

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное агентство по образованию  
Московский физико-технический институт  
(государственный университет)

*A.Л. Ларин*

**ОСНОВЫ  
ЦИФРОВОЙ ЭЛЕКТРОНИКИ**

**С изменениями 05.2012**

*Рекомендовано  
Учебно-методическим объединением  
высших учебных заведений Российской Федерации  
по образованию в области прикладных математики и физики  
в качестве учебного пособия*

МОСКВА 2008

УДК 621.37(075)  
ББК 32.814я73  
Л25

Рецензенты:

Кафедра управления и информатики  
Московского энергетического института (технического университета)  
(зав. кафедрой доктор технических наук, профессор *В.М. Беседин*)

Кандидат технических наук, доцент *Н.Д. Дмитриева*

**Ларин А.Л.**

Л25      Основы цифровой электроники: Учебное пособие. – М.:  
МФТИ, 2008. – 315 с.  
ISBN 978-5-7417-0228-4

Отражено содержание лекционного курса, который автор читает студентам телекоммуникационного направления факультета радиотехники и кибернетики Московского физико-технического института (МФТИ) в шестом семестре. Рассмотрен широкий круг вопросов, относящихся к цифровой электронике, программируемым логическим матрицам и микроконтроллерам.

Предназначено для студентов вузов и аспирантов начинающих изучать вопросы, связанные с цифровыми методами в электронике, а также для студентов других факультетов, проходящих лабораторный практикум на кафедре радиотехники.

УДК 621.37(075)  
ББК 32.814я73

**ISBN 978-5-7417-0228-4**

© Ларин А.Л., 2008

© Московский физико-технический институт  
(государственный университет), 2008

# ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	7
ГЛАВА 1. ИМПУЛЬСНЫЕ СХЕМЫ НА ОСНОВЕ ОПЕРАЦИОННЫХ УСИЛИТЕЛЕЙ .....	10
1.1. Элементы импульсной техники .....	10
1.1.1. Компаратор и триггер Шмитта .....	10
1.1.2. Мультивибратор .....	13
1.1.3. Ждущий мультивибратор .....	15
1.1.4. Генератор напряжения треугольной формы .....	18
ГЛАВА 2. ЛОГИЧЕСКИЕ СХЕМЫ .....	20
2.1. Логические сигналы и вентили .....	20
2.2. Семейства логических схем .....	25
2.3. Логические КМОП-схемы .....	27
2.3.1. Логические уровни КМОП-схем .....	27
2.3.2. МОП-транзисторы, используемые в логических схемах .....	28
2.3.3. Схема КМОП-инвертора .....	30
2.3.4. КМОП-схемы И-НЕ И ИЛИ-НЕ .....	32
2.4. Связь помехоустойчивости логических схем с логическими уровнями .....	35
2.5. Неиспользуемые входы логических схем .....	38
2.6. Схемы с тремя состояниями .....	39
2.7. Динамические свойства КМОП-схем .....	42
2.7.1. Переходные процессы в логических схемах .....	42
2.7.2. Задержка распространения сигнала .....	49
ГЛАВА 3. КОМБИНАЦИОННЫЕ СХЕМЫ .....	52
3.1. Дешифраторы и шифраторы .....	52
3.1.1. Полные дешифраторы .....	53
3.1.2. Неполные дешифраторы .....	55
3.1.3. Дешифратор для семисегментных индикаторов .....	55
3.1.4. Шифраторы .....	56
3.2.1. Приоритетные шифраторы .....	57
3.2. Мультиплексоры .....	60
3.2.1. Мультиплексоры в интегральном исполнении .....	61
3.3. Логические элементы ИСКЛЮЧАЮЩЕЕ ИЛИ .....	63
3.3.1. Вентили ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ .....	63
3.3.2. Схемы проверки на четность .....	65
3.4. Временные соотношения в схеме .....	66
3.4.1. Временные диаграммы .....	67
3.4.2. Задержка распространения .....	70
3.4.3. Временные параметры .....	70
3.4.4. Временной анализ .....	71
3.4.5. Гонки в комбинационных устройствах .....	72
4.1. Компараторы .....	76

4.2. Сумматоры.....	78
4.2.1. Полусумматоры и полные сумматоры .....	78
4.2.2. Сумматоры со сквозным (последовательным) переносом .....	79
4.2.3. Сумматоры с ускоренным (параллельным) переносом .....	80
<b>ГЛАВА 5. ПОСЛЕДОВАТЕЛЬНОСТНЫЕ ЛОГИЧЕСКИЕ СХЕМЫ.....</b>	<b>85</b>
5.1. Элемент с двумя устойчивыми состояниями.....	87
5.1.1. Цифровой подход.....	87
5.1.2. Аналоговый подход .....	88
5.2. Неустойчивое равновесие.....	89
5.3. Защелки и триггеры .....	90
5.3.1. SR-защелка .....	90
5.3.2. <b>SR</b> -защелка .....	93
5.4. SR-защелка с входом разрешения.....	94
5.5. D-защелка .....	96
5.6. D-триггер, переключающийся по фронту .....	98
5.7. D-триггер с входом разрешения .....	103
5.8. Двухтактный SR-триггер .....	104
5.9. Двухтактный JK-триггер .....	106
5.10. JK-триггер, переключающийся по фронту .....	108
5.11. T-триггер.....	110
<b>ГЛАВА 6. СЧЕТЧИКИ, РЕГИСТРЫ СДВИГА.....</b>	<b>112</b>
6.1. Счетчики .....	112
6.1.1. Счетчики с последовательным переносом.....	112
6.1.2. Синхронные счетчики .....	113
6.1.3. Счетчики в ИС средней степени интеграции .....	115
и их применение .....	115
6.1.4. Декодирование состояний двоичного счетчика .....	124
6.2. Регистры сдвига .....	127
6.2.1. Структура регистра сдвига.....	127
6.2.2. Регистры сдвига в ИС средней степени интеграции .....	130
6.2.3. Последовательно-параллельное преобразование .....	132
6.2.4. Счетчики на регистрах сдвига .....	133
6.2.5. Кольцевые счетчики .....	133
6.2.6. Счетчики Джонсона.....	137
6.2.7. Генераторы псевдослучайных последовательностей .....	138
<b>ГЛАВА 7. ДИСКРЕТИЗАЦИЯ АНАЛОГОВЫХ СИГНАЛОВ .....</b>	<b>144</b>
7.1. Аналоговые, дискретные и цифровые сигналы .....	144
7.2. Аналогово-цифровое и цифроаналоговое преобразование .....	145
7.3. Частота Найквиста .....	146
7.4. Спектр дискретного сигнала .....	148
7.5. Теорема Котельникова.....	154
<b>ГЛАВА 8. МЕТОДЫ ДИСКРЕТИЗАЦИИ СИГНАЛОВ .....</b>	<b>158</b>
8.1. Цифроаналоговые преобразователи (ЦАП) .....	158
8.2. Аналогово-цифровые преобразователи (АЦП) .....	162

8.2.1. Аналогово-цифровые преобразователи, построенные по методу последовательного счета .....	163
8.2.2. АЦП поразрядного уравновешивания.....	168
8.2.3. АЦП параллельного преобразования .....	171
8.3. Устройства выборки и хранения (УВХ).....	173
ГЛАВА 9. ПАМЯТЬ.....	176
9.1. Постоянные запоминающие устройства .....	177
9.1.1. Применение ПЗУ для реализации комбинационных логических функций .....	178
9.1.2. Внутренняя структура ПЗУ .....	179
9.1.3. Двумерная адресация.....	181
9.1.4. Типы постоянных запоминающих устройств.....	183
9.1.5. Входы управления и временные параметры ПЗУ .....	187
9.2. Оперативные запоминающие устройства .....	190
9.3. Статические оперативные запоминающие устройства.....	191
9.3.1. Входы и выходы статического ОЗУ .....	191
9.3.2. Структура статического ОЗУ .....	192
9.3.3. Временные параметры статического ОЗУ .....	195
9.4. Динамические оперативные запоминающие устройства.....	199
9.4.1. Структура динамического ОЗУ .....	199
9.4.2. Временные параметры динамического ОЗУ .....	203
9.4.3. Синхронные динамические ОЗУ .....	207
ГЛАВА 10. ПРОГРАММИРУЕМЫЕ МИКРОСХЕМЫ ТИПА CPLD И FPGA .....	209
10.1. Интегральные схемы типа CPLD .....	209
10.1.1. Семейство ИС XC9500 фирмы Xilinx .....	210
10.1.2. Архитектура блока ввода/вывода .....	213
10.2. Интегральные схемы типа FPGA .....	215
10.2.1. ПЛИС семейств SPARTAN-II и SPARTAN-III.....	216
10.2.2. Архитектура SPARTAN-II .....	219
10.3. Описание структуры ИС SPARTAN-II.....	220
10.3.1. Матрица SPARTAN-II .....	220
10.3.2. Блок ввода-вывода .....	222
10.3.3. Ввод сигнала.....	224
10.3.4. Вывод сигнала .....	225
10.3.5. Банки ввода-вывода .....	226
10.4. Конфигурируемый логический блок CLB .....	228
10.4.1. Таблица преобразования LUT.....	229
10.4.2. Запоминающие элементы.....	229
10.4.3. Дополнительная логика .....	230
10.4.4. Арифметическая логика .....	230
10.4.5. Буферы с тремя состояниями.....	231
10.4.6. Блочная память (Block RAM) .....	231
10.5. Программируемая трассировочная матрица .....	233
10.5.1. Локальные связи .....	234

10.5.2. Трассировочные ресурсы общего назначения .....	235
10.5.3. Трассировочные ресурсы для блоков ввода-вывода .....	235
10.5.4. Специальные трассировочные ресурсы .....	236
10.5.5. Глобальные трассировочные ресурсы .....	237
10.6. Распределение сигналов синхронизации .....	237
10.7. Модули автоподстройки задержки (DLL) .....	238
10.8. Система проектирования .....	239
10.8.1. Размещение проекта в кристалл .....	241
10.8.2. Верификация проекта .....	242
10.8.3. Конфигурирование кристалла в составе устройства .....	243
10.8.4. Режимы конфигурирования .....	243
10.8.5. Сигналы конфигурации .....	244
10.8.6. Последовательность конфигурации .....	245
<b>ГЛАВА 11. МИКРОКОНТРОЛЛЕР .....</b>	<b>246</b>
11.1. Внутренняя структура современных микроконтроллеров .....	247
11.2. Характеристики микроконтроллеров .....	248
11.3. Микроконтроллер ATmega8535 .....	250
11.3.1. Назначение выводов микроконтроллера ATmega8535 .....	251
11.3.2. Структура микроконтроллера ATmega8535 .....	253
11.4. Дополнительные блоки микроконтроллера .....	267
11.4.1. Порт А .....	269
11.4.2. Порты В, С, D .....	271
11.4.3. Система прерываний .....	271
11.4.4. Таймер/счетчик 0 .....	276
11.4.5. Таймер/счетчик 1 .....	279
11.4.6. Таймер/счетчик 2 .....	282
11.4.7. Аналого-цифровой преобразователь .....	286
11.4.8. Последовательный синхронный интерфейс .....	290
11.4.9. Универсальный синхронно-асинхронный приемопередатчик .....	296
11.5. Система команд микроконтроллера ATmega8535 .....	307
<b>СПИСОК ЛИТЕРАТУРЫ .....</b>	<b>312</b>

## ВВЕДЕНИЕ

В отличие от *аналоговых* устройств, в которых происходит преобразование меняющихся во времени сигналов, способных принимать любые значения из непрерывного интервала величин, в *цифровых* схемах во многих случаях мы предполагаем, что это не так. Цифровой сигнал – это модель, согласно которой в любой момент времени сигнал принимает одно из двух дискретных значений, которые чаще всего называют *нулем* (0) и *единицей* (1) (или *низким* и *высоким* уровнями).

Появление цифровой техники относится к 1940-м годам двадцатого века и связано с развитием радиолокации и вычислительной техники. Однако широко применяется на практике цифровая электроника начала в 60-е годы XX века благодаря бурному развитию полупроводниковой электроники и, в частности, интегральных микросхем. В последние десятилетия цифровая техника существенно потеснила аналоговую электронику даже в таких, ранее сугубо аналоговых областях, как фильтрация сигналов, радиоприем и других. В литературе [5] приводятся следующие примеры систем, которые раньше были исключительно аналоговыми и теперь переходят в разряд цифровых:

*Фотография.* До начала XXI века в большинстве фотоаппаратов для регистрации изображения использовались галоидные соединения серебра. Однако увеличение объема цифровой памяти в одном кристалле привело к появлению цифровых фотокамер, в которых изображение фиксируется в виде массива точек (пикселов), например, в количестве  $1280 \times 1024$  и больше, где в каждом пикселе запоминается интенсивность красной, зеленой и синей цветовых составляющих, причем на каждую из них отводится по 8 битов. Этот большой массив данных можно преобразовать и сжать; например, в формате, называемом *JPEG*, размер запоминаемого массива данных может составлять только 5% от исходного объема в зависимости от количества деталей в изобра-

жении. Таким образом, принцип действия цифровых фотокамер основан на применении цифровой памяти и на цифровой обработке данных.

*Видеозапись.* На универсальном цифровом диске (*digital versatile disc, DVD*) видеоизображение запоминается в цифровом формате с большой степенью сжатия, например, *MPEG-2*. Согласно этому стандарту осуществляется кодирование малой части кадров видеоизображения в формате, подобном *JPEG*, а информация об остальных кадрах представляется в виде данных о различии между текущим кадром и предыдущим. Емкость одностороннего DVD-диска с записью в одном слое составляет 35 миллиардов битов, и этого достаточно для записи примерно двухчасового фильма с хорошим качеством, а емкость двухслойного двустороннего диска в четыре раза больше.

*Запись звука.* Если раньше все сводилось к запоминанию аналоговых колебаний в виде отпечатка на виниловой пластинке или на магнитной ленте, то в настоящее время для записи звука применяют цифровые компакт-диски (*CD*). Музыка запоминается на компакт-диске в виде последовательности 16-разрядных двоичных чисел, соответствующих выборкам, которые берутся из исходного аналогового колебания с интервалом 22.7 микросекунды в каждом из стереоканалов. Запись на целиком заполненном компакт-диске (73 минуты) содержит свыше 6 миллиардов битов информации.

Причин, по которым происходит революционный переход от аналоговой схемотехники к цифровой довольно много. Основными из них можно считать следующие:

- *Воспроизводимость результатов.* При одном и том же наборе входных сигналов правильно спроектированная и надежно изготовленная цифровая схема дает точно одни и те же результаты, в отличие от выходных сигналов аналоговой схемы, которые в той или иной степени зависят от температуры, напряжения питания и многих других факторов.

- *Простота проектирования.* Проектирование цифровых устройств представляет собой логическую задачу. Для ее решения не требуется сложных математических вычислений, и пове-

дение простой логической схемы можно представить себе без учета того, как действуют конденсаторы, транзисторы и другие элементы, для моделирования которых понадобились бы громоздкие вычисления.

• *Программируемость*. Основная работа по проектированию цифровых устройств выполняется сегодня путем написания программ на языках описания схем (*hardware description languages, HDLs*). Эти языки позволяют задать как структуру цифровой схемы, так и выполняемую ею функцию, или смоделировать их. Как правило, компилятор языка описания схем дополняется программами моделирования и синтеза. Эти программы используются для изучения поведения модели устройства до того, как оно будет реализовано, а затем и для синтеза, то есть для преобразования модели в схему согласно технологии выбранных компонентов.

*Быстродействие*. Современные цифровые устройства работают очень быстро. Транзисторы в самых быстрых интегральных микросхемах переключаются за единицы пикосекунд, а в заключенном сложном устройстве, построенном на таких транзисторах, формирование выходных сигналов после изменения сигналов на входах происходит менее чем за 1 наносекунду. Это означает, что такое устройство способно производить более 1 миллиарда операций в секунду.

# ГЛАВА 1. ИМПУЛЬСНЫЕ СХЕМЫ НА ОСНОВЕ ОПЕРАЦИОННЫХ УСИЛИТЕЛЕЙ

## 1.1. Элементы импульсной техники

Рассмотрим схемы, в которых присутствуют как аналоговые сигналы, так и цифровые сигналы, определенные выше. Как правило, в таких устройствах используются усилители с сильной положительной обратной связью.

### 1.1.1. Компаратор и триггер Шмитта

Часто требуется установить, какой из двух аналоговых сигналов больше, или определить, когда сигнал достигнет заданного значения. Простейшим устройством сравнения (компаратором) является операционный усилитель (ОУ) с большим коэффициентом усиления, используемый без отрицательной обратной связи (рис. 1.1).

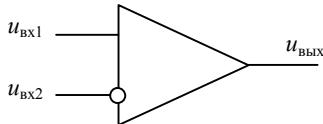


Рис. 1.1. Компаратор на операционном усилителе

При работе операционного усилителя в качестве компаратора следует иметь в виду следующее: в линейном режиме усилитель находится в течение малого времени, большую часть времени разность напряжений между входами велика; входное сопротивление при этом не остается высоким, характерным для операционного усилителя, работающего в линейном режиме. В результате при переключении ОУ из одного состояния в другое наблюдается изменение входного сопротивления и изменение входного тока, что может отразиться на работе компаратора при большом выходном сопротивлении источников сигналов.

На рис. 1.2 показан выходной сигнал при наличии синусоидального сигнала на неинвертирующем входе и линейно изменяющегося сигнала на инвертирующем входе.

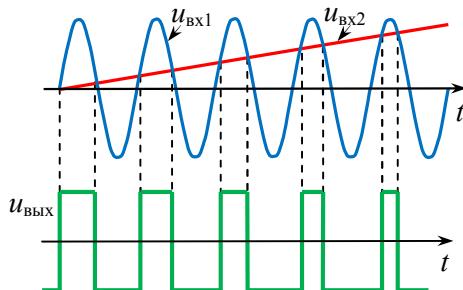


Рис. 1.2. Сигналы на входах и выходе компаратора

Простейшая схема компаратора имеет два недостатка. При медленно изменяющейся разности входных напряжений напряжение на выходе также может изменяться достаточно медленно. Более того, если во входном сигнале присутствует шум, то в выходном сигнале появляется *дребезг* в те моменты времени, когда разность входных напряжений проходит через нуль. Рис. 1.3 иллюстрирует данную ситуацию.

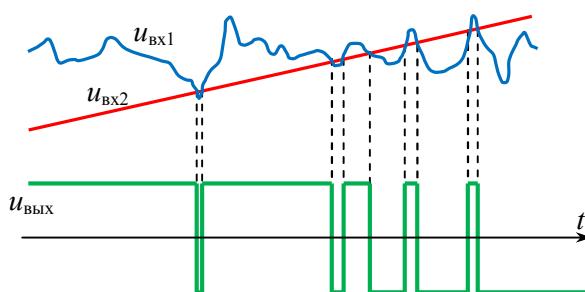


Рис. 1.3. Дребезг сигнала на выходе компаратора без обратной связи

Положительная обратная связь позволяет устраниить оба недостатка. Если в компараторе, построенном на ОУ, используется положительная обратная связь, то он обладает гистерезисными свойствами. Компаратор с положительной обратной связью (рис. 1.4) называется *триггером Шмитта*.

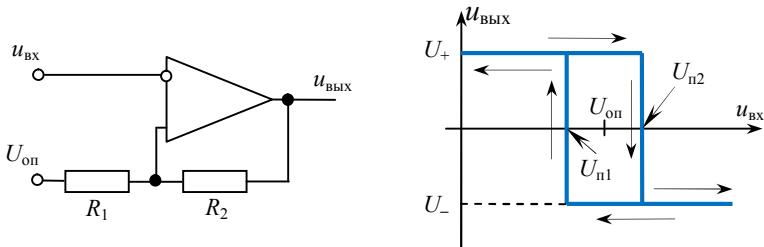


Рис. 1.4. Триггер Шмитта и его передаточная характеристика

Если, повышаясь, входное напряжение  $u_{\text{вх}}$  достигает уровня  $U_{\text{n}2}$ , то выходное напряжение ОУ, используемого в триггере Шмитта, достаточно быстро изменяется от максимального положительного напряжения  $U_+$  к максимальному (по модулю) отрицательному напряжению  $U_-$ . Время изменения выходного напря-

жения определяется выражением  $t_{\text{неп}} = \frac{U_+ + |U_-|}{V}$ , где  $V$  – макси-

мальная скорость изменения выходного напряжения ОУ. При понижении  $u_{\text{вх}}$  до величины  $U_{\text{n}1}$  происходит быстрое изменение выходного напряжения от максимального (по модулю) отрицательного напряжения  $U_-$  к максимальному положительному напряжению  $U_+$ . Пороговые напряжения определяются соотношениями

$$U_{\text{n}1} = \frac{R_1}{R_1 + R_2} \cdot (U_- - U_{\text{он}}) + U_{\text{он}} = \beta(U_- - U_{\text{он}}) + U_{\text{он}},$$

$$U_{\text{n}2} = \frac{R_1}{R_1 + R_2} \cdot (U_+ - U_{\text{он}}) + U_{\text{он}} = \beta(U_+ - U_{\text{он}}) + U_{\text{он}}.$$

На рис. 1.5 показана реакция триггера Шмитта на входной сигнал при нулевом напряжении  $U_{\text{он}}$ .

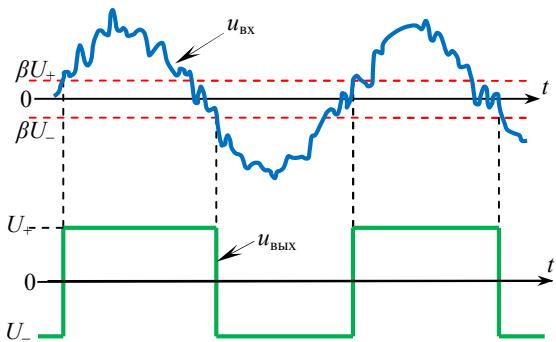


Рис. 1.5. Реакция триггера Шмитта на входной сигнал

### 1.1.2. Мультивибратор

Используя операционный усилитель можно создать генератор сигналов, имеющий прямоугольную форму выходного сигнала. На рис. 1.6 показан такой генератор, называемый *мультивибратором*.

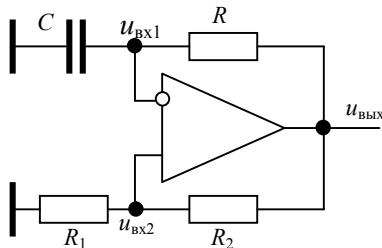


Рис. 1.6. Схема мультивибратора на ОУ

На рис. 1.7 приведены временные диаграммы, поясняющие работу мультивибратора.

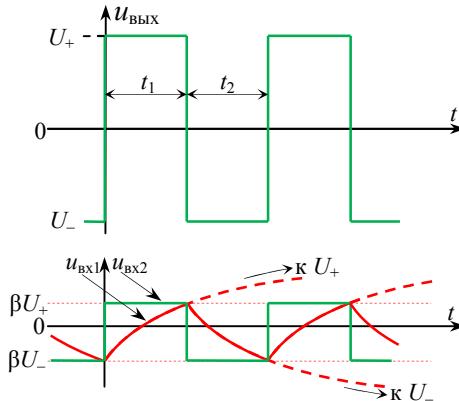


Рис. 1.7. Временные диаграммы, иллюстрирующие работу мультивибратора

Выходное напряжение мультивибратора попеременно изменяется между значениями  $U_+$  и  $U_-$ . Сигнал на неинвертирующем входе ОУ совпадает по форме с выходным сигналом, уменьшенным в  $\beta^{-1} = (R_1 + R_2)/R_1$  раз. Напряжение на выходе усилителя изменяется лишь в те моменты, когда напряжение на конденсаторе достигает величин  $\beta U_+$  или  $\beta U_-$ . Конденсатор  $C$  перезаряжается током, поступающим с выхода ОУ, через резистор  $R$ , т. е. постоянная времени перезаряда конденсатора равна  $RC$ . Полное напряжение, до которого мог бы зарядиться конденсатор, равно  $U_+$  или  $U_-$ , но при напряжении равном  $\beta U_+$  или  $\beta U_-$  имеет место смена полярности выходного сигнала ОУ. Интервалы времени  $t_1$  и  $t_2$ , в течение которых конденсатор перезаряжается, определяются исходя из того, что напряжение на конденсаторе изменяется согласно следующим выражениям:

$$u_C = U_+ - (U_+ - \beta U_-) \cdot e^{-t/RC}$$

при изменении напряжения от  $\beta U_-$  до  $\beta U_+$ , и

$$u_C = |U_-| - (|U_-| - \beta U_+) \cdot e^{-(t-t_1)/RC}$$

при изменении напряжения от  $\beta U_+$  до  $\beta U_-$ , где  $\beta = R_1/(R_1 + R_2)$ .

Если выходной сигнал симметричен относительно нуля, то есть  $U_+ = |U_-|$ , то

$$t_1 = t_2 = RC \ln \frac{1+\beta}{1-\beta},$$

а период повторения равен

$$T = t_1 + t_2 = 2RC \ln \frac{1+\beta}{1-\beta} = 2RC \ln \frac{2R_1 + R_2}{R_2}$$

и не зависит от свойств операционного усилителя. При  $\beta \ll 1$

$$T \approx 4\beta RC = \frac{4R_1}{R_1 + R_2} RC.$$

### 1.1.3. Ждущий мультивибратор

*Ждущий мультивибратор* вырабатывает на выходе импульс напряжения определенной длительности  $T$  после появления на его входе импульса запуска независимо от длительности этого импульса. Ждущий мультивибратор часто называют одновибратором. На рис. 1.8 показан один из вариантов схемы ждущего мультивибратора, а на рис. 1.9 временные диаграммы, иллюстрирующие его работу.

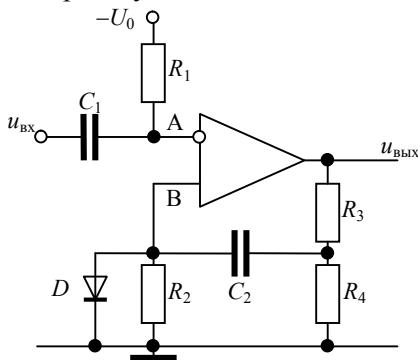


Рис. 1.8. Ждущий мультивибратор

В исходном состоянии (до подачи запускающего импульса  $u_{\text{вх}}$ ) операционный усилитель находится в состоянии насыщения с

уровнем выходного напряжения  $U_+$ , поскольку напряжение на неинвертирующем входе равно нулю, а на инвертирующем входе равно  $-U_0$ . Начальное напряжение на конденсаторе  $C_2$  при этом равно  $\beta U_+$ , где  $\beta = R_4 / (R_3 + R_4)$ .

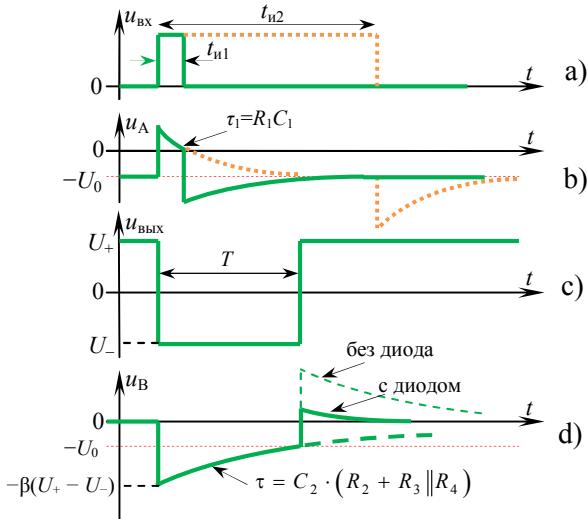


Рис. 1.9. Временные диаграммы, иллюстрирующие работу ждущего мультивибратора

При появлении на инвертирующем входе короткого запускающего импульса длительностью  $t_{и1}$  положительной полярности с амплитудой, превышающей опорное напряжение  $U_0$ , напряжение на выходе операционного усилителя изменяется от  $U_+$  до  $U_-$ . Напряжение на неинвертирующем входе скачком принимает значение  $-\beta(U_+ - U_-)$  и в дальнейшем по мере перезаряда конденсатора  $C_2$  изменяется по экспоненциальному закону с постоянной времени  $\tau = C_2 \cdot (R_2 + R_3 \| R_4)$ , стремясь к нулевому значению. Длительность генерируемого импульса  $T$  равна времени изменения напряжения на неинвертирующем входе от значения  $-\beta(U_+ - U_-)$  до значения  $-U_0$ :

$$T = \tau \ln(\beta(U_+ - U_-)/U_0).$$

Если  $U_+ = |U_-|$ , то

$$T = \tau \ln(2\beta U_+/U_0).$$

Диод  $D$  сокращает время восстановления начального напряжения на конденсаторе  $C_2$ . Это позволяет осуществлять новый запуск ждущего мультивибратора через меньшее время после окончания формирования очередного импульса.

Постоянная времени дифференцирующей цепи на входе  $\tau_1 = R_1 C_1$  должна быть малой по сравнению с длительностью выходного импульса  $T$ . В этом случае длительность запускающего импульса  $t_{ii}$  может быть больше длительности формируемого импульса (случай, когда  $t_{ii2}$  больше  $T$ , изображен на рис. 1.9а и 1.9б пунктиром). На рис. 1.9д тонкой пунктирной линией показана форма напряжения при отсутствии диода  $D$ .

Зависимость длительности выходного импульса от длительности  $t_{ii}$  наблюдается в том случае, когда  $t_{ii2}$  меньше  $T$ , и в момент окончания запускающего импульса отрицательное напряжение  $U_1$  в точке А по модулю превосходит напряжение  $U_2$  в точке В. Данный случай показан пунктирными линиями на рис. 1.10.

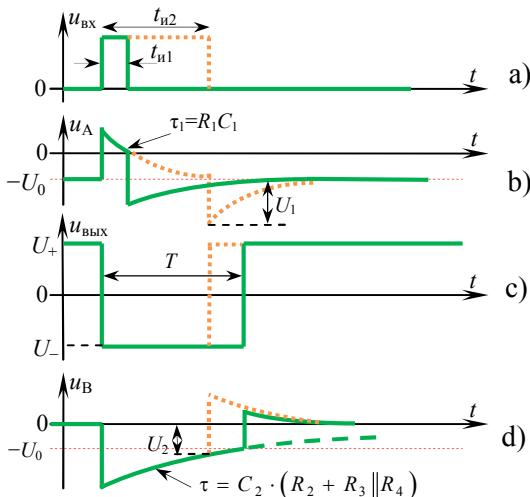


Рис. 1.10. Временные диаграммы, иллюстрирующие зависимость длительности выходного сигнала от длительности входного сигнала

Для устранения этого недостатка следует изменить цепь запуска так, как показано на рис. 1.11. При этом отрицательный перепад напряжения, появляющийся в момент окончания запускающего импульса в точке C, не попадает на инвертирующий вход операционного усилителя.

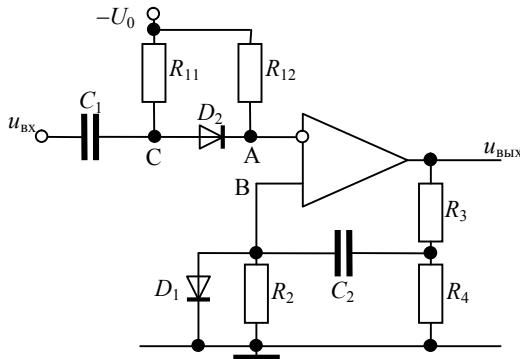


Рис. 1.11. Ждущий мультивибратор с измененной цепью запуска

#### 1.1.4. Генератор напряжения треугольной формы

В схеме на рис. 1.12 с использованием усилителя ОУ1, реистора  $R$  и конденсатора  $C$  реализован интегратор, а усилитель ОУ2 с положительной обратной связью выполняет функцию компаратора (триггер Шmittта). На неинвертирующем входе ОУ2 в пропорции, определяемой отношением  $R_2/R_1$ , смешиваются линейно изменяющееся напряжение с выхода интегратора и прямоугольные колебания с выхода компаратора, что приводит к периодическим изменениям напряжения на выходе компаратора.

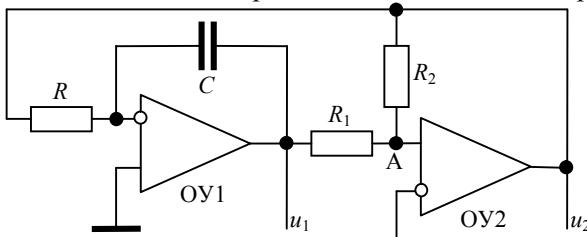


Рис. 1.12. Генератор напряжения треугольной формы

В течение одного полупериода компаратор находится в состоянии с высоким уровнем выходного напряжения  $U_+$ , а затем после переключения в течение второго полупериода – в состоянии с уровнем выходного напряжения  $U_-$ . На рис.1. 13 приведены временные диаграммы, иллюстрирующие работу генератора.

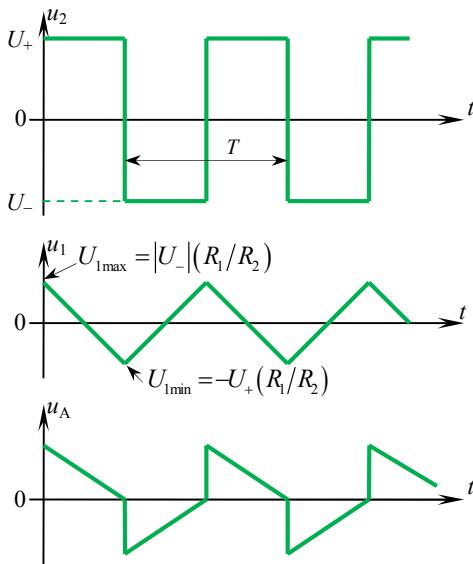


Рис. 1.12. Временные диаграммы, иллюстрирующие работу генератора напряжения треугольной формы

Период колебаний оказывается равным  

$$T = 4RC \cdot (R_1/R_2).$$

Колебания на выходе интегратора имеют треугольную форму. Амплитуда этих колебаний, при  $U_+ = |U_-|$ , равна

$$U_{\max} = U_{1\max} - U_{1\min} = 2U_+ (R_1/R_2).$$

## ГЛАВА 2. ЛОГИЧЕСКИЕ СХЕМЫ

### 2.1. Логические сигналы и вентили

*Цифровая логика* (алгебра логики, булева алгебра) оперирует только с двумя значениями переменных, соответствующих двум возможным числам или *логическим значениям*: 0 и 1. В результате цифровые логические схемы можно анализировать и синтезировать функционально, используя алгебру логики, таблицы и другие средства, способные описать взаимосвязь нулей и единиц в схеме.

Логическую величину 0 или 1 называют *двоичной цифрой* или *битом*. Для задания переменной, имеющей больше двух значений, используются наборы из  $n$  битов, позволяющие представить  $2^n$  различных значений.

Между двумя напряжениями, которые соответствуют значениям 0 и 1, имеется область неопределенности. Эта область необходима для того, чтобы состояния 0 и 1 можно было определить однозначно. Если область неопределенности, отделяющая состояния 0 от состояния 1, мала, то помехи могут привести к тому, что 0 может быть воспринят как 1 и наоборот.

Часто вместо 0 и 1 используют термины *низкий уровень* (LOW) и *высокий уровень* (HIGH):

низкий уровень – соответствует сигналу в диапазоне малых напряжений, который интерпретируется как логический 0;

высокий уровень – соответствует сигналу в диапазоне больших напряжений, который интерпретируется как логическая 1.

Присвоение значений 0 и 1 низкому и высокому уровням не является обязательным. Такой вариант взаимосвязи логических значений с уровнями напряжения называется *положительной логикой*. Соответствие, при котором значению 1 приписывается низкий уровень напряжения, а 0 – высокий уровень напряжения, называется *отрицательной логикой*.

Помехоустойчивость цифровых схем связана с тем, что каждому двоичному значению соответствует широкий диапазон напряжений, цифровая логика слабо чувствительна к замене компонентов, изменением напряжения питания и шумам.

Логическую схему можно представить просто в виде *черного ящика* с конечным числом входов и выходов. На рис. 2.1 показана логическая схема с двумя входами и четырьмя выходами. Однако такое представление логической схемы не позволяет ответить на вопрос о том, каким будет выходной сигнал при той или иной комбинации входных сигналов.

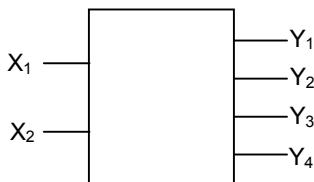


Рис. 2.1. Логическая схема  
(любое из напряжений может принимать только два значения)

Точное описание электрических процессов в электронной схеме требует привлечения большого количества информации. Однако поскольку на входы цифровой логической схемы поступают только дискретные сигналы 0 и 1, и на выходах получаем дискретные сигналы, то *логическую операцию*, реализуемую схемой, можно описать либо с помощью логической функции, либо с помощью таблицы. При этом в обоих случаях игнорируются электрические процессы в схеме и представлены только дискретные значения 0 и 1.

Логическая схема, для которой значения выходных сигналов однозначно определяется комбинацией значений ее входных сигналов в данный момент времени, называется *комбинационной схемой*. Операцию, выполняемую такой схемой, можно полностью описать *таблицей истинности*, в которой перечислены все комбинации входных сигналов и соответствующие им значения сигналов на выходах. Табл. 2.1 представляет собой пример таблицы истинности для логической схемы с двумя входами  $X_1$  и  $X_2$  и четырьмя выходами  $Y_1$ ,  $Y_2$ ,  $Y_3$  и  $Y_4$ , например, такой, какая приведена на рис. 2.1.

Таблица истинности для комбинационной логической схемы

$X_1$	$X_2$	$Y_1$	$Y_2$	$Y_3$	$Y_4$
0	0	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

В алгебре логики доказывается теорема: *Любая логическая функция может быть представлена суперпозицией трех функций – логического сложения или дизъюнкции (ИЛИ), логического умножения или конъюнкции (И) и отрицания или инверсии (НЕ)* (ее доказательство можно найти в [1]).

Таким образом, для построения любой комбинационной схемы достаточно только трех основных логических схем, реализующих функции И, ИЛИ и НЕ. На рис. 2.2 приведены условные обозначения логических схем (вентилей) и соответствующие им таблицы истинности. Обозначения и таблицы истинности для схем И и ИЛИ могут быть распространены на любое число входов. Функции, реализуемые схемами, определяются следующим образом:

- **Схема И (AND)** вырабатывает 1 на выходе только в том случае, когда на всех ее входах присутствуют 1. Логическая функция, реализуемая схемой И имеет вид  $Y = X_1 \cdot X_2 \cdot \dots \cdot X_m$ .
- **Схема ИЛИ (OR)** вырабатывает 1 на выходе только в том случае, когда 1 присутствует хотя бы на одном ее входе. Логическая функция, реализуемая схемой ИЛИ имеет вид  $Y = X_1 + X_2 + \dots + X_m$ .
- **Схема НЕ (NO)** вырабатывает на выходе сигнал, противоположный входному сигналу. Логическая функция, реализуемая схемой ИЛИ имеет вид  $Y = \bar{X}$ .

Кружок, изображенный на выходе инвертора является *символом инверсии* и используется в изображениях логических элементов для обозначения операции *инвертирования*.

При определении функций логических схем И и ИЛИ достаточно задать только условия на входе, при которых на выходе вырабатывается 1, поскольку в случае, когда выходной сигнал не равен 1, существует лишь одна возможность: он равен 0.

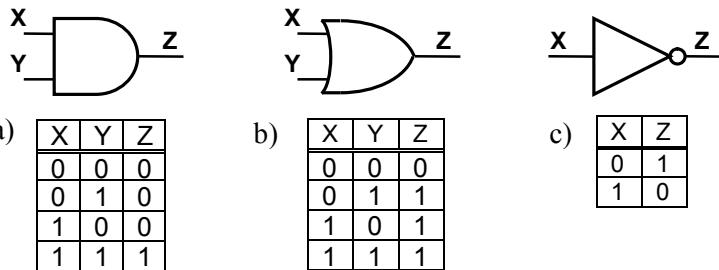


Рис. 2.2. Основные логические элементы и описывающие их таблицы истинности: а) И; б) ИЛИ; в) НЕ (инвертор)

Непосредственное сравнение таблиц истинности, задающих логические функции И и ИЛИ, показывают, что выполняются законы инверсии (формулы де Моргана):

$$\overline{X_1 \cdot X_2} = \overline{X_1} + \overline{X_2},$$

$$\overline{X_1 + X_2} = \overline{X_1} \cdot \overline{X_2}.$$

Логические функции, изображенные в левой части приведенных равенств, представляют собой объединение функции НЕ с функциями И и ИЛИ. На рис. 2.3 показаны условные обозначения и таблицы истинности для этих схем; функции, реализуемые этими схемами, также легко описать словами:

- **Схема И-НЕ (NAND)** вырабатывает на выходе сигнал, противоположный сигналу на выходе схемы И, то есть 0 только в том случае, когда на всех ее входах присутствуют 1.
- **Схема ИЛИ-НЕ (NOR)** вырабатывает на выходе сигнал, противоположный сигналу на выходе схемы ИЛИ, то есть 0 только в тех случаях, когда хотя бы на одном из ее входов присутствует 1.

Так же, как для схем И и ИЛИ, условные обозначения и таблицы истинности можно расширить на любое число входов для схем И-НЕ и ИЛИ-НЕ.

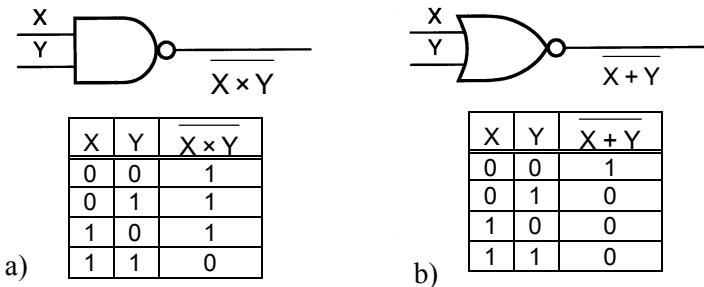


Рис. 2.3. Логические схемы с инверсией и описывающие их таблицы истинности: а) И-НЕ; б) ИЛИ-НЕ

На рис. 2.4 показано применение логических схем И и НЕ для синтеза схемы, реализующей функции  $Y_1, Y_2, Y_3$  и  $Y_4$ , соответствующие таблице истинности, приведенной в табл. 2.1.

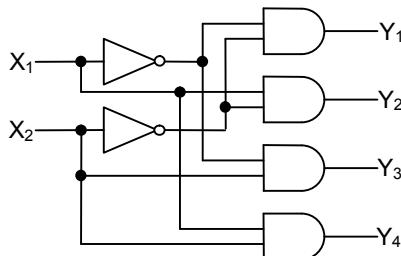


Рис. 2.4. Логическая схема, соответствующая таблице истинности в табл. 2.1

В реальных логических схемах сигнал одного уровня изменяется на сигнал другого уровня не мгновенно. На рис. 2.5 в качестве примера приведены временные диаграммы, показывающие возможную реакцию схемы, изображенной на рис. 2.4, при

изменении во времени комбинации входных сигналов. Из временных диаграмм видно, что имеется запаздывание между изменением сигналов на входе и соответствующим изменением выходного сигнала. В последующих разделах будут рассмотрены причины этих задержек, оценка их величины и влияние задержек на работу реальных схем.

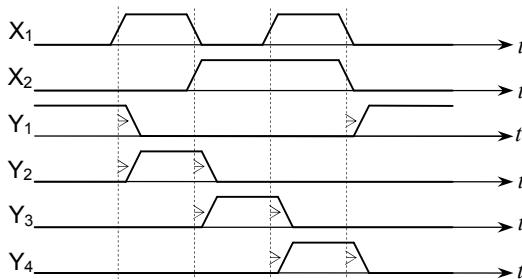


Рис. 2.5. Временные диаграммы для логической схемы, изображенной на рис. 2.4

Чтобы понимать логику работы цифровых схем, можно ничего не знать об аналоговой электронике. Но когда в процессе синтеза и отладки цифровой техники приходится учитывать временные соотношения, необходимо рассматривать аналоговые процессы, ограничивающие быстродействие цифровых схем.

## 2.2. Семейства логических схем

Первые семейства интегральных логических схем появились в начале 60-х годов прошлого века. *Семейством логических схем* называют набор различных интегральных схем (ИС), имеющих взаимно совместимые входные, выходные характеристики, выполняющих различные логические функции. Микросхемы одного семейства можно соединять между собой для реализации любой желаемой логической функции. Микросхемы разных семейств могут быть не совместимы из-за различных напря-

жений питания или из-за различных уровней, представляющих логические значения.

Одним из наиболее удачных оказалось *семейство логических схем на биполярных транзисторах – транзисторно-транзисторная логика (ТТЛ, TTL)*. Первые представители этого семейства были относительно медленными и потребляли значительную мощность. В настоящее время ТТЛ-схемы представлены несколькими семействами логических схем, совместимых друг с другом, но отличающихся быстродействием и потребляемой мощностью. В разных частях цифровой системы могут быть использованы микросхемы различных ТТЛ-семейств в соответствии с требованиями проекта. В 1990-е годы ТТЛ-схемы были в значительной степени вытеснены КМОП-схемами.

*Полевой транзистор со структурой «металл–окисел–полупроводник», или МОП-транзистор* был изобретен за десять лет до биполярного транзистора. Однако до 1960-х годов технология не позволяла производить МОП-транзисторы, способные конкурировать с биполярными транзисторами. Первое время логические схемы на МОП-транзисторах значительно отставали от биполярных интегральных схем по быстродействию и применялись там, где требовалась малая потребляемая мощность и большая степень интеграции, но не требовалось высокого быстродействия.

Начиная с середины 1980-х годов, прогресс в создании МОП-схем, особенно *комплементарных МОП-схем (КМОП-схем)*, позволил значительно улучшить их характеристики, и эти схемы постепенно стали вытеснять логические схемы ТТЛ-семейств. В настоящее время практически во всех сверхбольших интегральных схемах (микропроцессоры, микроконтроллеры, блоки памяти, программируемые логические схемы) использована КМОП-технология. Замена элементной базы, использующей ТТЛ-технологию, на элементную базу на основе КМОП-схем позволяет повысить быстродействие устройств и одновременно снизить потребляемую мощность и стоимость.

## 2.3. Логические КМОП-схемы

Принцип действия логических схем этого семейства основан на совместном использовании МОП-транзисторов с индуцированными каналами  $p$ - и  $n$ -типа, свойства которых дополняют друг друга (термин *комплементарный* означает *использующий принцип дополнения*). Основной особенностью логических схем этого типа является очень малая мощность, потребляемая от источников питания, зависящая от частоты переключений. Прежде, чем изучать логические КМОП-схемы, необходимо рассмотреть логические уровни, используемые в этих схемах.

### 2.3.1. Логические уровни КМОП-схем

Логические функции оперируют двоичными цифрами 0 и 1, а реальные логические схемы, реализующие эти функции, имеют дело с электрическими сигналами в виде уровней напряжения. В любой логической схеме имеется диапазон напряжений, соответствующий логическому 0, и не перекрывающийся с ним диапазон напряжений, соответствующий логической 1.

Диапазон напряжений питания различных семейств логических КМОП-схем достаточно широк – от 2.7 В до 12 В. Например, для семейства 74LVC напряжение питания равно 3.3 В. Такая схема воспринимает любое напряжение в диапазоне 0...0.8 В как логический 0 и напряжение в диапазоне 2.0...3.3 В – как логическую 1 (рис. 2.6).

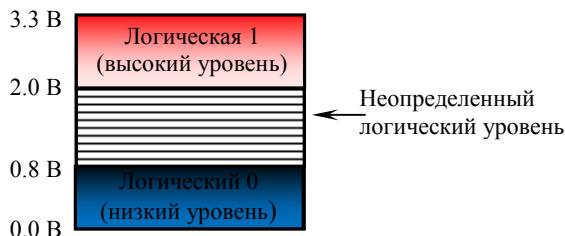


Рис. 2.6. Логические уровни для логических КМОП-схем семейства 74LVC

Предполагается, что напряжение не окажется в промежуточной области (0.8...2.0 В), кроме тех интервалов времени, когда сигнал переходит от одного уровня к другому; в противном случае логические значения будут не определены (то есть схема может интерпретировать их и как 0, и как 1). У КМОП-схем с другими напряжениями питания, например, 5.0 или 2.7 вольта, имеется аналогичное разделение диапазонов напряжений.

### 2.3.2. МОП-транзисторы, используемые в логических схемах

Стоковые характеристики МОП-транзистора с индуцированным каналом *n*-типа имеют вид, показанный на рис. 2.7.

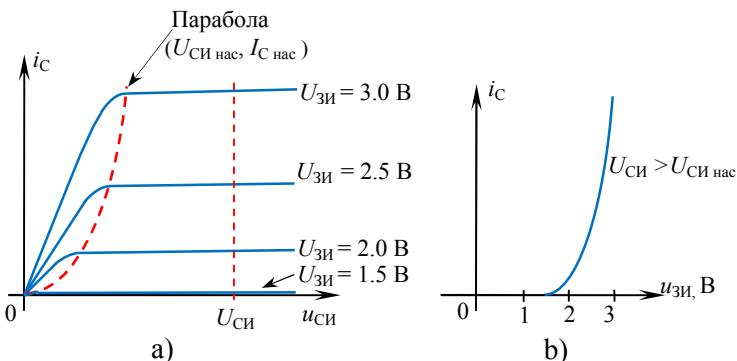


Рис. 2.7. Выходная (а) и проходная (б) характеристики МОП-транзистора с индуцированным каналом *n*-типа

Поскольку проводящий канал возникает лишь при подаче на затвор положительного напряжения больше 1.5 В, такой транзистор носит название МОП-транзистора с *индуцированным каналом*. На рис. 2.7б представлена проходная характеристика  $i_C = F(U_{\text{зи}})$  такого транзистора ( $U_{\text{си}} > U_{\text{си нас}}$ ).

МОП-транзистор можно представить как устройство, которое действует подобно управляемому напряжением резистору. В логических схемах МОП-транзистор работает так, что его сопротивление либо очень велико (при этом говорят, что транзи-

стор закрыт), либо очень мало (при этом говорят, что транзистор открыт).

Существуют два типа МОП-транзисторов: с *n*-каналом и с *p*-каналом; названия определяются типом примесей, введенных в легированный кремний в области под диэлектриком, используемой в качестве канала. Условное обозначение *МОП-транзистора с каналом n-типа* приведено на рис. 2.8. Выводы имеют следующие названия: *затвор*, *исток* и *сток*. В нормальных условиях потенциал стока выше потенциала истока.

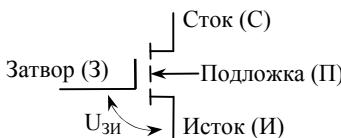


Рис. 2.8. Условное обозначение МОП-транзистора с каналом *n*-типа

Напряжение между затвором и истоком ( $U_{зи}$ ) у МОП-транзистора с каналом *n*-типа в логических схемах обычно равно нулю или положительно. Если  $U_{зи} = 0$ , то сопротивление между стоком и истоком ( $R_{си}$ ) очень велико и составляет, по крайней мере,  $10^6$  Ом или больше. По мере увеличения  $U_{зи}$  (то есть с увеличением напряжения на затворе)  $R_{си}$  уменьшается до очень малого значения порядка 10 Ом, а у некоторых транзисторов и меньше.

Условное обозначение *МОП-транзистора с каналом p-типа* приведено на рис. 2.9. Его работа аналогична работе МОП-транзистора с каналом *n*-типа, за исключением того, что обычно исток имеет потенциал более высокий, чем сток, а  $U_{зи}$  равно нулю или отрицательно. Если  $U_{зи}$  равно нулю, то сопротивление между истоком и стоком ( $R_{си}$ ) очень велико. С уменьшением  $U_{зи}$  (когда напряжение на затворе становится более отрицательным)  $R_{си}$  уменьшается, принимая очень малое значение.



Рис. 2.9. Условное обозначение МОП-транзистора с каналом *p*-типа

Затвор МОП-транзистора называют изолированным, поскольку он отделен от истока и стока диэлектриком, имеющим очень большое сопротивление. По этой причине, независимо от напряжения на затворе, ток практически не течет от затвора к истоку или от затвора к стоку. Сопротивления между затвором и другими выводами очень велики – десятки и сотни МОм. Ток, протекающий по этим сопротивлениям, очень мал, обычно меньше одного микроампера, и называется *током утечки*. Однако между затвором и истоком, а также между затвором и стоком имеется емкостная связь. В быстродействующих схемах мощность, расходуемая при заряде и разряде этих емкостей при каждом изменении входного сигнала, составляет заметную долю потребляемой схемой мощности.

### 2.3.3. Схема КМОП-инвертора

Схемы *КМОП-логики* (*CMOS logic*) образуются в результате совместного использования дополняющих друг друга *n*МОП- и *p*МОП-транзисторов. Простейшей КМОП-схемой является инвертор, который представляет собой последовательное соединение двух комплементарных транзисторов, как показано на рис. 2.10а. Напряжение питания  $U_{\Pi}$  обычно может быть в диапазоне от 2 до 12 В и часто выбирается равным 5.0 В для совместимости с ТТЛ-схемами.

В идеальном случае (у открытого транзистора сопротивление  $R_{\text{си}} = 0$ , а у закрытого оно бесконечно велико) поведение КМОП-инвертора можно описать таблицей, приведенной на рис. 2.10б:

- Если  $U_{IN}$  равно 0 В, то нижний *n*-канальный транзистор  $Q1$  закрыт, так как у него напряжение  $U_{3и}$  равно 0 В, а верхний *p*-канальный транзистор  $Q2$ , открыт, так как у него напряжение  $U_{3и}$  имеет большое по величине отрицательное значение (-5.0 В). Поэтому сопротивление между стоком и истоком транзистора  $Q2$ , включенного между шиной питания ( $U_{\Pi} = +5.0$  В) и выходом ( $U_{OUT}$ ), мало, и выходное напряжение равно 5.0 В.

- В том случае, когда  $U_{IN}$  равно 5.0 В открыт транзистор  $Q1$ , поскольку у него напряжение  $U_{3и}$  равно +5.0 В, а транзистор  $Q2$

закрыт, так как у него  $U_{\text{зи}}$  равно 0 В. Таким образом, транзистор  $Q1$  между стоком и истоком представляет собой малое сопротивление, и выходное напряжение равно 0 В.

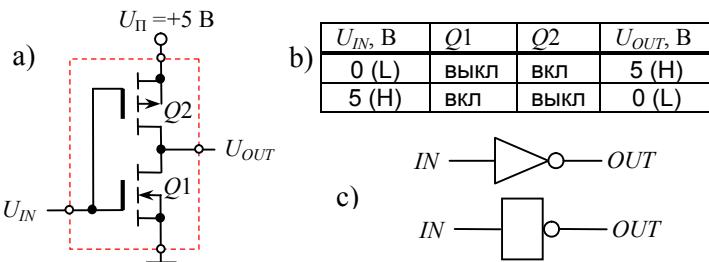


Рис. 2.10. КМОП-инвертор: а) принципиальная схема; б) таблица, описывающая работу схемы (L – низкий уровень, H – высокий уровень, вкл – открыт, выкл – закрыт); в) условные обозначения

В литературе, например [5], часто используется наглядный способ представления работы КМОП-схемы, состоящий в изображении транзисторов в виде ключей. Как показано на рис. 2.11а,  $n$ -канальный (нижний) транзистор заменяется ключом с нормально разомкнутым контактом, а  $p$ -канальный (верхний) транзистор – нормально замкнутым ключом. При подаче на вход высокого напряжения состояние каждого из ключей изменяется на состояние, противоположное первоначальному, как показано на рис. 2.11б.

Модель с ключами позволяет наглядно показать работу КМОП-инвертора. На рис. 2.12 показано влияние входного напряжения на состояние  $p$ - и  $n$ -канальных транзисторов. Когда к затвору  $n$ -канального транзистора ( $Q1$ ) приложено напряжение высокого уровня, он находится в *открытом* состоянии и ток течет от стока к истоку. Противоположная ситуация наблюдается в отношении  $p$ -канального транзистора ( $Q2$ ). Он находится в *открытом* состоянии, когда к его затвору приложено напряжение низкого уровня.

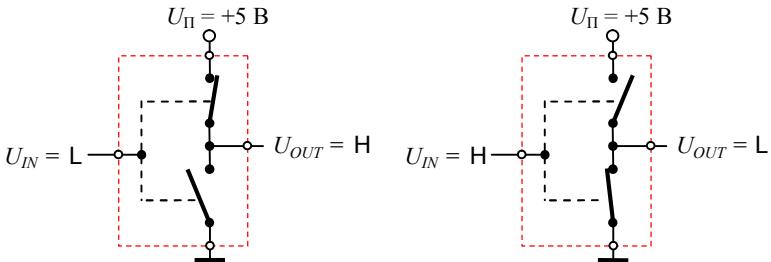


Рис. 2.11. Модель КМОП-инвертора с использованием ключей:  
а) низкое входное напряжение; б) высокое входное напряжение

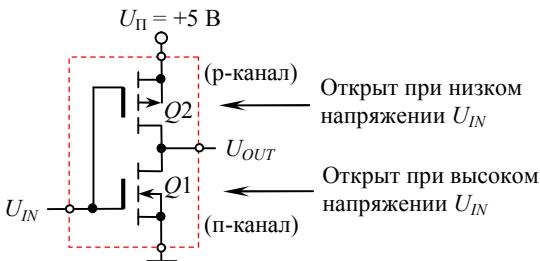


Рис. 2.12. Иллюстрация логики работы КМОП-инвертора

### 2.3.4. КМОП-схемы И-НЕ и ИЛИ-НЕ

Используя комплементарные МОП-транзисторы, можно создать схемы И-НЕ и ИЛИ-НЕ. Чтобы синтезировать логический элемент с  $k$ -входами, необходимо  $k$   $p$ -канальных и  $k$   $n$ -канальных транзисторов. На рис. 2.13 показана 2-входовая КМОП-схема И-НЕ. Если на одном из входов сигнал имеет низкий уровень, то выход  $Z$  через малое сопротивление *открытого p-канального транзистора* подключен к шине питания  $U_{\Pi}$ , а цепь на землю разомкнута *закрытым n-канальным транзистором*. Если на обоих входах присутствует сигнал высокого уровня, то цепь со стороны шины питания  $U_{\Pi}$  разорвана и выход  $Z$  через малое сопротивление

ние открытого *n*-канального транзистора подключен к земле. Рис. 2.14 иллюстрирует работу схемы И-НЕ на модели с ключами.

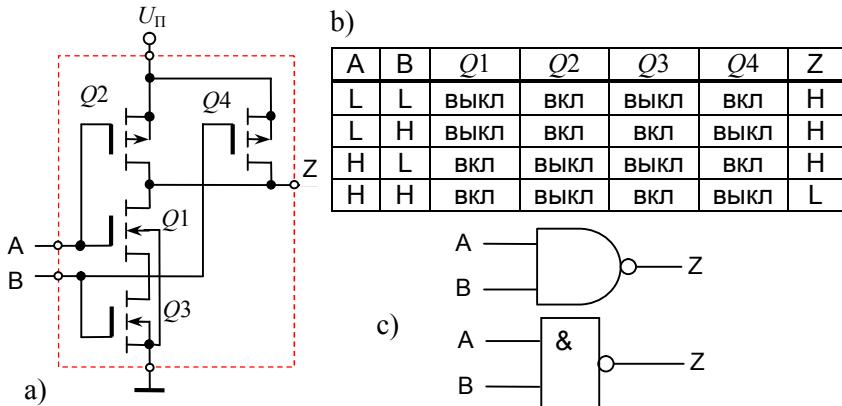


Рис. 2.13. Схема 2-входового КМОП-вентиля И-НЕ: а) принципиальная схема; б) таблица истинности; в) условные обозначения

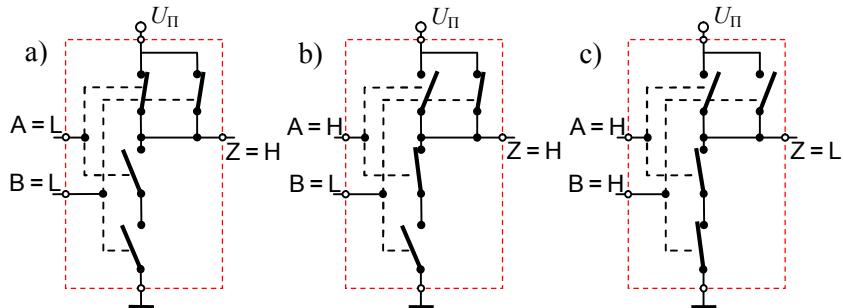


Рис. 2.14. Модель 2-входовой КМОП-схемы И-НЕ на основе ключей:  
 а) сигналы на обоих входах имеют низкий уровень (L);  
 б) сигнал на одном входе имеет высокий уровень (H);  
 в) сигналы на обоих входах имеют высокий уровень

Схема ИЛИ-НЕ с использованием комплементарных транзисторов приведена на рис. 2.15. Если сигналы на обоих входах имеют низкий уровень, то выход  $Z$  через малые сопротивления *открытых p-канальных транзисторов* подключен к шине питания  $U_{\Pi}$ , а цепь на землю разорвана *закрытыми n-канальными транзисторами*. Если сигнал хотя бы на одном из входов принимает высокий уровень, то цепь в сторону шины питания  $U_{\Pi}$  разомкнута и выход  $Z$  через малое сопротивление между стоком и истоком открытого *n-канального транзистора* подключен к земле.

КМОП-схемы И-НЕ и ИЛИ-НЕ имеют различные характеристики. При одной и той же площади кремниевого кристалла сопротивление между стоком и истоком *открытого транзистора с каналом n-типа* меньше, чем у транзистора с каналом *p-типа*. Поэтому у последовательно включенных  $k$  транзисторов с *n-каналом* сопротивление в *открытом состоянии* меньше, чем у  $k$  транзисторов с *p-каналом*. В результате, как будет показано ниже, быстродействие схемы И-НЕ с  $k$  входами выше, чем у  $k$ -входовой схемы ИЛИ-НЕ, и поэтому предпочтительнее использовать схемы И-НЕ.

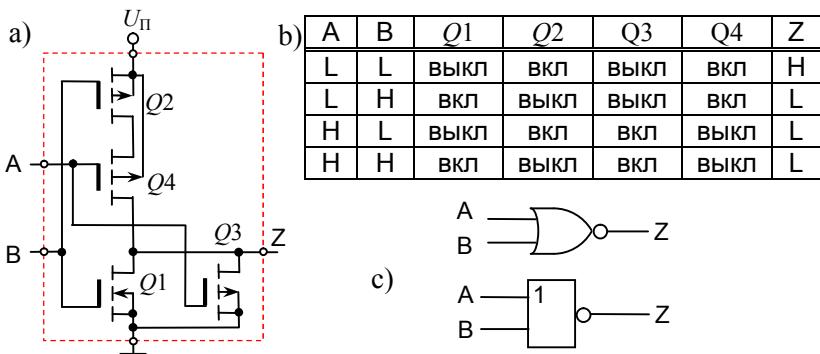


Рис. 2.15. 2-входовая КМОП-схема ИЛИ-НЕ:

- а) принципиальная схема;
- б) таблица, описывающая работу схемы;
- с) условные обозначения

## 2.4. Связь помехоустойчивости логических схем с логическими уровнями

В таблице на рис. 2.10б поведение КМОП-инвертора определено только при двух дискретных уровнях напряжения входного сигнала. Какими будут выходные напряжения при других входных напряжениях таблица ответа не дает. На рис. 2.16 изображена передаточная характеристика инвертора от входа до выхода при различных значениях входного напряжения для КМОП-схем, имеющих напряжение питания 5 В.

Согласно рис. 2.16 низким уровнем входного напряжения КМОП-схемы можно считать любое напряжение ниже 2.4 В, а высокому уровню может соответствовать любое напряжение выше 2.6 В. В таком случае при напряжении на входе между 2.4 В и 2.6 В выходное напряжение инвертора трудно сопоставить какому-либо логическому уровню.

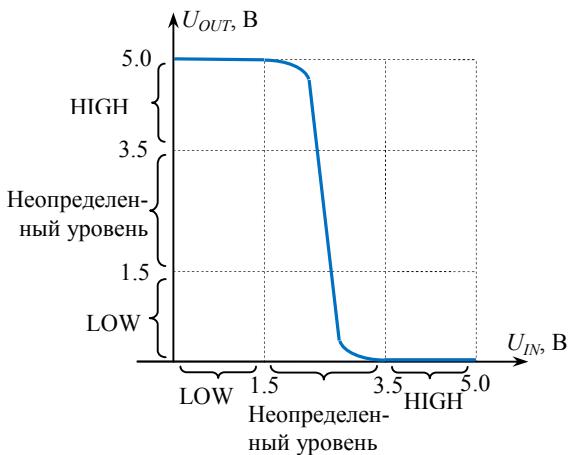


Рис. 2.16. Типичная передаточная характеристика КМОП-инвертора от входа до выхода

Однако типичная передаточная характеристика, показанная на рис. 2.16, не гарантирует, что все изготовленные микросхемы обладают одинаковыми переходными характеристиками. Кроме

того, при изменении напряжения источника питания, температуры и нагрузки на выходе передаточная характеристика значительно изменяется. Практика заставляет не ориентироваться на типичные характеристики при определении низкого уровня и высокого уровня. Принятые безопасные уровни для различных семейств КМОП-логики приведены на рис. 2.17. Эти параметры указываются изготовителями КМОП-схем в справочных данных и определяются следующим образом:

$U_{OH\min}$  – минимальное выходное напряжение высокого уровня;

$U_{IH\min}$  – минимальное входное напряжение, гарантированно опознаваемое как высокий уровень;

$U_{IL\max}$  – максимальное входное напряжение, гарантированно опознаваемое как низкий уровень;

$U_{OL\max}$  – максимальное выходное напряжение низкого уровня.

Уровни входных напряжений определяются, в основном, порогами переключения транзисторов, а выходные напряжения зависят от сопротивлений открытых транзисторов.

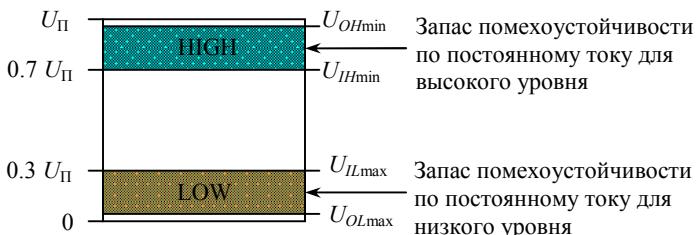


Рис. 2.17. Определение логических уровней и границы помехоустойчивости для КМОП-семейств

Изготовители КМОП-схем гарантируют значения параметров, приведенных на рис. 2.17, в некотором диапазоне температур и нагрузок на выходе. Напряжение источника питания  $U_\Pi$ , при котором схема сохраняет свои параметры, может изменяться в пределах  $\pm 10 \%$ .

Провод, подводящий напряжение питания  $U_{\Pi}$ , часто называют *шиной питания*. Гарантированные уровни КМОП-схем, как правило, являются функциями напряжения на шине питания:

$U_{OH\min}$	– $U_{\Pi} - 0.1$ В
$U_{IH\min}$	– 70 % от $U_{\Pi}$
$U_{IL\max}$	– 30 % от $U_{\Pi}$
$U_{OL\max}$	– земля + 0.1 В

Уровень помех, при котором в худшем случае напряжение, появляющееся на выходе схемы, не может быть правильно воспринято другой логической схемой, подключенной к выходу первой схемы, называется *запасом помехоустойчивости по постоянному току*. Для КМОП-схем серии 74HC, имеющих напряжение питания 5 В, напряжение  $U_{IL\max}$  низкого уровня (1.35 В) превышает напряжение  $U_{OL\max}$  (0.1 В) на 1.25 В, так что запас помехоустойчивости по постоянному току для низкого уровня равен 1.25 В. Запас помехоустойчивости по постоянному току для высокого уровня также равен 1.25 В. При подключении к выходам схем, принадлежащих КМОП-семействам, входов только КМОП-схем обеспечивается очень высокая помехоустойчивость по постоянному току.

Потребляемый входной ток КМОП-схем не зависит от входного напряжения. Он определяется током утечки транзисторов и очень мал. Изготовители указывают следующие максимальные значения входного тока:

$I_{IH}$  – максимальный входной ток при высоком уровне сигнала на входе.

$I_{IL}$  – максимальный входной ток при низком уровне сигнала на входе.

Входной ток для схем 74HC не превышает нескольких мкА, поэтому вход КМОП-схемы при любом входном сигнале потребляет от источника сигнала очень небольшую мощность. Это является одним из наиболее существенных достоинств КМОП-схем по сравнению с логическими схемами на биполярных транзисторах, таких как ТТЛ-схемы, входные токи которых достаточно велики как при низком, так и при высоком уровне входного сигнала.

## 2.5. Неиспользуемые входы логических схем

В реальных устройствах возникают ситуации, когда требуется логическая схема с  $n$  входами, а в наличии имеется только схема с  $n + 1$  входами, т.е. используются не все входы логического вентиля. Объединение двух входов в схеме с  $n + 1$  входами превращает ее в схему с  $n$  входами. На рис. 2.18а приведена схема И-НЕ с объединенными входами.

Возможен и другой вариант сокращения числа входов – подать на неиспользуемые входы напряжения, соответствующие логическим значениям. На неиспользуемый вход схем И и И-НЕ следует подавать логическую 1, как показано на рис. 2.18б, а на неиспользуемый вход схем ИЛИ и ИЛИ-НЕ необходимо подать логический 0, как на рис. 2.18с. В схемах, работающих с высокой частотой изменения сигналов на входах лучше применять способы, указанные на рис. 2.18б или 2.18с, а не объединять неиспользуемый вход с другим, как показано на рис. 2.18а.

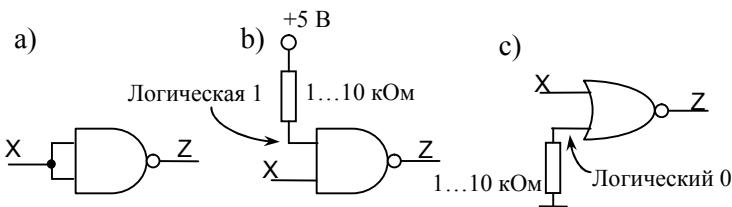


Рис. 2.18. Подключение неиспользуемых входов:

- а) объединение с другим входом;
- б) схема И-НЕ с одним из входов, подключенным к шине питания;
- в) схема ИЛИ-НЕ, один из входов которой заземлен

При объединении входов увеличивается емкостная нагрузка на логический элемент, к выходу которого эта схема подключена, что замедляет его работу. В вариантах 2.18б и 2.18с обычно используется резистор, имеющий сопротивление порядка нескольких кОм; к одному резистору можно подключить несколько

неиспользуемых входов. Возможен вариант соединения неиспользуемых входов непосредственно с шиной питания и землей.

Недопустимо оставлять неиспользуемые входы КМОП-схем ни к чему не подключенными. Может показаться, что такой *плавающий* вход будет вести себя так, будто к нему приложен низкий уровень сигнала, и обычно осциллограф или вольтметр, подключенные к такому входу, показывают значение 0 В. Это может наводить на мысль, что неиспользуемый вход схем ИЛИ и ИЛИ-НЕ можно оставлять неподключенными, потому что схема будет вести себя так, как будто на этот вход подан логический 0, и он не влияет на выходной сигнал вентиля. Однако входное сопротивление КМОП-схем очень велико, поэтому достаточно совсем небольших по мощности помех, чтобы время от времени незаземленный вход вел себя так, как если бы на нем был высокий уровень, создавая тем самым условия, при которых схема будет работать неустойчиво или совсем не будет работать.

Неподключенные входы КМОП-вентиляй часто являются причиной неустойчивой работы схемы, поскольку из-за шума потенциал неиспользуемого входа может значительно изменяться, и эти изменения сказываются в других местах схемы. Когда, пытаясь выяснить причину этой проблемы, вы касаетесь щупом осциллографа оставленного неподключенными входа, то емкости щупа часто бывает достаточно, чтобы ослабить шум и ликвидировать проблему. Однако отключение измерительного прибора снова приводит к неустойчивой работе схемы.

## 2.6. Схемы с тремя состояниями

Рассмотренные выше логические схемы имеют два состояния – низкий уровень выходного напряжения (LOW) и высокий уровень выходного напряжения (HIGH), соответствующие логическим значениям 0 и 1. Однако существует класс логических схем выходы которых могут находиться в третьем состоянии, называемом *высокоомным состоянием, состоянием Hi-Z*; этому состоянию не приписывается какое-либо логическое значение. В третьем состоянии выход ведет себя так, как будто он несоединен со схемой. При этом наблюдается малый ток утечки, вте-

кающий или вытекающий через выходной контакт. Итак, выход схемы с тремя состояниями может находиться в одном из трех возможных состояний – в состоянии соответствующем логическому 0, логической 1 или Hi-Z.

Схемы с тремя состояниями имеют дополнительный вход, называемый *входом разрешения выхода* или *входом запрещения выхода*. При подаче на этот вход соответствующего сигнала выход схемы можно перевести в высокоомное состояние.

Путем объединения нескольких выходов с тремя состояниями формируется *шина с тремя состояниями*. Блок, управляющий входами разрешения, должен гарантировать, что в любой момент времени не более одного выхода находится в состоянии логического 0 или логической 1. Остальные выходы, входящие в состав шины, должны находиться в состоянии Hi-Z. Задавать на шину логические уровни (высокий и низкий) может только одно устройство, на которое подан сигнал разрешения.

На рис. 2.19а показана принципиальная схема КМОП-буфера с тремя состояниями. Входящие в нее элементы И-НЕ, ИЛИ-НЕ и инвертор с целью упрощения схемы представлены условными обозначениями, а не в виде комбинации транзисторов; фактически в этой схеме используется 12 транзисторов.

