

**А. В. КИСТРИН, М. Б. НИКИФОРОВ**

# **ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ**

**УЧЕБНИК**

5-е издание, переработанное и дополненное

Рекомендовано Федеральным государственным автономным учреждением  
«Федеральный институт развития образования» в качестве учебника  
для использования в учебном процессе образовательных учреждений,  
реализующих программы среднего профессионального образования по  
специальности

Регистрационный номер рецензии



Москва  
Издательский центр «Академия»  
2015

УДК  
ББК  
К

Уважаемый читатель!

Рецензенты:

зав. кафедрой «Вычислительная техника»  
Владимирского государственного университета им. Я. Г. и Н. Г. Столетовых,  
д-р тех. наук, проф. *В. Г. Ланцов*;  
канд. тех. наук, проф. кафедры «Информатика и вычислительная техника»  
Рязанского государственного университета им. С. Я. Есенина  
*В. А. Григорьев*

К Проектирование цифровых устройств : учебник для студ. учреждений сред. проф. образования / А. В. Кистрин, М. Б. Никифоров. — М. : Издательский центр «Академия», 2015. — 000 с.  
ISBN 978-5-4468-0000-0

Изложены основные принципы построения, функционирования и проектирования цифровых устройств ЭВМ. Представлены элементы и устройства комбинированного типа и с памятью, заполняющие устройства и микропроцессоры. Основное внимание уделено изучению программных средств описания синтеза и анализа электронных средств. Включены разделы, связанные с автоматизацией конструирования и производства, а также с оформлением технической документации.

Для студентов учреждений среднего профессионального образования.

УДК  
ББК

*Оригинал-макет данного издания является собственностью  
Издательского центра «Академия», и его воспроизведение любым способом  
без согласия правообладателя запрещается*

© Коллектив авторов, 2015  
© Образовательно-издательский центр «Академия», 2015  
© Оформление. Издательский центр «Академия», 2015  
ISBN 978-5-4468-0000-0

## Список основных обозначений и сокращений

**АЛУ** — арифметико-логическое устройство.  
**АОИ** — автоматическая оптическая инспекция.  
**БИС** — большая интегральная схема.  
**ГОСТ** — Государственный стандарт.  
**ДТЛ** — диодно-транзисторная логика.  
**ЕСКД** — единая система конструкторской документации.  
**ЕСПД** — единая система программной документации.  
**ЖЦ** — жизненный цикл.  
**ЗУ** — запоминающее устройство.  
**ИИ** — ионизирующее излучение.  
**ИК** — инфракрасный.  
**ИС** — интегральная схема.  
**КМОП** — комплементарная структура металл—оксид—полупроводник.  
**МДНФ** — минимальная дизъюнктивная нормальная форма.  
**МИС** — малая интегральная схема.  
**МКНФ** — минимальная конъюнктивная нормальная форма.  
**МПП** — многослойная печатная плата.  
**ОЗУ** — оперативное запоминающее устройство.  
**ПЗУ** — постоянное запоминающее устройство.  
**ПЛИС** — программируемая логическая интегральная схема  
**ПП** — печатная плата.  
**ППЗУ** — программируемое ПЗУ.  
**РКТУ** — рентгеновские контрольные технологические установки.  
**РОН** — регистр общего назначения.  
**РПЗУ** — репрограммируемые ПЗУ с ультрафиолетовым стиранием.  
**РПЗУ-ЭС** — репрограммируемое ПЗУ с электрическим стиранием.  
**РЭА** — радиоэлектронная аппаратура.  
**САПР** — система автоматизированного проектирования.  
**СБИС** — сверхбольшая интегральная схема.  
**СДНФ** — совершенная дизъюнктивная нормальная форма.  
**СИС** — средняя интегральная схема.  
**СКНФ** — совершенная конъюнктивная нормальная форма.  
**ТЗ** — техническое задание.  
**ТМП** — технология монтажа на поверхность.  
**ТТЛ** — транзисторно-транзисторная логика.  
**ТТЛШ** — транзисторно-транзисторная логика с диодами Шоттки.  
**ТУ** — технические условия.

