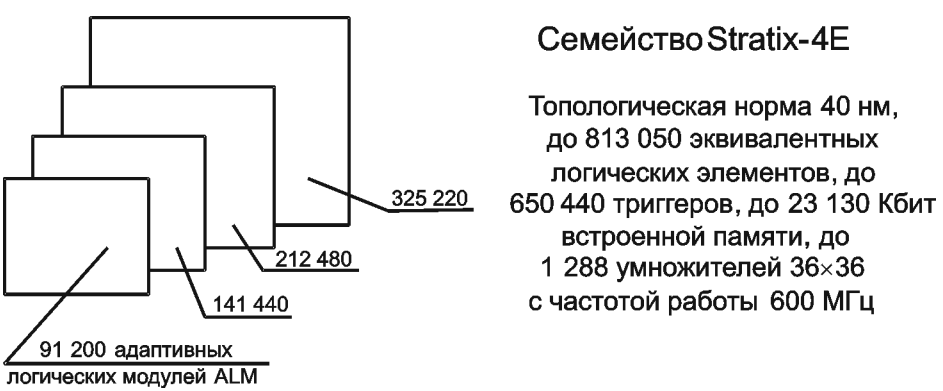


Рис. 10.7. Основные сведения о семействе Stratix-4E

План расположения ресурсов на кристалле микросхем семейства Stratix-4 показан на рис. 10.8.

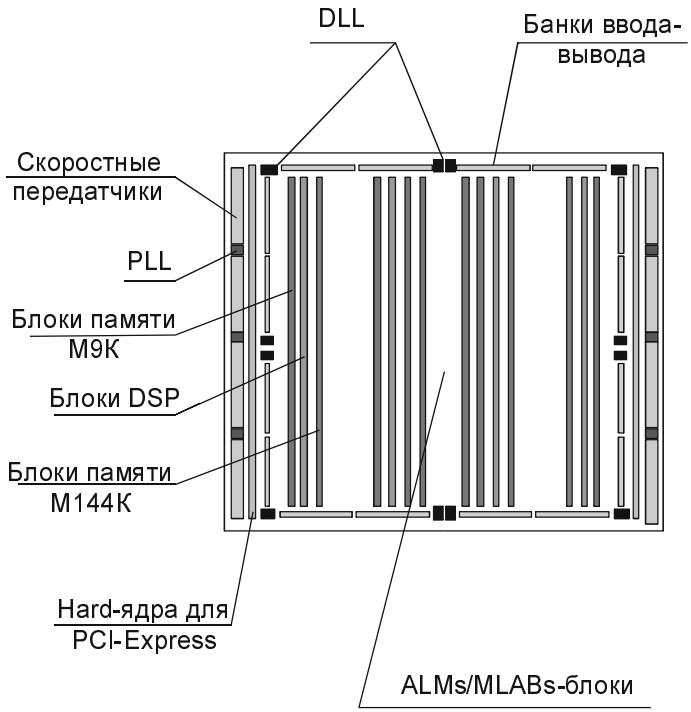


Рис. 10.8. План размещения ресурсов на кристалле микросхемы Stratix-4

Микросхемы предназначены для использования в сложных устройствах и квалифицируются как *наиболее сложные и производительные* среди современных FPGA (заметим, что такую же оценку дает и фирма Xilinx для своей серии Virtex-6).

К особенностям последних семейств Stratix следует отнести так называемую *технологию программируемой мощности* (PPT, Programmable Power Technology), согласно которой для минимизации мощности, потребляемой в каждом блоке (логическом, памяти, ЦОС-блоке), используются сведения о реализуемом проекте. Средства проектирования Quartus II автоматически оценивают требуемую производительность блоков. Изначально устанавливаются режимы малой мощности блоков, а затем блоки критических по времени путей переводятся в режим высокой скорости. Таким образом, блоки ранжируются по потребляемой мощности и, как заявлено фирмой, PPT обеспечивает высшее быстродействие и, одновременно, минимально возможное потребление мощности для данного конкретного проекта.

Целям снижения потребляемой мощности служит и возможность *выбора напряжения питания* ядра. Для проектов высокого быстродействия назначается напряжение питания 1,1 В, а для проектов минимальной мощности — 0,9 В.

В число основных частей микросхем входят адаптивные логические модули ALM, блоки быстродействующей памяти с изменяемой организацией и ЦОС-блоки. Бан-

ки ввода/вывода обеспечивают интерфейс для взаимодействия с различными внешними устройствами. Имеются средства шифрования битовых потоков конфигурирования по стандарту AES с длиной ключа 256 бит.

Функциональные преобразования выполняются модулями ALM, в которых одновременно присутствуют табличные преобразователи (LUT-блоки), сумматоры, другие логические расширения и триггеры (рис. 10.9). Модули ALM могут работать в одном из пяти режимов, в каждом из режимов ресурсы модуля используются различно. Модули при компиляции проекта средствами САПР автоматически конфигурируются соответственно тому или иному режиму.

В микросхемах Stratix-3/4 применена трехуровневая иерархия блоков памяти TriMatrix. Имеются блоки трех типов — MLAB, M9K и M144K. Емкость блоков MLAB 640 бит, они служат для построения регистров сдвига, небольших буферов FIFO, линий задержки в схемах фильтров и т. п. Емкость блоков M9K равна 9216 бит, они служат блоками общего назначения. Блоки M144K имеют емкость 147456 бит, они предназначены для хранения программ, видеоданных и т. п. Память работает на частотах до 600 МГц. Все блоки независимо конфигурируются как одно- или двухпортовое RAM, FIFO, ROM или сдвигающий регистр.

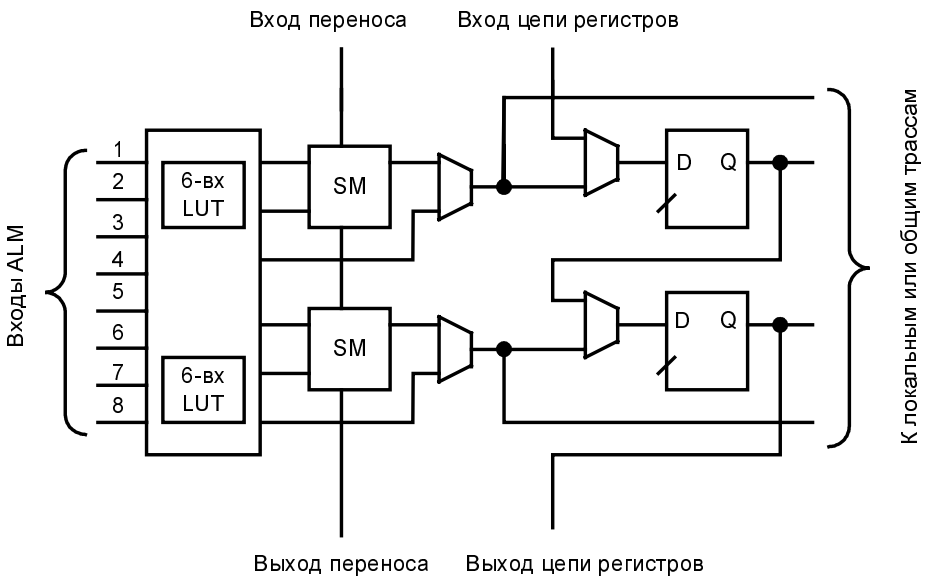


Рис. 10.9. Адаптивный логический модуль микросхем Stratix-3 и Stratix-4

Большое внимание уделено управлению тактированием, что особенно существенно для столь быстродействующих схем. Микросхемы семейства имеют до 16 глобальных, до 88 региональных цепей тактирования с быстродействием до 800 МГц и от 112 до 132 периферийных цепей тактирования.

Банки блоков ввода/вывода поддерживают более 40 стандартов, в том числе и наиболее производительные интерфейсы. Для внешней памяти поддерживаются интерфейсы DDR SDRAM (частота 200 МГц, скорость передачи 400 Мбит/с), DDR2 SDRAM и DDR3 SDRAM (частота 400 МГц, скорость передачи 800 Мбит/с), QDR2 (частота 350 МГц, скорость передачи 1400 Мбит/с), RLDRAM2 (частота 400 МГц, скорость передачи 800 Мбит/с). Микросхемы имеют до 130 дуплексных каналов LVDS со скоростью передачи 1,25 Гбит/с. Осуществляется динамическая подгонка фазы и терминирование дифференциальных линий на кристалле. Буферные каскады имеют программируемые крутизну фронтов выходных сигналов и нагрузочную способность. Предусмотрена программируемая компенсация задержек сигналов из-за различной длины трасс их передач.

Основные характеристики микросхем Stratix-3L и Stratix-4E приведены в табл. 10.1.

Таблица 10.1

Характеристики	**Значение**	
Семейство	**Stratix-3L**	**Stratix-4E**
Число ALM, тыс.	19…135	91…325
Логических элементов, тыс.	47…338	228…813
Частота, МГц	500…600*	600
Число регистров, тыс.	38…270	182…650
Блоков M9K	108…270	1 235…1 610
Блоков M144K	6…48	22…60
Общая емкость встроенной памяти, Мбит	1,8…17,2	17…33
Блоков MLAB	594…4 225	4560…10624
Умножителей 18 × 18	216…768	960…1 288…
Выводов пользователя	288…1104	780…1 760
Ориентировочная стоимость, USD	2 500…4 500	—

* Частота работы отдельных устройств.

Для проектов, которые уже хорошо проверены практикой и имеют большую тиражность производства, нет смысла сохранять на кристаллах средства репрограммирования. В таких случаях продукцию можно существенно удешевить, переведя реализацию проекта с FPGA на структурированный базовый кристалл. Для FPGA серии Stratix разработаны такие кристаллы — серия HardCopy (см. следующую главу).

Серия Virtex

Серия Virtex фирмы Xilinx — FPGA высшей сложности и быстродействия. Современные микросхемы этой серии представлены семейством Virtex-6. Архитектура микросхем Virtex отражает идеологию FPGA-платформ, поскольку созданию разнообразных проектов на их базе содействует не только программируемость схемотехнических ресурсов, но и разбиение этих ресурсов на *группы* (домены) с различными функциональными ориентациями. На кристалле можно комплектовать разные наборы доменов с оптимальным для данного проекта сочетанием свойств.

Как обычно, переход к новому поколению микросхем на основе технологии с уменьшенным минимальным размером сопровождается ростом их логических ресурсов, памяти и быстродействия, а также снижением стоимости и потребляемой энергии. Это в полной мере касается и перехода от прежних семейств серии Virtex к семейству Virtex-6.

Главное архитектурное новшество, появившееся впервые в микросхемах Virtex-4 — модульный блок ASMBL (Application-Specific Modular Block, произносится "assemble"), имеющий "доменную" (столбиковую) структуру (рис. 10.10, *а*). В доменах реализованы различные функции из числа указанных на рисунке. Для примера на рис. 10.10, *б* показан блок ASMBL с набором конкретных доменов. В микросхемах Virtex-6 реализованы блоки ASMBL третьего поколения.

Начиная с семейства Virtex-5 (2007 г.) в логическую ячейку (1/4 секции) входит *6-входовой LUT-блок* (ранее было 4 входа). Хотя это означает переход от 16 бит памяти в LUT-блоке к 64 битам, подобное решение сочли рациональным, поскольку оно позволяет сократить логическую глубину схем обработки данных и тем самым ускорить ее. Базовая разрядность блочной памяти и схем ЦОС была равна 18, для восприятия таких операндов нужны 3 шестивходовых LUT-блока, тогда как при применении 4-входовых блоков их потребовалось бы 5. Меньшее число блоков, анализирующих слова базовой разрядности, сокращает глубину логики их обработки.

Ядрами для микросхем могут служить процессоры Picoblaze, Microblaze и PowerPC405. Новый контроллер (интерфейс *APU*, Auxiliary Processor Unit) упрощает встраивание аппаратных ускорителей для ядра PowerPC путем создания прямого интерфейса между конвейером процессора и логикой FPGA. Благодаря прямому интерфейсу резко снижается число тактов шины, необходимое для доступа к аппаратуре ускорителя.

Выпускаются четыре варианта микросхем Virtex-6:

- LXT для высокопроизводительных средств решения логических задач при расширенных возможностях последовательных передач;
- SXT для решения задач ЦОС с высокой производительностью при расширенных возможностях последовательных передач;
- HXT с максимальными возможностями последовательных передач;
- CXT для решения задач, требующих применения последовательных передач со скоростью 3,75 Гбит/с при соответствующих возможностях логики.

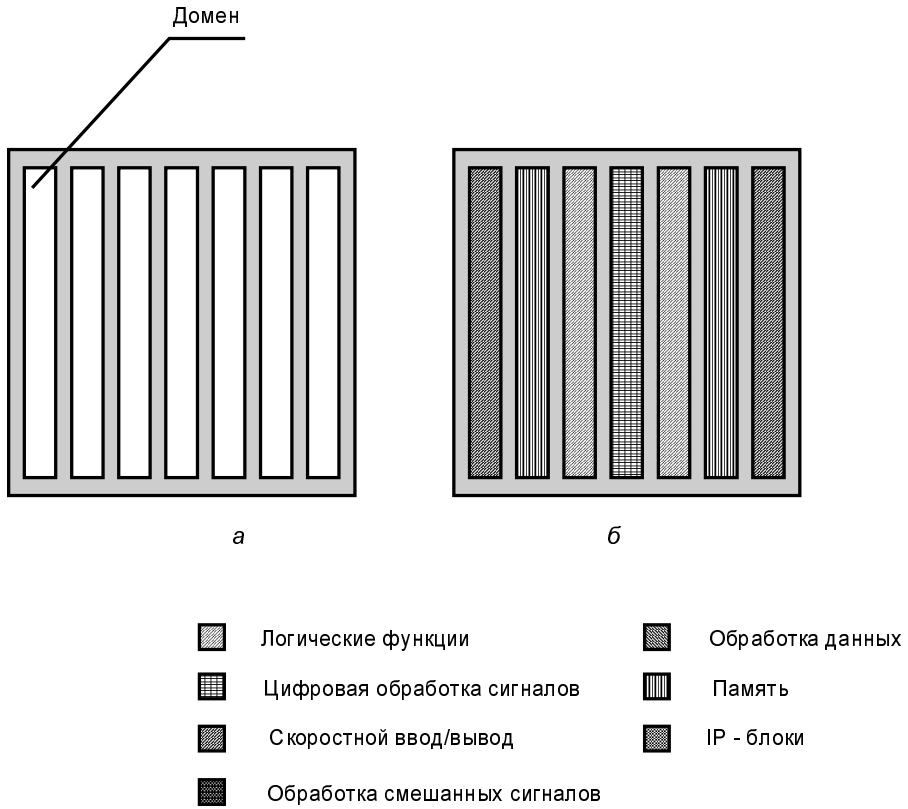


Рис. 10.10. Структура блока ASMBL (*а*) и блок с набором конкретных доменов (*б*)

Основные сведения о микросхемах варианта LXT представлены на рис. 10.11.

К существенным достоинствам семейства относятся:

- специальные секции ЦОС (DSP48E1 Slices) с умножителями 25×18, работающими в дополнительных кодах на частоте 600 МГц. Возможно конфигурирование секции в виде арифметического или логического устройства (последнее способно воспроизводить 10 функций двух переменных). Аккумуляторы могут быть использованы и как реверсивные счетчики, а умножители как быстрые сдвигатели;
- иерархия блоков памяти, в которую входит распределенная память, двухпортовые блоки встроенной памяти емкостью 36 Кбит (эти блоки можно применять и как пару блоков по 18 Кбит). Поддерживается реализация буферов FIFO с программированием параметров. Обмен с памятью осуществляется по технологиям DDR3 SDRAM, QDR2 SRAM, RLDRAM2, FCRAM2 и др.;
- усовершенствованные средства управления синхронизацией, позволяющие обслуживать множество областей синхронизации с дифференциальными каналами связи на высоких частотах и с пониженным дрожанием фронтов (Jitter). Введены специальные средства синхронизации для дифференциальных каналов;

- высокоскоростные блоки ввода/вывода данных SelectIO, обеспечивающие уровни сигналов от 1,2 до 2,5 В при синхронизации по методу Source-Synchronous (при этом источник данных генерирует и передает одновременно с данными свой синхросигнал).

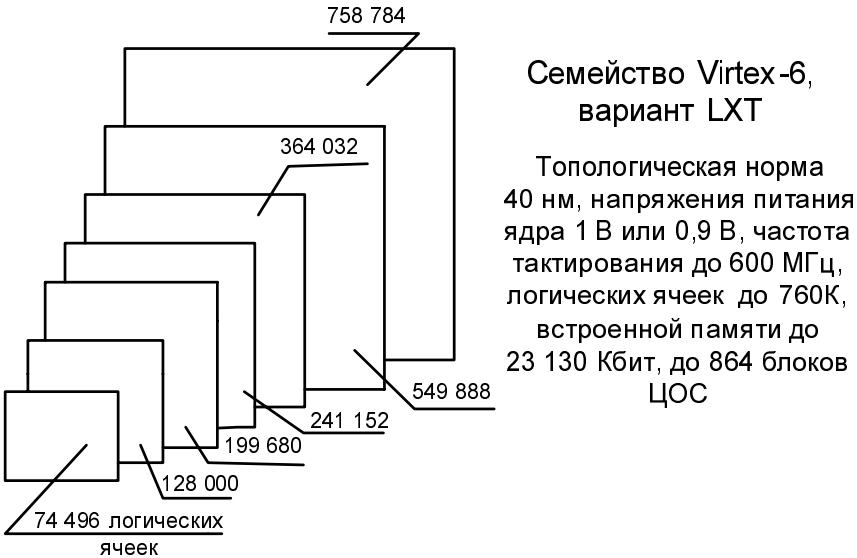


Рис. 10.11. Основные параметры микросхем Virtex-6 LXT

Многие приложения сочетают высокоскоростные последовательные передачи данных с более медленными операциями над многоразрядными операндами во внутренних устройствах. Это требует применения блоков SERDES. Каждый вход последовательных данных имеет доступ к своему блоку DES (Deserializer) с программируемой разрядностью 2, 3, 4, 5, 6, 7, 8 или 10. Каждый выход последовательных данных имеет доступ к своему блоку SER (Serializer) с программируемой разрядностью выводимых данных (до 8 разрядов при передачах с SDR и до 10 разрядов при DDR). В составе ресурсов Virtex-6 имеются эффективные средства (ChipSync Technology и др.) реализации отмеченного подхода к передачам информации:

- эффективные последовательные каналы с мультимегабитными передатчиками RocketIO третьего поколения. Для поддержки скоростных вариантов таких протоколов, как PCI Express и т. п., требуются трансиверы с высокими рабочими частотами. Это достигается с применением генераторов GTX (диапазон быстродействия от 150 Мбит/с до 6,5 Гбит/с) и GTH (диапазон быстродействия от 9,95 Мбит/с до более чем 11 Гбит/с). Целостность сигналов обеспечивается с применением специальной техники (Comprehensive Equalization). В каждый канал данных или синхронизации блока ввода/вывода SelectIO можно вводить задержки с дискретностью 78 пс для удовлетворения требований по предустановке и выдержке сигналов в трактах передач. Имеются активные средства внутрикристального терминирования цепей (DCI). Реализуется мелкозернистое

разбиение совокупности блоков ввода/вывода на банки и поддерживаются около 40 стандартов ввода-вывода.

Следует отметить также и некоторые другие возможности:

- системный мониторинг температуры и питающего напряжения, для чего на кристалле реализован аналого-цифровой преобразователь (ADC) с многоканальным входом (до 17 каналов). Температура контролируется с погрешностью не более 4 °C, а напряжение питания с погрешностью до 1%;
- шифрование битового потока конфигурирования по стандарту AES;
- включение в микросхемы встроенных средств соединения Ethernet, допускающих прямое соединение кристаллов без затрат ресурсов программируемой логики и наличие трехрежимных блоков Ethernet-MAC (Media Access Controllers), поддерживающих скорости 10/100/1000 Мбит/с.

Семейство Virtex-5 (год выпуска 2007 г., топологическая норма 65 нм, медные соединения, напряжение питания ядра 1,0 В). Микросхемы семейства содержат до 207 тыс. логических ячеек и до 12 Мбит блочной памяти, внутренняя частота до 550 МГц. Емкость основного блока встроенной памяти — 36 Кбит.

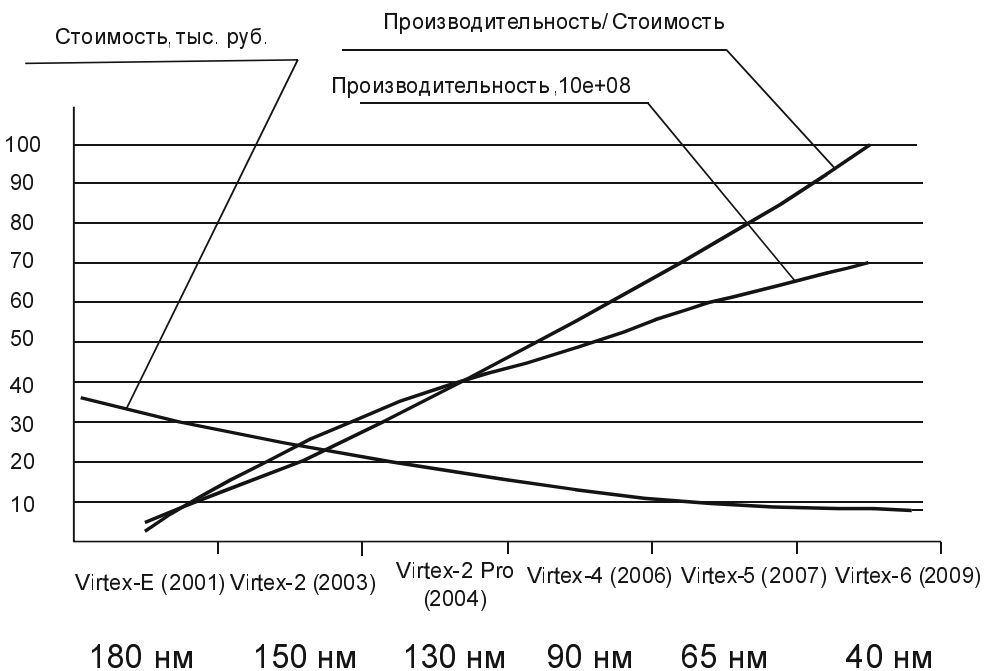


Рис. 10.12. Основные показатели развития FPGA семейства Virtex фирмы Xilinx

В память введены дополнительные схемы, нужные для построения буферов FIFO, и схемы коррекции ошибок. Разрядность блоков памяти может меняться в пределах от 1 до 72 (до 36 в режиме истинной двухпортовости). Частота работы блоков па-

мяти достигает 550 МГц. Поскольку проблема трассировки в FPGA остается важной (практически не удается получить 100% заполнение кристалла из-за нехватки ресурсов трассировки), в микросхемах Virtex-5 к средствам межсоединений были добавлены диагональные связи. В схемах управления тактовыми сигналами CMT (Clock Manager Tile) наряду с DLL применены и PLL.

Семейство Virtex-5 подразделяется на 4 подсемейства:

- LX — для проектов общего назначения;
- LXT — для проектов с расширенными возможностями последовательных интерфейсов;
- SXT — для обработки сигналов;
- FXT — для встроенных систем.

Сведения о FPGA Virtex-6 и Virtex-5 приведены в табл. 10.2.

Таблица 10.2

Параметр	Virtex-6, вариант LXT	Virtex-5, вариант LX
Логических элементов	75К…759К	19К…207К
Блоков ЦОС	288…864	32…192
Емкость распределенной памяти, бит	1М…8,3М	320К…3420К
Емкость блочной памяти, бит	5,6М…26М	1152К…11,6 М
Число выводов пользователя	360…1200	400…1200
Ориентировочная стоимость, USD	—	270…3000

На примере серии Virtex-6 можно видеть закономерности развития ПЛИС при смене их поколений, связанной с совершенствованием технологических процессов изготовления микросхем. На рис. 10.12 приведены графики изменения стоимости, производительности и отношения производительность/стоимость для поколений серии Virtex-6 (в книге И. А. Каляева и др. Реконфигурируемые мультиконвейерные вычислительные структуры. 2-е изд. Ростов-на-Дону, изд-во ЮНЦ, 2009 г., 344 с.).

Микросхемы с флэш-памятью конфигурации

На основе флэш-памяти конфигурации также созданы микросхемы серии ProASIC типа "системы на кристаллах" (фирма Actel). Логические блоки этих микросхем отличаются мелкозернистостью. Семейство ProASIC имеет варианты 3/E, nano, 3L, в которых реализуются микросхемы со сложностью до 3 млн системных вентилей и числом вводов-выводов пользователя до 620, емкость встроенной памяти до 500 Кбит. Предусмотрена работа с процессорным ядром ARM Cortex-M1. Систем-

ное быстродействие оценивается частотой до 350 МГц при частоте работы внутренних блоков до 500 МГц. Поддерживаются до 19 стандартов ввода-вывода. Производительность быстрых каналов передач — 700 Мбит/с. Число триггеров доходит до 10 700, число блоков ввода/вывода пользователя до 684.

Микросхемы потребляют крайне малую мощность (3 мВт) и имеют крайне малую стоимость (вплоть до 0,49 USD у младших представителей семейства со сложностью 10…15 тысяч системных вентилей).

Размещение ресурсов микросхем семейства ProASIC3 показано на рис. 10.13.

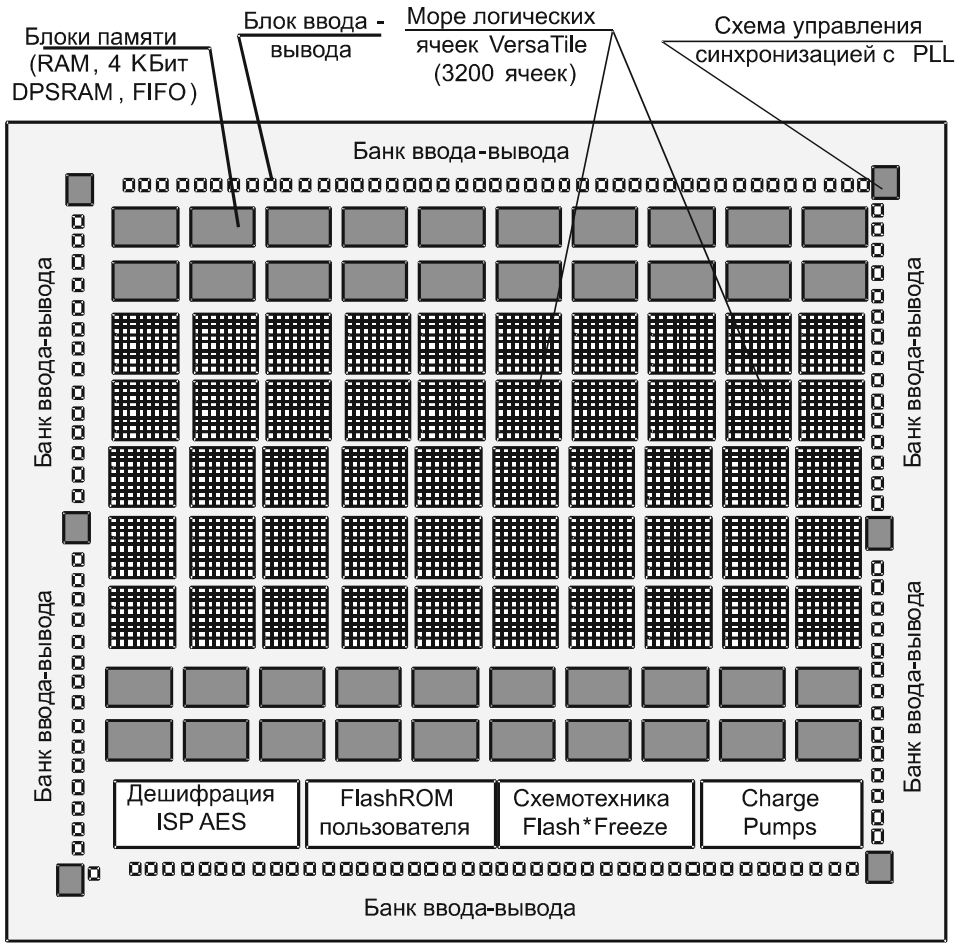


Рис. 10.13. Размещение ресурсов микросхем ProASIC3 на кристалле и состав ячейки VersaTile

§ 10.3. Системы на кристалле микроконтроллерного типа

Системы на кристалле невысокой сложности и стоимости (микроконтроллерного типа) с процессорными и другими hard-ядрами появились впервые в продукции фирм Triscend и Atmel. Далее появились и другие разработки. В микросхемах семейства PSoC компании Cypress Semiconductor сочетаются цифровые и аналоговые программируемые блоки.

Семейство FPSLIC. Разработка семейства AT94K фирмы Atmel, называемого *FPSLIC* (Field Programmable System-Level Integration Chip), была признана лучшим проектом США в области электроники в 1999 году. В одном кристалле FPSLIC были объединены микроконтроллер и FPGA, которые уже производились фирмой Atmel. Кроме них, к основным блокам кристалла относилась статическая память SRAM. В архитектуре микросхем FPSLIC были представлены следующие блоки:

- процессорное ядро AVR и его периферия;
- память программ и память данных;
- FPGA AT40K с логической емкостью от 10 тыс. до 40 тыс. эквивалентных вентилей.

Поскольку к настоящему времени микросхемы FPSLIC стали уже устаревшей продукцией, не будем рассматривать их подробно. Остановимся лишь на концепции кэш-логики, связанной с этими микросхемами, но интересной и вообще с точки зрения теории построения цифровых устройств.

Концепция кэш-логики. Важная особенность микросхем FPSLIC — *способность* к реализации концепции *кэш-логики* (Cache Logic), впервые введенной фирмой Atmel. Кэш-логика — новое достижение в области разработки адаптивных систем, она позволяет полностью или частично конфигурировать систему "на лету" без потери имевшихся данных и нарушения работы ее неизменяемой части. Данные, которые были выработаны к моменту перестройки той или иной части системы, сохраняются. Та часть аппаратуры, в которой в данный момент идет обработка информации, представлена "нормальной" схемой соответствующей конфигурации, созданной в FPGA, а пассивная часть аппаратуры представлена данными, сохраняемыми в недорогих устройствах памяти. В итоге экономно выполняются требуемые преобразования. Когда активизируются новые операции, новая конфигурация записывается поверх старой.

Задачи обработки информации в конечном счете раскладываются на стандартные операции низшего иерархического уровня (сдвиги, сложения, умножение, мультиплексирование и т. п.), причем в одно и то же время активна только малая часть операторов, соответствующих этим функциям. Каждая из подобных функций может быть использована многократно в разных обстоятельствах. Исключая избыточность и контролируя условия появления каждой операции, можно так организовать систему, что многие функции будут воспроизводиться с помощью небольшого

числа простых схем и, следовательно, на недорогих средствах программируемой логики. Так, например, в одном из известных примеров задача решалась обычным способом с использованием 10 тыс. вентилей, а на основе программируемых схем и концепции кэш-логики потребовались всего 2 тыс. вентилей и запоминание неизменной в данное время информации в дешевой системной памяти.

Концептуально различают *предопределенную и динамическую кэш-логику*. Первая подразумевает использование предопределенных функций и макросов, хранимых во внешней энергонезависимой памяти, заранее отработанных и имеющих уже сгенерированные битовые потоки конфигурирования. Выполнением этих функций управляют средства самой кэш-логики или процессор.

Вторая разновидность кэш-логики должна определять требуемые функции с их размещением и трассировкой и генерировать битовые потоки настройки в реальном масштабе времени. Такой режим ассоциируется с работой адаптивных систем, и пока существует только как концепция, не имеющая законченной физической реализации.

Применение кэш-логики снижает сложность программируемой части аппаратуры и, в конечном счете, повышает ее надежность (вследствие сокращения числа физически существующих схемных компонентов), снижает потребляемую мощность и стоимость. Предполагается применение кэш-логики, в первую очередь, в портативной аппаратуре, системах компьютерной графики и т. п.

Разработчики таких СБИС, как FPSLIC, подчеркивают, что подобные микросхемы не следует рассматривать как просто микроконтроллеры с программируемой пользователем периферией, поскольку для таких целей могут найтись и более простые решения. Для микросхем типа FPSLIC и им подобных *эффект достигается прежде всего при достаточно полном использовании их новых функциональных возможностей.*

Серия PSoC

Системы на кристалле серии PSoC (Programmable System on Chip), в которых сочетаются конфигурируемые цифровые и аналоговые программируемые блоки, память и процессор, выпустила фирма Cypress Semiconductor. В серии PSoC выпущены три семейства: PSoC 1 с процессором M8C (тактовая частота 24 МГц), PSoC 3 с процессором 8051 (67 МГц) и PSoC 5 с процессором ARM Cortex –M3 (80 МГц).

Семейство этой серии, обладающее архитектурными новшествами и большими возможностями относительно своих предшественников — PSoC 5. В этом семействе некоторые аналоговые компоненты, реализованные на кристалле, тем не менее обладают разрешающей способностью, соответствующей 20 двоичным разрядам, используемое 32-разрядное ядро процессора ARM Cortex-M3 работает на частоте до 80 МГц, схема потребляет ультранизкую мощность. Целесообразность применения того или иного семейства зависит от решаемой задачи. Зачастую достаточно возможностей, например, семейства PSoC 3. Семейство PSoC 5 предназначено для решения наиболее сложных задач, но далее рассмотрено именно оно как последнее

к настоящему времени, воплотившее в себе наивысший уровень развития микросхем с сочетанием программируемых цифровых и аналоговых средств.

Сочетание процессора с хорошо развитыми гибкими средствами аналоговой и цифровой обработки данных, а также обширными средствами их ввода-вывода, придает системе PSoC 5 широкую область применений в промышленных, потребительских, медицинских и других приложениях. Упрощенная структура микросхемы показана на рис. 10.14 и 10.15. На рис. 10.14 представлена часть микросхемы (подсистемы процессора, памяти и другие фиксированные подсистемы), вторая часть изображена на рис. 10.15. Кроме системной шины микросхема имеет две различных системы коммутации — цифровую и аналоговую (на рисунках не показаны).

Подсистема процессора строится вокруг 32-разрядного ядра ARM Cortex-M3 с трехступенчатым конвейером, работающего на частотах до 80 МГц. В подсистеме имеется контроллер вложенных векторных прерываний, контроллер прямого доступа к памяти (24 канала), а также контроллер флэш-кэша, который ускоряет работу с памятью, а также способствует снижению потребляемой мощности, благодаря более редким обращениям к основной флэш-памяти.

Подсистема памяти снабжена устройствами типа программной памяти флэш (256 Кбайт), EEPROM с байтовой записью (2 Кбайта для пользовательских данных), SRAM (64 Кбайта).

Процессор имеет возможность индивидуально репрограммировать блоки флэш, делая возможной загрузку и коррекцию boot-данных. Проектировщик может задействовать работу с корректирующими кодами при решении задач, требующих высокой надежности. Для защиты информации, доступной пользователю, возможно запирание выбранных блоков для записи и чтения. Хранение избранных данных конфигурации в энергонезависимой памяти позволяет схеме активизироваться сразу же после сброса при включении питания.

Подсистема отладки и программирования широко использует интерфейс JTAG (4 линии) или Serial Wire Debug (2 линии).

Подсистема управления мощностью обеспечивает как широкий диапазон работы в части значений рабочих напряжений, так и широкий диапазон режимов малой мощности. Допустима работа микросхемы от стандартных источников напряжений 1,8 В ± 5%; 2,5 В ± 10%; 3,3 В ± 10%; 5,0 В ± 10% или непосредственно от батарей разных типов. Имеется встроенный конвертор, благодаря которому можно питать схему от источников столь малых напряжений, как 0,5 В, и вырабатывать нестандартные напряжения. Возможны четыре режима малого потребления мощности: активный (с током потребления 2 мА при частоте тактирования 6 МГц), ожидания (с током 20 мкА и временем выхода в активный режим менее 1 мкс), спящий (2 мкА и менее 12 мкс) и hibernate (буквальный перевод — находящийся в зимней спячке) с параметрами 300 пА и менее 100 мкс).

В подсистеме тактирования работают несколько генераторов. Внутренний генератор играет основную роль, имеет диапазон частот от 1 до 72 МГц и погрешность частоты в 1% в полном диапазоне изменения температуры и напряжения питания.

Рис. 10.14. Специализированные подсистемы микросхемы PSoC-5

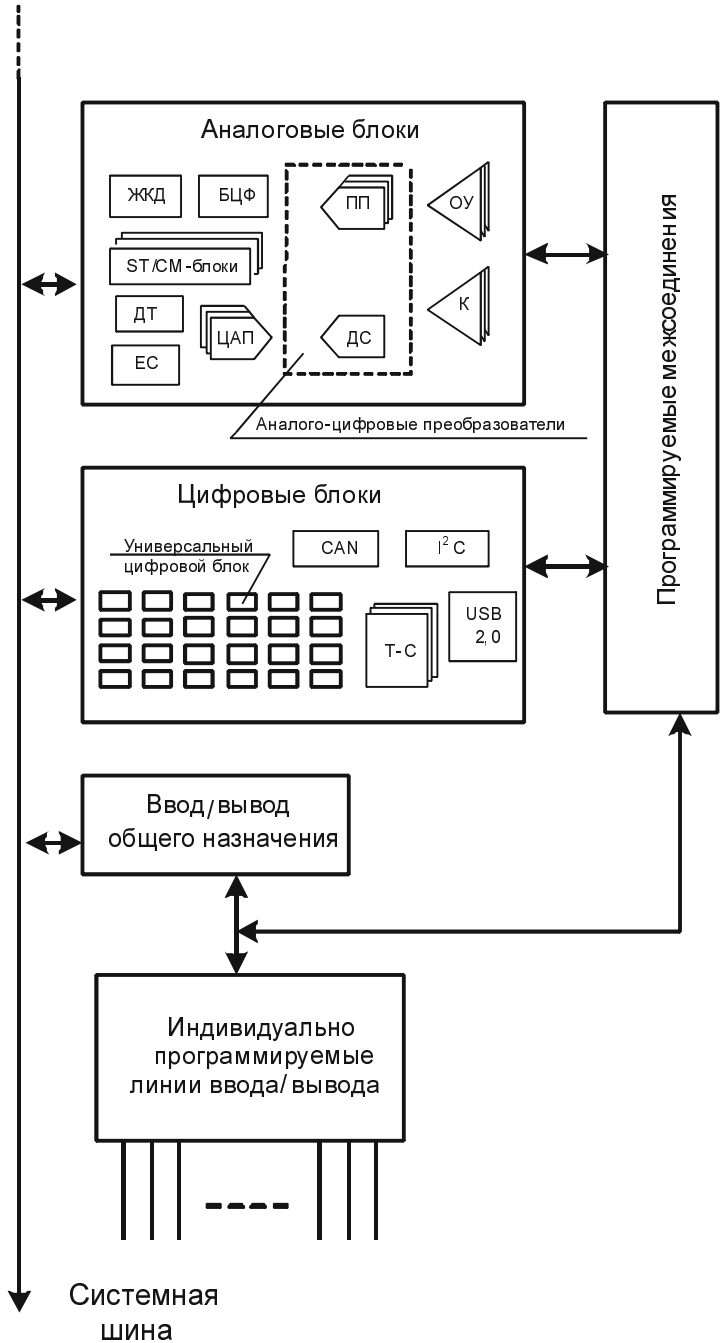


Рис. 10.15. Конфигурируемые цифровая и аналоговая подсистемы микросхемы PSoC-5

От него можно получать производные частоты, благодаря наличию PLL, в частности, можно получить частоты до 79 МГц (80 МГц с учетом допуска 1%). Кварцевый генератор, работающий в диапазоне частот от 4 до 33 МГц, обеспечивает по-

лучение частот высокой точности. Кроме того, имеется генератор с "часовым" кварцем (частота генерации 32768 Гц) в составе таймера реального времени (кварцевые резонаторы, как всегда, являются внешними элементами, находящимися вне кристалла). Отдельный низкочастотный генератор с очень малой потребляемой мощностью служит для питания таймеров спящего режима и сторожевого таймера (частоты 1 кГц, 100 кГц).

Структура цифровой и аналоговой конфигурируемых подсистем показана на рис. 10.15.

Аналоговая подсистема. В аналоговой подсистеме создаются и соединяются друг с другом или с контактами микросхемы комбинации стандартных и усовершенствованных блоков обработки аналоговой информации. Эти возможности обеспечиваются наличием таких составляющих структуры, как:

- иерархия межсоединений (от глобальных до локальных) и аналоговое мультиплексирование;
- различные АЦП и ЦАП, операционные усилители и компараторы;
- точные эталоны для генерации аналоговых опорных напряжений в блоках подсистемы.

Кроме того, в микросхеме предусмотрены специальные средства для создания устройств с емкостными сенсорами прикосновения.

В аналоговых блоках используются как схемы с точными резисторами (схемы типа CM, ContinuousMode), так и схемы с переключаемыми конденсаторами (схемы типа SC, Switched Capacitors).

На рисунке (см. рис. 10.15) приняты следующие обозначения для АЦП.

Один из АЦП (обозначен как ДС) строится по методу дельта-сигма модуляции с разрешающей способностью 20 разрядов и отличается высокой точностью (смещение нуля менее 100 мкВ, погрешность усиления 0,2%, интегральная и дифференциальные нелинейности 1 и 0,5 LSB соответственно, где LSB — цена младшего разряда двоичного кода. Отношение сигнал/шум в режиме 16 разрядов не хуже 90 дБ). Подобный АЦП пригоден для оцифровки выходных сигналов даже высокоточных датчиков. Номинальный режим преобразователя — 16 разрядов при 48К отсчетов/с. Можно сконфигурировать преобразователь для работы с 20 разрядами, если снизить скорость его работы до 180 отсчетов/с. При работе в режиме 12-разрядного преобразователя скорость данных может составлять 192К отсчетов/с.

Остальные АЦП работают по методу последовательных приближений (обозначены как ПП) с разрядностью до 12 при скорости работы 1М отсчетов/с, обладают малой нелинейностью и малым смещением нуля при отношении сигнал/шум лучше 70 дБ.

Выход любого АЦП может подключаться по выбору к входу того или иного программируемого цифрового блока через схемы прямого доступа, минуя процессор. Цифровой блок можно конфигурировать как цифровой фильтр КИХ- или БИХ-типа или как определяемое проектировщиком вычислительное звено.

Четыре скоростных восьмиразрядных ЦАП с выходами по напряжению или току работают на частотах до 8 МГц. Косвенным способом с привлечением цифровых блоков можно получить преобразование с большей разрядностью (до 10 разрядов при частотах до 48 кГц). Поддерживаются режимы широтно-импульсной модуляции.

Четыре компаратора (обозначены через К) могут опционно (по выбору) подключаться к выходам конфигурируемых цифровых функциональных блоков.

Четыре конфигурируемых SC/CM-блока для реализации в них функций операционных усилителей (ОУ), буферных повторителей, преобразователей импеданса и смесителя.

Четыре ОУ, имеющие большие выходные токи для внутреннего использования и подключения к цепям ввода-вывода.

Кроме перечисленных блоков в подсистеме имеются блок обслуживания жидкокристаллического дисплея (ЖКД), блок цифровой фильтрации (БЦФ), температурный датчик (ТД) и емкостной сенсор (ЕС).

Источник эталонного напряжения, позволяющий получать опорные напряжения, необходимые для работы многих аналоговых блоков.

Аналоговые блоки строятся на базе одного ОУ, который сочетается со схемами с переключаемыми конденсаторами либо с набором резисторов для получения нужной функциональной характеристики. К функциональным разновидностям блоков относятся: непосредственно усилители, буферы с единичными коэффициентами усиления по напряжению, ОУ с программируемыми коэффициентами усиления, преобразователи импедансов, смесители (mixers).

Непосредственно усилители предоставляют оба входа и выход для различных подключений внутренних или внешних сигналов, имеют полосу единичного усиления свыше 6 МГц и выходной ток до 650 мкА, допускают работу на нагрузочные сопротивления до 7,5 кОм. Операционные усилители с программируемыми коэффициентами усиления могут конфигурироваться на инвертирующие и неинвертирующие варианты при коэффициентах усиления до 50. При единичном усилении имеют полосу 6 МГц, при $K_U = 50$ полоса сужается до 215 кГц.

Цифровая подсистема. Цифровая подсистема включает в себя набор быстродействующих универсальных блоков УЦБ (UDB, Universal Digital Block), отличающихся малым потреблением мощности, и средства коммутации, подключающие любое устройство к любым контактам микросхемы или других блоков. Библиотека предварительно отработанных периферийных устройств, предоставляемая средствами проектирования PSoC Creator, содержит ядра UART, SPI, I^2C, LIN, CRC-генераторы, таймер, счетчик, ШИМ, конъюнкторы, дизъюнкторы и многие другие. Все эти устройства могут быть сконфигурированы в матрице УДЦ. Имеется также возможность создания устройств на базе булевских примитивов средствами графического ввода проекта. Матрица УДЦ однородна и допускает свободное размещение функциональных устройств в ее пределах. В зависимости от сложности воспроизводимых функций они могут размещаться в одном УЦБ или даже в его части, или во множестве УЦБ. Например, для построения интерфейса I^2C или UART тре-

буется не более трех УЦБ, для реализации интерфейса с внешней памятью достаточно одного.

Кроме того в подсистеме имеется таймер-счетчик (Т-С) и блоки обеспечения указанных на рисунке интерфейсов.

УЦБ состоит из двух небольших ПЛМ, структурируемого операционного устройства (АЛУ с регистрами и буферами FIFO) и гибкой схемы соединений между этими элементами, элементами ввода-вывода и другой периферией.

ПЛМ размерности 12, 8, 4 формируют комбинационные или регистровые выходы ДНФ логических функций. Имея 12 входов, ПЛМ вырабатывает 8 конъюнктивных термов, в каждом из которых может быть от 1 до 12 переменных (аргументов), для каждого аргумента может быть выбрано прямое или инверсное значение. Логическая сумма термов образует выходную функцию, в которой может быть от 1 до 8 термов.

ПЛМ могут быть единственными средствами, на которых синтезируются воспроизводимые цифровые функции. Однако более эффективно применение для этой цели комбинации ПЛМ и операционных устройств, где ПЛМ реализует только произвольную логику, а операционное устройство — более структурированные элементы. Операционное устройство — восьмиразрядное АЛУ с входными и выходными буферами FIFO, а также схемы сравнения и генерации условий. Реализован параллельный интерфейс между процессорной подсистемой и УЦБ.

Микросхемы PSoC обеспечены широким набором отладочных средств.

Контрольные вопросы

1. Какие параметры микропроцессорных систем улучшаются при их реализации на одном кристалле?
2. Какие разновидности систем на кристалле можно выделить с точки зрения областей их применения?
3. В чем состоят особенности проектирования систем на кристалле?
4. Что называют IP-ядрами (виртуальными компонентами)? Какие виды ядер используются в системах на кристалле? В чем состоят достоинства и недостатки применения ядер разных типов?
5. В чем заключается выигрыш и проигрыш введения в программируемую систему на кристалле аппаратных ядер?
6. Какие стили проектирования систем на кристалле называют блочным и платформенным?
7. Какие особенности soft-ядер процессоров специфичны для них в сравнении с традиционными реализациями процессоров?

8. Какова площадь, занимаемая на кристаллах процессорными ядрами, и какова ее доля в общей площади кристалла (ориентировочно)?
9. Какие свойства трактов передачи сигналов стремятся получить в системах на кристаллах? Почему эти свойства считают важными?
10. По каким качествам системы на кристалле с перемычками antifuse превосходят другие типы микросхем с программируемыми структурами? В чем состоят недостатки систем с программированием с помощью перемычек?
11. Укажите состав адаптивных модулей ALM микросхем Stratix-4. Назовите максимальную частоту работы блоков этой микросхемы.
12. В чем состоит новизна модульных блоков ASMBL микросхем Virtex-4 и последующих семейств серии Virtex? Чем особенность этого блока соответствует платформенному стилю проектирования систем?
13. Каковы разрядность и быстродействие АЦП, реализуемых на кристаллах микросхем серии PSoC 5?
14. В чем состоит концепция кэш-логики?
15. Аналоговые и цифроаналоговые блоки микросхем PSoC подразделяются на блоки типов Continuous Mode (CM) и Switched Capacitor (SC). Что означают эти термины?