Из таблицы на рис. 2.19б видно, что при низком уровне на входе разрешения EN (*Enable*) выходные транзисторы закрыты, и выход находится в третьем состоянии. При высоком уровне на входе разрешения EN на выходе возникает высокий уровень или низкий уровень в зависимости от сигнала, присутствующего на входе A. На рис. 2.19с приведено условное обозначение буфера с тремя состояниями.

Устройства с тремя состояниями на выходе проектируют так, чтобы задержка при подаче разрешающего сигнала, т.е. переход из третьего состояния на выходе к низкому уровню или к высокому уровню, была несколько больше, чем задержка при подаче запрещающего значения сигнала (переход от низкого уровня или от высокого уровня на выходе в третье состояние). Благодаря этому одновременная подача сигнала разрешения на управляющий вход одного устройства и запрещающего сигнала – на

управляющий вход другого устройства гарантирует, что второе устройство перейдет в третье состояние раньше, чем первое устройство выставит на шине высокий или низкий уровень.

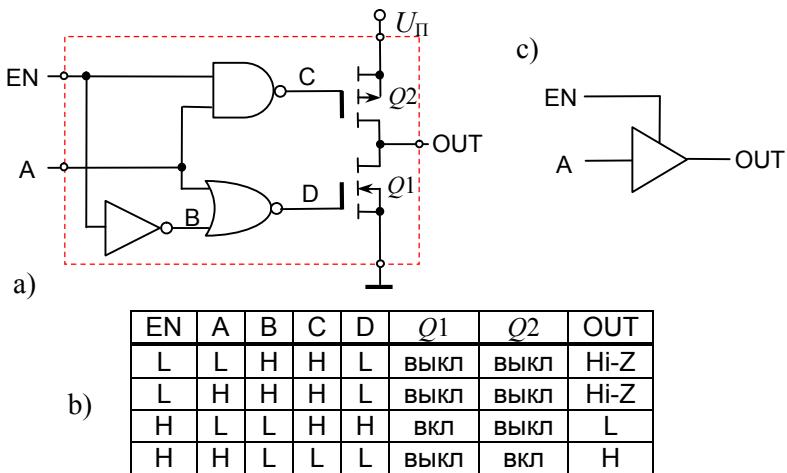


Рис. 2.19. КМОП-буфер с тремя состояниями:

- а) принципиальная схема;
- б) таблица, описывающая работу схемы;
- с) условное обозначение (A – вход данных, EN – управляющий вход, OUT – выход)

Если два выхода с тремя состояниями, подключенные к одной шине, в одно и то же время находятся в активном режиме и пытаются установить противоположные уровни, то возникает ситуация, когда на шине устанавливается напряжение, не соответствующее какому-либо логическому уровню. Если такое состояние имеет место в течение короткого времени, то схемы, вероятнее всего, не будут повреждены, но значительный ток, протекающий через соединенные между собой выходы, создает импульсную помеху, что может повлиять на работу других блоков в системе.

Когда КМОП-схема с тремя состояниями находится в третьем состоянии, по ее выходу течет ток утечки до 10 мА.

Этот ток, а также входные токи схем, подключенных к данному выходу, необходимо принять во внимание при расчете максимального числа устройств, которые можно подключить к шине с тремя состояниями. Другими словами, схема с тремя состояниями, которой разрешено установить на выходе низкий или высокий уровень, должна допускать втекание или вытекание по ее выходу токов утечки всех других выходов с тремя состояниями, подключенных к шине, каждый из которых может доходить до 10 мА, а также входных токов всех входов, подключенных к шине. Расчет следует проводить отдельно для низкого уровня и для высокого уровня.

## 2.7. Динамические свойства КМОП-схем

От характеристик схемы по переменному току или динамических характеристик зависит как быстродействие КМОП-схемы, так и потребляемая ею мощность. Нагрузка, подключенная к выходу логического элемента, также влияет на скорость изменения уровня сигнала на выходе. На этапе разработки плат необходимо учитывать влияние нагрузки на сигналы синхронизации, сигналы, передаваемые по протяженным шинам, и на другие сигналы, поступающие с выходов схем, к которым подключено большое число входов других логических элементов, а также во всех случаях, когда разводка осуществляется длинными соединениями.

Два временных параметра определяют быстродействие логических схем схемы: время переходного процесса и задержка распространения сигнала.

### 2.7.1. Переходные процессы в логических схемах

*Временем переходного процесса* называется интервал времени, в течение которого сигнал на выходе логической схемы изменяется от одного уровня до другого. На рис. 2.20а показано, как мог бы выглядеть сигнал на выходе логической схемы в идеальном случае при нулевом времени переходного процесса. В любой реальной схеме выходные сигналы не могут изменяться мгновенно, поскольку необходимо некоторое время для заряда

паразитной емкости, подключенной к выходу этой схемы. Эта емкость складывается из емкости проводников, соединяющих данный логический элемент с входами других элементов, и суммы входных емкостей всех подключенных элементов.

На рис. 2.20б изображен вид выходного сигнала при линейной аппроксимации переходных процессов в логическом элементе. В течение времени  $t_r$ , называемого *временем нарастания*, выходной сигнал изменяется от низкого уровня до высокого, а в течение времени  $t_f$ , называемого *временем спада*, сигнал изменяется от высокого уровня до низкого. Время  $t_f$  может отличаться от времени  $t_r$ .

Однако рис. 2.20б не отражает истинный процесс переключения, поскольку скорость изменения выходного напряжения не постоянна. Реально изменение напряжения в начале и в конце процесса переключения бывает медленнее, как показано на рис. 2.20с. Время нарастания и время спада напряжения на выходе определяют используя моменты достижения границ логических уровней, как показано на рисунке.

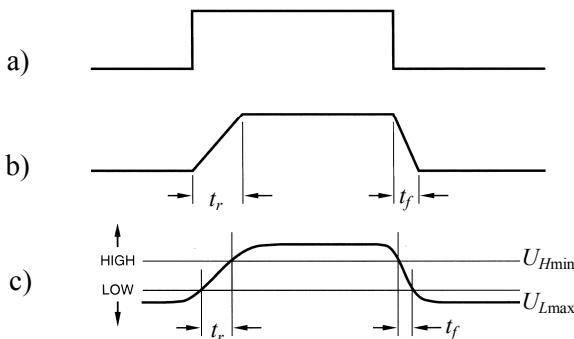


Рис. 2.20. Длительность переходного процесса:

- а) идеальный случай с нулевым временем переключения; б) линейная аппроксимация переходных процессов; в) реальная временная диаграмма

Время нарастания и время спада характеризуют насколько быстро выходное напряжение преодолевает область неопределенности между низким и высоким уровнями. Параметры  $t_r$  и  $t_f$ ,

не учитывают время изменения сигнала в пределах одного уровня. Эта часть переходного процесса является составной частью временного параметра, называемого задержкой распространения. Подробнее задержка распространения рассмотрена в 2.7.2.

Время нарастания и время спада сигнала на выходе КМОП-схемы определяются сопротивлением открытого транзистора и емкостью нагрузки. Наличие *паразитной* емкости обусловлено следующими причинами:

- Выходные цепи логических элементов, включающие выходные транзисторы, внутренние соединения и корпуса схем логических схем, имеют некоторую собственную емкость равную нескольким пикофарадам.
- Соединения на печатной плате какого-либо выхода с входами других схем имеют емкость около 0.5 пФ на сантиметр длины или больше в зависимости от технологии изготовления печатной платы.
- У большинства семейств логических схем входные цепи, включая транзисторы, внутренний монтаж и корпус микросхемы, имеют емкость несколько пикофарад на каждый вход.

На рис. 2.21 показана эквивалентная схема, воспользовавшись которой можно рассчитать время нарастания и время спада сигнала на выходе КМОП-вентиля.

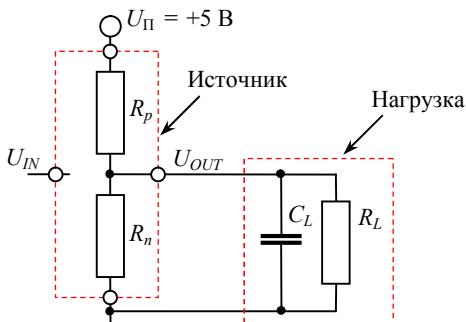


Рис. 2.21. Эквивалентная схема, используемая для расчета времени переходного процесса в выходной цепи КМОП-вентиля

Сопротивления между стоком и истоком  $p$ -канального и  $n$ -канального транзисторов заменены резисторами  $R_p$  и  $R_n$ . В зависимости от уровня сигнала на выходе сопротивление одного из резисторов велико, а другого – мало. Нагрузка на выходе представлена эквивалентной схемой нагрузки, состоящей из двух компонентов:

$R_L$  – резистор, отражает нагрузку по постоянному току; он определяет напряжения и токи в выходной цепи, когда сигнал на выходе имеет установившееся значение высокого уровня или низкого уровня. Поскольку у КМОП-схем входные сопротивления очень велики, то при изменении уровня сигнала на выходе нагрузка по постоянному току практически не оказывает влияния на переходной процесс;

$C_L$  – емкость, характеризует собой нагрузку по переменному току. Величина емкости определяет значения напряжений и токов в выходной цепи в процессе изменения выходного сигнала.

Определим время переходного процесса на выходе КМОП-вентиля. Предположим, что  $C_L = 20 \text{ пФ}$ , а сопротивления открытых  $p$ -канальных и  $n$ -канальных транзисторов равны  $200 \text{ Ом}$  и  $100 \text{ Ом}$  соответственно.

Рассчитаем время спада сигнала от уровня логической 1 до уровня логического 0. На рис. 2.22а изображено состояние схемы, предшествующее моменту изменения сигнала на выходе с высокого уровня на низкий (считаем  $R_L = \infty$ .)

Ради упрощения задачи предположим, что транзисторы в КМОП-схеме переходят из *открытого* состояния в *закрытое* и обратно мгновенно. С момента времени  $t = 0$ , когда начинается изменение сигнала на выходе КМОП-вентиля, справедлива эквивалентная схема, показанная на рис. 2.22б.

В момент времени  $t = 0$  напряжение  $U_{OUT}$  равно  $5.0 \text{ В}$ , поскольку напряжение на конденсаторе не может изменяться мгновенно. При  $t = \infty$  конденсатор должен полностью разрядиться и напряжение  $U_{OUT}$  будет равно  $0$ . Разряд конденсатора происходит по экспоненциальному закону:

$$U_{OUT} = U_{\Pi} \cdot \exp\left(\frac{-t}{\tau_{pa3}}\right), \quad (2.1)$$

где  $\tau_{pa3} = R_n C_L$  – постоянная времени разряда, равная в нашем случае 2 нс.

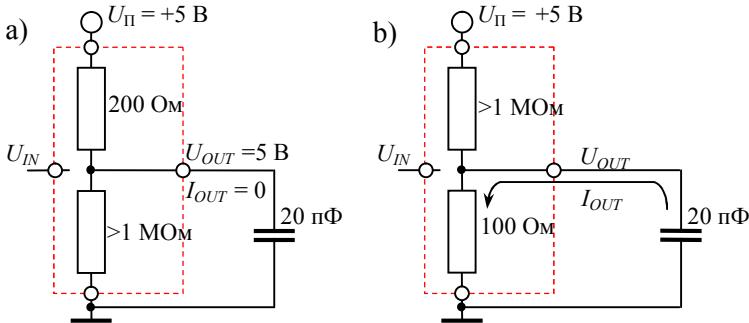


Рис. 2.22. Эквивалентные схемы выходной цепи КМОП-вентиля при переходе сигнала от высокого уровня к низкому:

- а) на выходе вентиля высокий уровень; б) состояние вентиля после того, как *p*-канальный транзистор оказывается *закрытым*, а *n*-канальный транзистор – *открытым*

На рис. 2.23 показана зависимость напряжения на выходе логической схемы  $U_{OUT}$  от времени. Время спада определяется как временной интервал между моментом пересечения напряжением  $U_{OUT}$  нижней границы высокого уровня сигнала  $U_{Hmin}$  и моментом пересечения выходным напряжением верхней границы низкого уровня сигнала  $U_{Lmax}$ .

Если напряжение питания  $U_{\Pi}$  равно +5 В, то значения границ для КМОП-схем равны  $U_{Hmin} = 3.5$  В и  $U_{Lmax} = 1.5$  В. Воспользовавшись соотношением (2.1) находим моменты времени в которые напряжение  $U_{OUT}$  пересекает указанные границы:

$$t = -R_n C_L \cdot \ln \frac{U_{OUT}}{U_{\Pi}},$$

$$t_{3.5} = 0.7 \text{ нс}, \quad t_{1.5} = 2.4 \text{ нс}.$$

Время спада согласно определению равно

$$t_f = t_{1.5} - t_{3.5} = 2.4 \text{ нс} - 0.7 \text{ нс} = 1.7 \text{ нс}.$$

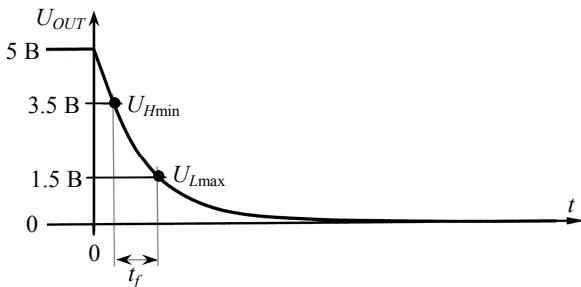


Рис. 2.23. Изменение напряжения на выходе КМОП-вентиля при переходе сигнала от высокого уровня к низкому уровню

Время нарастания сигнала рассчитывается аналогично. На рис. 2.24а изображено состояние схемы, предшествующее моменту изменения сигнала на выходе с низкого уровня на высокий.

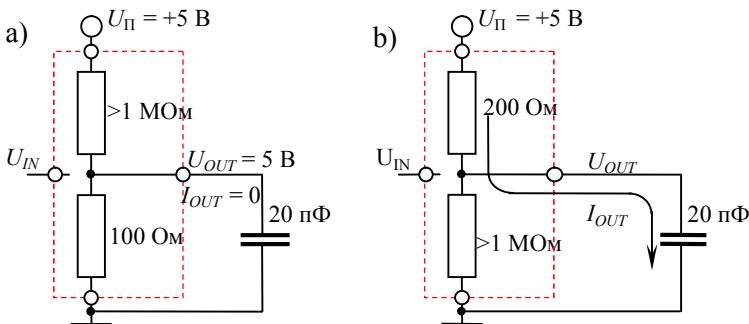


Рис. 2.24. Эквивалентные схемы выходной цепи КМОП-вентиля при переходе сигнала от низкого уровня к высокому:  
 а) низкий уровень на выходе; б) состояние схемы после того, как транзистор с  $n$ -каналом оказывается закрытым, а транзистор с  $p$ -каналом – открытым

Как и прежде, напряжение  $U_{OUT}$  не может измениться мгновенно, но при  $t \rightarrow \infty$  напряжение на конденсаторе  $U_{OUT}$  стремится к 5 В. Заряд конденсатора происходит по экспоненциальному закону:

$$U_{OUT} = U_{\Pi} \cdot \left( 1 - \exp\left(\frac{-t}{\tau_{\text{зап}}}\right) \right), \quad (2.2)$$

где  $\tau_{\text{зап}} = R_p C_L$  – постоянная времени заряда, равная в нашем случае 4 нс.

На рис. 2.25 показана зависимость напряжения  $U_{OUT}$  от времени. Чтобы получить время нарастания, воспользуемся выражением (2.2), откуда следует:

$$t = -R_p C_L \cdot \ln \frac{U_{\Pi} - U_{OUT}}{U_{\Pi}}.$$

При  $U_{OUT1} = U_{H\min} = 3.5$  В и  $U_{OUT2} = U_{L\max} = 1.5$  В получим  $t_{1,5} = 1.4$  нс,  $t_{3,5} = 4.8$  нс.

Время нарастания  $t_r$  определяется как разность между полученными двумя числами и равно 3.4 нс.

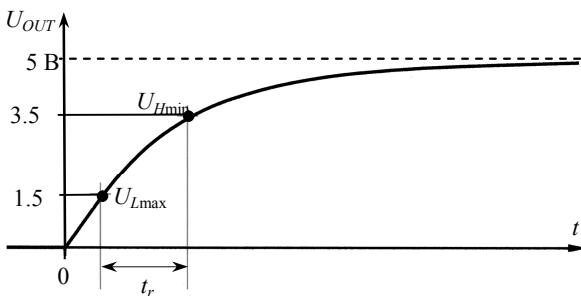


Рис. 2.25. Изменение напряжения на выходе КМОП-вентиля при переходе сигнала от низкого уровня к высокому

Время нарастания вдвое больше времени спада. Это объясняется тем, что мы предположили различающимися вдвое сопротивлениями *открытых p*-канальных и *n*-канальных транзисторов. В быстродействующих КМОП-устройствах *p*-канальные транзисторы иногда изготавливают большего размера, чтобы уменьшить его сопротивление в открытом состоянии и сделать времена переходов на выходе от высокого уровня к низкому уровню и обратно примерно равными.

С увеличением емкости нагрузки постоянная времени увеличивается и длительность переходных процессов на выходе растет. При создании быстродействующих схем необходимо минимизировать емкостную нагрузку уменьшая число входов, на которые поступает данный сигнал. Реализуется это путем создания нескольких копий сигнала, а также аккуратной разводкой схемы.

Расчетная длительность переходного процесса сильно зависит от выбора логических уровней. Если в рассмотренных примерах в качестве порогов высокого уровня и низкого уровня использовать значения 2 В и 3 В вместо 1.5 В и 3.5 В, то для каждого из переходных процессов получим меньшее время.

### 2.7.2. Задержка распространения сигнала

Время нарастания и время спада сигнала не полностью описывают поведение логического элемента в режиме переключения. В логическом элементе на пути от входа до выхода сигнал проходит через ряд электрических цепей. *Задержка распространения*  $t_p$  вдоль пути прохождения сигнала определяется временем, необходимым для того, чтобы изменение входного сигнала привело к изменению выходного сигнала.

Задержка распространения сигнала по заданному пути зависит от того, изменяется выходной сигнал с высокого уровня к низкому или наоборот. Сложный логический элемент с большим числом входов и выходов может иметь различные значения  $t_p$  при прохождении сигнала по различным путям. На рис. 2.26 указаны два различных значения задержки распространения от входа до выхода для КМОП-инвертора в зависимости от направления изменения сигнала на выходе:

$t_{pHL}$  – время между изменением сигнала на входе и соответствующим изменением сигнала на выходе при переходе выходного напряжения с высокого уровня на низкий;

$t_{pLH}$  – время между изменением сигнала на входе и соответствующим изменением сигнала на выходе при переходе выходного напряжения с низкого уровня на высокий.

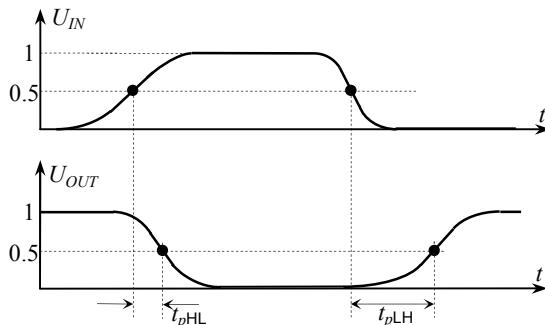


Рис. 2.26. Задержки распространения сигнала для КМОП-инвертора

Наличие задержки распространения вызвано многими причинами. Скорость изменения состояния транзисторов зависит как от физических процессов, происходящих в полупроводнике, так и от условия эксплуатации схемы. На величину задержки оказывает влияние скорость изменения входного сигнала, входная емкость логического элемента и характер нагрузки на выходе, которая в значительной степени определяет скорость изменения выходного сигнала. В многоуровневых схемах и в устройствах, реализующих сложные логические функции, часто требуется изменение состояния нескольких транзисторов, прежде чем уровень сигнала на выходе начнет изменяться. Каждый из перечисленных факторов вносит свой вклад в задержку распространения сигнала.

Производители в справочных данных логических микросхем обычно указывают задержку распространения по моментам перехода входного и выходного сигналов через среднюю точку, как показано на рис. 2.26, чтобы не учитывать влияния времени нарастания и времени спада. Однако когда на работе устройства

может отрицательно сказаться медленное нарастание и спад входного сигнала, задержки определяются и по моментам пересечения пороговых значений логических уровней. Например, для надежной работы некоторых схем длительность входного сигнала должна превышать некоторый минимум. В этом случае в справочных данных указывается минимальная длительность, необходимая для надежного срабатывания схемы, которая определяется как время между пересечениями некоторого логического уровня. Например, на рис. 2.27 показано определение минимальной длительности входного импульса по моментам пересечения высокого логического уровня.

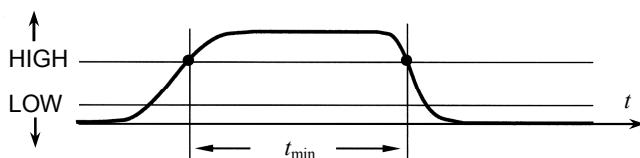


Рис. 2.27. Определение длительности интервала времени по граничным значениям логических уровней в наихудшем случае

Быстродействующие КМОП-схемы при слишком медленных изменениях входного сигнала могут потреблять чрезмерно большой ток, в связи с тем, что в течение некоторого времени оказываются открытыми оба выходных транзистора, или генерировать колебания. В связи с этим производитель может указывать абсолютные максимальные значения времени нарастания и спада входного сигнала, при которых гарантируется надежная работа.

## ГЛАВА 3. КОМБИНАЦИОННЫЕ СХЕМЫ

### 3.1. Дешифраторы и шифраторы

Дешифратором  $m \times n$  называется устройство с  $m$  входами  $X_i$  и  $n$  выходами  $Y_j$ ,  $n \leq 2^m$ , предназначенное для преобразования некоторой комбинации входных сигналов (кодового слова) в другую комбинацию выходных сигналов. Какая именно из возможных выходных комбинаций появится на выходах дешифратора зависит от вида входной кодовой комбинации.

Общая структура дешифратора приведена на рис. 3.1. Для того чтобы дешифратор нормально выполнял функцию отображения одной кодовой комбинации в другую, необходимо подать сигнал на вход разрешения EN, если такой имеется в данном дешифраторе. Если сигнал разрешения не активен, то выходы дешифратора находятся в третьем состоянии или отображают любое входное кодовое слово в некоторое запрещенное выходное кодовое слово состоящее, например, из всех нулей.

В большинстве случаев роль выходного кода играет  $n$ -разрядный код 1 из  $n$ , у которого в любой момент времени отличен от нуля один бит. Таким образом, в коде 1 из 4 с высоким активным уровнем сигнала на выходах кодовые слова имеют вид: 0001, 0010, 0100 и 1000. При низком активном уровне сигнала на выходах кодовые слова имеют вид: 1110, 1101, 1011 и 0111.

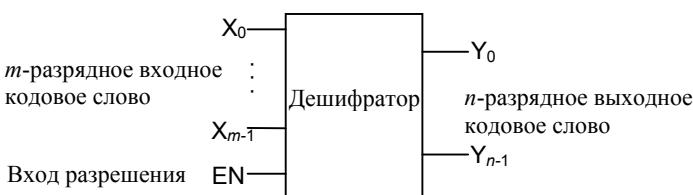


Рис. 3.1. Схематическое изображение дешифратора

Формально устройство, называемое *шифратором*, ничем не отличается от дешифратора. У него имеется  $n$  входов и  $m$  выходов, однако как правило, кодовая комбинация на выходе шиф-

ратора имеет меньшее число разрядов, чем входная кодовая комбинация.

### 3.1.1. Полные дешифраторы

Одним из наиболее распространенных является дешифратор  $m \times n$  или *полный дешифратор*. На вход такого дешифратора поступает  $m$ -разрядное двоичное кодовое слово, а на выходе возникает слово кода 1 из  $2^m$ . Полный дешифратор применяется в том случае, когда необходимо иметь активный уровень сигнала только на одном из  $2^m$  выходов, определяемом  $m$ -разрядным двоичным числом на входе.

Рассмотрим принцип построения полного дешифратора на примере  $m = 3$ ,  $n = 8$ . Записывая связь между входными и выходными сигналами, как показано в табл. 3.1, видим, что каждый из выходных сигналов может быть получен с помощью схемы И, на входы которой подаются либо входные сигналы, либо их инверсии и сигнал разрешения. На рис. 3.2 представлен вариант полного дешифратора, реализованный согласно таблице истинности 3.1.

Таблица 3.1

Таблица истинности для полного дешифратора  $3 \times 8$

Входы				Выходы							
EN	$X_0$	$X_1$	$X_2$	$Y_0$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Связи между входными и выходными сигналами для полного дешифратора  $3 \times 8$  имеют следующий вид:

$$\begin{array}{ll}
 Y_0 = \bar{X}_0 \cdot \bar{X}_1 \cdot \bar{X}_2 \cdot EN & Y_4 = \bar{X}_0 \cdot \bar{X}_1 \cdot X_2 \cdot EN \\
 Y_1 = X_0 \cdot \bar{X}_1 \cdot \bar{X}_2 \cdot EN & Y_5 = X_0 \cdot \bar{X}_1 \cdot X_2 \cdot EN \\
 Y_2 = \bar{X}_0 \cdot X_1 \cdot \bar{X}_2 \cdot EN & Y_6 = \bar{X}_0 \cdot X_1 \cdot X_2 \cdot EN \\
 Y_3 = X_0 \cdot X_1 \cdot \bar{X}_2 \cdot EN & Y_7 = X_0 \cdot X_1 \cdot X_2 \cdot EN
 \end{array}$$

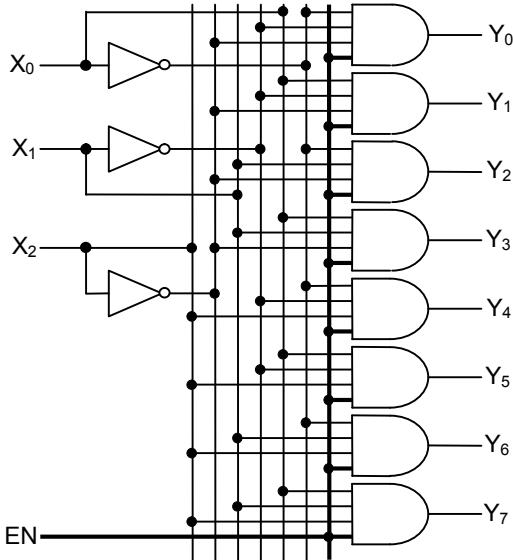


Рис. 3.2. Схема дешифратора  $3 \times 8$

Входное кодовое слово  $X_0, X_1, X_2$  можно представить как целое число из интервала от 0 до 7. Выходные кодовые слова  $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7$ , формируются по следующему правилу:  $Y_i$  равно 1 только в том случае, когда входное кодовое слово является двоичным представлением числа  $i$  и входной сигнал разрешения EN равен 1. Если EN = 0, то на всех выходах устанавливается логический 0.

В таблице истинности полного дешифратора среди входных комбинаций фигурирует символ  $\times$  безразличного значения. Если одно или большее число входных величин не влияет на зна-

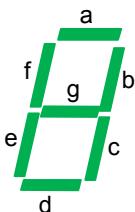
чения выходных сигналов при любой комбинации сигналов на остальных входах, то такие входные сигналы в данной комбинации отмечаются символом  $\times$ . Это правило позволяет значительно уменьшить число строк в таблице истинности и делает более ясной функцию, реализуемую данным устройством.

### 3.1.2. Неполные дешифраторы

Не всегда необходимо использовать все выходы дешифратора или даже декодировать все возможные входные комбинации. В том случае, когда  $n < 2^m$  дешифратор называют неполным. Примером может служить *двоично-десятичный дешифратор* или *двоично-десятичный дешифратор*  $4 \times 10$ , который декодирует только первые десять входных двоичных комбинаций 0000...1001, формируя на выходе сигналы  $Y_0 \dots Y_9$ . Выходы сигналов  $Y_{10} \dots Y_{15}$  у такого дешифратора отсутствуют, а при наличии на входах кодовых комбинаций 1010 ... 1111, сигналы на имеющихся выходах могут принимать некоторые наперед заданные значения, например, нулевые.

### 3.1.3. Дешифратор для семисегментных индикаторов

Существуют и более сложные дешифраторы, у которых единичные значения могут появляться одновременно на нескольких выходах. Таким дешифратором является преобразователь двоично-десятичного кода в семисегментный код с четырьмя входами и семьью выходами. Такой дешифратор применяется в устройствах индикации десятичных цифр на семисегментном световом табло. На рис. 3.3 дано изображение семисегментного индикатора и таблица истинности дешифратора для такого индикатора. Пользуясь таблицей истинности, не трудно составить принципиальную схему дешифратора.



Цифра	Двоично-десятичный вход				Семисегментный выход						
	$X_3$	$X_2$	$X_1$	$X_0$	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Рис. 3.3. Изображение семисегментного индикатора и таблица истинности дешифратора для такого индикатора

### 3.1.4. Шифраторы

Шифратор во многом напоминает дешифратор, однако у него код на выходе имеет меньшее число разрядов, чем код на входе. Простейшим является *двоичный шифратор* или шифратор  $2^n \times n$  (рис. 3.3а). Он реализует точно обратную функцию по отношению к полному дешифратору с кодом 1 из  $2^n$  на входе и  $n$ -разрядным двоичным кодом на выходе. Соотношения для шифратора  $8 \times 3$  с входными сигналами  $X_0 \dots X_7$  и выходными сигналами  $Y_0 \dots Y_2$  имеют вид

$$\begin{aligned} Y_0 &= X_1 + X_3 + X_5 + X_7, \\ Y_1 &= X_2 + X_3 + X_6 + X_7, \\ Y_2 &= X_4 + X_5 + X_6 + X_7. \end{aligned} \quad (3.1)$$

Принципиальная схема, реализующая приведенные выше соотношения, показана на рис. 3.3б. В общем случае шифратор  $2^n \times n$  можно построить, воспользовавшись  $n$  вентилями ИЛИ с  $2^{n-1}$  входами каждый. Если  $j$ -й разряд в двоичном представлении числа  $i$  равен 1, то  $j$ -й разряд входного кода поступает на  $j$ -й вентиль ИЛИ.

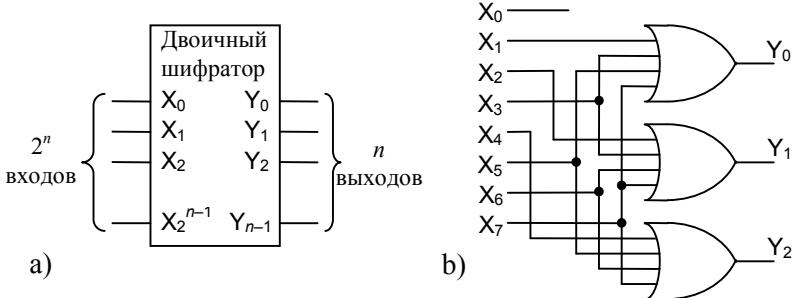


Рис. 3.3. Двоичный шифратор: а) условное обозначение;

б) шифратор  $8 \times 3$  на элементах ИЛИ

### 3.2.1. Приоритетные шифраторы

Выходные сигналы  $n$ -разрядного полного дешифратора, появляющиеся в любой момент времени только на одном из  $2^n$ , обычно используются для управления набором из  $2^n$  устройств в тех случаях, когда активным является только одно устройство. Существует и другая ситуация, когда имеется система с  $2^n$  *входами*, наличие сигнала на каждом из которых указывает на требования обслуживания. Такая структура, показанная на рис. 3.4 часто применяется в микропроцессорных системах ввода/вывода, где входные сигналы являются сигналами требования прерывания.

В ситуации, когда надо наблюдать за входами и принимать решение, какое из подключенных к ним устройств требует в данный момент времени обслуживания, кажется естественным применение двоичного шифратора, приведенного на рис. 3.3. Однако этот шифратор работает правильно только в том случае, если в любой момент времени требует обслуживания не более чем одно-го устройства. Если одновременно могут поступать запросы от нескольких устройств, то шифратор будет давать нежелательные результаты. Предположим, например, что на входах  $X_1$  и  $X_2$  шифратора  $8 \times 3$  одновременно присутствует 1; тогда на выходе появ-

вится комбинация 011, соответствующая двоичному представлению числа 3.

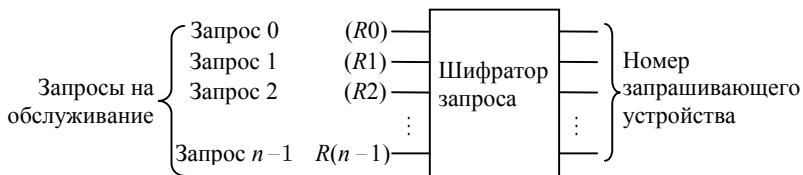


Рис. 3.4. Система с  $2^n$  запрашивающими устройствами и шифратором запроса, в любой момент времени показывающий, какой из сигналов запроса активен

При запросах от первого и второго устройств можно ожидать появления в двоичной записи числа 1 или 2, в то время как получено число 3. Простейший вариант решения указанной задачи состоит в присвоении каждому источнику запросов фиксированного приоритета. Например, группа из восьми запросов  $R7...R0$  ( $R$  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, то обрабатывается запрос с наибольшим номером. Приоритетный шифратор вырабатывает на выходе двоичный номер старшего запроса.

Схема, реализующая такой алгоритм, называется *приоритетным шифратором*.

Запишем логические выражения для выходных сигналов приоритетного шифратора. Предварительно введем восемь промежуточных переменных  $Z_0 - Z_7$  таких, что  $Z_i$  равно 1 только в том случае, когда  $X_i$  является входом с высшим приоритетом из числа тех, на которые поданы 1:

$$Z_7 = X_7$$

$$Z_6 = X_6 \cdot \bar{X}_7$$

$$Z_5 = X_5 \cdot \bar{X}_6 \cdot \bar{X}_7$$

$$Z_4 = X_4 \cdot \bar{X}_5 \cdot \bar{X}_6 \cdot \bar{X}_7$$

$$Z_3 = X_3 \cdot \bar{X}_4 \cdot \bar{X}_5 \cdot \bar{X}_6 \cdot \bar{X}_7$$

$$Z_2 = X_2 \cdot \bar{X}_3 \cdot \bar{X}_4 \cdot \bar{X}_5 \cdot \bar{X}_6 \cdot \bar{X}_7$$

$$Z_1 = X_1 \cdot \bar{X}_2 \cdot \bar{X}_3 \cdot \bar{X}_4 \cdot \bar{X}_5 \cdot \bar{X}_6 \cdot \bar{X}_7$$

$$Z_0 = X_0 \cdot \bar{X}_1 \cdot \bar{X}_2 \cdot \bar{X}_3 \cdot \bar{X}_4 \cdot \bar{X}_5 \cdot \bar{X}_6 \cdot \bar{X}_7.$$

Используя эти сигналы, составим соотношения для выходов  $Y_2$ ,  $Y_1$ ,  $Y_0$  аналогичные тем, какие были составлены для простого двоичного шифратора (3.1):

$$\begin{aligned} Y_0 &= Z_1 + Z_3 + Z_5 + Z_7, \\ Y_1 &= Z_2 + Z_3 + Z_6 + Z_7, \\ Y_2 &= Z_4 + Z_5 + Z_6 + Z_7. \end{aligned} \quad (3.2)$$

Условное обозначение 8-входового приоритетного шифратора показано на рис. 3.5. Вход  $X_7$  имеет высший приоритет. Если хотя бы на одном из входов присутствует сигнал, то на выходах  $Y_2$ ,  $Y_1$ ,  $Y_0$  появляется число, соответствующее входному сигналу с наивысшим приоритетом. Когда сигнал не подан ни на один из входов, появляется сигнал ожидания на выходе IDLE.

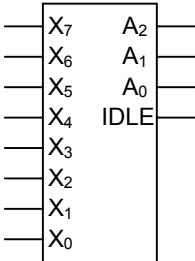


Рис. 3.5. Условное обозначение типичного 8-входового приоритетного шифратора

При наличии сигнала только на одном входе приоритетный шифратор работает так же, как и двоичный. Поэтому в сериях элементов двоичный шифратор как самостоятельный элемент может отсутствовать. Режим его работы является частным случаем работы приоритетного шифратора.

### 3.2. Мультиплексоры

Во многих случаях возникает необходимость объединения двоичных данных, поступающих от  $m$  различных источников, в поток сигналов, следующих один за другим по одной шине данных. Эта задача решается с помощью схемы, называемой *мультиплексором*. Мультиплексор осуществляет передачу на выход данных, поступающих от одного из  $m$  источников. На рис. 3.6а изображены входы и выходы мультиплексора с  $m$  входами по  $n$  разрядов каждый. Таким образом, имеются  $m$  источников  $n$ -разрядных данных и один  $n$ -разрядный выход. У типичных, выпускаемых серийно, мультиплексоров  $m = 1, 2, 4, 8$  или  $16$ , а  $n = 1, 2$  или  $4$ . Имеются  $k$  адресных входов (ADR), с помощью которых выбирается один из  $m$  источников, поэтому  $k = \lceil \log_2 m \rceil$ . При наличии на входе разрешения EN активного уровня на выходах мультиплексора появляются данные  $i$ -го источника, где  $i$  равно числу, поданному на входы выбора; когда EN = 0, сигналы на всех выходах равны 0. В условном обозначении схемы мультиплексора часто используют сокращение *MUX*.

В качестве иллюстрации принципа работы мультиплексора на рис. 3.6б приведена схема, являющаяся его механическим эквивалентом. В отличие от механического переключателя, мультиплексор является односторонним устройством – информация передается только от входов к выходам.

Для сигналов на выходе мультиплексора можно записать логическое выражение:

$$iY = \sum_{j=0}^{m-1} EN \cdot M_j \cdot iD_j.$$

Переменная  $iY$  – это  $i$ -й выходной бит ( $1 \leq i \leq n$ ), а переменная  $iD_j$  –  $i$ -й входной бит от  $j$ -го источника ( $0 \leq j \leq m - 1$ ).  $M_j = 1$ , если  $j$  есть двоичная запись  $k$ -разрядного числа на входе ADR. Таким образом, когда на вход мультиплексора подан сигнал разрешения, а число на входах выбора равно  $j$ , сигнал на каждом выходе  $iY$  принимает значение  $iD_j$  соответствующего бита выбранного входа.

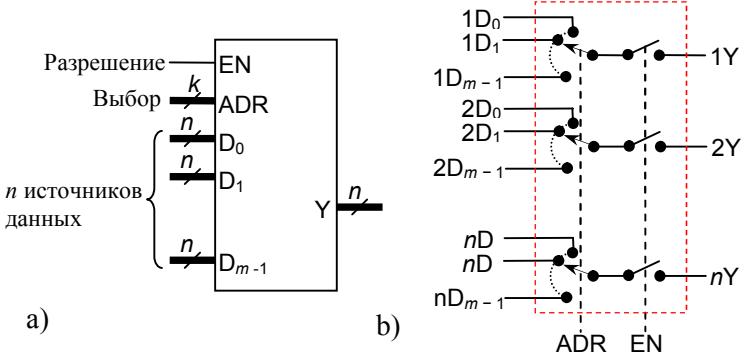


Рис. 3.6. Структура мультиплексора: а) входы и выходы;  
б) функциональный эквивалент

Мультиплексоры используются там, где двоичные данные от многих источников должны быть переданы адресату, например, в компьютерах для связи между регистрами процессора и его арифметико-логическим устройством (АЛУ). Примером может служить 16-разрядный процессор ( $n = 16$ ), у которого каждая команда имеет 3-разрядный адрес ( $k = 3$ ), определяющий один из восьми используемых регистров ( $m = 8$ ). Сигналы этого 3-разрядного адреса поступают на входы выбора 8-ходового 16-разрядного мультиплексора. Входы данных мультиплексора связаны с восемью регистрами, а данные с его выходов поступают в АЛУ для выполнения команды, использующей содержимое выбранного регистра.

### 3.2.1. Мультиплексоры в интегральном исполнении

Формально можно построить мультиплексор, имеющий любое число входов, однако серийно выпускаемые в виде ИС средней степени интеграции мультиплексоры обычно ограничены числом выводов у корпусов ИС. Например, показанная на рис. 3.7 схема мультиплексора  $74 \times 151$ , размещенная в корпусе с

16 выводами, осуществляет выбор сигнала, присутствующего на одном из восьми 1-разрядных входов.

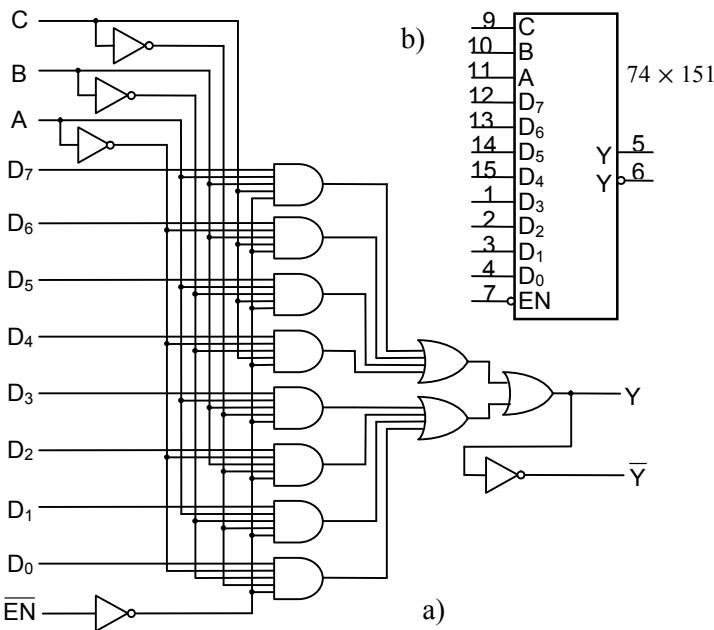


Рис. 3.7. 8-входовой 1-разрядный мультиплексор: а) принципиальная схема; б) традиционное условное обозначение с цоколевкой для стандартного корпуса DIP с 16 выводами

Входы адреса обозначены буквами С, В и А, где С является старшим разрядом. Сигнал на входе разрешения  $\bar{EN}$  имеет низкий активный уровень; у схемы есть выходы как с высоким активным уровнем сигнала Y, так и с низким активным уровнем сигнала  $\bar{Y}$ .

Таблица истинности мультиплексора приведена в табл. 3.2. Здесь использована расширенная система обозначений для таблиц истинности. В предыдущих таблицах истинности каждой входной комбинации соответствовал сигнал на выходе, равный 0 или 1. В данной таблице в графе «Входы» отсутствуют входы

$D_0 \dots D_7$ . Сигнал на каждом выходе задан как 0 или 1, либо как повторение или отрицание сигналов на входах  $D_i$  (например,  $D_0$  или  $\overline{D}_0$ ). При такой системе обозначений на восемь столбцов и восемь строк сокращаются размеры таблицы, а логическая функция представляется нагляднее, чем в случае, если бы таблица была полной.

Таблица 3.2

**Таблица истинности 8-входового 1-разрядного мультиплексора**

Входы				Выходы	
$\overline{EN}$	C	B	A	Y	$\bar{Y}$
1	x	x	x	0	1
0	0	0	0	$D_0$	$\overline{D}_0$
0	0	0	1	$D_1$	$\overline{D}_1$
0	0	1	0	$D_2$	$\overline{D}_2$
0	0	1	1	$D_3$	$\overline{D}_3$
0	1	0	0	$D_4$	$\overline{D}_4$
0	1	0	1	$D_5$	$\overline{D}_5$
0	1	1	0	$D_6$	$\overline{D}_6$
0	1	1	1	$D_7$	$\overline{D}_7$

### 3.3. Логические элементы ИСКЛЮЧАЮЩЕЕ ИЛИ

#### 3.3.1. Вентили ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ

Логический элемент *ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR)* представляет собой 2-входовую схему, у которой выходной сигнал равен 1 лишь в том случае, когда 1 присутствует только на одном из его входов. Возможно и другое определение: на выходе схемы ИСКЛЮЧАЮЩЕЕ ИЛИ появляется 1, если сигналы на его входах различны. Сигнал на выходе вентиля *ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ (XNOR)* имеет противоположное значение: он равен 1, если сигналы на входах вентиля одинаковы. Таблица истинности для этих

функций приведена в табл. 3.3. Операцию ИСКЛЮЧАЮЩЕЕ ИЛИ иногда обозначают символом  $\oplus$ , то есть

$$X \oplus Y = \bar{X} \cdot Y + X \cdot \bar{Y}.$$

Таблица 3.3

**Таблица истинности для функций ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ**

X	Y	$X \oplus Y$ (XOR)	$\overline{X \oplus Y}$ (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Логическая схема ИСКЛЮЧАЮЩЕЕ ИЛИ не является основной функцией двоичной алгебры, тем не менее дискретные схемы, реализующие функцию ИСКЛЮЧАЮЩЕЕ ИЛИ, широко применяются на практике. В используемых семействах логических схем нельзя непосредственно реализовать функцию ИСКЛЮЧАЮЩЕЕ ИЛИ, поэтому применяются схемы, состоящие из нескольких логических элементов, как показано на рис. 3.8.

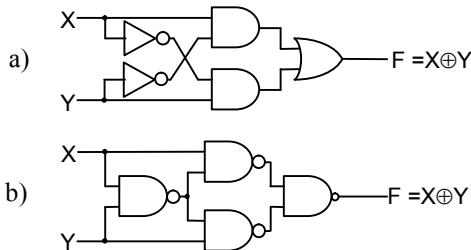


Рис. 3.8. Варианты реализации 2-ходовой схемы ИСКЛЮЧАЮЩЕЕ ИЛИ с помощью нескольких вентилей: а) на основе структур И-ИЛИ; б) трехуровневый вариант на основе вентилей И-НЕ

Условные обозначения схем, реализующих функции ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ, даны на рис. 3.9. Для каждой схемы возможны четыре варианта условных обозначений, вытекающих из простого правила: инвертирование любых

двах сигналов (обоих входных сигналов или одного входного и выходного) у схем ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ не изменяет результирующей логической функции.

При проектировании логических схем выбирается такое условное обозначение, которое является наиболее наглядным с точки зрения выполняемой логической функции.

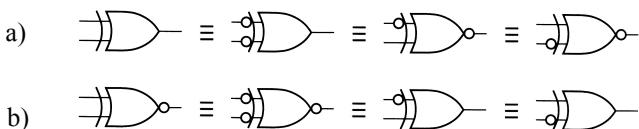


Рис. 3.9. Условные обозначения: а) схемы ИСКЛЮЧАЮЩЕЕ ИЛИ;  
б) схемы ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ

Схема ИСКЛЮЧАЮЩЕЕ ИЛИ имеет и другое название – *сумматор по модулю 2*. Эти логические элементы применяются для построения схем проверки на четность, полных сумматоров и в генераторах псевдослучайных последовательностей, которые рассмотрены ниже.

### 3.3.2. Схемы проверки на четность

При последовательном включении  $n$  схем ИСКЛЮЧАЮЩЕЕ ИЛИ, как показано на рис. 3.10а, получится схема ИСКЛЮЧАЮЩЕЕ ИЛИ с  $n + 1$  входами и одним выходом. Такая структура называется *схемой проверки на нечетность*, потому что сигнал на ее выходе равен 1, только в том случае, когда единицы присутствуют на нечетном числе входов. Недостатком этой схемы является большая задержка появления достоверного результата на выходе, в худшем случае равная  $n \cdot t_p$ , где  $t_p$  – задержка, вносимая одной схемой ИСКЛЮЧАЮЩЕЕ ИЛИ.

Схема, приведенная на рис. 3.10б, также осуществляет проверку на нечетность, но задержка появления достоверного результата у нее существенно меньше, поскольку в ней сумматоры по модулю 2 образуют древовидную структуру. При инвертировании выхода у представленных схем, получим *схему проверки на*

четность, у которой выходной сигнал равен 1, если единицы имеются на четном числе входов.

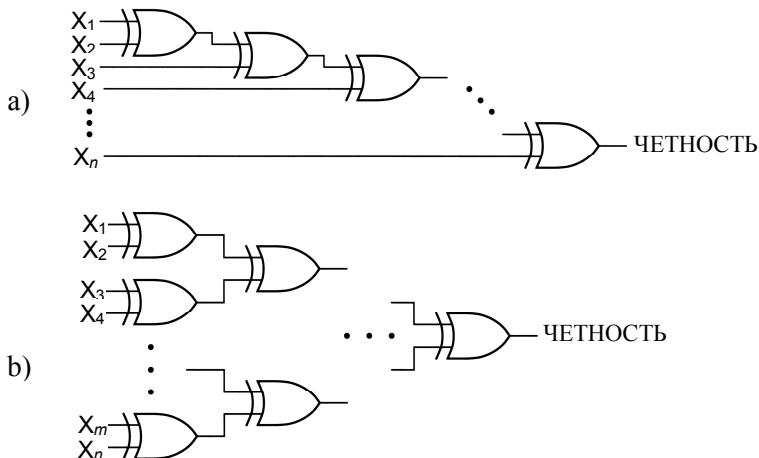


Рис. 3.10. Каскадное включение сумматоров по модулю 2:  
 а) последовательная структура; б) древовидная структура

### 3.4. Временные соотношения в схеме

Для возникновения на выходе реальной схемы реакции на входное воздействие требуется конечное время. Многие современные схемы могут иметь очень малое время реакции и системы, построенные на их основе, настолько быстры, что существенной становится даже задержка распространения сигнала от выхода одного логического элемента до входа другого в пределах платы или корпуса микросхемы.

Большинство цифровых систем представляет собой последовательное включение схем, работа которых синхронизируется периодическим сигналом, частота которого ограничена максимальным временем, которое требуется для завершения всех операций в системе в течение одного периода синхросигнала. Таким образом, чтобы быстродействующие схемы надежно работали

при всех условиях, необходимо четко представлять себе временные соотношения между сигналами.

### 3.4.1. Временные диаграммы

*Временная диаграмма* является наглядным представлением характера изменения сигналов в цифровой схеме в зависимости от времени. Справочные данные, сопровождающие любую относительно сложную цифровую систему, включают в себя временные диаграммы функционирования этой системы. Они используются как для объяснения временных соотношений между сигналами в пределах данной системы, так и для того, чтобы определить временные требования к внешним сигналам, которые поступают в систему.

В качестве примера на рис. 3.11а изображена блок-схема некоторого комбинационного устройства с двумя входами и двумя выходами. На рис. 3.11б показана задержка выходных сигналов относительно момента появления входного сигнала  $X$  в предположении, что входной сигнал  $EN$  к этому моменту времени имеет активный уровень. Переходы сигналов изображены в виде наклонных линий, напоминая о том, что в реальных схемах изменения сигналов не происходят мгновенно.

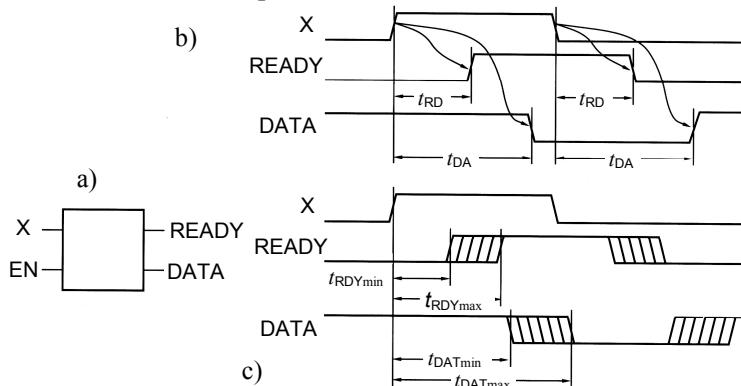


Рис. 3.11. Пример временных диаграмм для комбинационной схемы:  
а) блок-схема устройства; б) взаимосвязь между сигналами и задержки  
распространения; в) минимальные и максимальные задержки

Иногда, особенно в случае сложных временных диаграмм, стрелками отражают связи, показывающие изменение какого входного сигнала вызывает изменение того или иного выходного сигнала. Наиболее важной информацией, получаемой из временных диаграмм, является величина задержки между переходами в сигналах.

Вследствие различных путей прохождения сигнала в схеме, задержки могут иметь различные значения. На рис. 3.11б, например, показано, что задержка между сигналами X и READY меньше, чем задержка между сигналами X и DATA. Различной может быть задержка между входным сигналом EN и выходными сигналами, и это можно было бы наблюдать на другой временной диаграмме. Кроме того, задержка при прохождении сигнала по некоторому пути может различаться в зависимости от того, в каком направлении изменяется сигнал на выходе: от низкого уровня до высокого или от высокого уровня до низкого.

В реальной схеме задержка обычно измеряется между средними точками переходов в сигналах, как показано на временных диаграммах. Одна единственная временная диаграмма может содержать много различной информации о задержках. Каждая задержка имеет свое обозначение; на нашем рисунке это  $t_{RDY}$  и  $t_{DAT}$ . В сложных временных диаграммах задержки обычно нумеруются для облегчения ссылки на них (например:  $t_1$ ,  $t_2$ , ...,  $t_{32}$ ). Во всех случаях временная диаграмма сопровождается таблицей временных параметров, в которой указывается величина каждой задержки и условия, при которых получается эта задержка.

Величина задержки в реальных цифровых компонентах, как правило, сильно зависит от напряжения питания, температуры и различных производственных факторов. Поэтому в таблице временных параметров указывается диапазон значений, определяющий минимальное, типичное и максимальное значение для каждой задержки. Наличие некоторого диапазона возможных значений задержки иногда отражается на временной диаграмме,

как показано на рис. 3.11с, где переходы происходят не в строго фиксированные моменты времени, а в некотором диапазоне от  $t_{\min}$  до  $t_{\max}$ .

Далеко не всегда на временной диаграмме требуется изображать, как именно меняется значение сигнала в конкретный момент времени: от 1 до 0 или от 0 до 1; достаточно бывает показать, что происходит переход. Примером может служить сигнал, несущий информацию о бите данных: значение бита изменяется в зависимости от внешних обстоятельств, но, независимо от его значения, бит передается, сохраняется или обрабатывается в определенный момент времени, определяемый управляющим сигналом в системе. Временная диаграмма на рис. 3.12а поясняет эту идею. Предположим, что входной сигнал данных DATAIN имеет установленное значение, равное 0 или 1, а его переходы происходят только в определенные моменты времени. Понятие о не строго фиксированной величине задержки применимо и к выходным сигналам данных DATAOUT, как показано на рисунке.

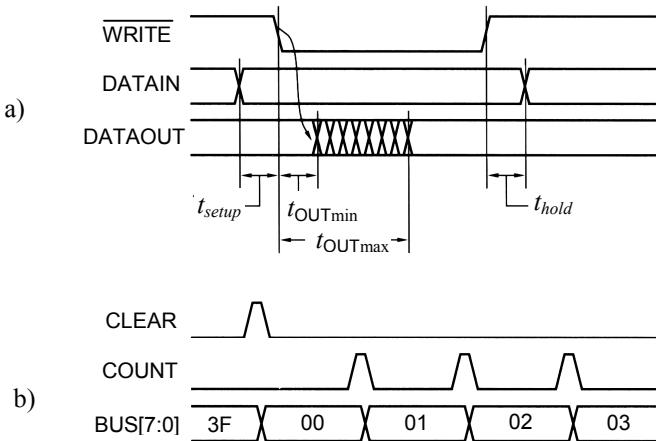


Рис. 3.12. Временные диаграммы сигналов данные: а) фиксированные и не строго фиксированные моменты переходов; б) последовательность значений сигналов на 8-разряднойшине

Довольно часто в цифровых системах обработка группы сигналов данных в шине производится идентичными схемами.

В этом случае все сигналы в шине имеют одни и те же временные параметры и могут быть представлены одной временной диаграммой и соответствующими записями в таблице временных параметров. Если известно, что сигналы в шине в течение определенного времени имеют постоянные значения, то иногда это отображают на временной диаграмме двоичными, восьмеричными или шестнадцатеричными числами, как показано на рис. 3.12б.

### 3.4.2. Задержка распространения

Ранее задержка распространения сигнала была формально определена как время, необходимое для того, чтобы изменение сигнала во входной цепи привело к изменению сигнала на выходе. В комбинационной схеме, имеющей несколько входов и выходов имеются различные пути прохождения сигнала, и задержки распространения по каждому пути могут иметь свое значение. Более того, задержка распространения при изменении выходного сигнала от низкого уровня до высокого ( $t_{pLH}$ ) может отличаться от задержки, когда этот сигнал изменяется от высокого уровня до низкого ( $t_{pHL}$ ).

В справочных данных комбинационных ИС обычно указывают все эти задержки распространения или, по крайней мере, задержки, которые могут представлять интерес в типичных приложениях. При объединении различных ИС в большой системе, для анализа временных соотношений в схеме используются индивидуальные характеристики отдельных микросхем. Задержка распространения сигнала при прохождении через все логические элементы схемы равна сумме задержек, вносимых отдельными элементами.

### 3.4.3. Временные параметры

Временные параметры устройства характеризуются минимальным, типичным и максимальным значениями для каждой задержки распространения и для каждого направления изменения сигнала:

*Максимальная задержка.* Этот параметр является одним из наиболее часто используемых, так как задержка распространения

ни при каких условиях не может превышать максимального значения.

*Типичная задержка.* Это один из параметров, чаще всего используемых разработчиками, которые не задумываются о том, как будет вести себя устройство в реальных условиях, когда возможны изменения окружающей температуры, напряжения питания и т. д. Типичная задержка – это величина, которая характеризует устройство, работающее при почти идеальных условиях.

*Минимальная задержка.* Это наименьшее значение задержки распространения, которое может быть получено при некоторых благоприятных условиях. Работа большинства правильно построенных схем не зависит от этого параметра, то есть они будут нормально работать, даже если задержка нулевая.

У схем малой степени интеграции задержка распространения от любого входа до выхода одна и та же. В случае ТТЛ-схем задержки, как правило, различны при изменении сигнала от низкого уровня до высокого и от высокого уровня до низкого ( $t_{pLH}$  и  $t_{pHL}$ ), а у КМОП-схем эти различия обычно незначительные. КМОП-схемы имеют более симметричные выходные характеристики, поэтому можно не обращать внимание на небольшое различие между этими двумя случаями.

Если минимальная задержка ИС не указана, то целесообразно принять ее равной нулю. Некоторые схемы не будут работать, если задержка распространения фактически стремится к нулю, но затраты на изменения в схеме, обеспечивающие ее работоспособность при нулевой задержке, могут оказаться непомерно высокими, тем более что такой случай, как предполагается, никогда не произойдет. Чтобы получить устройство, которое при разумных условиях всегда будет работать, разработчики часто принимают величину минимальной задержки ИС равной четверти или трети заявленного типичного значения задержки.

### 3.4.4. Временной анализ

Точный анализ временных соотношений в устройстве, содержащем большое число интегральных схем, приходится проводить с учетом его поведения до мельчайших подробностей. Ко-

гда, например, инвертирующие схемы (И-НЕ, ИЛИ-НЕ и т.д.) включаются последовательно, переход сигнала с низкого уровня на высокий на выходе одного из вентилей вызовет изменение сигнала от высокого уровня до низкого на выходе следующего вентиля, так что средняя задержка оказывается между величинами  $t_{pLH}$  и  $t_{pHL}$ . С другой стороны, при последовательном включении неинвертирующих вентилей (И, ИЛИ и т.д.) переключение вызывает изменение сигналов на всех выходах в одном и том же направлении, так что различие между значениями  $t_{pLH}$  и  $t_{pHL}$  увеличивается.

Анализ усложняется, если задержка определяется устройствами средней степени интеграции, или в том случае, когда от данного входа до выхода имеется много путей распространения сигнала. Таким образом, в больших схемах анализ задержки прохождения сигнала по всем возможным путям при различных изменениях сигналов может быть очень сложным.

Для упрощения анализа в наихудшем случае часто используют единственный параметр – задержку для наихудшем случае, которая равна наибольшему из значений  $t_{pLH}$  и  $t_{pHL}$ . В этом случае задержка вычисляется как сумма задержек, вносимых отдельными компонентами в наихудшем случае, независимо от направления переключения и других условий работы схемы. Это может дать чрезмерно завышенную оценку полной задержки, вносимой схемой, но сокращает время разработки и гарантирует работоспособность изделия.

### 3.4.5. Гонки в комбинационных устройствах

Если комбинационная схема выполнена на базе идеальных, т.е. безынерционных элементов, состояние выходов однозначно определяется состоянием входов в тот же момент времени. Однако инерционность элементов и наличие различных факторов, приводящих к задержке распространения сигнала, приводят к тому, что сигналы на выходе комбинационной схемы, соответствующие новому состоянию входных сигналов, появляются не сразу, а с некоторой задержкой. При этом в переходный период возможно появление на выходах устройства некоторых промежу-

точных значений сигналов, не соответствующих заданному состоянию устройства. Такое явление получило название *состязаний* или *гонок*. Обычно вырабатываемые комбинационной схемой промежуточные значения сигналов представляют собой импульсы очень малой длительности, являющиеся помехой для всей цифровой системы. Они могут вызывать непредусмотренное срабатывание триггеров, счетчиков и осуществлять нежелательные записи в регистры.

Рассмотрим в качестве примера фрагмент комбинационной схемы (рис. 3.13), где может наблюдаться явление гонок. Для наглядности логические элементы ЛЭ представлены в виде инверторов, хотя рассматриваемые процессы существуют в любых логических схемах.

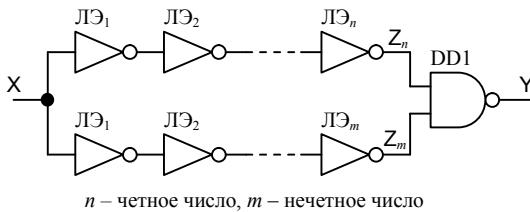


Рис. 3.13. Фрагмент комбинационной схемы, иллюстрирующий возможность появления гонок

На рис. 3.14 приведены временные диаграммы состояний различных цепей в идеальном и реальном случаях. Время задержки импульсов в цепях определяется средним временем задержки распространения сигнала всеми элементами этой цепи. Момент появления импульса помехи определяется соотношением числа инвертирующих элементов в конкурирующих цепях фрагмента комбинационной схемы.

Как следует из рис. 3.14а, в том случае, когда элементы схемы идеальные, т. е. безынерционные (что на практике не достижимо), на выходе комбинационной схемы сигнал помехи отсутствует. Однако в реальных схемах всегда имеет место явление гонок и необходимо создавать такие схемы, в которых влияние этого явления устранено.

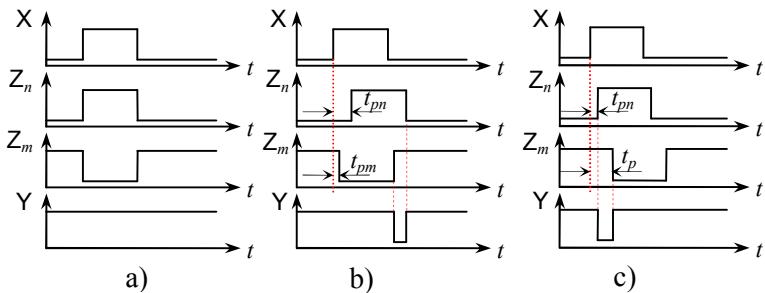


Рис. 3.14. Временные диаграммы, поясняющие процесс образования гонок: а) идеальный случай; б, с) логические элементы имеют задержку

Одним из наиболее эффективных методов борьбы с гонками является применение синхронных устройств. *Синхронным устройством* называют такую систему, в которой все сигналы изменяют свои значения только в моменты времени, определяемые специальными сигналами CLK тактового генератора. Интервалы времени между тактовыми сигналами являются постоянными и выбираются так, чтобы за это время закончились переходные процессы, связанные с изменением всех сигналов. Выходные сигналы считаются только во время активного уровня тактовых сигналов, когда под воздействием входных и промежуточных сигналов автомат уже перешел в новое состояние.

Пример реализации синхронного комбинационного устройства приведен на рис. 3.15.

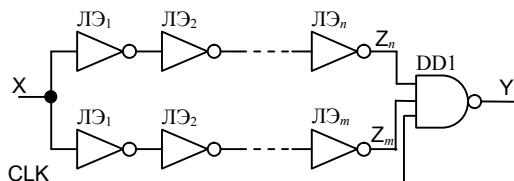


Рис. 3.15. Фрагмент синхронной комбинационной схемы с тактированием выходного сигнала сигналом CLK

На рис. 3.16 приведены временные диаграммы состояний различных цепей синхронной комбинационной схемы. Заштрихованными областями показаны интервалы времени, в течение которых выходные сигналы достоверны.

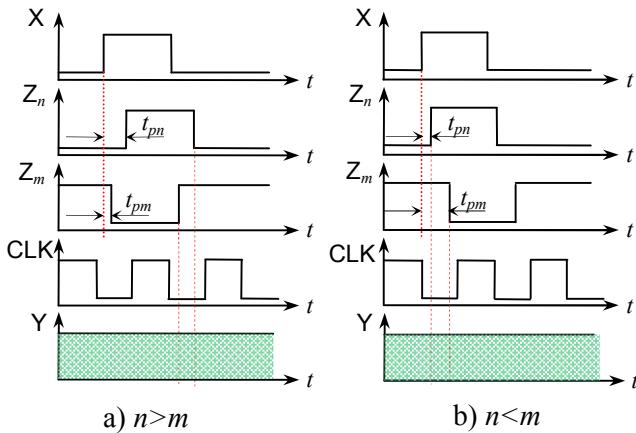


Рис. 3.16. Временные диаграммы, поясняющие работу синхронной комбинационной схемы

## 4.1. Компараторы

*Компараторами* называют устройства, определяющие отношения между двумя словами. Основными соотношениями, через которые можно выразить остальные, можно считать два – *равно* и *больше*. Сравнение двух двоичных слов с целью обнаружения их совпадения является широко используемой операцией в компьютерных системах и устройствах сопряжения. Некоторые компараторы интерпретируют входные слова как числа со знаком или без знака и выдают арифметическое соотношение между числами (больше или меньше). Эти устройства часто называются *компараторами значений*.

Схемы ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ можно считать 1-разрядными компараторами. На рис. 4.1а вентиль ИСКЛЮЧАЮЩЕЕ ИЛИ представлен как 1-разрядный компаратор.

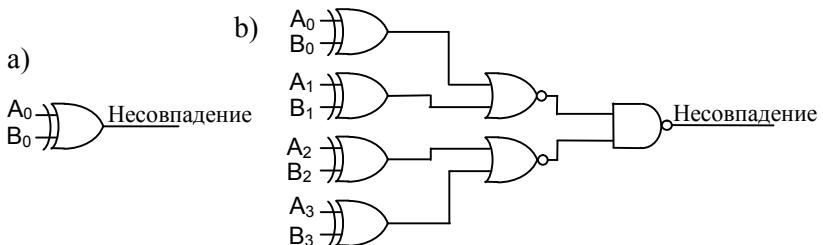


Рис. 4.1. Компараторы на основе схем ИСКЛЮЧАЮЩЕЕ ИЛИ:  
а) 1-разрядный компаратор; б) 4-разрядный компаратор

На выходе *Несовпадение* появляется сигнал высокого уровня в том случае, если сигналы на входах различны. Объединяя схемами ИЛИ-НЕ и И-НЕ выходы четырех схем ИСКЛЮЧАЮЩЕЕ ИЛИ, можно получить 4-разрядный компаратор, как показано на рис. 4.1б. Сигнал на выходе *Несовпадение* появляется в том случае, когда хотя бы в одном разряде значения входных сигналов различны.

Компараторы выпускаются в виде интегральных микросхем. На рис. 4.2 дано условное обозначение 4-разрядной ИС компаратора, а его принципиальная схема показана на рис. 4.3.

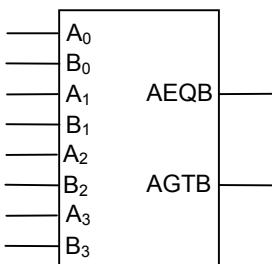


Рис. 4.2. Условное обозначение 4-разрядного компаратора

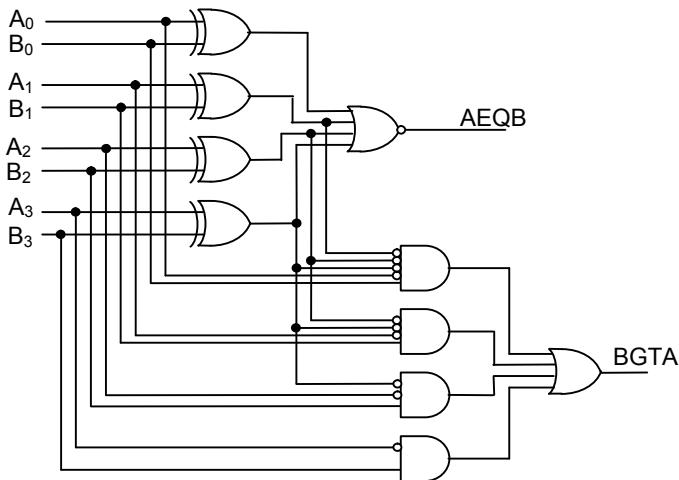


Рис. 4.3. Принципиальная схема 4-разрядного компаратора

В верхней части схемы проверяется равенство двух 4-разрядных входных слов. Сигнал появляется на выходе каждой схемы ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ, если не совпадают сигналы на его входах, а уровень сигнала на выходе AEQB становится активным, если попарно равны сигналы во всех четырех разрядах чи-

сел, поданных на входы. В нижней части схемы входные слова сравниваются арифметически, и вырабатывается сигнал на выходе АГТВ , если  $A[3-0] > B[3-0]$ .

## 4.2. Сумматоры

*Сумматором* называют комбинационную схему, выполняющую арифметическое сложение и вычитание чисел. Сумматоры составляют ядро схем арифметико-логических устройств (АЛУ), являющихся непременной частью всех процессоров. Одни и те же правила сложения справедливы для чисел без знака и для чисел, представленных в дополнительном двоичном коде; поэтому в обоих случаях используются одни и те же сумматоры. Сумматор может выполнять вычитание путем сложения уменьшаемого и дополнения к вычитаемому, то есть инвертированного вычитаемого.

### 4.2.1. Полусумматоры и полные сумматоры

*Полусумматором* называется, комбинационная схема выполняющая операцию сложения двух 1-разрядных операндов  $X$  и  $Y$ , образуя при этом 2-разрядную сумму. Сумма может принимать значения от 0 до 2 и для своего представления требует два бита. Младший бит суммы называют полусуммой  $HS$ , а старший бит – переносом  $CO$  (в старший разряд). Для величин  $HS$  и  $CO$  можно записать следующие соотношения:

$$HS = X \oplus Y = X\bar{Y} + \bar{X}Y,$$
$$CO = XY.$$

Чтобы сложить операнды с большим числом двоичных разрядов, необходимо учитывать перенос между разрядами. Схема, применяемая для выполнения этой операции, называется *полным сумматором*. Помимо входов для битов слагаемых  $X$  и  $Y$ , у полного сумматора есть вход для бита переноса  $CIN$ . Сумма трех входных битов может принимать значения от 0 до 3; для ее представления по-прежнему достаточно двух выходных битов  $S$  и  $COUT$ , значения которых определяются следующими соотношениями:

$$S = X \oplus Y \oplus C_{IN} = X \cdot \bar{Y} \cdot \bar{C_{IN}} + \bar{X} \cdot Y \cdot \bar{C_{IN}} + \bar{X} \cdot \bar{Y} \cdot C_{IN} + X \cdot Y \cdot C_{IN},$$

$$C_{OUT} = X \cdot Y + X \cdot C_{IN} + Y \cdot C_{IN}.$$

Здесь  $S = 1$ , если на нечетном числе входов присутствуют единицы, а  $C_{OUT} = 1$ , если единицы имеются на двух или большем числе входов. Эти соотношения определяются таблицей двоичного сложения.

Один из вариантов схем, реализующих соотношения, описывающие полный сумматор, приведен на рис. 4.4а. Условное обозначение полного сумматора дано на рис. 4.4б. Иногда для более компактного изображения схем с последовательно включенными полными сумматорами их обозначают так, как показано на рис. 4.4с.

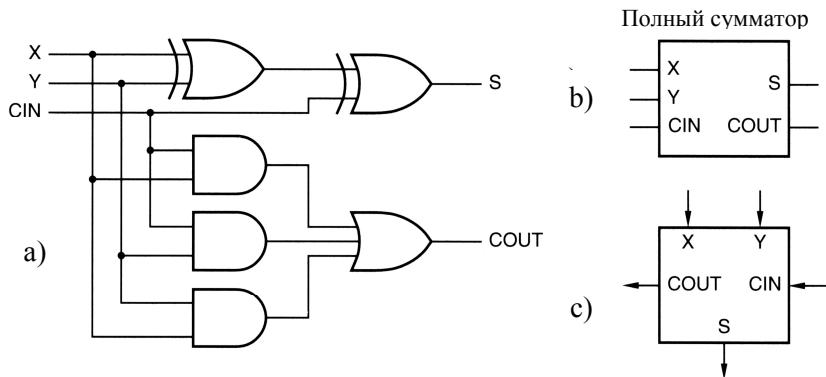


Рис. 4.4. Полный сумматор: а) принципиальная схема на уровне вентилей; б) условное обозначение; в) другое условное обозначение, удобное для изображения последовательного включения

#### 4.2.2. Сумматоры со сквозным (последовательным) переносом

Простейшим сумматором двух  $n$ -разрядных двоичных чисел может служить *сумматор со сквозным (последовательным) переносом*, состоящий из  $n$  последовательно включенных полных

сумматоров, каждый из которых оперирует с одной парой битов. Схема 4-разрядного сумматора со сквозным переносом показана на рис. 4.5. На входе переноса младшего разряда ( $c_0$ ) обычно устанавливается 0, а выход переноса каждого из полных сумматоров соединен с входом переноса полного сумматора в следующем разряде.