**ЧПУ** — числовое программное управление.  
**ЭВМ** — электронная вычислительная машина.  
**AHDL** (Altera Hardware Description Language) — язык описания аппаратных средств фирмы Altera.  
**BGA** (Ball Grid Array) — массив шариков.  
**CAD** (Computer Automation Design) — система автоматизированного проектирования.  
**CISC** (Complex Instruction Set Computer) — компьютер с полным набором программ.  
**CPLD** (Complex Programmable Logic Device) — сложное программируемое логическое устройство.  
**CPU** (Central Processing Unit) — центральный процессор.  
**CSP** (Chip Scale Package) — печатная плата с размерами кристалла.  
**DRAM** (Dynamic Random Access Memory) — динамическая память с произвольным доступом.  
**DSP** (Digital Signal Processor) — цифровой сигнальный процессор.  
**EEPROM** (Electrically Erasable Programmable Read-Only Memory) — электрически стираемое перепрограммируемое постоянное запоминающее устройство.  
**FPGA** (Field Programmable Gate Array) — программируемая матрица логических элементов.  
**GA** (Gate Array) — вентильная матрица.  
**PLA** (Programmable Logic Array) — программируемая логическая матрица.  
**SMD** (Surface Mount Device) — планарно монтируемый компонент.  
**RISC** (Reduced Instruction Set Computer) — компьютер с сокращенным набором команд.  
**SRAM** (Static Random Access Memory) — статическая оперативная память с произвольным доступом.  
**VHDL** (Very High Speed Integrated Circuits Hardware Description Language) — язык описания быстродействующих аппаратных средств.

Цифровые устройства обработки, передачи и хранения данных применяются во всех сферах деятельности человека. Для построения данных устройств используют интегральные схемы различной степени интеграции. В настоящее время элементной базой для цифровых устройств являются программируемые логические интегральные схемы (ПЛИС), обеспечивающие широкие функциональные возможности. В этом случае сокращаются сроки и стоимость разработки, обеспечиваются высокая надежность и компактность разрабатываемых систем.

Проектирование устройств на программируемых логических интегральных схемах выполняется с использованием систем автоматизированного проектирования (САПР), которые позволяют выполнить ввод проекта, компиляцию, моделирование, программирование ПЛИС. Запись полученного при проектировании файла конфигурации в ПЛИС позволяет получить аппаратную реализацию разработанного устройства.

Ввод исходных данных проекта в виде описаний с использованием специальных языков, что предусмотрено в современных САПР, является мощным универсальным и удобным в использовании современным средством проектирования, которое обеспечивает значительное сокращение трудоемкости и сроков разработки больших и сложных схем.

Процесс проектирования цифровых устройств на основе ПЛИС содержит несколько этапов: изучение функционирования устройства и принципов его проектирования, выполнение анализа и синтеза, составление описания на специальном языке для ввода в САПР, разработка методики тестирования.

В процессе изучения материала рекомендуется выполнять моделирование изучаемых устройств, основанное на использовании САПР *MAX + PLUS II* фирмы *Altera*, позволяющее изучить функционирование в различных режимах, определить параметры, а также освоить использование САПР для проектирования устройств на основе ПЛИС.

Предлагаемый учебник ориентирован в первую очередь на студентов средних специальных учебных заведений, обучающихся по специальности 230113 — Компьютерные системы и комплексы.

Учебник может быть полезен и студентам вузов, обучающимся по схожим специальностям, а также специалистам, работающим на предприятиях соответствующего профиля.

При написании учебника широко использован опыт ОАО «Рязанский государственный приборный завод» и Рязанского государственного радиотехнического университета. Авторы искреннее благодарны коллективам этих организаций.

## АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВОЙ ТЕХНИКИ

### 1.1. СИСТЕМЫ СЧИСЛЕНИЯ

Основой проектирования цифровых устройств являются базовые положения, связанные с выбором способов представления данных выполнения основных арифметических и логических операций.

**Системой счисления** называется совокупность символов и правил записи чисел. В цифровых устройствах используют позиционные системы счисления, в которых значение определенного символа зависит от его позиции или места расположения в последовательности символов, представляющих число.

Основание системы счисления  $p$  определяет количество символов (цифр), используемых для записи любого числа.

Число записывается в виде последовательности цифр:  $a_{n-1}, a_{n-2}, \dots, a_1, a_0$ . Отдельные позиции в записи числа называются *разрядами*. В записи числа старший разряд расположен слева, а младший — справа. Номер разряда, или индекс, для младшего разряда равен нулю. Каждому разряду соответствует вес. В любой системе счисления вес младшего разряда равен 1, вес следующего разряда равен основанию системы счисления. Если число имеет  $n$  разрядов, то номер старшего разряда равен  $n - 1$ . Всего можно записать  $p^n$  чисел от  $p^n - 1$  до 0.

Значение числа в позиционных системах определяется суммированием цифр с весовыми коэффициентами, которые являются степенью основания системы счисления  $p$  и зависят от номера разряда:

$$X = a_{n-1} \cdot p^{n-1} + a_{n-2} \cdot p^{n-2} + \dots + a_1 \cdot p^1 + a_0 \cdot p^0 = \sum_{i=0}^{n-1} a_i \cdot p^i.$$

Основание записывается как и в любой системе счисления  $p = 10$ : в двоичной системе  $2_{10} = 10_2$ , в шестнадцатеричной  $16_{10} = 10_{16}$ .