Литература к главе: [18], [32], [62], [66], [III], [VI], [VIII], [XXXIV].

Глава 11

Микросхемы, программируемые с участием изготовителя

Микросхемы, программируемые пользователем, открыли новую страницу в развитии микроэлектроники, но не отменили ситуации, в которых по экономическим или иным соображениям целесообразно применение других разнообразных средств цифровой техники, в частности, традиционных микропроцессорных систем. Применительно к БИС/СБИС речь идет о таких альтернативных относительно ПЛИС кристаллах, которые не содержат цепей для конфигурирования схемы или имеют уменьшенное количество таких цепей. Полное исключение дополнительных схем конфигурирования специализированных схем свойственно дорогостоящему заказному проектированию. К БИС/СБИС, имеющим уменьшенное количество схемных элементов конфигурирования, относятся полузаказные, которым и посвящена эта глава.

§ 11.1. Базовые матричные кристаллы (вентильные матрицы, программируемые изготовителем)

Основные сведения

Максимальные показатели качества БИС (минимальные площадь кристалла и мощность потребления, максимальное быстродействие и т. д.) достигаются при дорогостоящем индивидуальном *полностью заказном* проектировании, учитывающем все особенности реализуемого устройства. При разработке полностью заказных схем проектировщик имеет полную свободу действий вплоть до уровня параметров транзисторов.

Стоимость разработки снижается, если при проектировании используются заранее разработанные библиотечные фрагменты схем. Такое проектирование ассоциируется с понятием "*схемы на стандартных ячейках*" (*SCL*, Standard Cells Logic). Схемы SCL уступают полностью заказным по уровню технических параметров, но дешевле в разработке и получили достаточно широкое распространение. В то же время их разработка также обходится недешево — на разработку СБИС требуются затраты порядка десяти миллионов долларов.

Схемы на основе базовых матричных кристаллов (БМК) относят к *полузаказным*. При их создании используется стандартный полуфабрикат (постоянная часть БИС), который доделывается далее по конкретным условиям заказчика (переменная часть БИС). При этом на заказчика падают расходы только по доработке полуфабриката, что в несколько раз дешевле полной разработки схемы. Соответственно снижаются и сроки проектирования. Схемы, разработанные на основе БМК, называют *МАБИС* или *БИСМ*. В английской терминологии БМК называются GA (Gate Arrays), чему в русском языке соответствует термин ВМ *(вентильные матрицы)*. Наряду с SCL вентильные матрицы входят в число ИС, называемых *ASICs* (Application Specific Integrated Circuits).

> ***Примечание***
>
> Термин *БМК* (базовый матричный кристалл) в литературе на русском языке наиболее распространен и поэтому используется и здесь, хотя термин *ВМ* (вентильная матрица) в силу единообразия с международным вариантом представляется предпочтительным.

Первые образцы БМК появились в 1975 г. как средство реализации нестандартных схем высокопроизводительной ЭВМ без применения микросхем малого и среднего уровней интеграции. БМК позволили выполнить и нетиповые части компьютера на БИС. Формулировку "позволили выполнить" в данном случае следует понимать с позиций экономических факторов. Технологически можно было реализовать БИС по индивидуальному заказу, однако стоимость таких БИС была бы неприемлемо высока.

Промышленное производство БМК широко развернулось с начала 80-х годов, и в течение 10…15 лет они составляли основу элементной базы ЭВМ. В настоящее время область применения БМК значительно сузилась из-за успешного развития конкурирующих средств цифровой техники, однако многие специалисты считают, что *структурированные* БМК сохранят свою нишу и в современных условиях.

Итак, БМК является заготовкой, которая преобразуется в требуемую схему (МАБИС) реализацией изготовителем кристалла необходимых соединений по заказу потребителя. Потребитель может создавать на основе БМК некоторое множество устройств определенного класса, заказав для кристалла тот или иной вариант рисунка межсоединений компонентов.

Первые БМК выполнялись по схемотехнике ЭСЛ, для которой полный процесс изготовления включал 13 операций с фотошаблонами (для современных технологий нужны около 40 фотошаблонов). Для изготовления схемы на основе БМК требовались только 3 индивидуальных (переменных) шаблона для задания рисунка межсоединений. Соответственно этому сроки и стоимость проектирования МАБИС сокращались в сравнении с заказными БИС в 3...5 раз. Заметим, кстати, что с уменьшением топологических норм проектирования стоимость шаблонов резко возрастает. В последние годы нормы уменьшались и составляли последовательно 350, 180, 130, 90, 65 и 45 нм. При переходе к следующему поколению стоимость

шаблонов удваивалась и к настоящему времени достигла 200…300 тысяч долларов за один шаблон.

Плата за сокращение сроков и стоимости проектирования — неоптимальность результата. МАБИС проигрывают по площади кристалла и быстродействию полностью заказным схемам, т. к. часть их элементов оказывается избыточной (не используется в данной схеме), взаимное расположение элементов и пути межсоединений не являются оптимальными и т. д.

Классификация БМК

Классификация БМК показана на рис. 11.1.

Масочное и лазерное программирование. Соответственно технологии создания соединений различают БМК с масочным и лазерным программированием. Основная часть БМК — *масочные* (*MPGA*, Mask Programmable Gate Array). Для получения на их основе требуемой МАБИС для переменной части межсоединений изготавливаются фотошаблоны, с помощью которых выполняются операции нанесения на кристалл металлизированных дорожек. В БМК *с лазерным программированием* межсоединений (*LPGA*, Laser Programmable Gate Array) вначале изготавливаются все возможные межсоединения, а при программировании часть их разрывается под действием точно сфокусированных лазерных лучей. Программирование также требует обращения на предприятие электронной промышленности, при этом сроки выполнения заказа в сравнении с MPGA сокращаются, но стоимость LPGA выше.

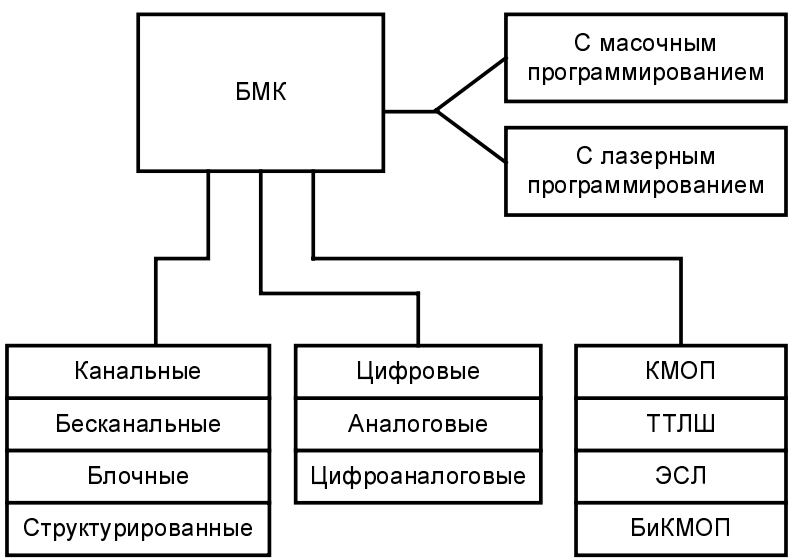


Рис. 11.1. Классификация базовых матричных кристаллов

По архитектурным признакам БМК делятся на канальные, бесканальные, блочные и структурированные. Все БМК имеют внутреннюю область (ВО), окруженную периферийной областью (ПО), расположенной по краям кристалла. В периферийной области расположены ПБЯ, набор схемных элементов которых ориентирован на решение задач ввода/вывода сигналов, а также контактные площадки (КП).

Канальные БМК. Во внутренней (центральной) области таких БМК (рис. 11.2) расположена матрица базовых ячеек — 1 и каналы для трассировки — 2. Каналы могут быть вертикальными и горизонтальными, как на рис. 11.2, *а*, либо только вертикальными (рис. 11.2, *б*). Канальные БМК могут иметь большие возможности по созданию связей, но имеют низкую плотность упаковки из-за значительных затрат площади кристалла на области межсоединений. Канальная архитектура характерна для биполярных БМК, т. к. значительная мощность рассеивания биполярных БЯ сама по себе препятствует их плотной упаковке.

Бесканальные БМК. Повышение уровня интеграции БМК ведет к быстрому усложнению межсоединений в МАБИС, а значит и площади, отводимой для них. Поиск путей создания БМК высокого уровня интеграции с минимизацией площади, отводимой под межсоединения, привел к *бесканальной архитектуре* БМК (бесканальные архитектуры имеют также названия *"море вентилей"*, *"море транзисторов"*, *"универсальные БМК"*). Внутренняя область такого БМК содержит плотно упакованные ряды базовых ячеек и не имеет фиксированных трассировочных каналов (рис. 11.2, *в*). В этом кристалле любая область, в которой расположены БЯ (строка, столбец либо их часть), может быть использована для создания как логической схемы, так и межсоединений. Вследствие более рационального расположения связей в бесканальном БМК уменьшается и задержка передачи сигналов по связям, т. к. длины и паразитные емкости межсоединений уменьшаются.

Бесканальные БМК характерны для КМОП-схемотехники, в которой компактность схемных элементов и малая мощность рассеяния БЯ при их работе на не слишком высоких частотах способствуют возможностям плотной упаковки базовых ячеек.

В бесканальных БМК при проектировании МАБИС площадь кристалла может перераспределяться между трассировочными каналами и функциональными ячейками, поэтому потери площади кристалла снижаются. В БМК с плотным расположением на рабочем поле рядов транзисторов в одних рядах реализуются логические элементы, в других — трассировочные каналы, в которых транзисторы остаются нескоммутированными и не используются (над ними проходят трассы). В зависимости от загруженности каналов, для них может быть отведено различное число рядов транзисторов.

В КМОП БМК используются и архитектуры с переменной длиной ячеек (рис. 11.2, *г*). Здесь каждая строка состоит из пар n- и p-канальных транзисторов. Если в такой длинной строке в заданных местах запереть пары транзисторов, то цепочку можно разделить на ячейки произвольной длины. Возможность варьирования длиной БЯ ведет к более рациональному построению МАБИС и, следовательно, к повышению уровня их интеграции. Рисунок 11.2, *д* иллюстрирует расположение различных областей БМК

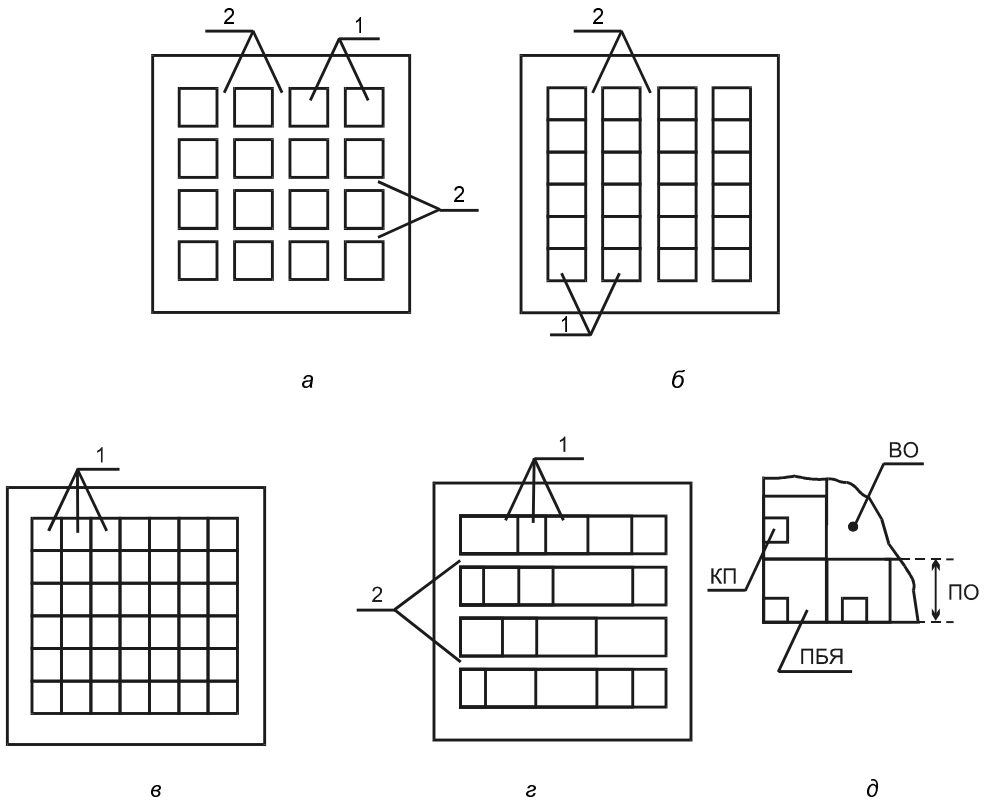


Рис. 11.2. Структуры БМК различных типов (*а*), (*б*), (*в*), (*г*) и расположение областей БМК (*д*)

Блочные БМК. Рост уровня интеграции ведет к усложнению МАБИС. Это вызвало к жизни *блочные структуры* БМК, архитектура которых упрощает построение комбинированных систем, содержащих как блоки логической обработки данных, так и память и другие специализированные блоки. В блочных БМК реализуются несколько блоков-подматриц, каждый из которых имеет как бы структуру БМК меньшей размерности. Между блоками располагаются трассировочные каналы (рис. 11.3). На периферии блоков изготовляются внутренние буферные каскады, формирующие достаточно мощные сигналы для передач по межблочным связям, имеющим большую длину.

Структурированные вентильные матрицы. Вследствие указанных ранее факторов МАБИС в свое время получили широкое распространение. В настоящее время возможности систем автоматизированного проектирования (САПР) настолько выросли, что трудности разработки методом стандартных ячеек уменьшились и зачастую стали более чем окупаться тем улучшением качества схем, которое дает переход от МАБИС к SCL. Поэтому метод стандартных ячеек стал вытеснять разработку микросхем на основе БМК. Реакцией на это явилось появление новых типов вентильных матриц — структурированных и "платформенных".

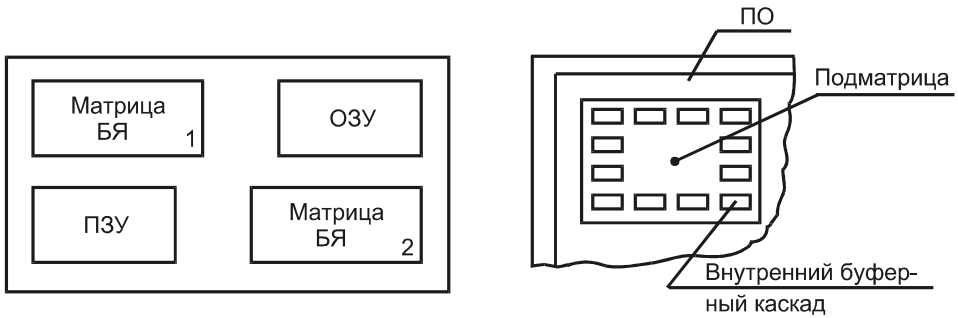


Рис. 11.3. Блочная структура БМК

Производители структурированных вентильных матриц стремятся совместить в них менее длительную и более дешевую разработку проектов, свойственную применению программируемой логики, с более экономичным расходом площади кристалла и более высоким быстродействием ASIC (специализированных БИС/СБИС).

Классические БМК имеют простые логические блоки, идентичные вентилям 2И-НЕ, или близкие к ним. "Мелкозернистость" блоков способствует полноте их использования в логических структурах и, как правило, повышает быстродействие схем. В то же время для полной трассировки мелкозернистых элементов требуется относительно много (до 5—6) заказных слоев металлизации и, следовательно, дорогостоящих фотошаблонов.

Логические блоки структурированных вентильных матриц более сложны ("крупнозернисты"). Это снижает эффективность использования логических ресурсов при реализации проектов, поскольку многие логические преобразования не требуют привлечения всей функциональной мощности блока. В то же время система коммутации для структур с крупнозернистыми блоками упрощается, и для ее создания могут оказаться достаточными 2 или даже 1 слой заказной металлизации.

Таким образом, структурированные вентильные матрицы *требуют для своей индивидуализации (Customisation) меньшее число заказных слоев металлизации.*

Несмотря на краткость своей истории, структурированные вентильные матрицы уже отличаются большим разнообразием. Можно выделить из их числа *базовые* (Basic) и *расширенные* (Advanced) реализации. Первый вариант подразумевает размещение на кристалле основного (типичного) набора ресурсов, в который входят, например, массив логических блоков, память, блоки PLL (DLL), стандартные каналы ввода/вывода. К таким базовым реализациям и относится непосредственно термин Structured ASICs.

Второй вариант подразумевает более полный набор функциональных блоков, в который включаются, например, процессорные ядра, блоки ЦОС, каналы SERDES и т. д. Более полный набор блоков обычно придает микросхеме проблемную ориентацию, т. е. направленность на решение задач определенного класса. Этот вариант относят к микросхемам *класса платформ* (Platform ASICs).

Разновидности БМК по типу данных. Тип обрабатываемых сигналов (цифровые, аналоговые) влияет на качество и состав схемных элементов базовых ячеек. В связи с этим БМК подразделяются на *цифровые, аналоговые и цифроаналоговые.* Аналоговые и цифроаналоговые БМК, появившиеся позднее цифровых и менее распространенные, имеют состав базовых ячеек, позволяющий получать на их основе такие схемы, как операционные усилители, аналоговые ключи, компараторы и т. д.

Классификация БМК по схемотехнологии. Приведенная на рис. 11.1 краткая классификация отражает только основные схемотехнические варианты БМК. Основное место занимает схемотехника КМОП, проявляющая свойственные ей известные достоинства. Сохраняются в некоторой мере БМК на основе ТТЛШ, существуют и быстродействующие БМК на основе схемотехники БиКМОП, кремний на диэлектрике и др.

Компонентный состав базовых ячеек

На рис. 11.4 показан *компонентный состав БЯ БМК* типа ЭСЛ, рассчитанный на реализацию двухъярусных логических элементов. Не рассматривая функциональные возможности схем, получаемых на основе таких БЯ, укажем только, что резисторы R0, входящие в состав источников тока для вышележащих переключателей, могут включаться параллельно или последовательно.

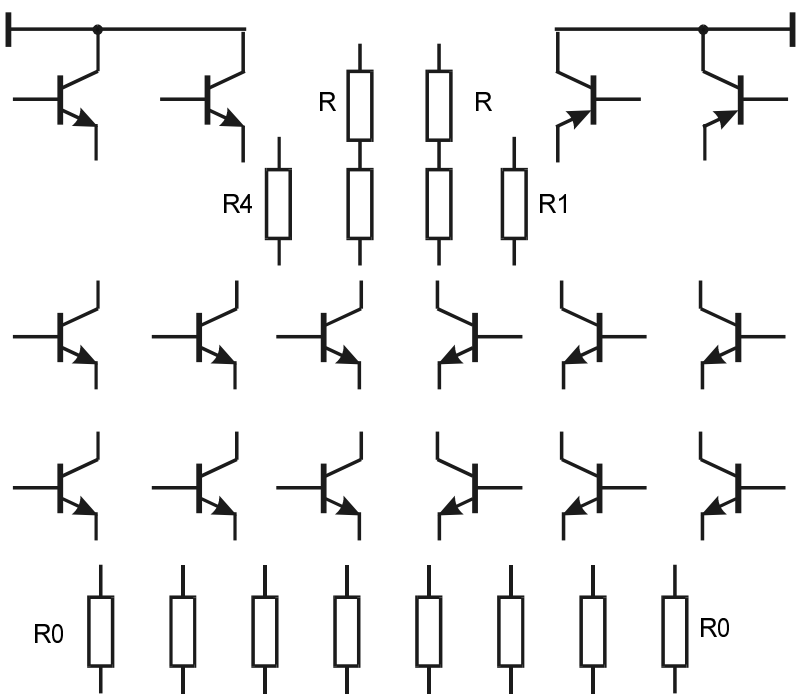


Рис. 11.4. Компонентный состав базовой ячейки БМК типа ЭСЛ

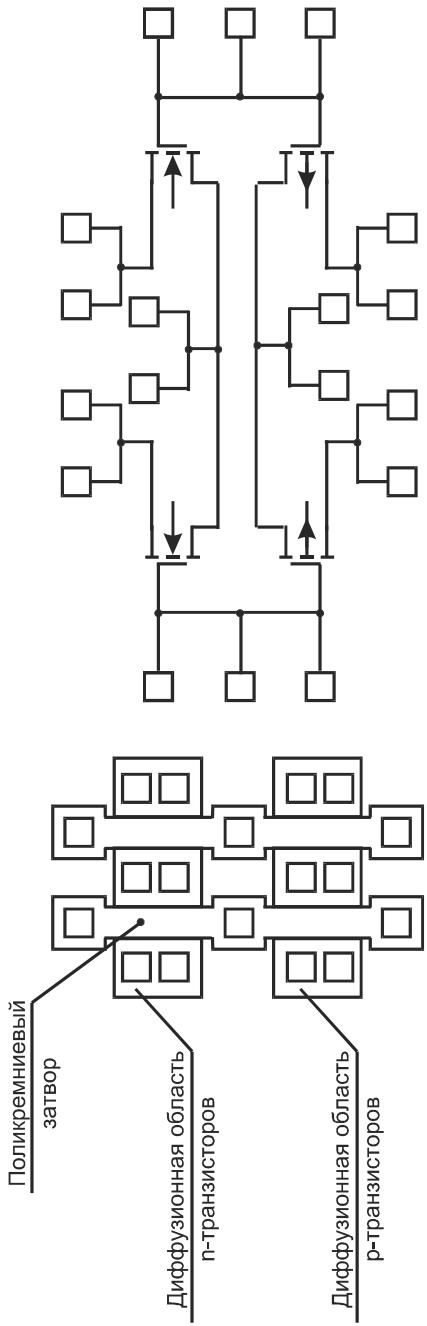


Рис. 11.5. Вариант базовой ячейки БМК типа КМОП

Это дает возможность получить несколько значений переключаемых токов, т. е. модификации схем, отличающиеся быстродействием и потребляемой мощностью.

На рис. 11.5 представлена БЯ в схемотехнике КМОП, компонентами которой служат только транзисторы с p- и n-каналами. Число транзисторов в ячейке выбирается по результатам анализа частоты использования различных логических элементов в устройствах заданного класса и преобладающих требований по нагрузочной способности, быстродействию и т. д. Высокий коэффициент использования транзисторов дают кристаллы с числом транзисторов в ячейке 4, 8 или 10. На рис. 11.5 показаны топология и электрическая схема ячейки с 4 транзисторами. Квадратные элементы топологического рисунка — контактные площадки к затворам и фиксированные контактные окна к элементам ячейки. Транзисторы можно соединять последовательно или параллельно, т. е. можно получать типовые подсхемы логических элементов И-НЕ и ИЛИ-НЕ. В схемотехнике КМОП транзисторы с противоположными по типу проводимости каналами всегда используются попарно, поэтому пары транзисторов имеют общий затвор.

Усложнение схем достигается объединением простых ячеек в группу.

Проектирование БМК. При проектировании БМК стремятся наилучшим образом сбалансировать число базовых ячеек, трассировочные ресурсы кристалла и число контактных площадок для подключения внешних выводов. Неудачные соотношения между указанными параметрами могут существенно ограничивать полноту использования ресурсов кристалла при построении МАБИС. Выбор параметров БМК основан на эмпирических формулах, выведенных на основании статистики при анализе множества проектов.

Основные понятия и определения

Базовая ячейка (БЯ) — набор схемных компонентов, регулярно повторяющийся во внутренней области БМК или по периметру кристалла. Этот набор может состоять из нескоммутированных, или частично скоммутированных компонентов. Базовые ячейки внутренней области БМК именуются *матричными базовыми ячейками* — (МБЯ), ячейки периферийной зоны — *периферийными базовыми ячейками* (ПБЯ). Применяются два способа организации ячеек БМК:

- из компонентов МБЯ может быть сформирован один логический элемент, а для реализации более сложных функций используются несколько ячеек;
- из компонентов МБЯ может быть сформирован любой функциональный узел, а состав компонентов ячейки определяется схемой самого сложного узла.

Функциональная ячейка (ФЯ — функционально законченная схема, реализуемая путем соединения компонентов в пределах одной или нескольких БЯ.

Библиотека функциональных ячеек — совокупность ФЯ, используемых при проектировании МАБИС. Эта библиотека создается при разработке БМК и избавляет проектировщика МАБИС от работы по созданию на кристалле тех или иных типовых подсхем, т. к. предоставляет для их реализации готовые решения. Библиотека содержит

большое число (сотни) функциональных элементов, узлов и их частей. Пользуясь библиотекой, проектировщик реализует схемы, работоспособность которых уже проверена, а параметры известны. Работая с библиотекой, он ведет проектирование на функционально-логическом уровне, поскольку проблемы схемотехнического уровня уже решены при создании библиотеки. Библиотечные элементы имеют различную сложность (логические элементы, триггеры, более сложные узлы или их фрагменты). В состав библиотечного элемента могут входить одна или несколько БЯ. Площадь библиотечного элемента кратна площади БЯ. При проектировании МАБИС функциональная схема изготовляемого устройства, как принято говорить, должна быть покрыта элементами библиотеки.

Эквивалентный вентиль (ЭВ) — группа элементов БМК, соответствующая возможности реализации логической функции вентиля (обычно это двухвходовой элемент И-НЕ либо ИЛИ-НЕ). Понятие "эквивалентный вентиль" предназначено для оценки логической сложности БМК.

Каналы трассировки — пути для возможного размещения межсоединений.

Параметры БМК

Параметры БМК можно разделить на 4 группы:

- параметры, характеризующие функциональные возможности БМК (число эквивалентных вентилей, тип БЯ, число МБЯ и ПБЯ, состав библиотеки функциональных ячеек и т. п.);
- электрические параметры (уровни напряжений, кодирующих логические сигналы, напряжения питания, потребляемые токи, задержки распространения сигналов, максимальные частоты переключений и т. п.);
- конструктивно-технологические параметры (тип корпуса, число выводов, число уровней металлизации, площадь кристалла и т. п.);
- эксплуатационные характеристики (устойчивость к воздействию внешних факторов, надежность и т. п.).

В табл. 11.1 приведены основные параметры отечественных БМК производства ОАО "Ангстрем" (топологическая норма проектирования 0,54 мкм).

Таблица 11.1

Схема БМК	Количество ячеек	Количество элементов библиотеки	Максимальная частота/задержка вентиля, МГц/нс
1806ХМ1	1500	125	6/8
1515ХМ1	3200	25	10/5
1593ХМ1	3200	70	40/1,5

Таблица 11.1 (окончание)

Схема БМК	Количество ячеек	Количество элементов библиотеки	Максимальная частота/задержка вентиля, МГц/нс
1593ХМ2	6400	70	40/1,5
1537ХМ1	4500	51	30/2,2
1537ХМ2	17600	51	30/2,2
1592ХМ1	100000	230	50/1

Компания UniqueICS (г. Зеленоград) освоила производство БМК по технологии с минимальным размером 0,25 мкм с числом ячеек до 200000, рабочей частотой 90 МГц и 116 контактными площадками для элементов ввода/вывода и питания на кристалле размерами $5{,}2 \times 5{,}2$ мм^2. Библиотека ядра содержит комбинационные логические элементы, элементы с тремя состояниями, триггеры, усилители тактовых импульсов, фрагменты сумматоров, мультиплексоры. В библиотеке ввода/вывода имеются элементы с разной выходной мощностью (диапазоном выходных токов от 4 до 16 мА. Две контактные площадки предназначены для подключения кварцевого резонатора. Напряжение питания 3,0…3,6 В, мощность потребления 1,5 Вт, корпус типа PLCC на 100 выводов, рабочий диапазон температур от –40 до 85 °С. При объеме заказа более 100 тыс. себестоимость микросхемы составляет 4…5 USD [2].

Параметры БМК фирмы IBM (2007 г.): технология с минимальным размером 65 нм, медная металлизация и другие технологические усовершенствования дали БМК с более чем 120 миллионами эквивалентных вентилей, задержками вентилей 6…9 пс и потреблением мощности менее 5 нВт/МГц/вент. Кристалл имеет встроенные блоки SRAM, DRAM и блоки процессоров.

Этапы проектирования МАБИС

Укрупненные этапы проектирования МАБИС показаны на рис. 11.6.

Исходное описание подлежащего реализации проекта может проводиться в разных вариантах. Ориентируясь на условия проектирования ОАО "Ангстрем", укажем следующие возможности представления проекта:

- в виде схемы, составленной из элементов библиотеки БМК;
- в виде описания на языках VHDL или Verilog (наиболее популярных языках из числа языков описания аппаратуры, называемых HDL, Hardware Description Languages);
- в виде схемы, составленной в элементном базисе микросхем программируемой логики, выпускаемых фирмами Xilinx, Altera, Actel;

- в виде электрической схемы, выполненной в любой библиотеке элементов;
- на основе технического задания.

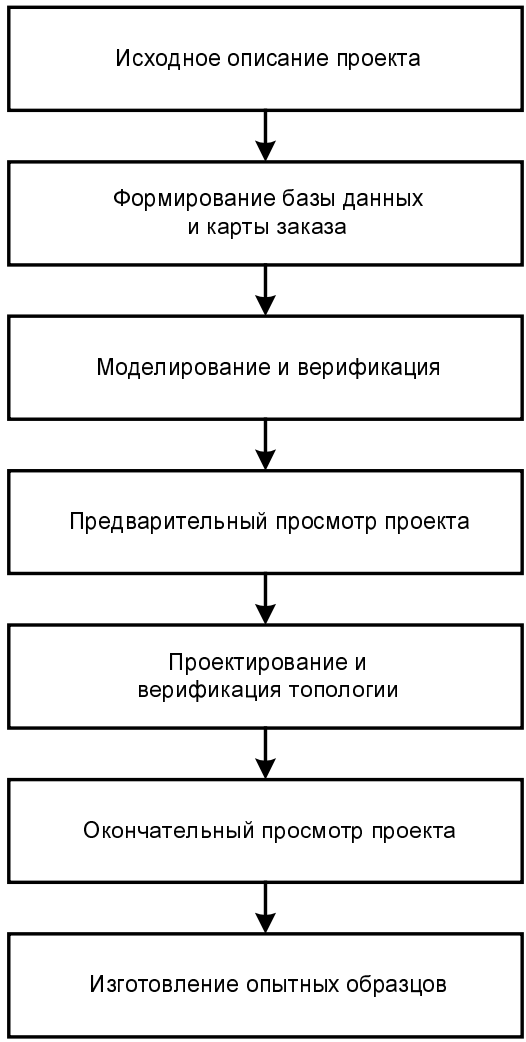


Рис. 11.6. Маршрут проектирования МАБИС

Библиотека стандартных элементов разрабатывается изготовителем БМК и является основой для проектирования МАБИС. Представление проекта в виде схемы, содержащей библиотечные элементы, выполненное заказчиком, упрощает задачи изготовителя. При текстовом описании проекта на языках типа HDL изготовитель, получив такое описание от заказчика, переводит его в описание на основе библиотечных элементов БМК. Точно так же, получив описание проекта в одном из перечисленных выше базисов микросхем программируемой логики или в виде электрической схемы в других базисах, изготовитель производит автоматизированный

перевод описания проекта в базис библиотеки БМК. Сформулировав техническое задание на проект, заказчик совместно с изготовителем могут разработать требуемую схему в библиотечном базисе БМК.

В решении задач, перечисленных для последующих этапов маршрута проектирования МАБИС, участвуют и изготовитель и заказчик. В зависимости от конкретных условий участие заказчика может быть более или менее активным, и он может взять на себя выполнение значительной доли общей работы. Как минимум требуется его участие в предварительном и окончательном просмотрах проекта.

Для разных по сложности БМК полный цикл проектирования МАБИС занимает от 1,5 до 3 месяцев.

§ 11.2. Структурированные вентильные матрицы

Структурированные вентильные матрицы СтрВМ (Structured Gate Arrays, *SGA*) — новый класс специализированных микросхем, который в последние годы успешно "вклинивается" между FPGA и традиционными БМК.

Классические БМК имеют мелкозернистые логические блоки, идентичные вентилям 2И-НЕ, или близкие к ним. Разработчики СтрВМ позаимствовали у FPGA крупнозернистые логические блоки и создали *вентильные матрицы, требующие для своей индивидуализации (Customisation) меньшее число заказных слоев металлизации*. Структурированные вентильные матрицы сейчас выпускаются приблизительно десятью фирмами.

Конвертация проектов

Промышленные изделия в своем жизненном цикле проходят несколько стадий:

- разработки изделия;
- активной реализации;
- спада производства (в связи с устареванием изделия и заменой его более новым и совершенным).

При выборе средств создания электронной аппаратуры (стандартных, программируемых, полузаказных или заказных ИС) учитываются особенности разных стадий жизненного цикла изделия.

На стадии разработки для *отработки прототипа* изделия чрезвычайно полезны микросхемы, репрограммируемые пользователем, как средства, позволяющие легко и быстро изменять проект, доводя его до полностью работоспособного устройства.

Останется ли вариант с репрограммируемыми микросхемами конечной продукцией, во многом зависит от объема его производства. Если изделие имеет высокий спрос, то не исключена целесообразность перевода его на реализацию в виде полузаказных или заказных схем, так как это позволит частично или полностью *устранить из устройства средства программирования структуры, что снизит стои-*

мость изготовления каждого экземпляра микросхемы и может повысить ее быстродействие. Затраты на проектирование некоторого числа фотошаблонов, необходимых для производства полузаказных или заказных схем, и другие однократные технологические затраты (NRE, Non Recurrent Expends) не зачеркнут указанный выигрыш, если тиражность изделия достаточно велика.

На последней стадии жизненного цикла изделия объем его производства снижается и становится неустойчивым. При этом может оказаться выгодным возврат от варианта с полузаказными и заказными микросхемами к варианту на программируемых структурах (FPGA, CPLD, SoPC), т. к. для них не существует проблемы заказа в виде только больших партий, и можно избавиться от риска экономических потерь при уценках и распродажах. Кроме того, легко быстро выпустить и дополнительную партию изделий любого объема, если конъюнктура этого потребует.

Между разными средствами реализации одних и тех же изделий желательна взаимосвязь, поскольку иначе процесс перехода от одних средств к другим (*конвертация проектов*) окажется затрудненным. Такая взаимосвязь выражается в сходстве архитектуры, электрических характеристик и конструктивных параметров микросхем программируемой логики и средств их замены. Вскоре после появления программируемой логики были созданы FPGA, которые подстраивались под определенные БМК с целью их эффективного макетирования, а определенные БМК приспосабливались по логической структуре, библиотеке схемных решений и электрическим параметрам к конкретным FPGA.

Применительно к проблеме конвертации в связи со сложностью проектов можно говорить о существовании *трех различных зон*. Проекты с небольшими логическими емкостями почти все остаются реализованными на микросхемах программируемой логики и в качестве конечной продукции. Далее идет область проектов со средней логической емкостью (приблизительно от 30 до 70 тыс. эквивалентных вентилей, хотя эти цифры отнюдь не бесспорны), для которой могут приниматься различные решения на основе конкретной ситуации. В третьей зоне (для проектов с большой логической емкостью) конвертация как правило целесообразна. Границы зон подвижны и смещаются в сторону увеличения. Конвертация основана на заранее созданной и проверенной базе данных, тогда как обычный перевод схемы из реализации на FPGA в реализацию на БМК или по методу стандартных ячеек нуждается в повторной верификации, что сопряжено с большими затратами времени.

Практические разработки

Структурированные вентильные матрицы выпускались и выпускаются несколькими крупными фирмами, в том числе Altera, Fujitsu, LSI Logic, AMIS, NEC, eASIC и др. Остановимся на наиболее известных разработках.

Микросхемы HardCopy. Фирма Altera для перевода проектов, реализованных на FPGA, в проекты полузаказного типа (ASIC) выпустила микросхемы HardCopy, поколения которых соответствуют поколениям серии Stratix (в частности, для мик-

росхем Stratix-2/3/4 имеются микросхемы HardCopy-2/3/4). Технология производства и архитектура у FPGA Stratix и HardCopy идентичны. Специализация кристаллов HardCopy достигается с помощью двух верхних слоев металлизации. Сообщается, что при этом экономится до 70% площади кристалла. Одновременно снижается стоимость микросхем и сохраняются типы их корпусов и их pin-совместимость (совместимость по внешним выводам). Переход от проекта, реализованного на FPGA, к эквивалентному проекту на микросхемах HardCopy занимает приблизительно 8 недель. С учетом времени, которое будет затрачено заказчиком на верификацию проекта, полное время до выпуска продукции оценивается приблизительно в 16 недель. Перевод отработанного на кристалле Stratix проекта в версию HardCopy снижает стоимость кристалла на 30…50%, почти удваивает его быстродействие, снижает потребляемую мощность на 40%.

Программное обеспечение проектирования на основе HardCopy Stratix допускает *прямую компиляцию проекта*, т. е. его реализацию без промежуточной разработки на FPGA Stratix. Для подобных ситуаций расходы NRE (бесповторные расходы на разработку) оцениваются величинами порядка 200 тыс. USD (значительно колеблются в зависимости от конкретных условий). Время появления первых образцов около 8 недель, промышленной продукции — около 18 недель. До освоения промышленного выпуска можно использовать реализацию проекта на FPGA семейства Stratix.

Концепция EasyPath. Конвертации проектов свою долю внимания отдала и фирма Xilinx. В течение некоторого времени она развивала линию микросхем, ориентированных на обычный вариант конвертации (HardWire), но затем предпочла путь, названный Easy Path. Этот путь нашел применение для проектов, выполненных на микросхемах серий Spartan и Virtex.

Кристаллы, предназначенные к реализации в рамках концепции Ease Path, остаются прежними. Изменяется методика их тестирования в сторону удешевления, увеличивается процент выхода годных. Фирма ведет работу с заказчиками, имеющими законченный проект. Программное обеспечение Easy Path имеет анализатор конфигурации кристалла и согласует требования тестирования с условиями работы логических ячеек в данном проекте, значительно уменьшая время и стоимость тестирования. При обычном тестировании какие-то кристаллы попадают в брак даже из-за 1—2 неисправных ячеек. В рамках тестирования Easy Path это может быть допустимым, если от неисправных ячеек не требуется выполнение соответствующих функций. Уменьшение числа забракованных кристаллов увеличивает выход годных и снижает их стоимость. Отмечается существенная экономическая эффективность концепции Easy Path. Время перехода от завершенного проекта к производству изделия оценивается приблизительно в 12 недель.

Микросхемы фирмы eASIC. С 2007 г. на рынке появились структурированные вентильные матрицы семейства Nextreme фирмы eASIC (технология 90 нм), а затем и микросхемы семейства Nextreme-2 (технология 45 нм). Эти серии представляют собою еще один шаг в направлении развития от БМК к FPGA. Структура и технология производства микросхем Nextreme являются инновационными.

Как известно, ИС — многослойные конструкции. В микросхемах Nextreme три нижних слоя, в которых реализуются логические структуры, конфигурируются как FPGA, т. е. загрузкой битового потока.

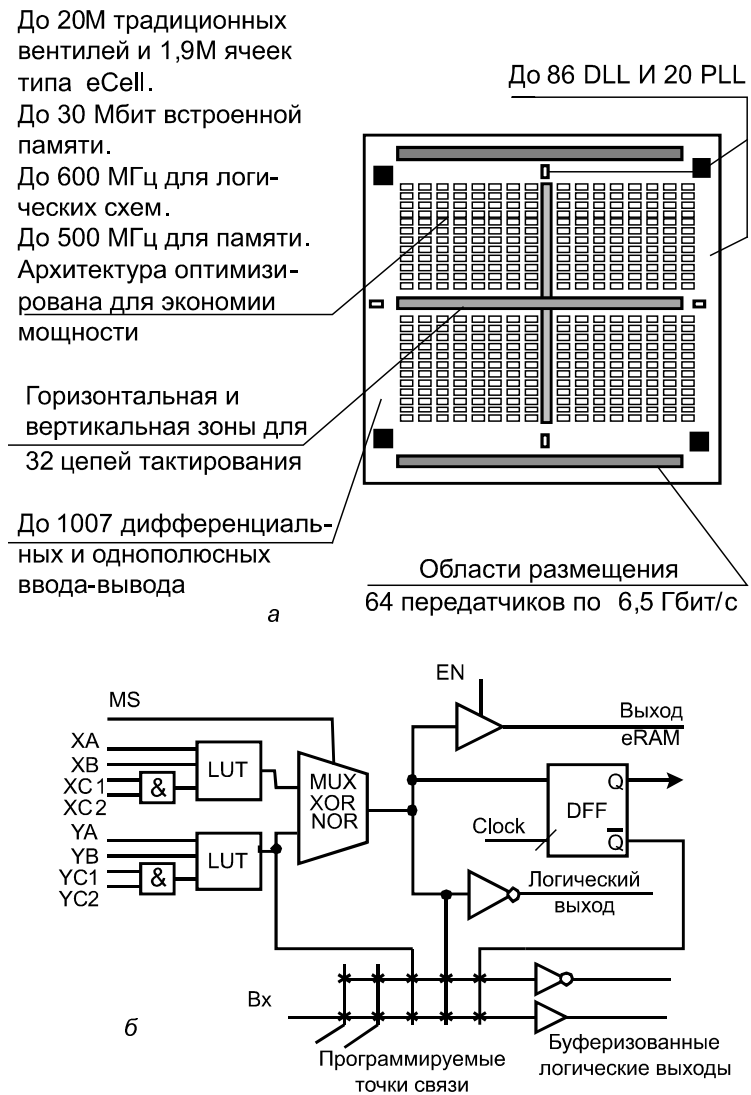


Рис. 11.7. Размещение ресурсов на кристалле микросхемы Nextreme-2 (*а*) и схема ячейки eCell (*б*)

В них расположены традиционные для FPGA крупнозернистые логические блоки (е-ячейки, eCells), построенные на основе связки "LUT-блок + триггер". Следующие четыре слоя конфигурируются с помощью металлических сегментов, межсоединения которых реализуются перемычками Via без помощи шаблонов по луче-

вой технологии Direct Write eBeam. И только один верхний слой специализируется шаблоном. Достигается высокая плотность трассировки при применении всего одного шаблона и гибкость функциональных структур, свойственная конфигурированию битовым потоком.

В рамках описанной технологии получены структурированные матрицы с улучшенными характеристиками (высокая плотность, высокая тактовая частота, малая потребляемая мощность, время подготовки к производству 4…6 недель и т. д.). Кроме того, лучевая технология изготовления межсоединений позволяет на одной пластине получать кристаллы с реализацией разных проектов (это снимает ограничения по минимальному объему заказа).

Семейство Nextreme-2 насчитывает шесть представителей. Параметры младшего и старшего представителей семейства приведены в табл. 11.2, в которой приняты обозначения: $N_{ЭВ}$ — число эквивалентных вентилей; $N_{В/В}$ — число вводов-выводов; bRAM — блочная RAM; Via ROM — постоянная память, программируемая перемычками.

Таблица 11.2

Микросхема	$N_{ЭВ}$ (млн)	е-ячеек	Via ROM (Кбит)	BRAM (блоков)	BRAM (Кбит)	Регистровых файлов (Кбит)	PLL/DLL	$N_{В/В}$
N2X260	2,6	258 048	1,024	112	4 032	112	16/28	521
N2X1900	19,2	1 916 928	1,024	832	29 952	832	20/86	1 008

На рис. 11.7 показаны размещение ресурсов микросхем Nextreme на кристалле (*а*) и схема е-ячейки (*б*).

Микросхемы фирмы Lattice Semiconductor. В рамках методологии Freedom Chip осуществляется конвертация проектов, выполненных на FPGA фирмы Lattice Semiconductor, со снижением стоимости микросхем на 30…75%. Микросхемы Lattice SC/M (2007 г.) имеют до 25K LUT-блоков в корпусе с 1020 выводами (корпус типа flip-chip BGA) и 115K LUT-блоков в корпусе с 1704 выводами. При окончательной отработке проекта и его готовности к массовому производству можно заказывать его конвертацию в вариант Freedom Chip с минимальным объемом заказа 1200…3600 шт. при поставке в течение 12 недель и NRE-расходах порядка 75 000 USD, что на порядки дешевле разработки заказных вариантов.

Контрольные вопросы

1. Поясните смысл термина "полузаказные БИС/СБИС". Каким другим вариантам БИС/СБИС противопоставляется этот термин?

2. Что представляет собой базовый матричный кристалл БМК (вентильная матрица ВМ)?
3. Укажите основные достоинства и недостатки полузаказных БИС/СБИС в сравнении с другими их вариантами.
4. Перечислите разновидности БМК по архитектурным признакам и по схемотехнике.
5. Поясните термины "базовая ячейка", "функциональная ячейка", "периферийная ячейка".
6. Каким образом оценивается логическая сложность традиционных вентильных матриц? Какова максимальная логическая сложность современных БМК?
7. Перечислите варианты описания проекта, предназначенного для реализации в виде МАБИС средствами САПР.
8. Какой процесс называют "конвертацией проектов"? В каких условиях целесообразен этот процесс?
9. Чем структурированные вентильные матрицы отличаются от традиционных? Какие преимущества создают эти отличия? Какие варианты структурированных вентильных матриц называют платформенными?
10. Какова сравнительная степень привязки микросхем HardCopy и Nextreme к применению для конвертации проектов, выполненных на конкретных прототипных FPGA?
11. Благодаря технологии электронно-лучевого программирования межсоединений удается на одной кремниевой пластине (современные пластины имеют диаметр 300 мм) получать микросхемы, в которых реализованы разные проекты. На какие технико-экономические показатели производства структурированных вентильных матриц влияет такая возможность?

Литература к главе: [2], [51], [56], [58], [III], [X], [XV], [XVII].

Глава 12

Методика и средства автоматизированного проектирования цифровых устройств

В предыдущих главах вопросы проектирования освещались применительно к частным проблемам создания цифровых устройств. Цель этой главы — дать целостную картину процесса проектирования ЦУ и рассмотреть его этапы, имея в виду применение систем автоматизации проектирования (САПР).

§ 12.1. Общее описание процесса проектирования

Проектирование — разработка технической документации, позволяющей изготовить устройство с заданными функционированием и свойствами в заданных условиях.

Сложность современных систем заставляет рассматривать их с различных позиций и с различной степенью детализации. Устройство (и/или его модели) могут быть описаны в формах:

- определения выходных откликов на входные воздействия — это *функциональное описание*;
- в терминах соединения более примитивных элементов — это *структурное описание*.

Как правило, используется смесь обоих способов описания в иерархически или смешанно организованной форме.

Физическая реализация проектируемого устройства находит отражение в *конструкторско-топологическом представлении*.

Взгляду на проектируемую систему может условно соответствовать последовательность концентрических вложенных друг в друга окружностей (рис. 12.1, *а*).

В зависимости от условий и целей рассмотрения практически любую систему можно разделять на аспекты (функциональный, алгоритмический, структурный, конструкторский, технологический), на иерархические уровни детализации внутри каждого аспекта, на отдельные стадии (научно-исследовательская, опытно-конструкторская, технического и рабочего проектирования, испытаний опытного образца и т. д.) и на этапы [1].

Процесс проектирования можно рассматривать как создание последовательности моделей, проведение экспериментов над ними и анализ полученных результатов. Модели могут быть концептуальными или имитационными, в некоторых случаях, даже физическими прототипами и отражать нужные свойства всей будущей системы или ее отдельных фрагментов. В соответствии с видом модели и эксперименты будут мысленными, машинными или физическими. В процессе анализа проверяется не только правильность работы, но и некоторые показатели, характеризующие будущее устройство. По результатам анализа производится корректировка модели или (если результаты очередного этапа разработки удовлетворяют исходным требованиям) переход к разработке следующих блоков или к следующему уровню иерархии проектирования.

Под *этапами проектирования* понимаются его условно выделенные части, сводящиеся к выполнению одной или нескольких проектных процедур, объединенных по признаку принадлежности к одному иерархическому уровню и (или) аспекту рассмотрения. Все *этапы проектирования многовариантны*.