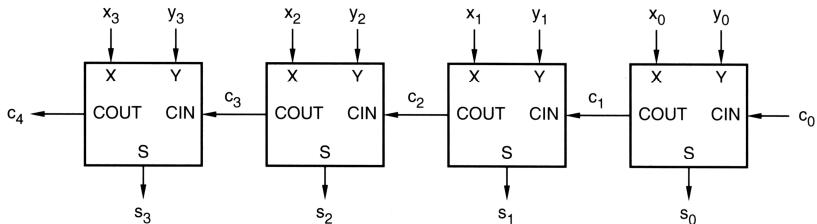


Рис. 4.5. 4-разрядный сумматор со сквозным переносом

Сумматор со сквозным переносом обладает малым быстродействием, поскольку в худшем случае сигнал переноса должен распространяться от младшего полного сумматора до старшего. Такая ситуация наблюдается, например, если одно из слагаемых равно 11 ... 11, а другое – 00 ... 01. Если все биты слагаемых поступают на входы одновременно, то полная задержка в худшем случае равна

$$t_{ADD} = t_{XYCOUT} + (n - 2) \cdot t_{CINCOUT} + t_{CINS},$$

где  $t_{XYCOUT}$  – задержка от входов X или Y до выхода COUT в сумматоре младшего разряда,  $t_{CINCOUT}$  – задержка от входа CIN до выхода COUT в сумматорах средних разрядов, а  $t_{CINS}$  – задержка от входа CIN до выхода S в сумматоре старшего разряда.

#### 4.2.3. Сумматоры с ускоренным (параллельным) переносом

Сумматоры с ускоренным переносом не используют последовательное распространение переноса от младших разрядов к старшим. Сигналы переноса для каждого разряда формируются

специальными схемами, на входы которых поступают все переменные, необходимые для выработки сигнала переноса.

Логическое соотношение для суммы  $i$ -го разряда двоичного сумматора можно записать очень просто:  $s_i = x_i \oplus y_i \oplus c_i$ .

Сложнее представить  $c_i$  через  $x_0 \dots x_{i-1}$ ,  $y_0 \dots y_{i-1}$  и  $c_0$ ; реальные неприятности возникают в связи с ростом числа схем ИСКЛЮЧАЮЩЕЕ ИЛИ. Однако если мы хотим предотвратить увеличение числа этих схем, то можно, по крайней мере, упростить логику формирования  $c_i$ , используя идею *ускоренного переноса*.

На рис. 4.6 продемонстрирована основная идея.

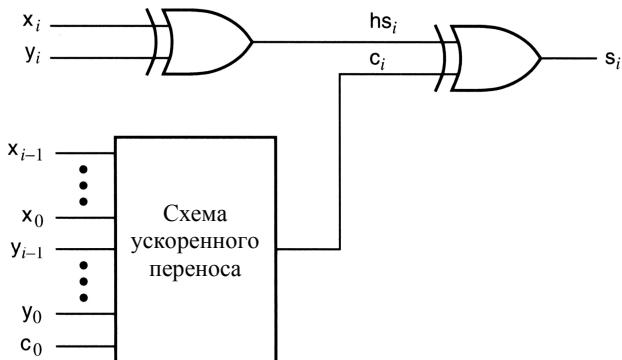


Рис. 4.6. Структура одного каскада сумматора с ускоренным переносом

В блоке, названном *схема ускоренного переноса*, значение  $c_i$  вычисляется по правилам, предусматривающим небольшое, фиксированное число логических уровней (последовательно включенных логических вентилей) при любом разумном значении  $i$ . Для схемы ускоренного переноса основными являются следующие два определения:

При заданной комбинации сигналов на входах  $x_i$  и  $y_i$  в  $i$ -м разряде сумматора генерируется сигнал переноса  $g_i$ , если в этом разряде вырабатывается 1 на выходе переноса ( $c_{i+1} = 1$ ) независимо от значений входных сигналов  $x_0 \dots x_{i-1}$ ,  $y_0 \dots y_{i-1}$  и  $c_0$ .

При заданной комбинации сигналов на входах  $x_i$  и  $y_i$  в  $i$ -м разряде сумматора происходит *передача сигнала переноса*  $p_i$ , если в этом разряде вырабатывается 1 на выходе переноса ( $c_{i+1} = 1$ ) в присутствии такой комбинации на входах  $x_0 \dots x_{i-1}$ ,  $y_0 \dots y_{i-1}$  и  $c_0$ , которая вызывает появление 1 на выходе переноса данного каскада ( $c_i = 1$ ).

В соответствии с этими определениями логические равенства для сигнала генерации переноса  $g_i$  и сигнала передачи переноса  $p_i$  в каждом разряде сумматора с ускоренным переносом можно записать в следующем виде:

$$\begin{aligned} g_i &= x_i \cdot y_i, \\ p_i &= x_i + y_i. \end{aligned}$$

Записанные соотношения говорят о том, что на выходе  $i$ -го разряда всегда генерируется сигнал переноса, если оба бита слагаемых равны 1, и передается перенос предыдущего разряда, если хотя бы один из битов слагаемых равен 1. Теперь сигнал на выходе переноса можно выразить через сигналы генерации и передачи переноса:

$$c_{i+1} = g_i + p_i c_i.$$

Чтобы исключить сквозной перенос, мы для каждого разряда рекурсивно находим значения  $c_i$  и, разнося множители по слагаемым, получаем выражения в виде двухуровневых функций И-ИЛИ. Используя этот метод, можно найти следующие выражения для сигналов переноса первых четырех каскадов сумматора:

$$\begin{aligned} c_1 &= g_0 + p_0 c_0; \\ c_2 &= g_1 + p_1 c_1 = \\ &= g_1 + p_1 (g_0 + p_0 c_0) = \\ &= g_1 + p_1 g_0 + p_1 p_0 c_0; \\ c_3 &= g_2 + p_2 c_2 = \\ &= g_2 + p_2 (g_1 + p_1 g_0 + p_1 p_0 c_0) = \\ &= g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0; \\ c_4 &= g_3 + p_3 c_3 = \\ &= g_3 + p_3 (g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0) = \\ &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0. \end{aligned}$$

Из полученных выражений следует, что задержка формирования сигнала переноса в любом разряде определяется суммой задержек трех логических элементов: первый элемент, вносящий задержку, связан с образованием сигналов генерации  $g_i$  и передачи переноса  $p_i$ , а два других – с образованием суммы произведений. В блоках ускоренного переноса каждого разряда (рис. 4.6) сумматора с ускоренным переносом используются трехуровневые выражения типа приведенных выше. Выходной сигнал суммы в каждом разряде формируется путем комбинации бита переноса с битами двух слагаемых данного разряда, как показано на рисунке.

Несмотря на то, что полученные выражения достаточно сложные, время формирования сигнала переноса в любой разряд с помощью вспомогательных функций определяется только временем задержки распространения сигнала на двух элементах. Эти функции реализуются специальным комбинационным устройством – схемой ускоренного переноса.

Схема 4-разрядного сумматора с ускоренным переносом приведена на рис. 4.7.

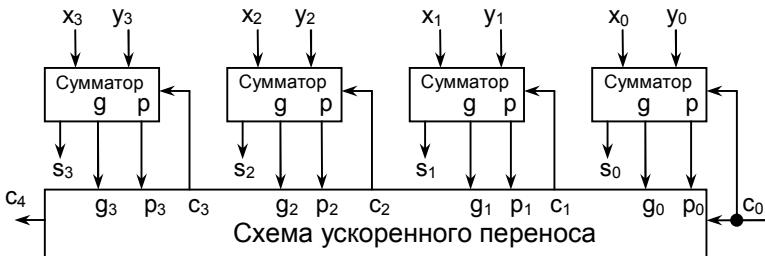


Рис. 4.7. Схема 4-х разрядного сумматора с ускоренным переносом

На рис. 4.8 изображена схема устройства ускоренного переноса для 4-разрядного сумматора. Видно, что сложность схемы растет с ростом числа разрядов, однако задержка формирования сигналов переноса остается неизменной.

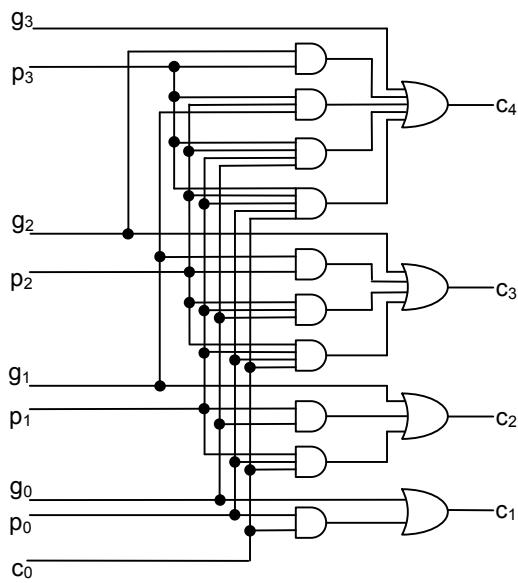


Рис. 4.8. Схема ускоренного переноса для 4-разрядного сумматора

## ГЛАВА 5. ПОСЛЕДОВАТЕЛЬНОСТНЫЕ ЛОГИЧЕСКИЕ СХЕМЫ

Схема, выходные сигналы которой зависят не только от текущих значений входных сигналов, но и от последовательности значений входных сигналов в прошлом, называется *последовательностной схемой* или *схемой с памятью*.

Трудно, а иногда и невозможно, описать поведение последовательностной схемы таблицей истинности, в которой выходные сигналы перечисляются как функции последовательности входных сигналов, принятой к настоящему моменту времени. Чтобы знать, в каком состоянии окажется схема в следующий момент времени, нужно знать, в каком состоянии она находится в настоящий момент. Поведение такой схемы можно описать *таблицей состояний*, которая определяет сигнал на ее выходе и следующее ее состояние в зависимости от текущего состояния и значений сигналов на входах. Эта зависимость может простираться далеко назад.

*Состоянием* последовательностной схемы называется совокупность *переменных состояния*, чьи значения в любой фиксированный момент времени содержат исчерпывающую информацию о прошлом, необходимую для того, чтобы описать поведение схемы в будущем.

У цифровой схемы переменные состояния принимают два значения, соответствующие логическим сигналам в этой схеме. Схема с  $n$  двоичными переменными состояния может находиться в одном из  $2^n$  состояний. Как бы велико ни было число  $2^n$ , оно всегда конечно, поэтому последовательностные схемы называют *конечными автоматами*.

Как правило, в последовательностных схемах изменение состояния происходит в моменты времени, задаваемые *тактовым сигналом (clock)*. На рис. 5.1 приведены временные диаграммы и используемые обозначения для тактовых сигналов. Обычно считается, что активным у тактового сигнала является высокий уровень, если изменение состояния происходит в момент, задаваемый нарастающим фронтом тактового сигнала, или тогда, ко-

гда тактовый сигнал имеет высокий уровень HIGH; в противоположном случае говорят, что активным уровнем тактового сигнала является низкий уровень. Интервал времени между соседними переходами, совершаемыми сигналом в одном и том же направлении называют *периодом тактового сигнала*, а *частотой тактового сигнала* – величину, обратную периоду. Переход или импульс в пределах периода, а иногда и сам период называются *тактом*. Источником тактового сигнала в различных цифровых системах часто служит кварцевый генератор.

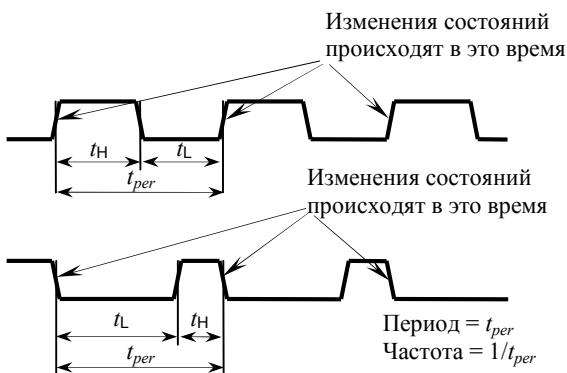


Рис. 5.1. Тактовые сигналы: а) с высоким активным уровнем;  
б) с низким активным уровнем

Большинство практических цифровых устройств содержат последовательностные схемы двух типов, которые будут рассмотрены ниже. *Последовательностная схема* состоит из логических вентилей, охваченных обратной связью. Благодаря наличию обратных связей последовательностная схема обладает памятью. Такие схемы являются стандартными узлами цифровых устройств и называются защелками и триггерами. Эти узлы, в частности, переключающиеся по фронту D-триггеры, являются основными элементами, использующимися в *тактируемых синхронных конечных автоматах* для создания устройств, в которых происходит опрос входных сигналов и выходные сигналы изменя-

няются в моменты времени, задаваемые управляющим тактовым сигналом.

## 5.1. Элемент с двумя устойчивыми состояниями

Два инвертора, охваченные петлей обратной связи, как показано на рис. 5.2, образуют простейшую последовательностную схему. У этой схемы отсутствуют входы и есть два выхода:  $Q$  и  $\bar{Q}$ .

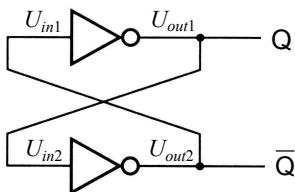


Рис. 5.2. Элемент с двумя устойчивыми состояниями

### 5.1.1. Цифровой подход

Схему, приведенную на рис. 5.2, называют *бистабильной*, так как у нее есть два устойчивых состояния, в чем можно убедиться анализируя возможные состояния на выходах. Если на выходе  $Q$  имеет место высокий уровень, то, будучи подан на вход нижнего инвертора, он обеспечивает наличие низкого уровня на выходе этого инвертора, который в свою очередь удерживает сигнал на выходе верхнего инвертора на высоком уровне. Но если  $Q$  имеет низкий уровень, то нижний инвертор с низким уровнем на входе вырабатывает сигнал высокого уровня на своем выходе, который вынуждает верхний инвертор иметь низкий уровень на его выходе  $Q$ . Состояние такой схемы можно описать, воспользовавшись единственной переменной состояния, а именно – значением сигнала на выходе  $Q$ ; возможны только два состояния:  $Q = 0$  и  $Q = 1$ . Из этого можно сделать вывод, что рассмотренный элемент является ячейкой памяти, позволяющей хранить одно из двух значений переменной.

Однако изменять существующее устойчивое состояние этого элемента нет возможности, поскольку у него нет входов. Когда на схему подается напряжение питания, в ней случайно устанавливается то или другое состояние, в котором схема остается до тех пор, пока не снято напряжение питания.

### 5.1.2. Аналоговый подход

Анализ работы бистабильного элемента с аналоговой точки зрения позволяет увидеть более тонкие детали. Линией 1 на рис. 5.3 изображена статическая (по постоянному току) передаточная характеристика для одного инвертора; выходное напряжение является функцией входного напряжения:  $U_{out} = F(U_{in})$ . Для двух инверторов, охваченных петлей обратной связи, как показано на рис. 5.2, имеем:  $U_{in1} = U_{out2}$  и  $U_{in2} = U_{out1}$ ; поэтому на одном и том же графике можно начертить передаточные характеристики для обоих инверторов, откладывая по осям координат соответствующие величины. Таким образом, кривая 1 представляет собой передаточную характеристику верхнего на рис. 5.3 инвертора, а кривая 2 – передаточную характеристику нижнего инвертора.

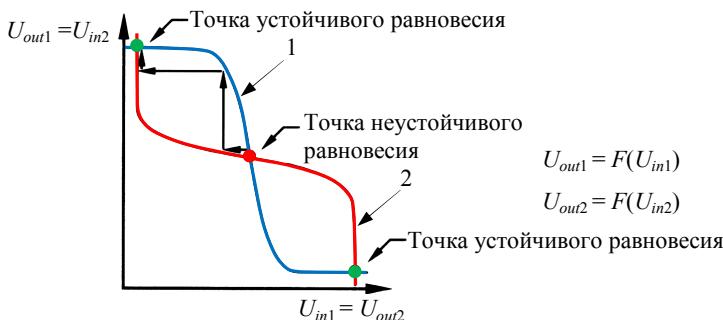


Рис. 5.3. Передаточные характеристики инверторов в петле обратной связи с двумя устойчивыми состояниями

Рассматривая статическое поведение схемы с петлей обратной связи, видим, что равновесие наступает тогда, когда на-

пряжения на входах и выходах обоих инверторов имеют постоянные значения, согласующиеся с требованиями, определяемыми конфигурацией схемы, и с передаточными характеристиками инверторов по постоянному току. То есть должны выполняться равенства

$$\begin{aligned} U_{in1} &= U_{out2} = T(U_{in2}) = T(U_{out1}) = T(T(U_{in1})), \\ U_{in2} &= T(T(U_{in2})). \end{aligned}$$

На рис. 5.3 видно, что имеется не две, а три точки равновесия – это точки, в которых пересекаются кривые, изображающие передаточные характеристики. Две из них, отмеченные как точки *устойчивого равновесия*, соответствуют тем двум состояниям, к которым мы пришли при цифровом подходе: сигнал на выходе Q имеет значение 0 (низкий уровень) или 1 (высокий уровень).

Третья точка, отмеченная как *неустойчивое равновесие* (*метастабильное состояние*), соответствует случаю, когда напряжения  $U_{out1}$  и  $U_{out2}$  находятся примерно посередине между значениями напряжений, соответствующих логической 1 и логическому 0; в этой точке ни Q, ни  $\bar{Q}$  не являются логическими сигналами в том смысле, в каком они были определены ранее. Однако уравнения, описывающие поведение схемы, удовлетворяются и в этой точке; если бы нам удалось поставить схему в состояние, соответствующее данной точке, то теоретически схема могла бы оставаться в этом состоянии неограниченно долго.

## 5.2. Неустойчивое равновесие

В точке неустойчивого равновесия любая флуктуация выводит схему из этой точки в одно из устойчивых состояний.

Предположим, что рассматриваемая схема находится в точке неустойчивого равновесия, показанной на рис. 5.3. Предположим теперь, что из-за небольшого шума в цепи напряжение  $U_{in1}$  немного уменьшилось. Это небольшое изменение вызывает некоторое *увеличение* напряжения  $U_{out1}$ . Поскольку выходное напряжение верхнего инвертора  $U_{out1}$  является входным напряжением нижнего инвертора  $U_{in2}$ , мы можем перейти по первой горизонтальной стрелке из окрестности точки неустойчивого равновесия к другой точке на второй передаточной характеристике, согласно

которой напряжение  $U_{out2}$  должно иметь теперь меньшее значение и, значит, меньшим должно стать равное ему напряжение  $U_{in1}$ . Вернемся к началу рассуждений, за исключением того, что изменение напряжения  $U_{in1}$  теперь много больше, чем то, которое было вызвано шумом, и рабочая точка продолжает смещаться. Этот *регенеративный* процесс происходит до тех пор, пока не будет достигнуто устойчивое положение рабочей точки в верхнем левом углу на рис. 5.3. Если провести анализ любой из устойчивых точек, то увидим, что обратная связь возвращает схему назад в устойчивое состояние, а не в сторону от него.

### 5.3. Защелки и триггеры

Защелки и триггеры являются основными элементами в большинстве последовательностных устройств.

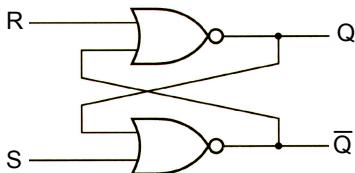
Триггером называют последовательностную схему, в которой выходные сигналы могут изменяться только в моменты времени, задаваемые тактовым сигналом, с учетом состояния других входных сигналов. Кроме триггеров, существует последовательностная схема *защелка* (*latch*), которая чувствительна к сигналам на входах в течение всего времени, и значения сигналов на выходах такой схемы могут изменяться в любой момент независимо от тактового сигнала.

Функциональное поведение защелок и триггеров совершенно различно, поэтому при создании цифровых схем важно знать, к какому типу относится используемый элемент.

#### 5.3.1. SR-защелка

На рис. 5.4а показана *SR-защелка* (*latch*) на вентилях ИЛИ-НЕ. У этой схемы два входа  $S$  и  $R$  и два выхода  $Q$  и  $\bar{Q}$ . Сигнал  $\bar{Q}$  в нормальных условиях представляет собой инверсию сигнала  $Q$ .

Если оба входных сигнала  $S$  и  $R$  равны 0, то *SR*-защелка ведет себя как рассмотренная выше схема с двумя устойчивыми состояниями: благодаря наличию петли обратной связи сохраняется одно из двух логических состояний:  $Q = 0$  или  $Q = 1$ .



S	R	$Q_n$	$\bar{Q}_n$
0	0	$Q_{n-1}$	$\bar{Q}_{n-1}$
0	1	0	1
1	0	1	0
1	1	0	0

Рис. 5.4. SR-зашелка: а) принципиальная схема на вентилях ИЛИ-НЕ; б) таблица состояний, описывающая работу схемы

Как указано в таблице на рис. 5.4б, подавая сигналы на входы  $S$  и  $R$ , можно перевести схему в желаемое состояние. Сигнал  $S$  устанавливает (*set*) или задает (*preset*) состояние, при котором выходной сигнал  $Q$  равен 1; сигнал  $R$  сбрасывает (*reset*) или очищает (*clear*) схему, в результате чего выходной сигнал  $Q$  становится равным 0. После того как входной сигнал  $S$  или  $R$  переходит на неактивный уровень, защелка остается в том состоянии, в какое ее установил входной сигнал. На рис. 5.5 приведены временные диаграммы поведение SR-зашелки при воздействии на нее некоторой последовательности входных сигналов. Стрелками показано какие переходы во входных сигналах вызывают те или иные переходы в выходных сигналах.

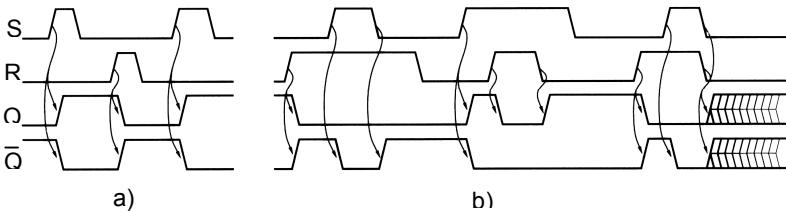


Рис. 5.5. Временные диаграммы, иллюстрирующие работу SR-зашелки:  
а) нормальные комбинации входных сигналов; б) сигналы  $S$  и  $R$  одновременно имеют активный уровень

На рис. 5.6 приведены варианты условных обозначений SR-зашелки. Эти обозначения различаются тем, как изображен инверсный выход. В первом из этих условных обозначений низ-

кий активный уровень сигнала на этом выходе указывается в имени сигнала внутри прямоугольника, изображающего функциональный блок. Однако при логическом проектировании предпочтительнее является второе обозначение, согласно которому символ инверсии располагается снаружи функционального прямоугольника.

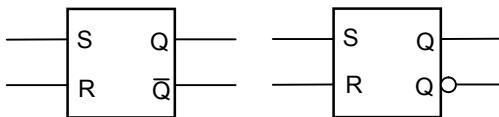


Рис. 5.6. Условное обозначение SR-защелки: а) без символа инверсии; б) предпочтительное обозначение при логическом проектировании

На рис. 5.7 показаны временные параметры SR-защелки. Задержка распространения  $t_p$  – это интервал времени между моментом перехода входного сигнала с неактивного уровня на активный и моментом изменения уровня сигнала на выходе. При наличии у защелки или триггера нескольких входов, как правило, указывается несколько значений задержки распространения, относящихся к каждой паре вход-выход.

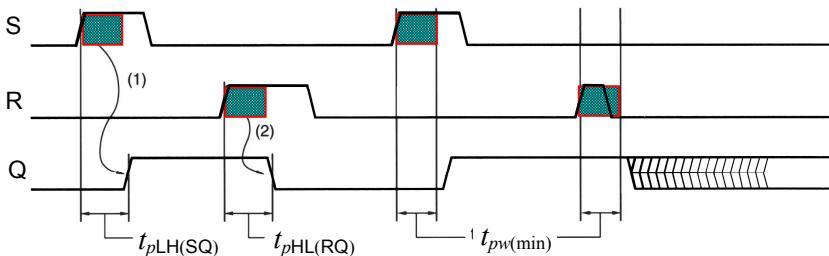


Рис. 5.7. Временные параметры SR-защелки

Кроме того, значения задержки распространения могут быть различными в зависимости от того, в каком направлении

изменяется выходной сигнал, от низкого уровня до высокого или наоборот. У SR-защелки переход входного сигнала S с низкого уровня на высокий может вызвать переход выходного сигнала Q с низкого уровня на высокий; этому переходу соответствует задержка распространения  $t_{pLH(SQ)}$  (случай 1 на рис. 5.7). Аналогично изменение входного сигнала R с низкого уровня на высокий может вызвать изменение выходного сигнала Q с высокого уровня на низкий с задержкой распространения  $t_{pHL(RQ)}$  (случай 2 на рис. 5.7).

Для входных сигналов S и R задается *минимальная длительность*. Если на входах S или R появляется импульс, длительность которого меньше минимального значения  $t_{pw(min)}$ , то, как показано на рис 5.7, защелка может войти в состояние неустойчивого равновесия и оставаться в нем в течение некоторого времени. Надежная работа защелки гарантирована только в том случае, когда выполнены требования в отношении минимальной длительности импульса на входах S и R.

Как уже упоминалось, SR-защелка может войти в квазиустойчивое состояние, если входные сигналы S и R одновременно переходят на неактивный уровень. Понятие одновременно для защелок, относящихся к конкретным логическим семействам, характеризуется вполне определенной величиной (это имеет место, например, если переходы сигналов S и R на неактивный уровень происходят в пределах отрезка времени длительностью менее некоторой определенной величины). Соответствующий параметр называют *временем восстановления*  $t_{rec}$  (*recovery time*). Это минимальный временной интервал между переходами S и R на неактивный уровень, при котором эти события считаются неодновременными. Параметр  $t_{rec}$  связан с минимальной длительностью импульса. Оба параметра характеризуют, как долго защелка приходит в равновесие при изменении состояния.

### 5.3.2. $\bar{S}\bar{R}$ -защелка

На рис. 5.8 представлена защелка на вентилях И-НЕ с низким активным уровнем сигналов установки и сброса. В логических семействах ТТЛ и КМОП  $\bar{S}\bar{R}$ -защелки используются гораз-

до чаще, чем SR-защелки, поскольку вентили И-НЕ предпочтительнее вентилей ИЛИ-НЕ.

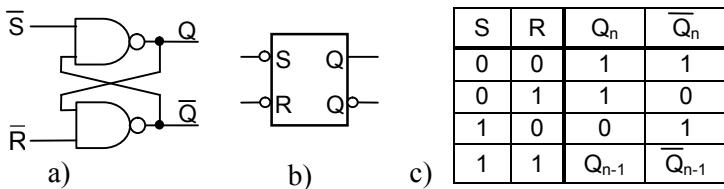


Рис. 5.8.  $\bar{S}\bar{R}$ -зашелка: а) принципиальная схема на вентилях И-НЕ; б) условное обозначение; в) таблица состояний, описывающая работу схемы

Из таблицы состояний, описывающей работу схемы, видно, что принцип действия  $\bar{S}\bar{R}$ -зашелки подобен механизму работы SR-зашелки; правда, с двумя существенными отличиями. Во-первых, у сигналов  $\bar{S}$  и  $\bar{R}$  активным является низкий уровень, так что при  $\bar{S} = \bar{R} = 1$  защелка сохраняет свое предыдущее состояние; на активный низкий уровень входных сигналов указывают символы инверсии на входах в условном обозначении. Во-вторых, при одновременном присутствии сигналов активного уровня на входах  $\bar{S}$  и  $\bar{R}$  оба выходных сигнала защелки становятся равными 1, а не 0, как это было в SR-зашелке. За исключением этих отличий,  $\bar{S}\bar{R}$ -зашелка работает точно так же, как SR-зашелка, в том числе в отношении временных требований и метастабильности.

#### 5.4. SR-зашелка с входом разрешения

SR- и  $\bar{S}\bar{R}$ -зашелки чувствительны к входным сигналам  $S$  и  $R$  в течение всего времени. Однако их легко видоизменить таким образом, чтобы схема была чувствительна к этим сигналам только тогда, когда подан сигнал на вход разрешения  $C$ . Такая SR-зашелка с входом разрешения показана на рис. 5.9. Как видно из таблицы состояний, описывающей работу схемы, при  $C$  рав-

ном 1, данная схема ведет себя так же, как SR-защелка, а при С равном 0, она не изменяет своего состояния при изменении сигналов S и R.

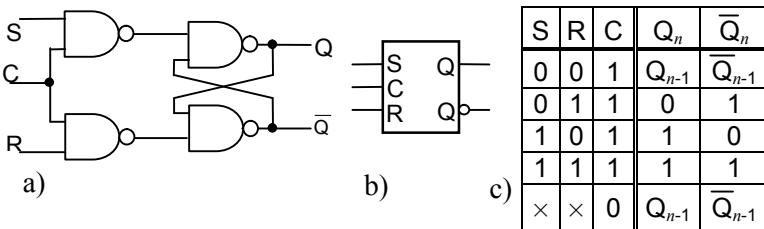


Рис. 5.9. SR-защелка с входом разрешения: а) принципиальная схема на вентилях И-НЕ; б) условное обозначение; в) таблица состояний, описывающая работу схемы

На рис. 5.10 приведены временные диаграммы, иллюстрирующие поведение этой схемы при некотором наборе входных сигналов. Если оба сигнала S и R равны 1 в момент, когда сигнал C переходит из 1 в 0, то схема ведет себя подобно SR-защелке при одновременном переходе сигналов S и R на активный уровень: следующее состояние непредсказуемо, и выходная цепь может стать метастабильной.

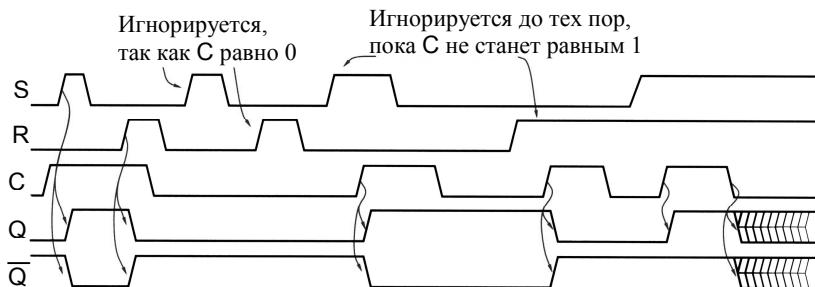


Рис. 5.10. Временные диаграммы работы SR-защелки с входом разрешения

## 5.5. D-защелка

SR-защелки применяются в управляющих устройствах, где возникают такие ситуации, когда в ответ на то или иное событие требуется запомнить этот факт (выставить *флаг*), а при изменении условий флаг сбросить. В таких случаях управление входами установки и сброса осуществляется фактически независимо. В тех случаях, когда нужно запоминать биты информации, поступающие каждый по отдельной сигнальной линии, используются так называемые D-защелки.

D-защелка показана на рис. 5.11. Ее схема содержит SR-защелку с входом разрешения и дополнительный инвертор, с помощью которого из единственного входного сигнала D формируются входные сигналы S и R. При этом устраняется присущий SR-защелке недостаток, связанный с одновременной подачей входных сигналов S и R активного уровня. На рис. 5.11с управляющий входной сигнал обозначен буквой C, но могут встречаться и обозначения ENABLE или CLK. Существуют варианты D-защелок у которых активным является низкий уровень управляющего сигнала.

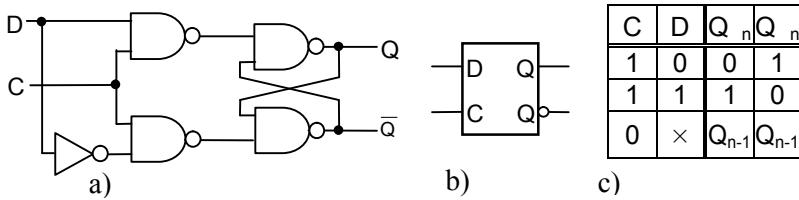


Рис. 5.11. D-защелка: а) принципиальная схема на вентилях И-НЕ; б) условное обозначение; в) таблица состояний, описывающая работу схемы

Пример временных диаграмм, иллюстрирующих работу D-защелки, приведен на рис. 5.12. Когда подан управляющий сигнал C, выходной сигнал Q повторяет значения входного сигнала D. В этом случае говорят, что защелка *прозрачна*; по этой

причине данную схему иногда называют *прозрачной защелкой*. Когда сигнал C принимает низкий уровень, защелка запирается; выходной сигнал Q сохраняет свое последнее значение до тех пор, пока C остается на неактивном уровне.

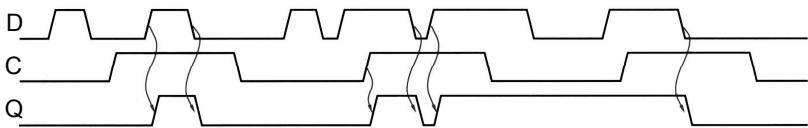


Рис. 5.12. Временные диаграммы, иллюстрирующие работу D-защелки при различных входных сигналах

Более детально процессы, происходящие в D-защелке, показаны на рис. 5.13. Переходные процессы описываются четырьмя параметрами, характеризующими прохождение сигналов от входов C и D до выхода Q. Рассмотрим ситуацию, предшествующую моментам переходов 1 и 4. Защелка заперта, а значение входного сигнала D противоположно значению выходного сигнала Q, поэтому когда C становится равным 1, защелка отпирается и выходной сигнал Q изменяет свое значение с задержками  $t_{pLH(CQ)}$  и  $t_{pHL(CQ)}$ . В моменты переходов 2 и 3 входной сигнал C равен 1 и защелка открыта, так что выходной сигнал Q напрямую повторяет переходы во входном сигнале D, но с задержками  $t_{pHL(DQ)}$  и  $t_{pLH(DQ)}$ . Аналогичные задержки наблюдаются и при изменениях выходного сигнала  $\bar{Q}$ .

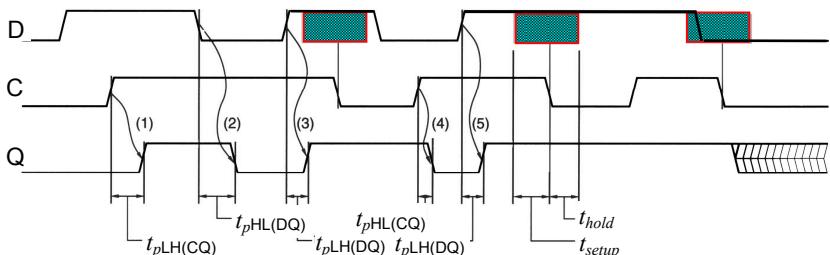


Рис. 5.13. Временные параметры D-защелки

У D-защелки не возникает ситуация, подобная той, которая имеет место у SR-защелки, когда  $S = R = 1$ , однако проблемы, связанные с метастабильностью, все же остаются. На рис. 5.13 штриховкой показаны интервалы времени до и после спадающего фронта в сигнале C, в пределах которых входной сигнал D не должен изменяться. Интервал, начинающийся до спадающего фронта в сигнале C, называется *временем установления* ( $t_{setup}$  (*setup time*)). Второй интервал времени  $t_{hold}$  называют *временем удержания* (*hold time*). Если входной сигнал D изменяется внутри интервала, состоящего из времени установления и времени удержания, то значение сигнала на выходе защелки непредсказуемо и состояние выходной цепи может оказаться метастабильным; этот случай изображен на рисунке вслед за последним перепадом, переводящим защелку в режим хранения.

## 5.6. D-триггер, переключающийся по фронту

*D-триггером, переключающимся по положительному фронту*, называют схему, показанную на рис. 5.14, в которой изменение выходных сигналов Q и  $\bar{Q}$  происходит только в моменты времени, задаваемые нарастающим фронтом управляющего сигнала CLK, в соответствии с значением сигнала на входе D. Первая защелка называется *ведущей* (*master*); при значении CLK, равном 0, она открыта и ее выходной сигнал повторяет входной сигнал. Когда сигнал CLK становится равным 1, ведущая защелка закрывается, а значение ее выходного сигнала переписывается во вторую защелку, называемую *ведомой* (*slave*). Ведомая защелка открыта в течение времени, пока значение CLK остается равным 1. Необходимо обратить внимание, что изменение сигнала на ее выходе возможно только в самом начале этого интервала, так как с этого момента ведущая защелка заперта и сигнал на ее выходе остается неизменным на протяжении всего этого отрезка времени.

Треугольник на входе CLK у D-триггера указывает на срабатывание схемы по фронту. Такой вход называется *динамическим*.

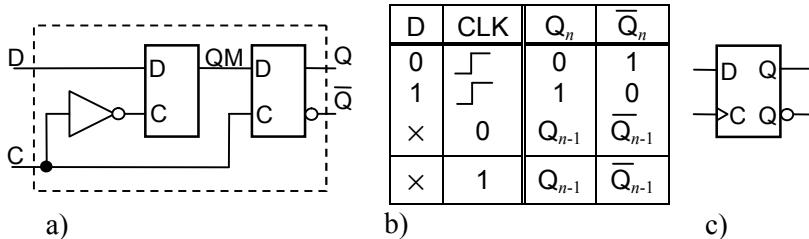


Рис. 5.14. D-триггер, переключающийся по положительному фронту:

а) принципиальная схема на D-защелках; б) таблица состояний, описывающая работу схемы; в) условное обозначение

На рис. 5.15 приведены примеры временных диаграмм, иллюстрирующих работу D-триггера. Выходной сигнал ведущей (Master) защелки обозначен как QM. Этот сигнал изменяется только при CLK, равном 0. Когда CLK становится равным 1, текущее значение QM сохраняется в ведомой защелке (Slave) и появляется на выходе Q. Сигнал QM не может измениться до тех пор, пока CLK снова не станет равным 0.

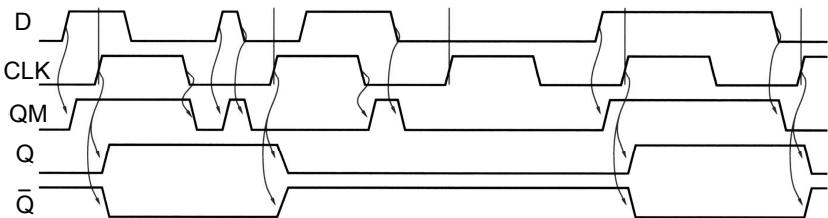


Рис. 5.15. Временные диаграммы D-триггера, переключающегося по положительному фронту

Более детально временные зависимости сигналов в D-триггере представлены на рис. 5.16. Задержки распространения измеряются относительно нарастающего фронта в сигнале CLK, поскольку только наступление этого события может вызвать изменение выходного сигнала. При переходе выходного сигнала с

низкого уровня на высокий и в обратном направлении задержки могут различаться.

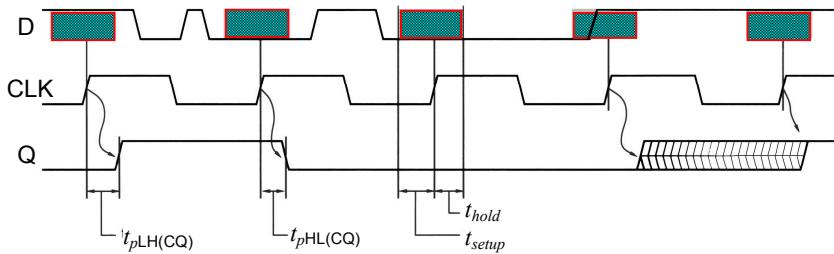


Рис. 5.16. Временные диаграммы сигналов в D-триггере, переключающемся по положительному фронту

Для D-триггера, как и для D-защелки задается интервал времени, состоящий из времени установления и времени удержания, в течение которого входной сигнал D должен оставаться неизменным. Этот интервал времени, расположенный до и после переключающего фронта сигнала CLK, на рис. 5.16 отмечен штриховкой. В случае невыполнения требований, предъявляемых временем установления и временем удержания, выход триггера переходит в одно из устойчивых состояний 0 или 1, хотя предсказать, в какое именно, нельзя. Иногда на выходе могут наблюдаться колебания или схема переходит в метастабильное состояние посередине между 0 и 1, как показано на рисунке после предпоследнего такта управляющего сигнала. Если триггер попадает в метастабильное состояние, то со случайной задержкой он сам вернется в одно из своих устойчивых состояний. Если в момент действия следующего переключающего фронта тактового сигнала, сигнал на входе D не изменяется в пределах времени установления и времени удержания, то триггер перейдет в устойчивое состояние; этот случай наблюдается в последнем такте управляющего сигнала на рис. 5.16.

Если на вход первой D-защелки подавать неинвертированный тактовый сигнал, а на вход второй D-защелки – инвертированный, то получим *D-триггер, переключающийся по отрица-*

тельному фронту, так что все изменения состояния триггера будут происходить на спадающем фронте сигнала CLK. Таблица состояний, описывающая работу такого триггера, и его условное обозначение приведены на рис. 5.17.

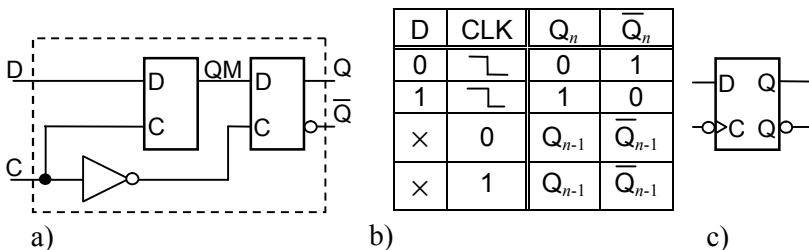


Рис. 5.17. D-триггер, переключающийся по отрицательному фронту:  
а) принципиальная схема на D-защелках; б) таблица состояний,  
описывающая работу схемы; в) условное обозначение

Большинство D-триггеров имеет *асинхронные входы*, подавая сигналы на которые можно переводить триггер в то или другое состояние независимо от значения сигналов на входах CLK и D. Эти входы обозначаются PR (*preset*, установка в единичное состояние) или S (*set*) и CLR (*clear*, сброс) или R (*reset*); по отношению к сигналам на этих входах D-триггер ведет себя как SR-защелка. Условное обозначение переключающегося по фронту D-триггера с такими входами и его принципиальная схема на вентилях И-НЕ приведены на рис. 5.18. Обычно эти входы используются для установки последовательностной схемы в заданное начальное состояние.

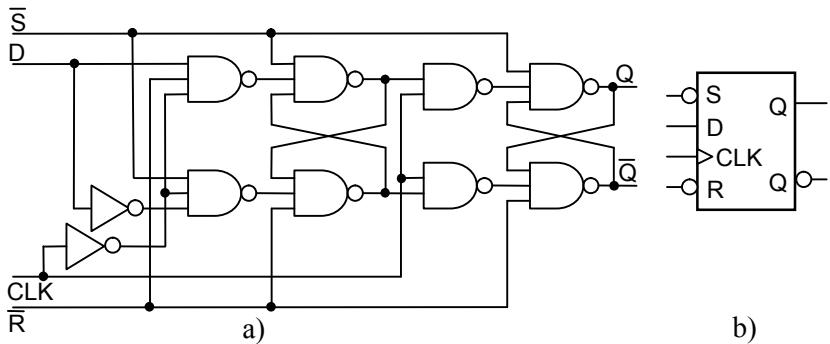


Рис. 5.18. D-триггер, переключающийся по положительному фронту, с входами установки и сброса: а) принципиальная схема с использованием элементов И-НЕ; б) условное обозначение

D-триггеры, переключающиеся по положительному фронту, выпускаемые в виде микросхем малой и средней степени интеграции, а также в программируемых интегральных схемах, устроены не по принципу *ведущий–ведомый*. Триггеры реализованы по схеме, изображенной на рис. 5.19. Эти триггеры содержат меньшее количество вентилей, и работают быстрее.

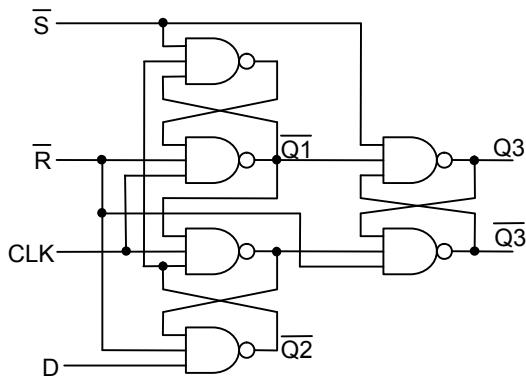


Рис. 5.19. Серийная схема D-триггера, переключающегося по положительному фронту

## 5.7. D-триггер с входом разрешения

Во многих случаях требуется, чтобы выполняемая D-триггерами функция состояла не в загрузке нового значения на каждом такте управляющего сигнала, а в удержании последнего запомненного значения. Это осуществляется путем добавления *входа разрешения (enable input)*, обозначаемого как EN или CE (*clock enable, разрешение тактового сигнала*). Назначение дополнительного входа состоит не в том, чтобы пропускать или не пропускать тактовый сигнал. Как показано на рис. 5.20а, с помощью 2-входового мультиплексора выбирается сигнал, подаваемый на внутренний D-вход триггера.

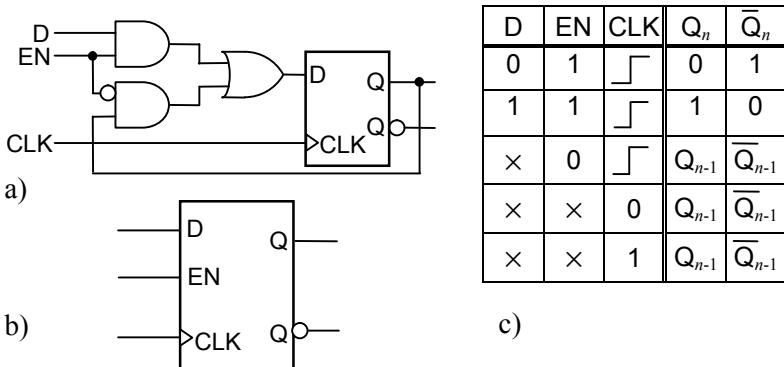


Рис. 5.20. Переключающийся по фронту D-триггер с входом разрешения:  
а) принципиальная схема; б) таблица, описывающая работу схемы;  
с) условное обозначение

При активном значении сигнала EN, выбирается сигнал, поступающий на внешний D-вход; когда сигнал EN переходит на неактивный уровень, через мультиплексор пропускает на внутренний D-вход текущее значение сигнала на выходе триггера. Таблица, описывающая работу схемы, имеет вид, представленный на рис. 5.20б. Условное обозначение такого триггера приведено на рис. 5.20с; у некоторых триггеров активным является

низкий уровень сигнала на входе разрешения, что отражается помещением символа инверсии на входе EN.

## 5.8. Двухтактный SR-триггер

SR-защелки полезны в тех устройствах, где возможны независимые условия установки 1 и 0. Если предполагается, что состояние защелки должны меняться только в определенные моменты времени, определяемые тактовым сигналом, то нужен SR-триггер, подобный D-триггеру, у которого сигналы на выходах изменялись бы только на определенном перепаде тактового сигнала. Далее описываются триггеры, используемые при решении таких задач.

Если D-защелки в D-триггере, переключающемся по отрицательному фронту (рис. 5.17а), заменить SR-защелками, то получим *двуихтактный SR-триггер* (триггер, действующий по принципу ведущий–ведомый), показанный на рис. 5.21.

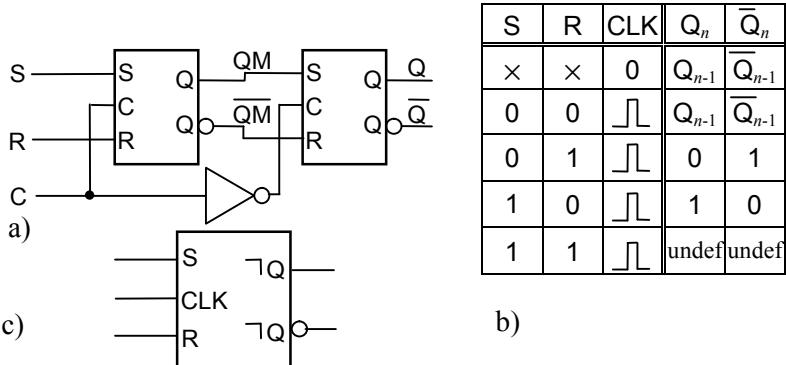


Рис. 5.21. Двухтактный SR-триггер: а) схема на SR-защелках; б) таблица, описывающая работу схемы (undef – неопределенное значение);  
с) условное обозначение

Подобно D-триггеру, сигналы на выходе SR-триггера изменяются только по спадающему фронту тактового сигнала С. Однако новое значение выходного сигнала зависит теперь не от

значений входных сигналов точно в тот момент времени, на который приходится спадающий фронт, а от значений входных сигналов на протяжении всего интервала времени, в течение которого сигнал  $C$  был равен 1 перед отрицательным перепадом.

Как следует из временных диаграмм на рис. 5.22, относительно короткий импульс  $S$  в любом месте в пределах интервала времени, когда  $C$  равно 1, может установить ведущую защелку в единичное состояние; точно так же импульс на входе  $R$  может сбросить ее в нулевое состояние. Значение, появляющееся на выходе триггера в момент действия спадающего фронта сигнала  $C$ , зависит от того, каким было последнее состояние ведущей защелки, пока сигнал  $C$  оставался равным 1.

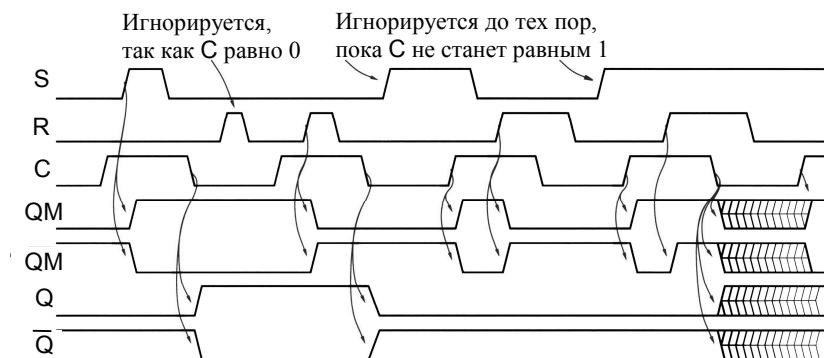


Рис. 5.22. Временные диаграммы, иллюстрирующие работу двухтактного SR-триггера

В соответствии с рис. 5.21с, в условном обозначении двухтактного SR-триггера не используется указатель динамического входа, поскольку этот триггер не является в полном смысле переключающимся по фронту. Он представляет собой защелку, в которой происходит повторение ее входного сигнала в течение интервала времени, на котором  $C$  равняется 1, но изменения сигнала на выходе триггера отражают запомненное значение только тогда, когда  $C$  становится равным 0. В условном обозначении

SR-триггера этот факт отражается индикатором задержки срабатывания. Триггеры, ведущие себя таким образом, называют триггерами со срабатыванием по импульсу.

Если в момент отрицательного перепада в сигнале С оба входных сигнала S и R имеют активный уровень, то состояние двухтактного SR-триггера оказывается непредсказуемым. В этом случае к моменту появления спадающего фронта оба выходных сигнала ведущей защелки QM и  $\overline{QM}$  равны 1. Когда сигнал С становится равным 0, значения сигналов на выходах ведущей защелки предсказать нельзя; более того, ее выходная цепь может оказаться метастабильной. В то же самое время ведомая защелка открывается, и весь этот мусор проходит на выход триггера.

## 5.9. Двухтактный JK-триггер

Сложности, связанные с тем, что делать, когда одновременно активные значения имеют сигналы S и R, решаются в двухтактном JK-триггере. У JK-триггера входы J и K аналогичны входам S и R. Однако, как показано на рис. 5.23, сигнал J попадает на вход S ведущей защелки только в том случае, когда текущее значение сигнала на выходе триггера  $\overline{Q}$  равно 1 (то есть сигнал Q равен 0), а сигнал K попадает на вход R ведущей защелки только тогда, когда текущее значение Q, равно 1. Таким образом, при одновременном действии сигналов на входах J и K триггер переходит в состояние, противоположное тому, в котором он находился.

На рис. 5.24 приведены временные диаграммы работы двухтактного JK-триггера при некотором наборе входных сигналов. Для изменения сигнала на выходе триггера в момент окончания переключающего импульса не требуется подавать входные сигналы J и K именно в этот момент времени. Объясняется это тем, что только один из входных сигналов проходит через вентили на входы S и R ведущей защелки, сигнал на выходе триггера может стать равным 1, при условии, что предыдущее значение Q равнялось 0, даже в том случае, когда к концу переключающего импульса действует только сигнал K, а сигнал J снят. Такое пове-

дение, называемое *захватом единиц*, показано на рисунке на предпоследнем переключающем импульсе. Аналогичное явление, называемое *захватом нулей*, показано на последнем переключающем импульсе. Из-за этого свойства двухтактного JK-триггера входные сигналы J и K должны оставаться неизменными в течение всего интервала времени, пока C равно 1.

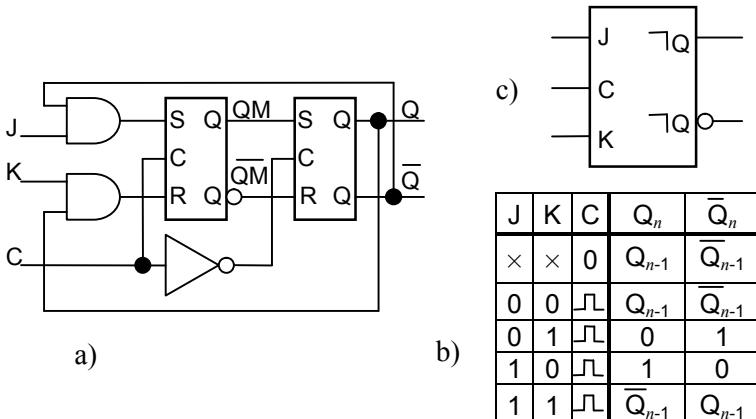


Рис. 5.23. Двухтактный JK-триггер: а) схема на SR-защелках; б) таблица, описывающая работу схемы; в) условное обозначение

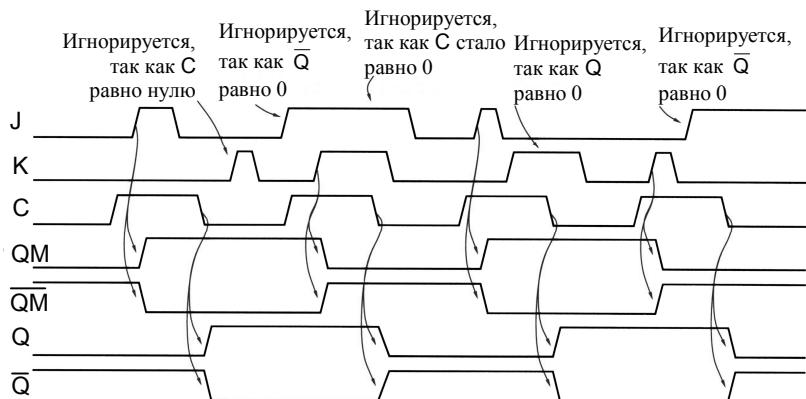


Рис. 5.24. Временные диаграммы, иллюстрирующие работу двухтактного JK-триггера

## 5.10. JK-триггер, переключающийся по фронту

Проблема захвата единиц и нулей решена в *JK-триггере, переключающемся по фронту*, схема которого изображена на рис. 5.25. В нем используется переключающийся по фронту D-триггер, благодаря чему входы JK-триггера опрашиваются на нарастающем фронте тактового сигнала и очередное значение выходного сигнала вырабатывается в соответствии с выражением:  $Q_n = J \cdot \overline{Q_{n-1}} + \overline{K} \cdot Q_{n-1}$ .

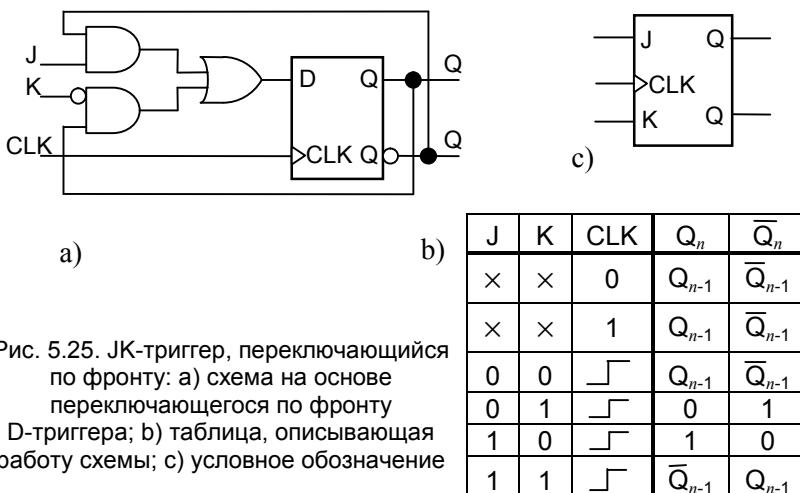


Рис. 5.25. JK-триггер, переключающийся по фронту: а) схема на основе переключающегося по фронту D-триггера; б) таблица, описывающая работу схемы; в) условное обозначение

Временные диаграммы, иллюстрирующие работу переключающегося по фронту JK-триггера представлены на рис. 5.26. Так же, как и в случае сигнала на D входе D-триггера, для правильной работы JK-триггера его входные сигналы J и K до и после переключающего фронта тактового сигнала должны удовлетворять требованиям, вытекающим из значений времени установления и времени удержания.

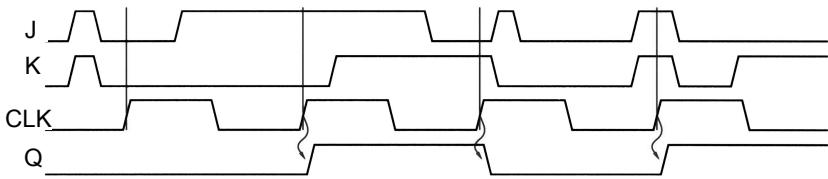


Рис. 5.26. Поведение JK-триггера, переключающегося по положительному фронту

В JK-триггерах, переключающихся по фронту, отсутствуют проблемы, связанные с захватом единиц и нулей и проблемы, возникающие при одновременном появлении активных уровней управляющих входных сигналов. Эти триггеры вытеснили переключающиеся по фронту триггеры более старых типов. Микросхемы  $74 \times 109$  (рис. 5.27) в семействе ТТЛ представляют собой JK-триггеры, переключающиеся по положительному фронту сигнала CLK, с активным низким уровнем входного сигнала K, который обозначается  $\bar{K}$ .

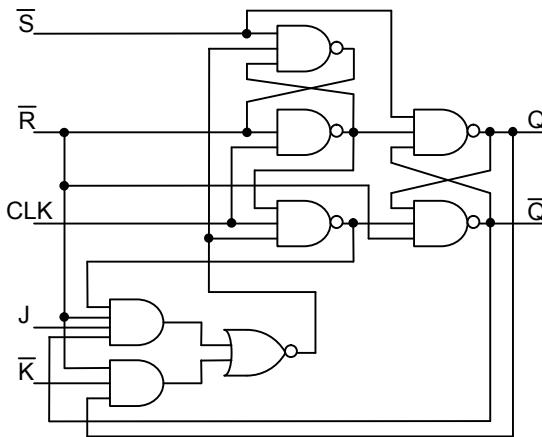


Рис. 5.27. Внутреннее устройство переключающегося по положительному фронту JK-триггера 74LS109

Структура ИС  $74 \times 109$  похожа на структуру ИС  $74LS74$ , схема которой была приведена на рис. 5.19. Сравнение схем на рис. 5.19 и 5.27 показывает, что микросхема '109 получается из микросхемы '74 заменой нижнего левого вентиля, реализующего характеристическое уравнение  $Q_n = D$ , на структуру И-ИЛИ, реализующую характеристическое уравнение JK-триггера:

$$Q_n = J \cdot \overline{Q}_{n-1} + \overline{K} \cdot Q_{n-1}.$$

Самым распространенным применением JK-триггеров являются тактируемые синхронные конечные автоматы. Однако в большинстве случаев конечные автоматы строятся на D-триггерах, поскольку проектирование при этом проще и большинство создаваемых последовательностных устройств состоит из D-триггеров, а не из JK-триггеров. Поэтому в дальнейшем основное внимание будет обращено на D-триггеры.

## 5.11. T-триггер

Триггер, изменяющий свое состояние на каждом периоде тактового сигнала, называется *T-триггером*. Частота переключений сигнала на выходе триггера Q равна половине частоты переключений входного сигнала T. На рис. 5.28 приведены условное обозначение T-триггера, переключающегося по положительному фронту тактового сигнала, и временные диаграммы, иллюстрирующие работу такого триггера. T-триггер можно реализовать используя D- или JK-триггера, как показано на рис. 5.29. T-триггеры чаще всего используются в счетчиках и делителях частоты.

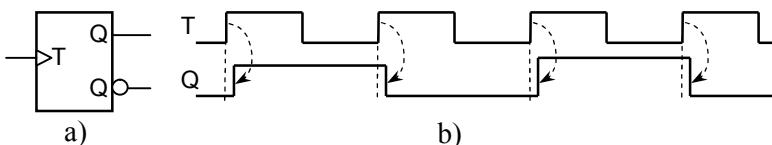


Рис. 5.28. T-триггер, переключающийся по положительному фронту:  
а) условное обозначение; б) временные диаграммы

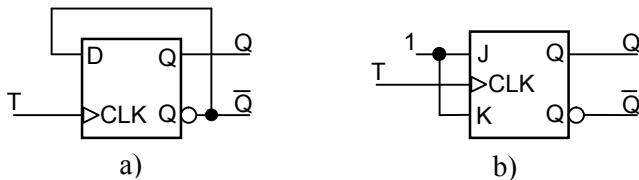


Рис. 5.29. Возможные схемы Т-триггеров: а) на основе D-триггера; б) на основе JK-триггера

В тех случаях, когда требуется чтобы Т-триггер переключался не на каждом такте, можно воспользоваться *T-триггером с входом разрешения* (рис. 5.30). Такой триггер изменяет свое состояние на переключающем фронте тактового сигнала только в том случае, когда на входе EN присутствует разрешающий сигнал. Подобно сигналам на входах D, J и K у других триггеров, переключающихся по фронту, сигнал на входе EN должен удовлетворять требованию оставаться неизменным в интервале, равном времени установления и времени удержания, в окрестности переключающего фронта тактового сигнала. Вход разрешения EN можно получить воспользовавшись схемами, приведенными на рис. 5.31.

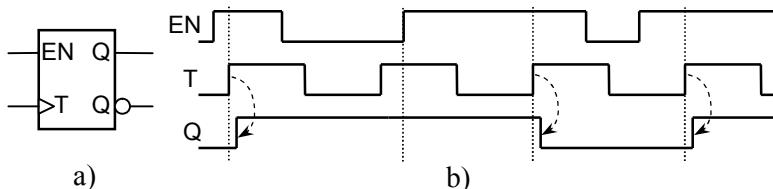


Рис. 5.30. Т-триггер с входом разрешения, переключающийся по положительному фронту: а) условное обозначение; б) временные диаграммы

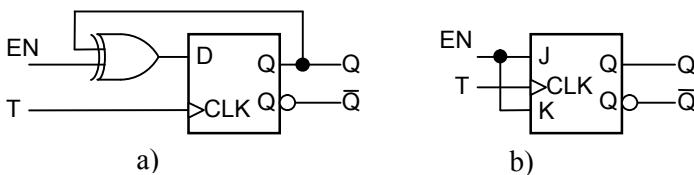


Рис. 5.31. Возможные схемы Т-триггеров с входом разрешения: а) на основе D-триггера; б) на основе JK-триггера

# ГЛАВА 6. СЧЕТЧИКИ, РЕГИСТРЫ СДВИГА

## 6.1. Счетчики

*Счетчиком (counter)* называют любую тактируемую последовательностную схему, у которой диаграмма состояний представляет собой единственное кольцо (рис. 6.1). *Модулем счета* называется число состояний в этом кольце. О счетчике с  $m$  состояниями говорят как о *счетчике с модулем счета  $m$* ; иногда его называют также *делителем частоты на  $m$* . У счетчика с модулем счета, не равным степени 2, есть состояния, в которые он не попадает при нормальной работе.

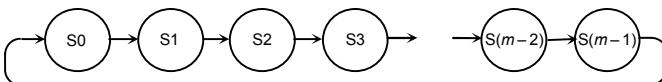


Рис. 6.1. Диаграмма состояний счетчика – единственное кольцо

Широко распространеными являются *n-разрядные двоичные счетчики*. Такой счетчик состоит из  $n$  триггеров, и у него имеется  $2^n$  состояний, через которые он проходит в последовательности  $0, 1, 2, \dots, 2^n - 1, 0, 1, \dots$ . Каждое состояния характеризуется соответствующим  $n$ -разрядным двоичным числом.

### 6.1.1. Счетчики с последовательным переносом

Для построения простейшего  $n$ -разрядного двоичного счетчика потребуется только  $n$  триггеров и никаких других компонентов. Для  $n = 4$  такой счетчик показан на рис. 6.2. Напомним, что состояние Т-триггера меняется (на противоположное) с каждым нарастающим фронтом сигнала на его тактовом входе. Таким образом, содержимое того или иного разряда в счетчике меняется на противоположное только тогда, когда значение бита в разряде, непосредственно предшествующем этому разряду, изменяется с 1 на 0. Это соответствует двоичному счету в прямом направлении: когда бит, хранящийся в данном разряде, изменяется с 1 на 0, возникает перенос в следующий по старшинству разряд.

Такой счетчик называют *счетчиком с последовательным переносом*, так как информация о переносе поочередно передается от младших разрядов к старшим. Двоичный счетчик может быть создан с использованием любых триггеров, рассмотренных выше.

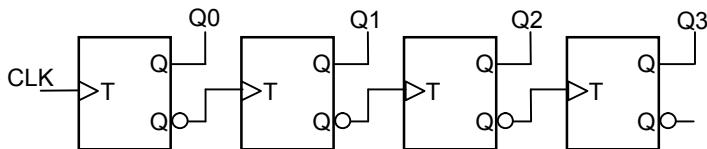


Рис. 6.2. 4-разрядный двоичный счетчик с последовательным переносом

### 6.1.2. Синхронные счетчики

Хотя для построения счетчика с последовательным переносом требуется компонентов меньше, чем для двоичного счетчика любого другого типа, за это приходится платить наименьшим быстродействием по сравнению с любыми другими счетчиками. В худшем случае, когда должен измениться бит в самом старшем разряде, выходной сигнал примет правильное значение только спустя время, равное  $n \cdot t_{TQ}$ , после нарастающего фронта тактового сигнала CLK, где  $t_{TQ}$  – задержка распространения сигнала в T-триггере от входа до выхода.

В *синхронном счетчике* к тактовым входам всех триггеров подводится один и тот же общий тактовый сигнал CLK, так что изменения значений сигналов на выходах всех триггеров происходят в один и тот же момент времени с задержкой только на  $t_{TQ}$ . Как видно из рис. 6.3, для этого нужно воспользоваться T-триггерами с входом разрешения; сигнал на выходе триггера примет противоположное значение в момент, задаваемый нарастающим фронтом сигнала на его входе T, только в том случае, если сигнал разрешения EN имеет активный уровень. Какие именно триггеры перейдут в состояние, противоположное предыдущему, на очередном нарастающем фронте сигнала на входе T, определяется комбинационной логикой, включенной на входах разрешения EN.

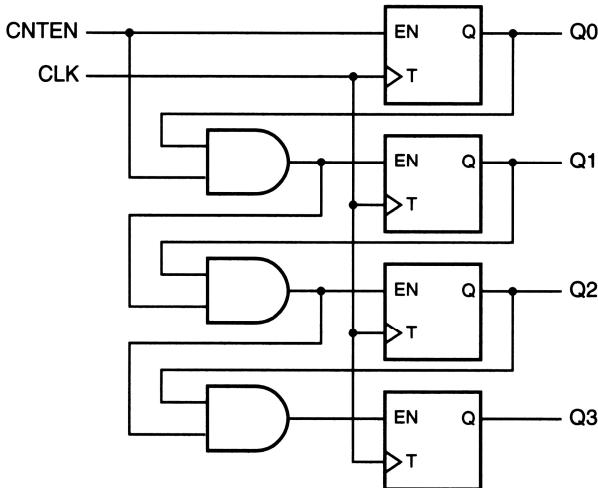


Рис. 6.3. 4-разрядный синхронный двоичный счетчик с последовательной логикой разрешения

На рис. 6.3 показан счетчик, снабженный сигналом разрешения счета CNTEN. Любой из Т-триггеров может переключиться тогда и только тогда, когда сигнал CNTEN имеет единичное значение и равны 1 биты во всех разрядах, младше данного. Как и в случае двоичного счетчика с последовательным переносом,  $n$ -разрядный синхронный счетчик можно построить так, чтобы на один разряд приходилось фиксированное число логических схем; в данном случае в каждом разряде необходимы Т-триггер с входом разрешения и 2-входовой вентиль И.

Счетчик, изображенный на рис. 6.3, называют *последовательным синхронным счетчиком*, поскольку сигналы разрешения проходят через комбинационную логику последовательно от младшего разряда к старшему. Если период тактового сигнала слишком мал, то изменение в младшем разряде счетчика может не успеть дойти за это время до старшего разряда. Этот недостаток устранен в схеме на рис. 6.4, где сигнал разрешения на входе разрешения EN каждого триггера вырабатывается соответствую-

щим вентилем И всего лишь с одним уровнем логики. В результате получается схема двоичного счетчика с самым высоким быстродействием, который носит название *параллельного синхронного счетчика*.

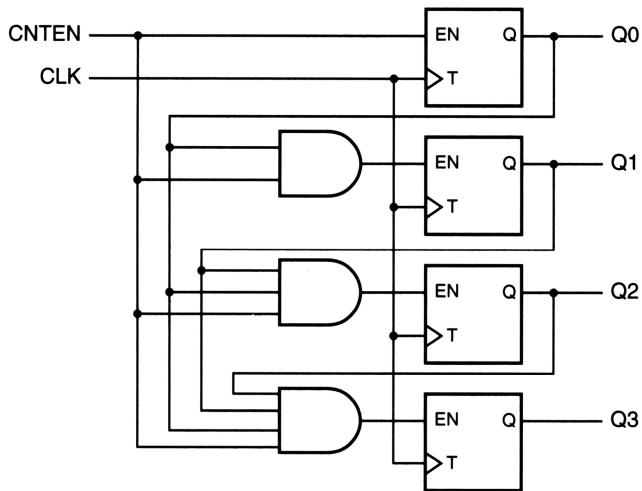


Рис. 6.4. 4-разрядный синхронный двоичный счетчик с параллельной логикой разрешения

### 6.1.3. Счетчики в ИС средней степени интеграции и их применение

Самым популярным счетчиком в ИС средней степени интеграции является 4-разрядный синхронный двоичный счетчик 74 × 163 с входами сброса и загрузки; сигналы на этих входах имеют низкий активный уровень. Условное обозначение такого счетчика приведено на рис. 6.5. Работа этого счетчика описывается таблицей состояний (табл. 6.1), а его принципиальная схема показана на рис. 6.6.

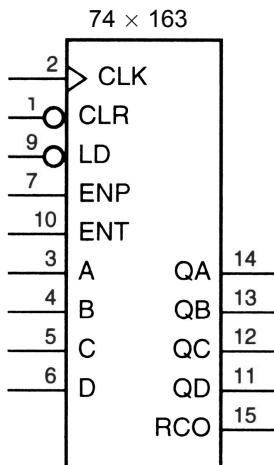


Рис. 6.5. Традиционное условное обозначение ИС  $74 \times 163$

В этой интегральной схеме используются не Т-триггеры, а D-триггеры, чтобы упростить функции загрузки и сброса. На D-вход каждого триггера сигнал поступает с выхода 2-ходового мультиплексора, состоящего из двух вентилей И и вентиля ИЛИ. Выходной сигнал мультиплексора равен 0, если входной сигнал  $\overline{CLR}$  равен 0. В противном случае верхний из вентилей И пропускает входной сигнал данных (A, B, C или D) на выход, если сигнал  $\overline{LD}$  равен 0. Если ни у одного из сигналов  $\overline{CLR}$  и  $\overline{LD}$  уровень не является активным, то нижний из вентилей И пропускает на выход мультиплексора выходной сигнал вентиля ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ.

Функция счета в ИС '163 выполняется с помощью вентилей ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ. На один из входов этого вентиля в каждом разряде поступает инверсия бита, хранящегося в этом разряде (QA, QB, QC или QD); на другой вход подана логическая 1, благодаря чему на выходе этого вентиля вырабатывается дополнение к биту, хранящемуся в данном разряде, но только в том случае, когда оба сигнала разрешения ENP и ENT имеют актив-

ный уровень и во всех разрядах счетчика, младше данного, биты равны 1.

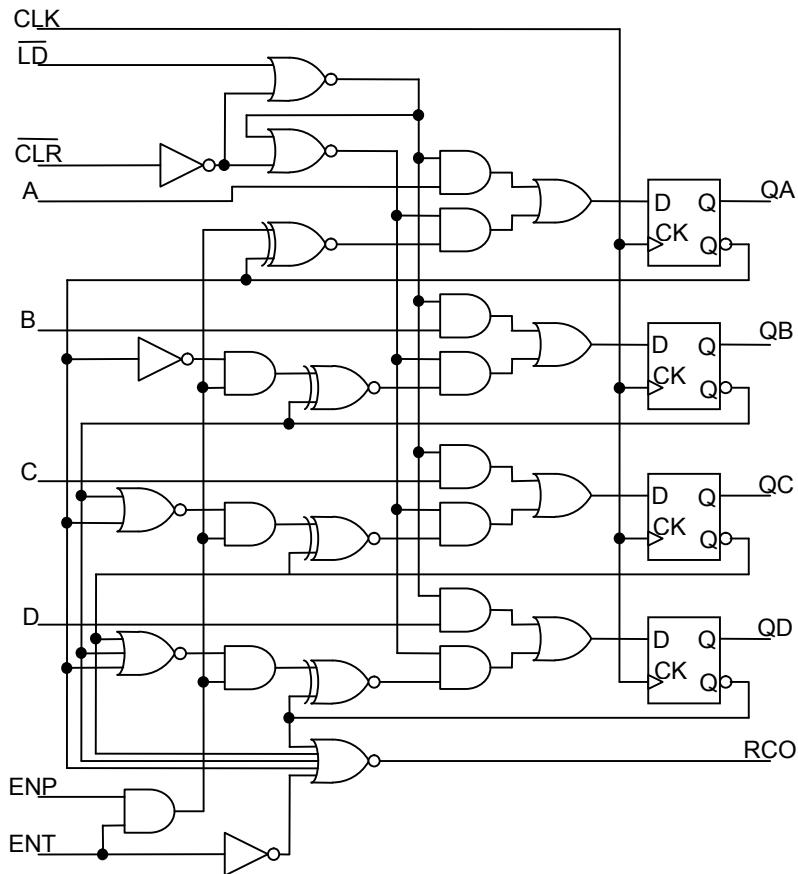


Рис. 6.6. Принципиальная схема синхронного 4-разрядного двоичного счетчика 74 × 163

Сигнал RCO (выход сквозного переноса) означает наличие переноса из самого старшего разряда; он равен 1, когда равны 1 биты, хранящиеся во всех разрядах счетчика, и подан сигнал разрешения ENT.