Нижний индекс в записи числа указывает основание системы счисления. Если индекс отсутствует, то по умолчанию система счисления десятичная.

Во всех системах счисления используется **общее правило для процесса счета**: как только значение любого из разрядов доходит до своего максимума, значение старшего по отношению к нему разряда увеличивается на единицу, а значение текущего разряда устанавливается равным нулю.

**Десятичная система счисления.** Десятичная система счисления используется в цифровых устройствах при вводе исходных данных и для отображения результатов.

В десятичной системе для записи любого числа используется 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Вес разрядов справа налево, как известно, равен: 1, 10, 100, 1000 и т.д. Это степени основания  $p = 10$ . Число, имеющее  $n$  разрядов, позволяет записать  $10^n$  чисел от 0 до  $10^n - 1$ .

Рассмотрим запись числа  $X = 2014$  в десятичной системе. Количество цифр в данном числе  $n = 4$ . Номер старшего разряда будет равен 3. Представим заданное число в виде суммы в соответствии с формулой определения значения числа (табл. 1.1):

$$\begin{aligned} X &= a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 = \\ &= 2 \cdot 1000 + 0 \cdot 100 + 1 \cdot 10 + 4 \cdot 1 = 2014 \end{aligned}$$

**Двоичная система счисления.** В цифровой технике двоичные коды используются для представления любых обрабатываемых данных: чисел, символов, команд. Цифровые устройства обычно работают в двоичной системе счисления. Это связано с удобством представления электрическими сигналами цифр 0 и 1, которые используются в двоичной системе для записи любого числа.

В цифровых устройствах используются двоичные коды различной разрядности. Код, имеющий всего один двоичный разряд, называют *бит* (от англ. *binary digit* — двоичная цифра). Бит может быть равен 0 или 1.

Таблица 1.1

Цифры числа	2	0	1	4
Номер разряда	3	2	1	0
Вес разряда	$10^3$	$10^2$	$10^1$	$10^0$
Слагаемые числа	$2 \cdot 1000$	$0 \cdot 100$	$1 \cdot 10$	$4 \cdot 1$

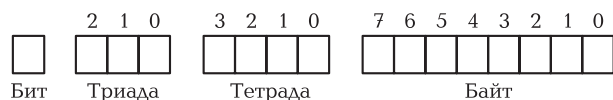


Рис. 1.1. Двоичные коды различной разрядности

Двоичный код, содержащий несколько разрядов, называется *слово*, а группа бит в слове — *поле*.

Двоичный код, содержащий три разряда, называется *триада*, четыре разряда — *тетрада*, а восемь разрядов — *байт*. Байт содержит два поля по четыре разряда, или две тетрады (рис. 1.1).

Число в двоичной системе счисления записывается в виде суммы произведений, состоящих из цифр (0 или 1), умноженных на весовые коэффициенты, равные степеням числа 2. Веса разрядов справа налево в двоичной системе — это степени числа два: 1 — 2 — 4 — 8 — 16 — 32 — 64 — 128 — 256 и т.д.

Вес младшего разряда равен 1, а вес следующего основанию системы — 2. Основание системы — число 2 — записывается в двоичной системе как единица и ноль.

С помощью  $n$ -разрядного двоичного кода можно представить  $2^n$  последовательных чисел от 0 до  $2^n - 1$ . В цифровых устройствах нередко используют коды, содержащие четыре или восемь разрядов. С помощью 4-разрядного двоичного кода можно представить  $2^4 = 16$  чисел в диапазоне от 0 до 15, а с помощью 8-разрядного кода можно представить  $2^8 = 256$  чисел в диапазоне от 0 до 255.

**Преобразование числа из двоичной системы счисления в десятичную.** Выполняется сложением двоичных весов всех ненулевых разрядов числа в соответствии с выражением

$$X = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0.$$

Для преобразования 4-разрядного двоичного числа  $B = 1011_2$  в десятичное необходимо выполнить сложение весов разрядов (табл. 1.2).

Таблица 1.2				
Запись числа	1	0	1	1
Номер разряда	3	2	1	0
Вес разряда	23	22	21	20
Слагаемые числа	1·8	0·4	1·2	1·1

Число  $B$  может быть записано в виде суммы:  $B = 1 \cdot 8 + 1 \cdot 2 + 1 \cdot 1 = 11_{10}$ .