Иллюстрацией взаимосвязи уровней детализации проекта с аспектами процесса проектирования может служить обобщенная и модифицированная диаграмма Гайского—Кана [59], [61], приведенная на рис. 12.1, б. Процесс проектирования отображается движением по спирали, начинающимся от функциональной спецификации и завершающимся заданием топологии отдельных технологических слоев обработки кремниевой подложки (полигонов). В определенных случаях подобное движение может и начинаться и заканчиваться на внутренних уровнях. Например, разработка СБИС или печатной платы со стандартными ИС начинаются и заканчиваются на разных уровнях иерархии. Из рисунка видно, что проектирование современной системы управления, содержащей, например, компьютер, исполнительные двигатели и антенную решетку, может последовательно проходить по всем уровням детализации. А проект, заключающийся в создании одиночной ИС, может начинаться с блочно-функционального представления.

Применяются методологии *нисходящего и восходящего проектирования* ("сверху-вниз" и "снизу-вверх"). Первая предусматривает переход от технического задания до электрических схем, файлов прошивки ПЗУ и конфигурирования программируемых приборов, а также конструкции устройства в целом. Вторая предусматривает объединение простейших модулей в более сложную структуру до тех пор, пока, в конце концов, не будет достигнут желаемый результат. *Исходные модули* — это решения, созданные проектировщиком на более ранних этапах работы или в ходе работ над другими проектами, а также модули, доступные проектировщику и входящие в состав имеющихся библиотек систем автоматизированного проектирования — САПР.

Современным условиям, при которых сложные проекты выполняются с привлечением большого числа разработчиков, преимущественно соответствует *смешанная стратегия*, объединяющая достоинства обоих подходов. Для такой смешанной стратегии используется название спиралевидная [32].

Сущность процесса нисходящего проектирования удачно выражена в работе [18] следующим образом.

Стратегия проектирования — функциональная декомпозиция. Для системы в целом и ее блоков используется концепция "черного ящика". Для "черного ящика" разрабатывается функциональная спецификация, включающая внешнее описание блока (входы и выходы) и внутреннее описание — функцию или алгоритм работы: $F = \Phi(X, t)$, где X — вектор входных величин, F — вектор выходных величин, t — время. При декомпозиции функция Ф разбивается на более простые функции $\Phi_1 ... \Phi_K$, между которыми должны быть установлены определенные связи, соответствующие принятому алгоритму реализации функции Ф. В результате разбиения, в конечном счете, получается структура. *Переход от функции к структуре — синтез.* Функциональная спецификация на разрабатываемое устройство может быть задана различными способами. Для конкретизации свойств проектируемой системы может привлекаться тот или иной формальный аппарат. Синтез также неоднозначен. Проект можно реализовать разными способами (в том числе и из компонентов различных производителей).

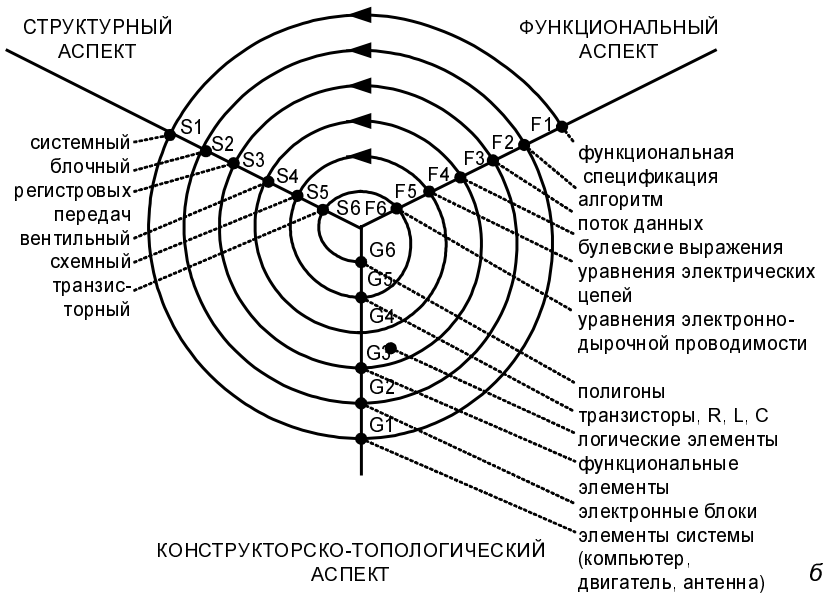
Детализация функций фрагментов (декомпозиция) выполняется до тех пор, пока не получатся блоки, каждый из которых может быть реализован элементами выбранного уровня иерархии.

Процесс проектирования — многоуровневый, многошаговый и итерационный, с возвратами назад и пересмотром принятых ранее решений.

Последовательная декомпозиция проекта на отдельные фрагменты (с определением функций каждого фрагмента и его интерфейса) присуща всем иерархическим уровням и характерна для разработки широкого класса цифровых устройств, начиная от проектирования отдельных БИС/СБИС и кончая устройствами, содержащими в своем составе большое число микросхем.

В соответствии с [1] *декомпозиция заканчивается при получении типовых решений, соответствующих различным уровням иерархии.* Следует отметить, что независимо от используемой техники, технологий, компонентов и т. п. целевая направленность и последовательность работ, выполняемых на любом уровне иерархии, практически совпадают (рис. 12.2). При этом учитываются особенности элементной базы и взаимоотношения разработчика и изготовителя с точки зрения конструкторско-топологического аспекта. Так, на верхнем уровне (при многоплатной реализации) декомпозиция заканчивается при представлении проекта в виде отдельных плат, на следующем уровне в виде отдельной платы (типового элемента замены), еще ниже декомпозиция осуществляется до реализации функций с помощью той или иной стандартной микросхемы. При использовании программируемых или разрабатываемых пользователем микросхем декомпозиция завершается на уровне детализации, определяемом составом технологических библиотек изготовителя или используемой САПР.

а



б

Рис. 12.1. Многоаспектное и многоуровневое отображение проектируемой системы (*а*) и процесса проектирования (*б*)

Техническая сторона проектирования определяется тремя основными составляющими:

- "строительными кирпичиками проекта" (используемой компонентной базой);
- инструментарием проектировщика (при ориентации на разработку аппаратно-программных систем — это САПР);
- методикой использования инструмента и "кирпичиков" (обычно для этого используется термин *Design Flow — маршрут проектирования*).

Рис. 12.2. Этапы в типовой процедуре проектирования

Три указанных составляющих настолько тесно связаны между собой, что конкретизация одной из них сразу резко сокращает возможные варианты выбора других. Практически невозможно говорить об одной составляющей, не упоминая об остальных. Именно поэтому в параграфах этой главы рассматривается один и тот же маршрут проектирования, но упор в каждом из них делается только на одну составляющую. Предыдущие главы дали достаточно информации о различных "строительных кирпичиках" современной цифровой схемотехники. В данной главе основное внимание уделено двум оставшимся составляющим процесса проектирования.

Для определенности далее (кроме специально оговоренных случаев) будем считать, что объектом проектирования является печатная плата, содержащая набор ИС различной степени сложности. Такой объект сохраняет общность в широком диапазоне исходных требований к проекту, поскольку необходимость разработки конструкции печатной подложки возникает даже в случае реализации всех функций проекта в форме одиночной ИС.

Один из основных факторов, влияющих на специфику проектирования, — *тип обрабатываемой информации* и связанные с ним способы ее обработки. Проект или его фрагменты могут включать аналоговые, аналого-цифровые и (или) цифроаналоговые элементы, могут строиться на основе дискретных компонентов или микропроцессорных технологий. Отсюда возникает многообразие вариантов проектирования (проектирование для чисто цифровых, смешанных аналого-цифровых, смешанных аппаратных и программных объектов, проектирование с ориентацией на синтезируемые цифровые или аналоговые схемы и др.).

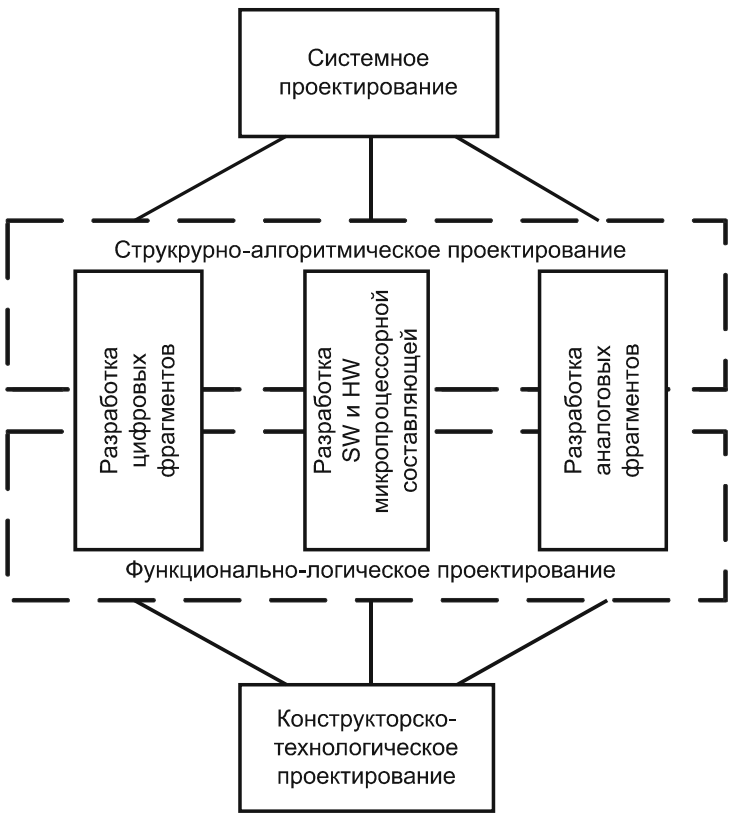


Рис. 12.3. Разбиение типовой процедуры проектирования

При традиционном разбиении процедуры на этапы системного, структурно-алгоритмического, функционально-логического и конструкторско-технологического проектирования *наиболее существенным представляется этап системного проектирования.* На этом этапе, исходя из требуемого функционирования устройства, проектировщик осуществляет разбиение проекта на отдельные фрагменты, определяет множества входных и выходных сигналов (как устройства в целом, так и его составных частей), их характер и взаимосвязь, а также выбирает способы реализации фрагментов.

Различие теоретической базы, понятийного аппарата и технических средств, привлекаемых на разных стадиях проектирования аппаратно-программных систем, приводит к тому, что процесс их проектирования необходимо разделять как по иерархическому принципу детализации, так и по функциональным отличиям в параллельных ветвях. Подобное разбиение процедуры проектирования после выполнения системного этапа показано на рис. 12.3. Составление маршрутов проектирования необходимо не только для проекта целиком, но и для его отдельных составляющих.

Порядок работы по параллельным ветвям процедуры проектирования произволен и может во времени выполняться как параллельно или последовательно, так и в произвольных комбинациях. Более того, даже этап конструкторско-технологического проектирования при ориентации на современные СБИС может начинаться раньше окончательного завершения работ над отдельными фрагментами проекта.

Общая методология процесса проектирования не зависит от варианта его разбиения на отдельные уровни и ветви, но содержание, методы и средства проектирования для этих уровней и ветвей специфичны и существенно зависят как от типа применяемой компонентной базы, так и от способа изготовления конечного продукта.

Прежде всего, рассмотрим влияние на процедуру проектирования выбора технологической реализации компонентов проекта.

§ 12.2. О выборе альтернативных средств реализации проекта

Выбор технической базы и технологического способа реализации проекта — одна из важнейших проблем, стоящих перед разработчиком. Как правило, одно и то же электронное изделие может быть реализовано различными способами. При выборе должен быть дан ответ на вопрос — будет ли проект построен на стандартных микросхемах или будут использоваться те или иные специализированные ИС или комбинация различных типов ИС.

Задача выбора средств реализации проекта всегда имеет множество решений. В последнее время проблемы выбора средств реализации проектов усложнились в связи с такими свойствами современной компонентной базы электроники, как наличие и степень программируемости кристаллов, уровень их интеграции и т. д., что предоставляет проектировщику широкие возможности варьирования решений.

Напомним вкратце состав микросхем, которыми может воспользоваться проектировщик устройств и систем (подробно эти возможности рассмотрены в предыдущих главах).

Стандартные микросхемы разных уровней интеграции (МИС, СИС, БИС, СБИС) производятся массовыми тиражами в различных схемных и технологических вариантах и реализуют элементы и устройства, функционирование которых предопре-

делено и не зависит от места их применения. К этой группе микросхем принадлежат логические элементы вентильного уровня, буферные элементы, триггеры и регистровые схемы, дешифраторы, мультиплексоры и т. п. К стандартным схемам высокого уровня интеграции относятся микросхемы памяти (ЗУ), интерфейсные схемы микропроцессорных систем и аналого-цифровые схемы: цифроаналоговые преобразователи (ЦАП), аналого-цифровые преобразователи (АЦП).

Микропроцессоры и микроконтроллеры являются *стандартными для производителя*, но способны изменять свое функционирование соответственно нуждам разработчика проекта. Они настраиваются на решение определенной задачи методом *изменения программы*.

Микросхемы ПЛИС и ПАИС с программируемой структурой также *стандартны для производителя* и приспосабливаются к решаемым задачам путем *изменения структуры,* выполняемого потребителем.

Структурированные и обычные *вентильные матрицы* являются полуфабрикатами, которые используются проектировщиками для заказа на их основе нужных им схем. При этом в разработке проекта участвует и изготовитель, услуги которого достаточно дороги, но зато позволяют достичь более высоких технических параметров проекта в сравнении с ПЛИС и ПАИС, содержащими избыточные схемные ресурсы для программирования структур. Вентильные матрицы относятся к классу *полузаказных* схем.

Заказные микросхемы разделяются на *полностью заказные* и схемы *на основе стандартных ячеек*. Полностью заказные схемы целиком проектируются по требованиям конкретного заказчика. Такие схемы очень дороги и имеют длительные циклы проектирования. Таким способом имеет смысл проектировать микросхемы, которые будут иметь заведомо широкий спрос, и это оправдает высокие затраты на их разработку.

Схемы *на стандартных ячейках* отличаются от полностью заказных тем, что их фрагменты берутся из заранее разработанной библиотеки схемных решений, что облегчает и удешевляет проектирование. Наивысших технических параметров добиваются от полностью заказных схем, однако метод стандартных ячеек более популярен, т. к. при небольших потерях в технических характеристиках с его помощью можно заметно упростить проектирование схемы.

Повышение уровня интеграции микросхем улучшает целый комплекс показателей их качества. Оно же порождает эволюцию средств реализации проектов и методов проектирования, причем сейчас сосуществуют как традиционные подходы к этим задачам, так и появившиеся в последнее время.

Традиционная реализация проектов

Такая реализация проектов подразумевает разработку устройств и систем в виде набора микросхем, размещенных на печатных платах. Для построения системы (имеется в виду самый популярный объект разработок — микропроцессорная

система) потребуются стандартные компоненты: процессор, память, периферийные ИС, интерфейсные схемы, а также некоторое количество логики малого или среднего уровней интеграции. Проблема выбора средств реализации проекта выступает здесь как проблема выбора конкретных типов стандартных микросхем из спектра имеющихся возможностей с учетом требований проекта и совместной работы применяемых блоков. Традиционную разработку можно начать без существенных предварительных затрат при незначительных рисках проектирования. Разработка облегчается наличием большого числа инженеров, обладающих опытом соответствующей деятельности. Для некоторых систем получаемый результат можно считать хорошим решением. Однако при современных возможностях микроэлектроники это уже в значительной мере пройденный этап. Во-первых, традиционное проектирование сопровождается построением устройств и систем из относительно большого числа корпусов микросхем, что неблагоприятно отражается на ряде параметров проекта. Во-вторых, системы традиционного типа не обеспечивают той функциональной гибкости и легкости модификаций, в том числе многократных, которыми обладают реализации на основе микросхем с программируемой структурой.

Реализация проектов на кристаллах с программируемыми структурами

С ростом уровня интеграции микросхем появляется возможность перейти к реализации проекта на меньшем числе кристаллов или даже на одном кристалле, снизив число компонентов системы и габариты конечного продукта, удешевив сборку, повысив надежность и улучшив другие технико-экономические показатели изделия.

В работе [32] отмечено: "Ранее созданные разработчиками систем и аппаратуры функционально-законченные узлы, блоки, отдельные субсистемы, конструктивно реализованные в виде печатных плат, с десятками типов универсальных и специализированных микросхем и отдельных компонентов, успешно трансформируются в качественно новую реализацию — *сверхбольшие интегральные схемы типа "система на кристалле"*. Эти революционные изменения (сравнимые по значимости с созданием транзистора и первых монолитных интегральных микросхем) привели к качественно новой методологии и инфраструктуре проектирования и производства СБИС SoC и систем на их основе".

Там же сказано: "При технологии уровня 350—250 нм стало возможным осуществить интеграцию всех основных цифровых компонентов конечного продукта на одной кремниевой подложке… Такие "системы в кремнии" уже четко *определили направление эволюции радиоэлектронной аппаратуры* с развитием полупроводниковой технологии".

Остановимся подробнее на вопросах рационального выбора *конкретного варианта программируемых схем* для разработки проектов, имеющих характер *окончательной продукции* (область применения программируемых пользователем

схем *в качестве прототипов* при разработке и отладке проектов весьма широка и неоспорима).

При выборе типа микросхем для реализации проекта в первую очередь учитываются следующие факторы:

- сложность проекта;
- требуемое быстродействие (производительность);
- потребляемая мощность и габаритные показатели;
- время подготовки производства изделия;
- стоимость производимой продукции.

Выбор средств реализации проекта по критериям сложности, быстродействия и потребляемой мощности выделяет из области возможностей некоторую зону, внутри которой выбор определяется *экономическими* соображениями — стремлением получить требуемый результат с наименьшими затратами средств и времени. На этом этапе большое влияние на принимаемые решения оказывает объем производства разрабатываемого изделия — его *тиражность*.

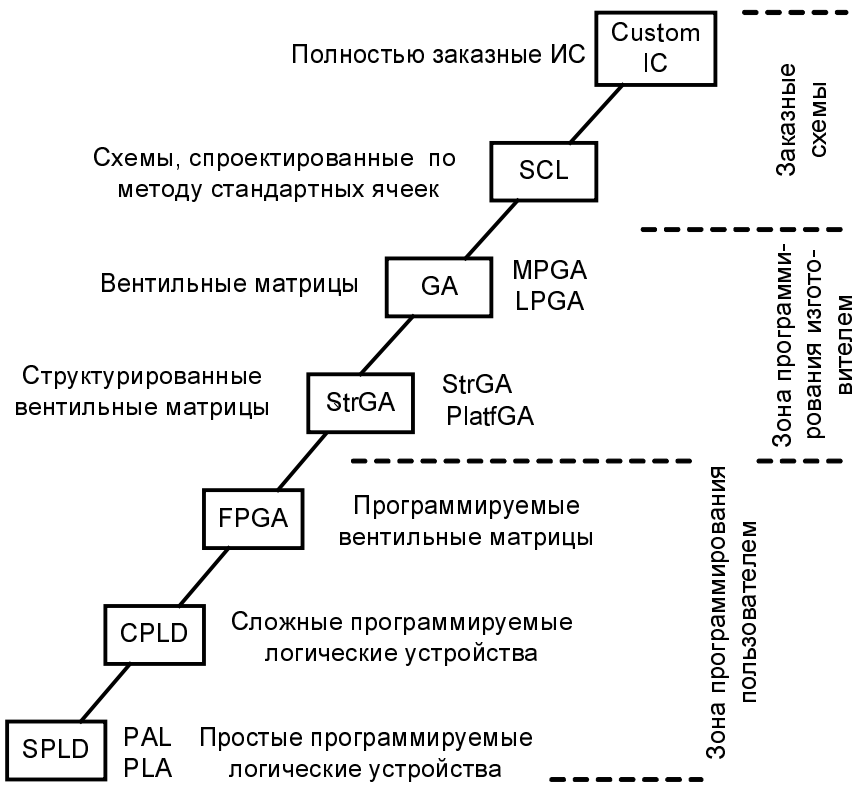


Рис. 12.4. Средства реализации проектов различной сложности и тиражности

На рис. 12.4 показаны средства (микросхемы с программированием структур как потребителем, так и с участием изготовителя), применение которых целесообразно для реализации проектов различной сложности и тиражности.

Стоимость микросхемы $C_{МС}$, изготовленной уже освоенным технологическим процессом, определяется выражением:

$$C_{МС} = C_{ИЗГ} + C_{ПР}/N,$$

где $C_{ИЗГ}$ (Unit cost) — стоимость изготовления микросхемы (стоимость кристалла и других материалов, стоимость технологических операций и контрольных испытаний), $C_{ПР}$ — стоимость проектирования микросхемы и N — объем выпускаемой партии микросхем. Затраты на изготовление относятся к каждой микросхеме и повторяются столько раз, сколько микросхем будет произведено. Затраты на проектирование однократны и раскладываются на весь объем производимой продукции.

Стоимость всей партии микросхем $C_{ПАР} = NC_{МС}$ и определяется соотношением:

$$C_{ПАР} = NC_{ИЗГ} + C_{ПР},$$

график которого имеет вид рис. 12.5, *а*.

Целесообразность применения микросхем того или иного типа *в составе окончательной продукции* определяется сложностью и тиражностью проектов.

Использование SPLD и CPLD мотивируется достаточно ясными соображениями, при их умеренных уровнях интеграции большого разнообразия конкурирующих решений, как правило, не возникает. Программируемые схемы SPLD и CPLD активно отвоевывают области применения у схем на стандартных микросхемах невысокого уровня интеграции.

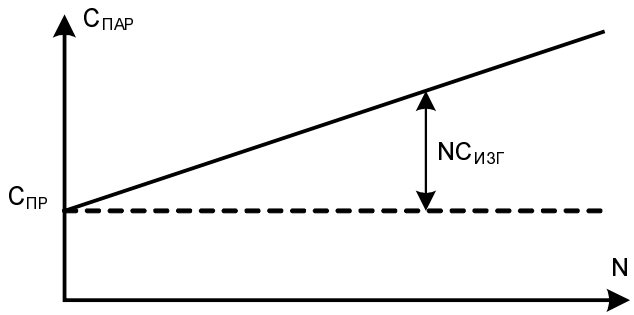
Для смежных областей "*FPGA — структурированные БМК — обычные БМК — схемы на стандартных ячейках*" (*FPGA — StrGA — GA — SCL*) взаимная конкуренция выражена достаточно сильно. Разработки проектов на базе перечисленных средств отличаются значениями $C_{ПР}$ и $C_{ИЗГ}$. При применении FPGA отсутствуют затраты на проектирование кристалла, но затраты на изготовление каждого кристалла максимальны, т. к. средства конфигурирования схемы существенно увеличивают площадь кристалла. Для всех других средств затраты на проектирование (бесповторные затраты NRE — Nonrecurring Engineering) значительны, поскольку для них требуется проектирование шаблонов (масок), с помощью которых специализируется схема. Для структурированных вентильных матриц (в том числе и платформенного типа) число проектируемых шаблонов минимально, и в некоторых случаях снижено до одного. Для обычных вентильных матриц число масок больше, но затраты на реализацию самой схемы меньше. Схемы на основе стандартных ячеек требуют разработки полного набора шаблонов, но имеют еще более эффективную реализацию схемы как по площади кристалла, так и по быстродействию.

Зависимость стоимости производства партии изделий от ее размеров для вариантов реализации проекта на основе FPGA, структурированных и обычных вентильных матриц показана на рис. 12.5, *б*. Такие зависимости конкретизируются с учетом

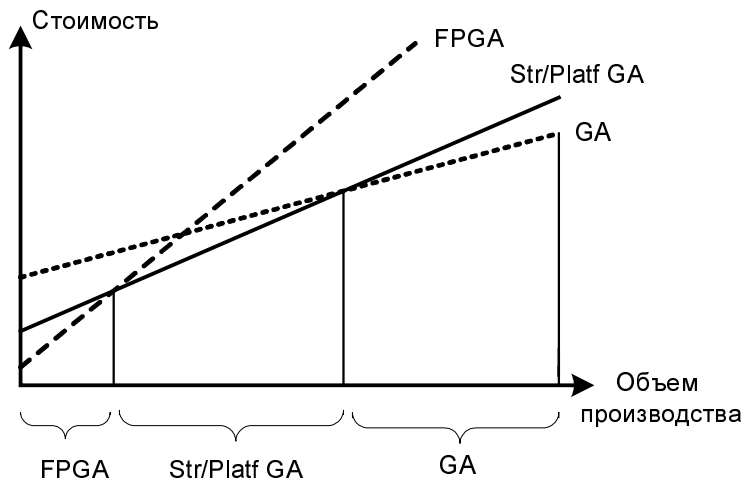
сложности проекта, для которого анализируются варианты реализации. В работе [56] приведены, без указания сложности проекта, следующие цифры: точка пересечения графиков для FPGA и структурированных GA соответствует объему производства 2000 экз., а точка пересечения графиков для структурированных и обычных вентильных матриц соответствует объему производства 150 000 экз.

Другим примером может служить оценка фирмой Altera диапазона тиражности проекта, позволяющего снизить стоимость микросхем высокой сложности при замене FPGA на структурированные вентильные матрицы HardCopy как 10…50 тыс. шт.

По мнению некоторых компаний вскоре станет экономически выгодной реализация проектов на FPGA (в том числе в виде СнПК) при тиражности проекта менее 300 тыс. шт.



а



б

Рис. 12.5. Зависимость стоимости партии микросхем от объема партии

Достоинства ИСПС (интегральных схем с перестраиваемой структурой) рождают спрос на них, спрос стимулирует предложение, в результате появилось много их разновидностей, в которых непросто ориентироваться без понимания особенностей различных архитектур. Знание архитектур и особенностей существующих ИСПС необходимо для успешного проектирования устройств и систем, в частности, этапа выбора элементной базы и оценки, а при необходимости, и "ручной" коррекции результатов работы компилятора САПР по синтезу схемной части. При выборе подходящей ИСПС для своего проекта системотехник не может полагаться на какой-либо параметр микросхем или даже 2—3 параметра. Проблема выбора достаточно сложна и требует проведения квалифицированного анализа совокупности параметров микросхемы с учетом специфики подлежащего реализации проекта.

Место программируемой логики в процессе создания современной аппаратуры

Проектирование заказными методами — удел крупных специализированных фирм. На долю системотехников приходятся главным образом другие разработки: цифровых устройств малой сложности на МИС и СИС, микропроцессорных систем для целей управления техническими объектами и технологическими процессами, малотиражной аппаратуры либо прототипов систем на основе ИС программируемой логики.

Проектирование на основе МИС, СИС — наиболее традиционный процесс, в котором используются как эвристические подходы, так и формализованные методики. Проектировщик задает структуру устройства на базе своих знаний, идей и освоения опыта предшественников, а при определении функций отдельных блоков пользуется и формальными методами. Требуется знание типовых функциональных узлов, их свойств и параметров.

Микропроцессорная система создается в результате разработки комплекса программно-аппаратных средств. Разработка аппаратной части сводится к компоновке системы из типовых модулей: центрального процессорного элемента, различных видов памяти, адаптеров, контроллеров и внешних устройств. Способы подключения модулей к шинам микропроцессорной системы, описания основных модулей, *сведения о методике их программирования и применения приведены в предыдущих главах*. Дополнительно может использоваться литература [28], [29], [37], [39] и др. Довольно много отечественных и зарубежных изданий посвящено *вопросам разработки программного обеспечения*. Достаточно обратиться к [14].

Около 30 лет назад появление микропроцессоров предоставило широкому кругу разработчиков возможность не только самостоятельно проектировать, но и изготовлять устройства с достаточно сложным заданным поведением и высокими эксплуатационными характеристиками. В МПС реализуемость проектов с большим объемом программного обеспечения позволяет решать задачи практически любой алгоритмической сложности. Однако при этом выполняется последовательность достаточно

примитивных команд, поэтому скорость получения результатов оказывается зависящей от сложности проблемы.

В современных условиях многообразия задач, решаемых на конструктивно совпадающих устройствах, можно достичь, применяя *программируемую логику*. Структурная организация ПЛИС в наибольшей степени отвечает разбиению исходной задачи на параллельное исполнение отдельных ее составляющих. Поэтому целесообразно применение ПЛИС для решения задач, требующих повышенного быстродействия и допускающих распараллеливание обработки информации. Большие перспективы открываются при совмещении в одном кристалле обеих концепций. Изменение программ и перестройка архитектуры позволяют оптимизировать распределение между программной и архитектурной частями решения задачи в зависимости от требуемой сложности и заданного быстродействия устройства.

На основе дискретных ИС системы целесообразно строить при их относительно малой сложности. Усложнение функций системы ведет к резкому увеличению затрат не только на приобретение комплектующих изделий, но и на изготовление и отладку многокомпонентных печатных плат. При этом уменьшается надежность таких плат, увеличиваются габариты, потребляемая мощность и т. д. Привлекательность этой элементной базы для многих разработчиков состоит в отсутствии необходимости разрабатывать программное обеспечение.

Для других разработчиков, напротив, возможность реализовать новую продукцию, изменив только программное обеспечение, предопределяет стремление использовать микропроцессорную технику. В 80-х годах прошлого века микроконтроллеры и микропроцессорные системы феноменально быстро завоевали обширный рынок, что не в последнюю очередь обусловлено малым влиянием количества и сложности решаемых задач на требуемые ресурсы.

В настоящий момент от повального перевода всех проектов на ПЛИС (программируемые логические интегральные схемы) или СнПК (системы на программируемом кристалле) удерживают две основные причины. Во-первых, относительно высокие цены на комплектацию, а во-вторых, неготовность многих разработчиков переходить на новую элементную базу. В некоторых случаях свою роль играет и отсутствие отечественных ИС этого типа.

Львиной долей инженерных разработок аппаратуры в условиях современной России, по-видимому, как раз и явится *использование схем с программируемой структурой* для создания требуемых устройств и (или) их отладки. При этом программируемые ИС могут использоваться как в виде автономных устройств, так и в составе микропроцессорных систем.

Системы на кристалле и ПЛИС, благодаря определенной структурной избыточности и перестраиваемости как организации их блоков, так и связей между ними, рационально применять для отработки прототипов будущего устройства или даже на ранних этапах его внедрения.

В СнПК связь МП-ядра с периферийным оборудованием осуществляется без выхода за пределы кристалла, что позволяет существенно повысить производительность

МП-системы (путем увеличения как тактовых частот, так и разрядностей обрабатываемых данных).

Совмещение в СнПК аналоговых и цифровых блоков упрощает проектирование одного из самых сложных и трудоемких этапов — конструирования аналоговой части проекта. Профессионально решенные проблемы реализации аналоговых элементов и их стыковки как между собой, так и с цифровыми элементами, существенно упрощают работу проектировщиков.

В этой главе основное внимание уделено рассмотрению процедуры проектирования схем с программируемой структурой. Причинами этого являются:

- перспективность реализации проектов на схемах ПЛИС;
- недостаточное освещение этих вопросов в отечественной литературе;
- в процедуре проектирования ПЛИС содержатся почти все проектные процедуры, характерные для других типов специализированных микросхем;
- специфические этапы проектирования других типов специализированных микросхем менее интересны большинству разработчиков, поскольку выполняются изготовителем ИС, а не ими.

§ 12.3. Инструментарий проектировщика

Современное проектирование (даже для проектов не очень высокой сложности) немыслимо без привлечения *инструментальных средств*, среди которых ведущее место занимают системы автоматизированного проектирования — САПР. Современная переориентация ведущих разработчиков САПР с дорогих и поэтому имеющих ограниченное распространение рабочих станций на дешевые персональные компьютеры способствует расширению областей применения автоматизированного проектирования.

Чем сложнее (и дороже) проект, тем важнее автоматизация каждого этапа проектирования и тем более сложные средства могут и должны привлекаться для реализации каждого этапа. Для оценки современного состояния проблемы проектирования электронных систем приведем некоторые количественные характеристики больших проектов:

- для схем ASIC (Application Specific Integrated Circuits) — более 20 млн вентилей в кристалле, для ПЛИС — более 5 млн вентилей;
- для описания поведения проекта на системном уровне требуется более 0,5 млн строк кода на языке СИ;
- при описании проекта на уровне регистровых передач используется более 5 млн строк кода RTL.

Сложность, широкая номенклатура доступных проектных средств, большая стоимость полного проектного комплекса привела к использованию фирмами-поставщиками САПР понятия *платформа проектирования*. *Платформа* — это уже не просто компьютер с набором всех программных пакетов фирмы, а ком-

плекс только тех аппаратно-программных средств, который необходим для решения конкретной задачи разработчика. Понятие платформы включает не только набор скомпонованных и взаимно состыкованных средств, но и методику проектирования на его основе.

Средства системного этапа проектирования

Как уже отмечалось (см. рис. 12.2), проектирование начинается с *системного этапа*, на котором формируются *основные идеи* проекта и анализируется его реализуемость. Определяются ориентировочные затраты и стоимость конечного продукта. Выбирается платформа проектирования. Именно здесь закладывается набор требуемых далее средств. Выполнение работ этого этапа очень плохо поддается автоматизации.

На системном этапе выбирается характер технологической реализации отдельных ИС (стандартные дискретные компоненты, ASIC, FPGA, CPLD и т. д.), определяются производители и выбираются конкретные семейства используемых ИС. На этом же этапе общесистемная задача разбивается на функционально обособленные фрагменты. Основа разбиения — ориентация на реализацию фрагментов на специфическом элементном базисе (аналоговом, цифроаналоговом, микропроцессорном, базисе цифровых микросхем с фиксированными или программируемыми структурами и т. д.).

Результатом этого этапа является *спецификация* проекта. Под спецификацией понимают краткое описание значимых аспектов системы. В спецификации, как правило, в форме требований дается описание характеристик функционирования, структурного состава и других условий реализации. В процессе проектирования спецификация претерпевает последовательные изменения, отражая основные этапы проектирования и уменьшая степень абстрагирования от конечной технической реализации.

В результате совместной работы заказчика и проектировщика на основании оценки требуемого времени и ресурсов разработки создается функциональная спецификация каждой обособленной ветви проектирования, включая перечень средств, необходимых для последующих этапов проектирования.

Для конкретизации данных об отдельных составляющих требуется определение общесистемных характеристик. Здесь и привлекаются ПК с соответствующими программными пакетами. Для дискретных фрагментов чаще всего используется математический аппарат теории массового обслуживания и соответствующие языки: *GPSS* (General Purpose Simulating System), Simula). При анализе частей проекта, описываемых в понятиях теории динамических систем, ориентируются на программные пакеты *MathCAD*, *MATLAB* и *SIMULINK*. Исследования аналоговых и цифроаналоговых фрагментов выполняются с привлечением программных пакетов, основой которых является модификации программы схемотехнического моделирования Spice (Simulation Program with Integrated Circuit Emphasis).

Разработка специфических фрагментов проекта

Последующие этапы проектирования выполняются с привлечением средств автоматизации. Следует отметить общность процедур проектирования по всем параллельным ветвям (независимо от специфики элементной базы). Как разработка программного обеспечения для МП (МК) ядра, так и разработка дискретной и аналоговой частей проекта укрупненно могут рассматриваться как последовательность *трех этапов работы с САПР*, выполняемых после составления спецификации:

- ввод в САПР исходной информации;
- обработка введенной информации;
- обработка и анализ полученных результатов.

Конкретное содержание этапов для аппаратной и программной части проекта (а тем более, цифровой и аналоговой частей), естественно, различное. Разработка аппаратной части проекта приводит к синтезу устройства (или устройств) в базисе заданных элементов, а компиляция программной части проекта приводит к синтезу кодового представления программ. Полученные результаты требуют тщательной проверки, поэтому за этапом синтеза следует этап анализа, осуществляемый моделированием и теоретической верификацией.

Моделирование, как правило, имеет несколько уровней с разной степенью отображения свойств реального объекта. На самом верхнем уровне проверяется правильность функциональной организации устройства или программы. На следующем уровне учитываются некоторые особенности структурной реализации объекта (элементная база, способ взаимодействия программных фрагментов и т. д.). На низшем уровне учитываются особенности физической реализации устройства (задержки сигналов в схемах устройства, связанные с конкретным размещением элементов, время исполнения отдельных программных фрагментов, влияние параметров трассировки на работу схем, учет паразитных эффектов, шумов, температуры и т. д.). В результате моделирования могут выявиться ошибки, требующие исправления, что придает процессу проектирования итеративный характер с возвратами к прежним этапам и введением в проект нужных коррекций.

По мере отработки решений по отдельным ветвям проектирования все большее значение приобретает анализ их взаимодействия. Наиболее остро эта проблема встает при стыковке программной и аппаратных частей проекта. Традиционно они детально анализировались и отрабатывались только на этапе комплексной отладки проекта. Естественно, такое последовательное проектирование (тем более с учетом многократных итерационных возвратов к началу проектных процедур) существенно замедляет проведение работ. Поэтому понятно стремление разработчиков использовать унифицированные технические средства и приемы, позволяющие ускорить процесс проектирования.

Средства разработки процессорной части проекта

При разбиении проекта разработчик, прежде всего, оценивает целесообразность применения процессорного блока со стандартной архитектурой. Наличие процессора позволяет попытаться возложить именно на него решение большинства задач проекта. Практическая невозможность поручить процессору решение всех задач заставляет выделить специфические задачи, которые (как уже отмечалось ранее) могут быть отнесены к одной из двух групп со специфическим подходом к их решению. К первым относятся задачи, требующие аналоговой и аналого-цифровой обработки. Ко вторым — задачи, связанные с быстродействием, не поддерживаемым выбранным типом МП.

Разработка процессорного фрагмента предполагает, как показано на рис. 12.6, выполнение двух параллельно решаемых задач — проектирования аппаратной и программной (в английской терминологии соответственно HW (*Hardware*) и SW (*Software*)) составляющих процессорной системы. Наиболее ответственным при этом является оптимальность распределения задач между ними. Традиционная методика проектирования аппаратной части микропроцессорных систем на основе дискретных компонентов предусматривает проектирование электрических схем и их моделирование. Электрические схемы лучше всего создавать с помощью программных пакетов, поддерживающих в дальнейшем автоматизацию разработки и изготовление печатных плат. К числу таких пакетов относятся *P-CAD*, *OrCAD*[41], *Protel* и некоторые другие. Наиболее мощные пакеты позволяют моделировать поведение аппаратуры не только на логическом уровне, но и с учетом паразитных эффектов в печатной подложке.

Для МПС, интегрированных в одном кристалле (МК или СнПК) необходимость отдельного моделирования аппаратной части, как правило, отсутствует. Необходимость определяется только степенью отработки архитектуры используемого МП ядра. Как правило, такая проверка может быть отодвинута на более отдаленные стадии проектирования и совмещена с комплексной отладкой.

Самая ответственная задача в разработке МПС — разработка программного обеспечения. Ключевым средством этого процесса является *компилятор* — компьютерная программа, упрощающая написание и отладку пользовательских программ. Обычно используемые языковые средства — либо язык ассемблера, либо C/C++. Сложность и длительность процедуры отладки заставляет привлекать разнообразные аппаратно-программные средства. Обычно весь комплекс таких отладочных средств, сконцентрированных вокруг отладочного (инструментального) компьютера, называют *средой или системой разработки*. Отладка может ориентироваться на работу с виртуальным прототипом (моделью), с физическим прототипом или с конечной продукцией.

Отладка на модели — способ, доступный с самых начальных этапов проектирования. Достоинство метода — наблюдаемость (видимость) и управляемость всеми объектами модели.

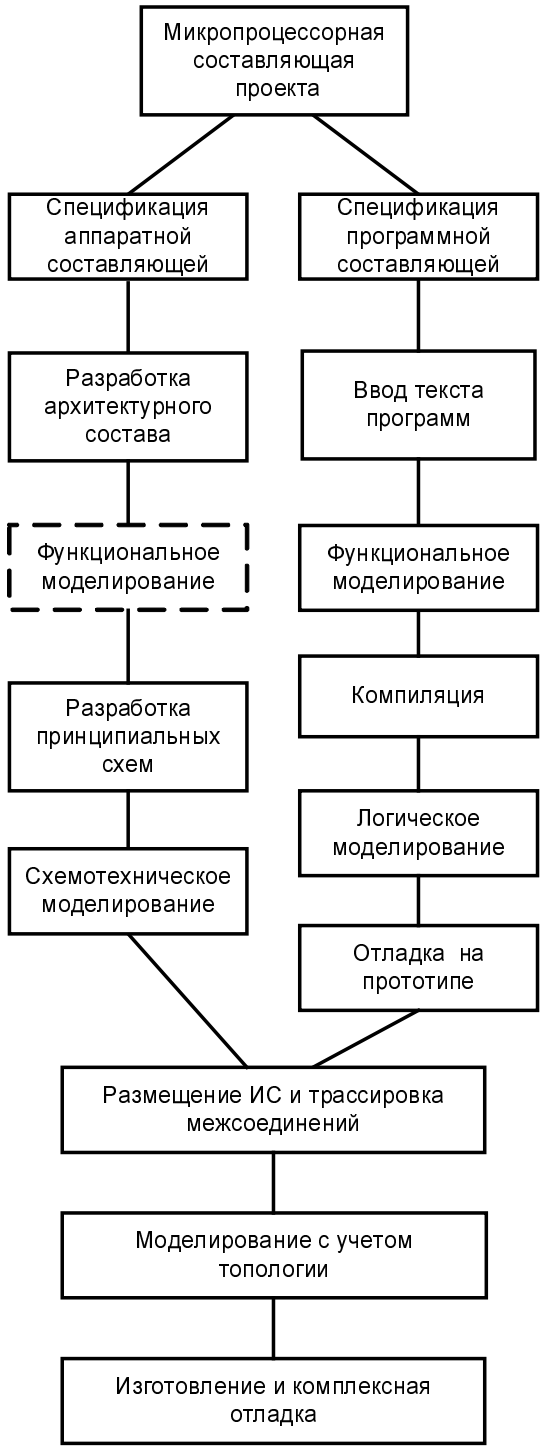


Рис. 12.6. Этапы разработки процессорной составляющей проекта

Степень учета в модели поведения не только программной составляющей, но и аппаратуры проекта определяет свойства и название системы моделирования. Чем больше аппаратная часть, тем больше замедляется работы модели, но точнее отражаются свойства объекта. Программное моделирование работы только процессорного блока носит название *кросс-ассемблирования*. Работа с программной моделью процессора и моделями стандартной периферии системы — *симуляция*. Совместная работа программной модели одной части проектируемой системы с физически реализованной другой частью носит название *эмуляции.* Под эмуляцией понимают и работу с физической реализацией фрагмента проектируемой системы, отличающейся от реализации, планируемой в конечной продукции. Программной моделью может быть любая часть системы или процессор, или периферийные блоки. Адекватность задания асинхронного взаимодействия модели среды и модели системы — основная проблема, с которой сталкивается проектировщик при работе с программами моделировщиками.

Отладка на прототипе — способ получения наиболее достоверных данных о работе разработанного программного обеспечения в его взаимодействии с аппаратурой системы и внешней средой.

Конечно, только отладка на конечном продукте при задании различных вариантов взаимодействия с окружающей средой может дать разработчику уверенность в правильности взаимного функционирования аппаратуры и программного обеспечения.

Средства разработки цифровой части проекта

Цифровая часть проекта может быть выполнена на дискретных компонентах с фиксированной или (и) программируемой логикой (ПЛИС). Целесообразность использования тех или иных компонентов с точки зрения затрат и достигаемых характеристик проектируемого устройства рассмотрена ранее, в этом разделе акцент делается на различиях в маршруте и средствах проектирования. Последовательность разработки цифровой части проекта приведена на рис. 12.7.

Технология *проектирования на дискретных ИС* имеет длительную историю. Несмотря на постоянное увеличение логической мощности дискретно монтируемых ИС, многое в процедуре проектирования осталось прежним. Современный элементный базис не исключает использования дискретных ИС, так как выполнение ряда функций (гальванической развязки, формирования тактовой частоты, преобразования питающих напряжений и т. д.) не включается в современные ПЛИС и даже СнПК.

В проектировании на дискретных компонентах следует выделять *две составляющие*: конструкторскую и схемотехническую. Конструкторское проектирование, не меняясь в своей сути, медленно эволюционирует под влиянием технологических изменений в организации корпусов ИС, способов их монтажа и т. д.

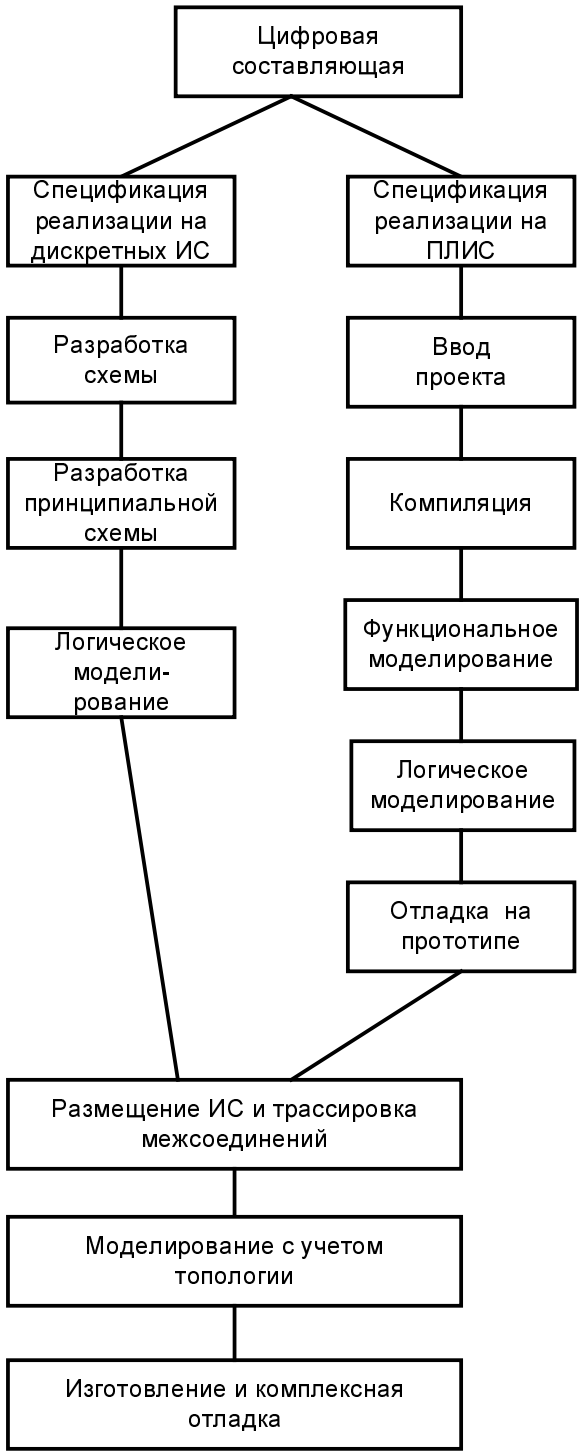


Рис. 12.7. Этапы разработки цифровой составляющей проекта

Изменения в схемотехнической разработке более динамичны. Реализация на СИС и МИС тесно связывает схемотехническое и конструкторское проектирование, в отличие от реализации на ПЛИС, когда взаимосвязь этих работ все более ослабляется.

Основа схемотехнического этапа работ — составленная спецификация проекта, содержащая функциональные и структурные схемы будущего устройства. Этот этап заканчивается составлением принципиальных электрических схем соединений ИС, выбранных для реализации проекта. В состав этих схем могут органически входить и аппаратные ресурсы микропроцессорной части. Разработка электрических схем хорошо поддерживается современными САПР с ориентацией на последующее использование электрических схем при конструировании печатных плат. Описания конструктивных параметров элементов дискретной логики включаются в технологические библиотеки фирм-изготовителей ИС и могут быть найдены разработчиками в Интернете. Если вся проектируемая система размещается в одной интегральной схеме, разработка конструкции печатной платы может быть начата задолго до завершения работ над внутренними схемами ИС.

Другая ситуация складывается с моделированием разрабатываемых схем. Программные пакеты ряда фирм (*Micro-Cap*, *DesignLab*, *Aplac*, *Electronics Workbench* и другие) были разработаны для моделирования на схемотехническом уровне. Однако уровень элементного базиса был очень низок и в лучшем случае соответствовал схемам малой степени интеграции. Это связано с тем, что эти пакеты были ориентированы на совместное моделирование цифровых и аналоговых фрагментов и базировались на языках и средствах смешанного представления сигналов (типа программы PSpice, основанной на решении дифференциальных уравнений).