Таблица 6.1

Таблица состояний 4-разрядного двоичного счетчика  $74 \times 163$ 

CLR	$\overline{LD}$	ENT	ENP	$QD_n$	$QC_n$	$QB_n$	$QA_n$	$QD_{n+1}$	$QC_{n+1}$	$QB_{n+1}$	$QA_{n+1}$
0	x	x	x	x	x	x	x	0	0	0	0
1	0	x	x	x	x	x	x	D	C	B	A
1	1	0	x	x	x	x	x	$QD_n$	$QC_n$	$QB_n$	$QA_n$
1	1	x	0	x	x	x	x	$QD_n$	$QC_n$	$QB_n$	$QA_n$
1	1	1	1	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	1	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	1
1	1	1	1	0	0	1	1	0	1	0	0
1	1	1	1	0	1	0	0	0	1	0	1
1	1	1	1	0	1	0	1	0	1	1	0
1	1	1	1	0	1	1	0	1	0	1	1
1	1	1	1	0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0	1	0	0	1
1	1	1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0

Несмотря на наличие входа разрешения, *счетчики*, выполненные в виде ИС средней степени интеграции, часто *работают в непрерывном режиме*, когда счет разрешен постоянно. На рис. 6.7 показано, какие соединения необходимо осуществить, чтобы счетчик '163 работал в таком режиме, а на рис. 6.8 приведены соответствующие этому режиму временные диаграммы. Начиная с сигнала  $QA$ , частота переключений каждого следующего сигнала вдвое меньше, чем предыдущего. Таким образом, в режиме непрерывного счета ИС '163 может играть роль делителя частоты на 2, 4, 8 или 16; при этом ненужные старшие разряды игнорируются.

Счетчик '163 является полностью синхронным; это означает, что сигналы на его выходах изменяются только на нарастаю-

щем фронте сигнала CLK. Но в некоторых приложениях бывает необходимо осуществлять сброс асинхронно; это позволяет ИС 74 × 161. У микросхемы '161 такая же цоколевка, как и у ИС '163, но внутри нее вход CLR подключен к асинхронным входам сброса триггеров.

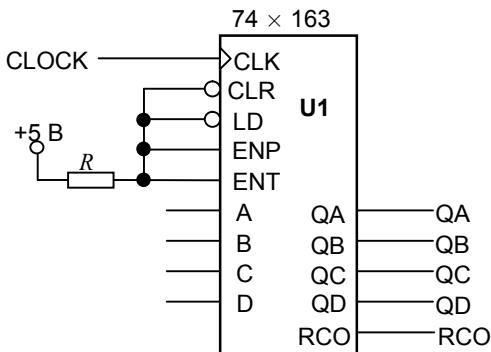


Рис. 6.7. Включение ИС 74 × 163 для работы в режиме непрерывного счета

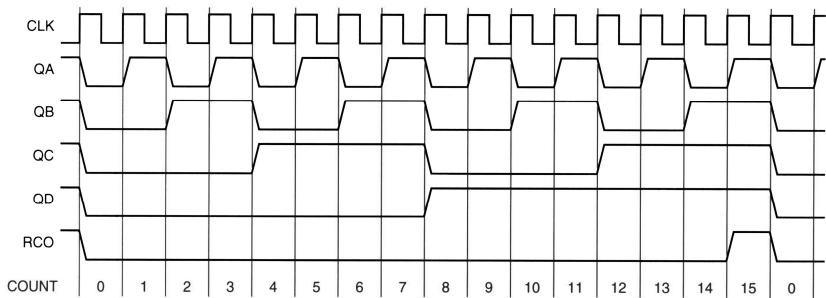


Рис. 6.8. Временные диаграммы тактового сигнала и сигналов на выходах отдельных разрядов в делителе частоты на 16

Другими вариантами счетчиков с точно такой же цоколевкой являются ИС 74 × 160 и 74 × 162; их функции в целом такие

же, как и у микросхем '161 и '163, за исключением того, что в них последовательность счета изменена: за состоянием 9 следует состояние 0. Другими словами, эти ИС представляют собой счетчики по модулю 10; их называют *декадными счетчиками*. На рис. 6.9 приведены временные диаграммы для счетчиков '160 и '162 в режиме непрерывного счета. Хотя частота сигналов на выходах QD и QC равна одной десятой от частоты сигнала CLK, коэффициент заполнения у сигналов QD и QC не равен 50 %, а сигнал QB, хотя и имеет частоту, в пять раз меньшую, чем частота тактового сигнала, его коэффициент заполнения не остается постоянным.

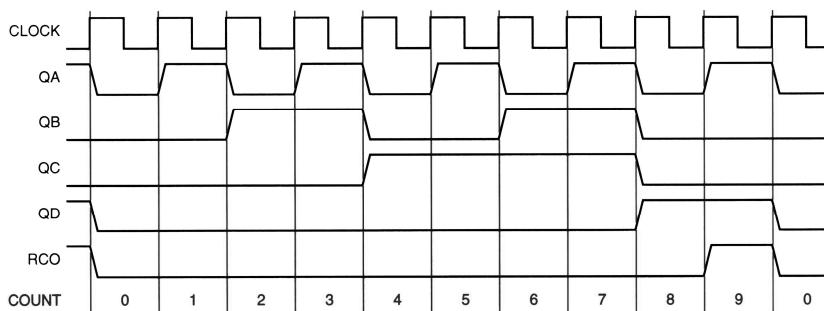


Рис. 6.9. Временные диаграммы тактового сигнала и сигналов на выходах отдельных разрядов делителя частоты на 10 в режиме непрерывного счета

Счетчик '163 сам по себе считает по модулю 16, но с помощью сигналов  $\overline{CLR}$  и  $\overline{LD}$  его можно заставить считать по модулю меньшему, чем 16, укоротив проходимую им последовательность состояний. На рис. 6.10, например, показано использование ИС '163 в качестве счетчика по модулю 11. Когда счетчик находится в состоянии 15, на выходе RCO возникает единичный сигнал, который заставляет счетчик перейти в следующее состояние, равное 5; поэтому схема считает от 5 до 15 и снова начинает счет с 5, так что всего в цикле счета 11 состояний.

На рис. 6.11 показан другой подход к решению задачи организации счетчика по модулю 11.

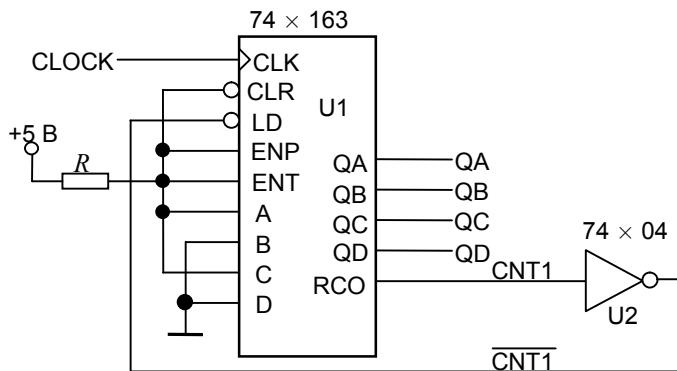


Рис. 6.10. Применение ИС  $74 \times 163$  в качестве счетчика по модулю 11  
с последовательностью счета 5, 6, ..., 15, 5, 6, ...

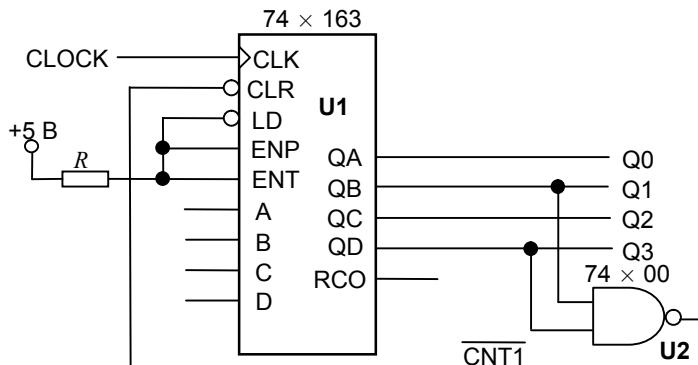


Рис. 6.11 Применение ИС  $74 \times 163$  в качестве счетчика по модулю 11  
с последовательностью счета 0, 1, 2, ..., 10, 0, 1, ...

В этой схеме для обнаружения состояния счетчика, равного 10, используется вентиль И-НЕ; сигнал, возникающий на выходе этого вентиля, заставляет счетчик перейти в состояние 0. Обрати-

те внимание на то, что для обнаружения состояния 10 (в двоичной записи – 1010) нужен вентиль только с двумя входами. Хотя для обнаружения состояния  $CNT10 = Q3 \cdot \overline{Q2} \cdot Q1 \cdot \overline{Q0}$  более естественным было бы применение 4-входового вентиля, можно обойтись 2-входовым вентилем благодаря тому, что в проходящем счетчиком последовательности состояний от 0 до 10 нет другого состояния с  $Q3 = 1$  и  $Q1 = 1$ . И в общем случае, для обнаружения состояния  $N$  двоичного счетчика, считающего от 0 до  $N$ , нужно объединить по И выходы только тех разрядов, на которых имеются единицы в двоичном представлении числа  $N$ .

Соединяя последовательно несколько ИС  $74 \times 163$ , можно реализовать двоичный счет по модулю, большему 16. На рис. 6.12 показана общая структура такой схемы.

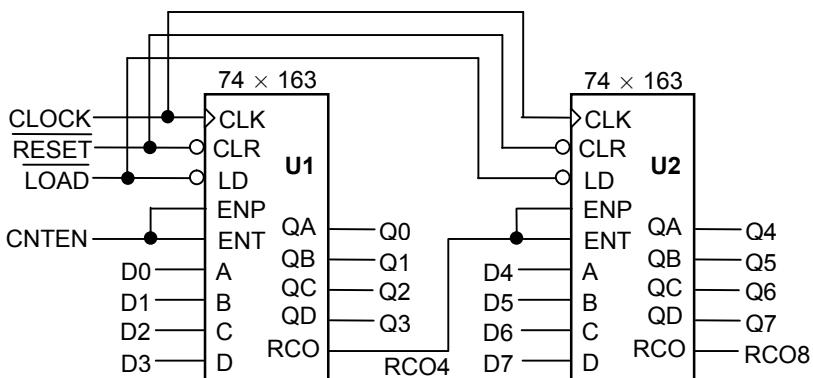


Рис. 6.12. Общая структура последовательного включения счетчиков  $74 \times 163$

Входы  $CLK$ ,  $\overline{CLR}$  и  $\overline{LD}$  всех ИС '163 соединены параллельно, так что все они переключаются, сбрасываются и загружаются в одно и то же время. Сигнал разрешения счета  $CNTEN$  подается на ИС '163, ответственную за младшие разряды. На выходе  $RCO4$  сигнал возникает только в том случае, если младшая ИС '163 находится в состоянии 15 и подан сигнал разрешения  $CNTEN$ ; вы-

ход RCO4 соединен с входами разрешения ИС '163, ответственной за старшие разряды. Таким образом, оба сигнала – информация о переносе и разрешение счета в целом – переходят от одного 4-разрядного счетчика к другому. Как и в случае последовательного синхронного счетчика, приведенного ранее на рис. 6.3, по этому принципу можно построить счетчик с любым числом разрядов; максимальная скорость счета ограничена задержкой прохождения сигнала переноса через все каскады.

В счетчиках типа 163' не всегда учитывают разницу между входами разрешения ENP и ENT, поскольку счет возможен только тогда, когда оба этих сигнала имеют активный уровень. Однако достаточно посмотреть на внутреннюю структуру ИС '163, чтобы увидеть различие между этими сигналами: сигнал ENT проходит также на выход сквозного переноса. Во многих случаях это различие является существенным.

Другой счетчик, похожий по выполняемым функциям на ИС 74 × 163, – это микросхема 74 × 169; ее условное обозначение приведено на рис. 6.13.

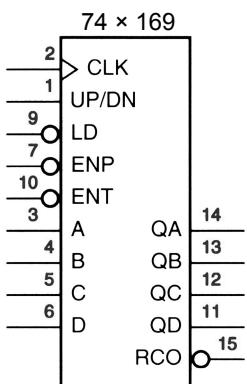


Рис. 6.13. Условное обозначение реверсивного счетчика 74 × 169

Одно из отличий счетчика '169 состоит в том, что выходной сигнал переноса и сигналы на входах разрешения имеют активный низкий уровень. Более существенно то, что ИС '169 является *реверсивным счетчиком*: он может считать в сторону увели-

чения или уменьшения содержащегося в нем двоичного числа в зависимости от значения входного сигнала UP/DN. Когда сигнал UP/DN равен 1, счет происходит в сторону увеличения; когда значение сигнала UP/DN равно 0, счет ведется в сторону уменьшения.

#### 6.1.4. Декодирование состояний двоичного счетчика

Объединяя двоичный счетчик с дешифратором, можно вырабатывать кодовые слова кода 1 из  $m$ , содержащие по одному единичному сигналу на каждое состояние счетчика. Это бывает полезно в том случае, когда с помощью счетчика управляют набором устройств: сигналы разрешения поступают на отдельные устройства, когда счетчик находится в соответствующем состоянии. При таком подходе каждый из выходных сигналов дешифратора служит сигналом разрешения для одного из устройств.

На рис. 6.14 показано, как можно связать между собой счетчик  $74 \times 163$ , считающий по модулю 8, и дешифратор  $3 \times 8$  типа  $74 \times 138$ ; в результате получается схема, вырабатывающая восемь сигналов, каждый из которых соответствует одному из состояний счетчика. На рис. 6.15 приведены временные диаграммы для этой схемы. Сигнал на каждом выходе дешифратора имеет активный уровень (нулевой) в течение периода тактового сигнала.

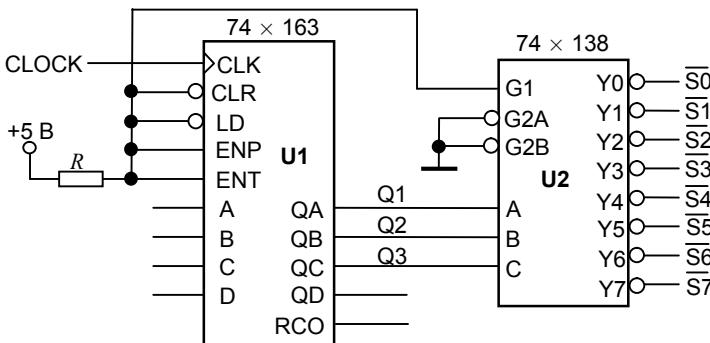


Рис. 6.14. Двоичный счетчик по модулю 8 с дешифратором

При изменении в счетчике содержимого двух или большего числа разрядов на выходах дешифратора могут возникать *паразитные импульсы*. Даже в синхронном счетчике типа '163 выходные сигналы изменяются не точно в одно и то же время, что связано с различным временем перехода из состояния 0 в состояние 1 и наоборот, хотя моменты изменения отличаются очень незначительно.

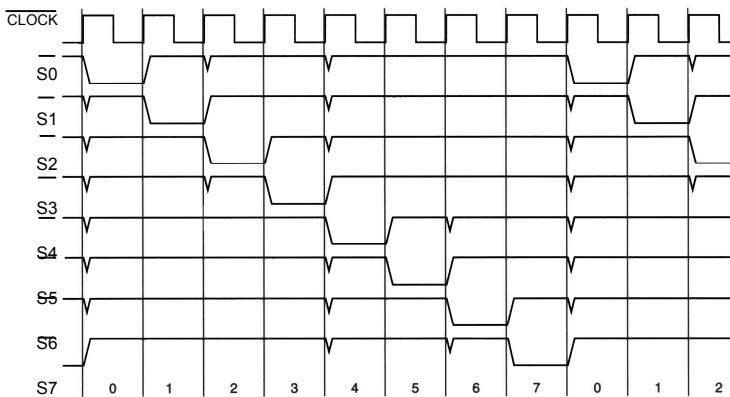


Рис. 6.15. Временные диаграммы для двоичного счетчика по модулю 8 с дешифратором, на которых видны паразитные импульсы на выходах дешифратора

Более важным является то, что в дешифраторе типа '138 задержки прохождения сигналов по разным путям различны; например, задержка прохождения сигнала от входа В к выходу  $\bar{Y}_1$  меньше, чем задержка на пути от входа А к выходу  $\bar{Y}_1$ . Поэтому даже если значения входных сигналов, равные 011, изменяются одновременно и становятся равными 100, в сигнале на выходе  $\bar{Y}_1$  может появиться паразитный импульс. В данном примере мы видим, что паразитные импульсы могут возникать при любой реализации функции двоичного декодирования; в таком случае говорят о наличии *функционального источника опасности*.

В большинстве приложений выходные сигналы дешифратора, изображенные на рис. 6.15, используются в качестве сигна-

лов, управляющих регистрами, счетчиками и другими устройствами, переключающимися по фронту. В этом случае указанные на рисунке паразитные импульсы на выходах дешифратора не страшны, поскольку они возникают строго *после* перепада в тактовом сигнале и кончаются задолго до очередного фронта тактового сигнала, когда выходные сигналы дешифратора принимаются во внимание другими устройствами, переключающимися по фронту. Однако паразитные импульсы могли бы вызвать затруднения при попытке использовать выходные сигналы дешифратора в качестве управляющих  $\bar{S}$  или  $\bar{R}$  сигналов в  $\bar{S}\bar{R}$ -защелках. Точно так же подобные сигналы, в которых потенциально могут иметь место паразитные импульсы, ни в коем случае нельзя применять в качестве тактовых сигналов для переключающихся по фронту устройств.

Если нужно очистить сигналы от паразитных импульсов, показанных на рис. 6.15, то один из способов сделать это состоит в подаче выходных сигналов дешифратора '138 на 8 D-триггеров (8-разрядный регистр), в котором установившиеся значения этих сигналов фиксировались бы на следующем фронте тактового сигнала (рис. 6.16).

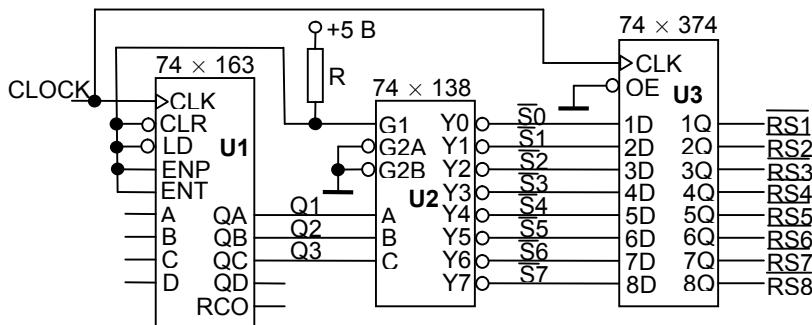


Рис. 6.16. Схема двоичного счетчика по модулю 8 с дешифратором без паразитных импульсов на выходах

Сигналам на выходах регистра присвоены другие имена, чтобы учесть задержку на один такт при прохождении через регистр. Однако существует менее дорогое решение, состоящее в

использовании 8-разрядного *кольцевого счетчика*, на выходах которого непосредственно вырабатываются декодированные сигналы без паразитных импульсов, как это будет показано в разделе 6.2.5.

## 6.2. Регистры сдвига

### 6.2.1. Структура регистра сдвига

*Регистром* называют последовательностное логическое устройство, используемое для хранения  $n$ -разрядных двоичных чисел и выполнения преобразований над ними. Регистр представляет собой упорядоченную последовательность триггеров, число которых соответствует числу разрядов в слове. С каждым регистром обычно связано комбинационное цифровое устройство, с помощью которого обеспечивается выполнение некоторых операций над словами.

*Регистр сдвига* – это  $n$ -разрядный регистр, содержимое которого можно сдвигать на один разряд на каждом такте. На рис. 6.17 показана структура регистра сдвига с *последовательным вводом* и *последовательным выводом*. Последовательный входной сигнал SERIN – это новый бит, который *вдвигается* с одного конца регистра на данном такте. Этот бит появляется на последовательном выходе SEROUT спустя  $n$  тактов и теряется на следующем такте. Таким образом,  $n$ -разрядный регистр с последовательным вводом и последовательным выводом можно использовать для задержки сигнала на  $n$  тактов.

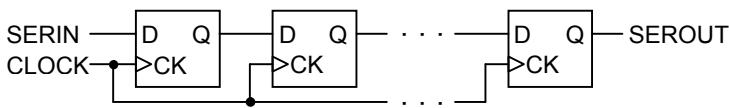


Рис. 6.17. Регистр сдвига с последовательным вводом и последовательным выводом

У *регистра сдвига с последовательным вводом и параллельным выводом*, приведенного на рис. 6.18, имеются выходы для всех хранимых в нем битов, благодаря чему они доступны

для других схем. Таким регистром можно воспользоваться для выполнения преобразования последовательного кода в параллельный.

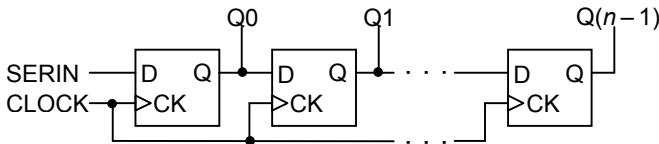


Рис. 6.18. Регистр сдвига с последовательным вводом и параллельным выводом

Можно построить *регистр сдвига с параллельным вводом и последовательным выводом*. На рис. 6.19 представлена схема такого устройства. В зависимости от значения сигнала на управляющем входе LOAD/SHIFT на каждом такте происходит либо загрузка новых данных с входов 1D...ND, либо сдвиг уже имеющегося содержимого регистра. В первый триггер регистра сдвига при этом записывается информация, присутствующая на входе SERIN. В этом устройстве на D-входе каждого триггера стоит 2-ходовой мультиплексор, позволяющий выбирать тот или иной сигнал. С помощью регистра сдвига с параллельным вводом и последовательным выводом можно осуществить *преобразование параллельного кода в последовательный*.

Если в регистре сдвига с параллельным вводом добавить выводы для всех сохраняемых в нем битов, то получим показанный на рис. 6.20 *регистр сдвига с параллельным вводом и параллельным выводом*. Такое устройство обладает возможностями всех упомянутых ранее регистров сдвига.

Рассмотренные регистры сдвига позволяют сдвигать информацию только в одном направлении – от младших разрядов к старшим. В тех случаях, когда требуется сдвигать информацию как *вправо*, так и *влево*, применяются реверсивные регистры сдвига. В этих регистрах информация в соответствующие разряды записывается либо из предыдущего разряда, либо из последующего, что обеспечивается мультиплексорами, включенными на входе каждого разряда регистра сдвига.

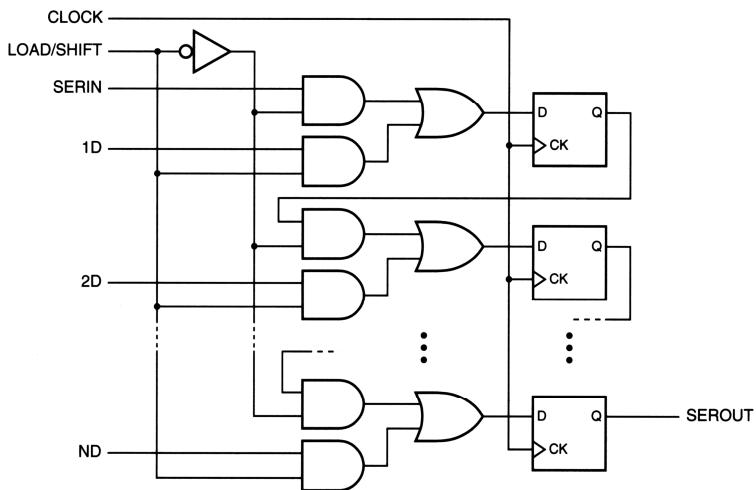


Рис. 6.19. Регистр сдвига с параллельным вводом и последовательным выводом

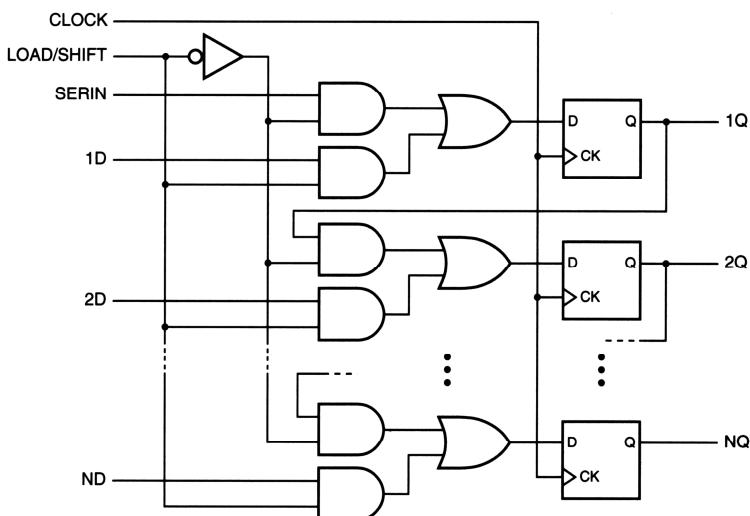


Рис. 6.20. Структура регистра сдвига с параллельным вводом и параллельным выводом

### 6.2.2. Регистры сдвига в ИС средней степени интеграции

На рис. 6.21 приведены условные обозначения трех популярных 8-разрядных регистров сдвига, выполненных в виде ИС средней степени интеграции. ИС 74 × 164 – это устройство с последовательным вводом и параллельным выводом, у которого имеется также асинхронный вход сброса  $\overline{CLR}$ . У этого регистра имеется два последовательных входа, объединенных внутри ИС по правилу логического И. Другими словами, для записи единицы в первый разряд регистра необходимо, чтобы единичные значения были у обоих входных сигналов SERA и SERB.

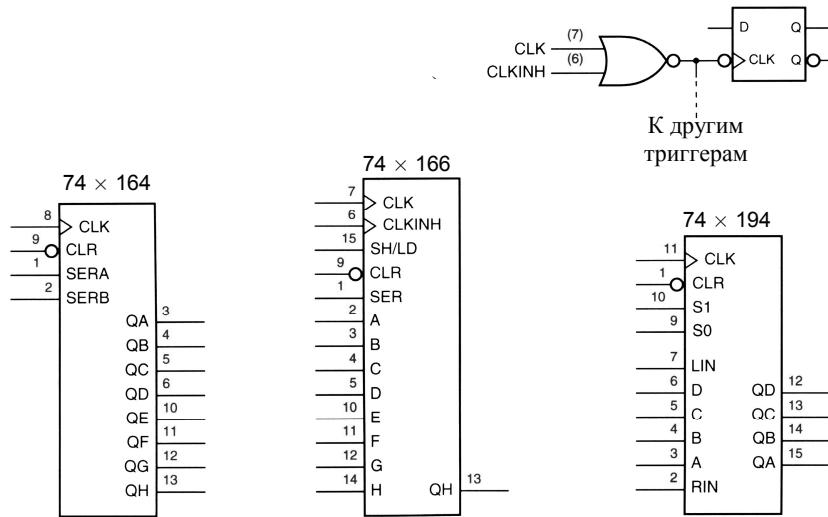


Рис. 6.21. Условные обозначения регистров сдвига: а) 8-разрядный регистр сдвига с последовательным вводом и параллельным выводом 74 × 164; б) 8-разрядный регистр сдвига с параллельным вводом и последовательным выводом 74 × 166; в) эквивалентная схема входной цепи для тактового сигнала в ИС 74 × 166; г) универсальный регистр сдвига 74 × 194

У регистра сдвига с параллельным вводом и последовательным выводом  $74 \times 166$  также есть асинхронный вход сброса. Сдвиг в этом устройстве происходит в том случае, когда входной сигнал **SH/LD** равен 1; в противном случае загружаются новые данные. В ИС '166 необычной является схема обработки тактового сигнала, который называют в этом случае стробируемым тактовым сигналом: имеются два тактовых входа, подключенных к триггерам внутри ИС (рис. 6.21с). Создатели ИС '166 имели в виду, что на вход **CLK** будет поступать сигнал от источника тактового сигнала, работающего в непрерывном режиме, а на вход **CLKINH** будет подаваться сигнал запрета **CLK**, чтобы в пределах отрезка времени, пока действует сигнал **CLKINH**, не происходили ни сдвиг, ни загрузка, то есть текущее содержимое регистра сохранялось. Но для того, чтобы схема работала именно так, сигнал **CLKINH** должен изменяться только в те моменты времени, когда **CLK** равен 1; в противном случае на тактовых входах триггеров внутри ИС будут возникать нежелательные перепады сигнала.

Табл. 6.2 представляет собой функциональное описание ИС  $74 \times 194$  в сжатом виде: в ней отсутствуют столбцы, соответствующие большинству входов (**A...D**, **RIN**, **LIN**) и текущему состоянию (**QA...QD**).

Таблица 6.2

**Функциональное описание 4-разрядного универсального регистра сдвига  $74 \times 19$**

Функция	Входы		Следующее состояние			
	S1	S0	$QA_{n+1}$	$QB_{n+1}$	$QC_{n+1}$	$QD_{n+1}$
Хранение	0	0	$QA_n$	$QB_n$	$QC_n$	$QD_n$
Сдвиг вправо	0	1	$RIN$	$QA_n$	$QB_n$	$QC_n$
Сдвиг влево	1	0	$QB_n$	$QC_n$	$QD_n$	$LIN$
Загрузка	1	1	A	B	C	D

ИС '194 иногда называют *универсальным* или *реверсивным* регистром сдвига, так как его можно заставить вести себя как любой из регистров сдвига менее общего типа, которые мы рас-

сматривали до сих пор (например, как односторонний регистр сдвига, как регистр сдвига с последовательным вводом и параллельным выводом или как регистр сдвига с параллельным вводом и последовательным выводом). Поэтому во многих примерах в дальнейшем фигурирует ИС '194, включенная таким образом, что используется подмножество функций из числа тех, которые способна выполнять эта микросхема.

### 6.2.3. Последовательно-параллельное преобразование

На рис. 6.22 приведен типичный пример последовательной передачи данных между двумя модулями.

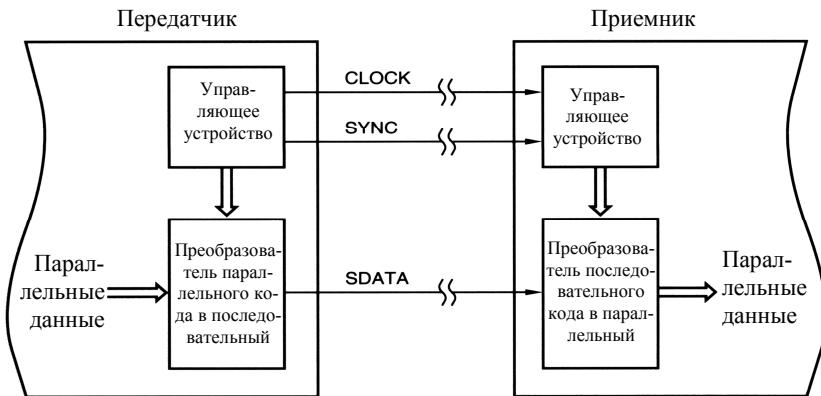


Рис. 6.22. Система с последовательной передачей данных между модулями

Обычно передача в таком соединении от источника сигналов к месту назначения происходит по трем сигнальным линиям:

- **Тактовый сигнал** задает темп передачи, указывая интервалы времени, отводимые на передачу одного бита. Если система состоит всего лишь из двух модулей, то тактовый сигнал может генерироваться блоком управления, размещенным в передатчике сигнала, как показано на рисунке. В более крупных системах может быть один общий источник тактового сигнала, разводимого по модулям.

- *Последовательные данные*: данные передаются побитно по одной линии.

- *Синхронизация*. Импульсом синхронизации указывается точка отсчета в формате данных, например, начало байта или слова в последовательном потоке данных. В некоторых системах этот сигнал бывает опущен, а синхронизация достигается передачей по линии данных последовательности специального вида.

#### 6.2.4. Счетчики на регистрах сдвига

Последовательно-параллельное преобразование представляет собой обработку данных, но регистры сдвига применяются также и в тех случаях, когда речь не идет о данных. В результате объединения регистра сдвига с комбинационной логикой образуется конечный автомат, у которого диаграмма состояний является циклической. Такую схему называют *счетчиком на регистре сдвига*. В отличие от двоичного счетчика последовательность состояний счетчика на регистре сдвига не образует ряд двоичных чисел, перебираемых в сторону увеличения или уменьшения, но такая схема все же полезна во многих приложениях, связанных с управлением.

#### 6.2.5. Кольцевые счетчики

В простейшем случае, используя  $n$ -разрядный регистр сдвига, можно получить счетчик с  $n$  состояниями, называемый *кольцевым счетчиком*. На рис. 6.23 показана схема кольцевого счетчика.

Универсальный регистр сдвига 74 × 194 включен так, что в нем происходит сдвиг влево (на вход S1 подана логическая 1). Но если подать сигнал RESET, то в него загружается комбинация 0001 [см. функциональную таблицу ИС '194 (табл. 6.2)]. Если же сигнал RESET снят, то на каждом такте происходит сдвиг содержимого ИС '194 влево. Последовательный вход LIN соединен с *крайним левым* выходом, так что последовательность состояний имеет вид: 0010, 0100, 1000, 0001, 0010, ... . Следовательно, счетчик проходит через четыре различных состояния, прежде чем они

начинают повторяться. На рис. 6.24 приведены соответствующие временные диаграммы. В общем случае  $n$ -разрядный кольцевой счетчик проходит в цикле через  $n$  состояний.

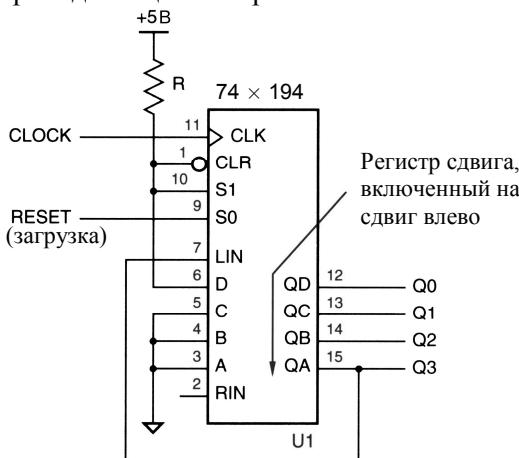


Рис. 6.23. Простейший 4-разрядный кольцевой счетчик с 4 состояниями, в котором циркулирует одна 1

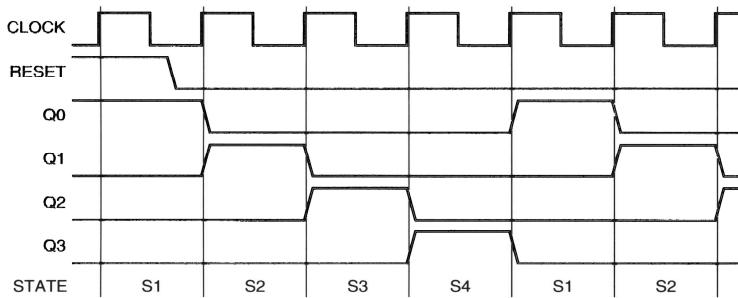


Рис. 6.24. Временные диаграммы работы 4-разрядного кольцевого счетчика

У кольцевого счетчика на рис. 6.23 есть серьезный недостаток: он не надежен. Если циркулирующая в нем единственная 1 будет потеряна вследствие временной аппаратной неисправности

(например, из-за шумов), то счетчик войдет в состояние 0000 и останется в нем навсегда. Точно так же возникновение лишней 1 (например, переход в состояние 0101) вынудит счетчик в течение всего времени в дальнейшем проходить по неправильному циклу. Эти проблемы становятся очевидными, если начертить *полную* диаграмму состояний такого счетчика, число которых равно 16. Как видно из рис. 6.25, 12 состояний не являются частью нормального цикла работы этой схемы. Если счетчик почему-либо выйдет из нормального цикла, то он уже не вернется в него.

*Самокорректирующийся счетчик* построен так, что из всех неправильных состояний имеются переходы, приводящие в нормальные состояния. Самокорректирующиеся счетчики целесообразно применять по следующей причине: если происходит что-то неожиданное, то счетчик или конечный автомат должен попадать в *безопасное* состояние.

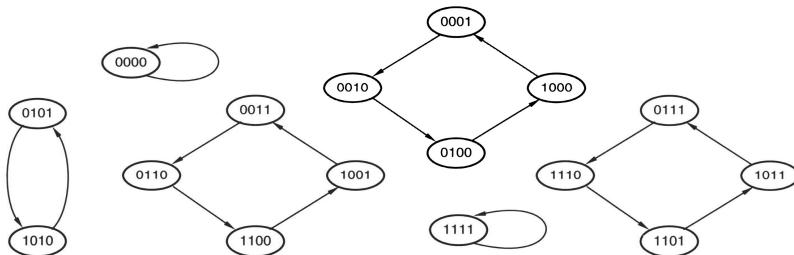


Рис. 6.25. Диаграмма состояний простого кольцевого счетчика

На рис. 6.26 показана схема *самокорректирующегося кольцевого счетчика*. В ней с помощью вентиля ИЛИ-НЕ единица возникает на входе LIN только в том случае, когда содержимое трех младших разрядов равно 0.

В результате получаем диаграмму состояний, приведенную на рис. 6.27; из всех неправильных состояний схема возвращается в нормальный цикл. В этой схеме нет необходимости в подаче сигнала RESET. Независимо от начального состояния, в которое счетчик попадает при включении, он окажется в состоянии 0001 в пределах первых четырех тактов. Поэтому принудительный

брос требуетсѧ только в том случае, если нужно, чтобы счетчик начинал правильно работать синхронно с другими узлами в системе, или для задания известной начальной точки при моделировании.

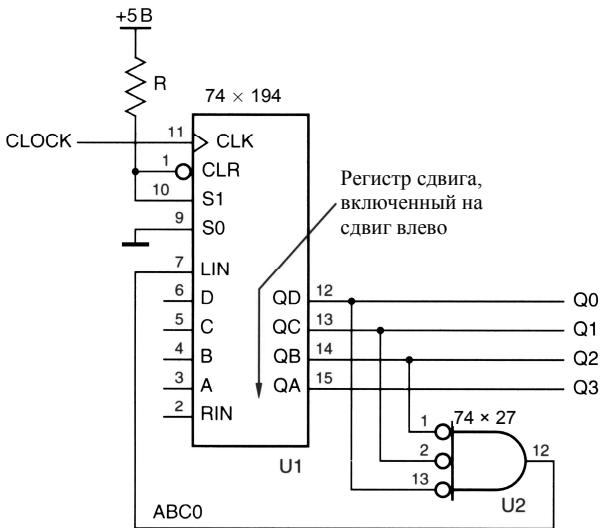


Рис. 6.26. Самокорректирующийся 4-разрядный кольцевой счетчик с 4 состояниями, в котором циркулирует одна 1

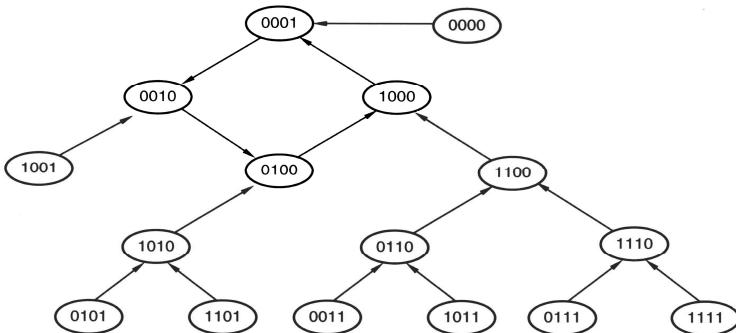


Рис. 6.27. Диаграмма состояний для самокорректирующегося кольцевого счетчика

В общем случае для построения  $n$ -разрядного самокорректирующегося кольцевого счетчика потребуется  $(n - 1)$ -входовой вентиль ИЛИ-НЕ, и такая схема входит в нормальное состояние не позднее, чем через  $n - 1$  тактов.

Основное достоинство кольцевого счетчика с точки зрения его применения в задачах управления состоит в том, что его состояния, выражаемые совокупностью сигналов на выходах триггеров, являются словами кода 1 из  $n$ . Это значит, что всегда только один из выходных сигналов триггеров имеет активный уровень. Кроме того, в выходных сигналах кольцевых счетчиков нет паразитных импульсов; сравните этот подход со случаем, когда двоичный счетчик дополняется дешифратором (рис. 6.14).

### 6.2.6. Счетчики Джонсона

У  $n$ -разрядного регистра сдвига с инвертором в цепи обратной связи между последовательным выходом и последовательным входом имеется  $2n$  состояний. Такая конструкция носит название *счетчика Джонсона*. Основная схема счетчика Джонсона показана на рис. 6.28, а временные диаграммы, иллюстрирующие его работу, приведены на рис. 6.29.

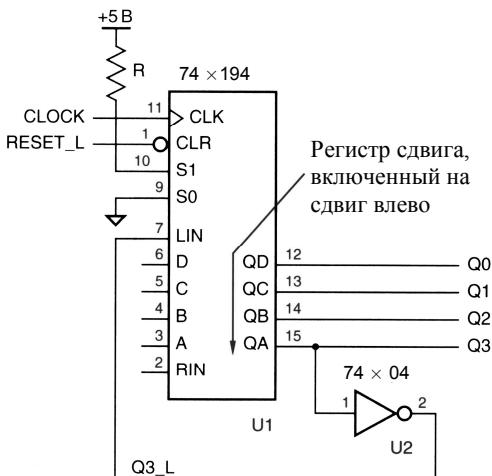


Рис. 6.28. Схема 4-разрядного счетчика Джонсона с 8 состояниями

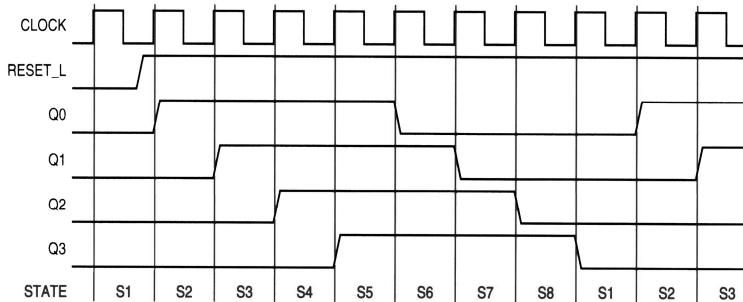


Рис. 6.29. Временные диаграммы работы 4-разрядного счетчика Джонсона

### 6.2.7. Генераторы псевдослучайных последовательностей

Число нормальных состояний у рассматривавшихся до сих пор схем на  $n$ -разрядных регистрах сдвига было далеко от максимально возможного числа состояний, равного  $2^n$ . Схема на основе  $n$ -разрядного *регистра сдвигом с линейной обратной связью* может иметь  $2^n - 1$  состояний, то есть почти максимум. Такую схему часто называют *генератором последовательности максимальной длины* или *генератором псевдослучайной последовательности* (*генератор ПСП*).

Генераторы ПСП строятся на основе теории *конечных полей*, развитой французским математиком Эваристом Галуа (1811–1832). В работе генератора реализуются операции над  $2^n$  элементами в конечном поле.

На рис. 6.30 представлена структура  $n$ -разрядного генератора ПСП. На последовательный вход регистра сдвига поступает сумма по модулю 2 битов, содержащихся в определенном наборе разрядов регистра сдвига. Этой обратной связью определяется последовательность состояний, через которые проходит счетчик. Принято всегда нумеровать разряды так, как показано на рисунке, и считать, что сдвиг происходит в указанном направлении.

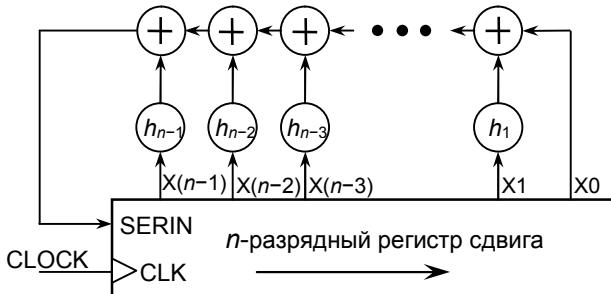


Рис. 6. 30. Общая структура генератора ПСП

При сдвиге содержимого регистра вправо в  $(n - 1)$ -й разряд заносится линейная комбинация (взвешенная сумма по модулю 2) значений логических переменных, которые хранились в разрядах регистра на предыдущем шаге. Значения двоичных весовых коэффициентов  $h_i$ ,  $i = n - 1, \dots, 1$  определяют, содержимое каких разрядов учитывается (при  $h_i = 1$ ) при формировании сигнала обратной связи, а содержимое каких разрядов игнорируется (при  $h_i = 0$ ). Если в какой-либо момент времени хотя бы один из разрядов регистра сдвига содержит 1, то при поступлении сигналов CLOCK на выходе регистра X0 (а также на любом другом) появится последовательность двоичных символов, отличная от нулевой.

Максимальный возможный период последовательности, генерируемой  $n$ -разрядным регистром с линейной обратной связью, равен  $2^n - 1$ : по мере генерирования выходной последовательности регистр переходит из одного состояния в другое, и период оказывается максимальным, когда последовательно перебираются без повторений все возможные ненулевые состояния регистра, полное число которых равно  $2^n - 1$ .

Согласно теории конечных полей Галуа, период последовательности, генерируемой регистром, максимальен и равен  $2^n - 1$  в том и только в том случае, когда многочлен  $h(x) = x^n + h_{n-1} \cdot x^{n-1} + \dots + h_1 \cdot x + 1$ , коэффициенты которого определяют конфигурацию цепи обратной связи, является неприво-

димым и примитивным. Неприводимость означает, что  $h(x)$  не делится ни на какой многочлен степени меньшей  $n$ , кроме многочлена 0-й степени, аналогично тому, как простое число не имеет делителей, кроме 1. Согласно второму условию, многочлен  $h(x)$  не должен быть делителем многочлена вида  $x^m + 1$  ни при каком  $m < 2^n - 1$ .

В табл. 6.3 для ряда значений  $n$  приведены уравнения, описывающие цепь обратной связи в тех случаях, когда результирующая последовательность оказывается последовательностью максимальной длины. Для каждого значения  $n$  больше 3-х существуют и другие уравнения обратной связи, обеспечивающие генерирование последовательностей максимальной длины, причем различным уравнениям соответствуют разные последовательности.

Т а б л и ц а 6.3

**Уравнения обратной связи для генераторов ПСП**

<i>n</i>	Уравнение обратной связи
2	$X_2 = X_1 \oplus X_0$
3	$X_3 = X_1 \oplus X_0$
4	$X_4 = X_1 \oplus X_0$
5	$X_5 = X_2 \oplus X_0$
6	$X_6 = X_1 \oplus X_0$
7	$X_7 = X_3 \oplus X_0$
8	$X_8 = X_4 \oplus X_3 \oplus X_2 \oplus X_0$
12	$X_{12} = X_6 \oplus X_4 \oplus X_1 \oplus X_0$
16	$X_{16} = X_5 \oplus X_4 \oplus X_3 \oplus X_0$
20	$X_{20} = X_3 \oplus X_0$
24	$X_{24} = X_7 \oplus X_2 \oplus X_1 \oplus X_0$
28	$X_{28} = X_3 \oplus X_0$
32	$X_{32} = X_{22} \oplus X_2 \oplus X_1 \oplus X_0$

Генератор ПСП со структурой, указанной на рис. 6.30 или рис. 6.31, никогда не проходит в цикле через все возможные  $2^n$  состояний. Независимо от конфигурации соединений следующим

состоянием за тем, при котором во всех разрядах находятся нули, является то же самое состояние с нулями во всех разрядах.

На рис. 6.32 (без заштрихованных вентиляй) показана принципиальная схема 3-разрядного генератора ПСП. Последовательность состояний этого генератора приведена в левых трех столбцах табл. 6.4. Начиная с любого ненулевого состояния (в таблице – с состояния 100), генератор проходит через семь состояний, прежде чем он возвращается в исходное состояние.

Схему генератора ПСП можно видоизменить так, чтобы у него было  $2^n$  состояния, включая состояние со всеми нулями; в схеме 3-разрядного генератора ПСП, приведенной на рис. 6.32, в заштрихованном прямоугольнике показано, как это сделать. В результате последовательность состояний будет такой, какая указана в трех правых столбцах табл. 6.4. То же самое можно сделать и в случае  $n$ -разрядного генератора ПСП: для этого необходимы вентиль ИСКЛЮЧАЮЩЕЕ ИЛИ и вентиль ИЛИ-НЕ с  $n - 1$  входами, которые должны быть подключены к выходам регистра сдвига, за исключением выхода  $X_0$ .

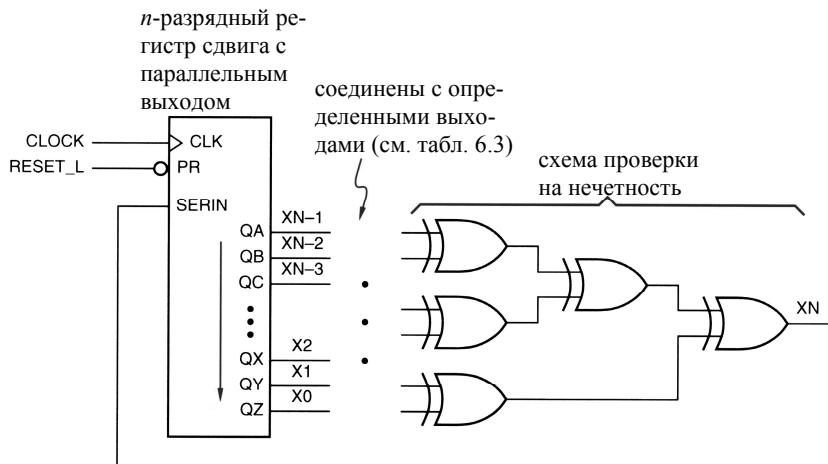


Рис. 6.31. Схема генератора ПСП

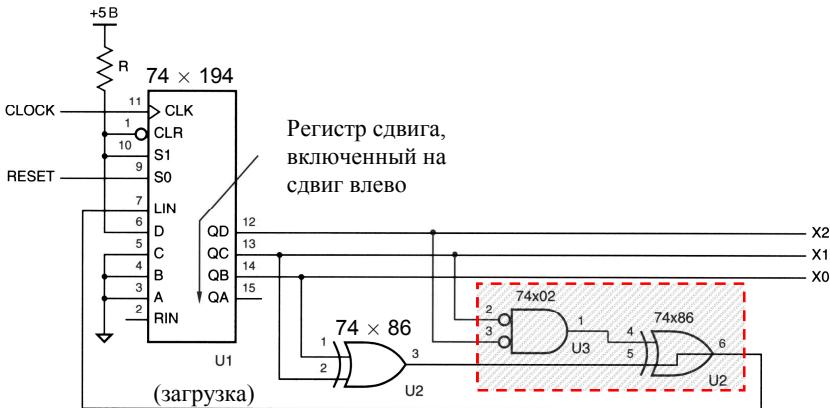


Рис. 6.32. 3-разрядный генератор ПСП, имеющий период  $2^3$

Таблица 6.4

Последовательность состояний 3-разрядного генератора ПСП, приведенного на рис. 6.32

Исходная последовательность			Измененная последовательность		
X2	X1	X0	X2	X1	X0
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1
0	1	1	0	1	1
0	0	1	0	0	1
1	0	0	0	0	0
.	.	.	1	0	0
.	.	.	.	.	.

Генератор ПСП переходит из одного состояния в другое не в порядке двоичного счета. Но во многих приложениях именно это свойство генераторов ПСП и является их достоинством. Основное применение генераторов ПСП состоит в генерировании тестовых входных сигналов для логических схем. В большинстве случаев *псевдослучайная* последовательность комбинаций, выда-

ваемых таким генератором, предпочтительнее с точки зрения обнаружения ошибок, нежели последовательность комбинаций, перебираемых в порядке двоичного счета. При передаче данных генератор ПСП часто является одним из узлов высокоскоростных модемов и сетевых интерфейсов, с помощью которых осуществляется *скремблирование* и *дескремблирование* данных. Достигается это путем пропускания выходного сигнала генератора ПСП и потока данных через элемент **ИСКЛЮЧАЮЩЕЕ ИЛИ**. Даже в том случае, когда в данных имеются длинные последовательности нулей и единиц, смешивание их с псевдослучайным выходным сигналом генератора ПСП улучшает баланс передаваемого сигнала по постоянному току (среднее значение становится близко к нулю при передаче нулей и единиц напряжениями  $-U$  и  $+U$ ) и создает достаточно большое число переходов, облегчающих извлечение приемником информации о тактовом сигнале.

# ГЛАВА 7. ДИСКРЕТИЗАЦИЯ АНАЛОГОВЫХ СИГНАЛОВ

## 7.1. Аналоговые, дискретные и цифровые сигналы

Исходный физический сигнал, как правило, является непрерывной функцией времени. Такие сигналы, определенные во все моменты времени, называют аналоговыми. Последовательность чисел, представляющая сигнал при цифровой обработке, является дискретным рядом и не может полностью соответствовать аналоговому сигналу. Числа, составляющие последовательность, являются значениями сигнала в отдельные (дискретные) моменты времени и называются отсчетами сигнала. Как правило, отсчеты берутся через равные промежутки времени  $T$ , называемые периодом дискретизации (или интервалом, шагом дискретизации). Величина, обратная периоду дискретизации, называется *частотой дискретизации*:  $f_d = 1/T$ . Соответствующая ей круговая частота определяется следующим образом:  $\omega_d = 2\pi/T$ .

В общем случае представление сигнала набором дискретных отсчетов приводит к потере информации, так как мы не имеем сведений о поведении сигнала в промежутках между отсчетами. Тем не менее, как будет показано далее, имеется класс аналоговых сигналов, для которых не происходит потери информации и которые можно *точно* восстановить по значениям своих дискретных отсчетов.

Процесс преобразования аналогового сигнала в последовательность отсчетов называется *дискретизацией*, а результат такого преобразования – *дискретным сигналом*.

При цифровой обработке сигнала его отсчеты представляются в виде двоичных чисел, имеющих ограниченное число разрядов. Вследствие этого отсчеты могут принимать лишь конечное множество значений и, следовательно, при представлении сигнала неизбежно происходит его округление. Процесс преобразования отсчетов сигнала в числа называется *квантованием по уровню*. Возникающие при этом ошибки округления называются *ошибками* (или *шумами*) *квантования*.

Сигнал, дискретный во времени, но не квантованный по уровню, называется *дискретным* сигналом, а сигнал, дискретный во времени и квантованный по уровню, называют *цифровым* сигналом. Разницу между аналоговыми, дискретными и цифровыми сигналами иллюстрирует рис. 7.1.



Рис. 7.1. Аналоговый (слева), дискретный (в центре) и цифровой (справа) сигналы

Вычислительные устройства, предназначенные для обработки сигналов, могут оперировать только цифровыми сигналами. Существуют также устройства, построенные в основном на базе аналоговой схемотехники, которые работают с дискретными сигналами, представленными в виде импульсов различной амплитуды или длительности. Чтобы подчеркнуть отсутствие квантования по уровню, такие устройства иногда называют *дискретно-аналоговыми* (ДАУ).

## 7.2. Аналого-цифровое и цифроаналоговое преобразование

Структура системы цифровой обработки сигналов приведена на рис. 7.2. На вход поступает аналоговый сигнал  $s_{in}(t)$ . Его временная дискретизация и квантование по уровню производятся в *аналого-цифровом преобразователе* (АЦП; *Analog-to-Digital Converter, ADC*). Вообще эти два процесса – дискретизация и квантование – являются независимыми друг от друга, но они, как правило, выполняются одним устройством. Выходным сигналом АЦП является последовательность чисел, поступающая в цифровой процессор, выполняющий требуемую обработку. Задачей

процессора является реализация некоторых математических операций над входными отсчетами; полученные на выходе отсчеты и промежуточные результаты могут сохраняться в памяти процессора для использования в последующих вычислениях. Результатом работы процессора является новая последовательность чисел, представляющих собой отсчеты выходного сигнала. Аналоговый выходной сигнал  $s_{out}(t)$  восстанавливается по этой последовательности чисел с помощью цифроаналогового преобразователя (ЦАП; *Digital-to-Analog Converter, DAC*). Напряжение на выходе ЦАП имеет ступенчатую форму (это показано на рис. 7.2); при необходимости оно может быть преобразовано в плавно меняющийся выходной сигнал с помощью сглаживающего фильтра  $\Phi$ .

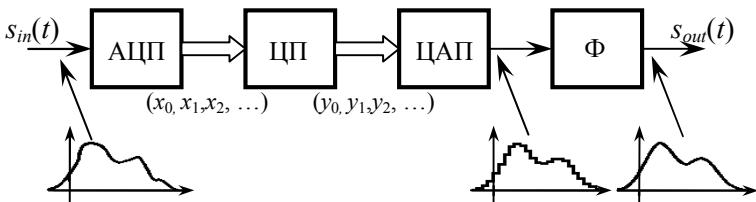


Рис. 7.2. Структурная схема системы цифровой обработки сигналов

Устройства со структурой, показанной на рис. 7.2, могут иметь самый разнообразный характер. В цифровой форме можно создавать фильтры, анализаторы спектра, нелинейные преобразователи сигналов и многое другое.

### 7.3. Частота Найквиста

Гармонический сигнал может быть адекватно представлен дискретными отсчетами в том случае, если его частота не превышает половины частоты дискретизации (этот частота называется *частотой Найквиста (Nyquist frequency) –*  $f_N = f_d / 2 = 1/(2T)$ ;  $\omega_N = \omega_d / 2 = \pi/T$ ). Происхождение этого ограничения поясняет рис. 7.3. В зависимости от соотношения между частотой дискрети-

зируемого гармонического сигнала и частотой Найквиста возможны три случая.

1. Если частота гармонического сигнала *меньше* частоты Найквиста, дискретные отсчеты позволяют правильно восстановить аналоговый сигнал (рис. 7.3а).
2. Если частота гармонического сигнала *равна* частоте Найквиста, то дискретные отсчеты позволяют восстановить аналоговый гармонический сигнал с той же частотой, но амплитуда и фаза восстановленного сигнала (он показан пунктирной линией) могут быть искажены (рис. 7.3б). В худшем случае все дискретные отсчеты синусоиды могут оказаться равными нулю.
3. Если частота гармонического сигнала *больше* частоты Найквиста, восстановленный по дискретным отсчетам аналоговый сигнал (как и в предыдущем случае, он показан пунктирной линией) будет также гармоническим, но с иной частотой (рис. 7.3с). Данный эффект носит название *появления ложных частот (aliasing)*.

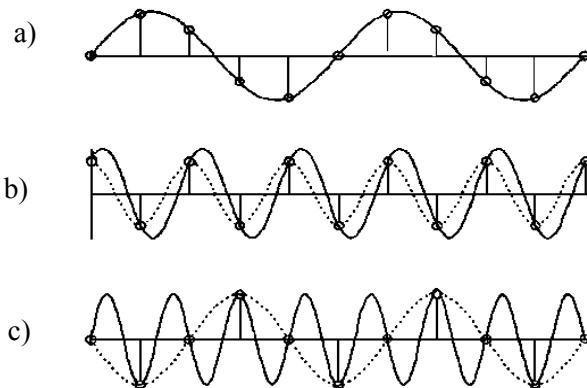


Рис. 7.3. Дискретизация гармонических сигналов с разной частотой

Эффекты, связанные с дискретизацией периодических процессов, наглядно проявляются при кино и видеосъемке вращаю-

ящихся объектов (таких, например, как спицы колеса автомобиля). Из-за недостаточно высокой частоты дискретизации (частоты смены кадров) быстро вращающееся колесо может выглядеть неподвижным либо медленно поворачивающимся, причем в любую сторону.

## 7.4. Спектр дискретного сигнала

Преобразование Фурье позволяет вычислить спектральную плотность сигнала, представляющего собой функцию (либо времени, либо пространственных координат). Дискретный же сигнал является последовательностью чисел, поэтому для анализа его спектра обычными (аналоговыми) средствами необходимо сопоставить этой последовательности некоторую функцию.

Традиционным способом такого сопоставления является представление отсчетов в виде дельта-функций с соответствующими множителями и задержками. Для последовательности отсчетов  $\{x(k)\}$  получится следующий сигнал:

$$s(t) = \sum_{k=-\infty}^{\infty} x(k) \delta(t - k). \quad (7.1)$$

Преобразование Фурье является линейным, спектр дельта-функции равен единице, а задержка сигнала во времени приводит к умножению спектра на комплексную экспоненту. Все это позволяет сразу же записать спектр дискретного сигнала:

$$\dot{S}(\omega) = \sum_{k=-\infty}^{\infty} x(k) e^{-j\omega k}. \quad (7.2)$$

Из этой формулы видно главное свойство спектра любого дискретного сигнала: спектр является периодическим, и его период в данном случае равен  $2\pi$  (то есть круговой частоте дискретизации, поскольку, составляя сигнал из дельта-функций, мы выбрали единичный интервал между ними, что дает  $\omega_d = 2\pi$ ):

$$\dot{S}(\omega \pm 2\pi) = \dot{S}(\omega).$$

Следует также обратить внимание на размерность спектральной функции дискретного сигнала: она совпадает с размерностью отсчетов. Это связано с тем, что дельта-функции времени,

из которых был составлен сигнал (7.1), имеют размерность частоты.

Формула (7.2) позволяет вычислить спектральную функцию по известным отсчетам  $x(k)$ . При конечном числе ненулевых отсчетов этот расчет несложен; он может быть выполнен с помощью функции MATLAB `freqz`.

Рассмотрим несколько иную задачу. Пусть значения  $x(k)$  являются отсчетами аналогового сигнала  $s(t)$ , взятыми с периодом  $T$ :

$$x(k) = s(kT).$$

Определим, как спектр дискретного сигнала (7.2) связан со спектром аналогового сигнала  $S(\omega)$ .

Мы рассматриваем дискретизированный сигнал в виде последовательности дельта-функций, *взвешенных* значениями отсчетов  $s(kT)$  аналогового сигнала  $s(t)$  (рис. 7.4):

$$s(t) = \sum_{k=-\infty}^{\infty} s(kT) \delta(t - kT). \quad (7.3)$$

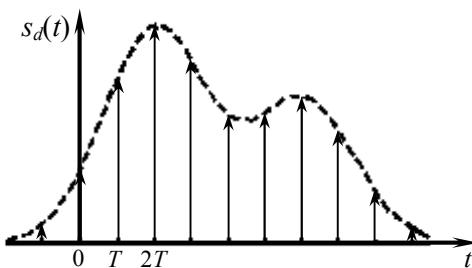


Рис. 7.4. Дискретизированный сигнал в виде последовательности дельта-функций

Термин *дискретизированный* в данном случае подчеркивает, что последовательность отсчетов получена именно в результате дискретизации аналогового сигнала.

Так как функция  $\delta(t - kT)$  равна нулю всюду, кроме момента  $t = kT$ , можно заменить в выражении (7.3) константы  $s(kT)$  на исходный непрерывный сигнал  $s(t)$ :

$$s_d(t) = s(t) \sum_{k=-\infty}^{\infty} \delta(t - kT). \quad (7.4)$$

Далее заметим, что сумма, входящая в выражение (7.4), является периодическим сигналом, а потому может быть представлена в виде ряда Фурье. Коэффициенты этого ряда равны

$$\dot{C}_n = \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) \cdot e^{-j\omega_n t} dt = \frac{1}{T}. \quad (7.5)$$

В формуле (7.5) было учтено, что в интервал интегрирования  $(-T/2, T/2)$  попадает только одна дельта-функция, соответствующая  $k = 0$ .

Таким образом, периодическая последовательность дельта-функций может быть представлена в виде комплексного ряда Фурье:

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j\omega_n t}, \quad (7.6)$$

где  $\omega_n = 2\pi n/T$ . Подставив (7.6) в (7.4), получим

$$s_d(t) = \frac{s(t)}{T} \sum_{n=-\infty}^{\infty} e^{j\omega_n t} = \frac{1}{T} \sum_{n=-\infty}^{\infty} s(t) e^{j\omega_n t}.$$

Умножение сигнала на  $\exp(j\omega_n t)$  соответствует сдвигу спектральной функции на  $\omega_n$ , поэтому спектр дискретизированного сигнала можно записать следующим образом:

$$\dot{S}_d(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \dot{S}\left(\omega - \frac{2\pi n}{T}\right). \quad (7.7)$$

Таким образом, спектр дискретизированного сигнала представляет собой бесконечный ряд сдвинутых копий спектра исходного непрерывного сигнала  $s(t)$  (рис. 7.5). Расстояние по частоте между соседними копиями спектра равно частоте дискретизации  $\omega_n = 2\pi n/T$ .

Следует также отметить, что из-за наличия в формуле (7.7) множителя  $1/T$  спектр дискретизированного сигнала имеет раз-

мерность, совпадающую с размерностью сигнала (как уже говорилось, это связано с тем, что функция  $\delta(t)$  имеет размерность частоты).

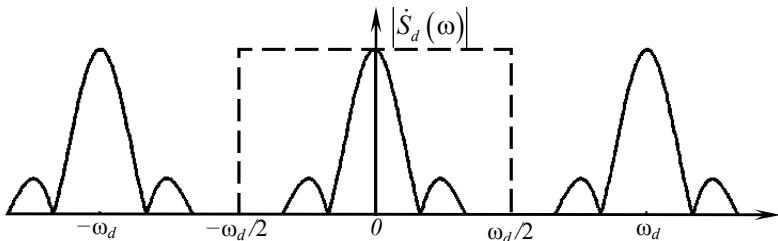


Рис. 7.5. Спектр дискретизированного сигнала

Характер спектра дискретизированного сигнала еще раз демонстрирует частотно-временную дуальность преобразования Фурье:

периодический сигнал  $\rightarrow$  дискретный спектр;

периодический спектр  $\rightarrow$  дискретный сигнал.

В начале данного раздела мы получили формулу (7.2), позволяющую рассчитать спектр последовательности отсчетов  $\{x(k)\}$ , никак не связывая эти отсчеты с аналоговым сигналом. Формула (7.7) предполагает, что отсчеты  $\{x(k)\}$  получены путем дискретизации аналогового сигнала  $s(t)$ , и показывает связь между спектрами дискретизированного и аналогового сигналов. Следует подчеркнуть, что эти две формулы дают одинаковый результат.

Отсюда следует еще один важный факт. Соединить отсчеты  $\{x(k)\}$  для получения аналогового сигнала можно произвольным образом. В каждом случае аналоговый сигнал будет, разумеется, иметь свой спектр. Однако результат суммирования сдвинутых копий спектров по формуле (7.7) всегда будет одним и тем же, поскольку определяется только значениями дискретных отсчетов  $\{x(k)\} = \{s(kT)\}$  и формулой (7.2).

Рис. 7.5 наглядно демонстрирует и способ восстановления непрерывного сигнала по дискретным отсчетам. Для этого необ-

ходимо пропустить дискретный сигнал через идеальный фильтр нижних частот (ФНЧ) с частотой среза, равной половине частоты дискретизации. АЧХ такого фильтра показана на рис. 7.5 пунктиром.

Очевидно, что точное восстановление сигнала возможно, если сдвинутые копии спектра не перекрываются. Из рис. 7.5 видно, что для этого необходимо, чтобы частота дискретизации как минимум в два раза превышала верхнюю граничную частоту в спектре сигнала:

$$\omega_d > 2\omega_0. \quad (7.8)$$

Спектральное представление дискретного сигнала позволяет объяснить появление ложных частот (*aliasing*), речь о которых шла в разделе 7.3. Пусть дискретизации подвергается гармонический сигнал с частотой  $\omega_0$ , превышающей частоту Найквиста, но меньшей частоты дискретизации. Спектр такого сигнала показан на рис. 7.6 сверху.

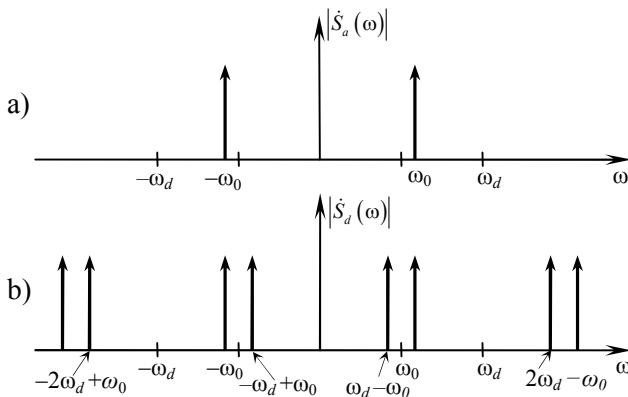


Рис. 7.6. Спектры аналоговой (а) и дискретизированной (б) синусоиды с частотой, превышающей частоту Найквиста

Сдвинутые копии спектра, возникающие при дискретизации, создают попадающие в полосу восстановления (от нуля до частоты Найквиста) спектральные составляющие с частотой  $\omega_d - \omega_0$  (рис. 7.6б). Спектры, получающиеся после дискретиза-

ции гармонических сигналов с частотами  $\omega_0$  и  $\omega_d - \omega_0$ , оказываются идентичными. Данный рисунок иллюстрирует в частотной области процесс дискретизации гармонического сигнала, показанный ранее на рис. 7.3.

В случае произвольного сигнала, если условие (7.8) не выполняется, сдвинутые копии спектра будут накладываться друг на друга, что приведет к неизбежным искажениям при восстановлении непрерывного сигнала (рис. 7.7).

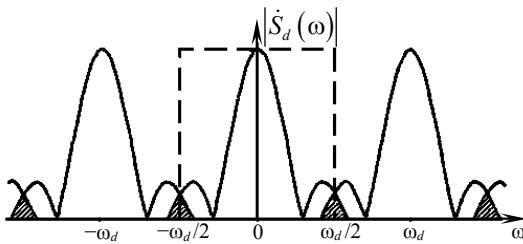


Рис. 7.7. Перекрытие сдвинутых копий спектра при недостаточно высокой частоте дискретизации

Эти искажения вызваны тем, что спектральные составляющие сигнала с частотами, превышающими частоту Найквиста, равную  $\omega_d/2$ , не могут быть восстановлены правильно – вместо этого они вызывают наложение *хвостов* соседних сдвинутых копий спектра и появление ложных частот.

Если подлежащий дискретизации сигнал может содержать спектральные составляющие с частотами, превышающими частоту Найквиста, полезно предварительно пропустить его через ФНЧ с частотой среза, равной частоте Найквиста (рис. 7.8). При этом все равно будут потеряны высокочастотные составляющие. Сохранить эти составляющие можно лишь путем повышения частоты дискретизации. Однако в этом случае благодаря отсутствию наложения *хвостов* не произойдет появления ложных частот и диапазон частот  $0 \dots \omega_d/2$  будет представлен в дискретном сигнале без искажений.

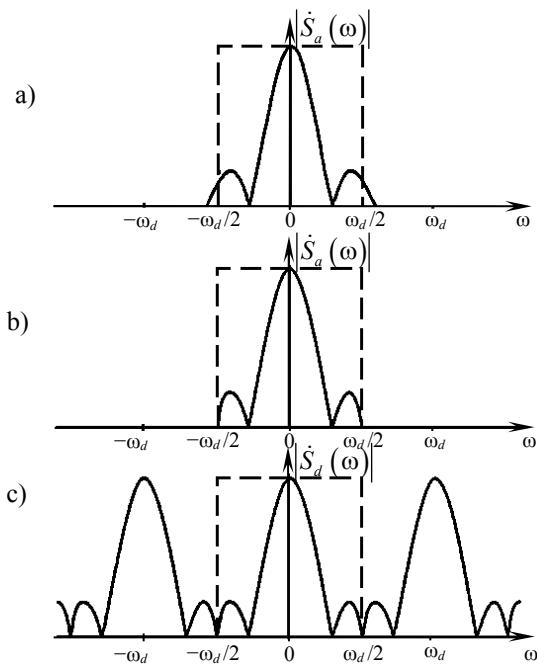


Рис. 7.8. При дискретизации сигнала, содержащего высокочастотные составляющие (а), полезно пропустить его через ФНЧ (б), чтобы избежать появления ложных частот (с)

## 7.5. Теорема Котельникова

В начале уже было сформулировано утверждение о том, что некоторые сигналы могут быть без потерь информации представлены своими дискретными отсчетами. Полученное в предыдущем разделе выражение для спектра дискретизированного сигнала позволяет выделить тот класс сигналов, для которых это возможно, и описать способ такого восстановления.

Согласно (7.7), спектр дискретизированного сигнала представляет собой сумму сдвинутых копий спектра исходного сигнала, при этом шаг сдвига равен частоте дискретизации  $\omega_d$ . На рис. 7.5 видно, что если в спектре аналогового сигнала не содер-

жится составляющих с частотами, превышающими частоту Найквиста ( $\omega_d/2$ ), то сдвинутые копии спектра не будут перекрываться. В этом случае использование идеального ФНЧ с прямоугольной АЧХ позволит выделить исходную (несдвинутую) копию спектра, сосредоточенную в окрестностях нулевой частоты, и, таким образом, в точности восстановить исходный аналоговый сигнал.