Аналогично выполняется преобразование 8-разрядного двоичного числа в десятичное. Пусть  $X = 10100101_2$ . Для заданного числа номер старшего разряда, равен 7, а вес разряда равен  $2^7 = 128$ . Число является суммой слагаемых:

$$X = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 128 + 32 + 4 + 1 = 165.$$

**Преобразование числа из десятичной системы счисления в двоичную.** Выполняется делением заданного числа на 2 и формированием результата из остатков от деления, начиная с младшего разряда (рис. 1.2).

Данный метод используют для преобразования десятичного числа в систему счисления с любым заданным основанием, при этом выполняется деление исходного числа на основание заданной (новой) системы счисления.

Первый остаток, получаемый от деления числа на 2, равен единице, если число нечетное. Это младший разряд двоичного числа. Затем полученное частное от деления вновь делится на 2 и определяется следующий разряд с большим весом. Процесс заканчивается, когда частное станет равным нулю. В результате выполнения преобразования в приведенном примере получим:  $165_{10} = 10100101_2$ .

**Шестнадцатеричная система счисления.** Применяется для компактной записи двоичных чисел. Она нередко используется в программах моделирования цифровых устройств для вывода результатов.

Для представления чисел в шестнадцатеричной системе используются 16 символов — цифры и латинские буквы: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Символ A соответствует десятичной цифре 10, B — 11, C — 12, D — 13, E — 14, F — 15. Основание, как в любой системе счисления, записывается как единица и ноль  $10_{16} = 16_{10}$ .

$$\begin{array}{rcl}
 165/2 = 81 + 1 & \text{Младший разряд} \\
 81/2 = 40 + 1 & \uparrow \\
 40/2 = 20 + 0 & \text{Направление чтения} \\
 20/2 = 10 + 0 & \text{двоичного числа} \\
 10/2 = 5 + 0 & \uparrow \\
 5/2 = 2 + 1 & \\
 2/2 = 1 + 0 & \\
 1/2 = 0 + 1 & \text{Старший разряд}
 \end{array}$$

Рис. 1.2. Преобразование числа из десятичной системы счисления в двоичную путем деления

Для шестнадцатеричной системы применимы все свойства позиционных систем счисления. Число может быть записано в виде суммы произведений, состоящих из цифр (от 0 до  $F$ ), умноженных на весовые коэффициенты, равные степеням числа 16:

$$H = a_{n-1} \cdot 16^{n-1} + a_{n-2} \cdot 16^{n-2} + \dots + a_3 \cdot 16^3 + a_2 \cdot 16^2 + a_1 \cdot 16^1 + a_0 \cdot 16^0.$$

Данная формула позволяет преобразовать шестнадцатеричное число в десятичное, например:  $A5_{16} = 10 \cdot 16 + 5 \cdot 1 = 165_{10}$ .

Шестнадцатеричное представление чисел наилучшим образом соответствует байтовой структуре ЭВМ, при которой разрядность большинства используемых чисел кратна байту. В свою очередь, 1 байт — это 8 бит, или две тетрады, а каждой тетраде соответствует одна цифра в шестнадцатеричной системе счисления.

Для преобразования числа из двоичной системы счисления в шестнадцатеричную необходимо разбить код числа на группы из 4 бит, начиная с младшего разряда, и представить каждую группу (или тетраду) в виде одной шестнадцатеричной цифры. Старшая группа бит может быть неполной, т.е. иметь менее четырех двоичных разрядов.

Для обратного преобразования числа из шестнадцатеричной системы в двоичную необходимо последовательно записать каждую шестнадцатеричную цифру в виде четырех двоичных разрядов, например:

$$10\ 0101_2 = 25_{16};\ A2C3_{16} = 1010\ 0010\ 1100\ 0011_2.$$

**Восьмеричная система счисления.** Она, как и шестнадцатеричная, применяется для сокращения записи двоичных чисел. Для представления чисел в восьмеричной системе используются 8 цифр: 0, 1, 2, 3, 4, 5, 6, 7.

Для восьмеричной системы число может быть записано в виде суммы произведений, состоящих из цифр (от 0 до 7), умноженных на весовые коэффициенты, равные степеням числа 8:

$$Q = a_{n-1} \cdot 8^{n-1} + a_{n-2} \cdot 8^{n-2} + \dots + a_3 \cdot 8^3 + a_2 \cdot 8^2 + a_1 \cdot 8^1 + a_0 \cdot 8^0.$$

Данная формула позволяет преобразовать восьмеричное число в десятичное, например:  $245_8 = 2 \cdot 64 + 4 \cdot 8 + 5 \cdot 1 = 165$ . Основание системы (в этой системе) также записывается как единица и ноль:  $10_8 = 8_{10}$ .