Основной проблемой для большинства современных САПР является интеграция в их систему моделирования моделей компонентов, описанных на языках высокого уровня. Попытку включить в состав своего пакета конструкторского проектирования моделирующую программу одной из первых предприняла фирма Personal CAD Systems. Система моделирования *PC-LOGS* для дискретных компонентов была основана на троичном асинхронном событийном моделировании и позволяла создавать собственные модели, дополняя ограниченную библиотеку встроенных моделей типовых компонентов. Пользовательские модели могли строиться на уровне структурного объединения элементов низшего уровня иерархии или на уровне поведенческого описания компонентов. Для создания моделей предлагался специальный язык описания поведения. Особенно большого распространения версия в DOS не получила, а в варианте Windows она даже не была реализована.

Наиболее известна мощная сквозная система проектирования DesignLab, ее возможности в версии 8.0 подробно изложены в книге [42]. Попытки решить в рамках одной системы все вопросы схемотехнического и конструкторского проектирования пока приводят к очень сложным, громоздким, не оперативным и, конечно, дорогим системам. В современных САПР влияние физической реализации проектов для простоты учитывается с функциональным выделением этой проблемы. Например, в большинстве САПР осуществляется обособленный анализ электрических

характеристик цепей печатных плат для учета их паразитного влияния на передаваемые сигналы или внешние узлы.

Ввиду важности и специфичности разработка цифровой части проекта на ПЛИС будет детально рассмотрена в следующем параграфе.

Средства разработки аналоговых и аналого-цифровых фрагментов

Аналоговые и аналого-цифровые фрагменты проекта (если они не интегрированы в СнПК) могут быть (как и цифровые фрагменты) реализованы в форме совокупности отдельных ИС с фиксированной архитектурой и параметрами либо в форме программируемых аналоговых ИС (ПАИС). Поэтому процедура проектирования таких фрагментов внешне совпадает с последовательностью, приведенной на рис. 12.7 для цифровой части проекта. Вместе с тем следует отметить, что, хотя объемы аналоговой части, как правило, не превышают 15% от общего объема проекта, их отладка "съедает" до 80% общего времени отладки. Те же соотношения характерны и для разработки топологии печатных плат с аналоговыми элементами. При применении дискретных компонентов этап разработки электрических схем отличается от соответствующего этапа для цифровых компонентов только набором исходных ИС.

Сложность проектирования аналоговых и смешанных схем определяет важность этапа их моделирования. Возможности отечественных проектировщиков использовать программные моделировщики ведущих мировых фирм ограничиваются высокой (сотни тысяч долларов) стоимостью лицензий на эти пакеты. Сердцевиной большинства таких систем являются современные разновидности программы PSpice. Последние разработки ведущих фирм, таких как Cadence [VII], представляют собой сквозные системы проектирования печатных плат, что позволяет моделировать проект до и после автоматической трассировки соединений его компонентов. Моделирование с учетом паразитного влияния разнообразных элементов топологии печатной платы существенно повышает эффективность разработки, увеличивает ее достоверность и сокращает общее время проектирования.

Работа и средства этапа комплексной отладки проекта

После завершения разработки принципиальных электрических схем фрагментов на цифровых, аналоговых элементах, создания файлов прошивки памяти команд МП и файлов конфигурирования программируемых БИС, а также проработки конструктивной реализации системы возможна реальная проверка работы всего устройства либо его отдельных фрагментов — физическая реализация проекта или его частей. Однако для экспериментальной проверки принятых решений совсем не обязательно ждать реализации будущей системы целиком. Для отладки могут использоваться средства моделирования, эмуляции (программной и аппаратной) и

средства физического прототипирования. Естественно, предпочтительной представляется практическая проверка проекта с использованием предлагаемых различными фирмами специальных отладочных средств. Названия таких средств отражают их целевую направленность. Спектр отладочных средств простирается от предназначенных для предварительного знакомства с БИС рассматриваемого класса (обычно называемых *Starter Kit*) и средств для отладки прикладных проектных решений (обычно называемыех *Evaluation Board* или *Development Board*) до средств, замещающих на начальных этапах выпуска готовой продукции оборудование, которое еще находится на этапах конструкторско-технологической разработки (обычно называемых *Prototype Plate*). Несмотря на некоторые отличия по имеющимся ресурсам и предлагаемым возможностям, любая плата позволяет производить конфигурирование системы под требования проектировщика и выполнять желаемые эксперименты.

Помимо перечисленных ранее средств применяются и традиционные средства отладки, такие как осциллографы, программаторы, эмуляторы, логические анализаторы и т. д.

Специфика конструирования и отладки проектов на ПЛИС и СнПК

Существуют специфические особенности процедуры создания и отладки проектов, конструктивно расположенных в одном кристалле — ПЛИС или СнПК. Хотя функциональное назначение элементов, входящих в ИС этих классов, остается таким же, как и при реализации на дискретных компонентах, резко возрастает как сложность самих проектов, так и интеллектуальная мощность привлекаемых отладочных средств. Возможность упреждающей конструкторской разработки приводят к тому, что самым "узким" местом становится комплексная проверка проекта по параллельным направлениям. Все усилия проектировщиков направлены на то, чтобы как можно раньше приступить к комплексным экспериментам.

Следует выделить *два основных момента*. Первый связан со сложностью отладки современных проектов традиционными методами и средствами. Второй связан с легкостью и оперативностью внесения изменений в проект на любом этапе (в том числе даже после передачи физических образцов заказчику).

Интеграция в одном кристалле огромной логической мощности сопровождается уменьшением количества внешних контрольных точек системы (вывод некоторых промежуточных точек проекта на выходные контакты бывает даже недопустим, так как может приводить к потере производительности проекта). Отсюда следует неэффективность применения традиционных методов отладки, базирующихся на подключении разнообразной контролирующей аппаратуры к тестовым точкам отлаживаемой системы. Поэтому резко возрастает интерес к системам моделирования, позволяющим при отладке одновременно наблюдать поведение совместной работы аппаратуры и программ.

Другой подход, открывающий проектировщику доступ к внутренним элементам проекта, состоит в интеграции фрагментов отладочных средств в состав целевой системы. Допустимыми считаются затраты на эти цели от единиц до десятков процентов общих ресурсов БИС. При использовании расширенного варианта интерфейса JTAG подобный подход даже не требует увеличения числа контактов БИС. Более того, репрограммируемость ПЛИС и СнПК позволяет загружать в них специальные тестовые конфигурации, которые ориентированы на контроль и (или) настройку оборудования, расположенного вокруг БИС.

Ввиду легкости репрограммирования аппаратной части проекта, а для СнПК и программной части, этап экспериментальных работ с БИС может быть отложен до завершения конструкторской разработки печатной платы. Даже значительные изменения схемы обычно не влекут за собой столь катастрофических последствий, как в проектах на основе жесткой стандартной логики. Наличие у фирм-разработчиков ПЛИС и СнПК широкой номенклатуры БИС различной логической мощности с совпадением общего числа выходных контактов, их функционального назначения и расположения делает модификацию проектов вопросом скорее экономическим (более мощные БИС стоят дороже), чем техническим. Поэтому экспериментальные работы целесообразно производить на различных отладочных прототипных системах исключительно для ускорения общего процесса проектирования, поскольку в этом случае удается совместить во времени этапы конструкторской разработки печатной платы и экспериментальных работ с разработанным проектом.

Итак, особенности БИС с программируемой структурой приводят к определенным изменениям не только в процедуре изготовления конечной продукции, но и в процедуре контроля и отбраковки готовой продукции.

§ 12.4. Системный этап проектирования цифровых устройств на базе ПЛИС

Все современные методики проектирования цифровых устройств (ЦУ) на базе сложных программируемых БИС/СБИС основаны на применении САПР. Однако прежде чем приступить к работе с ними, проектировщик как минимум должен решить, на какой САПР он будет выполнять проектирование. Выбор САПР, на которой (или на которых) будут выполняться дальнейшие работы, не в последнюю очередь связан с выбором средств описания проекта и его частей и способах занесения информации об этих основных частях в САПР. Для этого требуется предварительно представить исходную проектную проблему в блочно-функциональном виде. Все эти подготовительные работы на рис. 12.3 были включены в этап под названием системное проектирование. Автоматизация всех работ этого этапа в настоящий момент не выполняется, а ведущие мировые фирмы-производители САПР стремятся автоматизировать лишь отдельные работы этого этапа.

Выбор САПР

Правильный выбор САПР — важнейшее условие эффективного проектирования и ускорения выпуска продукции, т. е. сокращения времени до продажи (Time-to-Market).

В общем случае при таком выборе приходится учитывать целый ряд соображений, а именно:

- распространенность САПР;
- цену САПР, ее сопровождения и модификаций;
- широту охвата задач проектирования;
- эффективность (прежде всего скорость) выполнения различных этапов и уровней проектирования;
- удобство работы с САПР и ее "дружественность";
- наличие широкой библиотечной поддержки стандартных решений;
- возможность и простоту стыковки с другими САПР;
- возможности корпоративной работы.

Очевидно, трудно рассчитывать на такой выбор САПР, который удовлетворял бы сразу всем зачастую взаимно противоречивым требованиям и свойствам. Сложность работы с современными САПР постоянно возрастает. Ситуация с овладением новыми средствами в последнее время улучшается, поскольку начали появляться отечественные книги, посвященные описанию работы с САПР [4], [8], [21], [22] и др.

Наилучшие результаты достигаются при выборе маршрутов проектирования, основанных на *групповом использовании* САПР, когда каждая из них обеспечивает наилучшие решения для соответствующего этапа проектирования. Целесообразности такого подхода способствует и позиция фирм-производителей ПЛИС. Необходимость использования САПР этих фирм, как минимум на последних этапах компиляции проекта (связь смонтированного проекта с конфигурационными файлами), с одной стороны, и малая эффективность этих САПР на этапах синтеза (в меньшей степени и моделирования), с другой стороны, определяют целесообразность привлечения в маршрут проектирования САПР сторонних фирм. При экономической оценке подобных подходов следует учесть универсальность этих САПР и легкость их переориентации на элементную базу других фирм-производителей ПЛИС.

Представление проекта на блочно-функциональном уровне

Первая работа, выполняемая разработчиком над проектом — переход от *технического задания* (ТЗ) к формализованному описанию проектируемого устройства. Как правило, ТЗ является смесью словесного и технического описания, его формализация приводит к вычленению основных блоков устройства (или алгоритма) и определению их связей и/или взаимодействия. Эта процедура носит название —

спецификация проекта. Для спецификации проектов, содержащих микропроцессорную составляющую, чаще всего используются специальные разновидности языка C++ [VII, XXVII].

В сущности именно в этот момент реализуются начальные действия первого этапа проектирования. Формально же первый этап — разбиение задачи на отдельные функционально обособленные подзадачи — этап декомпозиции. Разработчик составляет содержательные граф-схемы алгоритма и (или) функциональные блок-схемы устройства.

Способ и средства разбиения чаще всего и прежде всего определяются симпатиями проектировщика и лишь иногда являются предопределенными. Сама форма ТЗ может провоцировать проектировщика на использование тех или иных средств, хотя не исключено, что более эффективным мог бы быть другой метод описания проекта или его фрагментов. Для спецификации микропроцессорной составляющей проекта проектировщики обычно стремятся к составлению схем алгоритмов, а для аппаратной части к составлению функциональных схем. Первый вариант больше тяготеет к последующей программной реализации, а второй приближает к уровню регистровых передач.

Как уже указывалось, возможно как только временное (поведенческое), так и только пространственное (архитектурно-структурное) описание проекта. Однако обычно целесообразно совмещать обе возможности.

Разбиение ЦУ на два блока: операционный и управления. Такое разбиение характерно для разработки цифровых устройств. Операционный блок (ОБ) выполняет преобразование данных и строится из стандартных частей, а блок управления (устройство управления — УУ) обеспечивает необходимую последовательность операций, выполняемых в ОБ (одном или нескольких). Для этого УУ передает на входы ОБ управляющие сигналы. Последовательность действий и, следовательно, управляющих сигналов зависит от результатов операций в ОБ и внешних воздействий. Отсюда видно, что *УУ удобно задавать в форме конечного автомата с памятью* (АП) того или иного типа. Разработаны и формальные методики преобразования алгоритмов, описывающих функционирование ЦУ, в описания конечных автоматов.

В сложных проектах возможно разделение ЦУ на несколько функционально слабо связанных пар ОБ-УУ на одном уровне иерархии или создание пары, иерархически погруженной в ОБ (реже в УУ).

Операционный блок обычно представлен набором регистров, логических схем (как правило, многофункциональных и управляемых), буферных схем и коммутируемых связей между ними. Важно лишь наличие на более низких иерархических уровнях описания проекта однозначной трактовки функционирования всех элементов ОБ.

Разработка общей структуры операционного блока. Основа этапа — выбор допустимых для данного уровня иерархии элементов, определение связей между ними и, если параметры элементов являются настраиваемыми, то и их настройка.

Описание работы управляющего автомата (УА). На этом этапе определяется функционирование УА, обеспечивающее требуемое взаимодействие элементов ОБ. Следует подчеркнуть, что два последних этапа сильно взаимосвязаны, и, если не разрабатываются параллельно, то обычно выполняются итерационно.

Средства описания проекта

Применение САПР требует эффективных, наглядных, управляемых и контролируемых средств описания проекта. Описания проектируемого устройства и его частей на различных этапах проектирования имеют различные цели, поэтому современные САПР допускают описание объектов разными способами. Основная задача любого способа описания — передача разработчиком в САПР моделей, описывающих, что и как надо реализовать в процессе обработки и преобразований входного описания.

Ведутся интенсивные поиски *универсального способа* описания, пригодного для всех уровней проектирования и независимого от формы реализации проектов, однако пока рано говорить об успешности этих поисков. Для описания сложных систем, реализация которых явно потребует использования в том или ином виде процессорного ядра, наибольшее распространение получили методы задания спецификации проекта на языках, производных от C++. Фирмы, ведущие работы в этом направлении, включают в свои САПР средства моделирования, работающие на уровне *исполняемых спецификаций*. Подобный подход позволяет уже на самых ранних этапах проектирования оценить работоспособность принимаемых решений.

Основная проблема на этом пути — трудность создания программ, автоматизирующих перевод описания проекта с уровня исполняемых спецификаций на уровень, "понятный" синтезирующим компиляторам. Фирма Synopsys [XXVII] выпустила пакет *CoCentric SystemC Compiler*, помогающий и упрощающий перевод описаний с *языка SystemC* на синтезируемые языки уровня регистровых передач. Однако широкого распространения эти компиляторы пока не получили.

В настоящее время к наиболее распространенным универсальным способам описания, применимым для любого уровня иерархии проекта, относят *графический и текстовый*. Реже используются непосредственная разводка схем FPGA в редакторе топологии, описания в виде требуемых временных диаграмм и др. Каждый из способов описания проекта имеет свои достоинства и недостатки.

Графическое представление проекта

Графическое представление проекта создается в базисе допустимых для выбранного редактора САПР примитивов (для схемного редактора, например, в базисе элементов стандартной серии ТТЛ(Ш)). Главное достоинство графического способа — его традиционность и наглядность, связанные с привычностью разработчиков к восприятию изображений схем. Конечно, это преимущество проявляется только при правильном иерархическом и структурном разбиении проекта.

Варианты схемотехнического задания представляются интуитивно легко осваиваемыми и более наглядными. Смешанные (текстовые и графические) методы позволяют совместить достоинства обоих подходов. В последнее время схемные методы все чаще оставляются только для отчетной и обучающей документации и повсеместно заменяются текстовыми.

Графическое представление может использоваться как наглядное отражение текстового описания. Большинство современных САПР обеспечивает одновременное формирование текстового и графического описаний, что позволяет разработчику в зависимости от ситуации пользоваться то одним, то другим методом.

В отличие от текстовых, графические способы представления проекта обычно узко специализированы и требуют особых средств или приемов для переноса информации о проекте в другую среду. Создание единой базы данных для САПР различных фирм — вопрос отдаленного будущего. На сегодня возможные варианты решения этой проблемы состоят либо в применении специальных универсальных языков передачи информации о проекте типа языка *EDIF* (Electronic Design Interchange Format), либо в задании проекта в текстовой форме.

Текстовое описание

Современные языки описания аппаратуры (HDL, Hardware Description Languages) допускают описание проектируемого устройства как с точки зрения его *поведения*, так и с точки зрения его *структуры*. Эти возможности делают все более распространенным представление проекта в форме текстового описания алгоритмов функционирования его фрагментов в сочетании с текстовым же описанием межблочных соединений для сложных проектов. Достоинства текстового способа описания проекта заключаются в его компактности и относительной простоте автоматизации любых преобразований, включая начальную генерацию описания проекта. В проекты легко вносить изменения (добавлять или удалять объекты), объекты проектов легче модифицируются, можно создавать блоки с легко задаваемыми и изменяемыми параметрами. Очень важна возможность использования стандартных универсальных языков типа HDL, обеспечивающая простоту переноса проекта с одной аппаратной платформы на другую и переход от одной САПР к другой.

Языковое описание аппаратуры получает все большее распространение. Текстовые описания имеют две основные разновидности — *языки низкого уровня* (аналоги языков программирования типа ассемблера) *и высокого уровня*.

Языки низкого уровня

Языки низкого уровня ближе к аппаратным средствам, вследствие чего представляют для компиляторов потенциальные возможности создания проектов с более выигрышными параметрами. Платой за это является обычно жесткая ориентация на определенную аппаратуру и производящую ее фирму. Примерами таких языков

могут служить языки *PLDASM* (фирма Intel), *AHDL* (фирма Altera) и *ABEL* (фирмы Xilinx). С помощью языков низкого уровня легче создавать проекты с наилучшими временными параметрами, т. к. в проектах будут учтены специфические особенности архитектуры CPLD или FPGA той или иной фирмы.

Сами эти языки развивались от описания комбинационных схем в форме логических выражений и их соединений с другими типовыми элементами программируемой логики (триггерами, элементами памяти и т. п.). Компиляторы с таких языков, прежде всего, должны были создать список соединений в базисе программируемой логики, для чего осуществляли перевод логических выражений, минимизацию и определение настраиваемых характеристик полученных схем и соединение компонентов между собой. На заключительных этапах компиляции САПР решали задачу реального размещения и соединения компонентов и формирование файла конфигурирования.

При разработке современных очень сложных проектов языки низкого уровня могут применяться для проектирования только небольших и ответственных фрагментов.

Языки высокого уровня

Языки высокого уровня менее связаны с аппаратными платформами и поэтому более универсальны. Среди них наиболее распространены языки *VHDL* и *Verilog*. Эти языки, как и другие алгоритмические языки высокого уровня, в принципе позволяют описать любой алгоритм в последовательной форме, т. е. через последовательность операторов присвоения и принятия решений. Основное их отличие в способности отражать также и параллельно исполняемые в аппаратуре действия, представляемые отдельными параллельно выполняемыми процессами с общим инициализирующим воздействием.

Средства описания автоматов

Автоматы играют важнейшую роль в проектировании современных систем. Формы и средства описания автомата разнообразны. *Современная тенденция состоит в переходе от прямой записи логических выражений, определяющих поведение автоматов, к предварительной графической форме задания*. Описание в виде *граф-схемы* переходов (диаграммы состояний) становится одним из самых распространенных вариантов исходного задания автоматов (в английской терминологии State Machins). Графические редакторы для создания автоматов включаются в состав средств ввода исходных проектов большинства современных САПР.

Хотя редакторы разных фирм имеют различный интерфейс, отличаются правилами записи условий и имеют другие особенности, для всех них характерны исключительная простота, естественность и "дружественность" интерфейса с пользователем, а также отсутствие жесткой необходимости знания выходного языка редактора. В большинстве случаев редакторы по выбору создают выходные тексты на

языках VHDL или Verilog. Графические редакторы обладают полным набором средств для выполнения всей проектной процедуры разработки УА, позволяющих реализовать следующие операции:

- рисовать граф переходов, включая наименования состояний, направления, условия и приоритеты условий переходов, формируемые сигналы и способы их образования;
- проверять корректность составленного графа переходов (повторение имен, неоднозначность или некорректность перехода и т. д.);
- компилировать проект (формировать выходной текстовый файл) в выбранном языковом базисе;
- моделировать поведение автомата в интерактивном или компиляционном режиме (дополнительные возможности некоторых редакторов).

В последующих разделах этой главы приведены результаты работы программы HDL Designer фирмы Mentor Graphics [XVII].

Заметим, что *специфика продукции той или иной фирмы-производителя ПЛИС сказывается и на языках высокого уровня*, выражаясь прежде всего в отличиях в библиотеках, требуемых для работы, и в сложности и вариантности допустимых синтаксических конструкций для компиляторов. Конечные результаты компиляции одной и той же исходной граф-схемы автомата или последующей компиляции одной и той же программы с языка высокого уровня в загрузочный файл ПЛИС, полученные от компиляторов разных фирм, могут существенно различаться и иметь различную эффективность. Программа StateCAD Version 3.2 пакета Workview Office (сейчас собственник фирма Aldec[2]) удобна тем, что перед трансляцией графа переходов нужно задать не только желательное языковое представление (VHDL, AHDL, Verilog, ABEL и т. д.), но и фирменные атрибуты, что позволяет оптимизировать запись автомата и избежать применения синтаксических конструкций, недопустимых для компиляторов соответствующих фирм.

Особенностью программы HDL Designer является разнообразие форм графического задания проектов, включающего структурное описание, потоковое описание, описание автоматов, таблиц истинности и в том числе редактора стандартных схемотехнических элементов. Выходной язык программы VHDL или Verilog.

Как уже отмечалось, при использовании графических редакторов от пользователя не требуется обязательное владение выходным языком редактора. Однако в определенных случаях такое владение исключительно полезно. Знание языковых конструкций полезно, например, в ситуациях, когда автомат должен быть минимизирован по тем или иным параметрам, прежде всего по временным интервалам между формируемыми выходными сигналами, что может приводить к временным состязаниям сигналов. Именно в этих случаях владение языком и искусство проектировщика облегчают получение наилучших результатов.

§ 12.5. Маршрут проектирования ПЛИС и возможности типовых САПР

Этапы проектных процедур с использованием САПР

Разработка проекта обычно выполняется в следующем порядке:

1. **Ввод данных о проекте в САПР.** После составления проекта и его функциональной проверки информация о нем заносится в САПР. Занесение информации осуществляется с помощью редакторов, входящих в состав выбранной САПР.

   Для сложных иерархически организованных проектов помимо редакторов низшего уровня большую роль играют программы управляющих менеджеров проекта, редакторов проектной иерархии, блочные редакторы, символьные редакторы.

2. **Компиляция проекта.** После занесения информации о проекте или о его наиболее ответственных частях можно приступать к самому ответственному этапу проектирования — *компиляции*. Именно во время или сразу после компиляции выявляется большинство скрытых ошибок и "нестыковок".

   Технически компиляция в САПР разбивается на ряд последовательных подэтапов: построение базы данных проекта, формирование списка соединений, проверка проектных правил и контроль соединений, логическая минимизация проекта, разбиение на блоки и их размещение, конкретизация физически реализуемых межсоединений, определение требуемых аппаратных ресурсов и в конце формирование загрузочного (конфигурационного) файла.

   На любом подэтапе могут возникать ошибки, требующие повторной компиляции после их коррекции. Результат компиляции — загрузочный файл, т. е. конфигурационная информация для выбранной микросхемы ПЛ. Помимо этого, обычно создается и файл отчета, содержащий всю информацию как о процессе компиляции, так и о его результатах.

3. **Тестирование проекта.** Тестирование разработанного устройства, а в маломальски сложных проектах и отдельных его фрагментах — один из важнейших этапов проектирования, поскольку практически не бывает бездефектных проектов, созданных с чистого листа. Обнаружение дефектов проекта — сложнейшая задача. Скорость и тщательность тестирования во многом зависят от искусства разработчика.

   Для сложных проектов важным представляется функциональное моделирование наиболее существенных фрагментов или блоков проекта. *На этапе функционального моделирования*, прежде всего, осуществляется разработка требуемых тестовых примеров. Далее производится ввод этих примеров и выполняется собственно моделирование. Результаты моделирования позволяют решить: переходить ли к следующему этапу или вернуться к предыдущему.

Программы для тестирования (так называемые *Test-Bench*) могут быть построены на основе архитектурно-поведенческого тела, в котором проектируемый модуль представлен как структурный компонент, а генератор воздействия — в поведенческой форме.

В большинстве реальных ЦУ после подачи на них некоторых начальных данных выполняются несколько повторяющихся циклов. Необходима проверка работы устройства на нескольких наборах однотипных данных, поэтому можно рекомендовать следующую структуру программного модуля (процесса), представляющего тестовое воздействие: генерация сигналов начальной установки, затем реализация двух вложенных циклов, причем внутренний цикл последовательно формирует тестирующие сигналы для выполнения действий на одном наборе входных данных, а во внешнем производится их изменение.

В современных САПР широко распространено тестирование путем выполнения экспериментов над *программной моделью* спроектированной схемы. В отличие от функциональной модели, модель отражает все характерные свойства схемы. Моделирование с учетом конкретной реализации позволяет определить, достигнуты ли при проектировании требуемые или допустимые временные характеристики. Тестирование требует задания не только моделируемого устройства, но и моделей внешних сигналов. Задание этого взаимодействия выполняется с помощью редакторов временных диаграмм, которые делятся на *компилирующие и интерпретирующие*.

В многооконных САПР интерпретирующего типа результаты моделирования отображаются для текущего момента модельного времени во всех видах отображения проекта (сигналы в электрических схемах, в топологии). Поведение тестовых сигналов задается интерактивно, поэтому в этих редакторах легко изменить дальнейший ход эксперимента и состав отображаемых сигналов.

В компилирующих редакторах отображается ход всего эксперимента сразу. Любые изменения условий проведения теста требуют нового цикла компиляции эксперимента. Достоинством компилирующих систем моделирования является минимизация временных затрат.

4. **Определение временных характеристик разработанного устройства.** Современные САПР имеют внутри себя полную информацию о структуре проектируемого устройства и временных параметрах всех его компонентов, и это позволяет автоматизировать процесс вычисления разнообразных временных характеристик проекта.

   В большинстве САПР предусмотрено автоматическое вычисление трех основных классов временных параметров:

   - минимальных и максимальных задержек между источниками (входными сигналами) и приемниками (выходными сигналами), информация о которых выдается в виде матрицы задержек;

- максимально возможной производительности устройства (пропускной способности) в виде максимальной частоты тактирования элементов памяти, используемых в проекте;
- времен предустановки и выдержки сигналов, гарантирующих надежную работу схем при фиксации сигналов в синхронных элементах памяти.

Многие САПР позволяют также выделять критические пути передачи и преобразования информации для схемного или топологического представления проекта.

Хотя выполнение перечисленных вычислений не гарантирует обнаружения всех ошибок проектировщика, связанных с временными процессами в ЦУ, оно существенно уменьшает число таких ошибок или, как минимум, позволяет обнаружить в проекте места, опасные с точки зрения сбоев.

5. **Организация натурных экспериментов.** Последний этап проектирования — экспериментальная проверка спроектированного устройства. При всей тщательности выполнения предыдущих этапов всегда существует далеко не нулевая вероятность того, что в проекте имеются дефекты, которые могут проявиться на этапе внедрения или даже штатного использования устройства и повлечь за собою нежелательные последствия.

Выполнение натурных экспериментов существенно увеличивает вероятность выпуска бездефектной продукции. Средства ускорения работ на этом этапе и возможности их переноса на ранние этапы разработки, т. е. до завершения изготовления конечного продукта, были рассмотрены ранее — это прототипные системы и средства проведения экспериментов с ними. *Прототипные платы* широко использовались и ранее, в частности, при создании микропроцессорных систем. Аналогична и ситуация при разработке систем и устройств на основе программируемой логики. Широкий спектр прототипных плат, содержащих микросхемы программируемой логики и дополнительную аппаратуру (прежде всего микросхемы быстродействующих ОЗУ), выпускается различными зарубежными фирмами. Здесь можно указать средства фирм Altera (Demo Board); PLD Applications (платы PCI Bus Evaluation Board); Xilinx, Virtual Computer Corp., Video Software (платы HOT PCI Design Kit) и др.

Как отмечалось ранее, определенная избыточность, заложенная разработчиком в БИС конечного продукта, позволяет использовать методы внутрикристальной отладки. В этих случаях в САПР вводятся такие функции, выполнение которых способствует упрощению процедуры отладки готового проекта. Например, в САПР Quartus фирмы Altera предусматривается наличие всех трех необходимых составляющих такой процедуры:

- отладочных средств, помещаемых в отлаживаемую СБИС;
- информационно-транспортных средств, связывающих отлаживаемую СБИС с САПР инструментального компьютера;

- программных средств в составе САПР, которые управляют отладкой и отображают ее результаты.

Спектр средств отладки САПР Quartus на настоящий момент (версия Quartus II 7.0) включает три основных компонента: SignalProbe, SignalTap Logic Analysis, мегафункция "sld_virtual_jtag".

САПР фирмы Quartus II поддерживает работу специального отладочного средства — SignalProbe. Средство служит для передачи наблюдаемых сигналов на внешние контакты. Смысл введения такой функции состоит в отсутствии ее влияния на отлаживаемый проект, малое время компиляции для добавляемых сигналов наблюдения, возможность введения специальных конвейерных регистров для фиксации значений наблюдаемых сигналов.

В САПР также встроены необходимые средства имплементации и работы со встраиваемыми логическими анализаторами Signal Tap II. SignalTap Logic Analysis позволяет после загрузки модифицированной версии конфигурационного файла (проекта с отладочными фрагментами) в реальном масштабе времени регистрировать состояния не только на контактах ПЛИС, но и в ее произвольных внутренних точках. Занесение этой информации в память встроенного анализатора и передача сохраненной информации в компьютер с помощью интерфейса JTAG позволяют отображать эту информацию в редакторе временных диаграмм SignalTap. Эти же средства при необходимости дают возможность подключения через имплементируемые интерфейсные средства внешних логических анализаторов (Logic Analyzer Interface, LAI).

Для создания разработчиком внутри отлаживаемой ИС отладочной инфраструктуры фирма в 2006 году выпустила специальную мегафункцию "sld_virtual_jtag", опирающуюся в своей работе на традиционные средства JTAG-интерфейса и существенно расширяющую возможности разработчика при отладке своих проектов.

При успешном завершении экспериментальных работ конфигурационные файлы могут использоваться для изготовления требуемых программируемых БИС либо для записи в конфигурационные ПЗУ. Файлы отчетов о результатах компиляции обычно содержат информацию о конкретных данных по монтированию проекта в реальную БИС. Поэтому уже после этапа компиляции проектов возможен переход к разработке печатных плат, являющихся, как правило, конечной продукцией проектирования.

Итерационные возвраты к повторным процедурам компиляции в ходе конструкторско-технологического этапа проектирования возникают в том случае, если целесообразно изменить месторасположение входных и/или выходных контактов БИС ПЛ, исходя из соображений как более эффективной разводки межсоединений БИС на печатной плате, так и их подключения к выходным разъемам платы. Подобная возможность следует из способности современных БИС ПЛ обеспечивать различные варианты монтирования одного и того же проекта в одну и ту же БИС.

§ 12.6. Основные сведения о языке VHDL

Остановимся на некоторых вопросах, относящихся к наиболее известному языку проектирования аппаратных средств — *VHDL* (Very-High-Speed Hardware Description Language), который использован далее при рассмотрении примера проектирования цифрового устройства средствами САПР.

Язык VHDL появился в начале 80-х гг. по запросам организаций Министерства обороны США. Первая его версия, предназначенная в основном для унификации описаний проектов в различных ведомствах, была принята в 1985 г. В 1987 г. язык VHDL был принят международным институтом IEEE (Institute of Electrical and Electronic Engineers) как стандарт VHDL-87. Он использовался, главным образом, для описания (спецификации) уже спроектированных систем. Использование для задач синтеза устройств (работа с компиляторами) началось с 1991 г. В 1993 г. IEEE принимает новый расширенный стандарт VHDL-93. С тех пор регулярно принимаются изменения стандарта языка, хотя основные концепции остаются неизменными.

Назначение и возможности языка

Язык может быть использован для проектирования ЦУ разных иерархических уровней — от вентильного уровня представления схем до уровня системы в целом. В настоящее время он является, видимо, самым популярным среди проектировщиков цифровой аппаратуры. Сравнимым по популярности является язык Verilog, и практически любая современная САПР средств ВТ или цифровых устройств имеет в своем составе компиляторы с этими языками (как входными, так и выходными).

Проблемная ориентация языка VHDL со временем трансформировалась. Вместо спецификации аппаратуры его первоочередной задачей стало описание структуры и/или поведения цифровых устройств и систем для их синтеза и/или моделирования и лишь затем подробное описание проекта. В соответствии с назначением, язык приспособлен для описания систем как с точки зрения их структурной организации (из модулей с известным поведением), так и с точки зрения поведения либо системы в целом, либо ее составных частей. Наибольшие ограничения на набор допустимых (относительно стандарта) операторов языка имеют компиляторы для синтеза спроектированных устройств, значительно меньше ограничений существует у систем моделирования. Все множество операторов языка может использоваться для описания (спецификации) аппаратуры.

Ориентация на применение языка VHDL в качестве рабочего инструмента для описания структуры и/или поведения широкого класса цифровых устройств заставила сделать его проблемно-ориентированным языком.

Проблемно-ориентированными и поэтому *наиболее важными средствами и понятиями языка VHDL являются*:

- средства задания и описания параллелизма для выполняемых действий и операторов;

- понятие сигнала для физических объектов, имеющих временное измерение для своих значений и средства для работы с ними;
- средства описания иерархии проекта для описания структуры и/или поведения отдельных объектов проекта.

Наиболее важным свойством языка VHDL является понятие параллелизма выполнения действий. Параллелизм начинается с введения понятия параллельно выполняемых операторов — процессов (Process), имеющих различную функциональную направленность и соответственно различные синтаксические формы. К процессам относится большинство традиционно последовательных операторов, таких как вызов процедуры и оператор присвоения. Для управления параллелизмом естественно введение соответствующих управляющих операторов. Возможность различными способами описать одну и ту же систему или объект, оставаясь в рамках одного структурного элемента языка, приводит к понятию стиля описания (программирования).

Второй важнейшей особенностью языка VHDL является введение физического типа данных. Понятие *сигнала* (`SIGNAL`) отражает основные свойства реальных входных и выходных данных проекта. Среди различных свойств сигналов важнейшими представляются их временные характеристики и, прежде всего, наличие у них прошлого, настоящего и будущего состояний. Специфические свойства сигналов потребовали введения оператора *назначение значения сигнала* (<=), основное отличие которого от оператора *присвоение значения переменной* (:=) состоит в задержке изменения текущего состояния сигнала до тех пор, пока не будут подготовлены результаты преобразования во всех одновременно инициированных процессах, и лишь после этого одновременно все сигналы могут приобретать новые значения.

Иерархическое построение описания системы в языке VHDL является развитием традиционного иерархического подхода и отличается тем, что распространяется не только на описание поведения, но и на описание структуры системы. Архитектурное тело (Architecture Body) — описывает поведение объекта или его структуру. Внутри архитектурного тела может быть и смесь структурного описания с поведенческим. То, что описанию архитектуры предшествует описание интерфейса объекта (`ENTITY`), не является существенным отличием языка VHDL и аналогично (в определенном смысле) описанию прототипа в языке СИ.

Основные понятия и синтаксические конструкции языка

Многие термины и понятия языка здесь и далее не затрагиваются, поскольку цель более или менее серьезного изучения может ставиться лишь в работах достаточно большого объема.

Целевая направленность языка — необходимость адекватного описания структур с параллельным поведением отдельных элементов даже для целей синтеза требует создания моделей. Поэтому программа может состоять только из параллельно ис-

полняемых операторов (как правило, последовательности). В отличие от обычных языков программирования, конечный результат исполнения фрагмента типа:

```
A <= B AND C;
B <= C OR A;
D <= B XOR A;
```

не должен зависеть от порядка просмотра (исполнения) операторов. Чтобы достичь подобной независимости, обмен информацией между параллельными операторами осуществляется с помощью специфических элементов языка — сигналов (**SIGNAL**). Сигналы в соответствии со своим типом (например, бит, целое, массив и т. д.) могут хранить данные, соответствующие, этому типу, но в отличие от обычных переменных, они хранят значения, соответствующие двум состояниям — текущему (*current*) и следующему (*next*). Следующее значение устанавливается в момент просмотра оператора, а замена текущего состояния следующим (для устранения коллизий не одновременности вычисления состояний различных фрагментов схемы) выполняется только после просмотра всех требующих учета операторов. Если новые значения сигналов могут влиять на систему и изменять ее состояние, просмотр операторов повторяется, и так до тех пор, пока в системе не будет достигнуто стабильное состояние. Параллельные операторы могут находиться в одном из трех состояний: выполненном, готовом (еще не проанализированном в текущем просмотре) или в ждущем (ожидающими изменения сигнала или времени). При синтезе параллельным операторам соответствуют соединения одновременно (параллельно) работающих блоков или фрагментов.

Основным оператором, описывающим функционирование, является оператор параллельного назначения сигнала (<=) с двумя условными вариантами, соответствующими конструкциям **IF** (**WHEN...ELSE**) и **CASE** (**WITH...SELECT**). Для более компактной записи сложных фрагментов язык вводит понятия блока (**BLOCK**) и вызова параллельной процедуры.

Помимо поведенческого описания язык VHDL поддерживает структурное и комбинированное (структурное плюс поведенческое) описания. Основным способом построения структуры является включение компонента в проект, которое осуществляется специальным параллельным оператором — вхождение компонента. Синтаксическая конструкция (**PORT MAP**) задает интерфейс структурного компонента, а предшествующая ей опционная конструкция (**GENERIC MAP**) может задавать настраиваемые свойства и параметры компонента. Соединения выполняются при помощи подключения сигналов к контактам компонентного порта. Сигналы в языке выполняют двойственную роль. С одной стороны, как структура данных для описания поведения, а с другой стороны, как имя цепи структурного соединения, по которой могут передаваться такие данные. Благодаря дуальности сигналов легко образуются смешанные комбинации (структурно поведенческие). Как и в классических языках программирования, первое же включение нового структурного компонента требует предварительного описания свойств интерфейса (синтаксическая конструкция — **COMPONENT...PORT**<интерфейс>), а в дальнейшем и конкретизацию структуры и\или поведения этого компонента.

Условные и циклические соединения компонентов между собой (или просто их тиражирование) достигаются применением операторов генерации фрагментов структуры (**IF**...**GENERATE** и **FOR**...**GENERATE**).

Конструкцией, объединяющей в себе свойства традиционного составного оператора и оператора описания компонентов, является конструкция блок — **BLOCK**. Как и в традиционных языках, группа параллельных операторов для удобства использования может объединяться в единую конструкцию (в языке введены понятия *описания параллельной процедуры* и *обращения к параллельной процедуре*).

Для определения систем, функционирующих в соответствии с алгоритмами, в которых временное изменение не является определяющим, возможно использование последовательных операторов, оперирующих с обычными переменными и помещенных в параллельный оператор **PROCESS**. Набор последовательных операторов языка достаточно традиционен и содержит как традиционные операторы действия (присвоения значения традиционной переменной (:=), условия (**IF**), выбора (**CASE**), цикла (**LOOP**), вызова последовательной процедуры), так и традиционные типы данных: числовые, логические, символьные, перечислительные и агрегатированные (массивы, записи и файлы). Не самым распространенным можно считать лишь набор ключевых слов и синтаксических правил составления предложений. Все синтаксические конструкции языка построены с использованием специальных разделительных и завершающих ключевых слов (типа **END IF**, **ELSIF**, **END LOOP**, **THEN**, **END PROCESS**, **END BLOCK** и т. д.), что позволяет объединять группы параллельных или последовательных операторов без использования понятия составной оператор. Применение последовательных операторов **NEXT**, **EXIT**, **RETURN** и **NULL** не отличается от вариантов традиционных языков и не требуют специальных пояснений. Для связи алгоритмов последовательных структур с параллельными операторами так же используются сигналы. Оператор последовательного назначения сигнала синтаксически очень похож на параллельный и отличается в отдельных деталях. Следует особо подчеркнуть, что составляющая сигнала *current* изменяется в момент исполнения последовательного оператора, но перезапись *current* в *next* осуществляется по стандартным правилам работы совокупности параллельных операторов.

Текущее исполнение процесса выполняется до тех пор, пока не будет достигнут его последний оператор или пока не встретится последовательный оператор **WAIT** в одной из своих синтаксических форм. Формы отличаются моментом запуска от текущего момента (абсолютных — **WAIT FOR** <время>, относительных — **AFTER** <время>), и операторов, задерживающих момент запуска до выполнения заданного условия (**WAIN UNTIL** <условия>, **WAIN ON** <сигналы>).

Активизация оператора **PROCESS** от точки ожидания (от начала процесса или оператора ожидания) со стороны других параллельных операторов происходит при выполнении условия запуска: изменении условия запуска или достижения заданного времени.

Описание проекта на языке VHDL

Описание проекта на языке VHDL имеет типовую структуру: в его начале указываются библиотеки функциональных элементов, которыми может пользоваться САПР (Library Declaration), далее следуют описание объектов (Entity Declaration), которые будут использованы как компоненты проектируемого устройства, и раздел архитектуры (Architecture Declaration), который может быть представлен в структурном, поведенческом или смешанном вариантах.

При составлении программ на языке VHDL используются следующие термины и понятия. Все проекты выражаются в терминах объектов проекта — **ENTITY**. Каждый объект проекта имеет объявление интерфейса объекта — Entity Declaration и описание архитектурного тела объекта — Architecture body. **ENTITY** содержит имя объекта и его интерфейс (входы и выходы). **ARCHITECTURE** содержит описание структуры или поведения объекта. Верхний уровень проекта описывается через объекты верхнего уровня, если устройство иерархично, то описания объектов верхнего уровня содержат в себе обращения к компонентам более низкого уровня, которые описываются как самостоятельные объекты нижнего уровня. В свою очередь, объекты нижнего уровня могут связываться с объектами еще более низкого уровня. Для определенности функционирования системы независимо от числа уровней иерархии все объекты нижних уровней иерархии должны иметь описание, определяющее их функционирование. Один и тот же объект может иметь несколько архитектурных тел (естественно, что при моделировании поведения системы или при ее синтезе специальные средства конфигурирования (Configuration Declaration) определяют единственный вариант поведения).

Примеры описаний элементов на языке VHDL

Проиллюстрируем поведенческий вариант описания на простейшем примере. Пусть требуется описать на языке VHDL логическое устройство, определяющее во входном восьмиразрядном коде место первой логической единицы, под наименованием first_one. Начиная с раздела Entity Declaration, описание может иметь вид, показанный в листинге 12.1.