АЧХ идеального ФНЧ, восстанавливающего аналоговый сигнал, приведена на рис. 7.9 а). Коэффициент передачи в полосе пропускания равен  $T$ , а не единице, чтобы компенсировать множитель  $1/T$  в формуле (7.7). С помощью обратного преобразования Фурье находится импульсная характеристика фильтра (рис. 7.9б):

$$h(t) = \frac{\sin\left(\pi \frac{t}{T}\right)}{\pi \frac{t}{T}}. \quad (7.11)$$

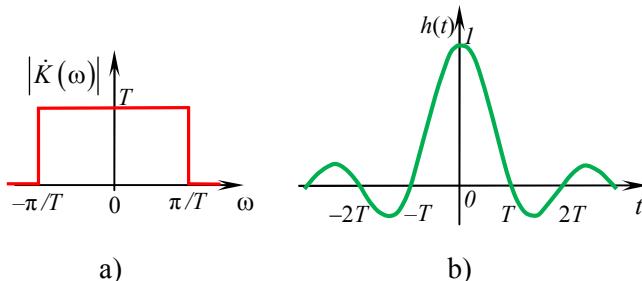


Рис. 7.9. Амплитудно-частотная (а) и импульсная (б) характеристики идеального восстанавливающего фильтра

Дискретизированный сигнал (7.3) представляет собой сумму дельта-функций. При прохождении такого сигнала через восстанавливающий ФНЧ каждая дельта-функция породит на выходе соответствующим образом сдвинутую и масштабированную копию импульсной характеристики фильтра. Выходной сигнал (в точности соответствующий исходному аналоговому сигналу), таким образом, будет представлять собой сумму сдвинутых и ум-

ноженных на отсчеты сигнала копий импульсных характеристик идеального ФНЧ (7.11):

$$s(t) = \sum_{k=-\infty}^{\infty} s(kT) \frac{\sin\left(\pi \frac{t-kT}{T}\right)}{\pi \frac{t-kT}{T}}. \quad (7.12)$$

Подводя итог сказанному, сформулируем *теорему Котельникова: любой сигнал  $s(t)$ , спектр которого не содержит составляющих с частотами выше некоторого значения  $\omega_h = 2\pi f_h$ , может быть без потерь информации представлен своими дискретными отсчетами  $\{s(kT)\}$  взятыми с интервалом  $T$ , удовлетворяющим следующему неравенству:*

$$T \leq \frac{1}{2f_h} = \frac{\pi}{\omega_h}. \quad (7.13)$$

Восстановление исходного непрерывного сигнала  $s(t)$  по набору его дискретных отсчетов  $\{s(kT)\}$  производится согласно формуле (7.12).

В зарубежных источниках данная теорема называется теоремой Найквиста, теоремой Шеннона или теоремой о выборках (*sampling theorem*).

Формула (7.12) представляет собой разложение сигнала  $s(t)$  в ряд по системе функций  $\{\varphi_k(t)\}$ , называемой базисом Котельникова:

$$\varphi_k(t) = \frac{\sin\left(\pi \frac{t-kT}{T}\right)}{\pi \frac{t-kT}{T}}.$$

Формирование непрерывного сигнала по его дискретным отсчетам поясняет рис. 7.10. Пунктиром показаны графики отдельных слагаемых, входящих в формулу (7.12), сплошной линией – восстановленный сигнал.

Рисунок 7.10 наглядно демонстрирует главное свойство сигнала с ограниченным спектром – его бесконечность во времени. Хотя отличны от нуля лишь несколько отсчетов показанного сигнала, аналоговый сигнал оказывается бесконечно колеблю-

щимся – между нулевыми отсчетами (на рисунке это отсчеты с номерами  $-2, -1, 4, 5, 6$ ) его значения отличны от нуля. Эти колебания нигде не заканчиваются, хотя их амплитуда стремится к нулю.

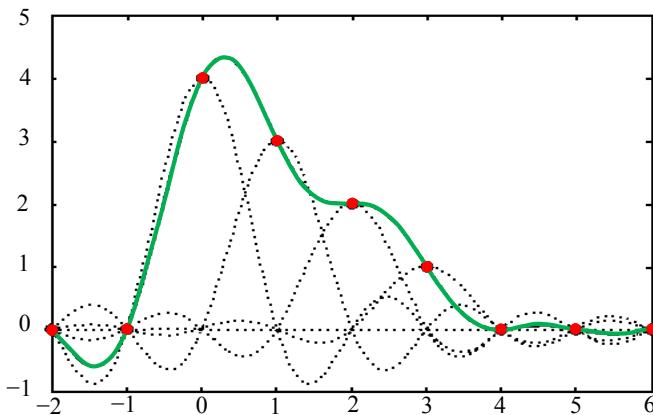


Рис. 7.10. Восстановление непрерывного сигнала по его дискретным отсчетам

Иногда можно встретить примерно следующее объяснение сущности теоремы Котельникова: если брать отсчеты достаточно часто, в промежутках между ними сигнал с ограниченными спектром не успеет сильно измениться, и мы сможем точно восстановить его. Такая трактовка является принципиально неправильной.

Когда говорят об ограниченной полосе частот сигнала, имеется в виду спектральная функция сигнала, имеющего бесконечную длительность. При этом мгновенные спектры отдельных фрагментов сигнала могут иметь сколь угодно высокие частоты.

Под мгновенным спектром подразумевается спектральная функция *вырезанного* из сигнала фрагмента конечной длительности.

# ГЛАВА 8. МЕТОДЫ ДИСКРЕТИЗАЦИИ СИГНАЛОВ

## 8.1. Цифроаналоговые преобразователи (ЦАП)

Наиболее распространенными преобразователями числа, представленного в том или ином коде, в напряжение или ток являются параллельные ЦАП; процесс преобразования в них происходит одновременно во всех разрядах с учетом их *веса*. Если на вход такого ЦАП подается число  $N_i$ , представленное в двоичном коде, то выходное напряжение ЦАП равно

$$U_{\text{ЦАП}} = \frac{N_i}{N_{\max}} U_{\max}, \quad (8.1)$$

где  $U_{\max}$  — максимальное значение выходного напряжения ЦАП, соответствующее максимальному значению числа  $N_{\max}$ .

Двоичная запись числа представляется в виде

$$N_i = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0, \quad (8.2)$$

где  $a_i = 0, 1$ .

С учетом того, что максимальное значение числа, содержащего  $n$  разрядов, равно  $N_{\max} = 2^n - 1$ , или, при большом  $n$ ,  $N_{\max} \approx 2^n$ , уравнение (8.1) можно преобразовать:

$$U_{\text{ЦАП}} \approx U_{\max} \left( a_{n-1} \cdot 2^{-1} + a_{n-2} \cdot 2^{-2} + \dots + a_1 \cdot 2^{-(n-1)} + a_0 \cdot 2^{-n} \right). \quad (8.3)$$

В этом случае выходной сигнал ЦАП состоит из  $n$  слагаемых, каждое из которых имеет свой вес или, точнее, коэффициент деления, определяемый номером данного разряда:

$$U_{\text{ЦАП}} = U_{\max} \sum_{i=0}^{n-1} a_i \cdot 2^{-(n-i)}. \quad (8.4)$$

Если число содержит знаковый разряд  $a_n$  в соответствии со значением которого должна меняться полярность выходного напряжения ЦАП, то

$$U_{\text{ЦАП}} = (-1)^{a_n} \cdot U_{\max} \sum_{i=0}^{n-1} a_i \cdot 2^{-(n-i)}. \quad (8.5)$$

Полученные общие формулы справедливы для всех типов ЦАП, однако они наиболее ясно отражены в схемах параллель-

ных ЦАП. Эти виды ЦАП имеют две основные разновидности: построенные на матрице с весовыми сопротивлениями и построенные на матрице с двумя номиналами сопротивлений, которую в дальнейшем будем называть матрицей  $R-2R$ .

В первом случае весовые сопротивления выбираются так, чтобы непосредственно реализовать весовые коэффициенты при слагаемых уравнения (8.3), то есть чтобы они были равны  $R \cdot 2^{n-i-1}$ . На рис. 8.1 приведена схема такого ЦАП. Входной код поступает на разрядные триггеры  $T_0, T_1, \dots, T_{n-1}$ , управляющие разрядами ЦАП. Так как входной код может существовать в течение сравнительно малого отрезка времени, то триггеры необходимы для запоминания и хранения входного кода на время преобразования или использования выходного напряжения ЦАП.

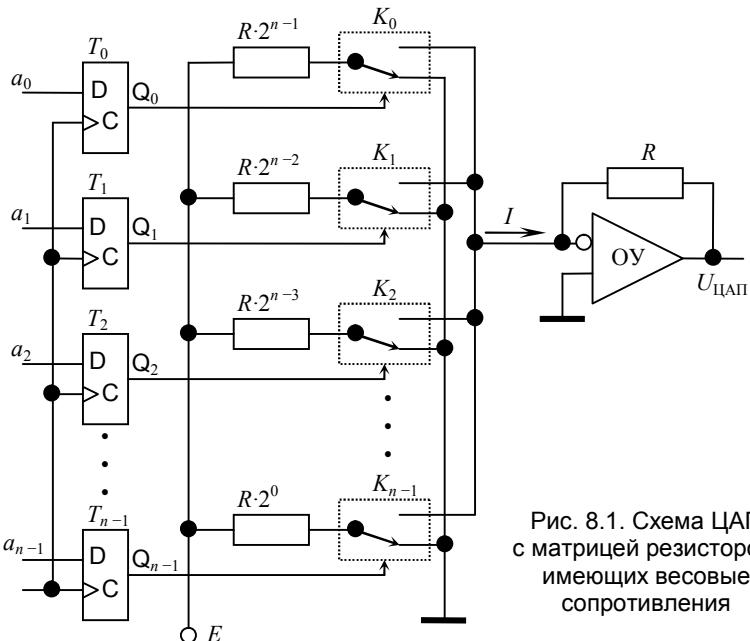


Рис. 8.1. Схема ЦАП с матрицей резисторов, имеющих весовые сопротивления

Триггеры управляют двухпозиционными ключами в разрядах ЦАП. Если в  $i$ -м разряде кода присутствует 1, то ключ  $K_i$  подсоединяет резистор  $R \cdot 2^{n-i-1}$  к входу операционного усилителя; ес-

ли же в данном разряде кода будет 0, то ключ подсоединит этот резистор к земле.

Если считать операционный усилитель идеальным, то при наличии отрицательной обратной связи входное сопротивление по инвертирующему входу равно нулю. Ключи в этом случае являются распределителями тока между почти эквипотенциальными точками. Это уменьшает влияние переходных процессов в резисторной схеме во время переключений, что приводит к повышению быстродействия ЦАП. Кроме того, в этом случае ток, потребляемый резисторной матрицей от источника опорного напряжения  $E$ , не зависит от значения числа. Этот факт положительно сказывается на стабильности источника опорного напряжения.

Ток, втекающий в усилитель (протекающий по резистору  $R$ ), определяется следующим выражением:

$$I = \sum_{i=0}^{n-1} \frac{E \cdot a_i}{R \cdot 2^{n-i-1}} = \frac{E}{R} \cdot \sum_{i=0}^{n-1} \frac{a_i}{2^{n-i-1}}, \quad (8.6)$$

а выходное напряжение ЦАП  $U_{ЦАП} = -IR$ , то есть

$$U_{ЦАП} = E \cdot \sum_{i=0}^{n-1} \frac{a_i}{2^{n-i-1}}. \quad (8.7)$$

Недостатком преобразователей с весовыми резисторами является необходимость подбора сопротивлений резисторов с различными номиналами; при этом номиналы сопротивлений в младшем и старшем разрядах отличаются в  $2^{n-1}$  раз и должны быть выдержаны с высокой точностью. Это создает определенные трудности, особенно при реализации ЦАП средствами интегральной технологии.

Схема ЦАП с матрицей типа  $R - 2R$  состоит из  $n$  одинаковых ветвей (рис. 8.2).

Здесь так же, как и в предыдущей схеме, ключи переключают токи. Несложно показать, что ток, протекающий по резистору  $2R$  в  $i$ -м разряде, равен

$$I_i = \frac{E \cdot a_i}{2R \cdot 2^{n-i-1}},$$

а полный ток

$$I = \sum_{i=0}^{n-1} \frac{E \cdot a_i}{2R \cdot 2^{n-i-1}}.$$

Отсюда следует, что напряжение на выходе ЦАП равно

$$U_{\text{ЦАП}} = \frac{E}{2} \cdot \sum_{i=0}^{n-1} \frac{a_i}{2^{n-i-1}}. \quad (8.8)$$

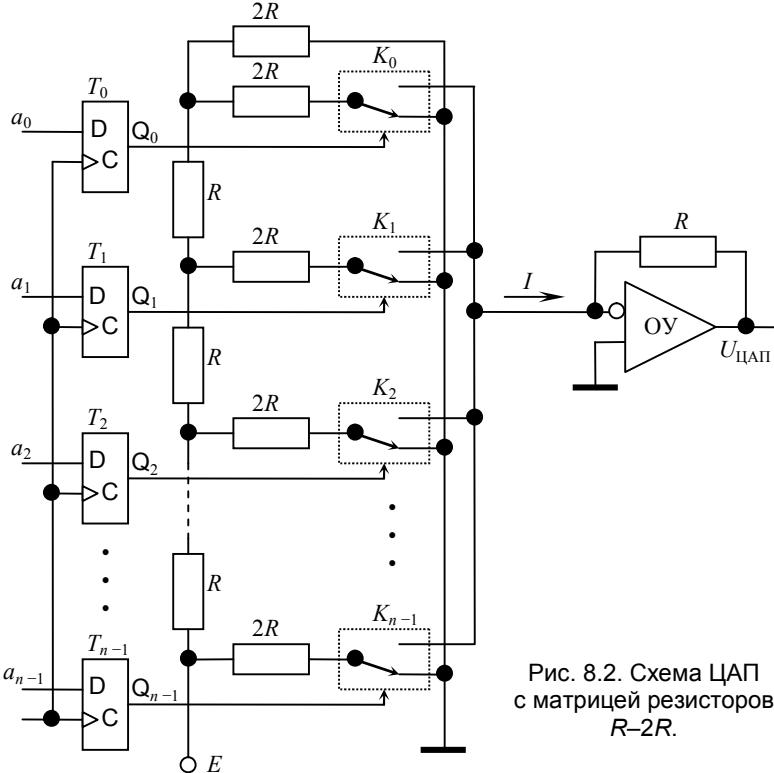


Рис. 8.2. Схема ЦАП с матрицей резисторов  $R-2R$ .

Большим преимуществом этого типа ЦАП является использование резисторов только двух номиналов, что сильно упрощает задачу создания матрицы резисторов в интегральном исполне-

нии. Именно по этой причине данный тип ЦАП стал в настоящее время основным.

Наиболее острыми проблемами, возникающими при создании таких ЦАП, являются обеспечение высоких точностей и скоростей преобразования. Если считать, что ОУ идеален и не вносит погрешностей в процесс преобразования, то, очевидно, что источниками погрешностей являются неточности в подборе сопротивлений и остаточные напряжения (или паразитные ЭДС) в ключах  $K_i$ .

Через резисторы  $2R$  матрицы текут соответствующие постоянные токи  $I_i$ , а ключи коммутируют эти токи между землей и входом усилителя. Так как для установившегося режима потенциал на входе ОУ близок к потенциальну земли, то отсюда следует, что быстродействие преобразователя будет зависеть не от постоянных времени в матрице резисторов, определяемой сопротивлениями резисторов и паразитными емкостями, а от полосы пропускания усилителя. Если ОУ имеет достаточно высокую максимальную скорость нарастания сигнала, то время преобразования определяется только временем переключения ключей в разрядах, что составляет доли микросекунды. При этом ток в матрице остается неизменным в силу близости потенциала на инвертирующем входе ОУ к потенциальну земли.

## 8.2. Аналого-цифровые преобразователи (АЦП)

Процесс аналого-цифрового преобразования состоит из многократного сравнения входного аналогового сигнала с набором эталонных сигналов. Основным признаком классификации АЦП является алгоритм его работы, который отражает набор операций, посредством которых устанавливается численное соответствие между аналоговой величиной, поданной на вход АЦП, и выбранными эталонными мерами. Наибольшее распространение получили три метода преобразования:

- метод последовательного счета;
- метод поразрядного уравновешивания;
- метод параллельного преобразования (метод считывания).

### **8.2.1. Аналого-цифровые преобразователи, построенные по методу последовательного счета**

Метод последовательного счета предполагает уравновешивание входной аналоговой величины суммой одинаковых минимальных эталонов, называемых *квантами*. Момент равенства входной величины и суммы эталонов определяется сравнивающим устройством (компаратором). Результат преобразования характеризуется числом квантов, использованных при преобразовании. Это число представлено в виде последовательности импульсов напряжения. С помощью двоичного счетчика результат преобразования переводится в двоичное число. При классическом исполнении этого метода необходим всего один эталон, равный кванту, причем этот эталон используется многократно при уравновешивании входной аналоговой величины.

**Последовательный АЦП со ступенчатым пилообразным напряжением.** В чистом виде метод последовательного счета использован в схеме АЦП со ступенчатым пилообразным напряжением. Этот преобразователь является типичным примером последовательных АЦП (рис. 8.3).

Импульс начала цикла преобразования, называемый импульсом запуска, поступая на вход  $S$  RS-триггера, записывает 1 в этот триггер. Благодаря этому сигналы с генератора тактовых импульсов ГТИ поступают на счетчик. Так как разряды счетчика соединены с разрядами ЦАП, то напряжение на выходе ЦАП  $U_{\text{ЦАП}}$  увеличивается по ступенчатому закону, причем ступень равна кванту, т. е. соответствует единице младшего разряда. Напряжение  $U_{\text{ЦАП}}$  будет увеличиваться линейно до момента срабатывания компаратора. Появление сигнала на выходе компаратора при  $U_{\text{ЦАП}} \geq U_x$  переводит RS-триггер в состояние 0, что препятствует прохождению импульсов от ГТИ через логический элемент И. Число  $N_x$ , соответствующее состоянию разрядов счетчика, будет соответствовать входному напряжению  $U_x$ .

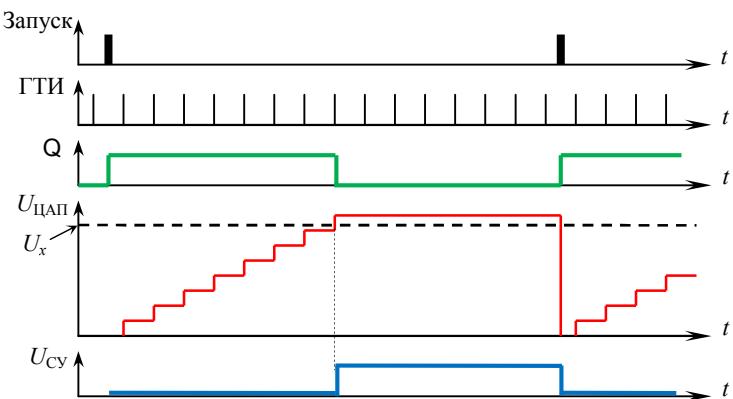
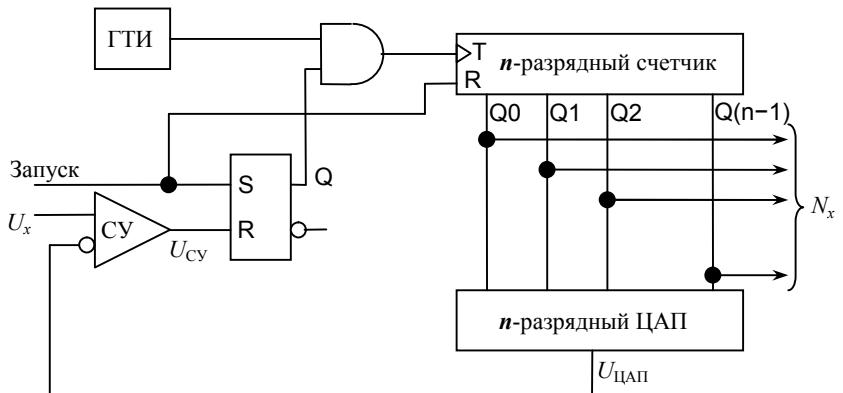


Рис. 8.3. Аналого-цифровой преобразователь со ступенчатым пилообразным напряжением и временные диаграммы, иллюстрирующие его работу

Время преобразования АЦП этого типа является переменным и определяется входным напряжением. Его максимальное значение соответствует максимальному входному напряжению и при числе двоичных разрядов счетчика  $n$  и периоде следования импульсов ГТИ  $t_{ГТИ}$  равно  $t_{ПРmax} = (2^n - 1)t_{ГТИ}$ . При большом числе разрядов время преобразования может оказаться достаточно большим. В связи с этим основные области применения

АЦП последовательного счета – цифровые вольтметры постоянного тока и цифровые системы, предназначенные для работы с постоянными и медленно изменяющимися напряжениями.

**Следящий АЦП.** Рассмотренный АЦП со ступенчатым, пилообразным напряжением легко превратить в преобразователь следящего типа. Для этого необходимо заменить суммирующий счетчик на реверсивный и использовать выход компаратора для управления направлением счета (рис. 8.4).

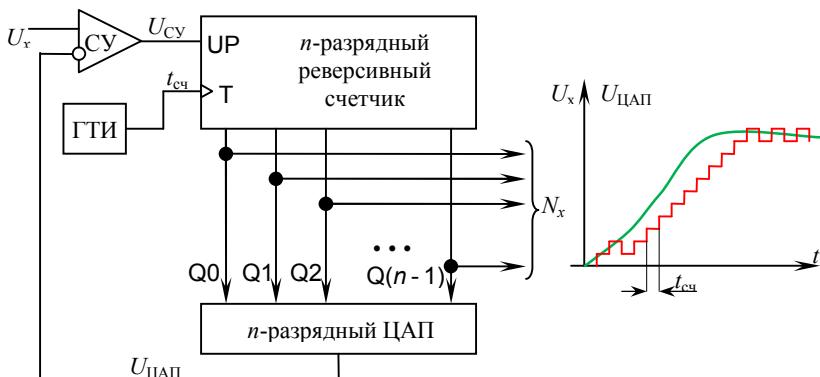


Рис. 8.4. АЦП следящего типа: а) структурная схема; б) временные диаграммы, поясняющие его работу

При равенстве входного напряжения и напряжения обратной связи  $U_{ЦАП}$  выходной код АЦП колеблется вокруг среднего положения с точностью до единицы младшего разряда, как это имеет место в любой дискретной следящей системе. Если в этом состоянии динамического равновесия входной сигнал начнет изменяться, то выходной код преобразователя будет отслеживать его с погрешностью, равной единице младшего разряда, при выполнении условия  $h \leq U'_x(t) \cdot t_{сч}$ , где  $U'_x(t)$  – скорость изменения входного напряжения,  $h$  – шаг квантования. Это соотношение определяет апертурную погрешность АЦП следящего типа, а период тактовых сигналов  $t_{сч}$  является его апертурным временем. Таким образом, период тактовых сигналов, характеризующий

быстродействие, необходимо выбирать исходя, с одной стороны, из допустимого значения апертурной погрешности для заданного входного сигнала, а с другой стороны, из обеспечения срабатывания всех цифровых устройств, компаратора и установления переходных процессов на выходе ЦАП с погрешностью, составляющей определенную часть единицы младшего разряда.

**АЦП с двухтактным интегрированием.** Недостатком рассмотренного последовательного АЦП является относительно низкая помехоустойчивость, что ограничивает его разрешающую способность, как правило, на уровне 8...10 разрядов. От этого недостатка в значительной мере свободны АЦП, использующие в процессе преобразования операцию интегрирования входного сигнала за фиксированный интервал времени.

Одним из наиболее распространенных вариантов преобразователей такого типа является АЦП с двухтактным интегрированием (рис. 8.5).

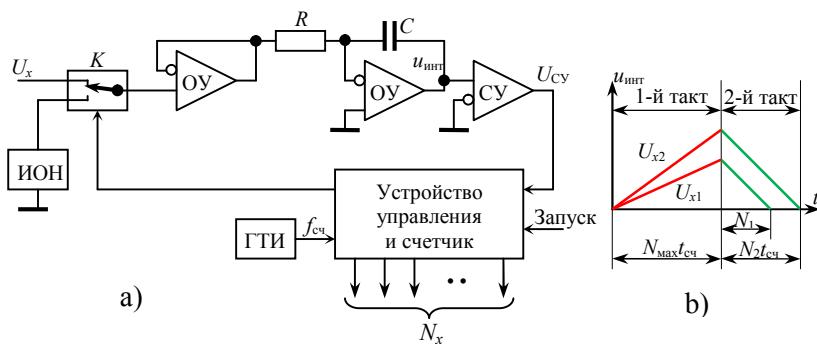


Рис. 8.5. АЦП с двухтактным интегрированием: а) структурная схема, б) временные диаграммы, поясняющие работу АЦП

Полный цикл его работы состоит из двух тактов. В первом с помощью аналогового интегратора происходит интегрирование входного напряжения за фиксированный интервал времени  $T = N_{\max} t_{\text{сч}}$ , где  $N_{\max}$  – емкость счетчика, определяющая разре-

шающую способность АЦП. В результате этой операции в конденсаторе интегратора накапливается заряд

$$q_1 = \bar{U}_x T / RC, \quad (8.9)$$

где  $\bar{U}_x$  – среднее значение входного напряжения за время  $T$ .

Во втором такте происходит разряд конденсатора от источника опорного напряжения (ИОН)  $U_0$ , полярность которого противоположна входному напряжению. Напряжение  $U_0$  подключается к интегратору с помощью переключателя  $K$ . Этот процесс продолжается до возвращения конденсатора в начальное состояние (рис. 8.5б), что фиксируется компаратором СУ. В результате удаленный из конденсатора заряд равен

$$q_2 = U_0 t / RC, \quad (8.10)$$

где  $t$  – время разряда конденсатора. Это время является переменным, и его последующее измерение путем подсчета числа импульсов от ГТИ, следующих с периодом  $t_{\text{сч}}$ , позволяет получить цифровой эквивалент  $\bar{U}_x$ . Действительно, поскольку условием работы АЦП является  $q_1 = q_2$ , то

$$\frac{\bar{U}_x}{RC} T = \frac{U_0}{RC} t, \quad (8.11)$$

откуда

$$t = \frac{\bar{U}_x}{U_0} T \quad (8.12)$$

или в пересчете на количество импульсов ГТИ

$$N_x = \frac{\bar{U}_x}{U_0} N_{\max}. \quad (8.13)$$

Благодаря процедуре двойного интегрирования, этот тип АЦП приобретает важные свойства.

Во-первых, интегрирование входного сигнала приводит к его усреднению и сглаживанию всех быстрых по сравнению с временем интегрирования помех, наводок и шумов. Интегрирование сигнала за время  $T$  эквивалентно его фильтрации с помощью цифрового фильтра низких частот. Характеристика такого фильтра позволяет подавить различные частотные составляющие,

присутствующие на входе интегратора. В частности, если выбрать интервал интегрирования  $T$  кратным периоду частоты питающей сети, например, 50 Гц, то на этой частоте будут полностью подавляться наводки, проходящие по цепям питания и являющиеся одним из факторов, ограничивающих точность АЦП.

Во-вторых, интегрирование входного сигнала приводит к уменьшению динамических погрешностей АЦП, связанных с изменением сигнала в процессе преобразования.

### 8.2.2. АЦП поразрядного уравновешивания

Преобразователь этого типа является наиболее распространенным вариантом аналого-цифровых преобразователей. В АЦП поразрядного уравновешивания входная величина последовательно сравнивается с суммой эталонов, имеющих значение  $2^i$  квантов, где  $i = n - 1, n - 2, \dots, 2, 1, 0$  ( $n$  – число двоичных разрядов числа, которым представлен результат преобразования). Таким образом, два соседних эталона отличаются в 2 раза по значению. Уравновешивание входной величины начинается с эталона, имеющего максимальное значение. Сравнивающее устройство выполняет сравнение этого эталона с входной величиной. В зависимости от результата сравнения определяется цифра старшего разряда двоичного числа, снимаемого с АЦП. Если эталон больше входной величины, то в старшем разряде числа ставится 0 и далее производится уравновешивание входной величины следующим эталоном в 2 раза меньшего значения. Если же первый эталон меньше (или равен) входной величины, то в старшем разряде двоичного числа ставится 1 и дальше производится уравновешивание разности входной величины и первого эталона. Аналогичные действия производятся для всех используемых эталонов. Следовательно, после окончания процесса преобразования входная величина будет уравновешена суммой тех эталонов, у которых в соответствующих им разрядах двоичного числа стоят единицы. Сравнение входной величины и суммы эталонов производится с помощью одного сравнивающего устройства. Это позволяет для  $n$ -разрядного АЦП выполнить весь процесс преобразования за  $n$  последовательных шагов приближения (итераций)

вместо  $2^n - 1$  при использовании единичных приближений и получить существенный выигрыш в быстродействии. Так, уже при  $n = 10$  этот выигрыш достигает двух порядков и позволяет получить с помощью таких АЦП в зависимости от числа используемых разрядов до  $10^6$ – $10^7$  преобразований в секунду. В то же время статическая погрешность этого типа преобразователей, определяемая в основном используемым в нем ЦАП, может быть очень малой, что позволяет реализовать разрешающую способность до 16 двоичных разрядов.

На рис. 8.6. приведена структурная схема АЦП поразрядного уравновешивания. Основным блоком АЦП поразрядного уравновешивания является регистр последовательных приближений (РПП).

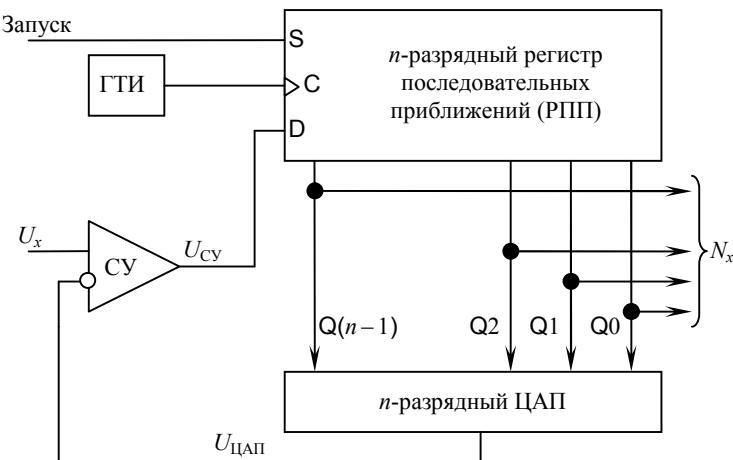


Рис. 8.6. Структурная схема АЦП поразрядного уравновешивания

*Алгоритм работы РПП.* После появления сигнала начальной установки на входе  $S$  в схеме осуществляется поочередная запись информационных символов, поступающих на вход данных  $D$ , в разряды с  $n - 1$  по 0. В результате начальной установки старший разряд регистра устанавливается в состояние 1 [ $Q(n - 1) = 1$ ], а остальные разряды в состояние 0. Пока сигнал на

входе  $S = 1$ , схема не чувствительна к сигналам, поступающим от ГТИ на вход  $C$ , и остается в указанном состоянии.

После того как сигнал на входе  $S$  примет значение 0, начинается продвижение единичного значения в сторону младших разрядов. Тот разряд, в котором в начале данного такта находится *продвигаемое единичное значение*, чувствителен к записи сигнала, имеющегося на входе  $D$ , то есть в нем устанавливается то значение, которое имеется на входе  $D$  в момент перехода сигнала на входе  $C$  из 0 в 1; остальные разряды регистра в это время нечувствительны к данным, имеющимся на входе  $D$ . Поочередная запись данных со входа  $D$  заканчивается по прошествии  $n$  тактов ГТИ, то есть когда произойдет запись в нулевой разряд регистра. Пока сигнал на входе  $S$  остается равным 0, во всех разрядах регистра сохраняются состояния, полученные в результате записи данных. Чтобы возобновить работу схемы, необходимо снова произвести начальную установку.

Когда РПП находится в исходном состоянии ( $Q(n-1) = 1$ ,  $Q(n-2) = \dots = Q1 = Q0 = 0$ ),  $U_{\text{ЦАП}} = U_{\max}/2$ . Если значение  $U_x$  лежит ниже середины диапазона  $(0, U_{\max})$ , то  $U_{\text{СУ}} = 0$ , и это значение сохранится в  $(n-1)$ -м разряде РПП в первом периоде тактового сигнала после установления  $S = 0$ . При этом произойдет переключение напряжения на  $(n-1)$ -м входе резисторной матрицы  $R-2R$  в ЦАП; в противном случае в  $(n-1)$ -м разряде останется 1, и переключения напряжения на  $(n-1)$ -м входе резисторной матрицы  $R - 2R$  не произойдет. В то же время, когда определяется, какое значение должно остаться в  $(n-1)$ -м разряде, примет единичное значение сигнал на выходе  $(n-2)$ -го разряда РПП, что соответствует добавлению в  $U_{\text{ЦАП}}$  напряжения, равного  $U_{\max}/4$ . В зависимости от соотношения между  $U_x$  и вновь возникающим значением  $U_{\text{ЦАП}}$  в очередном периоде тактового сигнала в  $(n-2)$ -м разряде РПП сохранится 1 или запишется 0; в последнем случае произойдет удаление из  $U_{\text{ЦАП}}$  составляющей, равной  $U_{\max}/4$ . Затем эта процедура с первоначальным присвоением единичного значения и последующей записью значения, возникающего на выходе СУ, выполняется для последующих разрядов

$(n - 2), \dots, 1, 0$ . Можно сказать, что на  $i$ -м шаге ( $i = 1, \dots, n$ ) решается вопрос о том, какому из двух соседних интервалов размежом  $U_{\max}/2^i$  принадлежит значение  $U_x$ , причем граница задается результатом выполнения предыдущих шагов и единичным значением разряда, о котором идет речь на данном шаге. По окончании преобразования на выходе ЦАП возникает ближайший снизу к  $U_x$  уровень напряжения, а на выходах РПП оказывается записанным его двоичное число.

На рис. 8.7 приведены временные диаграммы, поясняющие работу 4-разрядного АЦП поразрядного уравновешивания. В первом цикле работы АЦП (после окончания первого сигнала Запуск) преобразуемое напряжение  $U_x = 9.5$  единиц эталона, а во втором цикле  $U_x = 6.5$  единиц эталона.

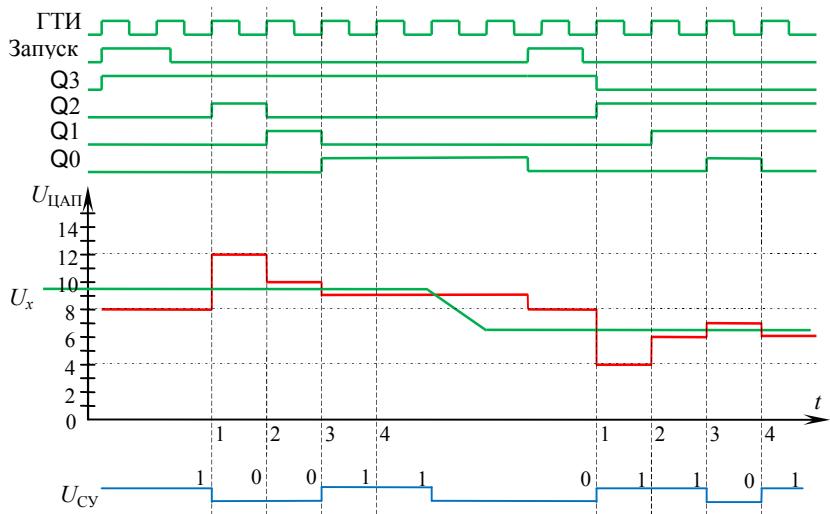


Рис. 8.7. Временные диаграммы работы 4-разрядного АЦП поразрядного уравновешивания

### 8.2.3. АЦП параллельного преобразования

В АЦП параллельного преобразования (метод считывания) используется набор из  $2^n - 1$  эталонов. Младший эталон равен

одному кванту, следующий – двум квантам и старший –  $2^n - 1$  квантам. Так как при этом методе преобразования входное напряжение одновременно сравнивается со всеми эталонами, то для осуществления такого сравнения необходимо столько сравнивающих устройств, сколько эталонов имеется в наборе, то есть  $2^n - 1$ . Результат преобразования фиксируется по числу сравнивающих устройств, зафиксировавших равенство или превышение входной величины по отношению к данному эталону. Отсюда следует, что непосредственным результатом преобразования является так называемый единичный (унитарный) код в виде единичных сигналов на выходах тех сравнивающих устройств, для которых выполнено указанное выше условие.

Полученный единичный код поступает на  $2^n - 1$  входов приоритетного шифратора, который преобразует этот код в  $n$ -разрядное двоичное число. Полученное двоичное число, как правило, сохраняется в  $n$ -разрядном регистре.

На рис. 8.8 приведена схема 3-разрядного параллельного АЦП. При увеличении входного напряжения сигналы на выходах компараторов устанавливаются в состояние 1 по очереди – снизу вверх. Такая очередность не гарантируется при быстром нарастании входного сигнала, так как из-за различия во временах задержки компараторы могут переключаться в другом порядке. Приоритетное кодирование позволяет избежать ошибки, возможной в этом случае, благодаря тому, что единицы в младших разрядах не принимаются во внимание приоритетным шифратором.

Благодаря одновременной работе компараторов параллельный АЦП является самым быстрым. Например, восьмиразрядный преобразователь типа MAX104 позволяет получить 1 млрд отсчетов в секунду при времени задержки прохождения сигнала не более 1,2 нс. Недостатком этой схемы является высокая сложность, что приводит к высокой стоимости (сотни долларов США) и значительной потребляемой мощности. Интегральная схема MAX104, например, потребляет около 4 Вт.

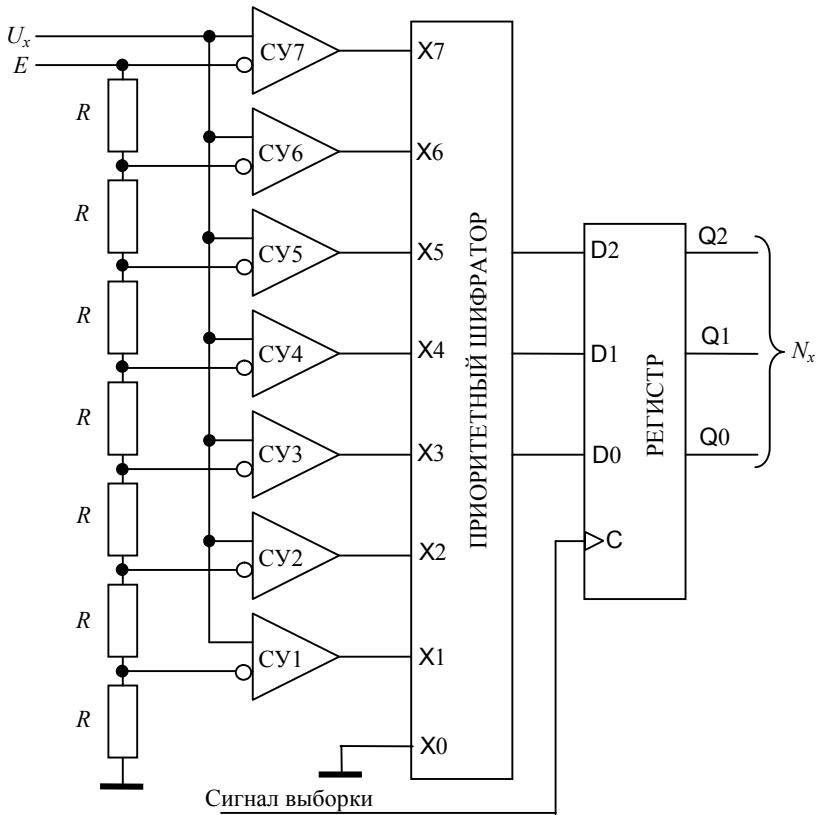


Рис. 8.8. Схема 3-разрядного параллельного АЦП

### 8.3. Устройства выборки и хранения (УВХ)

Устройства выборки и хранения необходимы для уменьшения динамических погрешностей, возникающих при дискретизации изменяющихся во времени непрерывных сигналов. Их работа основана на принципе фиксации мгновенного значения сигнала на время, необходимое для последующего преобразования в АЦП. В большинстве случаев для этого используют различные сочетания накопительного конденсатора, аналоговых ключей и усилительных каскадов. Безусловным достижением последних

лет является разработка полностью интегральных микросхем выборки и хранения.

В устройствах выборки и хранения осуществляется переход от непрерывной функции  $U(t)$  к последовательности аналоговых значений  $\{U(t_n)\}$  ( $n = 1, 2, \dots$ ), квантование выборочных значений происходит в АЦП. В цифровых устройствах сначала осуществляется дискретизация исходного сигнала, а уже затем его квантование и запоминание выборочных значений в цифровой форме.

Наибольшее распространение получили аналоговые устройства, использующие прямоугольные стробирующие импульсы постоянной длительности, что эквивалентно применению достаточно быстродействующих аналоговых ключей, времена включения и выключения которых не зависят от входного сигнала. Механизм образования отсчетов в таких устройствах связан, как правило, с использованием малой постоянной времени эквивалентной  $RC_{\text{h}}$ -цепи по сравнению с длительностью стробирующего импульса  $t_{\text{стр}} (\eta = t_{\text{стр}} / RC_{\text{h}} \gg 1)$  и отнесением отсчета (в виде напряжения на накопительном конденсаторе) к моменту окончания этого импульса (рис. 8.9).

Коэффициент передачи рассматриваемого устройства в режиме выборки имеет вид

$$\frac{u_{\text{out}}}{u_{\text{in}}} = \frac{1}{1 + j f / f_0}, \quad (8.14)$$

где  $f_0 = 1/2\pi RC_{\text{h}}$ .

Зависимость от частоты приводит к возникновению погрешности коэффициента передачи, равной

$$1 - \frac{u_{\text{out}}}{u_{\text{in}}} = \frac{-j f / f_0}{1 + j f / f_0}. \quad (8.15)$$

Как следует из выражения (8.15), при  $RC_{\text{h}} = 160$  нс ( $f_0 = 1$  МГц) уже на частоте  $f = 10$  кГц погрешность в передаче амплитуды синусоидального сигнала составляет 1%. При изменении частоты до 100 кГц эта погрешность увеличивается до 10%. Данная погрешность является одной из составляющих динамической погрешности устройств выборки и хранения.

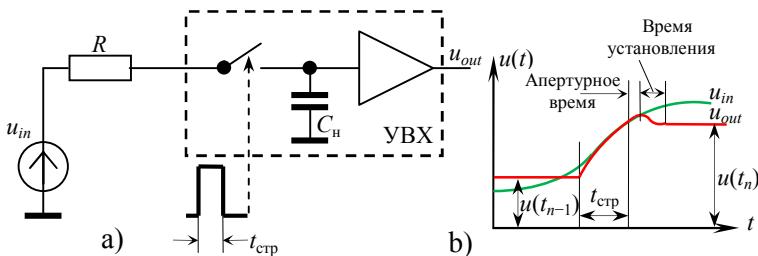


Рис. 8.9. Схема выборки и хранения, использующая прямоугольные стробирующие импульсы: а) эквивалентная схема, б) временные диаграммы, поясняющие ее работу при  $t_{\text{ср}}/RC_{\text{н}} \gg 1$

Для идеальных аналоговых ключей эта составляющая динамической погрешности будет единственной. Однако реальные аналоговые ключи вносят в процесс образования выборочных значений сигнала дополнительные погрешности. Это в первую очередь апертурная погрешность, имеющая принципиальное значение для процесса дискретизации. Апертурным временем называют временной интервал, характеризующий неопределенность момента преобразования входного аналогового сигнала. Неопределенность выражается в том, что напряжение на конденсаторе  $C_{\text{н}}$  несколько отличается от входного напряжения на момент окончания стробирующего сигнала. Возникновение апертурной погрешности в рассматриваемом устройстве связано с тем, что сопротивление реального ключа при переходе из замкнутого состояния в разомкнутое меняется не скачкообразно. Учет этого явления позволяет качественно оценить переход устройства из режима выборки в режим хранения. Апертурное время зависит от закона изменения входного сигнала.

Для сокращения постоянной времени  $RC_{\text{н}}$  заряда конденсатора и ее независимости от сопротивления источника сигнала на выходе УВХ включают повторитель, выполненный на широкополосном операционном усилителе.

## ГЛАВА 9. ПАМЯТЬ

Любая последовательностная схема обладает памятью, поскольку триггер или защелка хранят один бит информации. Однако когда говорят о памяти, подразумевают структуру, в которой биты хранятся в структурированной форме, обычно в виде двумерного массива, в котором одновременно доступна целая строка битов.

Применения памяти многочисленны и разнообразны. Постоянное запоминающее устройство (ПЗУ) может применяться в центральном процессоре (ЦП) микропроцессорной системы для хранения информации об элементарных шагах, совершаемых при выполнении команд из набора команд этого ЦП. Быстрая *статическая память*, расположенная рядом с ЦП, может служить в качестве кэша для хранения недавно использованных команд и данных. Микропроцессорные подсистемы основной памяти могут содержать сотни миллионов ячеек *динамической памяти*, в которых хранятся операционные системы, программы и данные.

Применение памяти не ограничивается микропроцессорными системами и даже чисто цифровыми системами. В аппаратуре телефонных систем общего пользования, например, для выполнения некоторых преобразований оцифрованных речевых сигналов применяются ПЗУ, а быстрая статическая память используется в качестве переключающего устройства при маршрутизации цифровых сообщений, посредством которых осуществляется связь между абонентами. Многие портативные проигрыватели звуковых компакт-дисков осуществляют *чтение вперед* и в течение нескольких секунд хранят звуковой фрагмент в динамической памяти, благодаря чему звучание продолжается даже при тряске (для сохранения звукового фрагмента длительностью в 1 секунду требуется около 1.4 миллиона битов). Кроме этого существует много других примеров современной аудио- и видеоаппаратуры, в которой память применяется для временного хранения оцифрованных сигналов при их обработке в цифровых сигнальных процессорах.

## 9.1. Постоянные запоминающие устройства

*Постоянное запоминающее устройство (ПЗУ, ROM)* является комбинационной схемой с  $n$  входами и  $b$  выходами (рис. 9.1). Входы называются *адресными входами* и обычно обозначаются  $A_0, A_1, \dots, A_{(n-1)}$ . Выходы называются *выходами данных* и обозначаются, как правило,  $D_0, D_1, \dots, D_{(b-1)}$ .

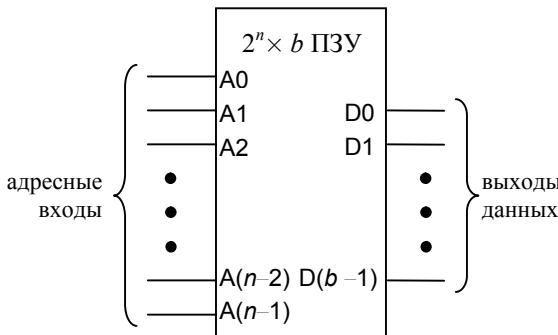


Рис. 9.1. Стандартное изображение ЗУ размером  $2^n \times b$

ПЗУ хранит таблицу истинности комбинационной логической схемы с  $n$  входами и  $b$  выходами. В табл. 9.1 представлена таблица истинности комбинационной схемы с 3 входами и 4 выходами. Содержимое этой таблицы можно хранить в ПЗУ размером  $2^3 \times 4 = (8 \times 4)$ .

Таблица 9.1

Таблица истинности для комбинационной логической схемы с 3 входами и 4 выходами

Входы			Выходы			
A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Если не учитывать задержки распространения, то можно сказать, что сигналы на выходах данных в любой момент времени равны битам той строки таблицы истинности, которая выбирается сигналами на адресных входах.

В большинстве случаев ПЗУ не является памятью в прямом смысле этого слова, поскольку это комбинационная, а не последовательностная схема. Постоянные запоминающие устройства называют памятью потому, что они функционируют как память.

Поскольку ПЗУ представляет собой комбинационную схему, можно сказать, что оно не является настоящей памятью. Описывая работу ПЗУ, можно считать, что это устройство подобно любому другому комбинационному логическому элементу. Однако можно также считать, что в ПЗУ при изготовлении или программировании была *сохранена* определенная информация.

Хотя мы полагаем, что ПЗУ – это один из типов памяти, важно отличать его от других интегральных схем памяти. ПЗУ является *энергонезависимой памятью*, то есть ее содержимое сохраняется в отсутствие напряжения питания.

### 9.1.1. Применение ПЗУ для реализации комбинационных логических функций

Табл. 9.1, в действительности, представляет собой таблицу истинности для дешифратора  $2 \times 4$  с управляемой полярностью выходных сигналов. Имеется два разных способа построения дешифратора: или на основе ПЗУ  $8 \times 4$ , в котором хранится таблица истинности (рис. 9.2), или с помощью дискретных вентилей, как показано на рис. 9.3.

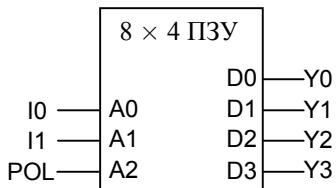


Рис. 9.2. Использование ПЗУ  $8 \times 4$ , в котором записана табл. 9.1, в качестве дешифратора  $2 \times 4$

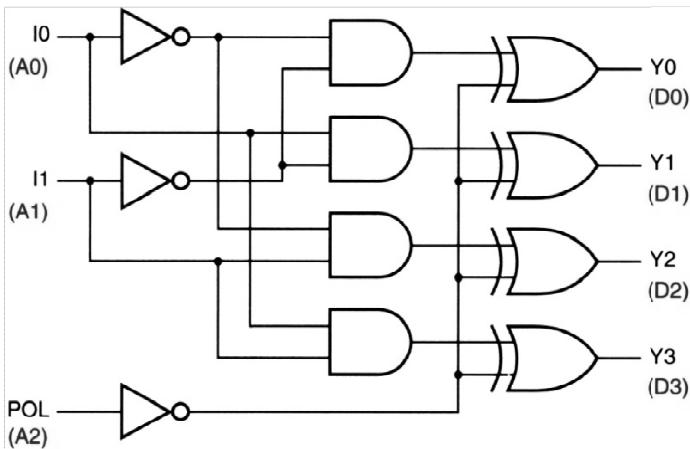


Рис. 9.3. Дешифратор  $2 \times 4$  с управляемой полярностью выходных сигналов

### 9.1.2. Внутренняя структура ПЗУ

Механизм хранения информации, применяемый в ПЗУ, различен для ПЗУ разных типов. В большинстве типов ПЗУ хранению 0 или 1 соответствует наличие или отсутствие диода или транзистора.

На рис. 9.4 приведена схема простейшего ПЗУ размером  $8 \times 4$ , которое можно собрать применяя дешифратор в виде ИС средней степени интеграции и некоторое количество диодов. Сигналами на адресных входах активизируется один из выходов дешифратора. Каждый выход дешифратора называется *линией слова* (*word line*): сигналом на этой линии выбирается одна строка или одно слово таблицы, хранимой в ПЗУ. На рисунке показан случай, когда  $A_2, A_1, A_0 = 101$ . При этом активное значение имеет сигнал на выходе ROW5.

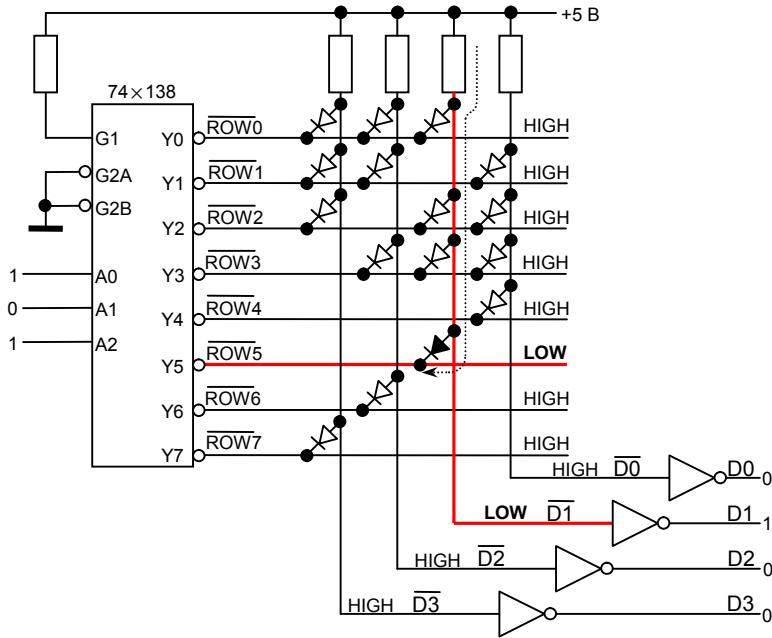


Рис. 9.4. Принципиальная схема простого диодного ПЗУ  $8 \times 4$   
(LOW – низкий уровень, HIGH – высокий уровень)

Каждая вертикальная линия на рис. 9.4 называется *линией бита* (*bit line*), так как она соответствует одному выходному биту ПЗУ. Появление низкого активного уровня  $L$  на линии слова приводит к тому, что низкий уровень устанавливается на тех линиях бита, которые через диод соединены с данной линией слова. В строке с номером 5 имеется только один диод и на соответствующей линии бита ( $\bar{D}_1$ ) возникает низкий уровень. Сигналы на выходах ПЗУ  $D_3, D_2, D_1, D_0$  образуются в результате прохождения сигналов с линий битов через инвертирующие буферы; в рассматриваемом случае – это 0010.

В изображенной на рис. 9.4 схеме ПЗУ каждому пересечению линии слова с линией битов соответствует один бит памяти.

Если на пересечении присутствует диод, то хранится 1, в противном случае хранится 0.

Показанное на рис. 9.4 включение диодов соответствует таблице истинности дешифратора  $2 \times 4$ , приведенной в табл. 9.1. Все это выглядит очень нерациональным: здесь использован дешифратор  $3 \times 8$  и некоторое количество диодов для создания ПЗУ, играющего роль дешифратора  $2 \times 4$ . Тоже самое можно реализовать воспользовавшись частью дешифратора  $3 \times 8$ . Однако существуют более эффективные структуры ПЗУ.

### 9.1.3. Двумерная адресация

Предположим, что вы хотите построить ПЗУ  $128 \times 1$ , воспользовавшись структурой, описанной в предыдущем разделе. Для этого придется применить дешифратор  $7 \times 128$ , содержащий 128 7-входовых вентилей И-НЕ и 14 буферов и инверторов с нагрузочной способностью каждого, равной 64. Имеющиеся в ПЗУ хранят миллионы битов, но в них нет дешифраторов  $20 \times 1$  048 576. Вместо этого для уменьшения дешифратора до размеров порядка квадратного корня из числа адресов применяется другой метод, называемый *двумерной адресацией*.

Основная идея двумерной адресации состоит в компоновке ячеек ПЗУ в виде матрицы, приближающейся, по возможности, к квадратной. На рис. 9.5 в качестве примера показана возможная структура ПЗУ  $128 \times 1$ . Три старших разряда адреса A6, A5, A4 используются для выбора строки. В каждой строке с начальным адресом (A6, A5, A4, x, x, x, x) хранится 16 битов. Когда на вход ПЗУ подан некоторый адрес, все 16 битов в выбранной строке считаются параллельно по линиям битов. 16-входовой мультиплексор выбирает требуемый бит данных по значениям младших разрядов адреса.

Применение двумерной адресации позволяет построить ПЗУ  $128 \times 1$ , используя дешифратор  $3 \times 8$  и 16-входовой мультиплексор.

типлексор (сложность которого сравнима со сложностью дешифратора  $4 \times 16$ ). ПЗУ  $1M \times 1$  можно построить, воспользовавшись дешифратором  $10 \times 1024$  и 1024-ходовым мультиплексором, что нелегко реализовать. Однако это значительно проще, чем одномерный вариант.

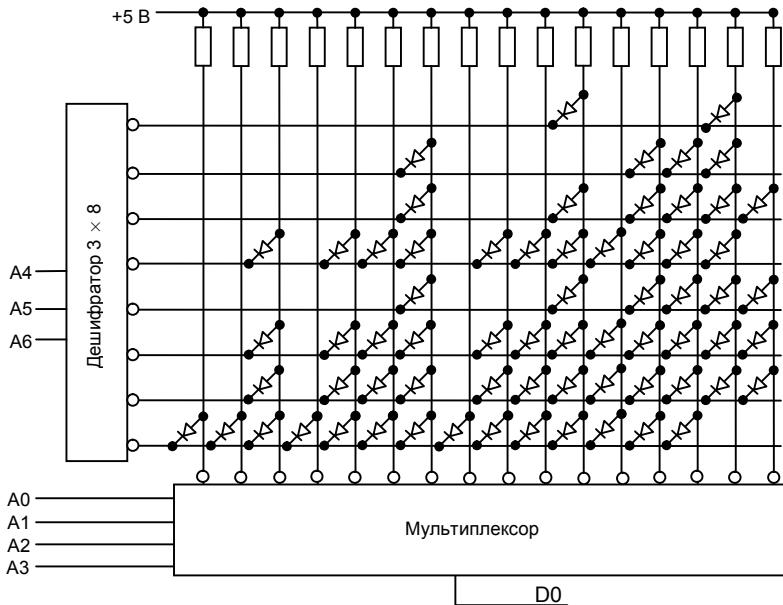


Рис. 9.5. Внутренняя структура ПЗУ  $128 \times 1$ , в котором применено двумерное декодирование

Фактически в ПЗУ, создаваемых на МОП-структуратах, в каждом месте, где хранится бит информации, применяются не диоды, а транзисторы.

Двумерная адресация, помимо уменьшения сложности, обладает и другим достоинством: она позволяет создать кристалл примерно квадратной формы, что важно при его изготовлении и размещении в корпусе. Микросхема, выполненная в виде матри-

цы  $1M \times 1$ , была бы очень длинной, выглядела бы как пленка, и ее нельзя было бы изготовить по экономическим соображениям.

#### 9.1.4. Типы постоянных запоминающих устройств

В настоящее время ПЗУ на дискретных диодах можно увидеть, пожалуй, только в музее вычислительной техники. Современные ПЗУ изготавляются в виде одной ИС. Для запоминания в ПЗУ информации применяются различные методы программирования, сведенные в табл. 9.2 и рассматриваемые ниже.

Т а б л и ц а 9.2

##### Типы изготавляемых серийно ПЗУ

Тип	Технология	Цикл чтения	Цикл записи	Примечания
Масочное ПЗУ	КМОП	10 нс	До 4 недель	Однократная запись; малая потребляемая мощность
Масочное ПЗУ	Биполярная	< 100 нс	До 4 недель	Однократная запись; большая потребляемая мощность; низкая плотность
PROM	Биполярная	< 100 нс	10 мкс/байт	Однократная запись; большая потребляемая мощность; нет расходов на изготовление маски
EPROM	КМОП	25 нс	10 мкс/байт	Многократное использование; малая потребляемая мощность; нет расходов на изготовление маски
EEPROM	nMOP	50 нс	10–50 мкс/байт	Ограниченоное число: $10^5$ – $10^6$ записей в каждой ячейке

Большинство первоначальных интегральных схем ПЗУ были *программируемыми с помощью фотошаблонов ПЗУ* или просто *масочные ПЗУ*. Масочные ПЗУ программируются трафаретом соединений в одной из *масок*, применяемых в процессе изготовления ИС. Для программирования или записи информации в ПЗУ заказчик дает производителю листинг требуемого содержимого

ПЗУ на каком-либо носителе информации. Используя эту информацию, производитель создает, согласно техническим условиям заказчика, одну или большее число масок для изготовления ПЗУ с требуемой конфигурацией. *Расходы производителей ПЗУ на изготовление* доходят до нескольких тысяч долларов; это обусловлено индивидуальным характером изготовления заказных масок. Помимо значительных расходов на изготовление масок, для получения запрограммированной микросхемы требуется срок порядка нескольких недель; поэтому, сегодня масочные ПЗУ применяются, как правило, только при массовом производстве. Для мелкосерийного применения имеются экономически более эффективные возможности, рассматриваемые ниже.

*Программируемые ПЗУ (PROM)* подобны масочным ПЗУ, за исключением того, что пользователь может запомнить значения данных (то есть *запрограммировать* ПЗУ) всего за несколько минут с помощью *программатора PROM*. При изготовлении микросхем PROM все диоды или транзисторы находятся во включенном состоянии. Это соответствует тому, что все биты имеют одно и тоже значение, как правило, равное 1. С помощью программатора ПЗУ отдельным битам можно придать противоположные значения. В биполярных ПЗУ это делается пережиганием тонких *плавких перемычек* внутри микросхемы PROM, соответствующих тем битам, которые должны быть равны нулю. Перемычка пережигается путем ее выбора с помощью сигналов на линиях адреса и линиях данных этой микросхемы с последующей подачей высоковольтного импульса (10 – 30 В) на специальный вход устройства.

У первых биполярных PROM были проблемы с надежностью. Иногда значения запомненных битов изменялись из-за недостаточно хорошо пережженных перемычек, которые со временем восстанавливались; кроме того, иногда возникали нерегулярные сбои из-за застывших капель металла, хаотически перемещающихся внутри корпуса ИС. Однако проблемы эти были преодолены и надежная технология плавких перемычек сегодня применяется не только в биполярных PROM, но и в биполярных ПЛУ.

*Стираемые программируемые ПЗУ (erasable programmable read-only memory, EPROM)* похожи на PROM, но их содержимое можно стереть, облучая микросхему ультрафиолетовым светом, в результате чего все биты принимают значение, равное 1. Очевидно, что свет не вызывает роста плавких перемычек. В EPROM применяется другая технология, а именно: МОП-транзисторы с плавающим затвором.

Как показано на рис. 9.6, в EPROM в месте хранения каждого бита имеется *МОП-транзистор с плавающим затвором*. У каждого транзистора есть два затвора. Плавающий затвор никакуда не подключен и окружен изоляционным материалом с очень малой проводимостью. При программировании EPROM с помощью программатора на неплавающий (управляющий) затвор в месте хранения бита, значение которого должно быть равно 0, подается импульс высокого напряжения, тем самым создается электрическое поле и возникает туннельный эффект. Часть электронов туннелирует сквозь слой изолятора и попадает на плавающий затвор. Заряд на плавающем затворе изменяет ширину канала сток-исток и его проводимость, что используется при чтении. В дальнейшем при выполнении операции чтения наличие отрицательного заряда на плавающем затворе препятствует выбранному МОП-транзистору включаться.

Производители EPROM гарантируют, что правильно запрограммированная ячейка удерживает 70% заряда в течение, по крайней мере, 10 лет, даже в том случае, когда микросхема хранится при температуре 125 °C; поэтому EPROM называют энергонезависимой памятью. Однако содержимое этой памяти можно стереть. Изоляционный материал вокруг плавающего затвора при облучении его ультрафиолетовым светом определенной длины волны становится немного проводящим. Таким образом, содержимое микросхемы EPROM можно стереть, облучая ее ультрафиолетовым светом в течение 5 – 20 минут. Обычно кристалл EPROM располагается в корпусе с кварцевым окошком, через которое его можно засветить, чтобы стереть записанную в нем информацию.

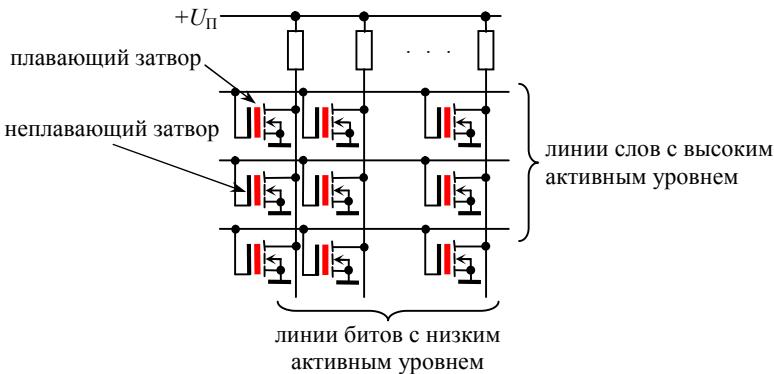


Рис. 9.6. Матрица памяти EPROM на МОП-транзисторах с плавающим затвором

Самым распространенным применением EPROM является хранение программ в микропроцессорных системах. Часто EPROM применяют при разработке программ, когда в процессе отладки программа или другая информация, хранимая в EPROM, должны неоднократно изменяться. Однако ПЗУ и PROM обычно дешевле, чем EPROM того же объема. Поэтому, как только программа отлажена, для снижения стоимости изделий можно воспользоваться ПЗУ или PROM. В действительности сегодня большинство PROM представляют собой EPROM, помещенные в недорогие корпуса без кварцевых окошек; иногда их называют *однократно программируемыми ПЗУ*.

Электрически стираемые программируемые ПЗУ (EEPROM) подобны EPROM, за исключением того, что ранее записанные биты можно стереть электрически. Плавающие затворы транзисторов в EEPROM окружены гораздо более тонким изолирующим слоем (порядка 200 Å), и имеющиеся на них заряды можно удалить, прикладывая к неплавающему затвору напряжение, полярность которого противоположна полярности напряжения, в результате подачи которого происходит накопление заряда. Большие EEPROM (емкостью 1 Мбит и больше) позволяют за одну операцию стирать информацию только блоками фиксиру-

ванного размера; типичный размер блока 128–512 Кбит (16–64 Кбайт). Память такого типа обычно называется *флэш-памятью* (*flash memory, flash EEPROM*), поскольку стирание данных группами происходит *мгновенно* (*in a flash*).

Как видно из табл. 9.2, программирование или запись ячеек EEPROM осуществляется гораздо дольше, чем чтение из нее, поэтому ИС EEPROM не годятся для использования в качестве оперативных запоминающих устройств, которые рассматриваются позже. Кроме того, поскольку изолирующий слой очень тонок, он может разрушиться при многократном выполнении программирования. В результате EEPROM можно перепрограммировать ограниченное число раз, до  $10^6$ . Поэтому, как правило, EEPROM применяется для записи данных, которые должны сохраняться при отключении источника питания и изменяются не очень часто; примером таких данных являются сведения о конфигурации компьютера по умолчанию.

### 9.1.5. Входы управления и временные параметры ПЗУ

Поскольку выходы ПЗУ часто подключаются к шине с тремя состояниями, сигналы в которую в разные моменты времени поступают от различных устройств, большинство серийных микросхем ПЗУ имеют выходы данных с тремя состояниями и *вход разрешения выхода OE*; на этот вход необходимо подать активный уровень, для того чтобы на выходах появились сигналы.

Часто, особенно в тех случаях, когда ПЗУ применяются для хранения программ, к шине подключается несколько микросхем ПЗУ, причем в каждый момент времени сигналы на шину выдает только одна из них. Чтобы упростить структуру таких систем, большинство микросхем ПЗУ снабжены *входом выбора кристалла CS*. Для того чтобы вывести выходы ПЗУ из третьего состояния, необходимо подать сигнал активного уровня не только на вход *OE*, но также и на вход *CS*.

Вход *CS* является не более чем вторым входом разрешения выхода; сигнал на входе *CS*, объединенный логикой И с сигналом на входе *OE*, переводит выходы с тремя состояниями в активный режим. Однако во многих ПЗУ вход *CS* используется так же как

*вход снижения потребляемой мощности.* Когда сигнал на входе CS имеет неактивный уровень, внутри ПЗУ отключается напряжение питания от дешифраторов, драйверов и мультиплексоров. В этом *режиме ожидания* типичное ПЗУ рассеивает менее 10% мощности, потребляемой в *активном режиме* при активном уровне сигнала на входе CS.

На рис. 9.7 показано, как используются сигналы, поступающие на входы CS и OE, внутри типичного ПЗУ. На рис. 9.8 изображены временные характеристики типичного ПЗУ и указаны следующие временные параметры:

$t_{AA}$  – *время доступа по шине адреса.* Этот параметр определяет задержку между моментом установления стабильных значений сигналов на адресных входах ПЗУ и моментом установления достоверных сигналов на выходах данных;

$t_{ACS}$  – *время доступа по выходу выбора кристалла.* Этот параметр характеризует задержку между моментом подачи сигнала на вход CS и моментом установления достоверных сигналов на выходах данных. Эта величина больше времени доступа по шине адреса, если схеме требуется время на переход из режима ожидания в активный режим. Когда сигнал на входе CS управляет только разрешением выхода, это время меньше;

$t_{OE}$  – *время разрешения выдачи данных.* Значение этого параметра много меньше, чем время доступа. Время разрешения выдачи данных равно задержке между моментом времени, когда сигналы на обоих входах OE и CS становятся активными, и моментом, когда выходные каскады с тремя состояниями выходят из высокоомного состояния. В зависимости от того, насколько давно сигналы на адресных входах приняли установленные значения, сигналы на выходах данных к этому времени могут быть верными или не верными;

$t_{OZ}$  – *время запрещения выдачи данных.* Эта величина равна задержке между моментом установления неактивных значений сигналов на входах OE и CS и моментом перехода выходных каскадов с тремя состояниями в высокоомное состояние;

$t_{OH}$  – время удержания данных на выходе. Время удержания данных на выходе равно интервалу, в течение которого сигналы на выходах данных сохраняют свои значения после изменения адреса или после принятия сигналами на входах  $\overline{OE}$  и  $\overline{CS}$  неактивных значений.

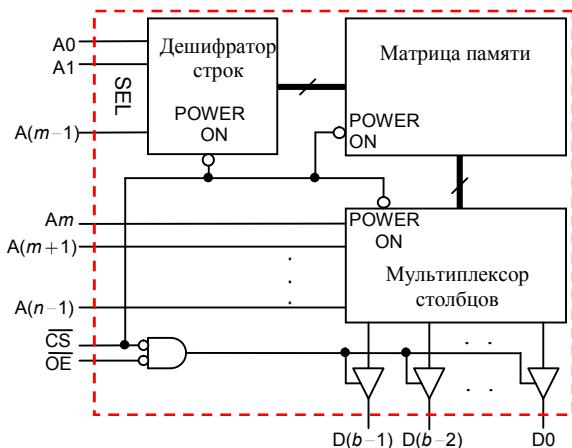


Рис. 9.7. Структура ПЗУ, иллюстрирующая использование сигналов, поступающих на входы управления (POWER ON - вход управления питанием)

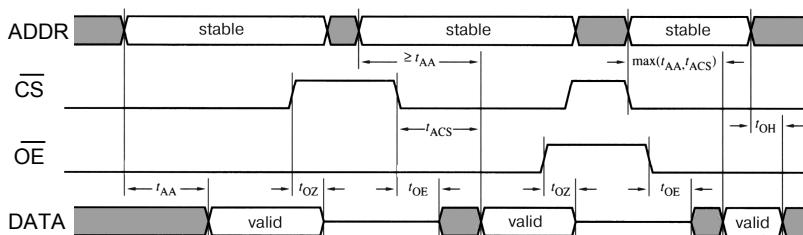


Рис. 9.8. Временные диаграммы, характеризующие работу ПЗУ (stable - установившиеся значения, valid – достоверные значения)

Производитель указывает максимальные и иногда типичные значения всех временных параметров. Обычно для  $t_{OE}$  и  $t_{OH}$

указываются также минимальные значения. Минимальное значение  $t_{OH}$  часто принимают равным 0; это означает, что фминимальная задержка комбинационной логики внутри ПЗУ равна нулю.

Самым распространенным применением ПЗУ является хранение программ в микропроцессорных системах. Однако во многих случаях ПЗУ могут обеспечить дешевую реализацию произвольных сложных комбинационных функций.

## 9.2. Оперативные запоминающие устройства

Если к памяти можно обратиться в любой момент времени, чтобы запомнить в ней или извлечь из нее информацию, то ее называют *памятью с чтением и записью* (*read/write memory, RWM*). Большинство устройств памяти такого типа, применяемых сегодня в цифровых системах, является *оперативной памятью* (ОЗУ) или *памятью с произвольным доступом* (*random-access memory, RAM*). Это означает, что каждый раз при чтении или записи можно выбрать любую ячейку памяти. С этой точки зрения ПЗУ (*ROM*) также является памятью с произвольным доступом, но название ОЗУ (*RAM*) обычно относится только к памяти с произвольным доступом, в которой возможны чтение и запись.

В *статическом ОЗУ* (*static RAM, SRAM*) слово, записанное однажды в какую-то ячейку, сохраняется в ней пока на микросхему подано напряжение питания, если только содержимое этой ячейки не изменяется в результате новой записи. В *динамическом ОЗУ* (*dynamical RAM, DRAM*) данные, сохраняемые в каждой ячейке, необходимо периодически обновлять путем их чтения и последующей повторной записи; в противном случае они будут потеряны.

В большинстве ОЗУ хранящаяся в них информация теряется при отключении питания; другими словами, ОЗУ является *энергозависимой памятью* (*volatile memory*). Но бывают также ОЗУ, называемые *энергонезависимой памятью* (*nonvolatile memory*), которые сохраняют записанную в них информацию даже при отключении питания.

## 9.3. Статические оперативные запоминающие устройства

### 9.3.1. Входы и выходы статического ОЗУ

Как и в случае ПЗУ, у ОЗУ имеются адресные входы, входы управления и выходы данных; но, кроме этого, у ОЗУ есть еще и входы данных. На рис. 9.9 показаны входы и выходы простого статического ОЗУ, предназначенного для хранения  $2^n \times b$  битов.