Чтобы преобразовать число из двоичной системы счисления в восьмеричную, необходимо разбить код числа на группы из 3 бит, начиная с младшего разряда, и представить каждую группу (триаду) в виде одной восьмеричной цифры, например:  $10\ 100\ 101_2 = 245_8$ . Старшая триада оказалась неполной.

Для обратного преобразования числа из восьмеричной системы в двоичную необходимо последовательно записать каждую восьмеричную цифру в виде трех двоичных разрядов.

Стандартный калькулятор, входящий в операционную систему Windows, позволяет переводить числа из одной системы счисления в другую. Он запускается через меню *Пуск / Стандартные / Калькулятор / меню Виг / Инженерный*. Калькулятор имеет переключатель систем счисления на четыре положения: *Hex* (шестнадцатеричная), *Dec* (десятичная), *Oct* (восьмеричная) и *Bin* (двоичная), а также дополнительные клавиши для набора латинских букв: **A—F**. Можно выбрать систему счисления, набрать число, а затем переключить калькулятор на другую систему и получить это число в новой системе счисления.

В различных языках программирования и в системах проектирования цифровых устройств применяется запись чисел в различных системах счисления. При этом в конце числа добавляют латинскую букву, обозначающую систему счисления. Для обозначения шестнадцатеричных чисел используется буква *h*, для восьмеричных — буква *o* (или *q*, чтобы не путать с нулем), а для двоичной системы — буква *b*. Десятичные числа можно записывать либо вообще без буквы либо с буквой *d*.

## 1.2. ВИДЫ ДВОИЧНЫХ КОДОВ

В цифровых устройствах для представления данных используются коды различных видов. В цифровой схемотехнике наиболее употребительными являются двоичные коды, представляющие целые числа без знака или со знаком.

**Целые беззнаковые двоичные числа.** Данный код — это рассмотренная ранее запись числа в двоичной системе счисления. Целые беззнаковые двоичные числа используются для представления величин, которые принципиально не могут быть отрицательными. Это коды счетчиков, адреса ячеек памяти, цифровые эквиваленты физических величин, для которых знак не указывается, например масса тела. Входные и выходные сигналы большинства цифровых устройств являются целыми беззнаковыми двоичными числами.

Каждый двоичный разряд представляет собой степень числа 2. Все разряды числа являются значащими. С помощью  $n$ -разрядного двоичного кода можно записать целые числа без знака в диапазоне от 0 до  $2^n - 1$ . Посредством 4-разрядного кода, например, можно записать 16 чисел в диапазоне от 0 до 15.



**Целые двоичные числа со знаком.** Для представления положительных и отрицательных чисел старший разряд числа используют для обозначения знака. Для положительных чисел в этот разряд, называемый *знаковым*, записывают 0, а для отрицательных чисел записывают 1. Остальные разряды являются значащими. В значащих разрядах записывают значение числа в прямом или в дополнительном коде.

**Прямой код**  $X_{\text{пк}}$  используют в значащих разрядах естественную двоичную запись абсолютного значения или модуля числа. Прямой код используют в устройствах вывода данных на индикацию, где отдельно отображаются знак числа и его модуль.

Диапазон чисел, которые можно представить в прямом  $n$ -разрядном коде, составляет от  $-(2^{n-1} - 1)$  до  $2^{n-1} - 1$ , а число 0 может быть положительным и отрицательным.

Прямой код можно использовать для суммирования чисел со знаком, но при этом сначала необходимо выполнить анализ знаков слагаемых, а затем подключить их к суммирующему или вычитающему устройству.

Примеры записи знаковых 4-разрядных чисел в прямом коде:

$$0\ 000_2 = 1\ 000_2 = 0; 0\ 001_2 = 1; 1\ 001_2 = -1; 0\ 111_2 = 7; 1\ 111_2 = -7.$$

**Дополнительный код** используют при суммировании чисел со знаком: положительных и отрицательных. Дополнительный код арифметической суммы двух чисел любых знаков равен арифметической сумме дополнительных кодов этих чисел. Суммирование чисел, представленных в дополнительном коде, выполняется без анализа знака, при этом получается правильный результат.

Все это становится возможным благодаря тому, что дополнительные коды чисел являются естественным кольцом чисел, получаемым в процессе счета, что можно пояснить, используя круговые диаграммы.

**Круговые диаграммы** (рис. 1.3). Они иллюстрируют процесс счета для 4-разрядных двоичных чисел без знака и для чисел со знаком в дополнительном коде.

Процесс счета является кольцевым. Любой счетчик с фиксированным количеством разрядов после прибавления единицы к максимальному значению сбрасывается в ноль, и начинается новый цикл счета.