Листинг 12.1

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
ENTITY first_one IS -- entity declaration
 PORT (       -- port statement
       A: IN Std_logic_vector(7 DOWNTO 0);
       Z: OUT integer   RANGE 0 TO 8);
 END first_one;
```

```
ARCHITECTURE one OF first_one IS
-- architecture "one" of entity
BEGIN
       Main_pr: PROCESS (A)
       VARIABLE i : integer;
       BEGIN
              Z <= 8; -- если нет ни одной единицы
              FOR  i IN 7 DOWNTO 0 LOOP
                     IF (A(i) ='1') THEN
                            Z <= i;
                            EXIT;
                     END IF;
              END LOOP;
       END PROCESS;
END;
```

Проиллюстрируем структурный вариант описания на простейшем примере. Пусть требуется описать на языке VHDL логическое устройство, определяющее функцию z, равную (a **AND** b) **OR** (c **AND** d) под наименованием f_structure. Структура фрагмента имеет вид, приведенный на рис. 12.8. Начиная с раздела Entity Declaration, описание головного модуля (верхний уровень иерархии) может иметь вид, приведенный в листинге 12.2.

Листинг 12.2

```
ENTITY f_structure IS -- entity declaration
PORT (        -- port statement
       a, b, c, d     : IN std_logic;
       z              : OUT std_logic);
END f_structure;
ARCHITECTURE two OF f_structure IS
-- architecture "two" of entity
-- интерфейс компонентов AND
COMPONENT AND_My
PORT (
       in1, in2       : IN std_logic;
       output         : OUT std_logic
       );
```

```
END COMPONENT AND_My;
-- интерфейс компонента OR
COMPONENT OR_My
PORT (
       in1, in2       : IN std_logic;
       output         : OUT std_logic
       );
END COMPONENT OR_My;
-- внутренние цепи проекта
SIGNAL net1, net2;
BEGIN
       U1: AND_My PORT MAP(a, b, net1);
       U2: AND_My PORT MAP(c, d, net2);
       U3: OR_My PORT MAP(net1, net2, z);
END;
```

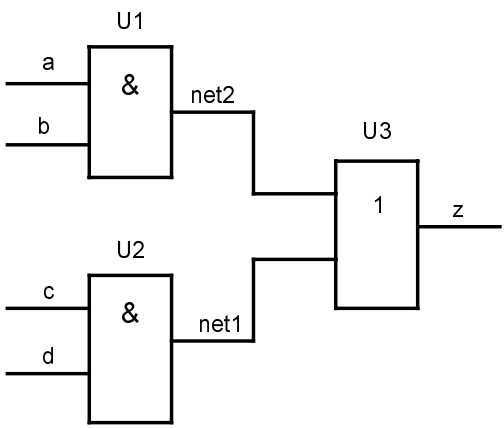


Рис. 12.8. Структура комбинационной схемы f_structure

Естественно, на нижних уровнях иерархии должно описываться функционирование используемых компонентов. Для модуля AND_My, начиная с раздела Entity Declaration, описание может выглядеть так, как показано в листинге 12.3.

Листинг 12.3

```
ENTITY AND_My IS -- entity declaration
 PORT (in1, in2       : IN std_logic;
       output         : OUT std_logic);
END AND_My;
```

```
ARCHITECTURE three OF AND_My IS
BEGIN
        Output <= '1' WHEN  in1='1' AND in2='1' ELSE '0';
END;
```

Структурное и поведенческое описание проекта

Когда следует использовать структурный, а когда поведенческий вариант описания проекта?

Структурный вариант предусматривает перечисление как типов компонентов и их интерфейса (их выводов), так и связей всех компонентов между собой, тем самым непосредственно отражая задаваемую для реализации схему, которая и будет создана в выбранной СБИС ПЛ.

Поведенческий вариант определяет функции, которые должны быть реализованы, но не говорит о том, каким именно способом это должно быть сделано. Так как схемотехнические реализации узлов и устройств практически всегда многовариантны, САПР получает для их реализации определенную свободу действий. Большое достоинство поведенческого варианта — его компактность и наглядное представление функционирования устройства, что хорошо видно хотя бы из приведенных примеров. Платой за это является ослабление контроля за способом реализации проектируемого устройства или его фрагмента, поскольку это отдается "на усмотрение" компилятора. При этом следует ожидать, что компилятор может принять реализацию схемы, которая не будет столь же эффективна по быстродействию и затратам ресурсов, как выполненная квалифицированным специалистом, хорошо знающим ресурсы и особенности структурной организации выбранной СБИС ПЛ. Поэтому чаще всего комбинируют использование структурных и поведенческих описаний в рамках одного и того же проекта. Для критичных по скорости фрагментов целесообразно использовать структурные описания. Например, задавая функции счета или суммирования, реализуют структуры с параллельными переносами, обеспечивающими наибольшее быстродействие, тогда как компилятор (при не установленных специальных опциях), возможно, создаст более простые структуры с последовательными переносами. В то же время остальные части проекта целесообразно описать в поведенческом варианте, что существенно упрощает задачу.

Язык VHDL для моделирования и синтеза

Помимо стандарта языка VHDL, определенного для моделирования, в 1999 г. был принят стандарт IEEE Std 1076.6, ориентированный на синтез — IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis. Фирмы-разработчики компиляторов с языка VHDL (несмотря на близость решений) трактуют синтаксис языка со зна-

чительными расхождениями. Совершенно естественно желание проектировщиков использовать одно и то же языковое описание сначала для моделирования проектируемой системы, а затем и для ее синтеза. Однако подобный подход (при расхождениях в трактовке стандарта) должен использоваться с осторожностью.

В современных системах моделирования удается реализовать почти полный стандарт языка, значительно меньшие возможности допускают синтезирующие средства. Вопрос не только в допустимых для использования в том и другом вариантах синтаксических конструкциях, но и в совпадении свойств модели и синтезированного устройства. Существующие сегодня САПР позволяют автоматизировать преобразование проектов, описанных на уровне регистровых передач (RTL — Register Transfer Level), в описание на уровне вентильных межсоединений. Конкретные реализации ячеек вентильного уровня, включая их макроблоки, содержатся в технологических библиотеках целевой ИС проекта.

Многие синтаксические конструкции языка ориентированы только на моделирование и игнорируются при компиляции. К игнорируемым относится группа конструкций, связанная с понятием модельного времени, например, выражения для установления временных задержек (**AFTER** <время>) в операторах сигнального присваивания и в операторах временного управления (**WAIT FOR** <время>). Другой игнорируемой группой являются операторы проверки (**ASSERT...**), обеспечивающие вывод на консоль информации о специфической ситуации при моделировании.

Результаты моделирования могут отличаться от результатов, получаемых при работе синтезированного по этому же описанию устройства. Различие, прежде всего, обусловлено замещением одиночной, но достаточно емкой синтаксической конструкции проекта на соединение ряда элементов используемой технологической библиотеки. Наличие у этих элементов определенных временных характеристик и многовариантность подобного замещения может приводить не только к расхождению по времени, но и появлению дополнительных (ложных) изменений выходных и промежуточных сигналов.

Различия поведения модели и синтезированного устройства могут быть связаны и с определенной трактовкой компилятором некоторых синтаксических конструкций языка. Например, при моделировании фрагмента схемы, описываемого процессом, его инициализация определяется составом списка чувствительности процесса, а при синтезе в качестве входных воздействий синтезируемого фрагмента выбираются все сигналы, определяющие поведение процесса. Все обращения к функциям и подпрограммам замещаются на встроенные экземпляры схемной реализации тел этих операторов.

О возможностях и средствах описания типовых узлов цифровой техники

Весьма существенным вопросом, интересующим большинство разработчиков, является наличие в языке возможностей и/или средств описания типовых узлов и

устройств цифровой техники. В соответствии с традиционным разбиением, необходимо оценить возможности описания следующих типов узлов: *комбинационных схем, регистровых схем и цифровых автоматов.*

Комбинационные схемы. Функционирование комбинационных схем можно описывать как в форме последовательных операторов в операторе **PROCESS**, так и в форме параллельных операторов. В операторе **PROCESS** допустимо применение достаточно широкого класса средств: арифметических и логических выражений, условных или селективных (по выбору) назначений сигнала. В разделе последовательных операторов допустимо использование операторов условия (**IF**) и выбора (**CASE**). Входы и выходы схемы могут быть представлены в виде сигнала или переменной. Как правило, используются следующие типы данных: bit, bit_vector, std_logic, std_logic_vector. Использование других типов данных может требовать определения функций взаимных преобразований.

Во избежание появления нежелательных дополнительных элементов для описания комбинационных схем целесообразно применять параллельные операторы. Для удобства моделирования и синтеза в качестве типов данных лучше использовать данные типа std_logic (с представлением, соответствующим девятисимвольному алфавиту). Пример образования схемы, реализующий функцию $Z = (a \vee b)c$ под наименованием input3_orand1, в языке VHDL, начиная с раздела Entity Declaration, может иметь вид, показанный в листинге 12.4.

Листинг 12.4

```
ENTITY input3_orand1 IS -- entity declaration
PORT (a,b,c    : IN std_logic; -- port statement
       Z       : OUT std_logic);
END input3_orand1;

ARCHITECTURE comb OF input3_orand1 IS
-- architecture "comb" of entity input3_orand1
BEGIN
       Z <= '1' WHEN (a='1' OR b='1') AND (c ='1') ELSE '0';
END;
```

Схемы с памятью. Схемы с памятью (триггеры, регистры, счетчики и т. д.) являются важнейшим представителем типовых средств цифровой техники. Простейшими элементами с памятью являются триггеры. Поведение триггера можно описать, опираясь как на параллельные, так и на последовательные операторы. В качестве примера рассмотрим описание поведения триггера RS-триггера с динамическим управлением, имеющего вход асинхронного сброса (листинг 12.5).

Листинг 12.5

```
ENTITY RS_ff IS -- entity declaration
PORT (         -- port statement
       R,S             : IN std_logic;
       Clk, Reset      : IN std_logic;
       Q               : BUFFER std_logic_vector(3 DOWNTO 0)
       );
END RS_ff;

ARCHITECTURE four OF RS_ff IS
-- architecture " four" of entity RS_ff
SIGNAL Q_int:  std_logic;
BEGIN
       --  комбинационная часть элемента
       Q_int <= '0' WHEN R ='1' ELSE '1' WHEN S ='1' ELSE Q;

       --  собственно элемент памяти с динамическим управлением
       Q <= '0' WHEN Reset = '1' ELSE Q_int WHEN Clk'event AND
              Clk='1' ELSE Q;
END four;
```

Несмотря на синтаксическую правильность фрагмента, чаще используется процессорное представление тактирующей части регистровых схем (второй оператор в программе). Выбор процессорного представления диктуется его адекватной трактовкой практически любой компилирующей или моделирующей САПР. При процессном представлении, внутри оператора **PROCESS** обычно пользуются операторами условия (**IF**) и выбора (**CASE**). В качестве примера рассмотрим архитектурное тело при процессном описании поведения RS-триггера (листинг 12.6).

Листинг 12.6

```
ARCHITECTURE five OF RS_FF IS
-- architecture " four" of entity RS_ff
SIGNAL Q_int : std_logic;
BEGIN
       base_ff: PROCESS (Clk, Reset, Q_int)
       BEGIN
              IF Reset ='1' THEN
                     Q<= '0' ;
```

```
            ELSIF Clk'event AND Clk = '1' THEN
                  Q <= Q_int;
            ELSE
                  Q<= Q;
            END IF;
      END PROCESS base_ff;

      Q_int <= '0' WHEN R='1' ELSE '1' WHEN S='1' ELSE Q;

END five;
```

Описание многоразрядных схем с памятью (регистров) практически не отличается от описания одиночных триггеров.

При создании регистровых схем необходимо уделять внимание наличию или отсутствию синхронизации входных сигналов и ее типу (асинхронная схема, статическое или динамическое тактирование) и способам задания входных, выходных сигналов и внутренних состояний при описании многоразрядных схем (счетчиков, регистров сдвига и т. д.). Для определения этих переменных или сигналов в многоразрядных схемах можно использовать понятие вектора или ограниченного целого (с диапазоном изменения, связанным с разрядностью схемы).

Целесообразным вариантом следует считать построение описания в форме двух параллельных операторов. Первый оператор — оператор **PROCESS**, описывающий собственно многоразрядную память. Память имеет информационные входы от комбинационной схемы, а также при необходимости может содержать цепи асинхронного сброса и предустановки. Второй параллельный оператор — условный оператор назначения сигнала, реализующий требуемое функционирование регистровой схемы. В качестве примера рассмотрим 4-разрядный регистр сдвига влево, вправо с занесением начального значения, имеющий вход асинхронного сброса (листинг 12.7).

Листинг 12.7

```
ENTITY Reg_Shift IS -- entity declaration
PORT (        -- port statement
      Clk, Reset           : IN std_logic;
      Load, Left, Right    : IN std_logic;
      D_in                 : IN std_logic_vector(3 DOWNTO 0);
      Q                    : BUFFER std_logic_vector(3 DOWNTO 0)
      );
END Reg_Shift;
```

```
ARCHITECTURE six OF Reg_Shift IS
-- architecture "six" of entity Reg_Shift
SIGNAL Q_int  : std_logic_vector(3 DOWNTO 0);
BEGIN
       base: PROCESS (Clk, Reset, Q_int)
       BEGIN
              IF Reset ='1' THEN
                     Q<= "0000";
              ELSIF Clk'event AND Clk = '1'THEN
                     Q <= Q_int;
              ELSE
                     Q<= Q;
              END IF;
       END PROCESS base;

       Q_int <= D_in WHEN Load= '1' ELSE '1'& Q(3 DOWNTO 1) WHEN
              Right ='1' ELSE Q(2 DOWNTO 0) & '1'  WHEN Left='1' ELSE  Q;

END six;

ENTITY Register IS -- entity declaration
 PORT (       -- port statement
Clk, Reset           : IN std_logic;
Load, Left, Right    : IN std_logic;
D_in                 : IN std_logic_vector(3 DOWNTO 0);
Q                    :BUFFER std_logic_vector(3 DOWNTO 0)
);
END Register;
```

Цифровые автоматы. Следующими типовыми фрагментами, играющими очень важную роль в проектировании, являются цифровые автоматы. Основные разновидности поведения автоматов сводятся к автоматам Мили или Мура, хотя расширенные структурные возможности схем ПЛИС привели к широкому практическому распространению разновидности автоматов Мили — автоматам Мили с синхронными выходами.

Выделение автоматов в самостоятельный класс типовых фрагментов связано не только с удобством разбиения (для разработчика) проектируемых устройств на операционные блоки и управляющий автомат (как описано ранее), но и со специфическим подходом компиляторов к оптимизации синтезируемой структуры. Из

двух видов автоматов (асинхронных и синхронных) наибольшее распространение получили последние. Хотя этот тип автоматов проигрывает в быстродействии, его использование в проектировании значительно проще.

Каноническое разбиение поведения автоматов сводится к двум разновидностям (автоматам Мили и Мура). Внешне структурное отличие между ними (рис. 12.9, *а* и *б*) сводится к отсутствию у автоматов Мили (рис. 12.9, *б*) подключения входных сигналов (X на рисунке) к блоку комбинационной логики, определяющей значения выходных сигналов (Y).

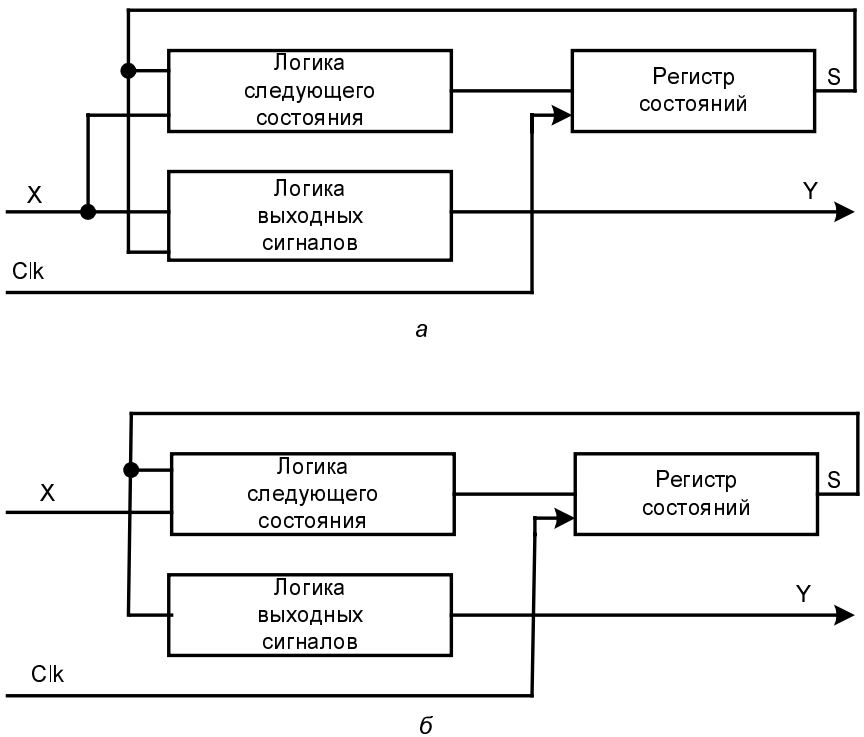


Рис. 12.9. Функциональная схема автоматов Мура (*а*) и Мили (*б*)

Базовым элементом автоматов является регистр состояния (не путать с термином "регистр состояния" для периферийных БИС, микропроцессоров и т. п.). Выходные сигналы (S) триггеров этого регистра определяют как следующее состояние автомата (устанавливаемое по фронту тактового сигнала Clk), так и значения выходных сигналов.

В условиях автоматического синтеза автомата на вентильном уровне достаточно часто могут появиться проблемы с использованием выходных сигналов. В моменты смены состояний автомата или изменения значений входных сигналов могут возникать паразитные импульсы, обусловленные рисками, появляющимися в комбинационных схемах выработки выходных сигналов. Поэтому выходные сигналы автомата опасно использовать в качестве тактовых сигналов или сигналов асин-

хронных сбросов и установок триггеров. Для устранения указанного эффекта используются автоматы с синхронными выходами (рис. 12.10). На рисунке Y_C — комбинационные сигналы, а Y_R — синхронизируемые тактовым сигналом. При подключении триггеров к выходам автоматов необходимо учитывать, что выходные сигналы триггеров начинают как бы "запаздывать" на один такт относительно сигналов входной комбинационной логики. Другими словами, логика управления выходными триггерами автомата должна формировать значения "следующего" (или будущего) состояния сигнала.

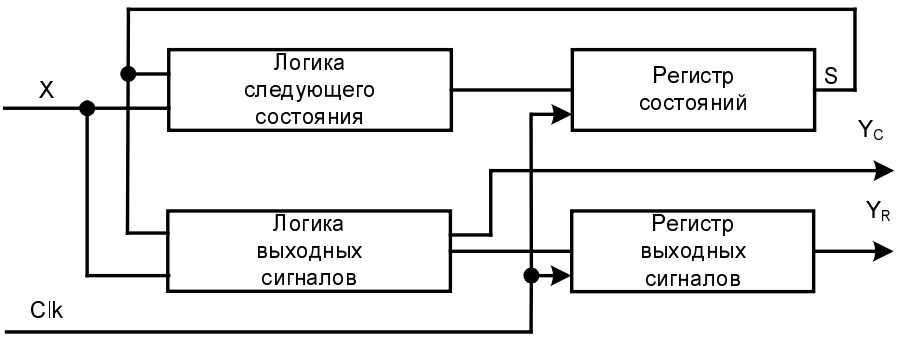


Рис. 12.10. Функциональная схема автомата с синхронными выходами

Другая проблема — возможность появления *временных перекосов* в блоках комбинационной логики автомата. Это может происходить вследствие асинхронности входных сигналов относительно тактовой частоты автомата. Когда изменение входного сигнала происходит в момент времени, близкий к фронту тактового сигнала, в логике, управляющей следующим состоянием, появляется вероятность перевода автомата либо в неправильное состояние, либо неопределенное.

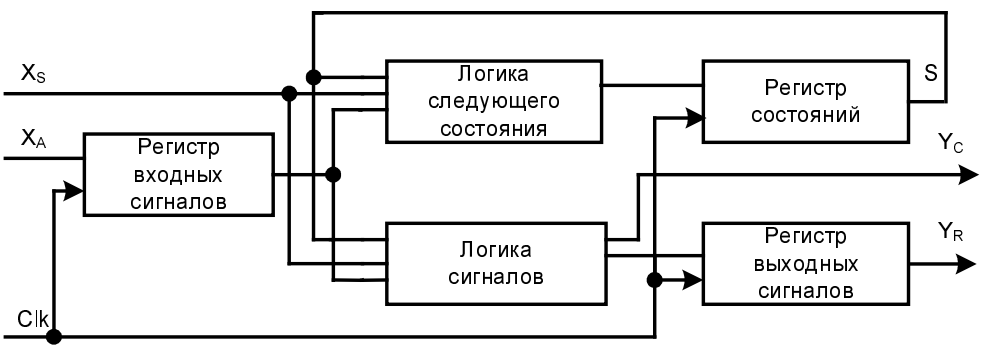


Рис. 12.11. Функциональная схема автомата с синхронизацией входов

В этой ситуации и в логике формирования выходных сигналов могут возникать ложные импульсы. И то, и другое чаще всего приводит к катастрофическим по-

следствиям для работы автомата. Модификация схемы автомата, снижающая вероятность появления сбоев, приведена на рис. 12.11. На рисунке: X_S — синхронные входные сигналы, а X_A — асинхронные.

Классическим описанием автоматов является табличное задание переходов и выходных сигналов. В клетках таблицы переходов записаны состояния, в которые переходит автомат из исходного состояния при соответствующих условиях, а в клетках таблицы выходов — выходные сигналы при тех или иных условиях. Достоинство такого способа определения автоматов — легкость контроля над пропуском неопределенных переходов или условий формирования выходных сигналов.

Наглядно отражает алгоритмы управления и поэтому получает все большее распространение задание автомата в форме *графа переходов*. Для упрощения их создания в состав все большего числа САПР вводят графические редакторы диаграмм состояний. Несмотря на некоторые различия в формальном способе конкретизации, все такие редакторы позволяют задать желательные структуру автомата и способ формирования выходных сигналов.

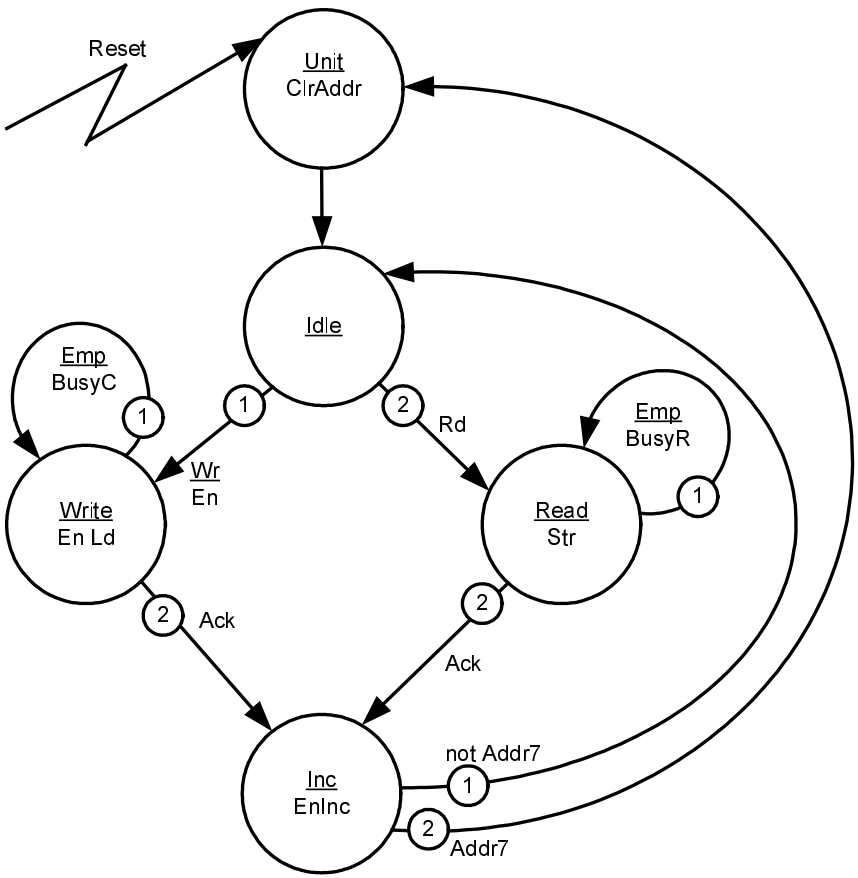


Рис. 12.12. Пример задания автомата в форме диаграммы переходов

В качестве примера описания рассмотрим автомат, диаграмма состояний которого приведена на рис. 12.12. В автомате под именем состояния перечисляются имена сигналов, активизируемых в этом (или этим) состоянием. Например, сигналы En и Ld в состоянии Write. Если из состояния возможен переход в несколько других состояний, то на дугах переходов указываются логические выражения, составленные из имен входных сигналов, определяющих условия переходов. Если из текущего состояния переходы являются условными, то обычно предполагается, что альтернативой является остановка в текущем состоянии. В примере, это сигналы Ack, Rd и Wr. В кружочках на дугах (или рядом) указывается приоритет переходов при одновременности выполнения условий для нескольких переходов. Кроме указания выражения, определяющего переход, на дуге может содержаться информация о сигналах, возникновение которых обусловлено выполнением условия этого выражения. Такие сигналы в примере — Busy. В примере два выходных сигнала BusyC и BusyR. Первый из них соответствует комбинационному типу, а второй — регистровому (синхронизируемого тактовым сигналом).

Поэтому комбинационный сигнал повторяет на выходе автомата во времени возникновение и исчезновение условия с задержкой, соответствующей задержке комбинационной схемы. А синхронизированный сигнал выставляется и снимается на выходе по условию его формирования в моменты тактирования. Из этого следует, что длительность первого сигнала совпадает с длительностью входного сигнала, а второго оказывается зависящей от условий его формирования и кратной периоду тактирования.

Для правильной совместной работы автоматов и триггерных схем, имеющих схемы пропускания входных сигналов (управляемые сигналом разрешения) и тактирующие входы (синхронные с управляющим автоматом), оказывается важным иметь не нулевым *время предустановки* (опережение сигналом разрешения сигнала тактирования). В примере такой парой являются сигналы En и Ld. Для обеспечения необходимого опережения сигнал En присутствует и на дуге перехода (комбинационный сигнал, возникающий почти сразу после появления Wr на входе автомата), и в списке сигналов инициируемых состоянием, т. е. синхронизируемых тактовыми импульсами. Объединение сигналов по логике ИЛИ дает сигнал En, обладающий необходимыми свойствами предустановки.

Возможность использования в языке перечислительного типа данных позволяет ввести перечислительные типы данных "состояния", а при желании разработчика и типы данных "входы" и "выходы". Предпочтительна процессная форма описания поведения автомата. В зависимости от желания разработчика и типа автомата, его архитектура может включать от одного до пяти процессов. Оператор варианта целесообразно использовать в качестве основного каркаса процессов, для чего каждому состоянию автомата назначается вариант в операторе выбора. Для описания альтернативных вариантов формирования переходов и выходных сигналов (если это необходимо) обычно применяется условный оператор. Приоритетность переходов соответствует порядку записи условий. Отдельный процесс может вводиться для описания процедуры тактирования и начальной установки автомата.

Описывающая автомат программа на языке VHDL может иметь вид, приведенный в листинге 12.8. Для создания текста программы использовался графический редактор описания конечных автоматов (в САПР этот тип редакторов обычно носит название editor of FSM — Finite State Machine). В данном случае это была программа HDL Designer (версии 2002.1b) фирмы Mentor Graphics [XVII].

Листинг 12.8

```
-- Generated by Mentor Graphics' HDL Designer(TM) 2002.1b (Build 7)
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY example IS
   PORT(
      Ack     : IN     std_logic;
      Addr7   : IN     std_logic;
      Clk     : IN     std_logic;
      Emp     : IN     std_logic;
      Rd      : IN     std_logic;
      Reset   : IN     std_logic;
      WR      : IN     std_logic;
      BusyC   : OUT    std_logic;
      BusyR   : OUT    std_logic;
      ClrAddr : OUT    std_logic;
      En      : OUT    std_logic;
      EnInc   : OUT    std_logic;
      Ld      : OUT    std_logic;
      Str     : OUT    std_logic;
      StrS    : OUT    std_logic                    -- вставка 1
   );
-- Declarations
END example ;
-- hds interface_end
--
-- VHDL Architecture exam.fsm
-- Generated by Mentor Graphics' HDL Designer(TM) 2002.1b (Build 7)
--
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ARCHITECTURE fsm OF example IS
   -- Architecture Declarations
   TYPE STATE_TYPE IS (
      Unit,      Idle,      Write,      Read,      Inc
                    );
   -- State vector declaration
   ATTRIBUTE state_vector : string;
   ATTRIBUTE state_vector OF fsm : ARCHITECTURE IS "current_state" ;
   -- Declare current and next state signals
   SIGNAL current_state : STATE_TYPE ;
   SIGNAL next_state : STATE_TYPE ;
   -- Declare any pre-registered internal signals
   SIGNAL BusyR_cld : std_logic  ;
   SIGNAL StrS_int : std_logic  ;          -- вставка 2
BEGIN
   ----------------------------------------------------------------
   clocked : PROCESS(  Clk,  Reset   )
   ----------------------------------------------------------------
   BEGIN
      IF (Reset = '1') THEN
         current_state <= Unit;
         -- Reset Values
         BusyR_cld <= '0';
         StrS <= '0';               -- вставка 3
      ELSIF (Clk'EVENT AND Clk = '1') THEN
         current_state <= next_state;
         -- Registered output assignments
         StrS <= StrS_int;                -- вставка 4
         -- Default Assignment To Internals
         BusyR_cld <= '0';
         -- Combined Actions for internal signals only
         CASE current_state IS
         WHEN Read =>
            IF (Emp = '1') THEN
```

```
            BusyR_cld <= '1' ;
         END IF;
      WHEN OTHERS =>
         NULL;
      END CASE;
   END IF;
END PROCESS clocked;
-----------------------------------------------------------------------
nextstate : PROCESS (
   Ack,      Addr7,      Emp,      Rd,      WR,      current_state
)
-----------------------------------------------------------------------
BEGIN
   CASE current_state IS
   WHEN Unit =>
         next_state <= Idle;
   WHEN Idle =>
      IF (WR = '1') THEN
         next_state <= Write;
      ELSIF (Rd = '1') THEN
         next_state <= Read;
      ELSE
         next_state <= Idle;
      END IF;
   WHEN Write =>
      IF (Emp = '1') THEN
         next_state <= Write;
      ELSIF (Ack = '1') THEN
         next_state <= Inc;
      ELSE
         next_state <= Write;
      END IF;
   WHEN Read =>
      IF (Emp = '1') THEN
         next_state <= Read;
      ELSIF (Ack = '1') THEN
         next_state <= Inc;
      ELSE
```

```
            next_state <= Read;
         END IF;
      WHEN Inc =>
         IF (Addr7 /= '1') THEN
            next_state <= Idle;
         ELSIF (Addr7 = '1') THEN
            next_state <= Unit;
         ELSE
            next_state <= Inc;
         END IF;
      WHEN OTHERS =>
         next_state <= Unit;
      END CASE;
END PROCESS nextstate;
-----------------------------------------------------------------------
output : PROCESS (
   Ack, Emp, Rd, WR,      current_state
                  )
-----------------------------------------------------------------------
BEGIN
   -- Default Assignment
   BusyC <= '0';
   ClrAddr <= '0';
   En <= '0';
   EnInc <= '0';
   Ld <= '0';
   Str <= '0';
   -- Default Assignment To Internals
   StrS_int <= '0';                    -- вставка 5
   -- Combined Actions
   CASE current_state IS
   WHEN Unit =>
      ClrAddr <= '1' ;
   WHEN Idle =>
      IF (WR = '1') THEN
         En <= '1' ;
      END IF;
      IF ((Rd = '1') AND (WR = '0')) THEN      -- вставка 6
```

```
            StrS_int <= '1' ;
         END IF;
      WHEN Write =>
         En <= '1' ;
         Ld <= '1' ;
         IF (Emp = '1') THEN
            BusyC <= '1' ;
         END IF;
      WHEN Read =>
         Str <= '1' ;
         IF (Ack='0')OR(Ack='1' AND Emp='1')THEN  -- вставка 7
            StrS_int <= '1' ;
         END IF;
      WHEN Inc =>
         EnInc <= '1' ;
      WHEN OTHERS =>
         NULL;
      END CASE;
   END PROCESS output;
   -- Concurrent Statements
   -- Clocked output assignments
   BusyR <= BusyR_cld;
END fsm;
```

В основе работы автомата лежит использование двух сигналов — current_state и next_state. Первый соответствует выходным сигналам регистра состояния (см. рис. 12.10), а второй — выходным сигналам блока комбинационной логики следующего состояния. Сигналы относятся к объявленному перечислительному типу state_type и заданы списком имен состояний. Объявлен сигнал внутренних цепей, поступающих на вход регистра выходных сигналов. Это сигнал BusyR_cld. Архитектурное тело автомата содержит четыре параллельных оператора: три оператора процесса и один условного присваивания сигнала.

Запуск процесса "Clocked" определяется изменением двух сигналов Reset и Clk. Сигнал сброса устанавливает в исходное состояние все триггеры автомата. Тактовый сигнал обеспечивает перепись в триггеры регистров состояний выходов блоков комбинационной логики (следующего состояния — current_state и выходных сигналов — BusyR_cld). Важно обратить внимание на то, что изменения состояний происходят в момент появления нарастающего фронта сигнала Clk, т. к. запускающее событие определено как "появление единицы и наличие переходного процесса на входе Clk".

Синтаксическая конструкция Clk'event называется *атрибутом сигнала*. Атрибут сигнала может принимать значение "истинно" или "ложно" и характеризует некоторые свойства сигнала на момент моделирования (в данном контексте — переходный режим).

Использование выражения "Clk'event" соответствует реальной структуре устройства, реализующего автомат, в котором состояние отображается состоянием регистра. Так как этот регистр является одновременно датчиком информации о текущем состоянии и приемником нового значения, во избежание гонок необходимо использовать регистры с динамическим управлением, реагирующие на изменение сигнала, что и задается используемой конструкцией условия.

Следующий процесс, имеющий имя "nextstate", описывает требуемое поведение комбинационной логики следующего состояния для всех переходов автомата. Приоритет выполнения условных переходов задается последовательностью записи условий. Список чувствительности процесса содержит перечисление сигналов, определяющих работу логики, включая текущее состояние автомата — current_state.

Процесс с именем "output" описывает поведение блока комбинационной логики выходных сигналов в зависимости от текущего состояния автомата. Как и в предыдущем описании, список чувствительности содержит перечисление всех входных сигналов логики. В начале описания процесса имеется группа операторов присваивания значений сигнала, которая соответствует значениям сигналов по умолчанию. Тем самым обеспечивается более компактная форма записи формируемых сигналов.

Последняя группа параллельных операторов присваивания значений сигналам (в нашем примере требуется только один — BusyR) соответствует описанию формирования сигналов на выходе регистра выходных сигналов.

Тестирование проектов и их фрагментов. Задача тестирования отдельных блоков или всего проекта целиком всегда решается одним и тем же способом. Обобщенная схема тестирования приведена на рис. 12.13. На входы тестируемого устройства подаются входные тестовые воздействия (стимулы) от устройства, формирующего эти стимулы. Выходы тестируемого устройства подключаются к устройству, анализирующему результаты экспериментов (реакции). Работа всех устройств тактируется общим тактовым сигналом. В зависимости от этапа разработки техническая реализация тестового комплекса будет различной. Она может быть программной, аппаратной или смешанной. В данном разделе рассматривается этап создания программной реализации фрагмента проекта. Работоспособность созданной программной модели может быть проверена двумя способами.

Первый способ (обычно применяемый при планируемой дальнейшей реализации в форме ПЛИС) заключается в компиляции модели в соответствующей САПР и последующем использовании встроенного редактора временных диаграмм этой же САПР. Для рассматриваемого примера воспользуемся ПЛИС фирмы Altera ACEX и соответственно САПР этой же фирмы Quartus II.

Средствами графического редактора создается временная диаграмма стимулов, при желании и ожидаемые реакции тестируемого устройства. При запуске моделиров-

щика выполняется функциональное (без учета реальных задержек элементов) или временное моделирование тестирующего комплекса. Результаты моделирования наблюдаются в этом же редакторе временных диаграмм.

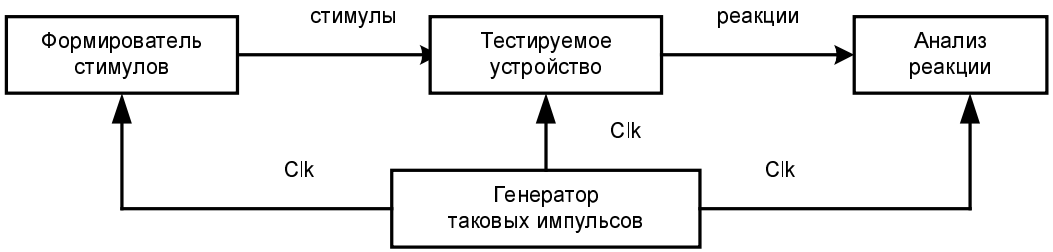


Рис. 12.13. Структура тестирующих комплексов

Второй способ заключается в использовании специальных пакетов моделирования (ModelSim фирмы Model Technology, Riviera или Active HDL фирмы Aldec и т. д.). В этом случае на языке проектирования аппаратуры создается специальная программа (обычно называемая Test-Bench), в которую как один из компонентов входит тестируемое устройство. Задачей программы является создание описаний блоков создания стимулов и анализа реакций. В простейших проектах блок анализа результатов опускается, и анализ правильности функционирования предполагается делать визуальным просмотром временных диаграмм. Для более или менее сложных схем полный визуальный просмотр всей диаграммы после любых изменений в тестовом комплексе становится неприемлемым и более рациональным является автоматизация анализа результатов моделирования.

Большинство современных проектов строится по синхронному принципу (достоинства и недостатки синхронных и асинхронных устройств уже были рассмотрены в *главе 3 и 4*). При этом важнейшим моментом при написании программ Test-Bench становится правильное написание временных соотношений для синхронизации подачи стимулов и анализа результатов относительно тактовой частоты основного проекта.

Без потери общности рассмотрения будем считать, что события в тестируемой схеме происходят по переднему фронту тактового импульса (системы, работающие по двум изменениям тактового импульса: фронту и спаду, достаточно редки и последующее изложение принципиально подходит и для них). На рис. 12.14 показан пример временной диаграммы синхронизации.

На диаграмме интервал T1 сигнала Clock определяет тактовую частоту работы тестируемого устройства. Все блоки тестового комплекса работают синхронно с этим сигналом. Минимально допустимое значение T1 для любого набора исходных данных соответствует максимально возможному быстродействию устройства. Допустимое значение интервала T2 зависит от организации схемы. В некоторых ситуациях значение этого интервала критично, но чаще всего (для удобства

схемотехнической реализации) его величина принимается равной 0,5 от интервала T1.

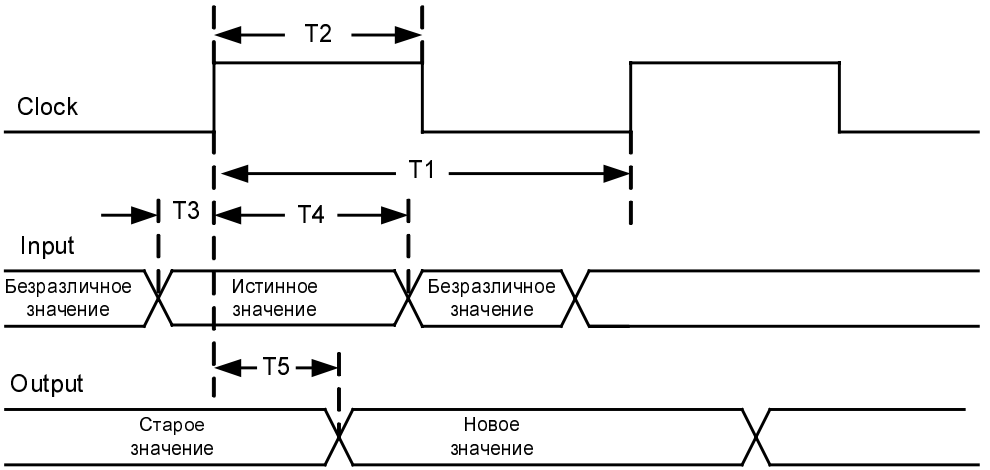


Рис. 12.14. Временная диаграмма поведения сигналов проектов

Группа сигналов Input соответствует входным сигналам тестируемого устройства (стимулам). Интервал T3 носит название "предустановка входного сигнала". Уменьшение интервала ниже предусмотренной границы приводит к пропуску такта. Для группы входных сигналов величина T3 должна выбираться соответствующей максимальной величине времени предустановки. Не соблюдение требований по величине времени предустановки вызывает серьезные проблемы при отладке работы аппаратуры, поскольку часть элементов отлаживаемой системы принимает новые данные, а другая часть на интервале в один такт сохраняет старые данные. Интервал T4 соответствует времени удержания входных сигналов. Необходимость введения интервала и его величина определяется как самой схемой, так и схемотехническими особенностями применяемых элементов.

Группа сигналов Output соответствует выходным сигналам тестируемого устройства (реакциям). Интервал T5 соответствует времени ожидания установления достоверных значений выходных сигналов. Как правило, интервал T5 соответствует времени, необходимому для завершения всех переходных процессов в тестируемой системе на наихудшем (требующем наибольшего времени ожидания) наборе входных данных.

В большинстве случаев при отсутствии жестких требований к быстродействию для упрощения рассмотрения выбирается вариант, при котором передний фронт тактового импульса является рабочим фронтом, а задний фронт этого же импульса выбирается в качестве момента переключения входных сигналов и момента контроля значений реакций устройства (т. е. T2 = T3 = T4 = T5).

В соответствии с указанными замечаниями обычно и строятся тестирующие программы. Фрагменты, создающие стимулы и осуществляющие контроль реакций, отражают их поведение и строятся в форме процессов, у которых раздельные вре-

менные интервалы образуются путем вставки синтаксических конструкций — операторов **WAIT FOR** T2; несмотря на определенную громоздкость текста, наиболее удобно строить процессы стимуляции в виде периодически повторяющихся фрагментов разделяемых операторами **WAIT**. Подобная организация программы допускает простое встраивание дополнительных проверок, комментариев и т. д.

Далее приводится листинг программы Test-Bench (листинг 12.9), тестирующей функционирование приведенного в примере автомата управления. В данном случае тестирование хотя и осуществляется для всех ветвей алгоритма, не является функционально полным, т. к. не включает все необходимые проверки значений выходных сигналов.

Листинг 12.9

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY Test_Example IS

END Test_Example;

ARCHITECTURE a_Test OF Test_Example  IS
CONSTANT T2: TIME:= 50 ns;
COMPONENT Example
PORT (
       Ack, Addr7, CLK, Emp, Rd, Reset, WR :  IN std_logic;
       BusyC, BusyR, ClrAddr, En, EnInc, Ld, Str, StrS   :  OUT std_logic
       );
END COMPONENT;

SIGNAL Start  :      std_logic;
SIGNAL  Ack, Addr7, Clk, Emp, Rd, Reset, Wr, BusyC, BusyR : std_logic;
SIGNAL ClrAddr, En, EnInc, Ld,  Str, StrS : std_logic;
BEGIN
       Base_Unit:  Example
       PORT MAP(
              Ack           => Ack,
              Addr7         => Addr7,
              Clk           => Clk,
```

```
            Emp          => Emp,
            Rd           => Rd,
            Reset        => Reset,
            WR           => Wr,
            BusyC        => BusyC,
            BusyR        => BusyR,
            ClrAddr      => ClrAddr,
            En           => En,
            EnInc        => EnInc,
            Ld           => Ld,
            Str          => Str,
            StrS         => StrS
      );

      Stim_Reack:   PROCESS
      BEGIN
--  ------------------------------------------
--  состояние Unit при действии сигнала Reset
CLK <= '0';                                     -- Takt 1
Reset <='1'; Wr <= '0'; RD <= '0'; ACK <= '0'; Emp <= '0'; Addr7<= '0';
-- ожидание  CLK
WAIT FOR T2;
CLK <='1';
ASSERT ClrAddr='1' REPORT "ClrAdr not 1" SEVERITY ERROR;
WAIT FOR T2;
--  ------------------------------------------
--  Clk=0 удержание состояния Unit при снятии сигнала Reset
CLK <= '0'; Reset<= '0';                        -- Takt 2
ASSERT ClrAddr='1' REPORT "ClrAdr not 1" SEVERITY ERROR;
WAIT FOR T2;
--  Clk=1 переход к состоянию
CLK    <='1';
WAIT FOR T2;
--  ------------------------------------------
--  удержание состояния Idle при отсутствии сигналов Wr или Rd
CLK <= '0';                                     -- Takt 3
ASSERT ClrAddr='1' REPORT "ClrAdr not 1" SEVERITY ERROR;
WAIT FOR T2;
```

```
CLK <='1';
ASSERT ClrAddr='0' REPORT "ClrAdr not 0" SEVERITY ERROR;
WAIT FOR  T2;
--  ----------------------------------------
CLK <= '0'; Wr <='1'; Rd <='1';                    -- Takt 4
WAIT FOR  T2;
CLK <='1';
ASSERT En='1'  REPORT "En not 1" SEVERITY ERROR;
WAIT FOR  T2;
--  ----------------------------------------
--  переход к состоянию Write при одновременном наличии Wr и Rd
CLK <= '0'; Emp <= '1';                            -- Takt 5
ASSERT En='1'ANDLd='1' REPORT "En or Ld = 0" SEVERITY ERROR;
WAIT FOR  T2;
CLK    <= '1';
ASSERT En='1' AND BusyC='1' AND Ld='1' REPORT
"En or BusyC or Ld  = 0" SEVERITY ERROR;
WAIT FOR  T2;
--  ----------------------------------------
--  действие сигнала Emp на сигнал Busyc
--  приоритетность сигнала Emp над сигналом Ack
CLK <= '0';  Ack  <= '1';                          -- Takt 6
WAIT FOR  T2;
CLK <='1'; Wr <= '1'; Rd <='1'; Emp <= '1'; Ack  <= '1';
WAIT FOR  T2;
--  ----------------------------------------
--  переход к состоянию Inc при наличии сигнала Ack
CLK <= '0';    Wr <= '0'; Rd <='0'; Emp <= '0';    -- Takt 7
ASSERT En='1'ANDLd='1' REPORT "En  or Ld = 0" SEVERITY ERROR;
WAIT FOR  T2;
CLK <='1';  Ack  <= '1';
WAIT FOR  T2;
--  ----------------------------------------
--  переход к состоянию Idle при отсутствии сигнала Addr7
CLK<= '0'; Ack<= '0';                              -- Takt 8
WAIT FOR  T2;
ASSERT EnInc='1'REPORT "EnInc=0" SEVERITY ERROR;
CLK<='1';
```

```
WAIT FOR  T2;
--  ------------------------------------------
--  переход к состоянию Read при наличии сигнала Rd
CLK <= '0'; Rd<='1';                              -- Takt 9
ASSERT EnInc='0' REPORT "EnInc=1" SEVERITY ERROR;
WAIT FOR  T2;
CLK <= '1';
WAIT FOR  T2;
--  ------------------------------------------
--  запаздывание на такт сигнала BusyR относительно Emp
CLK <= '0'; Ack <= '1'; Emp<='1';                 -- Takt 10
ASSERT  StrS='1' REPORT "StrS not 1" SEVERITY ERROR;
WAIT FOR  T2;
CLK <='1';
ASSERT Str='1'AND StrS='1'
REPORT "Str or StrS not 1" SEVERITY ERROR;
WAIT FOR  T2;
--  ------------------------------------------
--  переход к состоянию Inc при наличии сигнала Ack
CLK  <= '0'; Rd<='0'; Emp <= '0';                 -- Takt 11
ASSERT BusyR='1' REPORT "BusyR not 1" SEVERITY ERROR;
WAIT FOR  T2;
CLK     <='1';
ASSERT BusyR='1' REPORT "BusyR not 1" SEVERITY ERROR;
WAIT FOR  T2;
--  ---------------------------------------
Clk <= '0'; Ack <= '0'; Addr7<='1';               -- Takt 12
ASSERT EnInc='1' REPORT "EnINC not 1" SEVERITY ERROR;
WAIT FOR  T2;
CLK     <= '1';
ASSERT EnInc='1' REPORT "EnINC not 1" SEVERITY ERROR;
WAIT FOR  T2;
--  -------------------------------------
CLK <= '0'; Addr7 <= '0';                         -- Takt 13
ASSERT EnInc='0' REPORT "EnINC not 0" SEVERITY ERROR;
ASSERT ClrAddr='1' REPORT "ClrAddr not 1" SEVERITY ERROR;
WAIT FOR  T2;
CLK     <='1';
```

```
ASSERT ClrAddr='1' REPORT "ClrAddr not 1" SEVERITY ERROR;
WAIT FOR  T2;
--  ------------------------------------------------------------
CLK   <= '0';
ASSERT FALSE  REPORT "End Simulation" SEVERITY NOTE;
WAIT;
END PROCESS Stim_Reack;
END;
```

Полученная программа была проверена средствами моделирующего пакета ModelSim SE 5.8c фирмы Model Technology (подразделение фирмы Mentor Graphics). Результаты работы программы приведены на рис. 12.15. Поведение выходных сигналов автомата соответствовало ожидаемому. Однако в тестирующую программу была вставлена неверная проверка. В такте 3 сигнал ClrAdder реально не равен 0. В диалоговом окне САПР тестирующая программа выдала сообщение об ошибке на 80 нс.

Далее этот же текст автомата был откомпилирован в САПР MAX+Plus II фирмы Altera [III]. При компиляции в качестве планируемого семейства ИС было выбрано семейство ACEX 1K. При синтезе компилятор использовал 20 логических ячеек, из них в 5 использовались стандартные триггеры типа D управляемые фронтом (DFF). Результаты моделирования поведения синтезированного автомата приведены на рис. 12.16. На диаграмме можно увидеть, что поведение выходных сигналов в целом повторило результаты функционального моделирования, но появилось запаздывание выходных сигналов, обусловленное задержками элементов, образующих схему. Кроме того, у ряда сигналов возникли короткие паразитные импульсы. Такое поведение сигналов может служить причиной некорректной работы управляемых ими схем.

Если, например, сигнал ClrAddr планировался для использования в качестве асинхронного сигнала начальной установки счетчика, то паразитные импульсы могут сбрасывать счетчик в моменты времени, не предусмотренные разработчиком. Одной из причин возникшего расхождения между поведением автомата до и после синтеза является то, что при синтезе компилятор использовал (с целью экономии ресурсов) двоичное кодирование состояний автомата. Глубина требуемой при этом дешифрации состояний автомата привела к возникновению некоторых из наблюдаемых на диаграмме рисков. Переход к другим способам кодирования (например, к методу One Hot — один триггер соответствует одному состоянию) может снимать проблему. Общим принципом специальных кодировок является закрепление за интересующим проектировщика выходным сигналом одного из триггеров регистра состояния. В языке AHDL (см. [4]), например, в синтаксической конструкции, соответствующей заданию автомата, могут перечисляться такие сигналы и их значения в зависимости от состояний автомата.