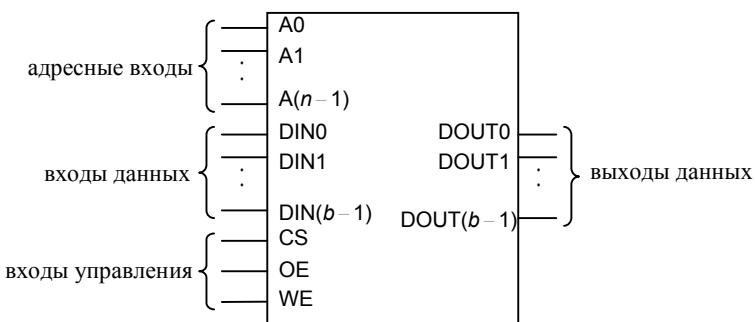


Рис. 9.9. Общая структура ОЗУ  $2n \times b$

Кроме тех же входов управления, что и у ПЗУ, имеется *вход разрешения записи WE (write-enable input)*. Входные данные записываются в выбранную ячейку памяти, когда сигнал на входе WE имеет активный уровень.

Ячейки памяти статического ОЗУ ведут себя так же, как D-защелки, а не как переключающиеся по фронту D-триггеры. Это означает, что всякий раз, когда на вход WE подан сигнал активного уровня, защелка в выбранной ячейке памяти *открыта* (или *прозрачна*): входные данные поступают на защелку и появляются на ее выходе. Фактически запоминается то значение, которое присутствует на входе защелки в момент ее закрытия.

Обычно у статического ОЗУ бывают только два режима доступа:

*Режим чтения.* На входы CS и OE поданы сигналы активного уровня, а на адресные входы поступают сигналы адреса. С выходов защелок выбранной ячейки памяти  $b$ -разрядные данные поступают на выходы данных DOUT.

*Режим записи.* На адресные входы подаются сигналы адреса, а на входы данных DIN – слово данных; затем на входы CS и WE поступают сигналы активного уровня. Открываются  $b$  защелок выбранной ячейки памяти и в них запоминается входное  $b$ -разрядное слово данных.

При организации доступа к статическому ОЗУ требуется некоторая осторожность, поскольку в том случае, когда не удовлетворяются временные требования, предъявляемые микросхемой ОЗУ, при записи в выбранную ячейку возможна непреднамеренная потеря информации, хранящейся в одной или в нескольких других ячейках.

### 9.3.2. Структура статического ОЗУ

Схема в каждом двоичном разряде статического ОЗУ (ячейка статического ОЗУ; SRAM cell) имеет вид, приведенный на рис. 9.10. Элементом, хранящим информацию в каждой ячейке, служит D-защелка. Когда на вход  $\overline{SEL}$  подан сигнал активного уровня, сохраняемая в ячейке информация появляется на ее выходе, который соединен с соответствующей линией битов. Если сигнал активного уровня поступает на оба входа  $\overline{SEL}$  и  $\overline{WR}$ , то защелка открыта и в ней запоминается новый бит данных.

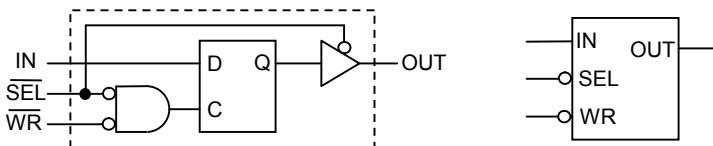


Рис. 9.10. Схема ячейки статического ОЗУ

На рис. 9.11 показано, как ячейки статического ОЗУ, объединенные в виде матрицы, вместе с дополнительной управляю-

щей логикой образуют законченное статическое ОЗУ емкостью  $8 \times 4$  битов. Как и в простом ПЗУ, с помощью дешифратора адресных линий в любой момент времени выбирается для доступа определенная строка статического ОЗУ.

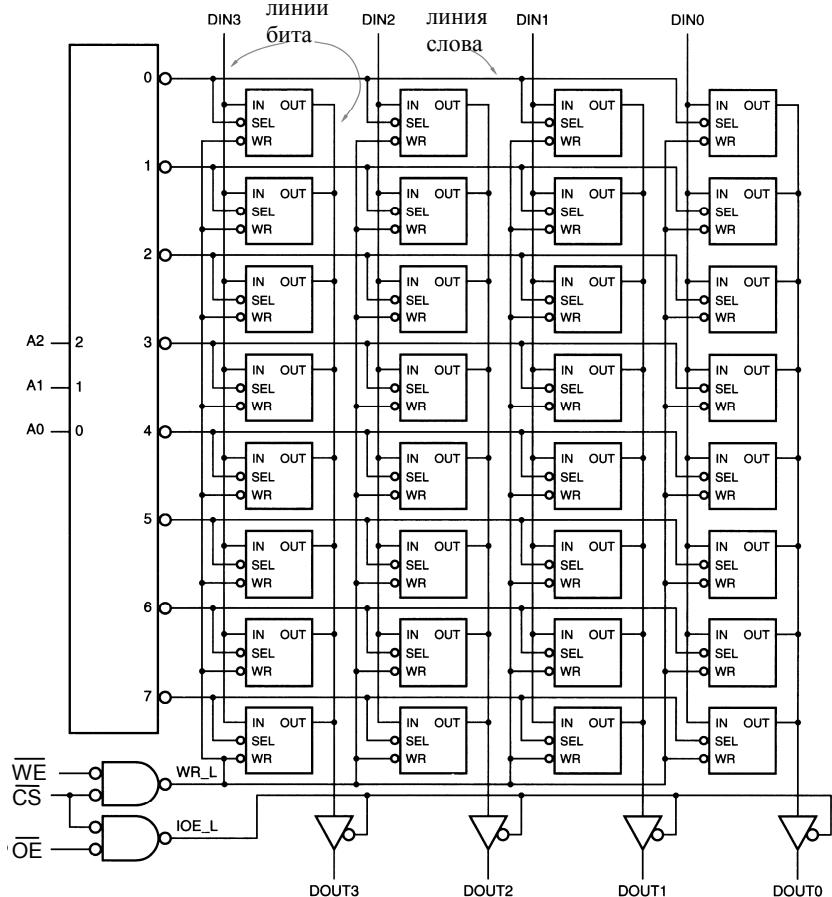


Рис. 9.11. Внутренняя структура статического ОЗУ  $8 \times 4$

Хотя на рис. 9.11 приведена несколько упрощенная модель внутренней структуры статического ОЗУ, она достаточно точно отражает основные моменты в работе этого устройства:

- При выполнении операций чтения выходные данные также, как и в ПЗУ, являются комбинационными функциями сигналов на адресных входах. Изменение адреса в то время, когда разрешено появление выходных данных на шине, не наносит никакого вреда. Время доступа при выполнении операции чтения отсчитывается от момента, когда последний из сигналов на адресном входе принимает установленное значение.
- При выполнении операций записи входные данные запоминаются в защелках. Это означает, что данные должны удовлетворять определенным требованиям по времени установления и времени удержания относительно заднего перепада в сигнале на входе разрешения защелки. Другими словами, сигнал данных на D-входе защелки не обязан оставаться неизменным в момент времени, когда сигнал  $\overline{WR}$  внутри схемы переходит на активный уровень; сигнал данных должен оставаться неизменным лишь в течение некоторого времени, предшествующего тому моменту, когда сигнал  $\overline{WR}$  переходит на неактивный уровень.
- Во время операций записи сигналы на адресных входах *не должны* изменяться в течение определенного времени установления до перехода сигнала  $\overline{WR}$  внутри схемы на активный уровень и в течение времени удержания после того, как сигнал  $\overline{WR}$  перейдет на неактивный уровень. В противном случае данные могут оказаться *размазанными* по некоторому массиву ячеек из-за паразитных импульсов на линиях  $\overline{SEL}$ , которые могут возникнуть при изменении сигналов на адресных входах дешифратора.
- Сигнал  $\overline{WR}$  переходит на активный уровень внутри схемы только в том случае, когда активные значения имеют сигналы  $\overline{CS}$  и  $\overline{WE}$ . Поэтому *цикл записи (write cycle)*

*cycle*) начинается с установления активного уровня сигналов  $\overline{CS}$  и  $\overline{WE}$  и заканчивается, когда любой из этих сигналов переходит на неактивный уровень. Время установления и время удержания адреса и данных определены относительно этих событий.

### 9.3.3. Временные параметры статического ОЗУ

На рис. 9.12 приведены временные диаграммы и определение временных параметров, которые обычно задаются для операции чтения из статического ОЗУ:

$t_{AA}$  – время доступа по шине адреса (*access time from address*). Этим параметром определяется время, спустя которое выходные данные принимают установленное значение после изменения адреса при условии, что сигналы OE и CS к этому времени уже имеют активный уровень или достаточно скоро должны стать такими. Например, когда разработчики говорят о 70-наносекундном статическом ОЗУ, обычно имеется в виду этот параметр;

$t_{ACS}$  – время доступа по входу выбора кристалла (*access time from chip select*). Этим параметром определяется время, спустя которое выходные данные принимают установленные значения после перехода сигнала CS на активный уровень при условии, что сигналы на адресных входах и сигнал OE уже имеют активный уровень или достаточно скоро должны стать такими. Часто значение этого параметра совпадает со временем  $t_{AA}$ , но иногда его величина больше  $t_{AA}$  при работе статического ОЗУ в режиме пониженного потребления мощности и меньше  $t_{AA}$ , когда статическое ОЗУ не находится в этом режиме.

$t_{OE}$  – время разрешения выхода (*output-enable time*). Этим параметром определяется время, через которое буферы с тремя состояниями на выходе выйдут из высокоомного состояния, после того как оба сигнала OE и CS перейдут на активный уровень. Этот параметр обычно меньше, чем величина  $t_{ACS}$ , поэтому внутри ОЗУ вызов данных возможен раньше, чем сигнал OE примет активное значение; во многих приложениях

это свойство используется для достижения малых времен доступа, чтобы избежать конфликтов в шине.

$t_{OZ}$  – время запрещения выхода (*output-disable time*). Этим параметром определяется время, необходимое для того, чтобы буферы с тремя состояниями на выходе перешли в высокоомное состояние, после того как сигналы  $\overline{OE}$  или  $\overline{CS}$  перейдут на неактивный уровень.

$t_{OH}$  – время удержания сигнала на выходе (*output-hold time*). Этот параметр показывает, как долго выходные данные сохраняют установленные значения после изменения адреса на входе.

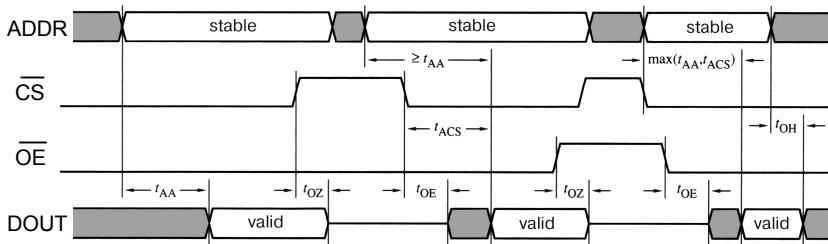


Рис. 9.12. Временные параметры для операции чтения ( $\overline{WE} = H$ ) из статического ОЗУ

(stable – установленные значения; valid – достоверные данные)

Временные диаграммы и временные параметры для режима чтения из статического ОЗУ идентичны с теми, которые были введены при рассмотрении режима чтения из ПЗУ в разделе 9.1.5. Когда не производится запись в статическое ОЗУ, его можно использовать точно так же, как ПЗУ. Позже мы увидим, что иначе обстоит дело с динамическими ОЗУ.

Временные параметры операции записи, указанные на рис. 9.13, определяются следующим образом:

$t_{AS}$  – время установления адреса до начала записи (*address setup*). Сигналы на всех адресных входах должны оставаться постоянными в течение указанного времени перед тем, как оба сигнала CS и WE примут активное значение. В про-

тивном случае данные могут быть искажены и нельзя сказать, в каких именно ячейках это может произойти;

$t_{AH}$  – время удержания адреса после окончания записи (*address hold*). Так же, как и в отношении параметра  $t_{AS}$ , сигналы на всех адресных входах должны поддерживаться неизменными в течение времени  $t_{AH}$ , после того как хотя бы один из сигналов CS или WE перейдет на неактивный уровень;

$t_{CSW}$  – время установления сигнала «выбор кристалла» до окончания записи (*chip-select setup time before end of write*). Уровень сигнала CS должен оставаться активным в течение отрезка времени длительностью не менее  $t_{CSW}$  перед окончанием цикла записи;

$t_{WP}$  – длительность импульса записи (*write-pulse width*). Для надежного запоминания данных в выбранной ячейке сигнал WE должен иметь активный уровень в течение времени, равного, по крайней мере,  $t_{WP}$ ;

$t_{DS}$  – время установления данных до окончания записи (*data setup time before end of write*). Сигналы на всех входах данных должны иметь постоянные значения в течение этого отрезка времени перед окончанием цикла записи. В противном случае данные могут оказаться незапомненными;

$t_{DH}$  – время удержания данных после окончания записи (*data hold time after end of write*). Аналогично параметру  $t_{DS}$ , сигналы на всех входах данных должны поддерживаться неизменными в течение этого интервала времени после окончания цикла записи.

Производителями статических ОЗУ определяются два типа цикла записи: запись по сигналу WE и запись по сигналу CS, как показано на рис. 9.13. Единственное различие между этими циклами состоит в том, какой из сигналов WE или CS последним переходит на активный уровень и у какого из этих сигналов первым уровень становится неактивным при разрешении операции записи внутри статического ОЗУ.

Требования к времененным параметрам при записи в статическое ОЗУ можно было бы несколько ослабить, если бы в ячейках вместо защелок применялись переключающиеся по фронту

D-триггеры с объединенными тактовым входом и входом разрешения, на которые подавались бы сигналы SEL и WR.

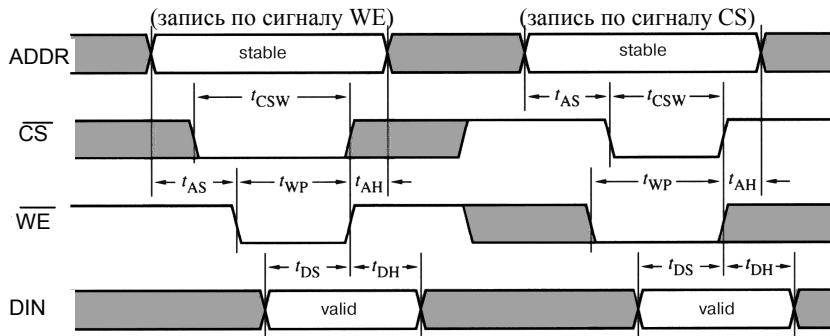


Рис. 9.13. Временные параметры для операции записи в статическое ОЗУ (stable – установленные значения; valid – достоверные данные)

Однако так не поступают, потому что при этом по меньшей мере вдвое увеличилась бы площадь, занимаемая каждой ячейкой в кристалле, так как D-триггер состоит из двух защелок. Таким образом, разработчику логических устройств приходится согласовывать временные параметры статических ОЗУ на защелках с временными параметрами переключающихся по фронту регистров и с временными требованиями конечных автоматов, используемых в системе.

Физические размеры матрицы в кристалле большого статического ОЗУ напрямую не связаны с емкостью памяти. Как и в ПЗУ, ячейки статического ОЗУ образуют почти квадратную матрицу, и при чтении внутри ОЗУ считывается полная строка. Например, структура микросхемы статического ОЗУ емкостью  $32K \times 8$  может быть очень похожа на структуру ПЗУ  $32K \times 8$ . Во время чтения требуемые данные проходят на выходную шину данных через мультиплексоры столбцов в соответствии с подмножеством значений адреса в определенных разрядах. Схема разрешения записи обеспечивает доступ при записи только к одному столбцу в каждом подмассиве, который определяется тем же самым подмножеством адресных битов.

Внутри разновидности статических ОЗУ, называемых *синхронными статическими ОЗУ* (*synchronous SRAM, SSRAM*), по-прежнему применяются защелки, но имеется тактируемый интерфейс для сигналов управления, адресных сигналов и сигналов данных. На пути адресных сигналов и сигналов управления находятся внутренние переключающиеся по фронту регистры. В результате действие, задаваемое перед нарастающим фронтом тактового сигнала, выполняется внутри микросхемы на следующем такте. Возможен также пакетный режим работы такой памяти, при котором данные читаются из следующих одна за другой ячеек. В этом режиме нет необходимости в каждом цикле подавать новый адрес.

## **9.4. Динамические оперативные запоминающие устройства**

Основной ячейкой памяти в статическом ОЗУ является *D-защелка*, для которой требуются четыре вентиля в дискретном исполнении и от четырех до шести транзисторов в заказном статическом ОЗУ в виде БИС. Для построения ОЗУ с более высокой плотностью (с большим числом двоичных ячеек в кристалле), были созданы ячейки, в которых на каждый бит приходится всего лишь по одному транзистору.

### **9.4.1. Структура динамического ОЗУ**

Используя только один транзистор, нельзя построить элемент с двумя устойчивыми состояниями. В ячейках памяти *динамического ОЗУ* (*dynamic RAM, DRAM*) информация сохраняется в виде напряжения на конденсаторе очень малой емкости, доступ к которому осуществляется с помощью МОП-транзистора. На рис. 9.14 показана ячейка памяти динамического ОЗУ, в котором запоминается один бит и обращение к которой происходит при подаче на линию слова напряжения высокого уровня.

Чтобы запомнить 1, на линию бита подается напряжение высокого уровня, которое через открытый транзистор поступает на конденсатор и заряжает его. Для сохранения 0 на линию бита

подается напряжение низкого уровня, в результате чего конденсатор разряжается.

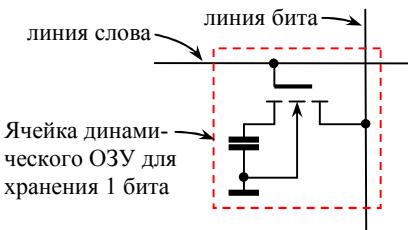


Рис. 9.14. Ячейка памяти в динамическом ОЗУ для хранения одного бита

Для чтения информации, хранящейся в ячейке динамического ОЗУ, на линии бита сначала устанавливается *напряжение предварительного уровня*, значение которого находится посередине между высоким и низким уровнями, а затем на линию слова подается напряжение высокого уровня. В зависимости от того, заряжен или разряжен конденсатор, напряжение на линии бита становится несколько выше или ниже предварительного уровня. С помощью усилителя *считывания* это небольшое изменение напряжения доводится до уровня логической 1 или логического 0 соответственно. Необходимо отметить, что при чтении содержимого ячейки изменяется исходное напряжение на конденсаторе, поэтому после чтения хранившаяся в ячейке информация должна быть снова в нее записана.

Емкость конденсатора в ячейке динамического ОЗУ очень мала, но подключенный к конденсатору МОП-транзистор в запертом состоянии имеет очень большое сопротивление между истоком и стоком. Поэтому требуется относительно большое время (несколько миллисекунд) для того, чтобы конденсатор разрядился настолько, что имевшееся на нем напряжение высокого уровня упало до значения, соответствующего низкому уровню. В течение этого времени конденсатор хранит один бит информации.

Работать с устройством, содержащим такую память, которую приходилось бы перезагружать из-за потери информации каждые несколько миллисекунд, не реально. Поэтому в системах памяти на основе динамических ОЗУ для обновления данных в

каждой ячейке предусмотрены периодически повторяющиеся циклы *регенерации*. В первых динамических ОЗУ регенерация производилась каждые четыре миллисекунды. Цикл регенерации включает последовательно выполняемые операции чтения нескольких ухудшенного содержимого каждой ячейки в D-защелку и повторной записи полноценного значения логического сигнала из защелки в ячейку. На рис. 9.15 показано напряжение на конденсаторе в ячейке памяти после записи и последующих циклов регенерации.

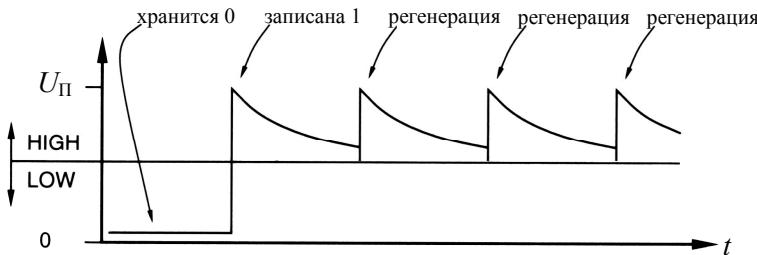


Рис. 9.15. Напряжение на конденсаторе в ячейке динамического ОЗУ после записи и выполнения циклов регенерации  
(Н - высокий уровень, Л - низкий уровень)

Первые динамические ОЗУ, появившиеся в начале 1970-х годов, содержали только 1024 ячейки, а емкость современных динамических ОЗУ достигает нескольких гигабит. Если бы требовалось обновлять все ячейки по очереди каждые четыре миллисекунды, то возникли бы проблемы: время, отводимое на регенерацию содержимого одной ячейки, было бы гораздо меньше 1 нс, и не оставалось бы никакого времени для выполнения полезных операций чтения и записи. К счастью, как будет показано ниже, динамические ОЗУ организованы в виде двумерных матриц, и за одну операцию регенерируется целая строка матрицы. У первых динамических ОЗУ было 256 строк, и требовалось 256 циклов регенерации каждые четыре миллисекунды, то есть цикл регенерации очередной строки должен был выполняться примерно через каждые 15.6 мкс. Современные матрицы памяти состоят из 4096 строк и их содержимое необходимо обновлять только

один раз за 64 мс, так что по-прежнему цикл регенерации очередной строки должен производиться каждые 15.6 мкс. В типичном случае длительность цикла регенерации составляет менее 100 нс, так что динамическое ОЗУ доступно для полезных операций чтения и записи в течение более чем 99% времени.

На рис. 9.16 приведена внутренняя структура динамического ОЗУ  $64K \times 1$ . Емкость логической матрицы составляет  $64K \times 1$  битов, но физически матрица представляет собой квадрат, состоящий из  $256 \times 256$  ячеек. Несмотря на то, что память содержит 64К ячеек, микросхема имеет только восемь *мультиплексированных адресных входов*. Полный 16-разрядный адрес поступает в микросхему за два шага по двум сигналам управления: *по стробу адреса строки RAS (row address strobe)* и *по стробу адреса столбца CAS (column address strobe)*. Благодаря мультиплексированию адресных входов удается сократить число выводов, что важно для компактной реализации запоминающих устройств, и, кроме того, мультиплексирование совершенно естественным образом согласуется с двухступенчатыми методами доступа к динамическому ОЗУ, которые вскоре будут описаны.

Современные динамические ОЗУ представляют собой большие матрицы, и часто состоят из нескольких матриц. Одно из достоинств применения нескольких матриц заключается в простоте решения электрических и физических проблем, которые возникают при проектировании матриц очень больших размеров. Но еще более важным является параллелизм, становящийся возможным при наличии нескольких матриц. Благодаря наличию в больших быстродействующих динамических ОЗУ нескольких матриц, современный контроллер динамического ОЗУ может выполнять параллельно несколько операций, например, завершать цикл записи в одной матрице, инициализируя цикл чтения в другой. В результате этого суммарный коэффициент использования памяти повышается.

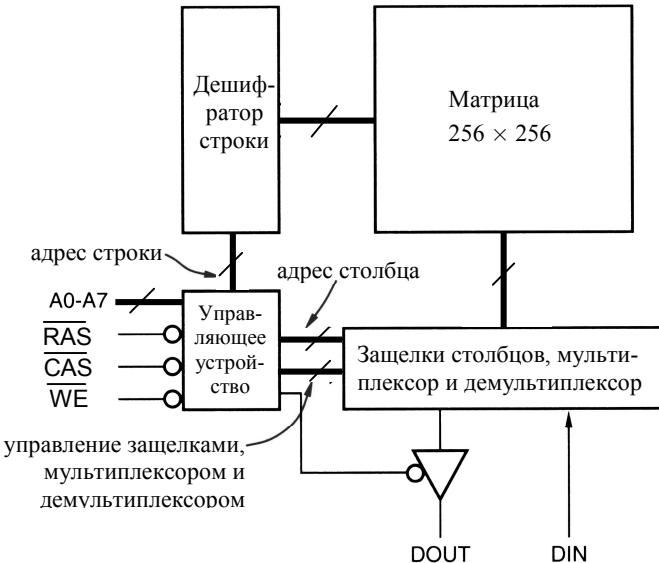


Рис. 9.16. Внутренняя структура динамического ОЗУ 64К × 1

#### 9.4.2. Временные параметры динамического ОЗУ

Существует много различных временных сценариев работы динамических ОЗУ разного типа. В этом разделе мы рассмотрим наиболее общие циклы работы обычного динамического ОЗУ и укажем на их связь с внутренней структурой устройства. Самое замечательное свойство динамического ОЗУ состоит в том, что отсутствует тактовый сигнал. Операции в динамическом ОЗУ начинаются на спадающем фронте сигналов  $\overline{\text{RAS}}$  и  $\overline{\text{CAS}}$  и заканчиваются на нарастающем фронте этих сигналов. Чтобы реализовать это, необходимы серьезные временные ухищрения, но промышленность справляется с такого рода затруднениями.

На рис. 9.17 приведены временные диаграммы цикла регенерации только по строке адреса строки. Этот цикл выполняется для обновления строки памяти фактически без чтения или записи каких-либо данных. Цикл начинается в тот момент, когда на мультиплексированных адресных входах (восемь битов в случае

динамического ОЗУ  $64K \times 1$ ) присутствует адрес строки и сигнал  $\overline{RAS}$  переходит на активный уровень.

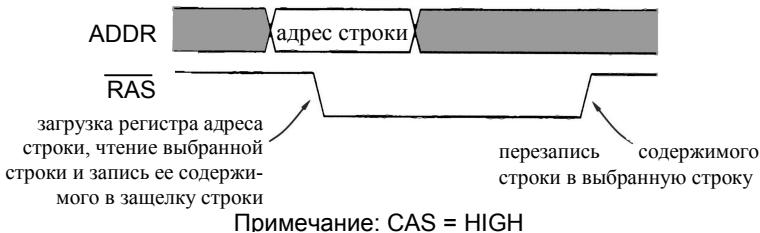


Рис. 9.17. Временные диаграммы цикла регенерации только по стробу адреса строки (HIGH – высокий уровень)

На спадающем фронте сигнала  $\overline{RAS}$  во внутренний *регистр адреса строки* записывается адрес строки, и в *защелку строки* на кристалле считывается выбранная строка матрицы памяти. Когда сигнал  $\overline{RAS}$  переходит на неактивный уровень, содержимое строки из защелки строки перезаписывается в память. Для обновления содержимого всего динамического ОЗУ емкостью  $64K \times 1$  разработчик системы должен позаботиться о том, чтобы каждые четыре миллисекунды выполнялись 256 таких циклов со всеми 256 возможными адресами строк. Для генерирования адреса строки можно применить внешний 8-разрядный счетчик, а для инициализации цикла регенерации каждые 15.6 мкс используется таймер.

*Цикл чтения*, показанный на рис. 9.18, начинается аналогично циклу регенерации, при этом содержимое выбранной строки считывается в защелку строки.

Затем на мультиплексированные адресные входы подается адрес столбца, который записывается во внутренний *регистр адреса столбца* по спадающему фронту сигнала  $\overline{CAS}$ . Адрес столбца используется для выбора одного бита только что прочитанной строки, который появляется на выводе DOUT динамического ОЗУ. Пока сигнал  $\overline{CAS}$  имеет активный уровень, выход DOUT с тремя состояниями открыт. Тем временем, как только

сигнал  $\overline{\text{RAS}}$  переходит на неактивный уровень, содержимое всей строки переписывается обратно в матрицу.



Рис. 9.18. Временные диаграммы цикла чтения из динамического ОЗУ (valid – установленившееся, достоверное значение)

Цикл записи, показанный на рис. 9.19, также начинается подобно циклу регенерации и чтения.

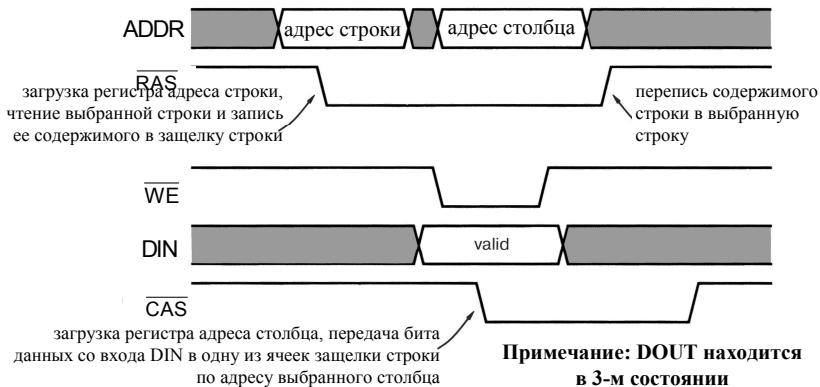


Рис. 9.19. Временные диаграммы цикла записи в динамическом ОЗУ (valid – установленившееся значение)

Однако для того, чтобы выполнить цикл записи, сигнал разрешения записи  $\overline{\text{WE}}$  (write enable) должен перейти на активный уровень прежде, чем будет установлен активный уровень

сигнала  $\overline{\text{CAS}}$ . Единственное, ради чего это делается, состоит в том, чтобы запереть выход DOUT на всю оставшуюся часть цикла, несмотря на то, что впоследствии сигнал  $\overline{\text{CAS}}$  переходит на активный уровень. Как только выбранная строка будет считана в защелку строки, бит, имеющийся на входе DIN, по сигналу  $\overline{WE}$  записывается в ту ячейку в защелке строки, которая выбрана адресом столбца. Затем, когда содержимое строки переписывается в матрицу по нарастающему фронту сигнала  $\overline{RAS}$ , в выбранном столбце этой строки присутствует новое значение.

В типичных динамических ОЗУ возможны циклы и другого типа, не показанные на рисунке:

*Цикл регенерации по стробу адреса столбца, предшествующий циклу регенерации по стробу адреса строки.* В этом цикле осуществляется регенерация без подачи адреса строки от внешнего счетчика. Вместо этого используется внутренний счетчик адреса строки, имеющийся в самом динамическом ОЗУ. Если активный уровень сигнала  $\overline{\text{CAS}}$  устанавливается раньше, чем активный уровень сигнала  $\overline{RAS}$ , то в динамическом ОЗУ регенерируется строка, определяемая содержимым внутреннего счетчика, и затем оно увеличивается на единицу. Такая возможность упрощает разработку систем с динамической памятью: пропадает необходимость во внешнем счетчике регенерации и число мультиплексируемых источников, от которых поступают сигналы на адресные входы динамического ОЗУ, сокращается с трех (строка, столбец, регенерация) до двух.

*Цикл чтение-модификация-запись* начинается подобно обычному циклу чтения, при котором данные появляются на выходе DOUT, когда сигнал  $\overline{\text{CAS}}$  переходит на активный уровень. Однако затем, для того чтобы в то же самое место записать новые данные, может быть установлен активный уровень сигнала  $\overline{WE}$ .

*Цикл постраничного чтения* позволяет прочесть целую строку (страницу) данных без повторения полного цикла RAS-CAS. Когда в защелке строки уже хранится содержимое

целой строки, для выполнения этого цикла просто требуется многократное повторение сигнала  $\overline{\text{CAS}}$  в виде импульсов низкого уровня, в то время как сигнал  $\overline{\text{RAS}}$  постоянно остается на активном уровне. На каждом спадающем фронте сигнала  $\overline{\text{CAS}}$  формируется новый адрес столбца и на выходе DOUT появляется новый бит. Данный цикл обеспечивает намного более быстрый доступ к памяти при последовательном чтении из соседних ячеек, то есть из ячеек со следующими друг за другом адресами; такой доступ к памяти часто осуществляется в микропроцессорных системах при выборке команд и при заполнении кэш-памяти.

Цикл постраничной записи аналогичен циклу постраничного чтения: он позволяет записать несколько битов строки, по одному сигналу  $\overline{\text{RAS}}$  при многократном повторении сигнала  $\overline{\text{CAS}}$ .

#### 9.4.3. Синхронные динамические ОЗУ

Протокол доступа по фронту сигналов RAS/CAS обычных динамических ОЗУ не только сложен; трудно заставить эту память работать быстро и укладываться во временные границы при связи с остальными блоками системы. В результате в начале 1990-х годов появились *синхронные динамические ОЗУ (SDRAM)*, в которых применен более традиционный синхронный интерфейс, а к концу XX века эти устройства стали доминирующими.

В синхронных динамических ОЗУ сохранен мультиплексный принцип адресации обычных динамических ОЗУ. Однако значения сигналов управления в синхронном динамическом ОЗУ, так же, как и значения сигналов на адресных входах, фиксируются только на нарастающем фронте общего тактового сигнала CLK с частотой 133 МГц и выше. Кроме того, в синхронных динамических ОЗУ введен сигнал разрешения тактового сигнала CKE: если он имеет неактивный уровень, то другие сигналы управления и сигналы адреса игнорируются. При записи значения данных принимаются во внимание в момент прохождения фронта

тактового сигнала, и при чтении данные поступают на выход по фронту тактового сигнала.

Так же, как и у обычного динамического ОЗУ, для реализации той или иной операции внутри синхронной памяти требуется выполнить определенное число шагов, а это занимает несколько тактов внешнего тактового сигнала. Синхронное устройство памяти состоит из нескольких банков динамического ОЗУ, как правило, четырех, в которых может осуществляться одновременного несколько операций.

В каждом периоде тактового сигнала сигналы управления  $\overline{\text{RAS}}$ ,  $\overline{\text{CAS}}$  и  $\overline{\text{WE}}$  интерпретируются синхронным динамическим ОЗУ как командное слово, а не как отдельные управляющие воздействия. В то же время старшие адресные биты воспринимаются ОЗУ как выбор банка: они указывают, к какому банку относится команда. Например, контроллер синхронного динамического ОЗУ может использовать четыре тактовых импульса, для того чтобы инициализировать операции чтения в четырех различных банках, а затем вернуться к первому из них и считать готовые результаты, затрачивая по одному такту на каждый банк.

Внутренняя синхронизация в синхронном динамическом ОЗУ осуществляется внешним тактовым сигналом, подаваемым на вход  $\text{CLK}$ . Обычно сигнал  $\text{RAS}$ , поступающий на внутреннюю матрицу, переходит на активный уровень немедленно после прохождения фронта синхросигнала, следующего за подачей команды чтения или команды записи. Для обеспечения внутренней синхронизации, внутренний сигнал  $\text{CAS}$  в микросхеме вырабатывается позднее, на сколько – зависит от частоты тактового сигнала  $\text{CLK}$  и быстродействия самой микросхемы памяти. Интервал времени между сигналами  $\text{RAS}$  и  $\text{CAS}$ , называемый  $\text{CAS}$ -задержкой, делается программируемым. Величину этой задержки и несколько других важных рабочих параметров необходимо загружать в синхронное динамическое ОЗУ при инициализации. Загрузка довольно проста: устройство памяти распознает команду загрузка параметров, когда одновременно переходят на активный уровень управляющие сигналы  $\overline{\text{RAS}}$ ,  $\overline{\text{CAS}}$  и  $\overline{\text{WE}}$ , а сами параметры поступают на адресные линии.

# ГЛАВА 10. ПРОГРАММИРУЕМЫЕ МИКРОСХЕМЫ ТИПА CPLD И FPGA

С момента своего появления несколько лет назад программируемые логические устройства (ПЛУ) стали основным, очень гибким инструментом цифрового проектирования. По мере развития технологии ИС, естественно, возрастал интерес к созданию ПЛУ с более развитой архитектурой, что позволило бы воспользоваться достоинствами повышенной плотности упаковки в кристалле.

## 10.1. Интегральные схемы типа CPLD

Эта идея была использована в *сложных программируемых логических устройствах* (*complex programmable logic device*, *CPLD*). Как показано на рис. 10.1, ИС типа *CPLD* является всего лишь совокупностью отдельных ПЛУ на одном кристалле с программируемой структурой взаимных связей, которая позволяет отдельным ПЛУ в пределах кристалла подключаться друг к другу так же, как это мог бы сделать опытный разработчик с отдельными ПЛУ вне кристалла. Здесь площадь кристалла, необходимая для реализации увеличенной в  $n$  раз логики, равна площади только  $n$  одиночных ПЛУ плюс площадь, занимаемая программируемой структурой взаимных связей.

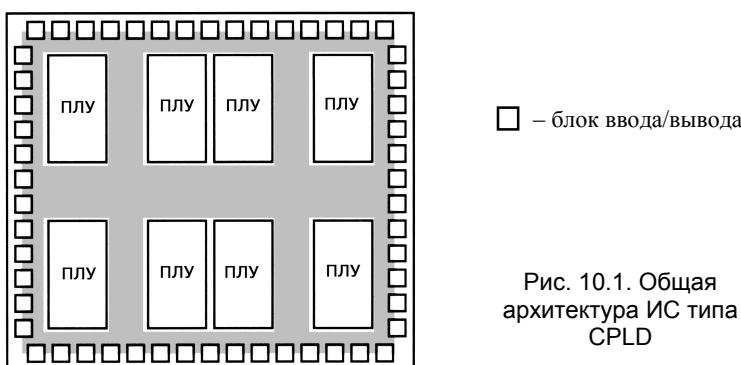


Рис. 10.1. Общая архитектура ИС типа CPLD

Разные производители перепробовали много различных вариантов общей архитектуры, показанной на рисунке. Эти варианты отличаются блоками ПЛУ (состоящими из матрицы И и макроячеек), блоками ввода/вывода и структурой программируемых соединений. Мы рассмотрим эти составляющие, воспользовавшись в качестве примера архитектурой ИС типа *CPLD* серии 9500 фирмы *Xilinx*.

### 10.1.1. Семейство ИС XC9500 фирмы Xilinx

Микросхемы XC9500 фирмы *Xilinx* представляют собой семейство ИС типа *CPLD* одинаковой архитектуры, но с различным числом внешних I/O-выводов и с разным числом внутренних ПЛУ, которые фирма *Xilinx* называет *функциональными блоками (FB)*. Каждое внутреннее ПЛУ содержит 18 макроячеек, имеющих 36 входов и 18 выходов. Маркировка микросхем определяется числом имеющихся в них макроячеек. Самый маленький представитель этого семейства XC9536 содержит 2 функциональных блока с 36 макроячейками, а самый большой XC95288 – 16 функциональных блоков с 288 макроячейками.

В большинстве случаев не требуется, чтобы все внутренние сигналы конечного автомата или подсистемы были видимы остальной частью системы и использовались ею.

Так, ИС XC95108 содержит 108 внутренних макроячеек, но при ее размещении в корпусе с 84 выводами наружу могут быть выведены выходы самое большее 69 макроячеек. На самом деле, как правило, большинство из 69 I/O-выводов используются как входы, поэтому извне будет доступно еще меньшее число выходов. И это правильно: остальные выходы макроячеек вполне можно использовать внутри, так как к ним можно подключиться внутри через структуру программируемых соединений. Макроячейки, выходы которых доступны только внутри, иногда называют *скрытыми макроячейками*.

На рис. 10.2 приведена блок-схема внутренней архитектуры типичной ИС типа *CPLD* из семейства XC9500.

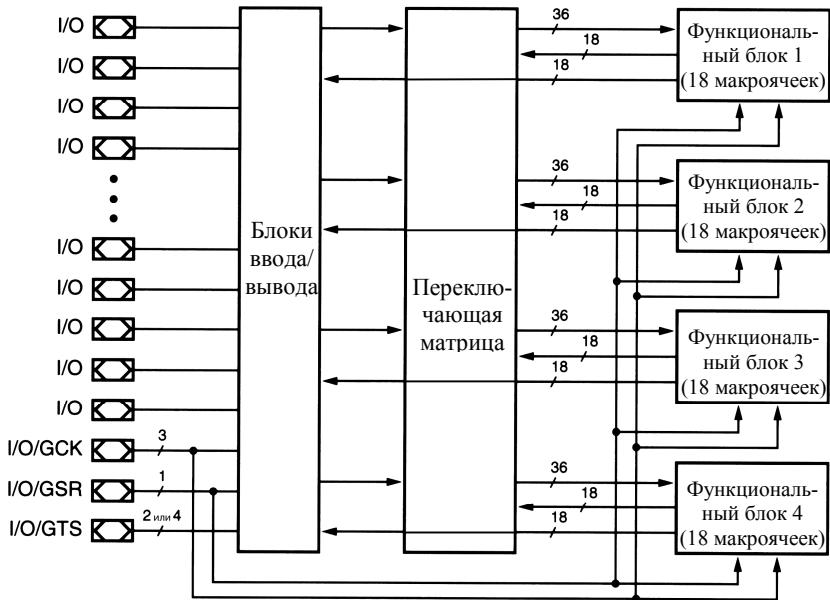


Рис. 10.2. Архитектура ИС типа CPLD семейства 9500 фирмы Xilinx

Каждый внешний I/O-вывод можно использовать в качестве входа, выхода или двунаправленного вывода в соответствии с тем, как запрограммировано устройство. Выводы, расположенные в нижней части рисунка, можно использовать также для некоторых специальных целей. На любой из трех выводов GCK можно подавать общие тактовые сигналы; каждую макроячейку можно запрограммировать так, чтобы на нее поступал тактовый сигнал с выбранного входа. Один вывод GSR можно использовать для подачи сигнала общая установка/сброс, и каждую макроячейку можно запрограммировать так, чтобы с помощью этого сигнала производилась асинхронная предварительная установка или сброс. Наконец, на любой из двух или из четырех выводов GTS (в зависимости от типа устройства) можно подавать сигнал, осуществляющий общее управление третьим состоянием; в каждой макроячейке можно выбрать один из этих сигналов для отпи-

рания или запирания соответствующего выхода, когда выход макроячейки подключен к внешнему I/O-выводу.

На рисунке показаны только четыре функциональных блока, но архитектура семейства XC9500 допускает наличие в ИС XC95288 16 функциональных блоков. Независимо от особенностей микросхемы, входящей в состав этого семейства, на входы каждого функционального блока путем программирования переключающей матрицы подаются 36 сигналов. На входы переключающей матрицы поступают сигналы с 18 выходов макроячеек от каждого функционального блока и внешние входные сигналы с I/O-выводов.

Кроме того, у каждого функционального блока есть 18 выходов, сигналы на которых проходят мимо переключающей матрицы, как показано на рис. 10.2, и поступают на блоки ввода/вывода. Это просто сигналы разрешения выхода для выходных каскадов блока ввода/вывода; эти сигналы действуют в том случае, когда выход макроячейки данного функционального блока подключен к внешнему I/O-выводу.

Каждая макроячейка содержит один триггер и разветвленную логику. Триггер макроячейки можно запрограммировать для работы в качестве D-триггера или в качестве T-триггера с входом разрешения счета; последний вариант полезен при реализации счетчиков того или иного типа. С помощью мультиплексора выбирается сигнал, подаваемый на тактовый вход триггера; этим сигналом может быть один из четырех входных сигналов мультиплексора: один из трех общих тактовых сигналов на входах ИС или их произведение.

У триггера есть также входы асинхронной установки в единичное состояние и сброса. Выбор сигналов, подаваемых на эти входы, осуществляется мультиплексорами. В большинстве случаев входы установки и сброса бывают соединены с общим входом установка/сброс данной ИС и используются только при начальном запуске системы. Однако по этим входам можно также получить доступ к SR-защелке, у которой вход CLK не используется.

В качестве выходного сигнала макроячейки OUT с помощью еще одного мультиплексора выбирается сигнал с выхода

триггера или сигнал, поступающий на его вход данных. Выходной сигнал OUT поступает на переключающую матрицу, где он может быть использован любой другой макроячейкой. Он может быть отправлен также к блокам ввода/вывода.

### 10.1.2. Архитектура блока ввода/вывода

Блок ввода/вывода микросхем семейства XC9500 является наглядным примером тенденции в архитектуре блоков ввода/вывода микросхем типа *CPLD* и *FPGA*: в нем, кроме управления логическими действиями вроде разрешения выхода, имеется возможность изменять многие *аналоговые* параметры. В блоке ввода/вывода можно изменять значения трех аналоговых параметров:

- *Управление скоростью изменения выходного напряжения.* Для того чтобы получать быстродействующее или медленно работающее устройство, можно задавать время нарастания и спада выходных сигналов. Установка наибольшего быстродействия обеспечивает наименьшую возможную задержку распространения, в то время как задание режима медленной работы устройства позволяет уменьшить помехи в линии передачи и шумы в системе за счет небольшой дополнительной задержки.
- *Включение резистора нагрузки между выходом и шиной питания.* Когда резистор нагрузки включен, он предотвращает появление на выходном выводе плавающего напряжения при подаче на микросхему напряжения питания. Это полезно в том случае, когда выходные сигналы поступают на входы разрешения других логических устройств с низким активным уровнем, в отношении которых не предполагается, что в момент включения питания на них будет подан сигнал, имеющий активное значение.
- *Образование программируемых пользователем выводов земли.* Эта возможность фактически позволяет перераспределить I/O-выводы так, чтобы те или иные выводы были выводами земли, а вовсе не сигнальными выводами. Это оказывается полезным в быстродействующих устройствах с вы-

сокой скоростью изменения сигналов. Необходимость в дополнительных выводах земли возникает в тех случаях, когда имеют место большие броски тока, возникающие при одновременном переключении сигналов на нескольких выходах.

Кроме этих особенностей, ИС семейства XC9500 совместимы с внешними устройствами с напряжением питания 5 В и 3.3 В. Входной буфер и внутренняя логика работают от источника питания с напряжением, равным 5 вольтам. В зависимости от напряжения питания внешних устройств в выходном каскаде используется напряжение питания  $V_{\text{ПО}}$ , равное 5 В или 3.3 В.

Важной операцией при разработке ИС типа *CPLD* и программного обеспечения является *привязка выводов*. В большинстве приложений ИС типа *CPLD* считается нормальным разрешить программе компоновки выбирать любые возможные выводы в качестве внешних входов и выходов данного устройства. Но когда проект закончен и изготовлена печатная плата, разработчик может пожелать зафиксировать назначение выводов, так чтобы они оставались теми же самыми при небольших (или даже при больших) изменениях, связанных с исправлением ошибок в проекте. Это приводит к экономии времени и стоимости и позволяет преодолеть препятствия, возникающие при переработке или доработке и переделке печатной платы.

Желаемая привязка выводов обычно указывается в файле, который читается программой компоновки. У первых ИС типа *CPLD* и *FPGA* привязка выводов до выполнения даже небольших изменений не гарантировала успеха: программа сообщала, что имеется слишком много ограничений. Если разблокировать назначение выводов, то программа компоновки, возможно, найдет новое распределение, которое будет работать, но оно может оказаться совершенно не похожим на исходное.

Эти проблемы вовсе не обязательно возникали по вине программы компоновки; просто у ИС типа *CPLD* и *FPGA* не было достаточного количества внутренних связей, чтобы выдерживать постоянные изменения проекта при условии привязки выводов. В дальнейшем схемы *CPLD* были усовершенствованы, и стали возможны изменения в процессе разработки проекта. В ре-

зультате некоторые микросхемы теперь имеют, например, выходную переключающую матрицу, которая гарантирует возможность соединения любого входа или выхода макроячейки, находящейся внутри данной ИС, с любым внешним I/O-выводом

## 10.2. Интегральные схемы типа FPGA

*Программируемая в условиях эксплуатации матрица вентилей (field-programmable gate array, *FPGA*)* напоминает *CPLD*, вывернутую изнутри наружу. Как показано на рис. 10.3, на кристалле расположено большое число программируемых логических блоков, каждый из которых меньше, чем ПЛУ. Они распределены по всему кристаллу среди программируемых соединений, а вся матрица окружена блоками ввода/вывода.

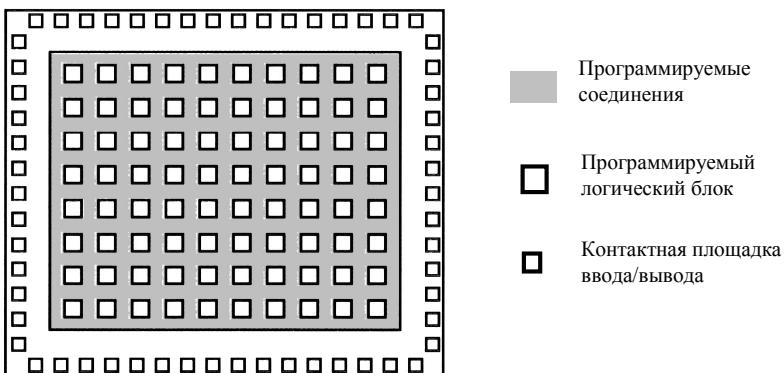


Рис. 10.3. Общая архитектура кристалла ИС типа *FPGA*

Программируемый логический блок ИС типа *FPGA* обладает меньшими возможностями, чем типичное ПЛУ, но одна микросхема типа *FPGA* содержит гораздо больше логических блоков, чем ИС типа *CPLD* при том же самом размере кристалла.

Микросхемы типа *FPGA* были разработаны фирмой *Xilinx, Inc.* в 1998 году. Для иллюстрации архитектуры ИС типа *FPGA* мы воспользуемся одним из популярных семейств этой фирмы – семейством *SPARTAN* II. Семейства *SPARTAN*-II и *SPARTAN*-IIЕ

– второе поколение ПЛИС (Программируемые Логические Интегральные Схемы). ПЛИС семейства SPARTAN-II(E) применяются в проектах как альтернатива специализированным интегральным схемам емкостью до 600000 вентилей и системным быстродействием до 200 МГц. Напряжение питания ядра кристалла семейства SPARTAN-II(E) составляет 2.5 В.

Семейство SPARTAN-II состоит из 6 типов кристаллов, а семейство SPARTAN-II(E) содержит 7 типов кристаллов, отличающихся логической емкостью. Сравнительные параметры некоторых ИС приведены в табл. 10. 1.

Таблица 10.1

**Основные характеристики семейства Spartan-II(E).**

Кристалл	Логические Ячейки (LC)	Системные вентили	Матрица CLB	CLB	Блочная ОЗУ, Бит	Пользовательские блоки ввода-вывода (IOB), max
XC2S15	432	15000	8 × 12	96	16384	86
XC2S30	972	30000	12 × 18	216	24576	132
....	....	....	....	....	....	....
XC2S600E	15552	600000	48 × 72	3 456	294912	514

### 10.2.1. ПЛИС семейств SPARTAN-II и SPARTAN-II(E)

Программируемые логические схемы семейства SPARTAN-II это:

- Высокопроизводительные, программируемые пользователем логические интегральные схемы с архитектурой *FPGA* (*Field Programmable Gate Arrays*).

Емкость от 15 000 до 600 000 системных вентилей.

Системная производительность до 200 МГц.

Совместимость с шиной PCI 66 МГц.

Поддержка большинства стандартов ввода-вывода (технология *SelectIO*).

16 высокопроизводительных стандартов ввода-вывода.

Размещение ИС в наиболее дешевые корпуса.

Совместимость по выводам ИС разной емкости в одинаковых корпусах.

- Встроенные цепи управления тактовыми сигналами

Четыре встроенных модуля автоподстройки задержек (*DLL - delay-locked loop*) для расширенного управления тактовыми сигналами внутри ПЛИС и в устройстве в целом.

Четыре глобальные сети распределения тактовых сигналов с малыми отклонениями фронтов, плюс 24 локальные тактовые сети.

- Иерархическая система элементов памяти

На базе 4-входовых таблиц преобразования (*4-LUT – Look Up Table*) конфигурируемых либо как 16-ти битовая *RAM*, либо как 16-ти битовая двухпортовая *RAM*, либо как 16-ти битовый сдвиговый регистр.

- Встроенная блочная память, каждый блок конфигурируется как синхронная двухпортовая *RAM* емкостью 4 Кбит.

- Быстрые интерфейсы для подключения к внешней высокопроизводительной *RAM*.

- Гибкая архитектура, позволяющая сбалансировать быстродействие и плотность упаковки.

- Специальная логика ускоренного переноса для высокоскоростных арифметических операций.

- Каскадируемые цепочки для функций с большим количеством входов.

- Многочисленные регистры/защелки с разрешением тактирования и синхронно/асинхронные цепи установки и сброса.

- Внутренние шины с тремя состояниями.

- Логика периферийного сканирования в соответствии со стандартом *IEEE 1149.1*.

Проектирование микросхем осуществляется пакетами программного обеспечения *ISE* или *WEBPack ISE*, устанавливаемыми на ПК

Конфигурация кристалла хранится во внешнем ПЗУ и загружается в ПЛИС после включения питания автоматически или принудительно. Допускается практически неограниченное число циклов загрузки.

## **Процесс конфигурации**

Конфигурационная последовательность (*bitstream*) может загружаться в кристалл непосредственно в системе практически неограниченное число раз. Инициализация ИС *FPGA* производится автоматически (из загрузочной флэш-памяти *Xilinx*) при подаче напряжения питания или принудительно по специальному сигналу. В зависимости от емкости ИС процесс инициализации занимает от 20 до 900 мс, в течение которых выводы ИС находятся в высокоомном состоянии и через резисторы подключены к напряжению источника питания.

## **Потребление энергии**

Статическое потребление энергии достаточно мало и составляет единицы микроватт. Динамическое же потребление пропорционально возрастает с частотой работы устройства и зависит от степени заполнения кристалла, характера логической структуры проекта на кристалле, параметров режима внешних выводов ИС и т. д.

*FPGA* фирмы *Xilinx* выполнены по *SRAM* кМОП технологии. Они характеризуются высокой гибкостью структуры и большим числом триггеров на кристалле. При этом логика реализуется посредством так называемых *LUT*-таблиц (*Look Up Table*), а внутренние межсоединения – посредством разветвлённой структуры металлических линий, коммутируемых специальными быстродействующими транзисторами.

Отличительными особенностями ИС *FPGA* являются:

- Внутренние буфера с возможностью переключения в высокоомное состояние и тем самым позволяющие организовать системные двунаправленные шины.
- Индивидуальный контроль высокоомного состояния и времени нарастания фронта выходного сигнала по каждому внешнему выводу.
- Наличие общего сброса/установки всех триггеров, имеющихся в составе микросхемы.
- Множество глобальных линий с низкими задержками распространения сигнала.

- Наличие внутреннего распределённого ОЗУ, реализующегося посредством тех же *LUT*-таблиц.
- Наличие внутреннего блочного ОЗУ, один блок имеет емкость 4 кбит, количество блоков достигает 72 на кристалл.

Программируемые логические блоки в ИС типа *FPGA* семейства SPARTAN-II и SPARTAN-II~~E~~ фирмы *Xilinx* названы *переconfigурируемыми логическими блоками* (*CLB – Configurable Logic Block*). Наименьшая микросхема семейства SPARTAN-II *XC2S15* содержит матрицу логических блоков размером  $8 \times 12$ , а в самой большой ИС семейства SPARTAN-II~~E~~ *XC2S600* – 3456 логических блока в виде матрицы размером  $48 \times 72$ .

Подобно семейству *CPLD*, семейство SPARTAN-II включает набор микросхем разных размеров с разным числом I/O-выводов. В справочных данных (табл. 10.1) указывается максимальное число доступных пользователю блоков ввода/вывода, имеющихся в микросхеме. Однако ИС серии SPARTAN-II размещают в различных корпусах и в случае корпуса меньших размеров не все имеющиеся входы/выходы выведены на внешние контакты. Как и в случае с ИС типа *CPLD*, пользователь микросхем типа *FPGA* имеет возможность перенести свой проект, выполненный на основе небольшой ИС, в микросхему большего объема, размещенную в том же самом корпусе, и наоборот.

### 10.2.2. Архитектура SPARTAN-II

Основными особенностями архитектуры кристаллов семейства SPARTAN-II являются гибкость и регулярность. Кристаллы содержат матрицы конфигурируемых логических блоков *CLB*, которые окружены программируемыми блоками ввода-вывода *IOB*. Все соединения между основными элементами *CLB*, *IOB* осуществляются с помощью набора иерархических высокоскоростных программируемых трассировочных ресурсов. Обилие таких ресурсов позволяет реализовывать на *FPGA* семейства SPARTAN-II даже самые сложные проекты.

Кристаллы семейства SPARTAN-II производятся на основе статического ОЗУ (*SRAM*), поэтому их функционирование определяется загружаемыми во внутренние ячейки памяти конфигурационными данными. Конфигурационные данные могут загружаться в кристалл несколькими способами. В ведущем последовательном режиме *Master Serial* загрузка осуществляется из внешнего ПЗУ и полностью управляет самой *FPGA* SPARTAN-II. В других режимах управление загрузкой осуществляется внешними устройствами (подчиненный параллельный режим *Slave Parallel*, подчиненный последовательный *Slave Serial* и *JTAG*).

Конфигурационные данные создаются пользователем при помощи программного обеспечения проектирования *ISE* или *WebPACK ISE*, разработанного фирмой *Xilinx*. Программное обеспечение включает в себя модули схемного и текстового ввода, моделирования, автоматического и ручного размещения и трассировки, создания, загрузки и верификации конфигурационных данных.

## **Быстродействие**

Кристаллы SPARTAN-II обеспечивают более высокую производительность, чем предыдущие поколения *FPGA*. Проекты могут работать на системных частотах до 200 МГц и частотах внутри кристалла, превышающих 350 МГц. Блоки ввода-вывода SPARTAN-II полностью соответствуют спецификациям *PCI* шины, поэтому микросхемы позволяют реализовывать интерфейсные схемы, работающие на частоте 33 МГц или 66 МГц.

## **10.3. Описание структуры ИС SPARTAN-II**

### **10.3.1. Матрица SPARTAN-II**

Структура программируемой пользователем вентильной матрицы кристалла серии SPARTAN-II показана на рис.10.4.

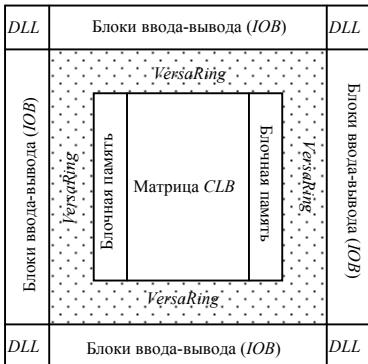


Рис.10.4: Структура FPGA Spartan-II

Основными программируемыми элементами матрицы являются:

конфигурируемые логические блоки (*CLB*), на основе которых реализуется вся логика,

блоки ввода-вывода (*IOB*), осуществляющие связь между контактами микросхемы и *CLB*.

Соединение между *CLB* осуществляется с помощью главных трассировочных матриц - *GRM* (*General Routing Matrix*). *GRM* - это матрица программируемых транзисторных двунаправленных переключателей, расположенных на пересечении горизонтальных и вертикальных линий связи. Каждый *CLB* окружен локальными линиями связи (*VersaBlock*), которые позволяют осуществить соединения с матрицей *GRM*.

Интерфейс ввода-вывода *VersaRing* создает дополнительные трассировочные ресурсы по периферии кристалла. Эти трассы улучшают общую структуру соединений в устройстве и возможности трассировки после закрепления электрических цепей к конкретным контактам.

Архитектура SPARTAN-II включает также следующие элементы, которые соединяются с матрицей *GRM*:

- Специальные блоки памяти *SRAM* размером 4096 бит каждый.
- Четыре модуля автоподстройки задержек *DLL*, предназначенных для компенсации задержек тактовых сигналов, а

также деления, умножения и сдвига фазы тактовых частот.

- Буферы с тремя состояниями *BUFT*, которые расположены вблизи каждого *CLB* и управляют горизонтальными сегментированными трассами.

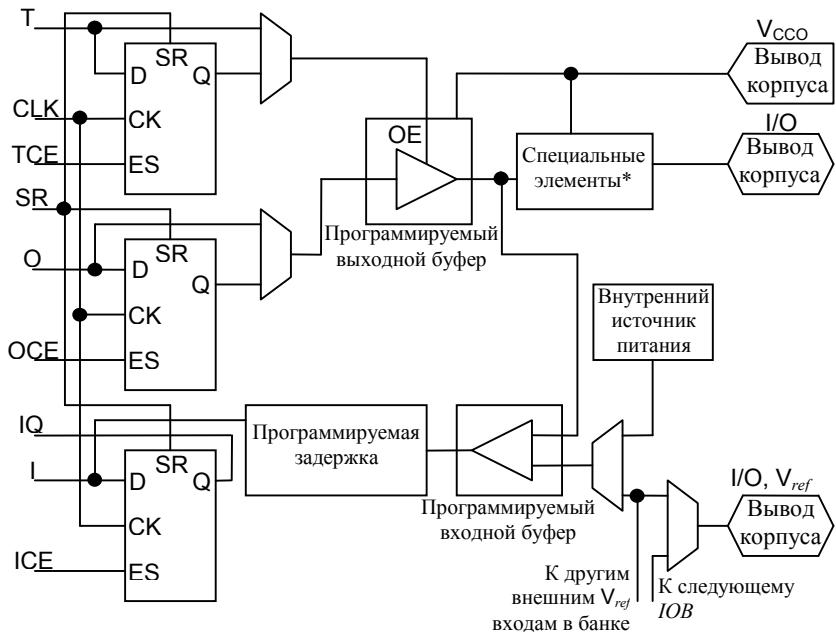
Коды, записанные в ячейки статической памяти, управляют настройкой логических элементов и коммутаторами трасс, осуществляющих межсоединения в схеме. Эти коды загружаются в ячейки после включения питания и могут перезагружаться в процессе работы, если необходимо изменить реализуемые микросхемой функции.

### 10.3.2. Блок ввода-вывода

Основным отличительным свойством блоков *IOB* семейства SPARTAN-II является поддержка широкого спектра стандартов сигналов ввода-вывода, что позволяет сопрягать SPARTAN-II с большинством быстродействующих элементов памяти и шинных интерфейсов. На рис. 10.5 представлена структурная схема *IOB*. В табл. 10.2. перечислены поддерживаемые стандарты.

*IOB* содержит три запоминающих элемента, функционирующих либо как D-триггеры, либо как триггеры-защелки. Каждый *IOB* имеет входной сигнал синхронизации (*CLK*), распределенный на три триггера и независимые для каждого триггера сигналы разрешения тактирования (*Clock Enable - CE*).

Кроме того, на все триггеры заведен сигнал Сброса/Установки (*Set/Reset - SR*). Для каждого триггера этот сигнал может быть сконфигурирован независимо как сигнал синхронной установки (*Set*), синхронного сброса (*Reset*), асинхронной предустановки (*Preset*) или асинхронного сброса (*Clear*).



\*Специальные элементы включают:

- Программируемый резистор, соединенный с земляной шиной (*pull-down*).
- Программируемый резистор, соединенный с шиной питания (*pull-up*).
- Маломощная схема удержания последнего состояния (*week-keeper*).
- Цепи защиты от перенапряжения и электростатического разряда

Рис. 10.5. Блок ввода-вывода ИС Spartan-II

Входные и выходные сигналы буферов, а также все управляющие сигналы в *IOB* допускают независимое инвертирование. Данное свойство не отображено на блок схеме *IOB*, но контролируется программой проектирования.

Все контакты защищены от повреждения электростатическим разрядом и от всплесков перенапряжения. Реализованы две формы защиты от перенапряжения, одна допускает 5 В совместимость, а другая нет. Для случая 5 В совместимости, структура, подобная диоду Зенера, замыкает на землю контакт, когда напряжение на нем возрастает приблизительно до 6,5 В. В случае, когда требуется 3,3 В PCI совместимость, обычные диоды огра-

ничения могут подсоединяться к источнику питания выходных каскадов,  $V_{CCO}$ . Тип защиты от перенапряжения может выбираться независимо для каждого контакта.

По выбору, к каждому контакту может подключаться:

- резистор, соединенный с земляной шиной (*pull-down*),
- резистор, соединенный с шиной питания (*pull-up*),
- маломощная схема удержания последнего состояния (*weak-keeper*).

До начала процесса конфигурирования микросхемы все выводы, не задействованные в этом процессе, принудительно переводятся в состояние высокого импеданса. *Pull-down* резисторы и элементы *weak-keeper* неактивны, а *pull-up* резисторы можно активировать.

Активация *pull-up* резисторов перед конфигурацией управляется внутренними глобальными линиями через управляющие режимные контакты. Если *pull-up* резисторы не активны, то выводы находятся в состоянии неопределенного потенциала. Если в проекте необходимо иметь определенные логические уровни до начала процесса конфигурирования, то нужно использовать внешние резисторы.

### 10.3.3. Ввод сигнала

Входной сигнал  $IOB$  может быть подан к блокам внутренней логики либо непосредственно, либо через входной триггер. Кроме того, между выходом буфера и D-входом триггера может быть подключен элемент задержки, исключающий время удержания для случая контакт–контакт. Данная задержка согласована с внутренней задержкой распределения сигнала тактирования *FPGA*, что гарантирует нулевое время удержания для распределения сигналов контакт–контакт.

Каждый входной буфер может быть сконфигурирован таким образом, чтобы удовлетворять одному из стандартов ввода–вывода, поддерживаемых устройством. В некоторых из этих стандартов входной буфер использует напряжение порогового уровня  $V_{ref}$ , формируемое пользователем. Использование напряжений  $V_{ref}$  позволяет ввести в устройство принудительные опор-

ные величины для различных, близких по используемым логическим уровням стандартов.

К каждому входу после окончания процесса конфигурирования могут быть, по выбору, подключены внутренние резисторы (либо *pull-up*, либо *pull-down*). Номинал этих резисторов лежит в пределах 50–150 КОм.

#### 10.3.4. Вывод сигнала

Выходной сигнал проходит через буфер с тремя состояниями, выход которого соединен непосредственно с контактом. Сигнал может быть подан на вход буфера с тремя состояниями либо непосредственно от внутренней логической структуры, либо через выходной триггер блока ввода-вывода

Управление буфером с тремя состояниями также может осуществляться либо непосредственно от внутренней логической структуры, либо через специальный триггер *IOB*, который позволяет создать синхронное управление сигналом разрешения и запрещения для буфера с тремя состояниями. Каждый такой выходной каскад рассчитан на втекающий ток до 48 мА и вытекающий ток до 24 мА. Программирование мощности и скорости нарастания сигнала выходного каскада позволяет минимизировать переходные процессы в шинах.

Для большинства стандартов ввода-вывода выходной уровень логической единицы зависит от приложенного извне напряжения  $V_{ref}$ . Использование напряжения  $V_{ref}$  позволяет ввести в устройство принудительные опорные величины для различных, близких по используемым логическим уровням стандартов.

По выбору, к каждому выходу может быть подключена схема *week-keeper*. Если данная цепь активирована (задаётся пользователем на этапе создания схемы), то она следит за напряжением на контакте микросхемы и создает слабую нагрузку для входного сигнала, подключенную либо к земле (если на входе уровень логического нуля), либо к источнику питания (если на входе уровень логической единицы). Если контакт подключен к нескольким источникам сигнала, эта цепь удерживает уровень входного сигнала в его последнем состоянии, при условии, что

все источники были переведены в состояние с высоким импедансом. Поддержание таким путем одного из допустимых логических уровней позволяет ликвидировать неопределенность уровня шины.

Так как схема *week-keeper* использует входной буфер для слежения за входным уровнем, то необходимо использовать подходящее значение напряжения  $V_{ref}$ , если выбранный сигнальный стандарт требует этого. Подключение данного напряжения должно удовлетворять требованиям правил разбиения на банки.

### 10.3.5. Банки ввода-вывода

Некоторые из указанных выше стандартов требуют подключения напряжения  $V_{Pi}$  и/или  $V_{ref}$ . Эти внешние напряжения подключаются к контактам микросхемы, которые функционируют группами, называемыми банками.

Как показано на рис. 10.6, каждая кромка микросхемы разделена на два банка. Каждый банк имеет несколько контактов  $V_{Pi}$ . Но все они должны быть подключены к одному и тому же напряжению. Это напряжение определяется выбранным для данного банка стандартом выходных сигналов.

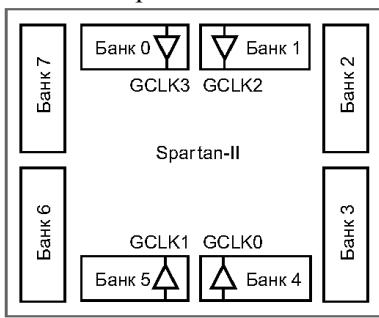


Рис. 10.6: Банки ввода-вывода SPARTAN-II

Некоторые стандарты требуют подачи соответствующих пороговых напряжений  $V_{ref}$  на входные каскады. При этом определенные *IOB* автоматически конфигурируются как входы, соответствующие напряжению  $V_{ref}$ . Приблизительно один контакт из шести в каждом банке может выполнять эту роль.

Контакты  $V_{ref}$  в пределах одного банка внутренне между собой соединены, следовательно, только одно значение напряжения  $V_{ref}$  может быть использовано в рамках одного банка. Для правильной работы все контакты  $V_{ref}$ , одного банка, должны быть подсоединенны к внешнему источнику напряжения.

В пределах одного банка можно одновременно использовать входы, которые требуют напряжения  $V_{ref}$  и входы, которые этого не требуют. Входные буферы, которые используют  $V_{ref}$ , не совместимы с сигналами 5В стандартов. *IOB*, запрограммированные на стандарты *LVTTL*, *LVCMSO2* и *PCI*, совместимы с 5 В стандартами.

Номера контактов  $V_{\Pi}$  и  $V_{ref}$  для каждого банка приведены в таблицах и диаграммах под конкретный корпус и кристалл. На диаграммах также показано к какому банку относится конкретный контакт ввода-вывода.

В рамках конкретного типа корпуса микросхемы, число контактов  $V_{\Pi}$  и  $V_{ref}$  может меняться в зависимости от емкости кристалла. Чем больше кристалл по логической ёмкости, тем большее число контактов ввода-вывода преобразовано в контакты типа  $V_{ref}$ . Поскольку для меньших кристаллов существует максимальный набор контактов  $V_{ref}$ , имеется возможность проектирования печатной платы, позволяющей также использовать на ней и большие кристаллы с таким же типом корпуса. Все контакты  $V_{ref}$ , предполагаемые к использованию для больших кристаллов, при этом, должны быть подсоединенны к напряжению  $V_{ref}$  и не должны использоваться как контакты ввода-вывода.

В меньших кристаллах некоторые из контактов  $V_{\Pi}$ , используемые в больших кристаллах, не соединены внутри корпуса. Эти не присоединенные контакты могут быть оставлены не присоединенными вне микросхемы или быть подключены к напряжению  $V_{CC0}$  при необходимости обеспечения совместимости разрабатываемой печатной платы с большими кристаллами.

В корпусах *TQ144* и *PQ208* все контакты  $V_{\Pi}$  соединены вместе внутри микросхемы и, следовательно, ко всем из них должно быть подключено одно и тоже напряжение  $V_{\Pi}$ . В корпусе *CS144* пары банков, расположенные на одной стороне, внутренне

соединены, обеспечивая, таким образом, возможность выбора только четырех возможных значений напряжения для  $V_{\Pi}$ . Контакты  $V_{ref}$  остаются внутренне соединенными в рамках каждого из восьми банков и могут использоваться, как было описано выше.

#### 10.4. Конфигурируемый логический блок CLB

Базовым элементом  $CLB$  является логическая ячейка  $LC$  (*Logic Cell*).  $LC$  состоит из 4-х входового функционального генератора, логики ускоренного переноса и запоминающего элемента. Выход каждого функционального генератора каждой логической ячейки подсоединен к выходу  $CLB$  и к D-входу триггера. Каждый  $CLB$  серии SPARTAN-II содержит четыре логические ячейки, организованные в виде двух одинаковых секций (*Slice*). Одна секция показана на рис. 10.7.

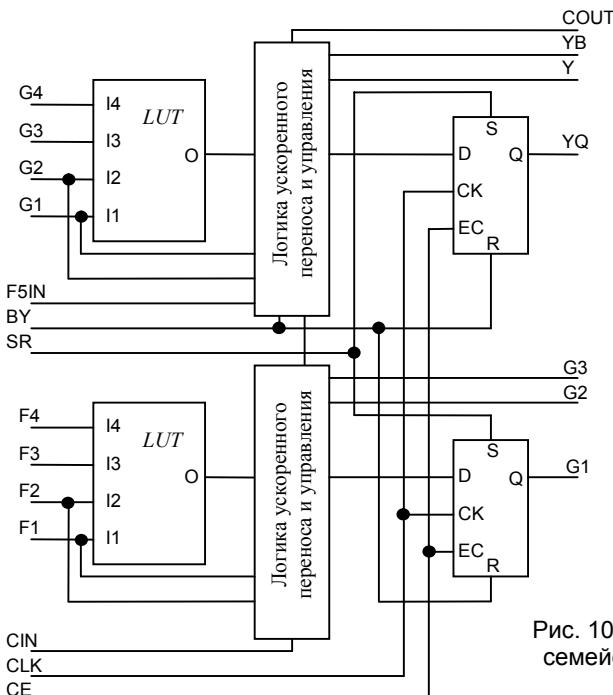


Рис. 10.7. Секция CLB семейства Spartan II

В дополнение к четырем базовым логическим ячейкам, *CLB* серии SPARTAN-II содержит логику, которая позволяет комбинировать ресурсы функциональных генераторов для реализации функций от пяти или шести переменных. Таким образом, при оценке числа эквивалентных системных вентилей для микросхем семейства *Spartan-II*, каждый *CLB* приравнивается к  $4,5\text{ LC}$ .

#### 10.4.1. Таблица преобразования LUT

Функциональные генераторы реализованы в виде 4-входовых таблиц преобразования (*Look-Up Table – LUT*), которые представляют собой статическую память  $16 \times 1$ . Для реализации той или иной логической функции в эту память записываются биты в соответствии с таблицей истинности. Кроме реализации логических функций, каждый *LUT*-элемент может быть использован как синхронная память типа *RAM* размерностью  $16 \times 1$  бит. Более того, из двух *LUT*-элементов в рамках одной секции можно реализовать синхронную *RAM*-память размерностью  $16 \times 2$  бита или  $32 \times 1$  бит, либо двухпортовую синхронную *RAM*-память размерностью  $16 \times 1$  бит. На *LUT*-элементе микросхемы SPARTAN-II может быть реализован 16-битовый сдвиговый регистр, который идеально подходит для захвата высокоскоростных или пакетных потоков данных. Этот режим используется и для запоминания данных в приложениях цифровой обработки сигналов.