На диаграммах снаружи каждого круга указаны двоичные коды. Сложению соответствует увеличение кода счетчика, что отображает перемещение по ходу часовой стрелки. Вычитанию соответствует противоположное направление перемещения.

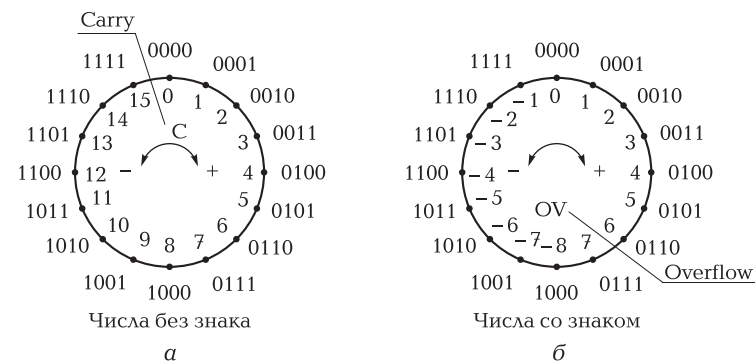


Рис. 1.3. Круговые диаграммы, поясняющие процесс счета:

$a$  — числа без знака;  $b$  — числа со знаком

Десятичные значения 4-разрядных чисел без знака показаны на диаграмме слева внутри круга. Для чисел без знака минимальное число равно нулю, максимальное равно 15, а количество наборов (вариантов) кодов составляет 16.

Результат вычислений должен соответствовать диапазону от 0 до 15. При выходе за пределы этого диапазона происходит переполнение разрядной сетки и возникает ошибка. В вычислительных устройствах при переходе от 15 к 0 или от 0 к 15 формируется признак переноса  $C$  (от слова *Carry* — перенос).

Кодирование чисел со знаком показано на диаграмме (рис. 1.3, б). Десятичные числа расположены на круге, как на шкале стрелочного прибора с нулем в середине шкалы. Влево от отметки 0 расположена отметка  $-1$ , затем  $-2$  и т.д. Старший разряд числа в дополнительном коде является знаковым разрядом, для положительных чисел он равен 0, а для отрицательных — 1.

Если из положительного числа многократно вычитать единицу, то число будет уменьшаться. На диаграмме этому процессу соответствует движение против часовой стрелки. Положительное число сначала будет уменьшаться, затем оно станет равным нулю, а при следующих вычитаниях произойдет переход в область отрицательных чисел.

Прибавлению единицы к отрицательному числу на диаграмме соответствует движение по часовой стрелке. Перенос  $C$ , возникающий при пересечении границы положительных и отрицательных чисел, не является признаком ошибки, поэтому он игнорируется независимо от направления перемещения.



Для знаковых 4-разрядных двоичных чисел в дополнительном коде имеем 16 различных значений кодов, при этом максимальное число равно +7, а минимальное число равно –8. Переполнение разрядной сетки при использовании дополнительного кода возникает при переходе границы между максимальным положительным числом +7 и минимальным числом –8. В вычислительных системах в этой ситуации формируют признак переполнения *OV* (от слова *Overflow* — переполнение).

Правила суммирования чисел со знаком, представленных в дополнительном коде, совпадают с правилами суммирования беззнаковых чисел, что позволяет использовать в этих случаях одинаковые суммирующие устройства. Это самое важное достоинство дополнительного кода.

Для вычисления дополнительного кода обычно используют два способа.

**Первый способ.** Дополнительный код отрицательного числа равен обратному коду абсолютной величины числа, плюс единица. **Обратный код** представляет собой инверсию всех разрядов прямого кода.

Для примера вычислим дополнительный код 4-разрядного числа –1.

Абсолютная величина числа:	0001;
Обратный код:	1110
Дополнительный код	1111.

Таблица 1.3

Число $X$	Модуль $X_{\text{АБС}}$	$X_{\text{А,К}}$ десятичный	$X_{\text{А,К}}$ двоичный
–1	1	$16 - 1 = 15$	1111
–2	2	$16 - 2 = 14$	1110
–3	3	$16 - 3 = 13$	1101
–4	4	$16 - 4 = 12$	1100
–5	5	$16 - 5 = 11$	1011
–6	6	$16 - 6 = 10$	1010
–7	7	$16 - 7 = 9$	1001
–8	8	$16 - 8 = 8$	1000

Второй способ основан на использовании правила: модуль отрицательного числа  $|X|$  и дополнительный код  $X_{\text{А,К}}$  в сумме равны  $2^n$ , где  $n$  — количество разрядов. Это правило позволяет записать два выражения, позволяющие определить дополнительный код для заданного модуля числа и, наоборот, вычислить модуль числа по заданному дополнительному коду:

$$X_{\text{А,К}} = 2^n - |X|, \text{ или } |X| = 2^n - X_{\text{А,К}}.$$

Эти вычисления целесообразно выполнять в десятичной системе, а затем переводить результат в двоичную систему (табл. 1.3).