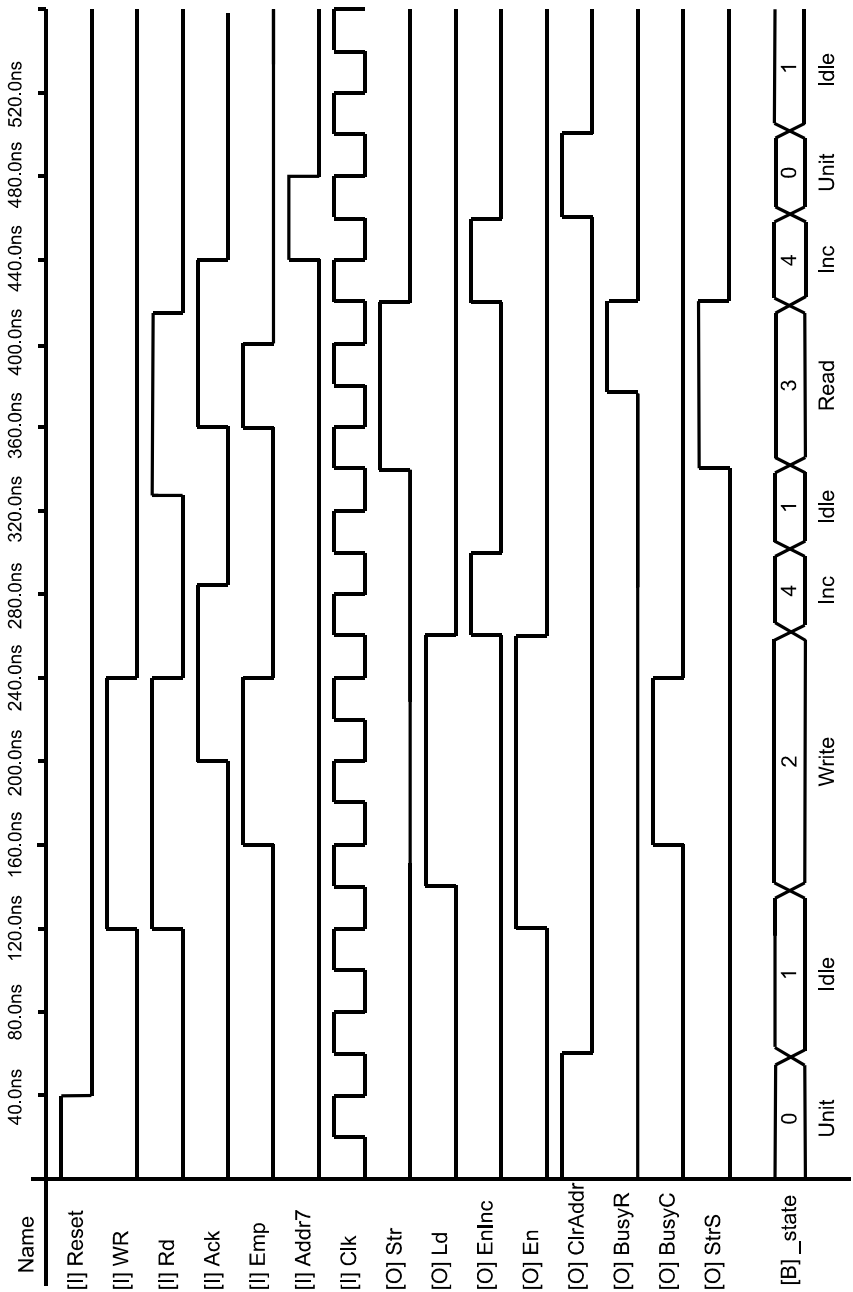


Рис. 12.15. Временная диаграмма поведения автомата (САПР ModelSim)

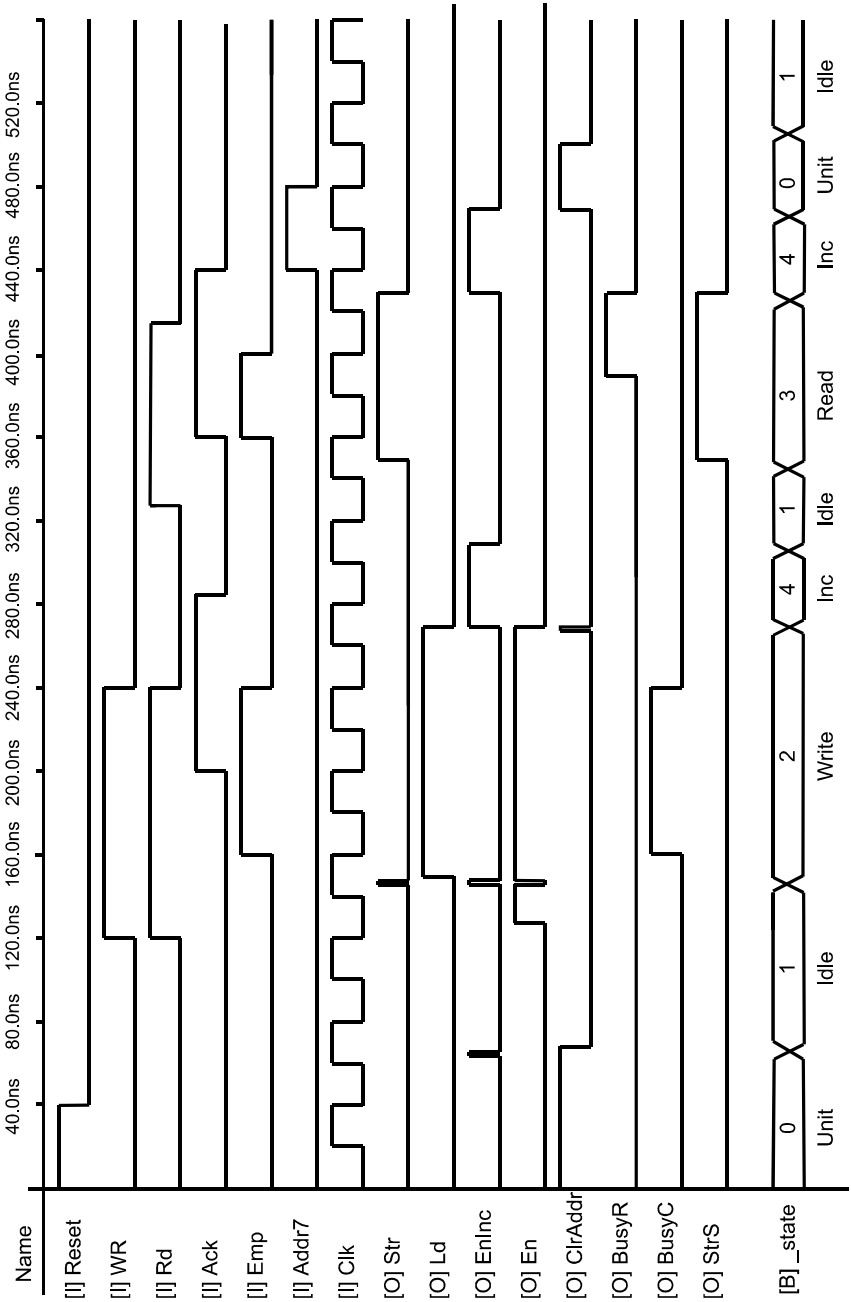


Рис. 12.16. Временная диаграмма поведения автомата (САПР MAX+Plus II)

Другим широко распространенным и универсальным (не зависящим от причины возникновения) способом борьбы с рисками у выходных сигналов является применение автоматов с синхронными выходами (см. рис. 12.10). Для демонстрации такого подхода в примере выбран сигнал Str. Убрать паразитный импульс (см. рис. 12.16) у исходного комбинационного сигнала можно при формировании синхронного сигнала StrS. Добавление сигнала StrS к проекту привело к необходимости ввода в листинг 12.8 семи вставок (отмечены пронумерованными комментариями):

- `вставка 1` — добавляет в проект новый выходной сигнал — StrS;
- `вставка 2` — объявляет внутренний сигнал проекта StrS_int, соответствующей комбинационной логике, которая подключена к входу триггера StrS;
- `вставка 3` — обеспечивает формирование асинхронного сброса триггера StrS;
- `вставка 4` — соответствует указанию на необходимость фиксации в триггере StrS состояния выхода комбинационной логики StrS_int по фронту тактирующего сигнала автомата Clk;
- `вставка 5` — определяет нулевое значение сигнала StrS_int по умолчанию;
- `вставка 6` — обеспечивает формирование единичного значения комбинационного сигнала StrS_int одновременно с формированием сигналов, обеспечивающих переход автомата из состояния Idle в состояние Read. Приоритетность перехода по сигналу Wr заставляет учесть это в записи условного выражения;
- `вставка 7` — позволяет обеспечить нулевое значение сигнала StrS после перехода автомата в состояние Inc. Перезапись комбинационного сигнала StrS_int в триггер блокируется при наличии сигнала Ack.

Из временной диаграммы (см. рис. 12.16) видно, что поведение сигнала StrS повторяет поведение сигнала Str, но в отличие от последнего не имеет паразитных выбросов.

Специальные виды кодировки состояний автомата, как правило, должен предусмотреть разработчик либо путем составления соответствующего (более детального) описания, либо заданием при синтезе указаний на использование соответствующих кодировок (для этого необходимо, чтобы компилятор в исходном тексте распознал описание автомата). Существенную помощь проектировщикам оказывают специально включаемые в используемый маршрут проектирования дополнительные пакеты, которые выполняют предварительную компиляцию проекта. Среди отечественных разработчиков наибольшее распространение получили два подобных пакета. Одним из них является синтезатор *Synplify* фирмы Synplicity [XXVIII], а другим *LeonardoSpectrum* фирмы Mentor Graphics [XVII]. Для использованного в примере программы HDL Designer в качестве такого промежуточного синтезирующего пакета логично ориентироваться на LeonardoSpectrum.

При компиляции текста листинга 12.8 пакет LeonardoSpectrum версии 2002e.16 создал структуру автомата в базисе ИС семейства ACEX 1K, требующую для своей реализации всего 12 логических ячеек (вместо 20). Синтезированная структура

опирается на 7 триггеров (DFF), что позволило практически для всех выходных сигналов (исключение составляют сигналы EN и BusyC) использовать выходы триггеров. Созданный синтезатором выходной файл (в формате EDIF) был откомпилирован в САПР Quartus II, а затем поведение результирующей конфигурации ИС было промоделировано. Как и следовало ожидать, на временной диаграмме у всех сигналов (выходов триггеров) отсутствовали ложные выбросы.

Рассмотренный пример показывает, что описание поведения устройства управления цифровыми блоками в форме автоматов открывает для проектировщика широкие возможности для гибкого управления работой используемых операционных блоков.

Введение в язык VHDL-AMS

Язык *VHDL-AMS* предназначен для моделирования смешанных (непрерывно-дискретных) систем. Стандарт этого языка IEEE Std 1076.1-1999 был утвержден в 1999 году. Язык ориентирован на описание систем достаточно широкого класса — электрических, механических, электромагнитных, тепловых и т. д. Для этого потребовалось не только добавить к возможности описания дискретных фрагментов средства описания поведения и структуры фрагментов, представленных обыкновенными нелинейными дифференциально-алгебраическими уравнениями, но и корректировать описание дискретных систем VHDL и добавить механизмы связи дискретных и непрерывных фрагментов. В настоящий момент язык может использоваться не только для описания непрерывно-дискретных систем, но и для моделирования поведения систем, описанных на этом языке. Примером пакета, поддерживающего решение таких задач, и к тому же имеющего удобный графический интерфейс, является программа SystemVision фирмы Mentor Graphics. Стремительное развитие элементной базы реконфигурируемых пользователем аналоговых и цифроаналоговых интегральных схем позволяет надеяться, что появление программных пакетов, синтезирующих описание аналого-цифровых устройств в описание конфигурации соответствующих типов ИС, вопрос не столь уж отдаленного будущего. Ниже остановимся на вопросах моделирования электронных аналого-цифровых систем.

Прежде всего, рассмотрим средства языка, требуемые для представления поведения аналоговых сигналов. Основной особенностью этих сигналов является их непрерывность как в диапазоне значений, так и в процессе изменения во времени. Необходимость описания поведения аналоговых сигналов потребовала введения в язык новых классов объектов **TERMINAL** и **QUANTITY**. Введенные объекты отражают основные свойства аналоговых сигналов. Для контроля допустимости соединения блоков между собой уже в описании интерфейса объектов (**ENTITY**) язык требует сначала указания принадлежности интерфейсных контактов к передаче сигналов определенных классов дискретных (**SIGNAL**) или непрерывных (**TERMINAL** или **QUANTITY**) и лишь затем приводится характеристика (технологическая) типа соединения (например, **OUT** bit и electrical соответственно).

В архитектурных телах описываемых объектов может использоваться два различных подхода к описанию поведения непрерывных сигналов. Первый подход, применяемый к консервативным системам (conserved systems), ориентирован на закон сохранения энергии при определении взаимодействия двух соединенных аналоговых элементов. В трактовке электрических цепей это соответствует поведению, подчиняющемуся законам Кирхгофа. Второй подход, характерный для неконсервативных систем (non-conserved systems), соответствует направленному воздействию одного аналогового объекта на другой. В этом случае описание аналогового объекта включает фрагменты, отвечающие за получение входных величин, а также фрагменты, соответствующие их последующему математическому преобразованию, и фрагменты, определяющие вывод преобразованных величин. Класс портов (контактов) `TERMINAL` служит для определения консервативных портов модели, предназначенных для подключения цепей с непрерывными сигналами. Класс портов (контактов) `QUANTITY` так же предназначен для подключения непрерывных сигналов, но служит для определения неконсервативных портов модели и в соответствии с принципом обработки данных таких портов требует указания направления передачи аналоговых величин `IN` или `OUT`.

В декларативной части архитектурных тел помимо определения дискретных компонентов потребовалось введение объявления объектов нового класса — `QUANTITY`. Основное свойство этих объектов — возможность принимать значения в непрерывном диапазоне и изменять их в промежутки времени между отдельными событиями. Существуют три основные разновидности quantity-параметра:

- free `QUANTITY`;
- branch `QUANTITY`;
- source `QUANTITY`.

Первый тип параметров (free `QUANTITY`) может использоваться для представления аналоговых величин в неконсервативных фрагментах. Этот тип параметров упрощает просмотр в моделирующих программах временного поведения сигналов и делает более простым и понятным описание поведения моделируемого фрагмента.

Второй тип параметров (branch `QUANTITY`) используется при определении консервативных систем. Описание электрических цепей (с точки зрения законов Кирхгофа) требует определения для `TERMINAL`-контакта или `QUANTITY`-параметра трех аспектов поведения. Один соответствует `ACROSS`-аспекту — потенциалу между `TERMINAL`-контактами, другой `THROUGH`-аспекту — току между контактами и, наконец, последний `TO`-аспект — идентификация связываемых контактов между собой.

Третий тип параметров (source `QUANTITY`) применяется при спектральном анализе сигналов или анализе шумов.

В основной части архитектурного тела поведение цифровых фрагментов описывалось в языке VHDL предложениями параллельных операторов (Concurrent Statement), а в языке VHDL-AMS поведение аналоговых фрагментов задается новым типом предложений (Simultaneous Statement — с непрерывными операторами). Оба

типа предложений могут размещаться в архитектурном теле в произвольном порядке.

Хотя синтаксис непрерывных операторов естественно отличается от синтаксиса параллельных операторов, набор операторов традиционен и содержит как безусловные, так и условные операторы. Прежде всего, требуется оператор, определяющий функционирование аналоговых сигналов. Оператор (**= =**), определенный в стандарте как элементарное непрерывное предложение (Simple_Simultaneous_ Statement), для неконсервативных систем задает функциональную зависимость выходного параметра (free **QUANTITY**) от времени. Частным случаем зависимости может быть и постоянство значения. Для консервативных же систем оператор, кроме задания зависимости параметра **QUANTITY** от времени, определяет правила взаимодействия **ACROSS-** и **THROUGH-**аспектов параметра (branch **QUANTITY**) между собой.

Далее необходимы условные непрерывные операторы (**IF** и **CASE**). Внешне синтаксис этих операторов совпадает с синтаксисом аналогичных последовательных операторов и отличается от них заменой ключевого слова **THEN** на **USE**. В теле этих операторов естественно должны использоваться непрерывные операторы. Непрерывные предложения могут включаться и в тело параллельных операторов (**FOR . . . GENERATE**).

Задание функциональной зависимости между различными аспектами объектов типа **QUANTITY** (для электрических цепей — током и напряжением) потребовало не только введения нового оператора присваивания "= =", но и новых типов атрибутов для аспектов **QUANTITY-**параметров в том числе: `dot` — временного дифференцирования параметра; `int` — интеграла во времени и delyaed(T) — задержки на время T изменения параметра.

Пример описания идеального конденсатора соответствует последовательности операторов, приведенных в листинге 12.10.

Листинг 12.10

```
ENTITY capacitor IS
  GENERIC (
    cap             : capacitance;        -- емкость в фарадах
    v_ic            : real := real'low);  -- начальное напряжение
   PORT (
    TERMINAL p1, p2 : electrical);
END ENTITY capacitor;
ARCHITECTURE ideal OF capacitor IS
  QUANTITY v ACROSS i THROUGT p1 TO p2;
BEGIN
  IF domain = quiescent_domain AND v_ic /= real'low USE
    v == v_ic;                      -- Учитываем начальное значение v_ic
```

```
  ELSE
    i   == cap * v'dot;            -- Основное дифференциальное уравнение
  END USE;
END ARCHITECTURE ideal;
```

В приведенном примере непрерывное изменение значений напряжений на обкладках конденсатора задается дифференциальным уравнением зависимости тока и напряжения $I = C \cdot dV/dt$, при этом учитывается начальное значение напряжения.

Ряд специальных синтаксических конструкций языка позволяет задать в моделирующей программе требование спектрального анализа (spectrum) или анализа шумов (noise) контролируемого параметра (модели электрического сигнала).

Смешанное моделирование требует введения механизмов и средств связи между событийно-управляемым моделированием цифровой части системы и решения задачи Коши для аналоговой части.

Чтобы обеспечить связь аналогового фрагмента с цифровым, аналоговый фрагмент должен сформировать событие. Инициализация события возникает в процессе решения дифференциальных уравнений, описывающих аналоговую часть системы, при достижении выбранным непрерывным параметром определенной величины. Чаще всего формируется сигнал, запускающий соответствующее действие (например, процесс). Для этого могут использоваться условные операторы последовательные или параллельные. Для записи условия достижения используется атрибут `above`, определенный в языке для **`QUANTITY`**. Примером фрагмента, описывающего поведение аналогового компаратора, который формирует единичное значение дискретного сигнала d при превышении разностью потенциалов V между входными контактами a и ref заданного порогового уровня threshold, может служить следующий приведенный текст (листинг 12.11).

Листинг 12.11

```
ENTITY comparator IS
  GENERIC (threshold: real:=0.0);
  PORT (TERMINAL a, ref : electrical;
    SIGNAL d : OUT bit);
END ENTITY comparator;

ARCHITECTURE simple OF comparator IS
  QUANTITY v ACROSS i THROUGH a TO ref;
BEGIN
    v == 1.0E6*i;            -- входное сопротивление компаратора 1 Мом.
    d <= '1' WHEN v'above(threshold) ELSE '0';
END ARCHITECTURE simple;
```

Передача управления от дискретных к аналоговым фрагментам происходит при возникновении соответствующего события. В этом случае событие в дискретном фрагменте инициализирует запуск или изменения параметров того или иного непрерывного параметра в аналоговом фрагменте. Инициализация достигается исполнением оператора присваивания непрерывного сигнала либо автономного, либо входящего в условный оператор. В качестве примера рассмотрим описание идеализированного переключателя электрических сигналов. При изменении состояния цифрового сигнала sw_state происходит соответствующее изменение параметров соединения выходного контакта p_out с входными контактами p_in1 и p_in2. Атрибут `ramp` дискретного сигнала r_sig1 в примере (листинг 12.12) обеспечивает формирование соответствующего значения аналогового параметра `r1` через заданный временной интервал с сохранением непрерывности значений параметра `r1`.

Листинг 12.12

```
ENTITY switch IS
  PORT ( sw_state : IN std_ulogic; --Цифровой управляющий вход
    TERMINAL p_in1, p_in2, p_out : electrical );
    -- Аналоговые  контакты переключателя
END ENTITY switch;
ARCHITECTURE ideal OF switch IS
  CONSTANT r_open : resistance := 1.0e6;
  -- Сопротивление открытого состояния переключателя
  CONSTANT r_closed : resistance := 0.001;
  -- Сопротивление закрытого  состояния переключателя
  CONSTANT trans_time : real := 0.00001;
  -- Время переключения в каждое состояние
  SIGNAL r_sig1 : resistance := r_closed;
  -- переменная сопротивление закрытого состояния ключа
  SIGNAL r_sig2 : resistance := r_open;
  -- переменная сопротивление открытого состояния ключа
  QUANTITY v1 ACROSS i1 TROUGH p_in1 TO p_out;  -- V и I от in1 до out
  QUANTITY v2 ACROSS i2 TROUGH p_in2 TO p_out;   -- V и I от  in2 до out
  QUANTITY r1 : resistance;
  -- время зависимое сопротивление от in1 до out
  QUANTITY r2 : resistance;
  -- время зависимое сопротивление от in2 до out
BEGIN
  PROCESS (sw_state) IS -- запуск от дискретного сигнала sw_state
```

```
  BEGIN
    IF sw_state = '0' OR sw_state = 'L' THEN
    -- закрыть sig1, открыть sig2
      r_sig1 <= r_closed;
      r_sig2 <= r_open;
    ELSIF sw_state = '1' OR sw_state = 'H' THEN
    -- открыть sig1, закрыть sig2
      r_sig1 <= r_open;
      r_sig2 <= r_closed;
    END IF;
  END PROCESS;
  r1 == r_sig1'ramp(trans_time, trans_time);
  -- Перевод сигнала в параметр с сохранением непрерывности сопротивления
  r2 == r_sig2'ramp(trans_time, trans_time);
  -- Перевод сигнала в параметр с сохранением непрерывности сопротивления
  v1 == r1 * i1; -- Применение закона Ома для контакта in1
  v2 == r2 * i2; -- Применение закона Ома для контакта in2
END ARCHITECTURE ideal;
```

Рассмотренные выше фрагменты используем при описании смешанной аналого-цифровой схемы, приведенной на рис. 12.17. Схема содержит осциллятор (генератор OSC цифровых сигналов V_OSC = V_INP с периодом 80 мкс и длительностью импульса 50 мкс), управляющий через инвертор U1 цифроаналоговым переключателем switch. Образуемые в результате переключения импульсы V_SW поступают на интегрирующую цепочку, состоящую из резистора R1 (10 ком) и конденсатора C1 (1 нф). Образуемое на конденсаторе напряжение V_INT сравнивается в аналого-цифровом компараторе comparator с пороговым уровнем LEVEL. Выходная цифровая схема 2И (блок U3) осуществляет временную селекцию цифрового сигнала V_MID, образуемого компаратором, с входным напряжением V_OSC = V_INP и формирует выходной сигнал V_OUT.

Языковое описание этой схемы создавалось с привлечением средств моделирующего пакета SystemVision (версия SV2002.0.16) фирмы Mentor Graphics. Для этого сначала текстовые описания рассмотренных ранее примеров описаний поведения блоков switch и comparator, идеализированной модели конденсатора и резистора, а также традиционных дискретных блоков были включены в состав проекта COMMON. Затем были образованы соответствующие этим компонентам графические блоки, из них и из стандартных элементов библиотеки SystemVision была создана схема системы, соответствующая рисунку. Результат компиляции верхнего уровня иерархии приведен в листинге 12.13.

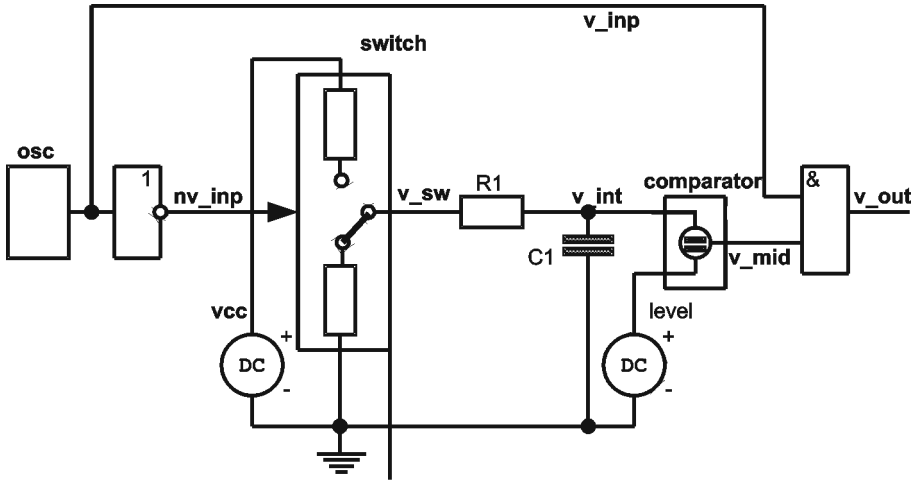


Рис. 12.17. Смешанная аналого-цифровая система

Листинг 12.13

```
--
-- File : d:\CADWork\AMS\Default\genhdl\vhdl\common.vhd
-- CDB  : d:\CADWork\AMS\default\default.cdb
-- By   : CDB2VHDL Netlister version 16.1.0.2

-- Entity/architecture declarations

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

LIDRARY IEEE_proposed;
USE IEEE_proposed.electrical_systems.ALL;

ENTITY common IS
END common;

ARCHITECTURE common OF common IS
    -- Component declarations
    -- Signal declarations
    TERMINAL LEVEL : electrical;
    SIGNAL nV_INP : std_logic;
    TERMINAL V_IN : ELECTRICAL;
```

```
    SIGNAL V_INP : std_logic;
    SIGNAL V_MID : std_logic;
    SIGNAL V_OUT : std_logic;
    TERMINAL V_SW : electrical;
    TERMINAL XSIG010010 : electrical;
BEGIN
    -- Signal assignments
    -- Component instances
    OSC1 : ENTITY work.OSC(ideal)
        PORT MAP(
            clock_out => V_INP
        );
    U1 : ENTITY work.inverter(ideal)
        PORT MAP (
            input => V_INP,
            output => nV_INP
        );
    switch1 : ENTITY work.switch(ideal)
        PORT MAP (
            sw_state => nV_INP,
            p_in1 => ELECTRICAL_REF,
            p_in2 => XSIG010010,
            p_out => V_SW
        );
    R1 : ENTITY work.resistor(ideal)
        GENERIC MAP(
            res => 10000.0
        )
        PORT MAP (
            p1 => V_SW,
            p2 => V_IN
        );
    C1 : ENTITY work.capacitor(ideal)
        GENERIC MAP (
            cap => 1.0e-9
        )
        PORT MAP (
            p1 => V_IN,
```

```
            p2 => ELECTRICAL_REF
        );
    comparator1 : ENTITY work.comparator(simple)
        PORT MAP (
            a => V_IN,
            ref => LEVEL,
            d => V_MID
        );
    v1 : ENTITY work.v_constant(ideal)
        GENERIC MAP (
            level => 5.0
        )
        PORT MAP (
            pos => XSIG010010,
            neg => ELECTRICAL_REF
        );
    v2 : ENTITY work.v_constant(ideal)
        GENERIC MAP (
            level => 2.5
        )
        PORT MAP (
            pos => LEVEL,
            neg => ELECTRICAL_REF
        );
    U3 : ENTITY work.and2(ideal)
        PORT MAP (
            in1 => V_INP,
            in2 => V_MID,
            output => V_OUT
        );
END common;
```

Как обычно, проект опирается на стандартную библиотеку (строка — `LIBRARY IEEE;`) и использует понятие стандартной логики (строка — `USE IEEE.std_logic_1164.all;`), кроме того, аналоговая часть проекта требует подключения новой библиотеки (строки — `LIBRARY IEEE_proposed;` и `USE IEEE_proposed.electrical_systems.ALL;`).

Пакет создал проект в форме структурного описания соединения компонентов структуры. Поэтому его вид характерен для традиционных VHDL-программ.

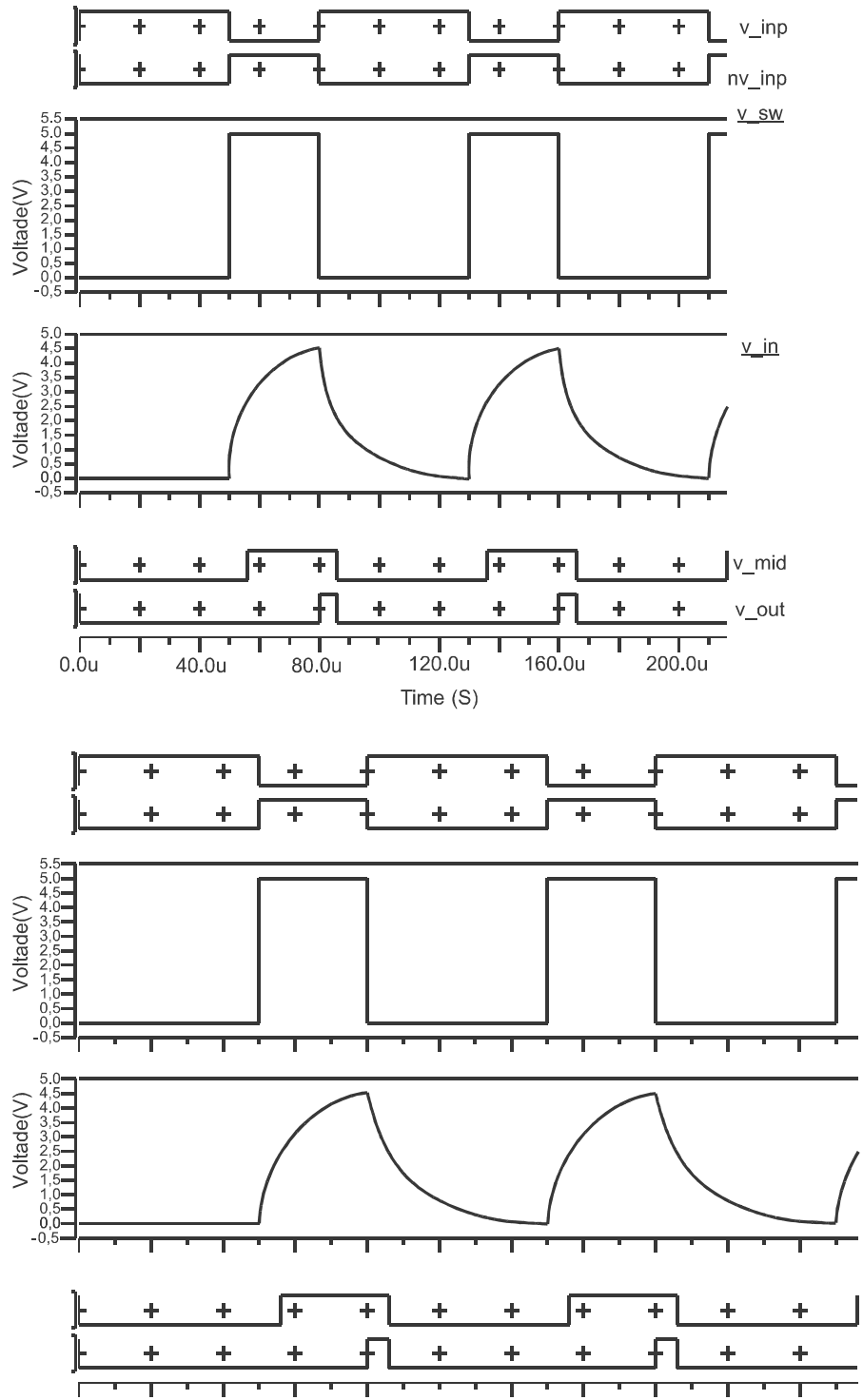


Рис. 12.18. Результаты моделирования аналого-цифровой схемы

Основное отличие состоит в том, что для соединений используются объявленные в архитектурном теле цифровые (`SIGNAL`) и аналоговые (`TERMINAL`) сигналы. Остальные отличия можно увидеть только в файле предыдущего уровня иерархии (в описаниях поведения аналоговых и смешанных компонентов).

Результаты компиляции использовались для моделирования поведения проекта во времени. Эти результаты приведены на рис. 12.18 и показывают изменение во времени как цифровых, так и аналоговых сигналов.

§ 12.7. Пример автоматизированного проектирования цифрового устройства с использованием языков описания аппаратуры

Современные методы и средства проектирования рассмотрим на примере разработки устройства, либо записывающего по запросу параллельный восьмиразрядный код в буферное ОЗУ, либо выводящего байт из заданного адреса буферного ОЗУ в форме последовательного кода. Для определенности будем ориентироваться на микросхемы программируемой логики фирмы Altera, а вследствие этого и на САПР этой же фирмы Quartus II.

Первый этап. Рассмотрение ТЗ на разрабатываемое устройство

Независимо от формы представления, ТЗ, очевидно, должно содержать следующие ключевые сведения:

- объем буферного ОЗУ 256 восьмиразрядных слов;
- запись в ОЗУ осуществляется сигналами внешнего управляющего устройства;
- внешнее чтение статуса устройства позволяет определить состояние его выходного регистра (пуст или полон);
- вывод последовательного кода осуществляется по запросу приемника последовательного кода и сопровождается стробирующими сигналами со стороны разрабатываемого устройства.

Перечисленные пункты ТЗ предопределяют основные блоки устройства и их взаимодействие. Блочная схема устройства приведена на рис. 12.19. Функциональное назначение блоков следует из их названий. Схема укрупненно отображает следующие процессы.

Внешнее управляющее устройство (процессор) обеспечивает запись байтов в ОЗУ, подавая на него помимо 8-разрядных данных также 8-разрядный адресный код и строб записи Write. Преобразователь параллельного кода в последовательный представляет в своей основе сдвигающий регистр, загружаемый параллельно байтом из ОЗУ, и затем по командам из устройства управления выводящий последова-

тельные данные во внешнее устройство — приемник последовательного кода. Появление разрядов последовательного кода отмечается во времени сигналами стробирования Strob. Загрузка данных в преобразователь параллельного кода инициируется процессором по сигналу Read, адрес загружаемых данных должен быть перед этим задан процессором.

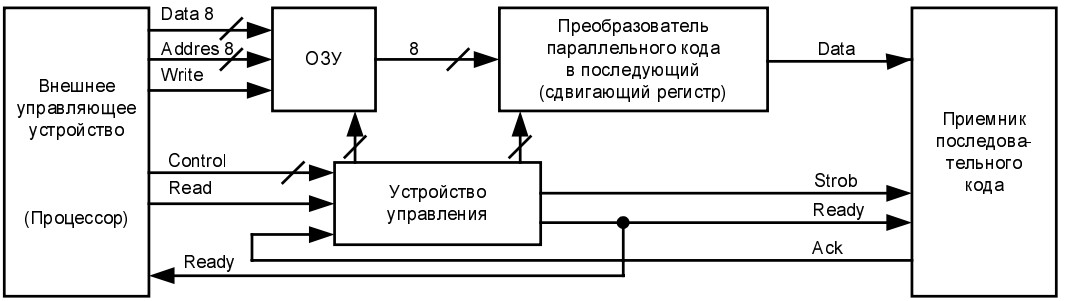


Рис. 12.19. Блок-схема устройства, принятого в качестве примера для проектирования средствами САПР

Готовность преобразователя кода к передаче данных индицируется сигналом Ready, поступающим как на приемник последовательных данных, так и на соответствующий вход процессора. Асинхронный характер обмена с приемником данных обеспечивается сигналом разрешения передачи Ack со стороны приемника. В состав сигналов шины Control входят сигналы тактирования, сброса и др.

Второй этап. Разработка общей структуры операционного блока

Сопоставляя функциональный состав библиотеки САПР Quartus II и блоков схемы (см. рис. 12.19), нетрудно увидеть, что для реализации рассматриваемого устройства из состава библиотеки выбранной САПР можно использовать следующий набор библиотечных параметризируемых модулей (LPM):

- блок ОЗУ (lpm_ram_dq) с организацией 256×8;
- 8-разрядный сдвигающий регистр выходного кода (lpm_shiftreg);
- счетчик сдвигов регистра вывода на 8 состояний (lpm_counter).

Понятие параметризуемых модулей соответствует возможности настроить выбранный библиотечный элемент на определенный режим функционирования, на определенную разрядность данных, их полярность и т. д.

Структурная схема устройства, включающая эти операционные блоки и автомат, управляющий считыванием кода из ОЗУ и его преобразованием в последовательную форму (AvtOutBt), может приобрести вид, приведенный на рис. 12.20.

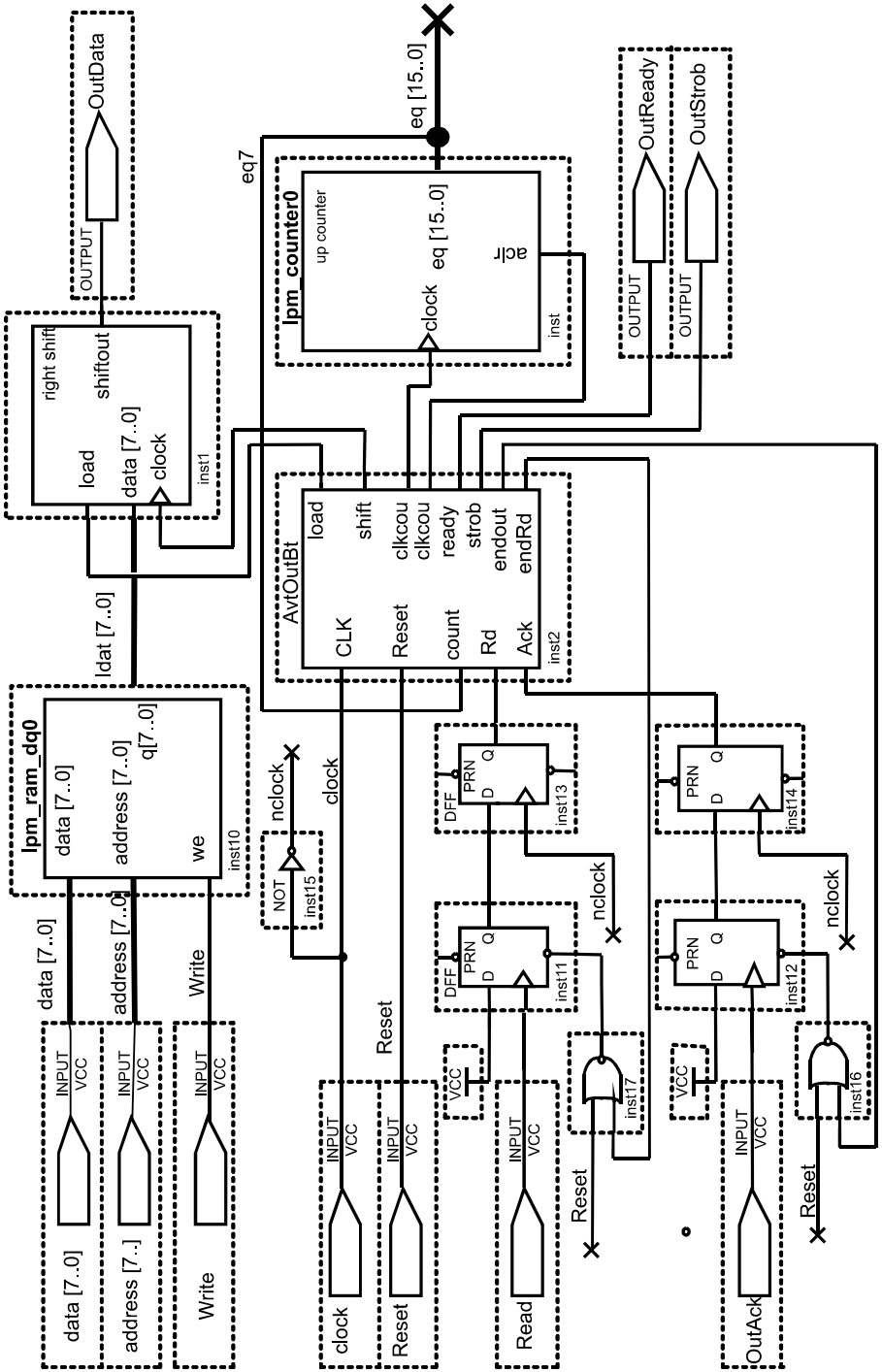


Рис. 12.20. Структурная схема устройства, проектируемого средствами САПР

Кроме указанных ранее базовых блоков, в схеме присутствует ряд дополнительных элементов. Условные обозначения всех элементов схемы соответствуют стандарту, принятому в САПР Quartus II. Необходимость введения дополнительных элементов (инвертора, четырех D-триггеров и двух схем 2ИЛИ-НЕ) диктуется требованиями временной или аппаратной совместимости отдельных блоков схемы. Более подробные пояснения будут приведены в следующем разделе, поскольку этапы разработки операционной части и устройства управления операционными элементами тесно связаны и обычно выполняются итерационно.

Третий этап. Описание работы управляющего автомата

При разработке поведения управляющего автомата необходимо учесть, что функционирование устройства определяется сигналом clock и происходит асинхронно относительно внешнего устройства, управляющего чтением и записью в ОЗУ и относительно другого внешнего устройства, запрашивающего и принимающего информацию в последовательной форме.

При выборе из библиотеки САПР в качестве ОЗУ модуля типа lpm_ram_dq (т. е. с раздельными шинами чтения и записи данных) и при его настройке на асинхронный режим работы исчезает целый ряд проблем. Во-первых, нет необходимости введения элементов, разделяющих данные для записи и считывания. Во-вторых, полностью снимается с управляющего автомата проблема организации синхронизации режима записи данных в ОЗУ со стороны внешнего устройства. А вот операция чтения из ОЗУ должна быть синхронизирована с работой автомата. Поэтому внешнее устройство, управляющее памятью, прежде чем снять установленный адрес, должно убедиться, что предыдущая информация из сдвигового регистра забрана. Для этой цели оно может ориентироваться на сигнал OutReady. Сигнал устанавливается, когда информация уже находится в сдвигающем регистре, и сбрасывается, когда выдан последний бит данных.

Возможный алгоритм работы устройства управления разрабатываемого устройства, отвечающий сформулированным выше требованиям, может приобрести вид, соответствующий граф-схеме переходов автомата, приведенной на рис. 12.21. Граф-схема переходов при помощи графического редактора программы *StateCAD Version 3.2* пакета Workview Office фирмы Viewlogic (сейчас Innoveda [XII]) была занесена в соответствующий диаграммный файл, что, как будет показано далее, существенно упрощает не только отладку и возможные корректировки алгоритма, но и создание соответствующих программных текстов.

Перейдем к анализу автомата AvtOutBt, управляющего выводом последовательного кода по запросу и сопровождающего такую выдачу сигналами стробирования.

Основу алгоритма составляют два последовательно выполняемых блока. Первый блок по сигналу Rd автомата, образуемому из сигнала Read внешнего управляющего устройства, считывает байт из памяти, а затем загружает его в сдвигающий регистр.

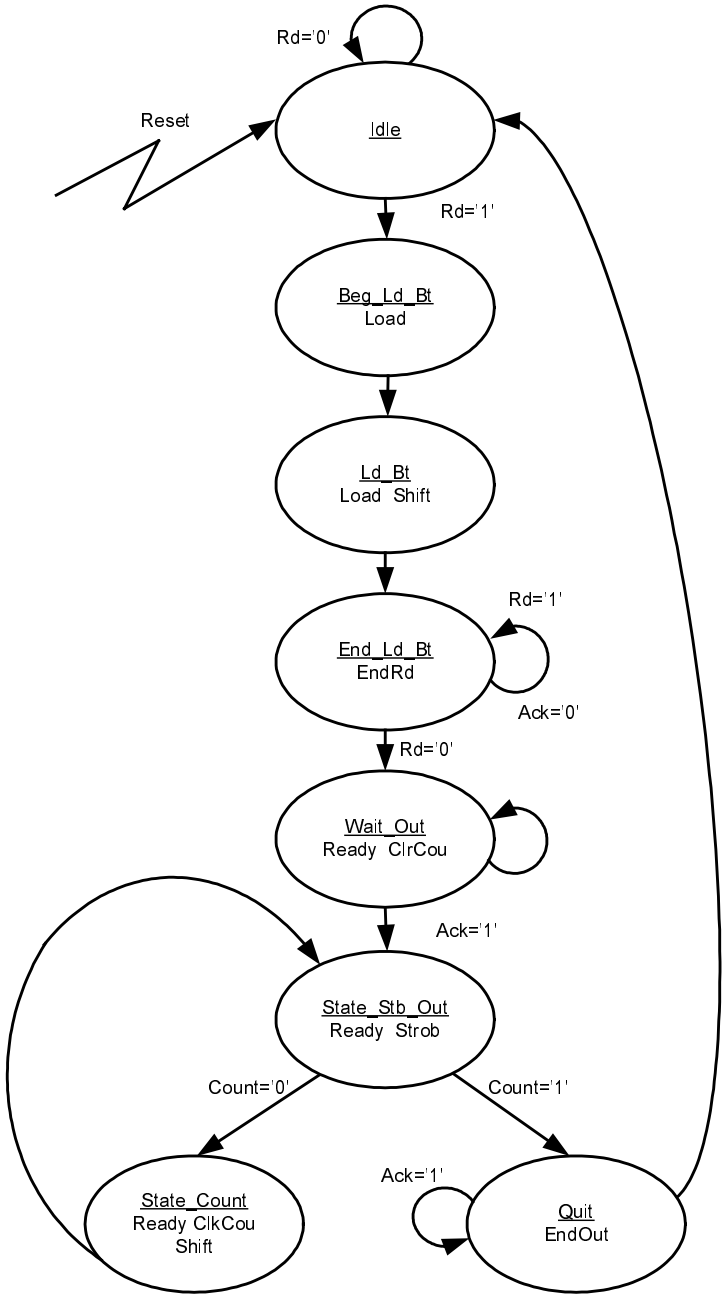


Рис. 12.21. Граф-схема автомата управления
для проектируемого средствами САПР устройства

Второй блок содержит последовательность действий, которая в ответ на запускающий внешний сигнал OutAck, преобразуемый в сигнал Ack автомата, осуществляет циклический вывод на выходной контакт OutData данных из сдвигающего

регистра и при этом сопровождает каждый бит стробирующим сигналом OutStrob (контакт автомата strob). Количество требуемых итераций цикла подсчитывает счетчик сдвигов. Следует обратить внимание на петли ожидания в состояниях Idle, EndLdBt и WaitOut, наличие которых обеспечивает необходимую синхронизацию сигналов запрос-ответ от внешнего устройства (квитирование). Определенные проблемы могли бы создать сигналы Read и OutAck ввиду асинхронности изменения их значений относительно тактовой частоты автомата.

Триггеры DFF inst13 и inst14 (см. рис. 12.20) обеспечивают требуемые времена предустановки и удержания входных сигналов и позволяют избежать некорректной работы автомата. Триггеры DFF inst11 и inst12 введены в схему, чтобы свести к минимуму вероятность появления состояния *метастабильности* (см. *§3.3*) у триггеров DFF inst13 и inst14.

Пояснения к синтаксису VHDL программы устройства управления

Для автомата нашего примера с помощью программы StateCAD Version 3.2 была выполнена трансляция диаграммы. Был создан вариант, ориентированный на возможности языка описания аппаратуры высокого уровня VHDL (листинг 12.14).

При компиляции из графической формы в текстовую программа StateCAD учитывает, для компилятора какой фирмы предполагается использовать описание автомата. Аналогичные соображения должны приниматься во внимание при ручном написании программ. Это ограничение возникает из-за того, что набор допустимых синтаксических конструкций языка для различных фирм существенно отличается от стандартного. Для примера выбрана ориентация на САПР Workview Office (как имеющую меньшие ограничения). В этой программе многое повторяет (в плане функционального назначения) предыдущую программу, и большинство отличий связано с синтаксисом операторов языка VHDL.

Листинг 12.14

```
-- VHDL code created by Visual Software Solution's StateCAD Version 3.2
-- This VHDL code (for use with Workview Office) was generated using:
-- enumerated state assignment with structured code format.
-- Minimization is enabled, implied else is enabled,
-- and outputs are manually optimized.

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

LIBRARY synth;
USE synth.vhdlsynth.ALL;
```

```
ENTITY AvtOutBt IS
  PORT (CLK,reset,count,rd,ack : IN std_logic;
   load,shift,clkcou,clrcou,ready,strob,endout,endRd: OUT std_logic);
END;

ARCHITECTURE BEHAVIOR OF AvtOutBt IS
  TYPE type_sreg IS (Idle,BegLdBt,LdBt,EndLdBt,WaitOut,StateStbOut,
   StateCount,Quit);
  SIGNAL sreg, next_sreg: type_sreg;
BEGIN
  PROCESS (CLK, next_sreg)
  BEGIN
   IF CLK='1' AND CLK'event THEN
     sreg<=next_sreg;
   END IF;
  END PROCESS;

  PROCESS (sreg,reset,count,rd,ack)
  BEGIN
   load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
   ready<='0'; strob<='0'; endout<='0'; endRd<='0';
   next_sreg<=Idle;

   IF (Reset='1') THEN
     load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
     ready<='0'; strob<='0'; endout<='0'; endRd<='0';
     next_sreg<=Idle;
   ELSE
     CASE sreg IS
      WHEN Idle =>
        load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
        ready<='0'; strob<='0'; endout<='0'; endRd<='0';
        IF (rd='0') THEN
         next_sreg<= Idle;
        END IF;
        IF (rd='1') THEN
         next_sreg<=BegLdBt;
        END IF;
```

```
      WHEN BegLdBt =>
        load<='1'; shift<='0'; clkcou<='0'; clrcou<='0';
        ready<='0'; strob<='0'; endout<='0'; endRd<='0';
        next_sreg<=LdBt;
      WHEN LdBt =>
        load<='1'; shift<='1'; clkcou<='0'; clrcou<='0';
        ready<='0'; strob<='0'; endout<='0'; endRd<='0';
        next_sreg<=EndLdBt;
      WHEN EndLdBt =>
        load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
        ready<='0'; strob<='0'; endout<='0'; endRd<='1';
        next_sreg<=Idle;
      WHEN WaitOut =>
        load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
        ready<='0'; strob<='0'; endout<='0'; endRd<='0';
        IF (ack='0') THEN next_sreg<=WaitOut; END IF;
        IF (ack='1') THEN next_sreg<=StateStbOut; END IF;
      WHEN StateStbOut =>
        load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
        ready<='1'; strob<='1'; endout<='0'; endRd<='0';
        IF (count='0') THEN next_sreg<=StateCount; END IF;
        IF (count='1') THEN next_sreg<=Quit; END IF;
      WHEN StateCount =>
        load<='0'; shift<='1'; clkcou<='1'; clrcou<='0';
        ready<='1'; strob<='0'; endout<='0'; endRd<='0';
        next_sreg<=StateStbOut;
      WHEN Quit =>
        load<='0'; shift<='0'; clkcou<='0'; clrcou<='0';
        ready<='0'; strob<='0'; endout<='1'; endRd<='0';
        IF (ack='0') THEN next_sreg<=Idle; END IF;
        IF (ack='1') THEN next_sreg<=Quit; END IF;
      WHEN OTHERS =>
     END CASE;
   END IF;
  END PROCESS;
END BEHAVIOR;
```

Разделы проектного модуля типичны для языка VHDL. В заголовочном разделе **ENTITY** (аналог прототипа в алгоритмических языках) перечислены имена и типы всех сигналов: входных внешних управляющих сигналов — тактового сигнала (CLK), начального сброса (reset), запросов на чтение и вывод (rd и ack) соответственно, и, наконец, сигнала окончания счета (count) и выходных управляющих сигналов — управления сдвигающим регистром (load, shift), управления счетчиком сдвигов (clrcou, clkcou), внешними выходными сигналами (ready, strob) и сигналами окончания подрежимов (endout, endRd).