#### 10.4.2. Запоминающие элементы

Запоминающие элементы в каждой секции *CLB* SPARTAN-II могут конфигурироваться как динамические триггеры (чувствительные к фронту сигнала) D-типа, либо как триггеры-защелки, чувствительные к уровню сигнала. D-вход триггера может управляться либо от функционального генератора в рамках той же секции *CLB*, либо непосредственно от входов данной секции *CLB*, минуя функциональные генераторы.

Кроме сигналов синхронизации (*Clock*) и разрешения синхронизации (*Clock Enable - CE*) в каждой секции *CLB* есть сигналы синхронной установки (*Set*) и сброса (*Reset*). Обозначение этих сигналов – *SR* и *BY* соответственно. Сигнал *SR* переводит запоминающий элемент в состояние, определённое для него в конфигурационных данных, а сигнал *BY* – в противоположное состояние. Эти же сигналы могут быть использованы также в качестве асинхронной предустановки (*Preset*) и очистки (*Clear*). Все сигналы управления могут быть независимо инвертированы. Они заведены на оба триггера в рамках конкретной секции *CLB*.

#### 10.4.3. Дополнительная логика

Дополнительная логика, входящая в каждый *CLB*, представлена двумя мультиплексорами: *F5* и *F6*.

На вход мультиплексора *F5* заведены сигналы с выходов функциональных генераторов данной секции *CLB*. Этот узел может работать как функциональный генератор, реализующий любую 5-входовую функцию, либо как мультиплексор 4:1, либо как некоторая функция от девяти входных переменных.

Аналогично, мультиплексор *F6* объединяет выходы всех 4 функциональных генераторов *CLB*, используя один из выходов мультиплексора *F5*. Это позволяет реализовать либо любую 6-входовую функцию, либо мультиплексор 8:1, либо некоторую функцию до 19-ти переменных.

Каждый *CLB* имеет четыре сквозных линии – по одной на каждую логическую ячейку. Эти линии используются как дополнительные входы данных, либо как дополнительные трассировочные ресурсы, не расходующие логические ресурсы.

#### 10.4.4. Арифметическая логика

Каждая *LC* содержит специальную логику ускоренного переноса, которая обеспечивает наилучшую реализацию на *FPGA* различных арифметических функций. *CLB* содержит две отдельные цепи переноса – по одной на каждую секцию. Размерность цепи переноса – два бита на *CLB*.

Арифметическая логика включает в себя элемент Исключающее ИЛИ (*XOR*), который позволяет реализовать однобитный сумматор в одной логической ячейке. В каждой логической ячейке имеется элемент, реализующий функцию И, который предназначен для построения быстродействующих умножителей.

Специальные трассы логики ускоренного переноса могут также использоваться для каскадного включения функциональных генераторов при необходимости создания функций с большим количеством входных переменных.

#### 10.4.5. Буферы с тремя состояниями

Каждый *CLB* SPARTAN-II содержит два буфера с тремя состояниями, которые нагружены на внутренние шины. Каждый буфер *BUFT* имеет независимый вход управления третьим состоянием и независимый входной контакт.

#### 10.4.6. Блочная память (Block RAM)

В *FPGA* SPARTAN-II встроена особая блочная память *Block SelectRAM* большой емкости. Она создана в дополнение к распределенной памяти небольшой емкости *SelectRAM*, реализованной на таблицах преобразования *LUTRAM* (*Look Up Table RAM*). Блоки памяти *Block Select RAM* организованы в виде столбцов. Все кристаллы SPARTAN-II содержат два таких столбца, по одному вдоль каждой вертикальной кромки. Каждый блок памяти равен по высоте четырем *CLB*, таким образом, микросхема SPARTAN-II, имеющая 8 *CLB* по высоте, содержит 2 блока памяти на колонку и 4 блока памяти в целом.

В таблице 10.2 приводятся емкости блочной памяти для некоторых кристаллов SPARTAN-II.

Таблица 10.2.

Емкость блочной памяти

Кристалл <i>Spartan-II</i>	Число блоков	Общий объём блоч- ной памяти, Бит
XC2S15	4	16384
XC2S30	6	24576
XC2S200	14	57344

Каждый блок памяти, как показано на рис. 10.8, это полностью синхронная двухпортовая *RAM* с независимым управлением для каждого порта. Размерность шины данных для обеих портов может быть сконфигурирована независимо, что позволяет создавать преобразователи размерности шины.

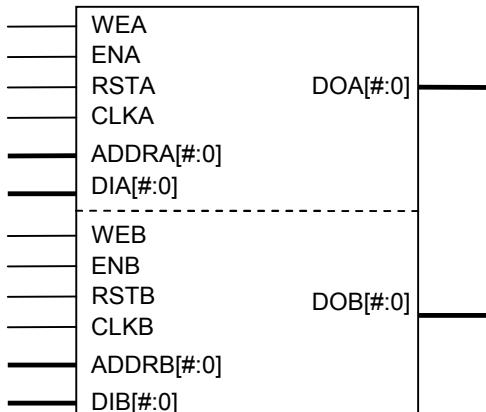


Рис. 10.8. Блок памяти

В таблице 10.3 показаны возможные соотношения размерностей шин данных и адреса.

Т а б л и ц а 10.3.

**Соотношение шин адреса и данных**

Разрядность	Колличество ячеек	Шина адреса	Шина данных
1	4096	ADDR<11:0>	DATA<0>
2	2048	ADDR<10:0>	DATA<1:0>
4	1024	ADDR<9:0>	DATA<3:0>
8	512	ADDR<8:0>	DATA<7:0>
16	216	ADDR<7:0>	DATA<15:0>

В кристаллах *Spartan-II* созданы специальные трассировочные ресурсы для связи блочной памяти с блоками *CLB* и другими блоками блочной памяти.

## 10.5. Программируемая трассировочная матрица

Матрица программируемых переключателей, представленная на рис. 10.9, является существенным компонентом структуры соединений, обеспечивающей связь между блоками. Установление соединений или их размыкание в элементарном программируемом переключателе позволяет продлевать и разрывать проводящие сегменты в шинах. Еще более важно, что матрица программируемых переключателей позволяет *поворнуть сигналы* на 90° путем соединения горизонтального и вертикального проводников. Без этого нельзя было бы соединить данный логический блок с другими блоками, находящимися в другой строке или в другом столбце матрицы, образуемой логическими блоками.

Матрица программируемых переключателей является необходимым компонентом, но за его использование приходится платить: при каждом прохождении сигналов через такую матрицу вносится небольшая задержка. Хорошая программа компоновки для ИС типа *FPGA* ищет не только какие-либо возможные размещения логических блоков и некоторую комбинацию соединений, которые будут работать.

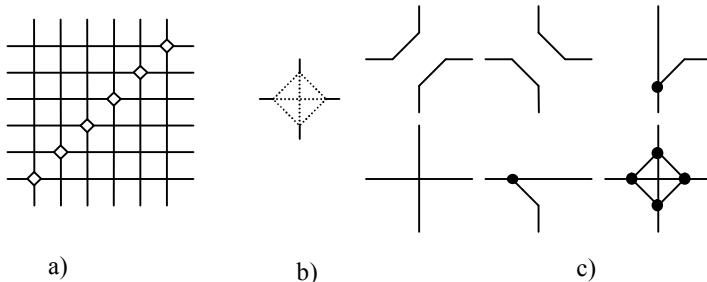


Рис. 10.9. Программируемые соединения в ИС серии SPARTAN II: а) матрица программируемых переключателей (PSM); б) элементарный программируемый переключатель (PSE); в) несколько возможных вариантов соединений

Быстродействие проекта, рассчитанного для наихудшего случая, ограничивает величина задержки, вносимой наиболее длинной трассой. Поэтому архитектура трассировочных ресурсов

и программы размещения и трассировки создавались с учетом использования их в едином процессе оптимизации. Этот совместный процесс оптимизации минимизирует наиболее длинные пути и, таким образом, создает проект с наилучшей системной производительностью.

Кроме того, совместная оптимизация сокращает время компиляции, так как программное обеспечение и архитектура микросхемы создавались с учетом наилучшего взаимодействия. Циклы проектирования, таким образом, сократились благодаря более коротким временам каждой из итераций всего процесса.

### 10.5.1. Локальные связи

Как показано на рис. 10.9, в кристалле *Spartan-II* созданы локальные трассировочные ресурсы, называемые *VersaBlock*. Они позволяют реализовать три типа соединений:

- Связи между таблицами преобразования *LUT*, триггерами и главной трассировочной матрицей *GRM*.
- Внутренние обратные связи *CLB*, которые создают высокоскоростные связи с таблицами преобразования *LUT* в рамках одного *CLB*, и позволяют соединять их в виде цепочек с минимальными задержками распространения сигналов.
- Прямые трассы, которые создают высокоскоростные соединения с соседними по горизонтали *CLB*, избегая при этом больших задержек, присущих трассам главной трассировочной матрицы.

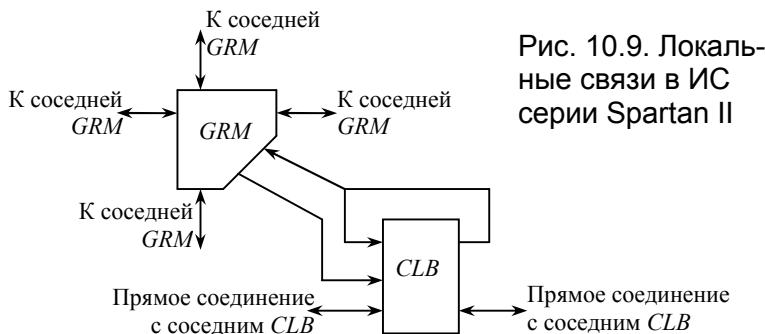


Рис. 10.9. Локальные связи в ИС серии *Spartan II*

### **10.5.2. Трассировочные ресурсы общего назначения**

Большинство связей в кристаллах SPARTAN-II реализуются с помощью трассировочных ресурсов общего назначения и, следовательно, большая часть ресурсов межсоединений связана с этим типом трассировочной иерархии. Трассировочные ресурсы общего назначения расположены в виде горизонтальных и вертикальных трассировочных линий и размещены в непосредственной близости от строк и столбцов матрицы, образованной блоками *CLB*. Ниже перечислены эти ресурсы:

- Примыкающая к каждому *CLB* главная трассировочная матрица *GRM*. *GRM* – это матрица переключателей, с помощью которых коммутируются горизонтальные и вертикальные трассы и посредством которых блоки *CLB* получают доступ к трассировочным ресурсам общего назначения
- *GRM* связана в каждом из четырех направлений с соседней *GRM* посредством 24-х трасс одинарной длины
- 96 буферизованных *HEX*-линий позволяют подать *GRM* сигналы на шесть других *GRM* в каждом из четырех направлений. *HEX*-линии организованы в виде зигзагообразных линий и могут подключаться к источникам сигнала только в своих конечных точках или серединных (три блока от источника). Одна третья часть *HEX*-линий является двунаправленными, в то время как остальные – односторонние.
- 12 длинных линий (*Long lines*) являются буферизованными, двунаправленными линиями, быстро передающие сигналы в микросхеме на достаточно большие расстояния. Вертикальные длинные линии имеют протяженность равную полной высоте кристалла, а горизонтальные длинные линии – полной ширине.

### **10.5.3. Трассировочные ресурсы для блоков ввода-вывода**

Кристалл SPARTAN-II имеет дополнительные трассировочные ресурсы, расположенные по периферии всей микросхемы.

Эти связи, называемые *VersaRing*, формируют добавочный интерфейс между блоками *CLB* и блоками *IOB*. Они улучшают возможности закрепления сигналов за контактами и переназначения уже сделанного закрепления, если это требование задается расположением проводников на печатной плате. При этом сокращается время изготовления всего проекта, т. к. изготовление и проектирование печатной платы можно выполнять одновременно с проектированием *FPGA*.

#### 10.5.4. Специальные трассировочные ресурсы

Некоторые классы сигналов требуют наличия специальных трассировочных ресурсов для максимизации быстродействия. В устройстве SPARTAN-II специальные трассировочные ресурсы создавались для двух классов сигналов:

- Горизонтальные трассировочные ресурсы создавались для реализации в микросхеме шин с тремя состояниями. Четыре разделенные линии шин реализованы для каждой строки *CLB*, позволяя организовывать сразу несколько шин в пределах одной строки (рис. 10.10)
- Две специальные линии для распространения сигналов быстрого переноса к прилегающему *CLB* в вертикальном направлении.

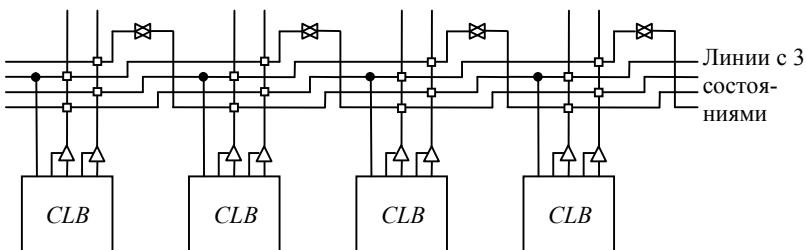


Рис. 10.10. Подключение буферов с 3 состояниями к горизонтальным линиям

### **10.5.5. Глобальные трассировочные ресурсы**

Глобальные трассировочные ресурсы распределяют тактовые сигналы и другие сигналы с большим коэффициентом разветвления по выходу на всем пространстве кристалла. Кристалл *Spartan-II* имеет два типа глобальных трассировочных ресурсов, называемых соответственно первичными и вторичными.

- Первичные глобальные трассировочные ресурсы представляют собой четыре специальные глобальные сети со специально выделенными входными контактами и связанными с ними глобальными буферами, спроектированными для распределения сигналов синхронизации с высоким коэффициентом разветвления и с минимальными расхождениями фронтов. Каждая такая сеть может быть нагружена на входы синхронизации всех *CLB*, *IOB* и *BlockRAM* - блоков микросхемы. Источниками сигналов для этих сетей могут быть только глобальные буферы. Всего имеется четыре глобальных буфера – по одному для каждой глобальной сети.
- Вторичные глобальные трассировочные ресурсы состоят из 24 магистральных линий, 12 вдоль верхней кромки кристалла и 12 вдоль нижней. По этим связям можно передать до 12 уникальных сигналов на колонку по 12 длинным линиям данной колонки. Вторичные ресурсы являются более гибкими, чем первичные, поскольку эти сигналы, в отличие от первичных, могут трассироваться не только до входов синхронизации.

### **10.6. Распределение сигналов синхронизации**

ИС *SPARTAN-II* имеют высокоскоростные, с малыми искажениями трассировочные ресурсы для распределения сигналов синхронизации на всем пространстве микросхемы. Типичное распределение цепей синхронизации показано на рис. 10.12.

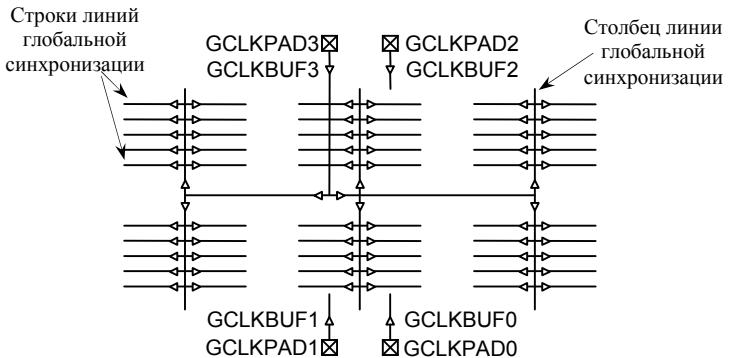


Рис. 10.12. Глобальные цепи синхронизации

В микросхеме имеется четыре глобальных буфера, два в середине верхней части микросхемы и два в середине нижней части. Эти буфера через первичные глобальные сети могут подводить сигналы синхронизации на любой тактовый вход.

Для каждого глобального буфера имеется соответствующий, примыкающий к нему контакт микросхемы. Сигнал на вход глобального буфера может подаваться как с этих контактов, так и от сигналов, разводимых ресурсами общего назначения.

## 10.7. Модули автоподстройки задержки (DLL)

Полностью цифровая автоподстройка задержки *DLL*, связанная с каждым глобальным буфером, может устранять различие задержек между синхросигналом на входном контакте микросхемы и сигналами на тактовых входах внутренних схем устройства. Каждая *DLL* может быть нагружена на две глобальные цепи синхронизации. Схема *DLL* отслеживает сигнал синхронизации на входном контакте микросхемы и тактовый сигнал, распределляемый внутри кристалла, затем автоматически устанавливает необходимую задержку. Дополнительная задержка вводится таким образом, что фронты сигналов синхронизации достигают внутренних триггеров в точности на один период синхронизации

позже их прихода на входной контакт. Эта система с обратной связью эффективно устраняет задержку распределения сигналов синхронизации, гарантуя, что фронты синхросигналов на входе микросхемы и на внутренних тактовых входах совпадают с большой точностью.

Вдобавок, для устранения задержек, возникающих при распределении тактовых сигналов, *DLL* создает новые возможности управления функциями синхронизации. Модуль *DLL* может создавать четыре квадратурные фазы из исходного источника синхросигнала; удваивать частоту синхросигнала или делить эту частоту на 1.5, 2, 2.5, 3, 4, 5, 8 или 16.

Модуль *DLL* также функционирует как тактовое зеркало. Путем вывода из микросхемы сигнала с выхода *DLL* и последующего ввода этого сигнала снова внутрь кристалла, схема может устраниć сдвиг фаз для тактовых сигналов на уровне печатной платы при работе с несколькими микросхемами SPARTAN-II.

Чтобы гарантировать, что системная синхронизация будет нормально функционировать до момента окончания конфигурирования системы и начала штатной работы, схема *DLL* имеет возможность задерживать процесс конфигурирования до нормальной синхронизации с системой.

## 10.8. Система проектирования

Разработка устройств на основе кристаллов SPARTAN-II осуществляется программным обеспечением проектирования *Xilinx ISE* и/или *WebPACK ISE*. Процесс проектирования включает: ввод проекта, размещение в кристалл и верификацию. Для ввода проекта могут применяться стандартные электронные САПР, таких фирм как *Cadence*, *Simplicity*, *Mentor Graphics* или *Synopsys*. Для размещения в кристалл и верификации используются специализированные под архитектуру САПР, выпускаемые только *Xilinx*.

Система проектирования фирмы *Xilinx* интегрирована в управляющую программу называемую *Xilinx Design Manager (XDM)*, которая обеспечивает доступ к общему пользовательско-

му интерфейсу, независимо от выбора вида программы ввода или верификации. Программа *XDM* упрощает выбор настроек, необходимых для выполнения проекта, благодаря наличию разветвленного меню и легко доступной справочной системе (*on-line help*).

Прикладные программы, начиная от создания схемы (*schematic capture*), до размещения и трассировки (*Placement and Routing – PAR*), доступны из программы *XDM*. Несколько расширенных свойств программного обеспечения облегчает проектирование микросхем SPARTAN-II. Например, схемные относительно расположенные макросы (*Relationally Placed Macros – RPMs*), в которых содержится информация о принудительной взаимной ориентации составных частей элементов проекта, дают необходимую информацию для их реального размещения на кристалле. Они помогают обеспечить оптимальное выполнение стандартных логических функций.

Для ввода проектов с помощью языков описания аппаратных средств (*Hardware Description Language – HDL*), система проектирования *Xilinx ISE* предоставляет интерфейсы к синтезаторам следующих фирм:

*Synopsis (FPGA Compiler, FPGA Express);*

*Exemplar (Spectrum);*

*Symplicity (Symplify);*

Для схемного ввода проектов, системы проектирования *Xilinx ISE* и *WebPACK ISE* предоставляют интерфейсы к следующим системам создания схем:

*Mentor Graphics V8 (Design Architect Quick Sim II);*

*Viewlogic System (Viewdraw).*

Существуют и другие производители, предлагающие аналогичные по функциям системы ввода проекта.

Для упрощения взаимодействия различных САПР существует стандартный формат файлов (*EDIF*), который поддерживаются всеми производителями САПР.

САПРы для SPARTAN-II включают унифицированную библиотеку (*Unified library*) стандартных функций. Эта библиотека содержит свыше 400 примитивов и макросов, от двухвходо-

вых вентиляй И, до 16 битовых аккумуляторов и включает арифметические функции, компараторы, счетчики, регистры данных, дешифраторы, шифраторы, функции ввода-вывода, защелки, булевы функции, мультиплексоры и сдвигающие регистры.

Среда проектирования поддерживает ввод иерархических проектов, в которых схемы верхнего уровня содержат основные функциональные блоки, в то время, как системы нижнего уровня определяют логические функции этих блоков. Данные элементы иерархического проекта автоматически объединяются соответствующими средствами на этапе размещения в кристалл. При иерархической реализации могут объединяться различные средства ввода проекта, давая возможность каждой из частей вводить наиболее подходящим для нее методом.

### 10.8.1. Размещение проекта в кристалл

Программное средство размещения и трассировки (*place-and-route, PAR*) обеспечивает автоматическое выполнение процесса размещение проекта в кристалл. Процедура разбиения на физические блоки получает исходную информацию о проекте в виде перечня связей формата *EDIF* и осуществляет привязку абстрактных логических элементов к реальным физическим ресурсам архитектуры *FPGA* (*IOB, CLB*). Затем процедура размещения определяет для них наилучшее место на кристалле, руководствуясь информацией о межсоединениях и желаемом быстродействии. В завершении, процедура трассировки выполняет соединения между блоками.

Алгоритмы программы *PAR* поддерживают автоматическое выполнение большинства проектов. Тем не менее, в некоторых приложениях пользователь, при необходимости, может осуществлять контроль и управление процессом. На этапе ввода проекта пользователь может задавать свою информацию для разбиения, размещения и трассировки.

В программное обеспечение встроено средство *Timing Wizard*, управляющее процессом размещения и трассировки с учетом требований к временем распространения сигналов. При вводе проекта пользователь задает эту информацию в виде вре-

менных ограничений для определенных цепей. Процедуры анализа временных параметров связей анализируют эти, заданные пользователем, требования и пытаются удовлетворить им.

Временные требования вводятся в схему в виде системных ограничений, таких как минимально допустимая частота синхронизации, или максимально допустимая задержка между двумя регистрами. При таком подходе результирующее быстродействие системы с учетом суммарной протяженности путей автоматически подгоняется под требования пользователя. Таким образом, задание временных ограничений для отдельных цепей становится не нужным.

### 10.8.2. Верификация проекта

В дополнение к обычному программному моделированию *FPGA*, пользователь может использовать метод непосредственной отладки реальных цепей. Благодаря неограниченному количеству циклов перепрограммирования кристаллов *FPGA*, работоспособность проектов можно проверить в реальном масштабе времени.

Система проектирования устройств SPARTAN-II поддерживает и программное моделирование и метод отладки непосредственно аппаратных цепей. Для выполнения моделирования система извлекает временную информацию, полученную после размещения из базы данных проекта, и вводит ее в сетевой перечень. Пользователь может и сам проверить критичные по времени части проекта, используя статический временной анализатор *TRACE*.

Для непосредственной отладки цепей существует кабель для загрузки конфигурационных данных и считывания данных из микросхемы. Этот кабель соединяет персональный компьютер с микросхемой *FPGA*, установленной в законченное устройство. После загрузки проекта в *FPGA*, пользователь может выполнить один шаг изменения логического состояния схемы, затем выполнить обратное считывание состояния триггеров в компьютер и проанализировать правильность работы схемы. Простейшие модификации проекта, при этом, можно осуществлять в считанные минуты.

### 10.8.3. Конфигурирование кристалла в составе устройства.

Конфигурирование – это процесс загрузки битовой последовательности, полученной с помощью программного обеспечения проектирования, во внутреннюю конфигурационную память кристаллов *FPGA*. SPARTAN-II может загружаться как побитно (ведущий/подчиненный последовательные режимы и *JTAG*), так и побайтно (подчиненный параллельный режим).

Конфигурационные данные при выключенном питании должны храниться во внешнем энергонезависимом устройстве статической памяти. Обычно для этого применяются микросхемы флэш-памяти *Xilinx* серии XC1700 или XC1800. В табл. 10.4 представлены объемы конфигурационных последовательностей, загружаемых в ИС семейства SPARTAN-II.

Т а б л и ц а 10.4

**Размер конфигурационной последовательности в *Spartan-II***

Кристалл	Объём конфигурационной последовательности, Бит
XC2S15	197728
XC2S30	336 800
XC2S50	559 232
XC2S100	781 248
XC2S150	1040 128
XC2S200	1335 840

### 10.8.4. Режимы конфигурирования

Микросхемы семейства SPARTAN-II поддерживает следующие четыре режима конфигурирования:

- подчиненный последовательный режим (*Slave-serial*),
- ведущий последовательный режим (*Master-serial*),
- подчиненный параллельный режим (*Slave Parallel*),
- режим периферийного сканирования (*JTAG – Boundary Scan*).

Комбинация сигналов, поданных на специальные входные контакты микросхемы (M2, M1, M0), позволяет выбрать один из

режимов конфигурирования, при этом четыре из восьми кодов соответствуют подтянутому до напряжения питания (*pull-up*) состоянию входов блоков ввода-вывода до начала процедуры конфигурирования, и еще четыре комбинации соответствуют состоянию неопределенного потенциала входов блоков ввода-вывода. Соответствие этих конфигурационных кодов необходимому режиму приведено в табл. 10.5.

Контакты M2, M1, M0 внутри кристалла подключены к *pull-up* резисторам, поэтому если на плате эти контакты никуда не подключены, то на них присутствует логическая 1.

Т а б л и ц а 10.5  
Конфигурационные коды

Режим	M2	M1	M0	CCLK	Разрядность данных	Последовательный выход DOUT	Контакты подтянуты
Master-serial	0	0	0	Выход	1	Есть	Нет
Boundary- scan	1	0	1		1	Нет	Нет
Slave Parallel	1	1	0	Вход	8	Нет	Нет
Slave-serial	1	1	1	Вход	1	Есть	Нет
Master-serial	1	0	0	Выход	1	Есть	Да
Boundary-scan	0	0	1		1	Нет	Да
Slave Parallel	0	1	0	Вход	8	Нет	Да
Slave-serial	0	1	1	Вход	1	Есть	Да

### 10.8.5. Сигналы конфигурации

Микросхемы SPARTAN-II конфигурируются путем загрузки конфигурационных данных во внутреннюю конфигурационную память. Часть специальных контактов, которые при этом используются, не могут применяться для других целей, в то же время некоторые из них могут после завершения конфигурирования служить в качестве контактов ввода-вывода общего назначения.

К специальным контактам конфигурирования относятся следующие:

- контакты режима конфигурирования (M2, M1, M0);

- контакт синхронизации процесса конфигурирования (CCLK);
- контакт PROGRAM;
- контакт DONE;
- контакты периферийного сканирования (TDI, TDO, TMS, TCK).

В зависимости от выбранного режима конфигурирования, контакт CCLK может быть либо источником сигнала синхронизации, либо наоборот – приемником сигнала от внешнего генератора синхросигналов.

#### **10.8.6. Последовательность конфигурации**

Процесс конфигурирования микросхем семейства SPARTAN-II состоит из трех фаз. В первой фазе конфигурирования очищается конфигурационная память. Следующая фаза – загрузка данных в конфигурационную память. Наконец, активизируется логика (фаза *Start-Up*). Обычно процесс конфигурирования запускается автоматически после подачи напряжения питания.

Первая и последняя фазы одинаковы для всех режимов конфигурирования, а фаза загрузки данных в конфигурационную память зависит от выбранного режима.

## ГЛАВА 11. МИКРОКОНТРОЛЛЕР

Микроконтроллер – это интегральная схема, содержащая процессор, память, вспомогательные устройства и устройства ввода-вывода, способная принимать сигналы от датчиков, обрабатывать их и выдавать управляющие сигналы на исполнительные устройства для выполнения той или иной задачи (рис. 11.1).

Микроконтроллеры *AVR* по праву считаются одним из самых интересных направлений, активно развивающихся корпорацией Atmel (<http://www.atmel.com/>). Они представляют собой мощный инструмент для создания современных высокопроизводительных и экономичных многоцелевых контроллеров. На настоящий момент соотношение цена - производительность - энергопотребление для микроконтроллеров *AVR* по-прежнему остается едва ли не лучшим на мировом рынке 8-разрядных микроконтроллеров. С устройством микроконтроллеров мы познакомимся на примере восьмиразрядного микроконтроллера ATmega8535 фирмы *Atmel*. Этот микроконтроллер является одним из лучших среди современных восьмиразрядных микроконтроллеров.

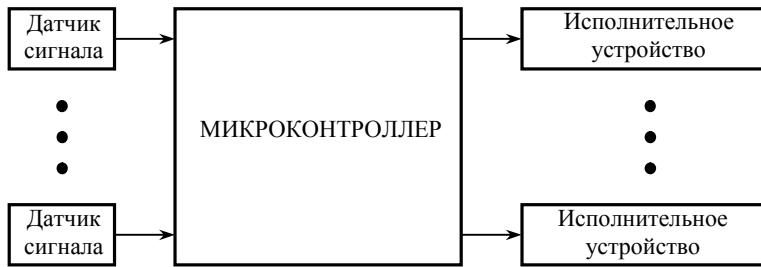


Рис. 11.1. Микроконтроллер в системе управления

Области применения микроконтроллеров многогранны – от простейших игрушек и интеллектуальных датчиков до сложных

промышленных систем управления и современного телекоммуникационного оборудования.

## 11.1. Внутренняя структура современных микроконтроллеров

Большинство современных микроконтроллеров имеет внутреннюю структуру, приведенную на рис. 11.2.

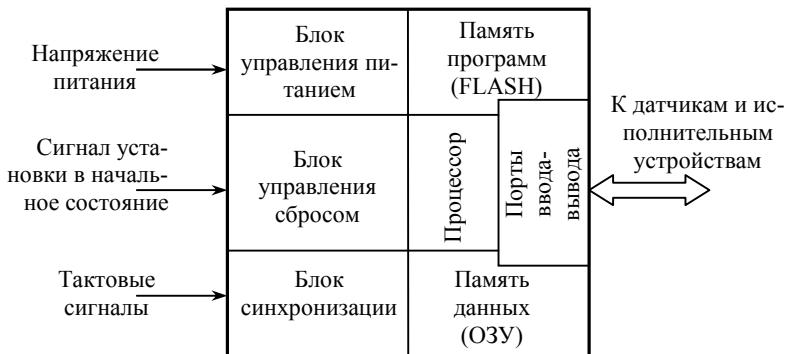


Рис. 11.2. Внутренняя структура микроконтроллера

Блоки, входящие в состав микроконтроллера, выполняют следующие функции:

**Процессор** – обеспечивает обработку информации, выполняя команды в соответствии с системой команд микроконтроллера. Различаются *CISC* (*Complex Instruction Set Computer*) и *RISC* (*Reduced Instruction Set Computer*) процессоры. В настоящее время в составе микроконтроллеров чаще применяются *RISC* процессоры, имеющие большее быстродействие.

**Память программ** – хранит программу, в соответствии с которой работает микроконтроллер. Программа записывается в память программ при программировании микроконтроллера и сохраняется в ней при выключенном питании. В современных микроконтроллерах применяется память программ, которую можно программировать в составе системы, где она используется

(*FLASH*). Такой способ программирования называется *ISP (In System Programming)*. При этом имеется возможность обновлять программу по мере необходимости в процессе эксплуатации устройства.

**ОЗУ** – оперативное запоминающее устройство – *RAM (Random Access Memory)*. ОЗУ используется для хранения промежуточных результатов. Информация в ОЗУ стирается при выключении питания микроконтроллера.

**Порты ввода-вывода** – через которые микроконтроллер обменивается информацией с внешним миром (считывает информацию с датчиков и выдает управляющие сигналы на исполнительные устройства).

**Блок управления питанием** – обеспечивающий правильный запуск микроконтроллера после включения питания.

**Блок управления сбросом** – вместе с входом *RESET* устанавливающий микроконтроллер в некоторое исходное состояние.

**Блок синхронизации** – вырабатывающий тактовые сигналы, необходимые для правильного взаимодействия всех внутренних блоков микроконтроллера.

## 11.2. Характеристики микроконтроллеров

**Предельная тактовая частота.** Тактовая частота определяет скорость работы микроконтроллера. Современные микроконтроллеры имеют предельную тактовую частоту от нескольких мегагерц до сотен мегагерц. При разработке различных устройств следует выбирать микроконтроллеры с наименьшей тактовой частотой, достаточной для решения поставленной задачи, поскольку такие микроконтроллеры дешевле и потребляют меньшую мощность, что особенно важно при питании от батарей.

**Число выводов.** Микроконтроллеры располагаются в стандартных корпусах, имеющих от 8 до 64 и более выводов. Микроконтроллеры с малым числом выводов дешевле, а микроконтроллеры с большим числом выводов позволяет создавать сложные системы управления.

**Объем памяти программ.** Микроконтроллеры выпускаются с памятью программ от 1 Кбайт до 128 Кбайт и более. Чем

больше память программ, тем больше возможностей у данного микроконтроллера, его цена при этом возрастает.

**Объем ОЗУ.** Оперативное запоминающее устройство (память чисел) служит для хранения промежуточных результатов и обрабатываемых данных. Объем ОЗУ в микроконтроллере может быть небольшим (от 64 байт). Этого во многих случаях бывает достаточно, однако в случае необходимости возможно подключение большой по объему внешней памяти чисел.

**Количество внутренних регистров общего назначения.**

В этих регистрах хранится информация, используемая при выполнении команд микроконтроллера. Большое количество регистров общего назначения позволяет уменьшить число обращений к памяти чисел, что повышает скорость выполнения программы.

**Разрядность внутренней шины микроконтроллера.**

Внутренняя шина микроконтроллера может иметь 8 разрядов, 16 разрядов или 32 разряда. Увеличение числа разрядов внутренней шины микроконтроллера повышает его производительность. В настоящее время наибольшее распространение получили 8-разрядные и 16-разрядные микроконтроллеры.

**Дополнительные функциональные блоки.** Любой микроконтроллер обязательно имеет порты ввода/вывода, один или несколько таймеров, систему обработки прерываний. В микроконтроллерах могут быть цифроаналоговые и аналого-цифровые преобразователи, устройства последовательной передачи данных для обмена информацией между микроконтроллерами, блокнотная память (*EEPROM*), часы реального времени, аналоговый компаратор и т.д. Эти дополнительные блоки обслуживаются регистрами ввода/вывода. К регистрам ввода/вывода можно обращаться с помощью команд *IN* и *OUT*, которые имеются в системе команд практически любого микроконтроллера. Каждый регистр ввода/вывода имеет свой адрес. Суммарность этих адресов называется пространством адресов ввода/вывода. Каждому функциональному блоку принадлежит один или несколько регистров ввода/вывода.

### 11.3. Микроконтроллер ATmega8535

Микроконтроллер ATmega8535 имеет следующие характеристики:

Улучшенная *RISC* архитектура.

Тактовая частота от 0 до 16 МГц.

Напряжение питания от 2,7 В до 6 В.

Система команд содержит 130 команд, большинство из которых выполняется за один такт. Только умножение 8-разрядных двоичных чисел осуществляется встроенным аппаратным перемножителем за два такта.

Память программ имеет объем 8К байт с возможностью программирования по параллельному или последовательному интерфейсу. Допускается до 10 000 циклов перезаписи.

ОЗУ (*RAM*) имеет объем 512 байт.

Микроконтроллер содержит 32 8-разрядных регистра общего назначения.

В микроконтроллере имеется 32 программируемые линии ввода/вывода (четыре восьмиразрядных порта ввода/вывода).

Микроконтроллер содержит следующие дополнительные блоки:

а) программируемый асинхронный последовательный порт (*USART* – *Universal Synchronous/Asynchronous Receiver and Transmitter*), скорость передачи информации можно менять дискретно от 2.4 до 250 Кбит/с;

б) программируемый синхронный последовательный порт (*SPI* – *Serial Peripheral Interface*), скорость передачи до 8 Мбит/с;

в) два 8-разрядных таймера/счетчика;

г) один 16-разрядный таймер/счетчик;

д) программируемый сторожевой таймер с внутренним генератором;

е) аналоговый компаратор;

ж) 10-разрядный аналого-цифровой преобразователь (8 каналов с мультиплексированием);

з) система обработки прерываний, позволяющая обрабатывать 17 внутренних и 4 внешних прерывания;

и) *EEPROM* (*Electrically Erasable Programmable Read Only Memory*) объемом 512 байт.

к) двухпроводный последовательный интерфейс (*TWI – Two-wire Serial Interface*) с задаваемой программно скоростью передачи до 400 Кбит/сек.

Микроконтроллеры ATmega8535 размещаются в прямоугольном двухрядном (по 20 выводов с каждой стороны) пластмассовом корпусе *DIP*, или в квадратном пластмассовом корпусе *TQFP* или *PLCC* (по 11 выводов на каждой стороне квадрата).

### 11.3.1. Назначение выводов микроконтроллера ATmega8535

Цоколевка микроконтроллера ATmega8535, размещенного в прямоугольном двухрядном корпусе *DIP40* и в корпусе *TQFP44*, приведена на рис. 11.3. Указанные далее номера выводов соответствуют корпусу *DIP40*.

*V<sub>CC</sub>* – вывод для подключения напряжения питания от 2.7 В до 6 В.

*GND* – вывод для подключения земли (цифровая земля).

Порт *A* (*PA7…PA0*) – восьмиразрядный двунаправленный порт ввода/вывода. Выводы порта *A* используются в качестве входов для подачи сигналов на аналого-цифровой преобразователь.

Порт *B* (*PB7…PB0*) – восьмиразрядный двунаправленный порт ввода/вывода. Выводы порта *B* могут обеспечивать выполнение некоторых специальных функций. Например, выводы 6, 7, 8 (*MOSI*, *MISO*, *SCK*) могут использоваться для последовательного программирования микроконтроллера, а вывод 3 для подачи сигнала внешнего прерывания *INT2*.

Порт *C* (*PC7…PC0*) – восьмиразрядный двунаправленный порт ввода/вывода. Выводы *PC0* и *PC1* могут служить для передачи сигналов двухпроводного интерфейса *TWI*. К выводам *PC6* и *PC7* можно подключать низкочастотный кварцевый резонатор ( $f_{\text{KB}} = 2^{15}$  Гц).

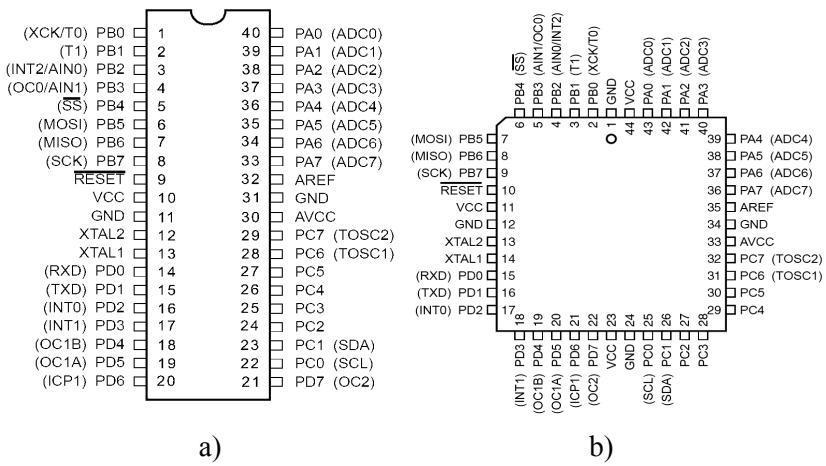


Рис.11.3. Цоколевка микроконтроллера ATmega8535: а) корпус DIP40, б) корпус TQFP44 или PLCC44

Порт *D* (*PD7...PD0*) – восьмиразрядный двунаправленный порт ввода/вывода. Выводы порта *D* позволяют выполнить некоторые специальные функции. Например, выводы 16, 17 можно использовать для подачи сигналов внешних прерываний *INT0* и *INT1*.

Каждый разряд любого из портов можно настроить индивидуально в качестве либо входа, либо выхода. Если вывод является выходом, то он может обеспечивать в нагрузку ток до 20 мА, что позволяет непосредственно подключать светодиод (без дополнительных усилителей) через ограничивающий ток резистор.

*RESET* – вход начальной установки. Низкий уровень на этом входе устанавливает микроконтроллер в некоторое исходное состояние. После окончания сигнала *RESET* микроконтроллер начинает свою работу из этого состояния. Что такое исходное состояние, будет пояснено далее при рассмотрении внутренней структуры микроконтроллера. Длительность сигнала *RESET* должна быть больше двух периодов тактового сигнала. При

включении питания в микроконтроллере автоматически вырабатывается внутренний сигнал *RESET*.

*XTAL1* и *XTAL2* – входы для подключения кварцевого резонатора. Если к этим выводам подключить кварцевый резонатор, то внутренний генератор формирует сигнал тактовой последовательности, частота которого совпадает с резонансной частотой кварцевого резонатора. Возможно использование внешнего тактового генератора, сигнал от которого следует подать на вход *XTAL1*. Выход *XTAL2* при этом никуда не подключается.

*AGND* – аналоговая земля. Для уменьшения помех этот вывод должен быть подключен к *GND* только в одной точке.

*AV<sub>CC</sub>* – вывод для подключения источника питания аналоговой части микроконтроллера (аналого-цифровой преобразователь и порт *A*). Если АЦП не используется, то этот вывод подключается к *V<sub>CC</sub>* напрямую. Если используется АЦП, то вывод *AV<sub>CC</sub>* соединяется с *V<sub>CC</sub>* через индуктивность 10 МкГн и от этого вывода к *AGND* должна быть подключена емкость 0.1 мкФ.

*AREF* – вход для подключения источника эталонного напряжения. Этот источник должен иметь напряжение в диапазоне от 2В до *AV<sub>CC</sub>*.

### 11.3.2. Структура микроконтроллера ATmega8535

Внутренняя структура микроконтроллера ATmega8535 представлена на рис. 11.4. Улучшенная *RISC* (*Reduced Instruction Set Computer*) архитектура микроконтроллеров *AVR* объединяет в себе комплекс решений, направленных на повышение быстродействия микропроцессорного ядра.

Арифметико-логическое устройство (*ALU*), в котором выполняются все вычислительные операции, имеет доступ к 32-м оперативным регистрам, объединенным в регистровый файл. Выборка содержимого регистров, выполнение операции и запись результата в регистровый файл выполняются за один машинный цикл. Для сравнения стоит отметить, что большинство микроконтроллеров имеет только один такой регистр, непосредственно доступный *ALU*, – аккумулятор, что требует включения в программу дополнительных команд его загрузки и считывания.

Основной идеей всех *RISC*, как известно, является увеличение быстродействия за счет сокращения количества операций обмена с памятью программ. Для этого каждую команду стремятся уместить в одну ячейку памяти программ. При ограниченной разрядности ячейки памяти это неизбежно приводит к сокращению набора команд микропроцессора.

У *AVR*-микроконтроллеров в соответствии с этим принципом практически каждая команда (исключая те, у которых одним из операндов является 16-разрядный адрес) хранится в одной ячейке памяти программ. Но сделать это удалось не за счет сокращения количества команд процессора, а путем расширения ячейки памяти программ до 16 разрядов. Такое решение позволяет *AVR*-микроконтроллерам иметь более развитую систему команд по сравнению с другими *RISC*-микроконтроллерами.

Организация памяти *AVR* выполнена по схеме гарвардского типа, в которой разделены не только адресные пространства памяти программ и памяти данных, но также и шины доступа к ним.

Вся программная память *AVR*-микроконтроллеров выполнена по технологии *FLASH* и размещена на кристалле. Она представляет собой последовательность 16-разрядных ячеек и имеет емкость 4К слов.

Во *FLASH*-память, кроме программы, могут быть записаны данные, которые не изменяются во время работы микропроцессорной системы. Это различные константы, таблицы знакогенераторов, таблицы линеаризации датчиков и т. п.

Достоинством технологии *FLASH* является высокая плотность размещения ячеек памяти на кристалле, а недостатком – то, что она не позволяет стирать отдельные ячейки. Поэтому всегда выполняется полная очистка всей памяти программ. При этом для *AVR* гарантируется как минимум 10 000 циклов перезаписи *FLASH*-памяти.

Для хранения данных микроконтроллер имеет внутреннюю оперативную *SRAM* память объемом 512 байт. Эта память используется также для хранения содержимого программного счетчика и регистров общего назначения при обработке прерываний.

Объем оперативной памяти можно увеличить, подключив внешнюю память объемом до 64 Кбайт.

Разделение шин доступа (рис. 11.4) к *FLASH* памяти и *SRAM* памяти дает возможность иметь шины данных для памяти данных и памяти программ различной разрядности, а также использовать технологию конвейеризации. Конвейеризация заключается в том, что во время исполнения текущей команды программный код следующей команды уже выбирается из памяти и дешифрируется.

Для сравнения стоит отметить, что у микроконтроллеров семейства *MCS-51* выборка кода команды и ее исполнение осуществляются последовательно, что занимает один машинный цикл, который длится 12 периодов тактового генератора.

В случае использования конвейера приведенную длительность машинного цикла можно сократить. Например, у *PIC*-микроконтроллеров фирмы *Microchip* за счет использования конвейера удалось уменьшить длительность машинного цикла до 4 периодов сигнала *CLK*. Длительность же машинного цикла *AVR* – микроконтроллеров составляет один период сигнала *CLK*. Таким образом, *AVR*-микроконтроллеры способны обеспечивать заданную производительность при более низкой тактовой частоте. Именно эта особенность архитектуры и позволяет *AVR*-микроконтроллерам иметь наилучшее соотношение энергопотребление – производительность, так как потребление КМОП микросхем, как известно, определяется их рабочей частотой.

*EEPROM* блок электрически стираемой памяти предназначен для энергонезависимого хранения данных. Это калибровочные коэффициенты, константы, конфигурационные параметры системы. *EEPROM* память имеет меньшую, по сравнению с *FLASH*, емкость, но при этом допускает возможность побайтной перезаписи ячеек, которая может происходить как под управлением внешнего процессора, так и под управлением собственно *AVR* - микроконтроллера во время его работы в соответствии с программой.

Программирование энергонезависимых блоков памяти может осуществляться как параллельно, так и последовательно через *SPI* (*Serial Peripheral Interface*) интерфейс.

Управление и обмен данными с *EEPROM* памятью и со всеми периферийными узлами осуществляется при помощи регистров ввода/вывода, которые имеются в каждом периферийном узле.

Рассмотрим подробнее функционирование микропроцессорного ядра, которое на рис. 11.4 обведено пунктирной линией и заштриховано. Это так называемое центральное процессорное устройство (ЦПУ). В состав ЦПУ входят следующие блоки:

### **Память программ**

В памяти программ (*Program Memory*) хранится программа, в соответствии с которой работает микроконтроллер. Эта память представляет собой множество 4К 16-разрядных ячеек (все ячейки последовательно пронумерованы от 000 до FFF), в каждой из которых может помещаться одна команда программы.

Программа может быть написана на Ассемблере, отлажена с помощью симулятора или эмулятора, преобразована Ассемблером в загружаемый код, который и записывается в память программ микроконтроллера.

Программа записывается в память программ с помощью либо процедуры параллельного программирования (используется параллельный программатор), либо с помощью процедуры последовательного программирования (используется последовательный программатор). Параллельное программирование осуществляется существенно быстрее, чем последовательное программирование микроконтроллера.

При параллельном программировании доступны некоторые специальные функции микроконтроллера (например, установка запрета чтения памяти программ), которые недоступны при последовательном программировании. Последовательное программирование имеет свои достоинства, главным из которых является возможность программирования в системе, то есть в составе устройства, в которое установлен микроконтроллер. Кроме того, последовательный программатор проще параллельного.

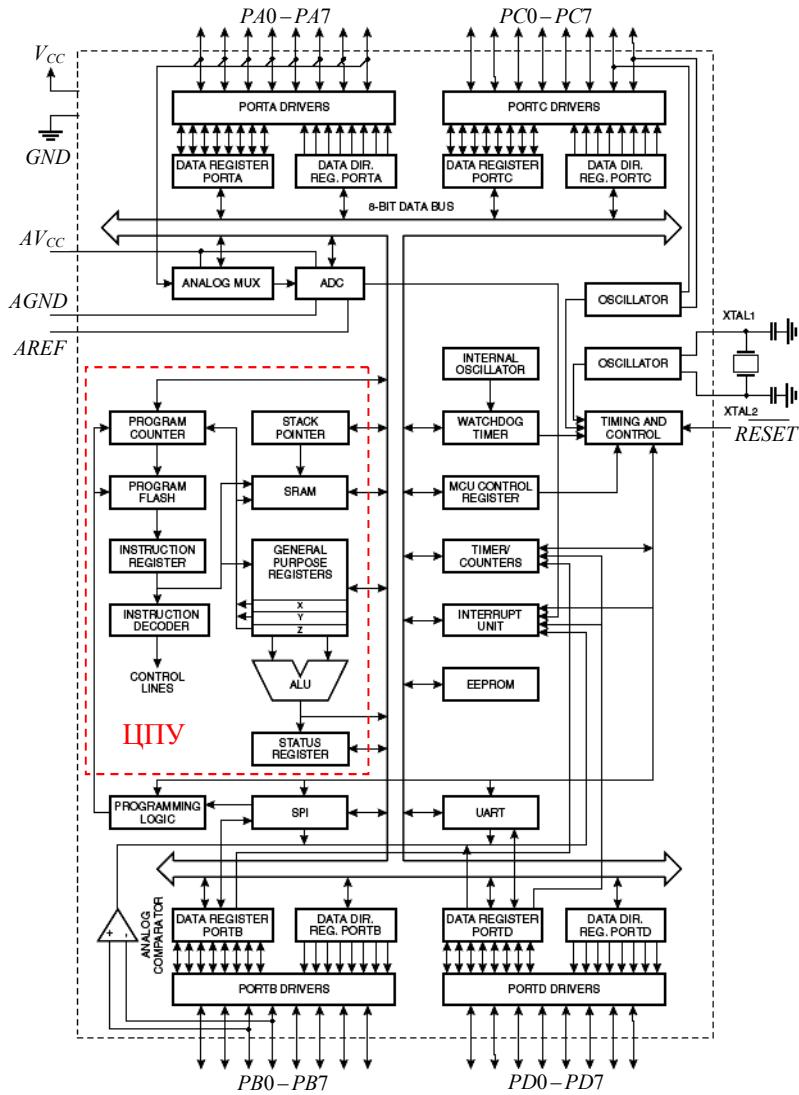


Рис. 11.4. Внутренняя структура микроконтроллера ATmega8535

## Программный счетчик (*Program counter*)

Это 12-разрядный счетчик, который содержит адрес выби-  
раемой из памяти программ команды. Сигнал *RESET* обнуляет  
программный счетчик, поэтому после окончания сигнала *RESET*  
программа всегда начинает выполнять команду, хранящуюся по  
адресу 000. После каждого использования содержимое про-  
граммного счетчика автоматически увеличивается на единицу,  
что обеспечивает последовательное выполнение команд в той по-  
следовательности, в какой они расположены в памяти программ.  
Однако при написании программ часто возникает необходимость  
изменять естественный порядок выполнения команд. Для этого в  
системе команд микроконтроллера имеются команды (31 коман-  
да), которые могут изменять содержимое программного счетчика.  
Это так называемые команды передачи управления, позволяющие  
менять естественный порядок выполнения команд. Примером та-  
кой команды может служить команда безусловной передачи  
управления *rjmp k*. Она изменяет содержимое программного  
счетчика. Новое содержимое программного счетчика с учетом его  
предыдущего состояния вычисляется следующим образом:

$$PC \leftarrow PC + k + 1, \quad -2K \leq k \leq 2K.$$

Смещение *k* – двенадцатиразрядное двоичное число. Для  
данного микроконтроллера, у которого память программ состав-  
ляет  $4K$  команд, с помощью данной команды можно передать  
управление из любого места памяти в любое другое.

Ассемблер позволяет с автоматически определить величи-  
ну смещения. В команде *rjmp k* вместо смещения *k* указывается  
метка, на которую надо передать управление. При трансляции  
Ассемблер замещает метку вычисленным значением смещения.

Рассмотрим пример Ассемблерной программы:

<i>loop:</i>	<i>nop</i>	; нет операции
<i>rjmp</i>	<i>loop</i>	; переход на метку <i>loop</i> .

При трансляции Ассемблер подставит в команду *rjmp* в качестве *k* код *3FE*, что и обеспечит передачу управления на одну команду вверх. После метки следует ставить двоеточие. После точки с запятой идет комментарий – произвольный набор символов до конца строки, который не учитывается при трансляции.

### **Регистр команд**

Регистр команд (*Instruction Register*) микроконтроллера *AT90S8535* имеет 16 разрядов. В этот регистр поочередно поступают команды, считываемые из программной памяти.

### **Декодер команд**

Декодер команд (*Instruction Decoder*) анализирует содержимое регистра команд и вырабатывает все сигналы, необходимые для выполнения текущей команды.

### **Блок регистров общего назначения**

Блок регистров общего назначения (*General Purpose Registers*) содержит 32 восьмиразрядных регистра, содержимое которых используют в своей работе многие команды микроконтроллера (77 команд). Все эти регистры пронумерованы и при написании программы на Ассемблере обозначаются *r0, r1, ..., r31*.

В качестве примера можно привести некоторые команды, использующие содержимое этих регистров:

```
mov r5, r3 ;содержимое регистра r3 копируется в регистр r5,  
add r30, r15 ;складывается содержимое регистров r30 и  
               r15 , результат помещается в регистр r30.
```

В таких командах всегда после кода операции первым указывается регистр-приемник (регистр, в который будет записан результат операции), а затем через запятую указывается регистр-источник (регистр, из которого берется второй байт, необходимый для выполнения операции).

мый для выполнения команды). Содержимое регистра-источника при выполнении данных команд не изменяется.

## Арифметико-логическое устройство

Арифметико-логическое устройство (*ALU*) осуществляет арифметические и логические операции над байтами в соответствии с системой команд. По итогам некоторых операций могут изменяться значения разрядов регистра признаков (*Status Register*).

## Регистр признаков

Регистр признаков (*Status Register*) – 8-разрядный регистр, назначение разрядов которого приведено на рис. 11.5. Этот регистр иногда называют регистром флагов.

Bit	7	6	5	4	3	2	1	0	
Адрес \$3F	I	T	H	S	V	N	Z	C	SREG
Initial value	0	0	0	0	0	0	0	0	

Рис. 11.5. Назначение разрядов регистра признаков

Сигнал  $\overline{RESET}$  устанавливает все разряды регистра признаков в нулевое состояние. Регистру признаков в Ассемблере присвоено имя *SREG*, и к нему можно обращаться с помощью команд *IN* (чтение) и *OUT* (запись) по адресу \$3F.

Для каждого разряда регистра признаков имеется команда, устанавливающая в этом разряде единицу, а также команда, устанавливающая в этом разряде нуль:

```
ses      ;установка в 1 четвертого бита (s) регистра при-
          ;знаков,
cls      ;установка в 0 четвертого бита (s) регистра при-
          ;знаков.
```

Рассмотрим назначение разрядов регистра признаков.

Бит  $7 - I$ , глобальное разрешение прерываний. Если в этом разряде установлен нуль, то никакие прерывания не будут обрабатываться микроконтроллером. Для разрешения работы системы прерываний необходимо установить в этом разряде единицу.

При возникновении любого прерывания этот бит автоматически обнуляется, а при выходе из прерывания, которое осуществляется с помощью команды *IRET* (*Interrupt Return*), автоматически устанавливается в единицу.

Бит  $6 - T$  предназначен для временного хранения бита. Команда *BST* позволяет запомнить в этом разряде любой бит из любого регистра общего назначения. Команда *BLD* позволяет записать содержимое этого разряда в любой бит любого регистра общего назначения. Таким образом, этот разряд регистра признаков позволяет с помощью команд *BST* и *BLD* передавать бит из одного регистра общего назначения в другой:

```
bst    r31, 7      ;запись значения седьмого разряда регистра  
                  ;r31 в T  
bld    r0, 3       ;запись из T в третий разряд регистра r0.
```

Остальные разряды регистра признаков могут устанавливаться в 1 или 0 по итогам выполнения арифметических и логических операций в АЛУ. При описании системы команд для каждой команды указывается, какие разряды регистра признаков она может изменять.

Бит  $5 - H$ , признак переноса между полубайтами. Устанавливается в единицу, если при выполнении операции в АЛУ имел место перенос между полубайтами.

Бит  $4 - S$  равен сумме по модулю 2 содержимого третьего и второго разрядов регистра признаков.

Бит  $3 - V$ , признак переполнения. Используется для организации вычислений с удвоенной длиной слова (16 бит).

Бит  $2 - N$ , признак отрицательного результата.

Бит  $1 - Z$ , признак нуля. Устанавливается в 1, если при выполнении операции в АЛУ получается нулевой результат (все разряды равны нулю).

Бит 0 –  $C$ , признак переноса. Устанавливается в 1, если при выполнении операции в АЛУ образуется перенос из старшего разряда.

Значение разрядов регистра признаков используется командами условной передачи управления. Каждому разряду регистра признаков соответствуют две дополняющие друг друга команды условной передачи управления. Ниже приведен пример команд *breq* (*branch if equal zero*) и *brne* (*branch if not equal zero*), использующих признак нуля  $Z$ :

*breq k* ; если  $Z=1$ , то  $PC = PC + k + 1$ , иначе  $PC = PC + 1$ ,  
– 64 ≤  $k$  ≤ 63  
*brne k* ; если  $Z=0$ , то  $PC = PC + k + 1$ , иначе  $PC = PC + 1$ ,  
– 64 ≤  $k$  ≤ 63 .

Смещение  $k$  не может превышать 64, поэтому команды этого класса не позволяют передавать управление из любого места памяти в любое другое, как это может делать команда *jmp*.

При написании программы на Ассемблере вместо смещения  $k$  указывается метка, на которую надо передать управление:

```
cp      r1,r0    ; сравнение содержимого r1 и r0
breq   equal    ; переход, если r1=r0
...
equal:  nop      ; начало подпрограммы.
```

При трансляции Ассемблер автоматически замещает метку вычисленным значением смещения.

## Память чисел

Память чисел (*Data SRAM*) емкостью 512 байт предназначена для временного хранения данных. В частности, память чисел используется для запоминания содержимого программного счетчика и регистров общего назначения при обработке прерываний. При выключении питания микроконтроллера содержимое памяти чисел не сохраняется.

## Указатель стека

Указатель стека (*Stack Pointer*) предназначен для организации работы со стеком. Указатель стека имеет 10 разрядов и состоит из двух восьмиразрядных регистров (*SPH* – старший байт, *SPL* – младший байт). К этим регистрам можно обращаться с помощью команд *IN* и *OUT*, при этом *SPH* имеет адрес *\$3E*, а *SPL* имеет адрес *\$3D* в пространстве адресов ввода/вывода. После сигнала *RESET* эти регистры обнуляются.

Стеком называется область памяти чисел (ОЗУ), обращение к которой происходит с помощью стековых операций. Стековые операции записи/считывания представлены в системах команд всех без исключения ЭВМ, так как они чрезвычайно удобны для работы с массивами, которые представляют собой линейный список данных. В своей работе эти команды в качестве адреса используют содержимое указателя стека *SP*.

Текущее содержимое *SP* определяет положение вершины стека. После каждой записи в стек содержимое указателя стека автоматически уменьшается на единицу, и вершина стека перемещается в сторону младших адресов. Для записи в память массива данных достаточно занести в *SP* начальный адрес массива и затем последовательно осуществлять запись элементов массива с помощью стековой операции. В результате массив данных расположится в памяти, начиная с исходного адреса, а вершина стека переместится в сторону младших адресов на соответствующее количество ячеек.

При чтении из стека содержимое *SP* сначала автоматически увеличивается на единицу, а затем используется в качестве адреса, по которому происходит чтение из ОЗУ. В результате элементы массива будут извлекаться из памяти в порядке, обратном тому, в котором они записывались (последний записанный байт будет прочитан первым).

В системе команд микроконтроллера *ATmega8535* имеется 6 команд, которые работают со стеком. Это команды *RCALL*, *ICALL*, *RET*, *IRET*, *PUSH*, *POP*. При обработке прерываний со-

держимое программного счетчика автоматически запоминается в стеке, одновременно осуществляется общий запрет прерываний.

Команда *RCALL* (*Relative Call to Subroutine*) – относительный переход на подпрограмму, аналогична команде *RJMP* за исключением того, что содержимое программного счетчика предварительно запоминается в стеке. Тем самым создается возможность вернуться к прежнему содержимому программного счетчика и продолжить выполнение основной программы.

Команда *ICALL* (*Indirect Call to Subroutine*) позволяет работать с 16-разрядным программным счетчиком, при работе с микроконтроллером *ATmega8535* можно обходиться без этой команды, заменяя ее командой *RCALL*.

Команда *RET* возвращает в программный счетчик то содержимое, которое было переписано в стек командой *RCALL* или *ICALL*. Команда *RET* должна быть последней командой подпрограммы.

Команда *IRET* отличается от команды *RET* тем, что она одновременно с восстановлением прежнего значения программного счетчика осуществляет общее разрешение прерываний. Команда *IRET* должна быть последней командой в программе обработки прерываний.

Команда *PUSH* позволяет записать в стек содержимое любого регистра общего назначения.

Команда *POP* позволяет извлечь из стека байт и записать его в любой регистр общего назначения.

Приведем пример использования стека при вызове подпрограммы:

```
rcall routin
; вызов подпрограммы routin, содержимое программного
; счетчика заносится в стек, PC=PC+k+1, SP=SP-2
; k – смещение, вычисляется Ассемблером, -2K≤k≤2K.
```

```
    nop
; продолжение основной программы
```

```
...
```

```
routin: push r14
```

```

; сохранение содержимого регистра r14 в стеке, PC=PC+1,
; SP=SP-1
...
; тело подпрограммы
    pop r14
; восстановление содержимого регистра r14 из стека
; PC=PC+1, SP=SP+1
    ret
; возврат из подпрограммы, два байта из стека
; переписываются в программный счетчик, SP=SP+2.

```

После выполнения команды *RET* указатель стека *SP* имеет то же значение, что и перед выполнением команды *RCALL*. Содержимое программного счетчика *PC* становится таким, что следующей выполняется команда, непосредственно расположенная после команды *RCALL* (в данном случае команда *NOP*). Таким образом, команда *RCALL* позволяет на некоторое время выйти из основной программы, выполнить некоторые действия с помощью подпрограммы *ROUTIN*, а затем продолжить выполнение основной программы. Например, подпрограмма *ROUTIN* может вычислить значение некоторой функции от аргумента, который основная программа поместила в регистр *r0*, а результат вычисления вернуть в тот же регистр *r0*. При этом подпрограмма *ROUTIN* использует в своей работе регистр *r14*.

Прежде чем начать использование стека, в указатель стека (*SPH* и *SPL*) необходимо занести начальный адрес вершины стека. Этот адрес не может быть меньше \$60.

## Управляющий регистр микроконтроллера

Управляющий регистр микроконтроллера (*MCU Control Register*) имеет 8 разрядов. Этому регистру присвоено имя *MCUCR* и обращение к нему осуществляется командой *IN* и *OUT* по адресу \$35. Назначение разрядов управляющего регистра приведено на рис. 11.6.

Bit	7	6	5	4	3	2	1	0	
Address \$35	—	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Initial value	0	0	0	0	0	0	0	0	

Рис. 11.6. Назначение разрядов управляющего регистра

Сигнал  $\overline{RESET}$  обнуляет все разряды этого регистра. Поэтому если внешние прерывания должны восприниматься уровнем (низкий уровень на соответствующем входе) или не используются, если не используется режим пониженного энергопотребления (*Sleep Modes*), то нет необходимости изменять значения разрядов этого регистра. В этом случае можно ничего не знать о назначении разрядов управляющего регистра и пропустить следующую ниже информацию:

Бит 6 – *SE (Sleep Enable)*.

Единица в этом разряде разрешит команде *SLEEP* включить спящий режим (режим пониженного энергопотребления). Рекомендуется устанавливать в этом разряде единицу непосредственно перед командой *SLEEP*.

Бит 4, 5, 7 – *SM0, SM1* и *SM2 (Sleep Mode)*.

Содержимое этих разрядов указывает, какой из возможных вариантов режима пониженного энергопотребления будет реализован. Более подробная информация содержится в полном описании микроконтроллера.

Бит 3,2 – *ISC11, ISC10 (Interrupt Sense Control)*.

00 – прерывание вырабатывается при низком уровне на входе *INT1*; 01 – резерв; 10 – прерывание вырабатывается при переходе сигнала на входе *INT1* от высокого уровня к низкому уровню; 11 – прерывание вырабатывается при переходе сигнала на входе *INT1* от низкого уровня к высокому.

Бит 1,0 – *ISC01, ISC00. (Interrupt Sense Control)*.

00 – прерывание вырабатывается при низком уровне на входе *INT0*; 01 – резерв; 10 – прерывание вырабатывается при переходе сигнала на входе *INT0* от высокого уровня к низкому уровню;

11 – прерывание вырабатывается при переходе сигнала на входе *INT0* от низкого уровня к высокому.

Микроконтроллер становится законченным устройством только тогда, когда к микропроцессорному ядру подключаются дополнительные блоки, которые на рис. 11.4 расположены за пределами пунктирной линии. Различные микроконтроллеры (одной серии) обычно имеют одно и то же микропроцессорное ядро и отличаются только набором дополнительных блоков, объемом памяти программ и памяти чисел.

## 11.4. Дополнительные блоки микроконтроллера

Каждый из дополнительных блоков обслуживается некоторым количеством так называемых регистров ввода/вывода. В микроконтроллере ATmega8535 используются 64 таких регистра.

К регистрам ввода/вывода можно обращаться с помощью команд *IN* (чтение) и *OUT* (запись). Для обеспечения такого обращения каждому регистру присвоен свой адрес и имя. Соответствующая информация приведена в табл. 11.1. Так, сторожевой таймер обслуживается одним регистром (*WDTCR*) с адресом \$21, а таймер/счетчик 1 обслуживают 12 регистров ввода/вывода.

В качестве названий регистров и их отдельных разрядов разработчики микроконтроллера использовали сокращения, некоторые из которых понятны (*PORTA*, *PINA*), а некоторые требуют расшифровки, которая приводится по мере необходимости.

Для успешного построения различных устройств на основе микроконтроллеров надо хорошо понимать, как работают те или иные дополнительные блоки. А для этого надо точно знать, что произойдет, если в тот или иной регистр ввода/вывода, обслуживающий данный блок, записать некоторый набор нулей и единиц (байт) с помощью команды *OUT*. Также надо понимать, что означает информация, прочитанная из того или иного регистра ввода/вывода с помощью команды *IN*.

Прежде чем перейти к изучению работы дополнительных блоков, рассмотрим на примере выполнение команд *OUT* и *IN*:

*in r25, \$18 ; чтение содержимого регистра с именем PORTB, результат заносится в r25*

*out \$1B, r25 ; запись содержимого r25 в регистр с именем PORTA.*

В команде *IN* сначала указывается номер регистра общего назначения, в который будет помещен прочитанный байт, а далее после запятой указывается адрес регистра ввода/вывода, из которого происходит чтение.

В команде *OUT* сначала указывается номер регистра ввода/вывода, в который будет записан байт, а далее после запятой указывается адрес регистра общего назначения, из которого этот байт будет взят.

Таким образом, приведенные выше две команды обеспечивают передачу байта из регистра с именем *PORTB* в регистр с именем *PORTA*.

Таблица 11.1

Таблица регистров ввода-вывода

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$3F	SREG	I	T	H	S	V	N	Z	C
\$3E	SPH	—	—	—	—	—	—	SP9	SP8
\$3D	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
\$3B	GIMSK	INT1	INT0	—	—	—	—	—	—
\$3A	GPIO	INTF1	INTF0	—	—	—	—	—	—
\$39	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	—	TOIE0
\$38	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	—	TOV0
\$35	MCUCR	—	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
\$34	MSUSR	—	—	—	—	—	—	EXTRF	PORF
\$33	TCCR0	—	—	—	—	—	—	CS02	CS01
\$32	TCNT0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$2F	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	—	—	PWM11	PWM10
\$2E	TCCR1B	ICNC1	ICES1	—	—	CTC1	CS12	CS11	CS10
\$2D	TCNT1H	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 8	Bit 8
\$2C	TCNT1L	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$2B	OCR1AH	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 8	Bit 8
\$2A	OCR1AL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$29	OCR1BH	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 8	Bit 8
\$28	OCR1BL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$27	ICR1H	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 8	Bit 8
\$26	ICR1L	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$25	TCCR2	—	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20
\$24	TCNT2	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$23	OCR2	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$22	ASSR	—	—	—	—	AS2	TCN2UB	OCR2UB	TCR2UB
\$21	WDTCR	—	—	—	WDTOE	WDE	WDP2	WDP1	WDP0
\$1F	EEARH	—	—	—	—	—	—	—	EEAR8
\$1E	EEARL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$1D	EEDR	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

## Продолжение таблицы 11.1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$1C	EECR	—	—	—	—	EER1E	EEMWE	EEWE	EERE
\$1B	PORTA	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$1A	DDRA	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$19	PINA	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$18	PORTB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$17	DDRB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$16	PINB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$15	PORTC	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$14	DDRC	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$13	PINC	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$12	PORTD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$11	DDRD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$10	PIND	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$0F	SPDR	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$0E	SPSR	SPIF	WCOL	—	—	—	—	—	—
\$0D	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
\$0C	UDR	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$0B	USR	RxC	TxC	UDRE	FE	OR	—	—	—
\$0A	UCR	RxCIE	TxCIE	UDRIE	RxEN	TxEN	CHR9	RxB8	TxB8
\$09	UBRR	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
\$08	ACSR	ACD	—	AC0	AC1	ACIE	ACIC	ACIS1	ACIS0
\$07	ADMUX	—	—	—	—	—	MUX2	MUX1	MUX0
\$06	ADCSR	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
\$05	ADCH	—	—	—	—	—	—	Bit 9	Bit 8
\$04	ADCL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

### 11.4.1. Порт A

Порт *A* является одним из дополнительных блоков, предназначенный для двунаправленной передачи байтов между микроконтроллером и внешним миром. Для этого порт *A* использует восемь выводов микроконтроллера. Каждому из этих выводов однозначно соответствует один разряд порта *A* (вывод 40 – *PA0*, вывод 33 – *PA7*). Эти выводы называются выводами порта *A* (*PA0* ... *PA7*). Порт *A* обслуживают три регистра ввода/вывода. Это регистр *PORTA* (*Data Register*) с адресом \$1B, *DDRA* (*Data Direction Register*) с адресом \$1A, регистр *PINA* (*Port A Input Pins*) с адресом \$19. Содержимое регистра *PINA* можно только читать, а регистры *PORTA* и *DDRA* допускают как чтение, так и запись. Рассмотрим роль этих регистров в работе порта *A*.

*DDRA* — регистр выбора направления передачи данных порта *A*. Этот регистр имеет восемь разрядов и каждый разряд управляет направлением передачи данных через соответствующий вывод порта *A*. Если в некоторый разряд регистра *DDRA* за-

писать 1, то соответствующий ему вывод порта *A* станет выходом. Если же в некоторый разряд регистра *DDRA* записать 0, то соответствующий ему вывод порта *A* станет входом. После сигнала *RESET* все разряды регистра *DDRA* имеют нулевое значение, поэтому после сигнала *RESET* все выводы порта *A* являются входами.

*PORTA* – регистр данных порта *A*. Каждому разряду регистра *PORTA* соответствует один вывод порта *A*. Но выходы регистра *PORTA* соединены только с теми выводами порта *A*, которые запрограммированы в качестве выходов. Поэтому разряды байта, записанного в *PORTA*, появятся только на тех выводах порта *A*, которые запрограммированы с помощью регистра *DDRA* в качестве выходов. При чтении *PORTA* будет прочитано то, что ранее в него было записано. После сигнала *RESET* *PORTA* содержит все нули.

*PINA* – строго говоря, это не регистр. При чтении *PINA* будет прочитано то, что в данный момент имеет место на выводах порта *A*. Если все выводы порта *A* запрограммированы как выходы, то чтение *PINA* и чтение *PORTA* будет давать один и тот же результат.

Для порта предусмотрена дополнительная возможность, которая облегчает подключение к выводам порта кнопок и датчиков (не требуется включать внешние резисторы). Существует возможность подключения вывода порта, который запрограммирован в качестве входа, к источнику питания  $V_{CC}$  через внутренний резистор (порядка 100 кОм). На самом деле при этом используется не резистор, а полевой транзистор в резисторном включении.

Для включения этого подтягивающего резистора к некоторому выводу, запрограммированному в качестве входа, достаточно записать единицу в тот разряд регистра *PORTA*, который соответствует этому выводу. Теперь можно подключать к этому выводу порта нормально разомкнутую кнопку, второй контакт которой заземляют. При разомкнутой кнопке на входе будет высокий уровень (лог. 1), а при нажатой кнопке – низкий уровень

(лог. 0). Если бы не было подтягивания, то при разомкнутой кнопке сигнал на этом входе имел бы неопределенное значение.

#### 11.4.2. Порты B, C, D

Все сведения о порте *A* в полной мере относятся и к порту *B*. Отличие заключается в том, что порт *B* использует другие выводы микроконтроллера (*PB0* – вывод 1, *PB7* – вывод 8), а обслуживающие его регистры имеют другие адреса: *PINB* – \$16, *DDRB* – \$17, *PORTB* – \$18.

Порт *C* использует другие выводы микроконтроллера (*PC0* – вывод 22, *PC7* – вывод 29), а обслуживающие его регистры имеют другие адреса: *PINC* – \$13, *DDRC* – \$14, *PORTC* – \$15.

Отличие порта *D* от порта *A* заключается в том, что порт *D* использует другие выводы микроконтроллера (*PD0* – вывод 14, *PD7* – вывод 21), а обслуживающие его регистры имеют другие адреса: *PIND* – \$10, *DDRD* – \$11, *PORTD* – \$12.