### 1.3. ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ В ДВОИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

Сложение двоичных чисел выполняется поразрядно, начиная с младшего разряда, по известным правилам арифметики. Поясним на примере, содержащем различные комбинации кодов в отдельных разрядах слагаемых (табл. 1.4).

В нулевом разряде оба слагаемых имеют значение 0, сумма в данном разряде равна нулю. В первом разряде одно из слагаемых равно единице, сумма равна единице. Во втором разряде оба слагаемых равны единице, а сумма равна двум. В двоичной системе необходимо в данном разряде записать нуль и добавить единицу переноса в старший третий разряд. В третьем разряде суммируются три единицы, поэтому в данном разряде результат равен единице, и возникает перенос в четвертый разряд.

Пример показывает, что при суммировании  $n$ -разрядных слагаемых сумма может иметь  $(n + 1)$  разряд.

Таблица 1.4

Номер разряда	4	3	2	1	0	—
Перенос	1	1	—	—	—	—
Первое слагаемое	—	1	1	0	0	$12_{10}$
Второе слагаемое	—	1	1	1	0	$14_{10}$
Сумма	1	1	0	1	0	$26_{10}$

**Вычитание двоичных чисел.** Обычно выполняется путем сложения уменьшаемого с дополнительным кодом вычитаемого, при этом суммирование и вычитание выполняет одно и то же устройство.

Другой способ нахождения разности при использовании отдельного вычитающего устройства выполняется по правилам:

$$0 - 0 = 0; 1 - 0 = 1; 1 - 1 = 0; 0 - 1 = 1 \\ \text{и заем 1 из старшего разряда.}$$

**Умножение двоичных чисел.** Выполняется подобно умножению в десятичной системе — столбиком. При этом используются очевидные правила:

$$0 \times 0 = 0; 0 \times 1 = 0; 1 \times 0 = 0; 1 \times 1 = 1.$$

$$\begin{array}{r} 101101 \rightarrow 45_{10} \\ \times 1011 \rightarrow 11_{10} \\ \hline 101101 \\ + 101101 \\ + 000000 \\ + 101101 \\ \hline 111101111 \rightarrow 495_{10} \end{array}$$

Пример умножения двоичных чисел показывает, что умножение в двоичной системе счисления сводится к выполнению операций сдвига первого числа (множимого) влево и суммирования полученных после сдвига чисел.

**Деление в двоичной системе.** Можно выполнить посредством операций вычитания и сдвига.

## 1.4. ОСНОВЫ АЛГЕБРЫ ЛОГИКИ

При анализе и синтезе цифровых схем используются тождества и теоремы алгебры логики, разработанной английским математиком Джорджем Булем, по имени которого ее называют **булевой алгеброй**.

Функции, используемые в алгебре логики, называют логическими, переключательными или булевыми. *Логические функции* могут быть одной, двух и более логических переменных. Все переменные, используемые в алгебре логики, могут принимать только два значения, которые обозначили как *True* — истина и *False* — ложь, что позволяет математически описывать высказывания. Впоследствии,

применительно к цифровым устройствам, эти значения обозначили как 1 и 0. При положительном кодировании цифровых сигналов, которое применяется чаще, чем отрицательное, значению 1 соответствует более высокое напряжение, а сигналу 0 — низкое. В английском языке логический сигнал 1 обозначают буквой *H* (*High* — высокий), а сигнал 0 — буквой *L* (*Low* — низкий).

Важную задачу алгебры логики составляет выбор функционально полной системы логических функций и соответствующего набора логических элементов. Функционально полная система, или **базис** — это такой набор простых логических функций и соответствующих элементов, с помощью которых можно представить любую более сложную логическую функцию. Существует несколько базисов.

основой для анализа и синтеза цифровых схем является **булев базис**, содержащий три типа элементов, выполняющих операции И, ИЛИ, НЕ. В алгебре логики доказано, что из элементов указанных трех типов можно построить любое цифровое устройство;

всего один элемент, выполняющий операцию И — НЕ, которая называется «штрих Шеффера», обладает свойством функциональной полноты, образует **универсальный базис**. Это значит, что, имея достаточное количество элементов только одного типа И — НЕ, можно построить цифровую схему любой сложности, например процессор. В первых образцах интегральных микросхем этот элемент был одним из основных;

- функция ИЛИ — НЕ, которая называется «стрелка Пирса», также обладает функциональной полнотой и является **универсальным базисом**;
- существует также базис, содержащий функцию «Сумма по модулю 2» (другое название — «Исключающее ИЛИ»), который использовался редко из-за некоторых сложностей схемотехнической реализации данной функции.