Следующий раздел — **ARCHITECTURE** представляет собой описание архитектуры или поведения (в нашем случае поведения) блока, интерфейс которого был описан в разделе **ENTITY**. Как и в обычных языках, в начале раздела дается описание типов и объявление переменных, используемых при описании действий, выполняемых в разделе **ARCHITECTURE**. В данном автомате определен перечислительный тип данных type_sreg со всем списком допустимых значений (они, естественно, совпадают с именами, введенными в граф-схеме переходов). Далее в тексте объявлены две переменные (класса **SIGNAL**) sreg и next_sreg введенного типа type_sreg. Введение двух переменных связано с желанием избежать гонок в автомате при переходе от одного состояния к другому.

Главная часть архитектурного тела содержит два оператора параллельного типа (процесса). Первый процесс запускается на исполнение каждый раз, когда происходит изменение сигналов CLK или next_state. Однако его основное действие — назначение автомату нового состояния происходит только по переднему фронту сигнала CLK. Использование для тактирования автомата переднего фронта синхронизирующего сигнала (предложение **IF** CLK='1' **AND** CLK'event **THEN** sreg<=next_sreg; **END IF;**) служит для синхронизации выбранных библиотечных операционных узлов и обеспечит стабильность входных управляющих сигналов в моменты тактирования.

Поведение управляющего автомата в тексте программы задано вторым процессом. Второй процесс запускается каждый раз, когда изменяется состояние автомата или изменяется какой-либо входной сигнал, содержимое этого процесса и определяет поведение управляющего автомата. При составлении программы автомата учитывалась необходимость его установки в исходное состояние при подаче сигнала сброса (выражение **IF** (Reset='1') **THEN** next_sreg<=Idle;).

Конечные автоматы в языке VHDL удобно описывать посредством оператора выбора **CASE**, используя в качестве ключа выбора варианта переменную состояния автомата в текущий момент времени. Внутри каждого варианта определяется состояние перехода и значения выходных сигналов, формируемых в соответствии с входными условиями. Состояние перехода из текущего состояния (назначение переменной next_sreg нового значения) задается условным оператором, логическое выражение которого совпадает с последовательностью условий, встречающихся на соответствующих путях переходов на схеме алгоритма. Аналогично определяются и выходные сигналы, вырабатываемые на переходах и задающие исполняемые в других блоках операции.

Четвертый этап. Компиляция проекта и основные параметры устройства

После создания всех фрагментов проекта и схемы проекта в целом выполняется его компиляция. Требуемый объем ОЗУ сделал целесообразным выбор в качестве БИС PLD семейства FLEX 10K. После успешной компиляции был получен файл отчета (*.rpt), показавший, что данный проект далеко не полностью использует возможности, предлагаемые самым младшим представителем семейства EPF10K10. Общие затраты БИС на реализацию проекта компилятор определил как 7%. Правда, на реализацию модуля ОЗУ компилятор использовал 33% имеющихся у БИС ресурсов.

Пятый этап. Тестирование проекта

Тестирование проекта также выполнялось средствами САПР Quartus II. Созданная тестовая последовательность должна была проверять лишь ключевые моменты работы разработанного устройства. Результаты моделирования приведены на рис. 12.22. Дадим пояснения основным фрагментам моделирования.

1. Системное время в интервале от 0 до 0,1 мкс. Режим начального сброса устройства. Проверка всех возможных исходных ситуаций перед сбросом весьма громоздка и в данном примере эти варианты из соображений большого объема не включены.
2. Системное время в интервале от 0,1 до 0,3 мкс. Режим записи в ОЗУ по адресу 04 (здесь, как и далее, значения всех адресов и данных будут даны в шестнадцатеричной системе счисления) содержимого, равного 45.
3. Системное время в интервале от 0,3 до 0,55 мкс. Режим записи в ОЗУ по адресу 05 данных, равных AA.
4. Когда системное время достигло значения 0,7 мкс, начался режим записи в сдвигающий регистр содержимого ОЗУ по адресу 04. Физически запись произошла при переходе автомата sreg в состояние LdBt, т. е. приблизительно в момент времени 1,1 мкс, однако поскольку внешним признаком завершения процесса перезаписи служит появление сигнала OutReady, то именно после появления этого сигнала внешнее устройство может изменять значения адреса ОЗУ от удерживаемого ранее значения 04. Поэтому окончанием четвертого этапа тестирования можно считать момент системного времени после 1,7 мкс.
5. В качестве следующего проверяемого фрагмента алгоритма работы устройства целесообразно выбрать вывод содержимого сдвигающего регистра в форме последовательного кода (со стороны старших разрядов) на выходной контакт OutData.

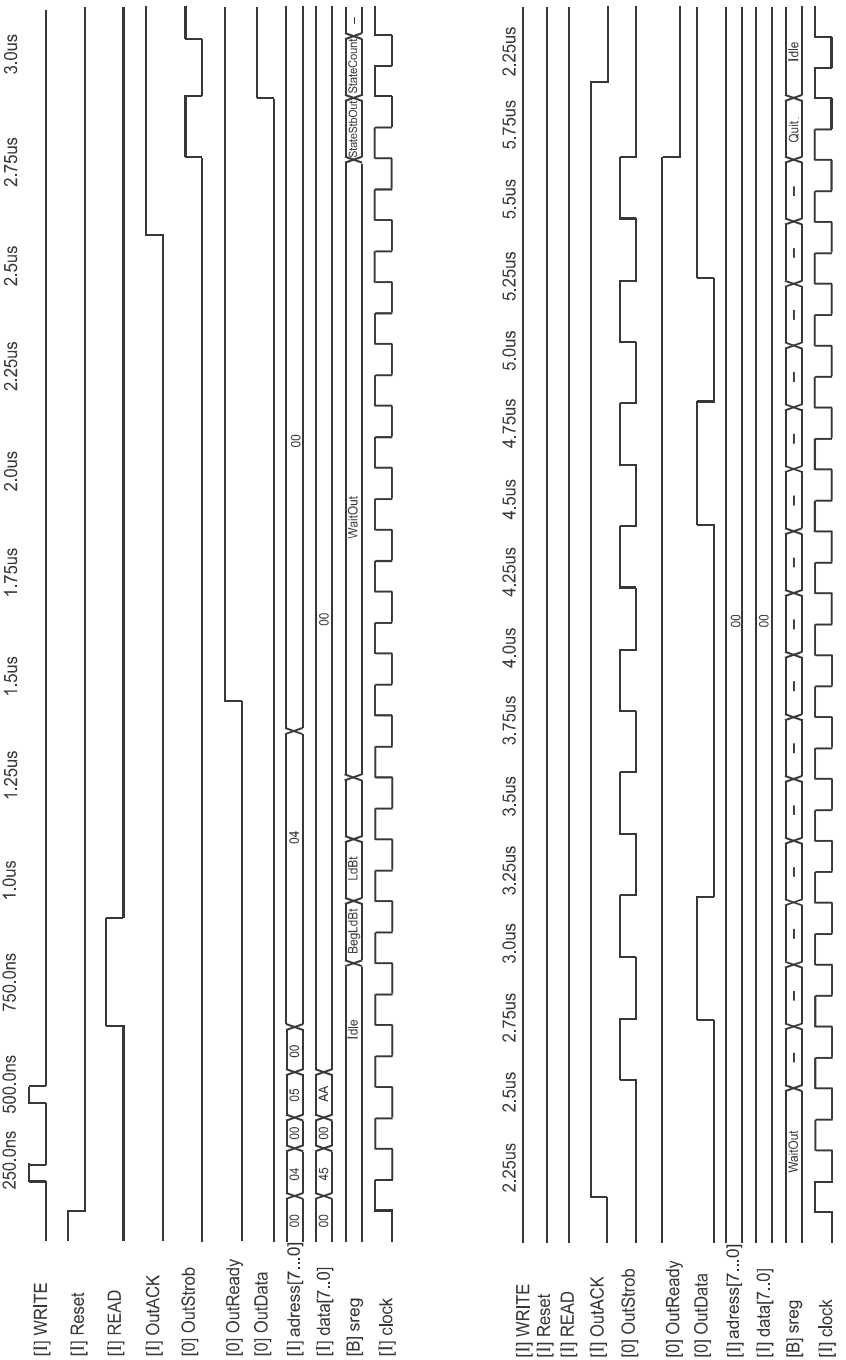


Рис. 12.22. Временные диаграммы, построенные средствами моделирования для устройства, спроектированного средствами САПР

Инициализирующим вывод сигналом является входной сигнал устройства OutAck, после его появления (в примере это момент времени 2,25 мкс) начинается цикл выдачи последовательного кода, каждый бит после его появления на выходном контакте через 0,2 мкс сопровождается стробирующим сигналом OutStrob. Признаком окончания цикла служит сброс сигнала OutReady (в примере это происходит после 5,7 мкс).

Анализ временных диаграмм позволяет не только проверить правильность функционирования устройства, но и исследовать временное поведение отдельных элементов проекта.

Шестой этап. Автоматическое определение временных характеристик устройства

Возможности САПР вычислять временные соотношения между различными фрагментами проекта существенно облегчают проектировщику задачу проверки правильности работы проекта во временной области. Автоматизация этого этапа проектирования избавляет от необходимости ручного перебора исходных данных проекта с целью обнаружения отклонений от допустимых временных установок.

Седьмой этап. Практическое использование результатов проектирования

Основным результатом работы компилятора является файл конфигурации БИС, соответствующий техническому заданию. Средства САПР позволяют на заключительных этапах работы поместить содержимое этого файла в интересующую проектировщика БИС, и на этом процесс проектирования может быть переведен в плоскость натурных экспериментов. Эксперименты могут производиться как с конечной продукцией, так и с прототипными средствами различных типов.

Контрольные вопросы и упражнения

1. В какой последовательности может выполняться проектирование электронной аппаратуры и как называются возможные варианты таких последовательностей?
2. Какие средства могут применяться для реализации проектов цифровых устройств?
3. Какие средства комплексной отладки современных проектов вам известны?
4. Дайте сравнительный анализ различных способов описаний проектов.
5. Как выполняется тестирование проектов на различных этапах их проектирования?

6. Какие типы стандартных элементов используются при проектировании цифровых устройств?
7. Назовите основные блоки, из которых строится программа, описывающая поведение цифрового автомата?
8. Какие понятия необходимо добавить к языкам описания цифровых устройств для применения к смешанным (аналого-цифровым) системам?
9. Какие основные этапы должен выполнить разработчик при создании современной системы на ПЛИС?
10. Что изменится в поведении проекта (листинг 12.1), если убрать оператор `EXIT`?
11. Написать на языке VHDL программу, соответствующую схеме рис. 12.23.

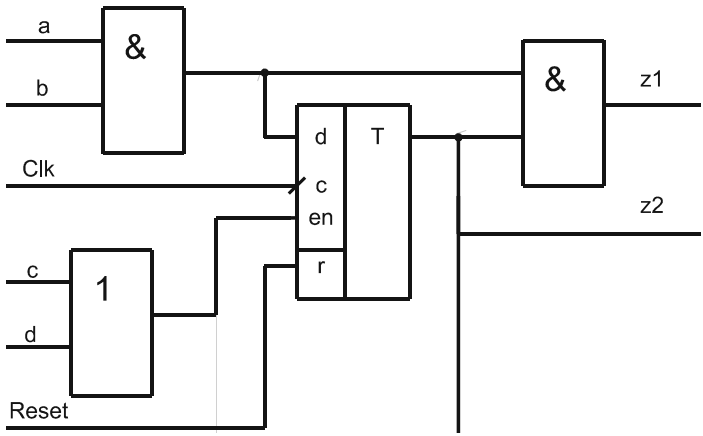


Рис. 12.23. Схема для контрольного вопроса 11

12. Написать на языке VHDL программу, соответствующую схеме рис. 12.24.

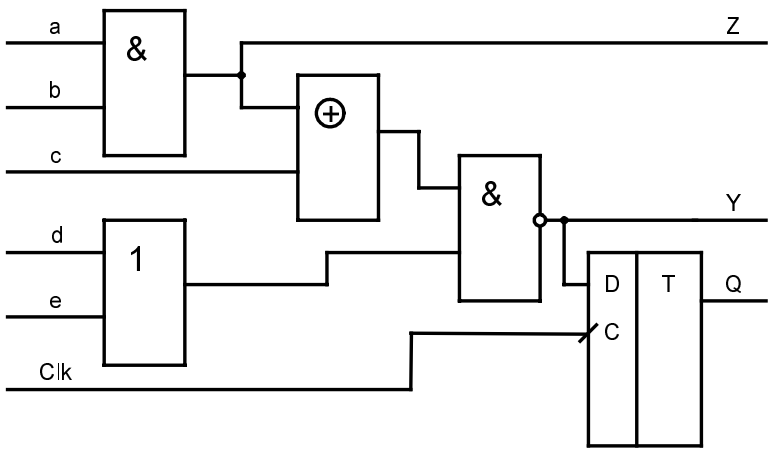


Рис. 12.24. Схема для контрольного вопроса 12

13. Написать на языке VHDL программу, соответствующую схеме рис. 12.25.

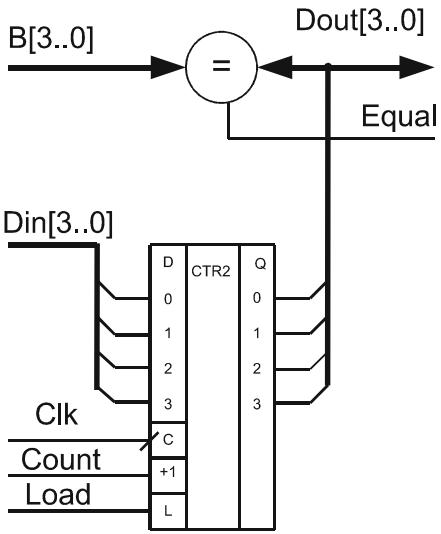


Рис. 12.25. Схема для контрольного вопроса 13

14. Написать на языке VHDL программу, соответствующую схеме рис. 12.26.

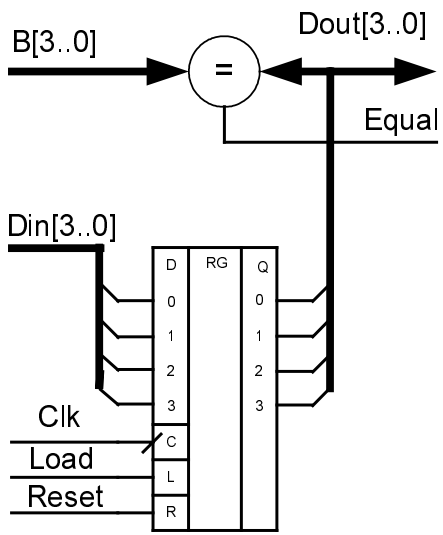


Рис. 12.26. Схема для контрольного вопроса 14

15. Написать на языке VHDL программу, соответствующую схеме рис. 12.27.

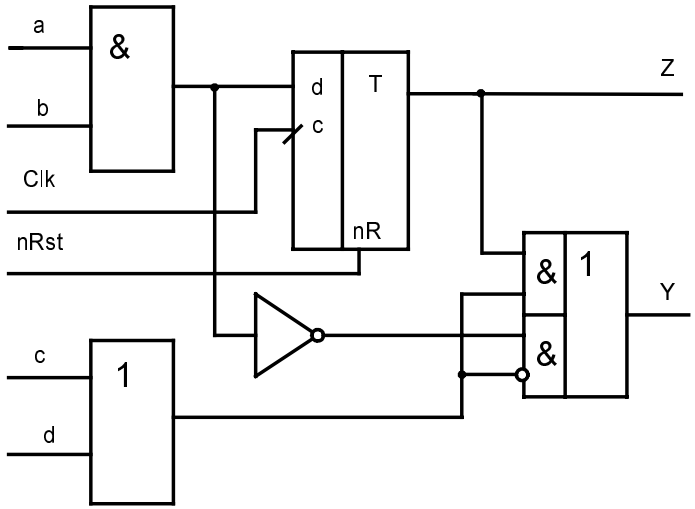


Рис. 12.27. Схема для контрольного вопроса 15

16. Написать на языке VHDL программу, соответствующую диаграмме переходов автомата, приведенной на рис. 12.28.

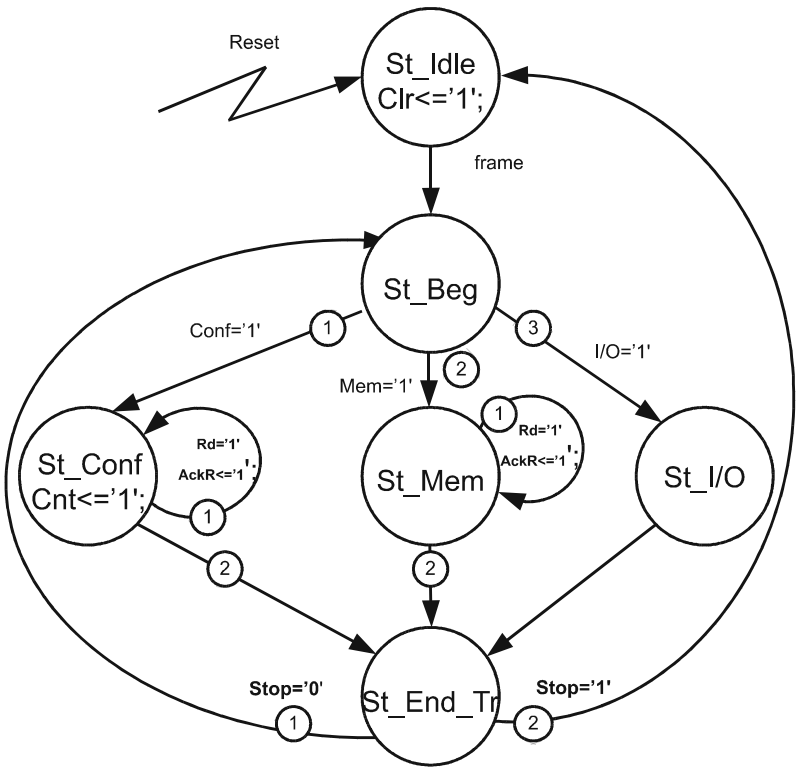


Рис. 12.28. Диаграмма переходов для контрольного вопроса 16

17. Написать на языке VHDL программу, соответствующую диаграмме переходов автомата, приведенной на рис. 12.29.

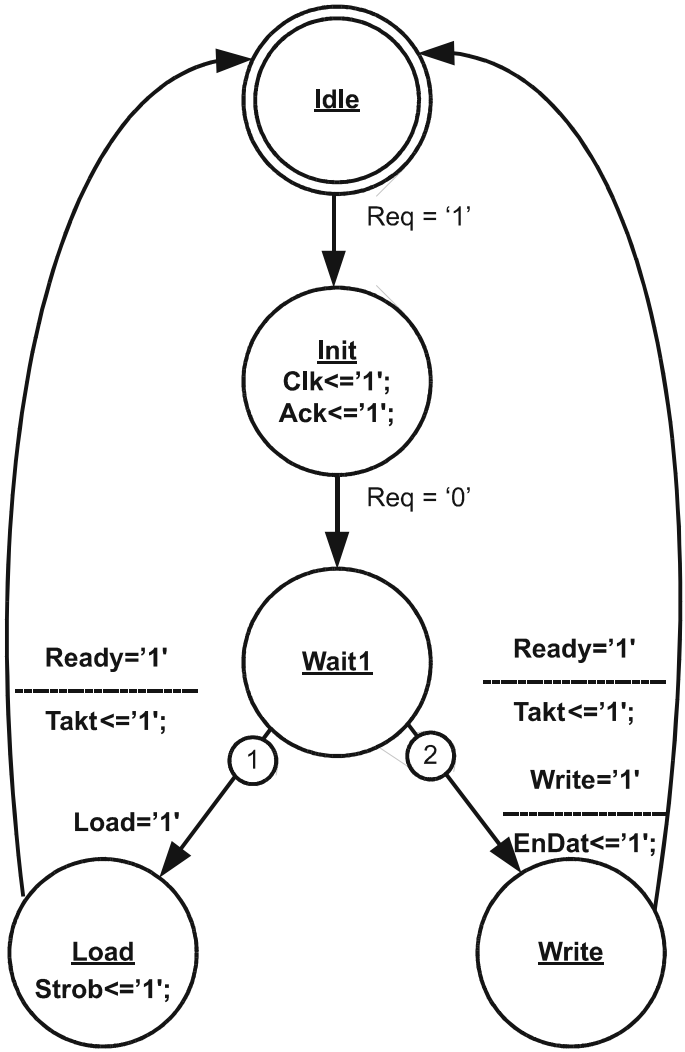


Рис. 12.29. Диаграмма переходов для контрольного вопроса 17

18. Написать на языке VHDL программу, соответствующую диаграмме переходов автомата, приведенной на рис. 12.30.

19. Написать на языке VHDL программу, соответствующую диаграмме переходов автомата, приведенной на рис. 12.31.

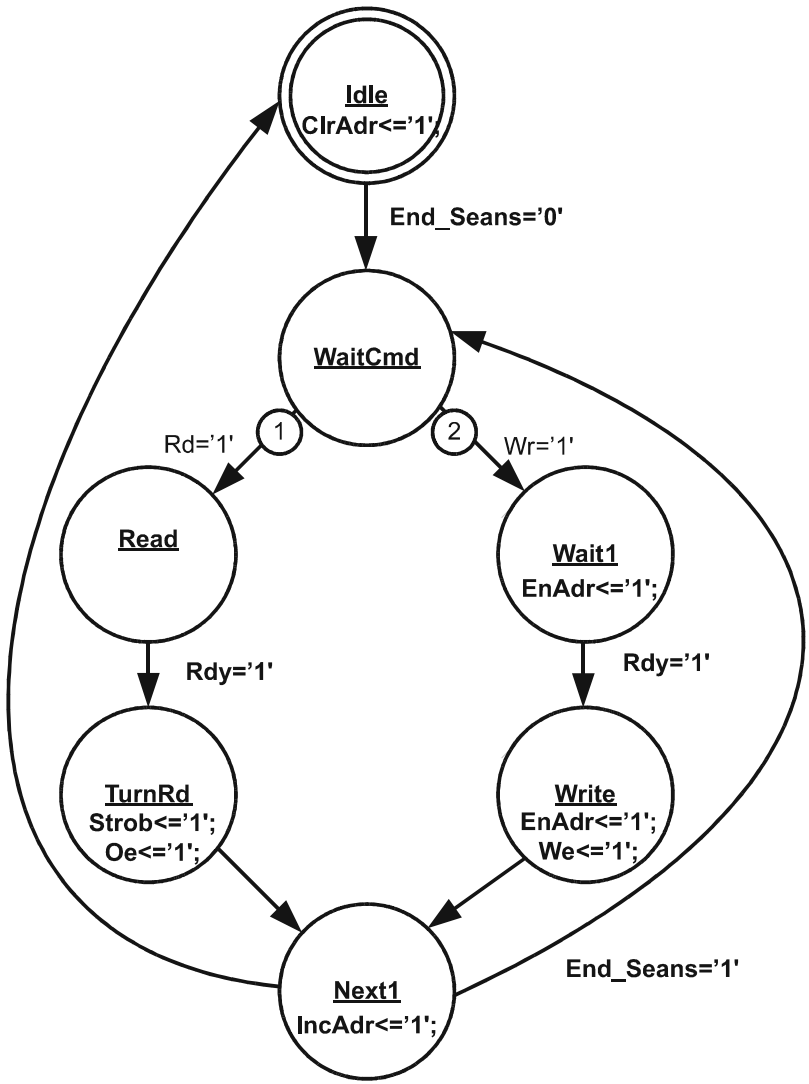


Рис. 12.30. Диаграмма переходов для контрольного вопроса 18

20. Написать на языке VHDL программу, определяющую для 8-разрядного входного кода его четность.

21. Написать на языке VHDL программу, определяющую для 8-разрядного входного кода номер первой единицы со стороны старших разрядов.

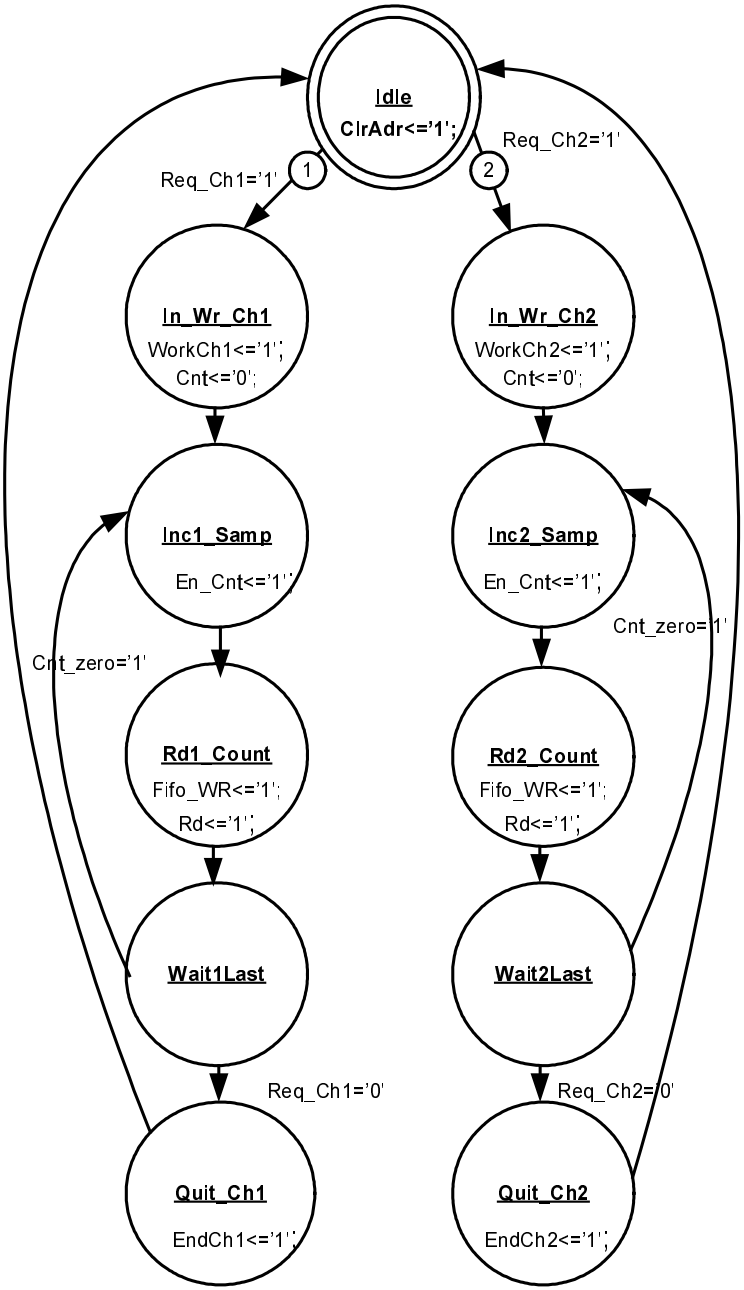


Рис. 12.31. Диаграмма переходов для контрольного вопроса 19

Литература к главе: [1], [4], [7], [8], [14], [17], [18], [21], [22], [28], [32], [35], [41], [42], [44], [54], [59], [61], [65], [II], [III], [VII], [IX], [XIII], [XVII], [XXVII], [XXVIII].

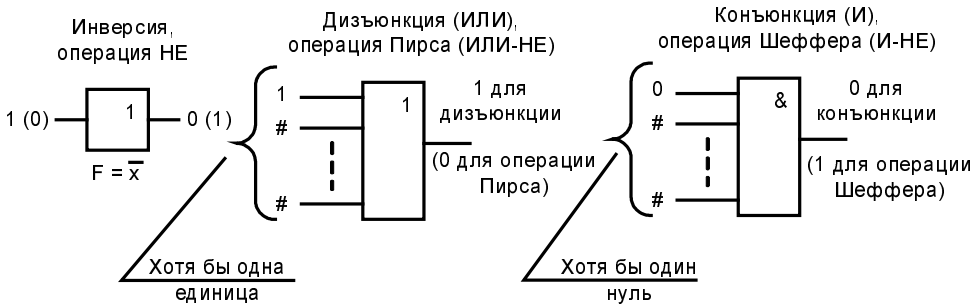
Приложение

Основные логические операции и законы

Для функций, чаще всего используемых при проектировании ЦУ, далее приведены — название, алгебраическое выражение и описание в виде "*комбинация входов* → *результат*".

- *Инверсия* (логическое отрицание, операция НЕ), $F = \overline{x}$. Результат операции имеет значение, противоположное значению аргумента (1→0, 0→1). Двойная инверсия не меняет значения переменной.
- *Конъюнкция* (логическое умножение, операция И), $F = x_1 x_2 ... x_m$. Если среди входных величин присутствует хотя бы один нуль → нуль на выходе; только единичные сигналы на всех входах приводят выход к единичному значению, т. е. $F = \min(x_1,...,x_n)$.
- *Дизъюнкция* (логическое сложение, операция ИЛИ), $F = x_1 \vee x_2 \vee ... \vee x_m$. Хотя бы одна единица среди входных величин → единица на выходе; только нули на всех входах делают выход нулевым, т. е. $F = \mathrm{Max}(x_1,...,x_n)$.
- *Операция Шеффера* (операция И-НЕ), $F = \overline{x_1 x_2 ... x_m}$. Хотя бы один нуль среди входов → на выходе единица; на всех входах единицы → на выходе нуль.
- *Операция Пирса* (операция ИЛИ-НЕ), $F = \overline{x_1 \vee x_2 \vee ... \vee x_m}$. Хотя бы одна единица среди входов → на выходе нуль; на всех входах нули → на выходе единица.

  Функционирование инвертора, элементов ИЛИ, ИЛИ-НЕ, И, И-НЕ графически иллюстрируется на рис. П1.1.
- *Неравнозначность* (исключающее ИЛИ, сложение по модулю 2, нечетность), $F = x_1 \oplus x_2 \oplus ... \oplus x_m$, при нечетном числе единиц среди входных сигналов выход имеет единичное значение, иначе нулевое.
- *Равнозначность* (четность), $F = \overline{x_1 \oplus x_2 \oplus ... \oplus x_m}$, при четном числе единиц на входах выход имеет единичное значение, иначе нулевое.



Символ # означает произвольное значение сигнала

Рис. П1.1. Пояснения к работе основных логических элементов

При преобразованиях логических формул используется ряд аксиом и теорем. Часто применяются законы (теоремы) де Моргана, поглощения и склеивания, а также разложение функций по Шеннону.

Закон де Моргана. При определении операций конъюнкции и дизъюнкции как дуальных, этот закон формулируется в следующем виде — *инверсия функции равна дуальной функции от инверсий переменных*. Применительно к функциям двух переменных закон де Моргана (справедливый для функций любого числа переменных) записывается следующим образом:

$$\overline{x_1 \vee x_2} = \overline{x}_1\overline{x}_2, \quad \overline{x_1 x_2} = \overline{x}_1 \vee \overline{x}_2.$$

Закон поглощения. Этот закон формулируется так:

$$x_1 \vee x_1 x_2 = x_1, \quad x_1(x_1 \vee x_2) = x_1.$$

Закон склеивания. Закон имеет вид:

$$x_1 x_2 \vee x_1 \overline{x}_2 = x_1, \quad (x_1 \vee x_2)(x_1 \vee x_2) = x_1.$$

Разложение по Шеннону. Это разложение можно произвести по разному числу переменных. Разложение по одному аргументу имеет вид:

$$F(x_0, x_1, \ldots x_{n-1}) = \overline{x}_0 F(0, x_1 \ldots x_{n-1}) \vee x_0 F(1, x_1, \ldots x_{n-1}).$$

Справедливость такого разложения видна непосредственно из подстановки в него значений $x_0 = 0$ и $x_0 = 1$.

Разложение функции по k аргументам записывается в виде:

$$F = \overline{x}_0\overline{x}_1 \ldots \overline{x}_{k-2}\overline{x}_{k-1}F_0 \vee \overline{x}_0\overline{x}_1 \ldots \overline{x}_{k-2}x_{k-1}F_1 \vee \ldots \vee x_0x_1 \ldots x_{k-2}x_{k-1}F_{2^k-1},$$

где

$$F_0 = F(0,0,\ldots,0,0,x_k,\ldots,x_{n-1}),$$

$$F_1 = F(0,0,\ldots,0,1,x_k,\ldots,x_{n-1}),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$F_{2^k-1} = F(1,1,\ldots,1,1,x_k,\ldots,x_{n-1}).$$

На рис. П1.2 показаны основные логические элементы и их эквиваленты, получаемые в соответствии с законами де Моргана.

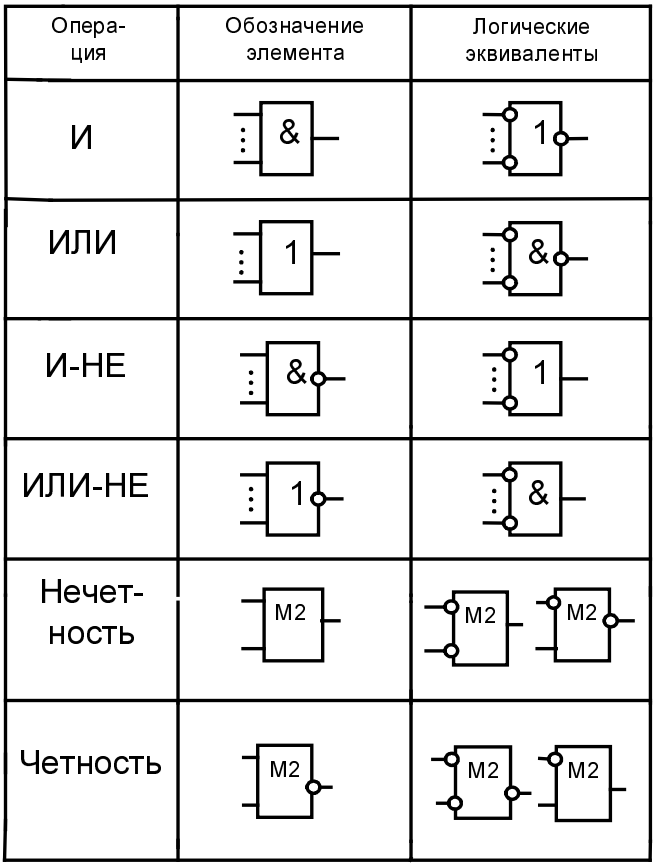


Рис. П1.2. Условные обозначения основных элементов одноступенчатой логики

Формы представления логических функций. Любую логическую функцию можно представить *дизъюнктивной нормальной формой* (ДНФ) или *конъюнктивной нормальной формой* (КНФ). Обе формы принципиально равноценны, но более наглядна ДНФ (в английской терминологии *SOP — Sum-Of-Products*, т. е. логическая сумма логических произведений). Произведения формируются из *литералов* переменных, т. е. их прямых и инверсных значений, и являются элементарными конъюнкциями, в которых операция инверсии может применяться только для отдельных переменных. При наличии как прямых, так и инверсных значений аргументов, ДНФ реализуется двухъярусной схемой И-ИЛИ.

По схемотехническим соображениям чаще используются альтернативные варианты двухъярусных схем из *элементов Шеффера (И-НЕ) или Пирса (ИЛИ-НЕ)*, более простых и быстродействующих, чем элементы И и ИЛИ (рис. П1.3).

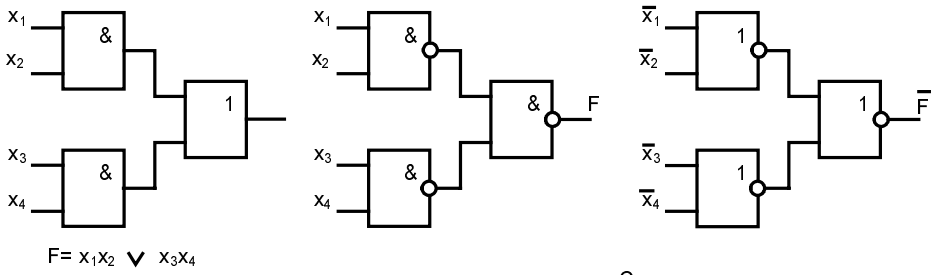


Рис. П1.3. Варианты реализации функции И-ИЛИ (дизъюнктивных нормальных форм)

Схема на элементах И-НЕ преобразуется к виду И-ИЛИ путем замены выходного вентиля его логическим эквивалентом с последующим исключением двойной инверсии в линиях связи первого яруса со вторым. Схема на элементах ИЛИ-НЕ получается переходом к логическим эквивалентам в первом ярусе.

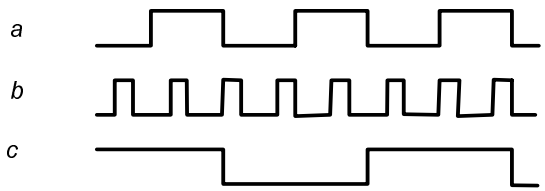
Многоуровневые логические выражения и схемы соответствуют произвольному размещению булевых операторов в выражениях для логических функций, реализуемому посредством введения скобок. Функция может быть представлена *факторизованной формой*, составленной из литералов, дизъюнктивных и конъюнктивных нормальных форм. Для одной и той же функции могут существовать многие варианты многоуровневых выражений. Применение многоуровневых представлений логических функций зачастую может дать более экономичные по числу элементов схемы их воспроизведения при увеличении задержки выработки результата.

Одни и те же преобразования логических переменных можно задать в различных формах: с помощью операций И, ИЛИ, НЕ (булевский базис), операции И-НЕ (базис Шеффера), операции ИЛИ-НЕ (базис Пирса), а также и другими способами. Выбор базиса зависит от простоты реализации той или иной операции схемами данной технологии. Чаще всего, как уже отмечалось, применяются *базисы Шеффера и Пирса*. В сериях цифровых ИС наряду с базовыми логическими элементами обычно имеется и ряд других, выполняющих другие операции.

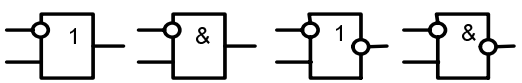
Контрольные вопросы и упражнения

1. Дайте формулировку теоремы де Моргана и примените ее к выражениям $\overline{a}\overline{b}\overline{c}$ и $\overline{a} \vee \overline{b} \vee \overline{c}$.
2. Сколько комбинаций входных сигналов создадут высокий уровень выхода у шестивходового элемента ИЛИ? А у шестивходового элемента И?

3. Логический элемент со входами *a*, *b*, *c* получает на эти входы сигналы, показанные ниже на временных диаграммах. Дополните эти диаграммы графиком выходного сигнала для двух вариантов: для логического элемента ИЛИ и для логического элемента И.



4. Из элементов, выполняющих операцию Шеффера над четырьмя переменными, составьте схему, выполняющую эту операцию для восьми переменных.
5. Сколько функциональных преобразователей с тремя входами потребуется для воспроизведения функции пяти аргументов с помощью ее разложения по Шеннону?
6. Составьте таблицу истинности для функции "сложение по модулю 2" при представлении переменных трехсимвольным алфавитом, содержащим символы 1, 0 и X (неопределенность).
7. Составьте таблицы истинности для функций, изображенных ниже графически, при представлении переменных трехсимвольным алфавитом, содержащим символы 1, 0 и X (неопределенность).



8. Нарисуйте схему воспроизведения функции $(a \vee b)(c \vee d)$ на элементах Шеффера.
9. Нарисуйте схему воспроизведения функции $(a \vee b)(c \vee d)$ на элементах Пирса.

Словарь иностранных сокращений и терминов

ABEL — язык описания аппаратуры фирмы Xilinx (низкого уровня).

ADC — *Analog-Digital Converter* — аналого-цифровой преобразователь.

AGP — *Advanced Graphics Port* — стандартный интерфейс графических систем.

ALM — *Adaptive Logic Module* — логический блок, применяемый в микросхемах FPGA фирмы Altera.

ASCII — *American Standard Code for Information Interchange* — стандартный американский код обмена информацией в последовательных интерфейсах.

AVR — семейство микроконтроллеров фирмы Atmel.

AHDL — язык описания аппаратуры фирмы Altera.

ASICs — *Application Specific Integrated Circuits* — специализированные ИС, изготовляемые тем или иным способом по индивидуальному техническому заданию (для конкретного проекта).

ASMBL — Application-Specific Modular Block — логический блок, применяемый в микросхемах FPGA фирмыXilinx.

BEDORAM — *Burst Extended Data Out RAM* — вариант динамических ОЗУ, близкий к EDORAM и отличающийся от него пакетным доступом к данным, позволяющим сократить цикл обращения внутри пакета.

BSCs — *Boundary Scan Cells* — ячейки граничного сканирования, т. е. дополнительные схемы в составе БИС/СБИС, обеспечивающие реализуемость их тестирования по интерфейсу JTAG.

BST — *Boundary Scan Testing* — граничное сканирование, т. е. тестирование БИС/СБИС по интерфейсу JTAG.

CD — *Coder* — шифратор.

CDR — *Clock-Data Recovery* — автоматическая взаимная временная подстройка передаваемых данных и синхросигналов.

CDRAM — *Cached DRAM* — динамическое ОЗУ повышенного быстродействия, достигаемого путем кэширования.

CISC — *Complex Instruction Set Computer* — архитектура процессора с полным набором команд.

Clock Boost — умножение частоты тактовых импульсов, одна из функций, выполняемых блоками PLL или DLL.

Clock Lock — коррекция временного положения тактовых импульсов, одна из функций, выполняемых блоками PLL или DLL.

Clock Skew — временной сдвиг тактового импульса относительно заданного положения, вызванный паразитными задержками в цепях тактирования.

CPLD — *Complex PLD* — БИС/СБИС программируемой логики, структура которой представляет собою совокупность блоков типа PAL или GAL, объединенных матрицей программируемых соединений. Программируется пользователем.

CRC — *Cyclic-Redundance-Check Code* — циклический избыточный корректирующий код.

CRU — *Carry Unit* — блок ускоренных переносов.

CSoC — *Configurable System on Chip* — конфигурируемая система на кристалле.

DAC — *Digital-Analog Converter* — цифроаналоговый преобразователь.

DC — *Decoder* — дешифратор.

DCM — *Digital Clock Manager* — блок управления параметрами синхросигналов.

DDR — *Double Data Rate* — передача сигналов с двойной скоростью.

DIMM — *Dual In Line Memory Module* — модуль памяти с двухрядным расположением выводов.

DLL — *Delay Locked Loop* — схема коррекции параметров синхросигналов.

DMA — *Direct Memory Access* — прямой доступ к памяти.

DMUX — *Demultiplexer* — демультиплексор.

DRAM — *Dynamic RAM* — динамическое ОЗУ.

DSP — *Digital Signal Processing* — цифровая обработка сигналов.

EAB — *Embedded Array Block* — встроенный блок памяти/обработки в микросхемах программируемой логики.

EDIF — *Electronic Design Interchange Format* — формат обмена проектов при разработке электронных схем. Список цепей в этом стандарте может быть получен из описаний проекта на языках VHLD или Verilog HDL с помощью стандартных программ. Файлы в формате EDIF могут формироваться пакетами программных средств ряда САПР для целей моделирования с помощью стандартного пакета моделирования EDIF.

EDORAM — *Extended Data Out RAM* — вариант динамического ОЗУ повышенного быстродействия.

EEPROM — *Electrically Erasable Read Only Memory* — электрически стираемое репрограммируемое ЗУ.

EPROM — *Electrically Programmable Read Only Memory* — репрограммируемое ЗУ с ультрафиолетовым стиранием данных.

EPROM-OTP — однократно программируемое ЗУ с программированием состояний плавающих затворов МОП-транзисторов.

ESB — *Embedded System Block* — встроенный системный блок памяти/обработки в микросхемах программируемой логики.

FACM — *Full Associative Cache Memory* — полностью ассоциативная кэш-память.

FCRAM — *Fast Cycle Random-Access Memory* — динамическое ЗУ с сокращенной длительностью цикла.

FIFO — *First-In – First-Out* — ЗУ с последовательным доступом к данным типа "очередь" (по правилу "первый вошел — первый вышел").

FRAM — *Ferroelectric RAM* — ферроэлектрическое ОЗУ.

FPGA — *Field Programmable Gate Array* — БИС/СБИС программируемой логики, структура которой представляет собой матрицу программируемых логических блоков, между строками и столбцами которой реализованы программируемые соединения. Программируется пользователем.

FPM — *Fast Page Mode* — быстрый страничный доступ.

FSM — *Finite State Machine* — конечный автомат.

GA — *Gate Array* — базовый матричный кристалл (вентильная матрица).

GRM — *General Routing Matrix* — главная матрица соединений.

HDL — *Hardware Description Language* — язык описания аппаратуры.

HSTL — *High Speed Transceiver Logic* — стандарт сигналов интерфейса.

Hit — сигнал "попадание" в схемах кэш-памяти, свидетельствующий о наличии запрашиваемой единицы информации в этой памяти.

I^2C — *Inter Integrated Circuits* — интерфейс синхронного последовательного обмена для соединения между микроЭВМ или между микроЭВМ и периферийными устройствами.

IP — *Intellectual Property* — интеллектуальная собственность, термин, принятый для наименования программных ядер (soft-ядер) микросхем с программируемыми структурами.

ISP — *In-System Programmable* — программируемость микросхемы непосредственно в системе.

JEDEC — *Joint Electronic Device Engineering Council* — объединенный инженерный совет по электронным устройствам, в области программируемой логики. Термин обозначает текстовый файл, содержащий информацию о программировании схемы в стандартной форме JEDEC.

JTAG — *Joint Test Action Group* — объединенная группа по вопросам тестирования, по имени которой названы методы тестирования БИС/СБИС без физического доступа к каждому их выводу и программирования микросхем программируемой логики с помощью JTAG-интерфейса.

LAB — *Logic Array Block* — логический блок микросхем программируемой логики.

LIFO — *Last-In – Fist-Out* — ЗУ с последовательным доступом к данным стекового типа (по правилу "последний вошел — первый вышел").

LPM — *Library of Parameterized Modules* — библиотека параметризируемых модулей.

LUT — *Look Up Table* — табличный функциональный преобразователь, логический блок программируемых микросхем, реализованный на основе схемы памяти.

LVCMOS — *Low Volt CMOS* — стандарт сигналов низковольтного интерфейса для КМОП-схем.