#### 11.4.3. Система прерываний

Обычно микроконтроллер выполняет некоторую основную программу и в процессе ее выполнения могут возникнуть события, которые требуют внимания со стороны микропроцессорного ядра. Например, таким событием может быть нажатие кнопки, подключенной к входу *INT0* или переполнение счетчика/таймера. Такие события называются запросами прерывания. Обнаружив запрос, система прерываний должна обеспечить приостановку работы основной программы и запуск некоторой программы, которую называют программой обработки прерывания. Для каждого прерывания должна быть написана своя программа обработки, которая размещается в памяти программ. Программа обработки прерывания выполняет действия, которые надо выполнить в качестве реакции на данное прерывание. Например, осуществляет чтение из одного или нескольких регистров или записывает необходимую информацию в те или иные регистры. Закончив свою работу, программа обработки прерывания должна обеспечить возвращение к прерванной основной программе.

Микроконтроллер ATmega8535 обладает достаточно простой и эффективной системой прерываний. Всего имеется 21 прерывание, четыре из которых являются внешними и вызываются сигналами, приходящими на выводы микроконтроллера *INT0*, *INT1*, *INT2* и *RESET*. Остальные 17 прерываний являются внутренними и обслуживают дополнительные блоки.

Все запросы прерываний поступают на блок обработки прерываний (*Interrupt Unit*). Обнаружив запрос, блок обработки прерываний определяет его номер (все запросы прерываний пронумерованы и имеют номера от 1 до 21). Чувствительность к новым прерываниям запрещается (в седьмой разряд регистра флагов записывается 0) на время обработки данного прерывания. После этого текущее содержимое программного счетчика заносится в стек, а на его место в программный счетчик заносится адрес, приписанный данному прерыванию. Номера прерываний и соответствующие им адреса приведены в табл. 11.2.

Если одновременно возникнет несколько прерываний, то первым будет обрабатываться прерывание, имеющее наименьший номер. Таким образом, в данном микроконтроллере используется система прерываний с фиксированными приоритетами.

Самое большое число прерываний, равное 4, принадлежит таймеру 1. Блоку *USART* (универсальный синхронно-асинхронный приемопередатчик) принадлежат три прерывания. Таймеру 2 отведено два прерывания. Остальные прерывания распределены по одному в соответствии с табл. 11.2.

Адрес, приведенный в табл. 11.2, является адресом начала программы обработки прерывания. Из табл. 11.2 следует, что каждая из 21 программы обработки прерываний имеет свое начало в первых 21 ячейках памяти программ, начиная с нулевого адреса. А поскольку в каждой ячейке памяти можно разместить только одну команду, то этой командой в таблице векторов прерываний должна быть команда *rjmp*.

Таблица 11.2

Таблица прерываний

N	Адрес	Источник	Причина прерывания
1	\$000	RESET	Сигнал на входе <i>RESET</i> , включение питания, сигнал от сторожевого таймера
2	\$001	INT0	Внешний запрос прерывания 0
3	\$002	INT1	Внешний запрос прерывания 1
4	\$003	TIMER2 COMP	Достижение порога в таймере 2
5	\$004	TIMER2 OVF	Переполнение в таймере 2
6	\$005	TIMER1 CAPT	Запоминание содержимого таймера 1 в регистре <i>ICR1</i>
7	\$006	TIMER1 COMPA	Достижение порога A в таймере 1
8	\$007	TIMER1 COMPB	Достижение порога B в таймере 1
9	\$008	TIMER1 OVF	Переполнение в таймере 1
10	\$009	TIMER0 OVF	Переполнение в таймере 0
11	\$00A	SPI, STC	<i>SPI</i> передачу закончил
12	\$00B	UART, RX	<i>UART</i> прием закончил
13	\$00C	UART, UDRE	Регистр данных <i>UART</i> пуст
14	\$00D	UART, TX	<i>UART</i> передачу закончил
15	\$00E	ADC	Аналого-цифровое преобразование завершено
16	\$00F	EE_RDY	<i>EEPROM</i> готов к новой записи
17	\$010	ANA_COMP	Аналоговый компаратор
18	\$011	TWI	Прерывание двухпроводного интерфейса
19	\$012	INT2	Внешний запрос прерывания 2
20	\$013	TIMER0	Достижение порога в таймере 0
21	\$014	SPM-RDY	Память программ готова к записи

Ниже приведено стандартное начало любой программы обработки прерываний:

Адрес	Метка	Команда	Комментарий
\$000	rjmp RESET		;Переход на программу <i>RESET</i>
\$001	rjmp EXT_INT0		;Обработка прерывания <i>INT0</i>
\$002	rjmp EXT_INT1		;Обработка прерывания <i>INT1</i>
\$003	rjmp TIM2_COMP		;Обработка прерывания таймера 2
\$004	rjmp TIM2_OVF		;Обработка прерывания таймера 2
\$005	rjmp TIM1_CAPT		;Обработка прерывания таймера 1
\$006	rjmp TIM1_COMPA		;Обработка прерывания таймера 1
\$007	rjmp TIM1_COMPB		;Обработка прерывания таймера 1
\$008	rjmp TIM1_OVF		;Обработка прерывания таймера 1

\$009	<i>rjmp</i>	<i>TIM0_OVF</i>	;Обработка прерывания таймера 0
\$00A	<i>rjmp</i>	<i>SPI_HANDLER</i>	;Обработка прерывания <i>SPI</i>
\$00B	<i>rjmp</i>	<i>USART_RXC</i>	;Обработка прерывания <i>USART</i>
\$00C	<i>rjmp</i>	<i>USART_DRE</i>	;Обработка прерывания <i>USART</i>
\$00D	<i>rjmp</i>	<i>USART_TXC</i>	;Обработка прерывания <i>USART</i>
\$00E	<i>rjmp</i>	<i>ADC</i>	;Обработка прерывания <i>ADC</i>
\$00F	<i>rjmp</i>	<i>EE_RDY</i>	;Обработка прерывания <i>EEPROM</i>
\$010	<i>rjmp</i>	<i>ANA_COMP</i>	;Обработка прерывания компаратора
\$011	<i>rjmp</i>	<i>TWSI</i>	;Обработка прерывания <i>TWI</i>
\$012	<i>rjmp</i>	<i>EXT_INT2</i>	;Обработка прерывания <i>INT2</i>
\$013	<i>rjmp</i>	<i>TIM0_COMP</i>	;Обработка прерывания таймера 0
\$014	<i>rjmp</i>	<i>SPM_RDY</i>	;Обработка прерывания памяти
\$015	<i>ldi</i>	<i>r16,high(RAMEND)</i>	;Начало основной программы
\$016	<i>out</i>	<i>SPH,r16</i>	;Установка в указателе стека максимального адреса <i>RAM</i>
\$017	<i>ldi</i>	<i>r16,low(RAMEND)</i>	
\$018	<i>out</i>	<i>SPL,r16</i>	
\$019	<i>sei</i>		;Разрешение прерываний
\$020	< <i>instr</i> >	xxxx	

Если какое-либо прерывание не используется, то в соответствующей ему ячейке вместо команды *RJMP* помещают команду *NOP* (нет операции).

В написании меток, на которые передают управление команды *RJMP*, применен символ подчеркивания. Это связано с тем, что в написании меток не может содержаться символ пробела. Существуют и некоторые другие ограничения на написание меток.

Все программы обработки прерываний должны иметь некоторые общие свойства. Так, в начале программы обработки прерываний должен быть блок, который обеспечит запоминание в стеке содержимого тех регистров микроконтроллера, которые могут быть изменены программой обработки данного прерывания. К этим регистрам чаще всего относится регистр флагов. В конце программы обработки прерывания необходимо вернуть

из стека содержимое этих регистров. Последней командой программы обработки прерываний должна быть команда *IRET*, которая перепишет в программный счетчик из стека прежнее содержимое и разрешит прерывания (установит 1 в седьмой разряд регистра флагов). Далее продолжится работа прерванной основной программы.

Разрешение прерываний можно осуществить и в начале программы обработки прерывания. В этом случае становятся возможными вложенные прерывания. Однако этим следует пользоваться осторожно, поскольку объем стека ограничен (память чисел – 512 байт) и возможно его переполнение.

Причины, по которым возникает то или иное прерывание, необходимо рассматривать в каждом конкретном случае. Так, первое прерывание возникает тогда, когда включается питание или на входе *RESET* появляется низкий уровень. Прерывание по этому входу запрещено быть не может. Другой причиной возникновения этого прерывания является переполнение в сторожевом таймере. Однако в этом случае прерывание будет обрабатываться только при условии активизации сторожевого таймера.

Прерывания 2, 3 и 19 вызываются сигналами на входах *INT0*, *INT1* и *INT2* соответственно. Вид сигнала, который приведет к прерыванию, программируется с помощью некоторых разрядов управляющего регистра микроконтроллера, который был описан выше. Эти прерывания могут индивидуально быть разрешены с помощью регистра *GIMSK* (*The General Interrupt Mask Register*). Если в разряде 7 этого регистра записать 1, то будет разрешено прерывание по входу *INT1*. Единица в шестом разряде разрешает прерывание по входу *INT0*, а единица в пятом разряде разрешает прерывание по входу *INT2*. Остальные разряды этого регистра не используются. Эти прерывания не будут обрабатываться, если в разряде 7 регистра флагов стоит 0, то есть имеет место общее запрещение прерываний. Однако факт возникновения соответствующего сигнала на входе запоминается в регистре *GIFR* (*The General Interrupt Flag Register*). Разряд 7 этого регистра – флаг *INT1*, разряд 6 – флаг *INT0*, разряд 5 – флаг *INT2*. Остальные разряды этого регистра не используются. Если после

общего разрешения прерываний в разрядах 5, 6 и 7 будет хотя бы одна единица, то соответствующее прерывание начнет обрабатываться (если оно не запрещено с помощью регистра *GIMSK*). При этом прерывание *INT0* имеет более высокий приоритет.

#### 11.4.4. Таймер/счетчик 0

Восьмиразрядный таймер/счетчик (*Timer/Counter 0*) будем сокращенно называть таймер 0. Этот таймер используется в большинстве приложений, поскольку практически всегда необходимо вырабатывать заданные интервалы времени. Таймер 0 имеет в своем исключительном распоряжении 2 регистра ввода/вывода, и еще два регистра он использует совместно с таймером 1 и таймером 2.

Счет осуществляется в регистре *TCNT0* (*The Timer Counter 0*). Сигнал *RESET* обнуляет содержимое этого регистра. Каждый пришедший на его вход импульс увеличивает содержимое этого регистра на единицу. В регистр *TCNT0* можно писать байт с помощью команды *IN*, а его содержимое можно читать с помощью команды *OUT*. Приняты меры, чтобы запись и чтение можно было осуществлять без остановки счета.

Сигналы на вход *TCNT0* приходят с выхода управляемого предварительного делителя частоты (*prescaler*), аналогичного делителю таймера 2 (рис. 11.7). Предварительный делитель обеспечивает ступенчатое деление тактовой частоты от 1 до 1024. Режим работы предварительного делителя задается тремя разрядами регистра *TCCR0* (*The Timer/Counter Control Register*). При этом используются разряды 2 (*CS02*), 1 (*CS01*) и 0 (*CS00*) этого регистра.

В табл. 11.3 приведено назначение разрядов регистра *TCCR0*. В младшие три разряда регистра *TCCR0* информация записывается с помощью команды *OUT*, а читается из них с помощью команды *IN*. Адрес этого регистра в пространстве адресов ввода/вывода \$33. После сигнала *RESET* регистр *TCCR0* содержит все нули во всех разрядах, поэтому таймер остановлен, счета нет. Запись всех нулей в младшие разряды *TCCR0* используется для остановки таймера.

Таблица 11.3

## Назначение разрядов регистра TCCR0

7	6	5	4	3	2	1	0	BIT
FOC 0	WGM 00	COM 01	COM 00	WGM 01	CS 02	CS 01	CS 00	Address \$33
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
0	0	0	0	0	0	0	0	Initial value

Связь между значениями этих разрядов и режимами работы предварительного делителя поясняет табл. 11.4.

Таблица 11.4

## Таблица режимов работы предварительного делителя

CS02	CS01	CS00	Режим работы
0	0	0	Таймер остановлен, нет счета
0	0	1	CK (На вход таймера поступает тактовая частота)
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Сигнал с входа T0, счет при переходе от 1 к 0
1	1	1	Сигнал с входа T0, счет при переходе от 0 к 1

Таймер 0 способен вырабатывать прерывание *TIMER0\_OVF* (*Timer 0 Overflow*). Это прерывание вызывается переполнением в таймере 0 (переполнение – это переход из состояния *все единицы* в состояние *все нули* при приходе очередного импульса на вход счетчика). Однако это прерывание будет обрабатываться только в том случае, если в разряде 0 регистра *TIMSK* (*The Timer/Counter Interrupt Mask Register*) будет стоять единица. Этот разряд называется *TOIE0* (*Timer/Counter 0 Overflow Interrupt Enable*). Вторым условием обработки прерывания является наличие общего разрешения прерывания (единица в разряде 7 регистра флагов *SREG*). Второе прерывание *TIMER0\_COMP* соответст-

вует ситуации, когда содержимое таймера достигает величины порога, записанной в регистре *OCR0*.

В табл. 11.5 приведено назначение разрядов регистра *TIMSK*.

Т а б л и ц а 11.5

**Назначение разрядов регистра TIMSK**

7	6	5	4	3	2	1	0	BIT
OCIE 2	TOIE 2	TICIE 1	OCIE 1A	OCIE 1B	TOIE 1	OCIE 0	TOIE 0	Address \$39
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
0	0	0	0	0	0	0	0	Initial value

Регистр *TIMSK* используется таймером 0 совместно с таймером 1 и таймером 2. Разряды 2, 3, 4, 5 этого регистра позволяют разрешать или запрещать прерывания, возникающие при работе таймера 1. Условия возникновения этих прерываний будут рассмотрены при изучении работы таймера 1.

Разряды 6, 7 этого регистра позволяют разрешать или запрещать прерывания, возникающие при работе таймера 2. Условия возникновения этих прерываний будут рассмотрены при изучении работы таймера 2.

В микроконтроллере имеется регистр флагов прерываний таймеров/счетчиков. Этот регистр называется *TIFR* (*The Time/Counter Interrupt Flag Register*) с адресом \$38 в пространстве адресов ввода/вывода. Его содержимое можно читать с помощью команды *IN*. После сигнала *RESET* все разряды этого регистра равны нулю.

В табл. 11.6 приведено назначение разрядов регистра *TIFR*. Разряд 0 этого регистра называется *TOV0* (*Timer/Counter 0 Overflow Flag*). При возникновении переполнения в таймере 0 он устанавливается в 1. Если эта единица возникла при разрешенном прерывании таймера 0, то прерывание начинает обрабатываться, а единица в разряде 1 регистра *TIFR* автоматически сбрасывается. Если же *TOV0* = 1 при запрещенном прерывании таймера 0, то прерывание обрабатываться не будет. Обработка начнется только

после того, как прерывание таймера 0 будет разрешено. Одновременно с началом обработки прерывания будет автоматически сброшен флаг переполнения таймера 0 *TOV0*.

Таблица 11.6

**Назначение разрядов регистра TIFR**

7	6	5	4	3	2	1	0	BIT
OCF 2	TOV 2	ICF 1	OCF 1A	OCF 1B	TOV 1	OCF 0	TOV 0	Address \$38
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
0	0	0	0	0	0	0	0	Initial value

Регистр *TIFR* используется таймером 0 совместно с таймером 1 и таймером 2. Разряды 2,3,4,5 этого регистра являются флагами прерываний, возникающих при работе таймера 1. Они устанавливаются в 1, когда в таймере 1 происходят соответствующие события, и сбрасываются, когда прерывание начинает обрабатываться.

#### 11.4.5. Таймер/счетчик 1

16-разрядный таймер/счетчик (*Timer|Counter 1*) в дальнейшем будем называть таймер 1. Этот таймер используется для выработки интервалов времени, для подсчета числа внешних событий, для формирования сигналов с широтно-импульсной модуляцией на выходах *OC1A* и *OC1B*. Таймер 1 имеет в своем единичном распоряжении 10 регистров ввода/вывода, и еще два регистра он использует совместно с таймером 0 и таймером 2.

Счет осуществляется в 16-разрядном регистре *TCNT1*, образованном двумя регистрами ввода/вывода: *TCNT1H* (старший байт) с адресом \$2D, *TCNT1L* (младший байт) с адресом \$2C (*The Timer Counter 1*). Сигнал *RESET* обнуляет содержимое этого регистра. Каждый импульс, пришедший на вход таймера, увеличивает его содержимое на единицу. В регистр *TCNT1* можно заносить информацию с помощью команды *IN*, а читать его содержимое с помощью команды *OUT*. Запись и чтение можно осуществлять

лять без остановки счета. Запись слова надо всегда начинать со старшего байта, а чтение слова – с младшего байта.

Сигналы на вход *TCNT1* приходят с выхода управляемого предварительного делителя частоты (*prescaler*), аналогичного делителю таймера 2 (рис. 11.7). Предварительный делитель обеспечивает ступенчатое деление тактовой частоты от 1 до 1024. Режим работы предварительного делителя задается тремя разрядами регистра *TCCR1B* (*The Timer/Counter Control Register*). При этом используются разряды 2 (*CS12*), 1 (*CS11*) и 0 (*CS10*) этого регистра.

Назначение разрядов регистра *TCCR1B* приведено в табл. 11.7. Связь между значениями трех младших разрядов и режимами работы предварительного делителя поясняет табл. 11.8.

Таблица 11.7

**Назначение разрядов регистра TCCR1B**

7	6	5	4	3	2	1	0	BIT
ICNC1	ICES1	-	WGM12	WGM12	CS12	CS11	CS10	Address \$2E
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	Read/Write
0	0	0	0	0	0	0	0	Initial value

Таблица 11.8

**Таблица режимов работы предварительного делителя**

CS12	CS11	CS10	Режим работы
0	0	0	Таймер остановлен, нет счета
0	0	1	СК (На вход таймера поступает тактовая частота)
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Сигнал с входа T1, счет при переходе от 1 к 0
1	1	1	Сигнал с входа T1, счет при переходе от 0 к 1

В регистр *TCCR1B* можно записывать с помощью команды *OUT*, и из него можно читать с помощью команды *IN*. Адрес

этого регистра в пространстве адресов ввода/вывода \$2E. После сигнала *RESET* регистр *TCCR1B* содержит все нули, поэтому таймер остановлен, счета нет. Запись всех нулей в младшие разряды *TCCR1B* используется для остановки таймера.

Разряды 7, 6 и 3 этого регистра, а также еще один управляющий регистр *TCCR1A* с адресом \$2F обслуживают специальные функции таймера 1.

В таймере 1 предусмотрена возможность установки порогов (порога *A* и порог *B*), при достижении каждого из которых вырабатывается свое прерывание (*TIM1\_COMPA* и *TIM1\_COMPB*). Для установки этих порогов используются 4 регистра ввода/вывода (по два для каждого порога). Регистры *OCR1AH* (адрес \$2B) и *OCR1AL* (адрес \$2A) используются для установки значения порога *A*, старший и младший байт соответственно. Регистры *OCR1BH* (адрес \$29) и *OCR1BL* (адрес \$28) – для установки значения порога *B*, старший и младший байт соответственно. Запись в эти регистры всегда следует начинать со старшего байта.

Бит 3 регистра *TCCR1B* (*CTC1*) при записи в него единицы вызывает специальный режим работы таймера. В этом режиме после достижения порога регистр *TCNT1* обнуляется и счет продолжается. В результате таймер 1 начинает работать как счетчик по модулю *C*, где *C* – значение порога, занесенное в регистр *OCR1A*. Если при этом в регистре *TCCR1A* (адрес \$2F) разряд *COM1A0* (бит 6) равен единице, а разряд *COM1A1* (бит 7) равен нулю, то на выходе *OC1A* (вывод *PD5*) появится симметричное прямоугольное колебание, период которого будет равен  $2CxT$ . Здесь *C* – значение порога, занесенное в *OCR1A*, а *T* – период сигнала, поступающего на счетный вход таймера 1 с выхода предварительного делителя. *PD5* должен быть запрограммирован как выход.

При переполнении таймера 1 вырабатывается прерывание *TIM1\_OVF*. Еще имеется прерывание *TIM1\_CAPT*, про которое ничего здесь говорить не будем. Желающие могут с ним разобраться самостоятельно по полному описанию таймера 1.

Каждое из 4 прерываний таймера 1 может быть индивидуально разрешено или запрещено с помощью разрядов регистра *TIMSK*. При возникновении события, вызывающего прерывание, в регистре флагов прерываний таймеров *TIFR* устанавливается в единицу соответствующий разряд. Как только прерывание начинает обрабатываться, соответствующий ему разряд в регистре *TIFR* сбрасывается.

#### 11.4.6. Таймер/счетчик 2

Восьмиразрядный таймер/счетчик 2 будем сокращенно называть таймер 2. Таймер 2 имеет в своем единоличном расположении 4 регистра ввода/вывода и еще два регистра он использует совместно с таймером 1 и таймером 0.

Счет осуществляется в регистре *TCNT2* (*The Timer Counter 2*). Сигнал *RESET* обнуляет содержимое этого регистра. Каждый пришедший на его вход импульс увеличивает содержимое счетчика на единицу. В регистр *TCNT2* можно писать байт с помощью команды *IN*, а его содержимое можно читать с помощью команды *OUT*. Запись и чтение можно осуществлять без остановки счета.

Сигналы на вход *TCNT2* приходят с выхода управляемого предварительного делителя частоты (*prescaler*). Схема предварительного делителя приведена на рис.11.7.

В состав предварительного делителя входят входной мультиплексор, 10-разрядный делитель частоты с отводами и выходной мультиплексор.

Входной мультиплексор позволяет подавать на вход 10-разрядного делителя частоты либо тактовую частоту микроконтроллера (*CK*), либо сигнал *TOSC1* с выхода дополнительного генератора, входящего в состав микроконтроллера. Этот дополнительный генератор инициализируется, если бит 3 регистра *ASSR* (регистр *ASSR* имеет адрес \$22) установлен в единицу. В этом случае выводы *PC6* (*TOSC1*) и *PC7* (*TOSC2*) перестают принадлежать порту *C* и становятся выводами для подключения кварцевого резонатора частоты 32,768 кГц (это так называемый часовий кварцевый резонатор). Использование такого кварцевого

резонатора позволяет реализовать на таймере 2 часы реального времени.



Рис. 11.7. Предварительный делитель таймера 2

Предварительный делитель обеспечивает ступенчатое деление тактовой частоты от 1 до 1024. Режим работы выходного мультиплексора задается тремя разрядами регистра *TCCR2* (*The Timer/Counter Control Register*). При этом используются разряды 2 (CS22), 1 (CS21) и 0 (CS20) этого регистра. Аналогичные предварительные делители имеются у таймера 0 и таймера 1.

В табл. 11.9 приведено назначение разрядов регистра *TCCR2*.

Т а б л и ц а 11.9

**Назначение разрядов регистра *TCCR2***

7	6	5	4	3	2	1	0	BIT
FOC 2	WGM 21	COM 21	COM 20	WGM 21	CS 22	CS 21	CS 20	Address \$25
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
0	0	0	0	0	0	0	0	Initial value

Связь между значениями этих разрядов и режимами работы предварительного делителя поясняет табл. 11.10.

Таблица 11.10

Таблица режимов работы предварительного делителя

CS22	CS21	CS20	Сигнал, поступающий на счетный вход таймера 2
0	0	0	Таймер остановлен, нет счета
0	0	1	PCK2
0	1	0	PCK2/8
0	1	1	PCK2/32
1	0	0	PCK2/64
1	0	1	PCK2/128
1	1	0	PCK2/256
1	1	1	PCK2/1024

В разряды с 0 по 6 регистра *TCCR2* можно записывать с помощью команды *OUT*, и из него можно читать с помощью команды *IN*. Адрес этого регистра в пространстве адресов ввода/вывода \$25. После сигнала *RESET* регистр *TCCR2* содержит все нули, поэтому таймер остановлен, счета нет. Запись всех нулей в младшие три разряда *TCCR2* используется для остановки таймера.

Бит 7 регистра *TCCR2* является резервным.

Бит 6. Запись единицы в этот бит включает режим широтно-импульсной модуляции (ШИМ). Более подробно об этом режиме смотри полное описание таймера 2.

Биты 5,4,3 обслуживают режим работы с порогом. В таймере 2 предусмотрена возможность установки порога, при достижении которого вырабатывается прерывание *TIM2\_COMP*. Для установки значения этого порога используется регистр ввода/вывода *OCR2* (адрес \$23).

Биты 5 и 4 позволяют выбрать, какое событие произойдет на выходе *OC2* (вывод *PD7*), если содержимое *TCNT2* совпадет с порогом (с содержимым *OCR2*). В табл. 11.11 приведена связь между значениями этих разрядов и событиями на выходе *OC2*.

Таблица 11.11

**Связь событий на выходе OC2 со значениями разрядов регистра TCNT2**

COM21	COM20	Событие на OC2 при достижении порога
0	0	Таймер 2 отключен от вывода OC2
0	1	Состояние на выводе OC2 меняется на противоположное
1	0	На выводе OC2 устанавливается нуль
1	1	На выводе OC2 устанавливается единица

Для реализации этих возможностей  $PD7$  должен быть запрограммирован в качестве выхода.

Бит 3. Единица в этом разряде устанавливает специальный режим работы таймера. В этом режиме после достижении порога регистр  $TCNT2$  обнуляется и счет продолжается. В результате таймер 2 начинает работать как счетчик по модулю  $C$ , где  $C$  – значение порога, занесенное в регистр  $OCR2$ . Если при этом в регистре  $TCCR2$  разряд  $COM20$  равен единице, а разряд  $COM21$  равен нулю, то на выходе  $OC2$  появится симметричное прямоугольное колебание, период которого будет равен  $2C \times T$ . Здесь  $C$  – значение порога, занесенное в  $OCR2$ , а  $T$  – период сигнала, поступающего на счетный вход таймера 2 с выхода предварительного делителя.

При переполнении таймера 2 (переход в процессе счета от состояния *все единицы* к состоянию *все нули*) вырабатывается прерывание *TIM2\_OVF*.

Каждое из двух прерываний таймера 2 может быть индивидуально разрешено или запрещено с помощью разрядов регистра  $TIMSK$ . При возникновении события, вызывающего прерывание, в регистре флагов прерываний таймеров  $TIFR$  устанавливается в единицу соответствующий разряд. Как только прерывание начинает обрабатываться, соответствующий ему разряд в регистре  $TIFR$  сбрасывается.

### 11.4.7. Аналого-цифровой преобразователь

Аналого-цифровой преобразователь (АЦП), входящий в состав микроконтроллера, обеспечивает преобразование аналогового напряжения в 10-разрядный двоичный код методом поразрядного уравновешивания. Минимальное время преобразования составляет 65 микросекунд. Блок-схема преобразователя приведена на рис.11.8.



Рис. 11.8. Аналого-цифровой преобразователь

В состав преобразователя входит 8-канальный аналого-вый мультиплексор, позволяющий осуществить преобразование любого из восьми аналоговых напряжений, которые имеются на его входах.

В качестве источника опорного напряжения  $AREF$  можно использовать как напряжение питания  $AV_{CC}$ , так и внутренний либо внешний источник опорного напряжения, которое обязано находиться в диапазоне между 2В и  $AV_{CC}$ . Преобразуемое напряжение должно находиться в диапазоне от нуля до  $AREF$ .

АЦП может функционировать в двух режимах:

- режим однократного преобразования, когда запуск каждого преобразования инициируется пользователем;
- режим непрерывного преобразования, при котором запуск преобразования выполняется регулярно через определенные интервалы времени.

Аналого-цифровому преобразователю принадлежат четыре регистра, к которым можно обращаться с помощью команд *OUT* и *IN*. Это два регистра результата – *ADCH* (адрес \$05) и *ADCL* (адрес \$04), регистр управления мультиплексором *ADMUX* (адрес \$07) и регистр управления и контроля *ADCSR* (адрес \$06).

В регистре *ADCL* образуется младший байт результата преобразования, а в двух младших разрядах регистра *ADCH* – два старших бита результата. Эти регистры доступны только для чтения, причем первым следует читать содержимое регистра *ADCH*, а затем содержимое регистра *ADCL*.

Три младших разряда регистра управления мультиплексором *ADMUX* (*ADC Multiplexer Select*) позволяют выбрать входной сигнал мультиплексора, который будет преобразован в число. Вход *ADC0* выбирается при всех нулях в этих разрядах, а вход *ADC7* – при всех единицах.

Регистр *ADCSR* (*ADC Control and Status Register*) обеспечивает общее управление работой аналого-цифрового преобразователя.

В табл. 11.12 приведено назначение разрядов регистра *ADCSR*, управляющего работой аналого-цифрового преобразователя.

Т а б л и ц а 11.12  
Назначение разрядов регистра *ADCSR*

7	6	5	4	3	2	1	0	BIT
ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	Address \$06
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Read/Write
0	0	0	0	0	0	0	0	Initial value

Бит 7 – *ADEN* (*ADC Enable*). Единица в этом разряде означает, что АЦП используется и выводы порта *A*, запрограммии-

рованные как входы, становятся его входами. Остальные выводы порта *A* могут использоваться для выдачи цифровых сигналов.

Бит 6 – *ADSC (ADC Start Conversion)*. Запись единицы в этот разряд запускает преобразование (если *ADEN* = 1). Запись единицы в разряды 6 и 7 можно производить одновременно. После окончания преобразования бит 6 автоматически устанавливается в 0.

Бит 5 – *ADFR (ADC Free Running Select)*. Если в этом разряде стоит нуль, то для запуска очередного преобразования всякий раз необходимо записывать единицу в бит 6. Если же в этом разряде стоит единица, то достаточно запустить только первое преобразование, а далее запуск будет осуществляться автоматически после окончания предыдущего преобразования.

Бит 4 – *ADIF (ADC Interrupt Flag)*. В этом разряде автоматически устанавливается единица после окончания очередного преобразования, то есть когда результат преобразования готов и может быть прочитан из регистров данных *ADCH* и *ADCL*. Когда *ADIF* становится равным единице, запускается соответствующее прерывание (*ADC Conversion Complete Interrupt*) при условии, что имеется общее разрешение прерываний. Когда это прерывание начинает обрабатываться, то *ADIF* автоматически обнуляется. Если же потребуется самостоятельно обнулить этот разряд, то это можно сделать путем записи в него, как ни странно, единицы.

Биты 2,1,0 – *ADPS2, ADPS1, ADPS0 (ADC Prescaler Select Bits)*. Эти биты задают коэффициент деления предварительного делителя (прескалера), изображенного на рис. 11.9.

Импульсы с выхода прескалера поступают на тактовый вход АЦП. Каждой цикл преобразования выполняется за 13 тактов. Не рекомендуется устанавливать частоту выходного сигнала предварительного делителя выше 200 кГц, поскольку это приводит к снижению точности АЦП. При этом не гарантирована достоверность младших разрядов результата преобразования.

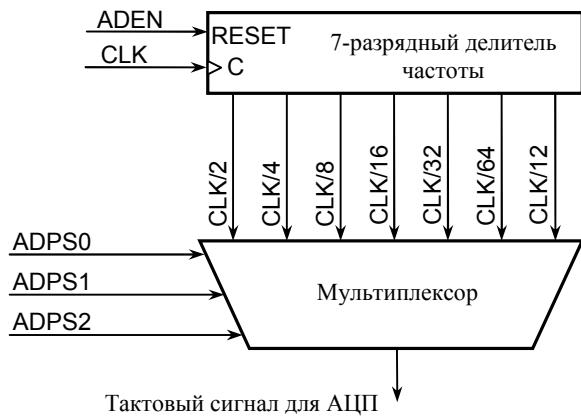


Рис. 11.9. Предварительный делитель (прескалер)

*CLK* – тактовый сигнал микроконтроллера.

Если *ADEN* = 0, то делитель частоты остановлен и сигналы тактовой частоты на АЦП не поступают.

В табл. 11.13 приведена связь между значениями *ADPS2*, *ADPS1*, *ADPS0* и коэффициентом деления предварительного делителя.

Таблица 11.13

Коэффициенты деления предварительного делителя

<i>ADPS2</i>	<i>ADPS1</i>	<i>ADPS0</i>	Коэффициент деления
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Переключения, связанные с изменением цифровых сигналов как внутри микроконтроллера, так и во внешних цепях вызывают значительные импульсные помехи для аналогоцифрового преобразования. Эти помехи приводят к снижению реальной точности преобразования. Для борьбы с этими помехами принимают следующие меры:

- Напряжение питания аналоговой части микроконтроллера подается от источника  $V_{CC}$  через  $LC$ -фильтр низких частот. Вывод  $AV_{CC}$  должен быть подключен к  $V_{CC}$  через индуктивность 10 мГн, и от этого вывода на  $AGND$  должен быть включен конденсатор емкостью 0.1 мкФ.
- Все аналоговые компоненты подключают к общей аналоговой земле  $AGND$ , аналоговая земля должна соединяться с цифровой землей  $GND$  только в одной точке.
- Высокоскоростные цифровые цепи располагают как можно дальше от аналоговых цепей.
- Цепи передачи аналоговых сигналов делают короткими.
- Если некоторые выводы порта  $A$  используются в качестве цифровых выходов, то не следует допускать их переключения во время аналого-цифрового преобразования.
- После запуска АЦП микроконтроллер переводится в режим пониженного энергопотребления (*Idle Mode*). При этом центральный процессор микроконтроллера перестает работать и уровень импульсных помех существенно снижается. Работа центрального процессора возобновится после окончания преобразования и появления соответствующего прерывания.

#### 11.4.8. Последовательный синхронный интерфейс

Последовательный синхронный интерфейс *SPI* (*The Serial Peripheral Interface*) позволяет обмениваться данными с высокой скоростью между контроллером ATmega8535 и различными внешними устройствами или между несколькими аналогичными контроллерами. Данный интерфейс часто используется для программирования микроконтроллера в режиме последовательного программирования. Дополнительной возможностью интерфейса

*SPI* является *пробуждение* микроконтроллера из *спящего* режима *Idle* при поступлении данных.

*SPI* характеризуется следующими параметрами:

- полнодуплексная (одновременно в двух направлениях) трехпроводная синхронная передача данных;
- предельная скорость передачи данных  $CK/4$  бит в секунду;
- передача осуществляется байтами;
- передачу ведется начиная либо со старшего бита, либо с младшего;
- по окончании передачи вырабатывается прерывание (адрес \$008);
- имеется флаг конфликтов при записи *WCOL* (*Write Collision Flag*).

Последовательный синхронный интерфейс обслуживается тремя регистрами:

*SPDR – Data Register* (адрес \$0F). В этот регистр можно писать и читать его содержимое.

*SPCR – Control Register* (адрес \$0D). В этот регистр можно писать и читать его содержимое.

*SPSR – Status Register* (адрес \$0E). Из этого регистра возможно только чтение.

Сигналом *RESET* данные регистры обнуляются.

На рис. 11.10 приведено назначение разрядов регистра *SPCR*.

	7	6	5	4	3	2	1	0	
Addr \$0D	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR

Рис.11.10. Назначение разрядов регистра SPCR

*SPIE – Interrupt Enable*. Разрешение прерывания после окончания передачи байта.

*SPE – SPI Enable.* Разрешение работы *SPI*. Если в этом разряде 0, то никакие функции *SPI* не будут реализованы.

*DORD – Data ORDer.* Порядок выдачи байта. Если *DORD* = 1, то сначала будет выдаваться младший бит, в противном случае первым выдается старший бит.

*MSTR – Master/Slave Select.* Если *MSTR* = 1, то *MASTER*, если *MSTR* = 0, то *SLAVE*.

*CPOL – Clock POlarity.* Такты выдаются на выход либо напрямую, либо в инвертированном виде. Если *CPOL* = 0, то при отсутствии тактов на выходе *SCK* низкий уровень. Если *CPOL* = 1, то при отсутствии тактов на выходе *SCK* высокий уровень.

*CPHA – Clock PHAse.* *CPHA* = 0 соответствует нормальному режиму появления тактовых сигналов. Если *CPHA* = 1, то тактовые сигналы появляются на выходе раньше на  $3/8 T$  ( $T$  – период сигнала на выходе *SCK*).

*SPR1* и *SPR0* определяют коэффициент деления тактовой частоты (*CLK*) перед выдачей на *SCK*.

Коэффициент деления тактовой частоты в соответствии с табл. 11.14 зависит не только от значений, хранящихся в разрядах *SPR1* и *SPR0* регистра *SPCR*, но и от содержимого разряда *SPI2X* регистра *SPSR*.

Таблица 11.14

**Задание коэффициента деления тактовой частоты**

SPI2X	SPR1	SPR0	SCK
0	0	0	CLK/4
0	0	1	CLK/16
0	1	0	CLK/64
0	1	1	CLK/128
1	0	0	CLK/2
1	0	1	CLK/8
1	1	0	CLK/32
1	1	1	CLK/64

Назначение разрядов регистра *SPSR* приведено на рис. 11.11.

	7	6	5	4	3	2	1	0	
Addr \$0E	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR

Рис.11.11. Назначение разрядов регистра SPSR

*SPIF – Interrupt Flag.* Устанавливается в единицу после завершения передачи. Если прерывание разрешено, то произойдет переход по адресу \$008 с запоминанием в стеке предыдущего значения программного счетчика.

Кроме того, *SPIF* устанавливается в единицу, если *SPI* является *MASTER*, вывод *SS* запрограммирован как вход, и на этом входе появляется нулевой потенциал.

Этот разряд сбрасывается автоматически при начале обработки прерывания. Он сбрасывается также, если сначала прочитать *SPSR*, а затем обратиться к *SPDR*.

*WCOL – Write Collision flag.* Этот флаг устанавливается в единицу, если в регистр *SPDR* была запись в то время, когда данные передавались. При этом запись будет произведена неправильно. Этот разряд обнуляется, если сначала прочитать *SPSR*, а затем обратиться к *SPDR*.

На рис. 11.12 приведена схема соединений для передачи данных между двумя микроконтроллерами.

Перед тем как начать использовать *SPI* для передачи данных, необходимо произвести соответствующие установки в регистре *SPCR*. Кроме того, в том микроконтроллере, *SPI* которого запрограммирован как *SLAVE*, следует сделать вывод *PB6 (MISO – Master Input, Slave Output)* выходом. Выводы *PB4 (SS)*, *PB5(MOSI – Master Output, Slave Input)*, *PB7(SCK)* автоматически станут входами, как только будет активизирована функция *SPI* (записана единица в разряд 6 регистра *SPCR*). На вход *PB4* следует подать лог. 0. Если на вход *PB4* подать лог. 1, то выводы *PB4*, *PB5*, *PB6*, *PB7* станут входами и *SPI* будет пассивен (не работает). Наличие вывода *SS* позволяет создавать системы, в которых один *MASTER* может поочередно обмениваться данными с несколькими *SLAVE*.

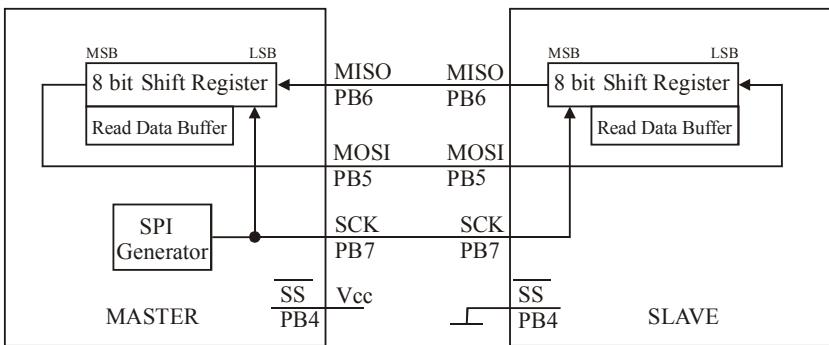


Рис. 11.12. Схема соединений для передачи данных между двумя микроконтроллерами

В микроконтроллере, выполняющем функции *MASTER*, выводы *PB5(MOSI)*, *PB7(SCK)* необходимо сделать выходами. Вывод *PB4* можно сделать входом, тогда на него следует подать лог. 1. При подаче на *PB4* лог. 0 данный *SPI* станет *SLAVE*. Если же *PB4* сделать выходом, то он перестает принадлежать *SPI* и может быть использован как обычный выход порта *B*. Вывод *PB6(MISO)* автоматически станет выходом, как только будет активизирована функция *SPI* (записана единица в разряд 6 регистра *SPCR*).

При записи байта в *SPDR* ведущего микроконтроллера этот байт поступает в сдвиговый регистр и *MASTER* начинает вырабатывать тактовые сигналы (8 тактов). Байт от ведущего контроллера передается в сдвиговый регистр ведомого. Одновременно из регистра сдвига ведомого байт (который был предварительно записан в регистр *SPDR* ведомого) переходит в регистр ведущего. После передачи байта такты перестают вырабатываться, а содержимое сдвигового регистра автоматически переписывается в регистр промежуточного хранения (*Read Data Buffer*). При чтении *SPDR* будет получено содержимое *Read Data Buffer*. Чтение следует произвести до окончания очередного цикла передачи байта.

После окончания передачи каждого байта в седьмом разряде *SPSR* ведущего и ведомого микроконтроллеров устанавливается

ся единица (*SPIF – Interrupt Flag*). Если прерывание *SPI* было разрешено, то оно произойдет, и управление передается по адресу \$008. При этом флаг прерывания автоматически сбрасывается.

Ниже приведен пример программы, содержащей те элементы, которые необходимы для обеспечения работы ведущего *SPI*:

EXAMPLE:

.EQU SPDR = \$0F ;Устанавливается соответствие между именами регистров и их адресами.

.EQU SPCR = \$0D

.EQU DDRB= \$17

.EQU SPH = \$3E

- - -

- - -

.CSEG

;Начало программы.

rjmp RESET

;Переход к процедуре начальной установки.

- - -

- - -

rjmp SPI

;Переход к программе обработки прерывания. Эта команда должна располагаться в ячейке памяти с адресом \$00A.

- - -

main:

;Начало главной программы.

- - -

- - -

RESET:

;Подпрограмма начальной установки.

ldi r16, \$A0

;PB7 и PB5 устанавливаются выходами.

out DDRB, r16

;

ldi r16, \$02

;

out SPH, r16

;Адрес стека становится \$200.

ldi r16, \$D0

;

out SPCR, r16

;Программируется режим работы SPI.

- - -

- - -

rjmp main

;Переход к главной программе.

<i>SPI:</i>		
<i>in</i>	<i>Rd, SPDR</i>	;Принятый байт передается в <i>Rd</i> .
<i>out</i>	<i>SPDR, Rr</i>	;Байт из <i>Rr</i> записывается в <i>SPDR</i> , запуская передачу.
- - -		;В этом блоке осуществляются операции над принятым байтом.
- - -		;Следующий байт, подлежащий передаче, записывается в <i>Rr</i> .
<i>reti</i>		; Возврат из прерывания.

Аналогично выглядит программа для ведомого *SPI*, только в подпрограмме *RESET* будут другие константы (вместо \$A0 будет \$40, вместо \$D0 будет \$C0).

В подпрограмме *SPI* первой помещается команда *OUT SPDR, Rr*. Это подготовит байт к передаче, которая будет инициализирована *MASTER*. После этого следует команда *IN Rd, SPDR* и так далее.

В ведущем микроконтроллере полезно предусмотреть выработку сигнала на выводе *PA1* непосредственно перед записью передаваемого байта в *SPDR*. Этот сигнал можно использовать для запуска развертки осциллографа при наблюдении сигналов на выводах *MISO*, *MOSI* и *SCK* в процессе передачи байта.

#### 11.4.9. Универсальный синхронно-асинхронный приемопередатчик

Во всех микроконтроллерах семейства *Mega* присутствует либо последовательный универсальный асинхронный приемопередатчик *UART* (*Universal Asynchronous serial Receiver and Transmitter*), либо универсальный синхронно-асинхронный приемопередатчик *USART* (*Universal Synchronous and serial Asynchronous Receiver and Transmitter*). Модули *USART* при работе в асинхронном режиме совместимы с модулями *UART* как по расположению разрядов управляющих регистров, так и по функционированию.

Отличительными особенностями работы *USART* являются:

- наличие программно-управляемого тактового генератора, специализированного только для обслуживания *USART*, обеспечивающего большой набор тактовых частот и возможность передачи данных на высоких частотах даже при низкой системной тактовой частоте;
- способность работы в дуплексном режиме (одновременная передача и прием данных);
- в модулях *USART* длина посылки может составлять от 5 до 9 разрядов;
- фильтрация помех на входе путем многократного опроса каждого бита;
- аппаратная фиксация ошибок переполнения и кадрирования (ложный стоп-бит) при приеме данных;
- формирование трех различных прерываний с индивидуальными адресами векторов прерывания: при завершении передачи (*TX Complete*), при завершении приема (*RX Complete*) и при освобождении регистра данных передатчика (*TX Data Register Empty*).

Формат передачи (рис. 11.13) предоставляет возможность после 8 бит данных передавать один служебный бит, который, например, может быть битом четности или дополнительным стоп-битом. В *AVR*-микроконтроллерах такой режим 9-битного обмена задается одновременно и для передачи, и для приема данных установкой бита  $\text{CHR9} = 1$  в регистре управления *UCR*.

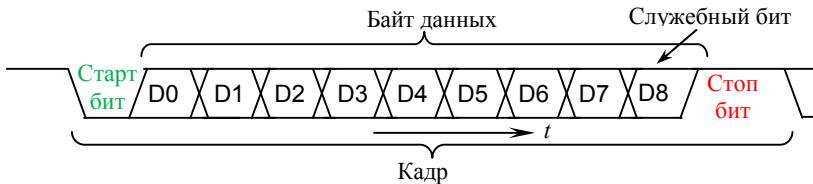


Рис. 11.13. Формат последовательной передачи данных

Упрощенная структурная схема модуля *USART / UART* приведена на рис. 11.14.

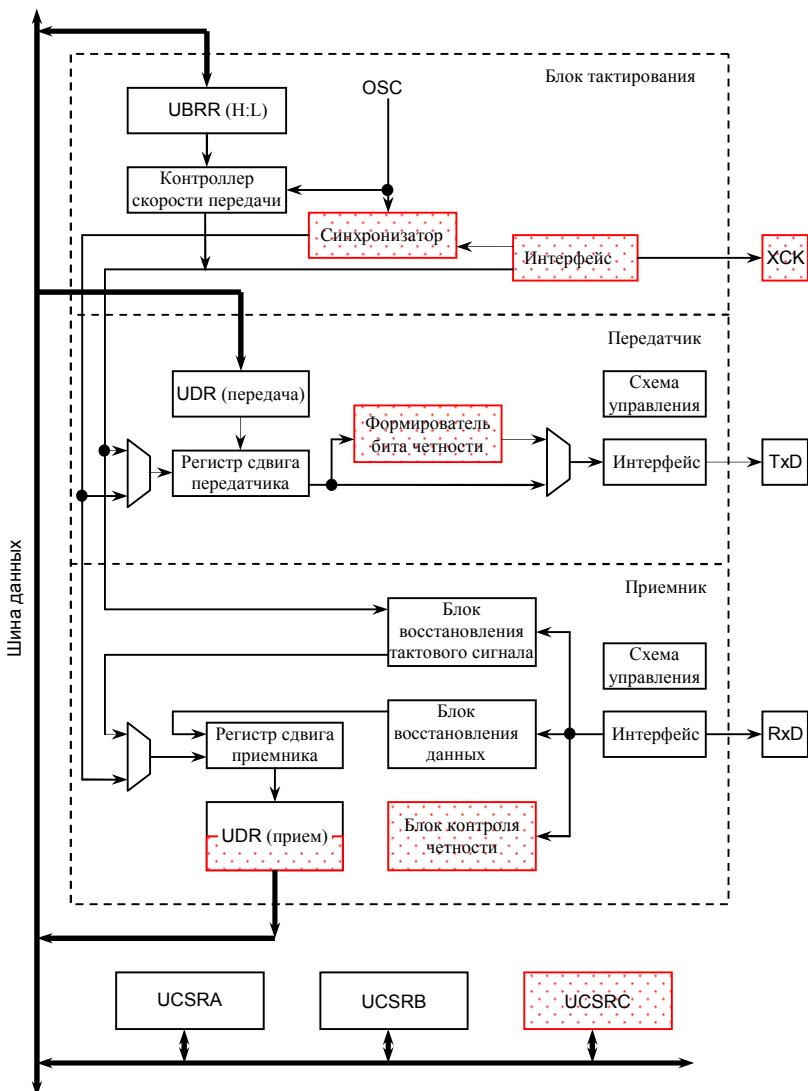


Рис. 11.14. Структурная схема модуля USART/UART

Элементы схемы, выделенные на рисунке красным цветом, имеются только в составе модулей *USART* и отсутствуют в модулях *UART*. Как показано на рис. 11.14, модуль состоит из трех основных частей: блока тактирования, блока передатчика и блока приемника. Блок тактирования модулей *USART* включает в себя схему синхронизации, которая используется при работе в синхронном режиме, и контроллер скорости передачи. В модулях *UART* блок тактирования содержит только контроллер скорости передачи.

Блок передачи включает одноуровневый буфер, регистр сдвига, схему формирования бита четности (только *USART*) и схему управления. Блок приемника включает схемы восстановления тактового сигнала и данных, схему контроля четности (только *USART*), двухуровневый (*USART*) или одноуровневый (*UART*) буфер, регистр сдвига и схему управления.

Для управления модулями *USART* используются три регистра: *UCSRA*, *UCSRB* и *UCSRC*. Значения разрядов этих регистров приведено в табл. 11.14, 11.15, 11.16 соответственно.

Т а б л и ц а 11.14  
Регистр статуса UCSRA

Разряд	Название бита	Выполняемые функции
0	MPCM	Режим мультипроцессорного обмена. Если этот разряд установлен в 1, то ведомые микроконтроллеры ожидают приема кадра, содержащего адрес.
1	U2X	Удвоение скорости обмена. Если этот разряд установлен в 1, то коэффициент деления прескальера контроллера скорости передачи уменьшается с 16 до 8, удваивая скорость асинхронного обмена. В синхронном режиме должен быть установлен в 0.
2	PE(Parity Error)	Флаг ошибки контроля четности. Устанавливается в 1, если в данных, находящихся в буфере приемника, обнаружена ошибка контроля четности. При отключении контроля четности в этом разряде постоянно 0.
3	OR (OverRun)	Флаг переполнения. Устанавливается в 1, когда символ, поступивший в регистр сдвига приемни-

		ка, был утерян из-за того, что к этому времени предыдущий байт не был считан из регистра данных приемника UDR. Сбрасывается во время перезаписи байта из регистра сдвига в регистр данных UDR, если между приемом этого байта и предшествующего не было потерь информации.
4	<b>FE</b> (Fram-ing Error)	Флаг ошибки кадрирования. Устанавливается в 1, когда стоп-бит входящего символа равен 0. Сбрасывается, когда стоп-бит полученных данных равен 1.
5	UDRE (UART Data Register Empty)	Регистр данных USART пуст. Устанавливается в 1, когда символ, записанный в UDR, послан в регистр сдвига передатчика, и передатчик готов получить новый символ от процессора. Если установлен бит UDRIE в регистре UCR, установка UDRE вызовет запрос прерывания <i>регистр данных пуст</i> . Сбрасывается UDRE записью регистра UDR; устанавливается после сброса, индицируя готовность передатчика.
6	TXC (UART Transmit Complete)	Флаг завершения передачи. Устанавливается в 1, когда передаваемый символ, включая стоп-бит, послан из регистра сдвига передатчика в последовательный канал и новый не был записан в регистр UDR. Если установлен бит TXCIE в регистре UCR, то при установке флага генерируется прерывание <i>передача завершена</i> . Сбрасывается при выполнении подпрограммы обработки прерывания.
7	RXC (UART Receive Complete)	Флаг завершения приема. Устанавливается в 1 при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Если установлен бит RXCIE, то при установке флага генерируется запрос на прерывание <i>прием завершен</i> . Сбрасывается флаг аппаратно после опустошения буфера.

Установка битов управления TXEN = 1 (*Transmitter Enable*) и RXEN = 1 (*Receiver Enable*) в регистре управления UCSRB разрешает соответственно передачу и прием данных по последовательному каналу. При этом выход передатчика и вход приемника соединяются с внешними выводами TXD и RXD микроконтролле-

ра и настраиваются соответственно на выход и на вход независимо от установки бит в регистре направления данных порта DDRx. При этом вход RXD, как и обычный цифровой вход AVR, может быть привязан к шине питания при помощи внутреннего резистора. Управление подключением/отключением этого резистора осуществляется обычным для выводов микроконтроллера способом – программированием соответствующих бит в регистре PORTx. При сбросе TXEN и RXEN выводы TXD и RXD могут использоваться как обычные цифровые входы/выходы.

Регистр данных приемника UDR и регистр данных передатчика UDR – это два физически разных регистра, имеющих один адрес в адресном пространстве регистров ввода/вывода. Процессор определяет, к какому регистру осуществлять обращение по типу выполняемой операции. При выводе (команда OUT UDR, Rx) данные записываются в регистр данных передатчика UDR, а при считывании (команда IN Rx, UDR) выборка осуществляется из регистра данных приемника.

Таблица 11.15

#### Регистр управления UCSRB

Разряд	Название бита	Выполняемые функции
0	TXB8 (Transmit Data Bit8)	Является 9-м битом передаваемого символа при 9-битном режиме передачи.
1	RXB8 (Receive Data Bit8)	Является 9-м битом принимаемого символа при 9-битном режиме передачи.
2	CHR9 (9 Bit Characters)	Если установлен в 1, то передаваемый и принимаемый символы имеют длину 9 бит плюс старт- и стоп-биты. 9-й бит читается и записывается с помощью битов RXB8 и TXB8 соответственно. 9-й бит данных может быть использован в качестве дополнительного стоп-бита или бита четности.
3	TXEN (Transmitter Enable)	Разрешение передачи UART, если установлен в 1. Запрет передатчика во время передачи произойдет после того, как символ в регистре сдвига и следующий символ в регистре UDR будут полностью посланы в последовательный канал.
4	RXEN (Receiver Enable)	Разрешение приема UART, если установлен в 1. Если прием запрещен, флаги статуса TXC, OR, FE не могут быть установлены. Если же они установлены, то их сброс не произойдет после запрета приема.

5	UDRIE (UDR Empty Int. Enable)	Установка этого бита и бита UDRE в 1 вызовет выполнение соответствующего прерывания, если установлен глобальный флаг разрешения прерываний.
6	TXCIE (TX Complete Int. Enable)	Установка этого бита и бита TXC в 1 вызовет выполнение соответствующего прерывания, если установлен глобальный флаг разрешения прерываний.
7	RXCIE (RX Complete Int. Enable)	Установка этого бита и бита RXC в 1 вызовет выполнение соответствующего прерывания, если установлен глобальный флаг разрешения прерываний.

Передача в последовательный канал инициируется записью байта в регистр данных передатчика UDR (рис. 11.14). Все остальные операции по преобразованию данных из параллельной формы в последовательную и формированию кадра передачи байта берет на себя передатчик *USART*. Он переписывает данные из регистра UDR в регистр сдвига передатчика, где к ним автоматически добавляются старт-бит, стоп-бит и, если установлен режим 9-битной передачи, бит TXB8 из регистра UCR.

Т а б л и ц а 11.16

#### Регистр управления UCSRC

Разряд	Название бита	Выполняемые функции
0	UCPOL(Clock Polarity)	Выбор полярности тактового сигнала. Используется только в синхронном режиме. Если установлен в 0, то выдача данных по спадающему фронту и прием данных по нарастающему. Если установлен в 1, то наоборот.
1	UCSZ0 (Character Size)	Формат посылок. Совместно с UCSZ1 определяет количество разрядов данных в кадре.
2	UCSZ1 (Character Size)	Формат посылок. Совместно с UCSZ0 определяет количество разрядов данных в кадре.
3	USBS (Siop Bit Select)	Количество стоп битов. Если установлен в "0", то передатчик посыпает один стоп-бит, если установлен в 1, то посыпается два стоп-бита.
4	UPM0 (Parity Mode)	Определяет режим работы схемы контроля четности.
5	UPM1 (Parity Mode)	Определяет режим работы схемы контроля четности.

6	UMSEL (USART Mode Select)	Режим работы USART. Если установлен в 1, то модуль USART работает в синхронном режиме.
7	URSEL (Register Select)	Выбор регистра. Если установлен в 1, то обращение производится к регистру UCSRC. Если установлен в 0, то обращение производится к регистру UBRRH.

Если в момент записи регистра сдвига UDR уже освободился от передачи предыдущего байта, то данные переписываются из UDR в регистр сдвига немедленно. В противном случае перепись осуществляется лишь после того, как стоп-бит передаваемого в настоящий момент байта появится на выводе TXD.

Когда данные посылаются в регистр сдвига из UDR, устанавливается бит UDRE (*UART Data Register Empty*) в регистре статуса USR (*UART Status Register*), что означает, что USART готов получить от процессора новый байт. Когда процессор записывает очередной байт в регистр данных UDR, флаг UDRE автоматически сбрасывается. Если же новые данные не будут записаны в UDR к тому времени, как закончится передача текущего байта из регистра сдвига, то дополнительно к установленному флагу UDRE устанавливается флаг окончания передачи TXC (*TX Complete Flag*) в регистре USR.

Приемник USART выполняет операции по преобразованию входных данных из последовательной формы в параллельную. Прием данных начинается сразу же после обнаружения приемником корректного старт-бита. Каждый разряд содержимого кадра затем считывается с частотой, определяемой установками контроллера скорости передачи или тактовым сигналом ХСК. Считанные разряды данных последовательно помещаются в сдвиговый регистр приемника до обнаружения первого стоп-бита кадра. После этого содержимое сдвигового регистра пересыпается в буфер приемника, из которого принятное значение может быть получено путем чтения регистра данных модуля. При использовании 9-разрядных слов данных значение старшего разряда может быть определено по состоянию флага RX8 регистра UCSRB. Причем в модулях USART содержимое старшего разряда данных

должно быть считано до обращения к регистру данных. Это связано с тем, что флаг RX8 отображает значение старшего разряда слова данных кадра, находящегося на верхнем уровне буфера приемника, состояние которого при чтении регистра данных изменится.

Если во время приема кадра была включена схема контроля четности, то она вычисляет бит четности для всех разрядов принятого слова данных и сравнивает его с принятым битом четности. Результат проверки запоминается в буфере приемника вместе с принятым словом данных и стоп-битами. Наличие или отсутствие ошибки контроля четности может быть затем определено по состоянию флага UPE. Этот флаг устанавливается в 1, если следующее слово, которое может быть прочитано из буфера, имеет ошибку контроля четности. При выключенном контроле четности флаг UPE всегда читается как 0.

Блок приемника модулей *USART/UART* имеет еще два флага, показывающих состояние обмена: флаг ошибки кадрирования FE и флаг переполнения DOR/OR. Флаг FE устанавливается в 1, если значение первого стоп-бита принятого кадра не соответствует требуемому, т. е. равно 0.

Флаги DOR в *USART* и OR в *UART* индицируют потерю данных из-за переполнения буфера приемника. В *UART* флаг устанавливается в 1, если к моменту окончания приема кадра (заполнения сдвигового регистра приемника) данные предыдущего кадра не были считаны из регистра данных. В *USART* флаг устанавливается в 1 в случае приема старт-бита нового кадра при заполненных буфере и сдвиговом регистре приемника. Установленный флаг DOR/OR означает, что между прошлым байтом, считанным из регистра UDR, и байтом, считанным в данный момент, произошла потеря одного или нескольких кадров. Выключение приемника осуществляется сбросом разряда RXEN регистра UCSRB.

Для обеспечения приема в асинхронном режиме используются схемы восстановления тактового сигнала и данных. Схема восстановления тактового сигнала предназначена для синхронизации внутреннего тактового сигнала, формируемого контролле-

ром скорости передачи, и кадров, поступающих на вход RXD микроконтроллера. Схема восстановления производит предварительную выборку сигнала на входе RXD с частотой, в 16 раз большей скорости обмена (рис. 11.15), или в 8 раз выше при ускоренном режиме (U2X = 1).

Когда линия последовательной передачи свободна (данные не передаются), она находится в состоянии логической 1. Обнаружение приемником логического 0 после того, как линия была свободна, интерпретируется как начало старт-бита и инициируется последовательность определения старт-бита. Приемник, считая от первой нулевой выборки, анализирует 8, 9 и 10 выборки, а в ускоренном режиме 4, 5 и 6 выборки. При обнаружении двух или более логических 1 в этих трех выборках старт-бит воспринимается как импульсная помеха, и приемник начинает следить за следующим переходом входного сигнала с высокого уровня на низкий уровень.

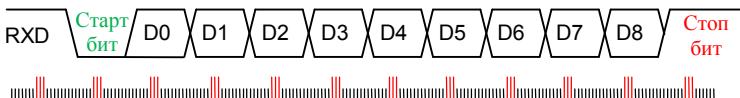


Рис. 11.15. Выборки входного сигнала

Если фиксируется старт-бит, то затем осуществляется распознавание бит данных, причем для каждого бита также анализируются три значения входного сигнала в выборках 8, 9 и 10, считая от начала бита. За истинное значение бита принимается логическое значение, которое появилось хотя бы в двух выборках. Такие значения, полученные в результате голосования, последовательно посылаются в регистр сдвига приемника.

При поступлении стоп-бита результатом голосования трех выборок должна быть логическая 1. Если обнаружены хотя бы два логических 0, устанавливается флаг ошибки кадрирования FE (*Framing Error*) в регистреUSR. Независимо от того, выявлен или нет стоп-бит в конце цикла приема байта, данные посылаются в регистр данных приемника UDR (рис. 11.14), и устанавливается флаг завершения приема RXC (*Receive Complete*) в регистреUSR.

Остальные процедуры в асинхронном режиме совпадают с процедурами в синхронном режиме.

Для тактирования приемопередатчика *UART* микроконтроллеры *AVR* содержат специализированный управляемый тактовый генератор, который является делителем системной тактовой частоты и задает скорость обмена в соответствии со следующими уравнениями:

- асинхронный режим (обычный,  $U2X = 0$ )  
 $BAUD = f_{ck} / (16(UBRR+1));$
- асинхронный режим (ускоренный,  $U2X = 1$ )  
 $BAUD = f_{ck} / (8(UBRR+1));$
- синхронный режим ведущего  
 $BAUD = f_{ck} / (2(UBRR+1)),$

где:  $BAUD$  – скорость обмена, бод;  $f_{ck}$  – частота тактового генератора микроконтроллера, МГц;  $UBRR$  – содержимое 8-битного доступного для записи регистра **UBRR** (0...255).

Таким образом, для задания скорости передачи или приема необходимо записать в регистр **UBRR** определенное числовое значение. В табл. 11.17 приведены эти значения для получения типовых скоростей обмена при разных системных тактовых частотах.

Таблица 11.17

**Программирование регистра UBRR для различных системных тактовых частот**

Скорость [бод]	1 МГц		1.8432 МГц		2 МГц	
	UBRR	Ошибка, %	UBRR	Ошибка, %	UBRR	Ошибка, %
2400	25	0.2	47	0	51	0.2
4800	12	0.2	23	0	25	0.2
9600	6	7.5	11	0	12	0.2
14400	3	7.8	7	0	8	-3.5
19200	2	7.8	5	0	6	-7.0
28800	1	7.8	3	0	3	8.5
38400	1	22.9	2	0	2	8.5
57600	0	7.8	1	0	1	8.5
115200	-	-	0	0	0	8.5

Рекомендуется использовать такие значения регистра UBRR, при которых получаемая скорость передачи отличается от требуемого значения меньше чем на 0.5%. Значения, дающие большее отклонение также можно использовать, однако следует иметь в виду, что при этом снижается помехозащищенность линии передачи.

При работе в синхронном режиме в качестве ведомого скорость приема и передачи определяется частотой сигнала, поступающего на вывод XCK. Частота этого сигнала должна удовлетворять выражению  $f_{XCK} < f_{osc}/4$ . Это ограничение связано с тем, что сигнал с вывода XCK сначала синхронизируется с тактовой частотой микроконтроллера, а затем проходит через детектор фронтов. Задержка сигнала при прохождении этих узлов равна двум машинным циклам.

## 11.5. Система команд микроконтроллера ATmega8535

Система команд микроконтроллера ATmega8535 содержит 130 команд, которые можно разделить на классы:

77 команд используют содержимое регистров общего назначения;

31 команда позволяет изменять содержимое программного счетчика;

26 команд оперируют содержимым памяти чисел;

2 команды (*IN* и *OUT*) обращения к регистрам ввода/вывода. В микроконтроллере 8535 имеется 64 таких регистра и с их помощью обеспечивается управление работой всех дополнительных функциональных блоков. Следует различать регистры ввода/вывода и порты ввода/вывода. В частности, каждыйпорт ввода/вывода обслуживается тремя регистрами ввода/вывода;

2 команды позволяют изменять отдельные биты регистров ввода/вывода;

19 команд изменяют отдельные биты регистра флагов (*SREG*);

1 команда пустая команда (*NOP*);

1 команда переводит контроллер в режим пониженного энергопотребления (*SLEEP*);

1 команда сбрасывает сторожевой таймер (*Watchdog Reset*);

4 команды выполняют операции с содержимым памяти программ;

1 команда используется для отладки программ (*BREAK*).

Если сложить все перечисленные команды, то получим число больше 130. Некоторые команды принадлежат одновременно двум из перечисленных классов.

Например, команда *PUSH r1* запишет байт из регистра общего назначения *r1* в стек (память чисел). То есть эта команда использует в своей работе содержимое регистра общего назначения и одновременно имеет дело с содержимым памяти чисел.

Ниже приведено краткое описание всех команд микроконтроллера.

Все команды можно разбить на 6 групп:

Команды передачи байтов (35 команд).

Команды работы с отдельными битами (28 команд).

Арифметические и логические команды (28 команды).

Команды передачи управления (31 команда).

Команды сравнения (3 команды).

Специальные команды (4 команды).

К **командам передачи байтов** относится команда *MOV Rd, Rr* (Move Between Registers), переписывающая байт из регистра *Rr* (источник) в регистр *Rd* (приемник). К этому же классу относятся команды *IN, OUT, PUSH, POP*. Все команды этой группы могут оперировать содержимым любого из 32 регистров общего назначения. Имеется только одно исключение. Команда *LDI Rd, K* (непосредственная загрузка константы  $1 \leq K \leq 255$  в регистр *Rd*) может работать только с регистрами, начиная с *R16* и старше ( $d \geq 16$ ). Ни одна команда передачи байтов не изменяет значение разрядов регистра флагов (*SREG*).

Команды работы с отдельными битами позволяют присваивать значение либо 1, либо 0 отдельным битам, расположенным в регистрах микроконтроллера. Так, команда *SBI P, b* (Set Bit in I/O Register) устанавливает 1 в разряде *b* ( $0 \leq b \leq 7$ ) регистра ввода/вывода с номером *P* ( $0 \leq P \leq 31$ ). Команда *CBI P, b* (Clear Bit in I/O Register) устанавливает 0 в разряде *b* ( $0 \leq b \leq 7$ ) регистра

ввода/вывода с номером  $P$  ( $0 \leq P \leq 31$ ). Эти команды могут работать не со всеми регистрами ввода/вывода, а только с 0-го по 31-й включительно.

Двадцать команд этой группы работают с разрядами регистра флагов (*SREG*). Так, команда *SEI* (*Global Interrupt Enable*) устанавливает в 1 флаг глобального разрешения прерываний (разряд 7 регистра *SREG*). Команда *CLI* (*Global Interrupt Disable*) устанавливает этот разряд регистра *SREG* в 0.

Большинство команд этой группы изменяет содержимое тех или иных разрядов регистра флагов.

**Арифметические и логические команды** выполняют арифметические и логические действия над содержимым регистров общего назначения. Большинство команд этой группы может работать с любым из 32 регистров общего назначения, однако есть 6 команд, которые могут работать только с 16 старшими регистрами. Примером может служить команда *ORI Rd, K* (*Logical OR Register and Constant*), осуществляющая поразрядное логическое умножение содержимого регистра *Rd* ( $d \geq 16$ ) и константы *K* ( $0 \leq K \leq 255$ ).

Все команды этой группы изменяют значения тех или иных разрядов регистра флагов (*SREG*). Исключение составляет команда *SER Rd* (*Set Register*), записывающая все единицы в регистр *Rd* ( $d \geq 16$ ).

**Команды передачи управления** (*Branch Instructions*) занимают важное место в системе команд микроконтроллера. Они позволяют создавать ветвящиеся программы, а это существенно облегчает сам процесс программирования. Часть команд этого класса уже рассматривалась выше (*RJMP*, *RET*, *RETI*). Эти команды обязательно изменяют содержимое программного счетчика, то есть относятся к командам безусловной передачи управления. Большинство же команд изменяют содержимое программного счетчика только при выполнении некоторого условия, поэтому их называют командами условной передачи управления.

Примером команды условной передачи управления может служить команда *CPSE Rd, Rr* (*Compare, Skip if Equal*), которая сравнивает содержимое регистров *Rd* и *Rr*. Если содержимое ре-

гистров совпадает, то содержимое программного счетчика увеличивается на 2. Тем самым обеспечивается перескок (*Skip*) через одну команду. Если же содержимое регистров не совпало, то содержимое программного счетчика увеличивается на 1, то есть сохраняется естественный порядок выполнения команд. Эта команда работает со всеми регистрами общего назначения и не изменяет разряды регистра флагов.

Имеются 4 команды типа *Skip*, которые работают аналогичным образом, однако условие, при котором происходит скачок, у каждой команды свое. Так, команда *SBIS P, b* ( $P \leq \$1F$ ,  $0 \leq b \leq 7$ ) осуществляет скачок, если в разряде  $b$  регистра ввода/вывода  $P$  стоит единица. Эта команда работает с младшей половиной регистров ввода/вывода и не изменяет значение разрядов регистра признаков.

Остальные команды условной передачи управления используют в своей работе значения разрядов регистра флагов. Примером может служить команда *BREQ k*, изменяющая содержимое программного счетчика, если в разряде  $Z$  регистра флагов стоит единица [*if (Z = 1), then PC = PC + k + 1*]. Эта команда может обеспечивать передачу управления только на малое расстояние ( $-64 \leq k \leq 63$ ). Она не изменяет значение разрядов регистра флагов.

Все команды передачи управления не изменяют значение разрядов регистра флагов. Имеется только одно исключение. Это команда *RETI (Interrupt Return)* – возврат из прерывания. Эта команда устанавливает единицу разряд 7 регистра флагов, разрешая тем самым прерывания.

**Команды сравнения** – это команды *CP (Compare)*, *CPC (Compare with Carry)* и *CPI (Compare Register with Immediate)*. Команда *CP Rd, Rr* осуществляет сравнение содержимого регистров  $Rd$  и  $Rr$ , вычисляя  $(Rd - Rr)$ . По результатам вычисления могут изменяться значения разрядов  $Z$ ,  $N$ ,  $V$ ,  $C$ ,  $H$  регистра флагов. Содержимое регистров  $Rd$  и  $Rr$  не изменяется.

Команда *CPC Rd, Rr* осуществляет сравнение содержимого регистров  $Rd$  и  $Rr$  с учетом значения разряда  $C$  регистра флагов, вычисляя  $(Rd - Rr - C)$ . По результатам вычисления могут изме-

няться значения разрядов  $Z, N, V, C, H$  регистра флагов. Содержимое регистров  $Rd$  и  $Rr$  не изменяется.

Команда  $CPI\ Rd, K$  осуществляет сравнение содержимого регистра  $Rd$  и константы  $K$  ( $0 \leq K \leq 255$ ), вычисляя  $(Rd - K)$ . По результатам вычисления могут изменяться значения разрядов  $Z, N, V, C, H$  регистра флагов. Содержимое регистра  $Rd$  не изменяется.

**Специальные команды** – это команды  $NOP$ ,  $SLEEP$  и  $WDT$ . Эти команды нельзя отнести к какому-либо из предыдущих классов. Так, команда  $NOP$  ничего не делает. Результатом ее выполнения будет только увеличение на единицу содержимого программного счетчика. Команда  $SLEEP$  позволяет перевести микроконтроллер в режим пониженного энергопотребления. Однако для ее выполнения необходимо произвести установки в некоторых регистрах ввода/вывода.

Команда  $WDT$  осуществляет сброс сторожевого таймера и выполняется только после некоторых предварительных установок в регистрах ввода-вывода.

## **СПИСОК ЛИТЕРАТУРЫ**

1. *Воронов Е.В., Ларин А.Л.* Элементы цифровой электроники: Учеб. пособие. – М.: МФТИ, 1992.
2. *Донов Г.И.* Применение микроконтроллеров. Учеб. пособие. – М.: МФТИ, 2007.
3. *Евстифеев А.В.* Микроконтроллеры AVR семейств Тіну и Mega фирмы ATMEL. – М.: Додэка-XXI, 2006.
4. *Джонс М.Х.* Электроника – Практический курс. – М.: Постмаркет, 1999.
5. *Предко М.* Руководство по микроконтроллерам. В 2-х т. – М.: Постмаркет, 2001.
6. *Титце У., Шенк К.* Полупроводниковая схемотехника. В 2-х т. – М.: Додека, 2008.
7. *Уэйкерли Дж.* Проектирование цифровых устройств. В 2-х т. – М.: Постмаркет, 2002.
8. *Хоровиц П., Хилл У.* Искусство схемотехники.– М.: Мир, 2003.

Учебное издание

*ЛАРИН Анатолий Леонидович*

## **ОСНОВЫ ЦИФРОВОЙ ЭЛЕКТРОНИКИ**

Редактор *И.А. Волкова*. Корректор *О.П. Котова*

Подписано в печать 29.03.2008. Формат 60 × 84 1/16. Бумага офсетная.  
Печать офсетная. Усл. печ. л. 19,0. Уч.- изд. л.18,25. Тираж 900 экз. Заказ №988

Государственное образовательное учреждение  
высшего профессионального образования

Московский физико-технический институт (государственный университет)  
Отдел автоматизированных издательских систем “ФИЗТЕХ-ПОЛИГРАФ”  
141700, Московская обл., г. Долгопрудный, Институтский пер., 9