LVDS — *Low Volt Differential Signaling* — стандарт быстродействующего низковольтного дифференциального интерфейса.

LVPECL — *Low Volt Positive ECL* — стандарт низковольтного интерфейса для схем положительной эмиттерно-связанной логики.

LVTL — *Low Volt Transistor Logic* — стандарт интерфейса для низковольтной транзисторной логики.

MAC — *Multiply-Accumulation* — операция умножения с накоплением, широко применяемая при цифровой обработке сигналов.

MAC — *Media Access Controllers* — блоки, входящие в состав средств обеспечения интерфейса Ethernet.

MDAC — *Multiplying Digital-Analog Converter* — умножающий цифроаналоговый преобразователь.

MDRAM — *Multibank DRAM* — многобанковая динамическая память, вариант повышения быстродействия ЗУ с помощью разбиения памяти на части (банки), что при кучности адресов последовательных обращений к памяти позволяет обращаться к банкам поочередно с более высокой скоростью.

MIPS — *Mega Instructions Per Second* — миллион команд в секунду, мера производительности процессора.

MIPS — *Microprocessor without Interlocked Pipeline Stages* — название известной фирмы, означающее, в сущности, "микропроцессор без задержек ожидания конвейера".

MLC — *Multilevel Cell* — многоуровневая ячейка памяти, позволяющая хранить в ней более одного бита.

MPL — *Multiplier* — умножитель.

MRAM — *Magnetic RAM* — перспективное ОЗУ, работа которого основана на применении магнитных материалов.

MUX — *Multiplexer* — мультиплексор.

NoBL — *No Bus Latency* — вариант ОЗУ с устранением задержки при реверсе шин.

NtRAM — *No Turnaround RAM* — вариант ОЗУ с устранением задержки при реверсе шин.

NvSRAM — *Non-Volatile SRAM* — статическое ОЗУ со свойством энергонезависимости.

OUM — *Ovonics Unified Memory* — перспективное ЗУ фирмы Ovonics, работа которого основана на применении материалов с изменяемыми фазовыми состояниями.

OTP — *One-Time Programmable* — "однократно программируемая", определение относится к микросхемам памяти типа РПЗУ-УФ, корпус которых не имеет прозрачного окна для стирания данных путем воздействия на кристалл ультрафиолетовым облучением.

PAL — *Programmable Array Logic* — см. Программируемая матричная логика.

PCI — *Peripheral Component Interconnect* — популярная шина расширения для соединения периферийных компонентов системы.

PCI Express — интерфейс с последовательными линиями передач высокого быстродействия.

PFRAM — *Polimeric Ferroelectric Random Access Memory* — перспективная память полимерно-ферроэлектрического типа.

PIP — *Programmable Interconnect Point* — программируемая точка соединений.

PLA — *Programmable Logic Array* — см. Программируемая логическая матрица.

PLD — *Programmable Logic Device* — общее наименование для схем PAL и PLA.

PLDASM — язык описания аппаратуры фирмы INTEL (низкого уровня).

PLL — *Phase Locked Loop* — схема следящей системы с чувствительным элементом, реагирующим на разность фаз импульсных последовательностей, используемая для управления временными параметрами синхросигналов цифровых устройств.

PRCD — *Priority Coder* — шифратор приоритетов.

PREP — *Programmable Electronics Performance Corporation* — консорциум компаний, предложивший набор эталонных схем и методику оценки сложности БИС/СБИС программируемой логики.

PROM — *Programmable Read-Only Memory* — однократно программируемая постоянная память.

PSoC — Programmable System On Chip — программируемая система на кристалле.

RAM — *Random-Access Memory* — оперативное ЗУ (ЗУ с произвольным доступом).

RDRAM — *Rambus DRAM* — динамическое ОЗУ высокого быстродействия, разработанное фирмой Rambus.

RISC — *Reduced Instruction Set Computer* — вариант архитектуры быстродействующих процессоров с сокращенным набором команд.

RLDRAM — *Reduced Latency DRAM* — быстродействующее динамическое ОЗУ.

ROM — *Read-Only Memory* — постоянная память (память с рабочим режимом только для чтения).

RTL — *Register Transfer Level* — уровень регистровых передач.

SDR — *Single Data Rate* — передача данных с одинарной скоростью.

SDRAM — *Synchronous DRAM* — синхронное ОЗУ динамического типа высокого быстродействия, в котором высокий темп передачи данных обеспечивается конвейерной организацией тракта передачи, тактируемого от синхросигналов, общих для процессора и памяти. Широко применяется в современных компьютерах.

SERDES — *Serialiser-Deserialiser* — блок преобразования параллельных данных в последовательные и наоборот.

SOPC — *System On Progammable Chip* — БИС/СБИС программируемой логики высшего уровня сложности, на которой можно реализовать целую систему.

SOI — *Silicon On Insulator* — схемотехнология интегральных схем, обеспечивающая минимальность паразитных параметров схемы, что, в конечном счете, приводит к улучшению ее технических характеристик.

SPI — *Serial Peripheral Interconnect* — синхронный последовательный интерфейс.

SPLD — *Simple Programmable Logic Device* — программируемая логическая схема невысокой сложности.

SRAM — *Static RAM* — статическое ОЗУ.

SSTL — *Stub Series Terminated Logic* — стандарт сигналов интерфейса, используемый для микросхем быстродействующих динамических ОЗУ.

StrataFlash — запоминающее устройство типа флэш с запоминанием двух битов в одном запоминающем элементе с помощью многоуровневого заряда плавающих затворов ЛИЗМОП-транзисторов.

TAP — *Test Access Port* — порт тестирования, четыре или пять выводов БИС/СБИС, специально выделенных для тестирования по интерфейсу JTAG.

UART — *Universal Asynchronous Receiver – Transmitter* — программируемый связной адаптер, реализующий асинхронные протоколы передачи последовательных данных.

Verilog HDL — язык описания аппаратуры фирмы Cadence. Наряду с языком VHDL относится к самым популярным языкам описания аппаратуры высокого уровня.

VHDL — *Very-High-Speed Hardware Description Language* — язык описания аппаратуры, стандарт IEEE, по-видимому, наиболее популярный язык описания аппаратуры высокого уровня.

VLIW — *Very Long Instruction Word* — архитектура процессора с очень длинными словами команд.

WDT — *Watchdog Timer* — сторожевой таймер.

XACT — пакет программных средств для проектирования БИС/СБИС программируемой логики фирмы Xilinx.

ZBT — *Zero Bus Turnaround* — вариант ОЗУ с устранением задержки реверса шины.

Принятые сокращения

А	адрес
АЗУ (CAM)	ассоциативное ЗУ
АП	автомат с памятью
АП	адресное пространство
АЦП (ADC)	аналого-цифровой преобразователь
БВВ (IOB)	блок ввода/вывода
БиКМОП	схемотехнология, сочетающая использование биполярных и МОП-транзисторов
БИС (LSI)	большая интегральная схема
БИСМ	БИС, реализованная на основе БМК
БМК (GA)	базовый матричный кристалл
БЯ	базовая ячейка
ВБП	встроенный блок памяти
ВУ	внешнее устройство
ГПСП	генератор псевдослучайных последовательностей
ГПСЧ	генератор псевдослучайных чисел
ДКОИ-8	восьмиразрядный двоичный код обмена информацией
ДНФ (SOP)	дизъюнктивная нормальная форма
ДОЗУ (DRAM)	динамическое ОЗУ
ЖКИ	жидкокристаллический индикатор
ЗГ	задающий генератор системы синхронизации
ЗУ	запоминающее устройство
ЗУПВ	запоминающее устройство с произвольной выборкой
ЗЭ	запоминающий элемент
ЗЯ	запоминающая ячейка
ИС	интегральная схема

ИСПС	интегральная схема с перестраиваемой структурой
КЛБ (LAB)	конфигурируемый логический блок
КМОП (CMOS)	комплементарная МОП-структура
КОИ-7	семиразрядный код обмена информацией
КПДП (DMA)	контроллер прямого доступа к памяти
КЦ	командный цикл
ЛБ (LB)	логический блок
ЛВ	линия выборки (словарная)
ЛЗС	линия записи-считывания (разрядная)
ЛИЗМОП	МОП-структура с лавинной инжекцией заряда
ЛПМС	локальная программируемая матрица соединений
ЛФ	логическая функция
ЛЭ (LE)	логический элемент
М	модуль счета
М	информационная емкость ЗУ
МАБИС	матричная БИС
МБ	множительный блок
МБЯ	матричная базовая ячейка
МИС (LSI)	малая интегральная схема
МК (MC)	микроконтроллер
МНОП	структура "металл-нитрид-оксид-полупроводник"
МОП (MOS)	структура "металл-оксид-полупроводник"
МП (MP)	микропроцессор
МПК	микропроцессорный комплект
МПС (MCS)	микропроцессорная система
МРТ	матрица распределения термов
МСБ	множительно-суммирующий блок
МЦ	машинный цикл
МЭТ	многоэмиттерный транзистор
ОБ	операционный блок
ОЗУ (RAM)	оперативное ЗУ

ОК (OC)	открытый коллектор
ОС	обратная связь
ОС	открытый сток
ПАИС	программируемая аналоговая интегральная схема
ПБ	переключательный блок
ПБЯ	периферийная базовая ячейка
ПДП (DMA)	прямой доступ к памяти
ПЗУ (ROM)	постоянное ЗУ
ПЗУМ	постоянное ЗУ с масочным программированием
ПИТ (PIT)	программируемый интервальный таймер
ПКП (PIC)	программируемый контроллер прерываний
ПЛ	программируемая логика
ПЛИС (PLD)	программируемая логическая интегральная схема
ПЛМ (PLA)	программируемая логическая матрица
ПЛУ (PLD)	программируемое логическое устройство
ПМЛ (PAL)	программируемая матричная логика
ПМС (PIA)	программируемая матрица соединений
ППА (PPI)	программируемый параллельный адаптер
ППВМ (FPGA)	программируемая пользователем вентильная матрица
ППЗУ (PROM)	программируемое постоянное ЗУ
ПС (BS)	периферийное сканирование
ПСА (PCI)	программируемый связной адаптер
ПТ (TAP)	порт тестирования (для интерфейса JTAG)
ПТС (PIP)	программируемая точка связи
ПЯ	периферийная ячейка
РВВ	регистр ввода/вывода
РИ	распределитель импульсов
РМП	реконфигурируемая матрица памяти
РОН	регистр общего назначения
РПЗУ-УФ (EPROM)	репрограммируемое ПЗУ со стиранием данных ультрафиолетовыми лучами

РПЗУ-ЭС (EEPROM)	репрограммируемое ПЗУ с электрическим стиранием данных
РСС	регистр слова состояния
РТС	распределитель тактовых сигналов
РУ	распределитель уровней
РУС	регистр управляющего слова
САПР	система автоматизированного проектирования
СБ	связной блок
СБИС (VLSI)	сверхбольшая интегральная схема
СБИС ПЛ	СБИС программируемой логики
СДНФ	совершенная дизъюнктивная нормальная форма
СИС (MSI)	средняя интегральная схема
СнПК	система на программируемом кристалле
СОЗУ (SRAM)	статическое ОЗУ
СпИС (ASIC)	специализированная ИС
СПЛИС (CPLD)	сложная программируемая логическая ИС
ССИ (SSI)	семисегментный индикатор
СФ-блок (IP)	сложный функциональный блок
Т	транзистор
Т	такт
Т	триггер
ТЗ	техническое задание
ТИ (CLK)	тактовые импульсы
ТС	третье состояние логического элемента
ТТЛ (TTL)	транзисторно-транзисторная логика
ТТЛШ (TTLS)	ТТЛ с диодами Шотки
УА	управляющий автомат
УАПП (UART)	универсальный асинхронный приемопередатчик
УВВ	устройство ввода/вывода
УЛМ	универсальный логический модуль
УС (CW)	управляющее слово

УСАПП (USART)	универсальный синхронно-асинхронный приемопередатчик
УУ	устройство управления
ФБ	функциональный блок
ФЯ	функциональная ячейка
ЦАП (DAC)	цифроаналоговый преобразователь
ЦОС (DSP)	цифровая обработка сигналов
ЦУ	цифровое устройство
ША (AB)	шина адреса
ШД (DB)	шина данных
ШУ (CB)	шина управления
ШФ	шинный формирователь
ЭВ	эквивалентный вентиль
ЭСЛ (ECL)	эмиттерно-связанная логика

Литература и источники информации в Интернете

Краткая библиография

1. Автоматизация проектирования БИС. В 6 кн.: Практ. пособие. /Под ред. Казеннова. — М.: Высшая школа, 1990.
2. Адамов Д. и др. БМК компании UniqueICS // Электроника: Наука, Технология, Бизнес, 7/2005.
3. Алексенко А. Г. Основы микросхемотехники. — 3-е изд., перераб. и доп. — М.: ЮНИМЕДИАСТАЙЛ, 2002. — 448 с.
4. Антонов А. П. Язык описания цифровых устройств AlteraHDL. Практический курс. — М.: ИП РадиоСофт, 2001. — 224 с.
5. Ашихмин А. С. Цифровая схемотехника. Современный подход. — М.: ТехБук, 2007. — 288 с.
6. Бабич Н. П., Жуков И. А. Компьютерная схемотехника. — Киев: МК-пресс, 2004. — 670 с.
7. Бибило П. Н. Основы языка VHDL. — М.: Солон-Р, 1999. — 200 с.
8. Бибило П. Н. Системы проектирования интегральных схем на основе языка VHDL. StateCad, ModelSim, LeonardoSpectrum. — М.: СОЛОН-ПрессР, 2005. — 384 с.
9. Бирюков С. А. Применение цифровых микросхем серий ТТЛ и КМОП. — 2-е изд., стер. — М.: ДМК, 2000. — 240 с.
10. Бойко В. И. и др. Схемотехника электронных систем. Цифровые устройства. — СПб.: БХВ-Петербург, 2004. — 512 с.
11. Бойко В. И. и др. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры. — СПб.: БХВ-Петербург, 2004. — 464 с.
12. Бродин В. Б., Калинин А. В. Системы на микроконтроллерах и БИС программируемой логики. — М.: Издательство ЭКОМ, 2002. — 400 с.
13. Волович Г. И. Схемотехника аналоговых и аналого-цифровых электронных устройств. — М.: ДОДЭКА-XXI, 2005. — 528 с.

14. Григорьев В. Л. Программное обеспечение микропроцессорных систем. — М.: Энергоатомиздат, 1983. — 208 с.

15. Грушвицкий Р., Ильин И., Михайлов М. Метод граничного сканирования для смешанных сигналов // Компоненты и технологии. 2006, № 8, № 9.

16. Грушвицкий Р., Ильин И., Михайлов М. Граничное сканирование современных высокоскоростных соединений // Компоненты и технологии. 2006, № 11, № 12.

17. Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. Проектирование систем на микросхемах с программируемой структурой. — 2-е изд., — СПб.: БХВ-Петербург, 2006. — 736 с.

18. Грушин С. И., Душутин И. Д., Мелехин В. Ф. Проектирование аппаратных средств микропроцессорных систем: Учеб. пособие. — Л.: ЛПИ им. Калинина, 1990. — 78 с.

19. Гук М. Ю. Аппаратные средства IBM PC: Энциклопедия. — 3-е изд. — СПб.: Питер, 2005. — 928 с.

20. Евстифеев А. В. Микроконтроллеры AVR семейства Classic фирмы Atmel. — М.: Издательский дом Додэка-XXI, 2002. — 288 с.

21. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. — М.: Горячая линия — Телеком, 2003. — 640 с.

22. Комолов Д. А., Мяльк Р. А., Зобенко А. А., Филиппов А. С. Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II. Краткое описание и самоучитель. — М.: ИП РадиоСофт, 2002. — 352 с.

23. Кузелин М. О., Кнышев Д. А., Зотов В. Ю. Современные семейства ПЛИС фирмы Xilinx. Справочное пособие. — М.: Горячая линия — Телеком, 2004. — 440 с.

24. Лапин А. А. Интерфейсы. Выбор и реализация. — М.: Техносфера, 2005. — 168 с.

25. Лебедев О. Н. Применение микросхем памяти в электронных устройствах: Справочное пособие. — М.: Радио и связь, 1994. — 216 с.

26. Логические ИС КР1533, КР1554: Справочник: В 2-х частях / И. И. Петровский, А. В. Прибыльский, А. А. Троян, В. С. Чувелев. — М.: БИНОМ, 1993. — 496 с.

27. Мелехин В. Ф., Павловский Е. Г. Вычислительные машины, системы и сети: учебник для студ. высш. учебн. заведений. — М.: Издательский центр "Академия", 2006. — 560 с.

28. Микропроцессорные системы: Учебное пособие для вузов / Под общ. ред. Д. В. Пузанкова. — СПб.: Политехника, 2002. — 935 с.

29. Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю. М. Казаринов, В. Н. Номоконов, Г. С. Подклетнов,

Ф. В. Филиппов; Под ред. Ю. М. Казаринова. — М.: Высшая школа, 1990. — 269 с.

30. Микропроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник: в 2-х т./ Н. Н. Аверьянов, А. И. Березенко, Ю. Н. Борщенко и др. / Под ред. В. А. Шахнова. — М.: Радио и связь, 1988. — Т. 1. — 368 с. Т. 2 — 368 с.

31. Микросхемы памяти, ЦАП и АЦП: Справочник — 2-е изд. / О. Н. Лебедев, А-Й. К. Марцинкявичус, Э-А. К. Багданскис и др. — М.: КУбК-а, 1996. — 384 с.

32. Немудров В., Мартин Г. Системы на кристалле. Проектирование и развитие. — М.: Техносфера, 2004. — 216 с.

33. Новиков Ю. В. Основы цифровой схемотехники. — М.: Мир, 2001. — 379 с.

34. Новожилов О. П. Основы цифровой техники / Учебное пособие. — М.: Радиософт, 2004. — 528 с.

35. Поляков А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. — М.: СОЛОН-Пресс, 2003. — 320 с.

36. Потемкин И. С. Функциональные узлы цифровой автоматики. — М.: Энергоатомиздат, 1988. — 320 с.

37. Предко М. Руководство по микроконтроллерам: В 2-х т. — Пер. с англ. — М.: Постмаркет, 2001. — Т. 1 — 415 с., Т. 2 — 487 с.

38. Применение интегральных микросхем памяти: Справочник /А. А. Дерюгин, В. В. Цыркин, Е. В. Красовский и др./ Под ред. А. Ю. Гордонова, А. А. Дерюгина. — М.: Радио и связь, 1994. — 232 с.

39. Пухальский Г. И. Проектирование микропроцессорных систем: Учебное пособие для вузов. — СПб.: Политехника, 2001. — 544 с.

40. Пухальский Г. И., Новосельцева Т. Я. Цифровые устройства: Учебное пособие для втузов. — СПб.: Политехника, 1996. — 885 с.

41. Разевиг В. Д. Система проектирования цифровых устройств OrCad. — М.: Солон-Р, 2000. — 160 с.

42. Разевиг В. Д. Система сквозного проектирования электронных устройств DesignLab 8.0. — М.: Солон-Р, 2000. — 698 с.

43. Строганов А. Схемотехника элементов БИС // Электроника: Наука, Технология, Бизнес, 1/2003, с. 24—34.

44. Суворова Е. А., Шейнин Ю. Е. Проектирование цифровых систем на VHDL. — СПб.: БХВ-Петербург, 2003. — 576 с.

45. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. — М.: Горячая линия — Телеком, 2005. — 252 с.

46. Точчи Д., Уидмер Н. Цифровые системы. Теория и практика, 8-е изд.: Пер. с англ. — М.: Издательский дом "Вильямс", 2004. — 1024 с.

47. Угрюмов Е. П. Проектирование элементов и узлов ЭВМ: Учеб. пособие для вузов. — М.: Высшая школа, 1987. — 318 с.

48. Уэйкерли Дж. Проектирование цифровых устройств. Т. 1, Т. 2: Пер. с англ. — М.: Постмаркет, 2002. — 1087 с.

49. Хоровиц. П., Хилл У. Искусство схемотехники: Пер с англ. — 6-е изд. — М.: Мир, 2001. — 830 с.

50. Цилькер Б. Я., Орлов С. А. Организация ЭВМ и систем: Учебник для вузов. — СПб.: Питер, 2004. — 668 с.

51. Цифровые интегральные микросхемы: Справочник / П. П. Мальцев, Н. С. Долидзе, М. И. Критенко и др. — М.: Радио и связь, 1994. — 240 с.

52. Шило В. Л. Популярные цифровые микросхемы: Справочник. 2-е изд. — Челябинск: Металлургия, Челябинское отделение, 1989. — 352 с.

53. Щелкунов Н. Н., Дианов А. П. Микропроцессорные средства и системы. — М.: Радио и связь, 1989. — 282 с.

54. Asheden P. J. The designer's guide to VHDL. — San Francisco: Morgan Kaufman Publishers. — 1996. — 688 p.

55. Bursky D. Demanding Applications Push NAND Flash Densities // Electronic Design, 2006. — April 13. — p. 51—58.

56. Bursky, Evaluate Platform ASIC Options To Match The Best Silicon To Your Application // Electronic Design, 2005. — November 7. — 7 p.

57. Dally W., Poulton J. Digital Systems Engineering. Cambridge University Press, 1998. — 693 p.

58. Dipert B. Silicon Segmentation // Electronic Design News, 2003. — September 18. — p. 57 — 66.

59. Gajsky D. Principles of Digital Design. Prentice Hall, New Jersey, 1997, 447 p.

60. Johnson H., Graham M. High-Speed Digital Design. A Handbook of Black Magic. — Prentice-Hall PTR, Englewood Cliffs, New Jersey, 2002 — 446 p.

61. Kang S., Lebelevici Y. CMOS Digital Integrated Circuits. Analysis and Design. Boston, McGrow-Hill, 1999.

62. Keating M., Bricand P. Reuse Methodology Manual. Third Edition. — Kluwer Academic Publishers, 2002. — 312 p.

63. Kresta D., Johnson T. High-Level Design Methodology Comes Into Its Own//Electronic Design. — 1999. — №12 — p. 57—60.

64. Manni V. Best of Both Worlds in Parallel Digital Adders // IEEE Circuits & Devices. — Vol 18, 2002, № 5. — p. 20—23.

65. Perry D. L. VHDL: Programming by Example. — Fourth Edition. McGraw-Hill, 2002. — 497 p.

66. Rabaey J. M. Digital Integrated Circuits: A Design Perspective. — Prentice Hall, 1997. — 734 p.

67. Sachin Sapatneker. Timing. — Kluwer Academic Publishers, 2004. — 308 p.

Интернет-ресурсы

Адреса в сети Интернет, по которым публикуются сведения фирм о выпускаемых ими микросхемах и средствах автоматизированного проектирования цифровых устройств

1. **www.actel.com**
2. **www.aldec.com**
3. **www.altera.com**
4. **www.amd.com**
5. **www.arm.com**
6. **www.atmel.com**
7. **www.cadence.com**
8. **www.cypress.com**
9. **www.exemplar.com**
10. **www.fujitsumicro.com**
11. **www.hitachi.com**
12. **www.infineon.com**
13. **www.ibm.com**
14. **www.innoveda.com**
15. **www.intel.com**
16. **www.latticesemi.com**
17. **www.lsilogic.com**
18. **www.mentor.com**
19. **www.microchip.com**
20. **www.micron.com**
21. **www.mips.com**
22. **www.mitsubishichips.com**

23. **www.model.com**
24. **www.freescale.com**
25. **www.quicklogic.com**
26. **www.rambus.com**
27. **www.ramtron.com**
28. **www.synopsis.com**
29. **www.synplicity.com**
30. **www.ti.com**
31. www.toshiba.com
32. **www.usa.samsungsemi.com**
33. **www.veribest.com**
34. **www.xilinx.com**
35. **www.zilog.com**

Примечания

1. В списке литературы, не претендующем на полноту, даются в основном отечественные книги, среди которых мало современных работ, что отражает объективное состояние дел в России. Иностранные книги, отражающие современное состояние дел в области схемотехники, хотя и имеются в природе, но в настоящее время практически недоступны для большинства российских читателей (в том числе и для авторов), так что в библиографии они представлены лишь частично.

2. Укажем некоторые журналы, публикующие полезные материалы по цифровой и аналоговой схемотехнике. К журналам практической направленности относятся: Electronic Design и Electronic Design News (USA) и отечественные журналы Chip News, Электроника-НТБ (наука, технология, бизнес), Электронные компоненты, Схемотехника, Компоненты и технологии, Компьютер пресс, BYTE-Россия. Периодические научно-теоретические издания (IEEE Transactions on Computers, IEEE Circuits&Devices, IEEE Transactions on VLSI Systems, Proceedings of the IEEE и др.) значительно сложнее для восприятия студентами и представляют интерес только для более глубокого изучения конкретных проблем.

Предметный указатель

A

ALU 130
Alternate Current (AC) 335
Aplac 680
APU 626
ASCII 450
ASICs 642

B

BD 375
Bus Driver 438

C

CAS 342
CoCentric SystemC Compiler 686
CPLD 506, 528
  блоки ввода/вывода 535
  логические расширители 530
  макроячейка 531
  микросхема CoolRunner-II 538
  программируемая матрица соединений 533
  системы коммутации 533
  структура 528
  фирмы Altera 536
  фирмы Xilinx 538
  функциональные блоки 529
CRC 262, 598
Cross talks 30
CSoC 607

D

DDR SDRAM 349
Design Flow 663
DesignLab 680
Development Board 682
DIMM 276
DMA 374, 475
DRAM 273
DSP 574
DSP-blocks 578

E

Editor of FSM 711
EEPROM 272, 301, 304
Electronics Workbench 680
End-Front Design 582
EPROM 272, 301, 304
EPROM-OTP 272, 304
Evaluation Board 682

F

FACM 290
FCRAM 359
Flash-память 272
FPGA 437, 506, 541
  LUT-блок 548, 563
  базовая архитектура 546
  блоки ввода/вывода 560
  воспроизведение 551
  встроенная память 566
  динамическая реконфигурация 543
  иерархия связей 555
  крупнозернистые логические блоки 549
  логические блоки 548
  области применения 542
  построение реконфигурируемых систем 542
  программируемые элементы 543
  процессоры 543
  распределенная память 563
  ресурсы памяти 563
  решение задач логической эмуляции 542
  сегментированные связи 554
  семейство Stratix-4 622
  семейство Virtex-5 629
  системы межсоединений 554
  с перемычками antifuse 601
  среднезернистые логические блоки 549
  усложненная архитектура 547
FPSLIC 607, 632, 633
FRAM 275, 362

G

GA 506
GMRAM 365
GPSS 674

H

Hardware 676
HDL 687

I

IP 609
IP-блоки 608

L

Latency 268
LPGA 643
LUT 548

M

MathCAD 674
MATLAB 674
MDAC 586
Micro-Cap 680
MIPS 621
MLC 316
ModelSim SE 5.7b 723
MPGA 643
MRAM 275, 365

N

NoBL 334
NtRAM 334
NV-SRAM 336

O

OLMC 525
One Hot 723

P

PAL 78, 506
P-CAD 676
PCI 454
PC-LOGS 680
PFRAM 276, 364
PIA 528
PIC 469, 471
PISO 231
PIT 375, 486
PLA 78, 506
PPI 443
PROM 272, 299
Protel 676
Prototype Plate 682
PSoC 607, 633
PSpice 681
Pull-down Resistors 24
Pull-up Resistors 24

R

RAM 272
RAS 342
RDRAM 348
Register Packing 552
RLDRAM 358
ROM 272, 297
  импульсное питание 321
  лазерные 299
  масочные 298
RS-триггер:
  режим арбитра 173
RTL 701, 702

S

SCI 462
SCL 641
SDRAM 348
SERDES 437
SGA 506, 653
SignalTap Logic Analysis 693
SIMM 276
Simula 674
SIMULINK 674
SIPO 231
SLC 78, 548
Software 676
SOP 757
SOPC 506, 541, 607
Spice 674
SPLD 506
SRAM 273, 325
SSRAM 331
Starter Kit 682
State Machins 688
StrataFlash 316, 317
Synchronous Burst SRAM 331

T

Test-Bench 691

U

UART 426, 454, 462
USART 454

V

VC 609

W

Weak Pin-keepers 25

Z

ZBT 334

А

Автомат 217
  автономный 219
  анализ лишних состояний 246
  линейный 259
  Мили 219
  Мура 219
  на триггерах с мультиплексным управлением 225
  реализация на D-триггерах 223
  реализация на JK-триггерах 224
  реализация с мультиплексным управлением 225
  с кодированием 1 из N 227
  синхронный 218
  с памятью (АП) 73, 145, 217
Адаптер:
  параллельный 374
  связной 374, 454
Адресация:
  абсолютная 402, 404
  косвенная 393, 429
  неабсолютная 402, 405
  непосредственная 393
  прямая 392, 429
  прямая регистровая 392
Адресное пространство 391
Аккумулятор 379
Арбитры 173
Арифметико-логическое устройство (АЛУ) 130, 379
Ассоциативный доступ 271
Атрибут сигнала 716

Б

Базис:
  Буля 758
  Пирса 758
  Шеффера 758
Базовые матричные кристаллы (БМК) 506, 582, 591, 642
  базовая ячейка 649
  бесканальные 644

библиотека функциональных ячеек 649
блочные 645
каналы трассировки 650
канальные 644
классификация 643
масочные 643
матричная базовая ячейка 649
параметры 650
проектирование 649
с лазерным программированием 643
функциональная ячейка 649
эквивалентный вентиль 650
Банки ввода/вывода 47
БИСМ 642
Блоки:
Data Recovery 202
DCM 201
DLL 198, 200
PLL 198
SERDES 50, 320, 437
PLL 198
множительно-суммирующие 133
ускоренного переноса 131
БМК с лазерным программированием 643
Бод 449
Буфер:
FIFO 206, 270, 284, 564
LIFO 271, 286
круговой 210, 286, 570
Буферные регистры 440
Быстрый страничный доступ 346

В

Ввод сигналов от механических ключей 170
Ввод/вывод:
обмен по прерываниям 416
отображенный на память 402
по прямому доступу к памяти 416
программный:
безусловный 415, 416
условный 415
Вентиль 79
системный 591
эквивалентный 591
Вентильные матрицы (ВМ) 506, 642, 666
структурированные 653
структурируемые 653
Вес комбинации 108
Видеопамять 271, 283
Витая пара 40
Внешние устройства 376
Внешняя память 267
Внутрикристальное согласование линий связи 47
Временные перекосы 708
Время:
доступа 270
предустановки сигнала 76, 153, 710
удержания сигнала 76, 153
Выход:
с открытым электродом 19
с открытым эмиттером 22
с тремя состояниями 17

Г

Генератор:
псевдослучайных последовательностей (ГПСП) 260

тактовых сигналов:
вторичный 187
кварцевый 184
управляемый напряжением (ГУН) 200
Граничное сканирование 493
команды 498
механизм 497
устройство управления 496
Граф переходов 709

Д

Двоичные дешифраторы 81
наращивание размерности 84
неполный 82
Двунаправленный КМОП-ключ 92
Дейзи-цепочка 468
Демультиплексоры 95
Деревья Уоллеса 134
Детекторы событий 54
Дискретизация 574
Дифференциальная передача сигнала 40
ДНФ 79
логической функции 757
Дребезг контактов 170

Ж

Ждущие синхронизаторы 172, 206
Жидкокристаллические индикаторы 59

З

Задачи:
контроля 106
синхронизации (классифиикация) 177
Закон Мура 3
Законы алгебры логики 756
ЗУ (запоминающие устройства) 267
асинхронные 273
блочные структуры 281
быстрый страничный доступ 294
важнейшие параметры 268
воспроизведение числовых функций 323
динамические 274
базовая структура 343
временные диаграммы 342
запоминающие элементы 338
мультиплексирование адреса 342
регенерация 273
типа BEDORAM 348
типа DDR SDRAM 349, 350
типа EDORAM 347
типа FPM 346
типа RDRAM 352
типа SDRAM 348
усилители-регенераторы 341
квазистатические 362
классификация 270
многобанковые 296
многопортовые 274
однопортовые 274
оперативные (ОЗУ) 272
организация 268
пакетный доступ 295
параметры быстродействия 268
перспективные 362
полоса пропускания 268
постоянные (ПЗУ) 272

производительность 268
псевдостатические 273
разрядность 268
реализация автоматов 323
реализация логических функций 322
регистровые 267
режимные параметры 268
репрограммируемые последовательные 320
с последовательным доступом 270
синхронные 273
ковейерная передача данных 331
конвейеризованные с пакетным обменом 332
с пакетным обменом 331
сквозная передача данных 331
статические 331
структура 333
статические:
с ускоренным реверсом шины 334
типа БиКМОП 337
структуры 2D 277
структуры 2DM 279, 280
структуры 3D 277
технологии DDR 295
технологии QDR 296
типа FRAM 362
типа MRAM 365
типа OUM 276, 366
типа PFRAM 364
энергонезависимость 270

И

Индикаторы:
жидкокристаллические 59
мультиплексное управление 61
светодиодные 57
семисегментные 57
Интегральные схемы (ИС):
на стандартных ячейках 666
полностью заказные 666
стандартные 665
Интерфейс 435
I^2C 438, 466
JTAG 493, 495
транспортный механизм 496
Rapid S 466
SPI 426, 438, 462
внутренний 435
параллельный 436
последовательный 437
RS-232c 438
RS-422 438
RS-422A 437
RS-485 438
с общей шиной 402
с раздельной шиной 402
системный 435
EISA 436
ISA 436
Microbus 435
PCI 436
среда передачи 437
Искажения сигналов в несогласованных линиях 33
Исключения 383

К

Каналы трассировки 650
Квантование:
по уровню 575
шум 575
Кварцевый резонатор 186

Классификация триггеров 147
Клонирование проектов 596, 597
КМОП-триггеры 165
Код 108
  ASCII 450
  Грея 222
  ДКОИ-8 450
  КОИ-7 450
  циклический 262
Кодеры и декодеры для кода Хемминга 115
Кодовая комбинация 108
Кодовое расстояние 108
Кольцевые генераторы импульсов 56
Командный цикл 386
Комбинационные цепи 73
Компараторы 102
  на равенство 103
  с тремя выходами 104
Компилятор 676
Компоненты виртуальные 609
Конвейеризация:
  обработки данных 296
  продвижения данных 348
Конвертация проектов 653
Контроллер:
  прерываний 374, 469
  прямого доступа к памяти 374, 375, 475
    включение в систему 480
    каскадирование 481
    назначение выводов и сигналов 480
    передачи 481
    структура 476
Контроль:
  по модулю 2 108
  с использованием кодов Хемминга 112
Контрольный разряд 108
Кратность ошибки 108
Критерии качества ЦУ 81
Критерий сложности схемы 80
Кэш-логика 632
Кэш-память 267, 287
  полностью ассоциативная 290
  с наборно-ассоциативной архитектурой 291
  с прямым размещением 291
  частично ассоциативная 291

Л

ЛИЗМОП 303
Линии связи:
  с гальваническими развязками 41
  типа "токовая петля" 42
Литерал переменной 99
Логический блок:
  матичного типа 79
  табличного типа 78
Логический выход 16

М

МАБИС 642
  этапы проектирования 651
Мажоритарные элементы 106
Матричные перемножители 133
Машинный цикл 386, 387
  тип 386
Микроконтроллер (МК) 372, 421
  семейства AVR 422, 423
  типа CISC 422
  типа RISC 422

Микропроцессор (МП) 4, 371
операционный блок 379
регистры общего назначения (РОН) 380
структура 379
типа CISC 400
типа RISC 400
типа VLIW 400
Микропроцессорная система (МПС) 371
архитектура:
Гарвардская 377
Принстонская 373, 376
фон Неймана 373, 376
структура:
магистрально-модульная 373
трехшинная 373
Микросхемы (МС) 3
HardCopy 654
MAX-2 600
большие (БИС) 3
заказные 3
малые (МИС) 3
ПАИС 4
ПЛИС 4
полузаказные 3
сверхбольшие (СБИС) 3
с программируемыми структурами 505
средние (СИС) 3
технология производства 593
топологические нормы 593
факторы, влияющие на стоимость 593
Минимизация логических функций 79
Многосимвольные алфавиты 9
МНОП-транзистор 301
Модемы 449
Монтажная логика 19
Мощность:
динамическая 13
полная 13
статическая 13
М-последовательность 260
Мультиплексная формула 91
Мультиплексное управление индикаторами 61
Мультиплексоры 91
многоразрядные 94
наращивание размерности 95

Н

Нагрузочные емкости в логических схемах 13
Наращивание:
матричных умножителей 134
размерности мультиплексоров 95
размерности приоритетного шифратора 89

О

Однополюсные и дифференциальные передачи сигналов 40
ОЗУ:
динамические 338
типа CDRAM 356
статические 325
временные диаграммы 329
запоминающие элементы 327
искусственная энергонезависимость 334
типа NV-SRAM 336
триггерные 325

Операция "исключающее ИЛИ" 122
Операция MAC 581
Опрос состояния кнопки 443
Оптрон 41
Основная память 267
Остаточная функция 100
Ошибка:
  пропуска 453
  формата 453
  четности 453

П

Память:
  истинно двухпортовый режим 570
  однопортовый режим 570
  простой двухпортовый режим 570
  тегов 288
Параллельное согласование волновых сопротивлений 33
Параллельный периферийный адаптер (ППА) 443, 444
  программирование 447
  режимы работы 445
Передача последовательных данных:
  асинхронная 451
  синхронная 451
    структура кадра 451
Переключательная активность 15
Перекрестные помехи 30
Перемычки antifuse 543
Пиксел 283
ПЛИС 542, 590
  конфигурирование 594
  оценки быстродействия 592
  режимы конфигурирования 594
Поллинг 468
Полоса пропускания 331
Помехи по цепям питания 26
Пороговый уровень логического сигнала 12
Порты параллельные 441
Последовательное согласование волновых сопротивлений 37
Порождающий полином 262
Последовательностные схемы 217
Преобразование:
  Фурье 577
    быстрое 578
    обратное дискретное 578
    прямое дискретное 577
Преобразователь параллельного кода в последовательный 236
Прерывания:
  аппаратные 383
  аппаратный опрос запросов 468
  вектор 384
  векторные 374, 384
  вложенность 469
  круговой приоритет 470
  маскирование 384
    запросов 470
    специальное 470
  приоритеты запросов 384
  программные 383
  программый опрос источников 468
  радиальные 384
  циклический приоритет 469
Приемопередатчики 438
Приоритетные шифраторы 87
Проблемы тактирования (классификация) 175
Программа:
  HDL Designer 689, 711
  LeonardoSpectrum 726

StateCAD Version 3.2 689, 740
Synplify 726
SystemVision 727
Программируемая логика 672
Программируемая матричная логика (ПМЛ) 506
PAL 22V10 525, 526
варианты с разделяемыми конъюнкторами 520
варианты с элементами памяти 519
макроячейки 525
обогащение функциональных возможностей 516
отечественной серии КР1556 521, 522
программирование выходных буферов 516
реализация двунаправленных буферов 518
структура 515
Программируемая пользователем память 272
Программируемые интервальные таймеры 375
Программируемые логические матрицы (ПЛМ) 506
вариант с матрицами ИЛИ-НЕ 512
варианты с элементами памяти 519
воспроизведение скобочных форм логических функций 511
обогащение функциональных возможностей 516
программирование выходных буферов 516
реализация двунаправленных буферов 518
структура 508
схема на вентильном уровне 508
схемотехника 511
упрощенное изображение 510
Программируемый выход 22
Программируемый интервальный таймер 482, 486
работа таймера 488
режим 0 489
режим 1 489
режим 2 491
режим 3 491
режим 4 492
режим 5 493
структура 487
Программируемый контроллер прерываний (ПКП) 469
включение в систему 470
каскадирование 474
программирование 472
структура 471
Программируемый контроллер прямого доступа к памяти:
структура 476
Программируемый связной адаптер (ПСА) 449, 454
передатчик 456
подключение к МП и терминалу 461
приемник 458
программирование 460
сигналы квитирования 455
Программный счетчик 381
Проектирование 659
аналоговые и аналого-цифровые фрагменты 681
блочное 609
восходящее 660
выбор САПР 684
групповое использование САПР 684

инструментальные средства 673
интерпретирующие
редакторы 691
исходные модули 660
компилирующие редакторы 691
компиляция проекта 690
конструкторско-топологическое
представление 659
кросс-ассемблирование 678
маршрут 663
микропроцессорная система 671
на БМК 642
на дискретных ИС 678
на основе МИС и СИС 671
на стандартных ячейках 641
нисходящее 660
определение временных
характеристик 691
основные идеи 674
основные составляющие 663
отладка на прототипе 678
платформа 673
платформенное 609
полностью заказное 641
программная модель 691
проекты на ПЛИС и СнПК 682
прототипные платы 692
процесс 661
разработка процессорного
блока 676
с использованием языков
описания 737
симуляция 678
системный этап 674
спецификация проекта 685
способ описания проекта:
графический 686
поведенческий 687
структурный 687
текстовый 686, 687
среда и система разработки 676
средства описания
автоматов 688
граф-схемы 688
средства отладки 681
стратегия 661
структурное описание 659
тестирование проекта 690
технического задания (ТЗ) 684
тип обрабатываемой
информации 664
функциональное
моделирование 690
функциональное описание 659
ЦУ на базе ПЛИС 683
экспериментальная проверка 692
эмуляция 678
этап системного
проектирования 664
этапы 660
этапы проектирования с
использованием САПР 690
языки описания аппаратуры 687
высокого уровня 688
низкого уровня 687
Процесоры:
типа CISC 400, 422
типа RISC 400, 422
типа VLIW 400
Прямой доступ к памяти 475
Псевдослучайные
последовательности 259

Р

Разностные преобразователи 54
Распределитель тактов 252
Регистр:
команд 381
флажков 379

Регистровые файлы 232
Регистры 230
буферные 440
параллельные 231
последовательно-параллельные 231
последовательные (сдвигающие) 231
сдвигающие 233
универсальные 234
Режимы неиспользуемых входов 62
Резистор:
"заземляющий" 24
подтягивающий 24
Реконструкция проектов 596, 597
Риски 73
динамические 74
сбоя 73
статические 74

С

САПР 551, 596, 660, 673
этапы работы 675
Светодиодные индикаторы 57
Свойство самозапуска 246
Сдвигатели:
с управлением двоичным кодом 141
с управлением кодом 1 из N 139
СДНФ 77
Семейство:
Cyclone-4E. 599
MAX-2 600
ProASIC3 631
PSoC 633
Spartan-6 600
Virtex-6 626
Семисегментные индикаторы 57
Сигнал:
аналоговый 574
время предустановки 269
время сохранения 269
время удержания 269
дискретный 574
длительность 269
мезохронный 206
плезиохронный 207
стробирующий 329
цифровой 574, 575
Сигнатурный анализатор 263
Синтез 661
Синхронизаторы:
мезохронных сигналов 207
плезиохронных сигналов 212
Синхронизация 175
Синхросигнал 176
Синхросимволы 453
Система на программируемом кристалле (СнПК) 607
блочного типа 610, 612
микроконтроллерного типа 632
семейство FPSLIC 632
семейство PSoC 633
однородного типа 610, 612
Системы тактирования:
замкнутые 180
разомкнутые 179
Согласование элементов КМОП-ТТЛШ по уровням сигналов 63
Соглашения положительной и отрицательной логики 8
Сопроцессор 372

Способы оценки задержек сигналов 10
Средняя задержка логического элемента 12
Стандарт:
LVDS 45
RSL 353
Стандартная логическая ячейка 592
Стандарты сигналов ввода/вывода 42
Старт-бит 451
Статическая помехоустойчивость логического элемента 11
Статические риски 74
Стек 380
Стоп-бит 451
Структурированные вентильные матрицы (СтрВМ) 653
Структурное уравнение триггера 156
Сумматор 116
групповой с цепным переносом 126
групповой структуры 126
накапливающий 129
одноразрядный 116
параллельный:
с параллельным переносом 123
с последовательным переносом 120
последовательный 119
с ключами передачи переноса 121
с параллельными межгрупповыми переносами 127
с условным переносом 128
СФ-блоки 609
Схемы:
запоминающие состояние контакта 25
контроля 105
полузаказные 642
размножения тактовых сигналов:
H-дерево 190
пирамидальные 189
свертки 109
ускоренного умножения 136
Счетчики 237
быстродействие 238
в коде 1 из N 251
на кольцевых регистрах 252
в коде Грея 249
вычитающие 237
групповой структуры 242
двоичные 238
асинхронные 238
синхронные 240
Джонсона 255
модуль счета 237
полиномиальные 258
реверсивные 237
с произвольным модулем 244
метод модификации связей 244
метод управляемого сброса 247
синхронные:
с параллельным переносом 240
с последовательным ереносом 242
способы использования 238
суммирующие 237

T

Таймер 482
микросхема ВИ54 486
сторожевой 485
формирование ШИМ-сигналов 484

Таймеры-счетчики 482
Тактирование 175
варианты Source-Synchronous и Clock-Data Recovery 176
влияние направления передачи тактовых сигналов 194
глобальное 179
сквозное 178
двухфазное 194
параметры сигналов 196
конвейерное 179
многофазное 197
однофазное 190
учет фазовых сдвигов 192
структура 183
Тактовые импульсы:
параметры 181
Такты машинного цикла 386, 387
Теорема Тевенина 36
Терминал 455
Терминатор 33
Технология:
DCI 47
DDR 48
БиКМОП 337
программируемой мощности 623
Токовые параметры логического элемента 12
Транзистор:
с плавающим затвором 303
с плавающим и управляющим затворами 303
Триггеры 145
T и JK
логические структуры 159
время предустановки 153
время удержания 153
двухступенчатые 167
защелки 152
классификация 147, 148, 149
круговые гонки 152
метастабильные состояния 154
на элементах КМОП 165
по схеме "трех триггеров" 163
с внутренней задержкой 164
синхронные 149
словари 156
счетные 148
типа D 148
режим прозрачности 152
типа JK 148
типа RS 148
запрещенная комбинация входных сигналов 147
типа T 148
управление уровнем 149
управление фронтом 149
Троичное моделирование 8

У

Указатели старшей единицы 90
Указатель стека 380
Умножители 572
комплексных чисел 574
матричные 132
Универсальные логические модули (УЛМ):
на основе мультиплексоров 98
Универсальный асинхронный приемопередатчик (УАПП) 454
Универсальный синхронно-асинхронный приемопередатчик (УСАПП) 454
Уровни логических сигналов 10
Устройства ввода-вывода 376

Ф

Факторизация 80
Фильтр:
  БИХ-фильтр 576
  КИХ-фильтр 575
  нерекурсивный 575
  рекурсивный 576
Фильтрация:
  напряжений питания 28
  цифровая 575
Флэш-память 307
  команды управления 311
  с зеркальным битом 317
  с многоуровневым хранением заряда 316
  с несимметричными блоками 312
  с симметричными блоками 314
  файловая (ФФП) 314
  ячейки типа NAND 309
  ячейки типа NOR 308
Формирователи импульсов по длительности 54
Функция:
  генерации 123
  прозрачности 123
  спектральная 577

Ц

Цепи с переключаемыми конденсаторами 583
Цепочки каскадирования 551
ЦОС 574
  блоки 578

Ч

Частота системная 593

Ш

Шина:
  CAN 438
  EISA 436
  ISA 436
  PCI 436
  PCI Express 436
  PCI-X 2,0 436
  USB 438
  VMEbus 436
  мультиплексируемая 376
Шинные:
  драйверы 438
  формирователи 373, 375, 438
Шифраторы:
  приоритетные 87
    наращивание размерности 89

Э

Эквивалентный вентиль 650
Элемент:
  задержки 51
  запоминающий 268
  памяти (ЭП) 217

Я

Ядра:
  firm 608
  hard 608, 612
  soft 608
  процессорные 614
    Microblaze 616
    Picoblaze 616

семейства Nios 614, 616
типа hard 618
Язык:
ABEL 688
AHDL 688
EDIF 687
GPSS 674
HDL 651
PLDASM 688
Simula 674
SystemC 686
Verilog 688
VHDL 688, 694
для моделирования и синтеза 701
VHDL-AMS 727
высокого уровня 688
низкого уровня 687
Ячейка:
запоминающая 268
периферийного сканирования 493