В настоящее время не существует никаких ограничений на сложность схемотехнической реализации логических элементов и на количество их типов. Поэтому для построения цифровых схем используют **расширенный базис** — более широкий набор элементов, который обладает избыточностью, но позволяет оптимизировать параметры разрабатываемых устройств.

При преобразованиях логических уравнений используют **логические тождества и теоремы**. Приведем основные из них.

Для логических сигналов справедливы известные **переместительный и сочетательный** законы:

$$a \cdot b = b \cdot a; a \vee b = b \vee a; a \cdot (b \vee c) = a \cdot b \vee a \cdot c; a \vee b \cdot c = (a \cdot b) \vee (a \cdot c).$$

При минимизации логических функций используются характерные законы алгебры логики, позволяющие упростить уравнение.

**Закон склеивания** выражается следующими формулами:

$$a \cdot b \vee a \cdot \bar{b} = a; (a \vee b) \cdot (a \vee \bar{b}) = a.$$

**Закон поглощения** описывают выражения:

$$a \vee a \cdot b = a; a \cdot (a \vee b) = a.$$

Важную роль для аналитических преобразований логических уравнений играют первый и второй **законы де Моргана**:

$$\bar{a} \vee \bar{b} \vee \bar{c} \vee \bar{d} = \overline{a \cdot b \cdot c \cdot d}; \quad \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} = \overline{a \vee b \vee c \vee d}.$$

Доказательство справедливости приведенных формул несложно выполнить методом подстановки или посредством таблиц истинности.

## 1.5. АНАЛИЗ БАЗОВЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Анализ логических элементов неразрывно связан с особенностями их применения. Он включает в себя изучение работы элемента, определение логических функций, составление таблиц истинности, построение временных диаграмм.

Логические элементы *расширенного базиса* (их также называют вентили — gates) — это наиболее простые цифровые устройства. При интегральном исполнении в одном корпусе микросхемы может располагаться от одного до шести одинаковых логических элементов. Обычно каждый логический элемент имеет несколько входов (от 1 до 12) и один выход. При этом связь между выходным сигналом и входными сигналами (таблица истинности) предельно проста. Каждой комбинации входных сигналов элемента соответствует уровень нуля или единицы на его выходе. Никакой внутренней памяти у логических элементов нет, поэтому они относятся к группе комбинационных микросхем.

**Логический элемент НЕ.** Этот элемент — инвертор, выполняет операцию *инверсия*:  $x = \bar{a}$ .

Условное графическое обозначение инвертора в российском стандарте — прямоугольник, вход — слева, выход — справа, функцию *инверсия* отображает кружок на выходе элемента (рис. 1.4, а).

В американском стандарте (рис. 1.4, б), который используется в графических редакторах систем автоматизированного проектирования цифровых устройств, инвертор обозначают треугольником, функциональное назначение элемента определяет надпись *NOT* и кружок на выходе.

Таблица 1.5

$a$	$x = \bar{a}$
0	1
1	0

Таблица истинности для инвертора содержит всего две строки (табл. 1.5).

Правила для логической операции некоторой переменной величины с константой или переменной величины с самой собой или ее инвертированным значением называются **аксиомами**.

Для операции НЕ справедлива аксиома:  $\bar{\bar{a}} = a$ , в соответствии с которой двойное отрицание возвращает исходное значение аргумента.

**Логический элемент И.** Он выполняет операцию *конъюнкция*, или *логическое умножение*. Базовые элементы И на кристаллах ПЛИС могут иметь от 2 до 12 входов.

Для операции И справедливы следующие аксиомы:

$$a \cdot 0 = 0; \quad a \cdot 1 = a; \quad a \cdot a = a; \quad a \cdot \bar{a} = 0.$$

В российском стандарте (рис. 1.5, а) логический элемент И обозначает прямоугольник, в котором функциональное назначение элемента, операцию И, обозначает знак &. В американском стандарте обозначение другое (рис. 1.5, б). Функциональное назначение элемента указывает надпись, например, *AND3*, что означает элемент И с тремя входами.

Работу элемента описывают логическая функция  $y = a \cdot b \cdot c$  и таблица истинности (табл. 1.6). Для выходного сигнала элемента И с любым числом входов справедливо правило: сигнал на выходе элемента И равен единице, если на всех входах сигналы равны единице.

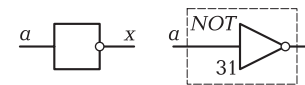


Рис. 1.4. Обозначения инвертора:

а — российский стандарт; б — американский стандарт

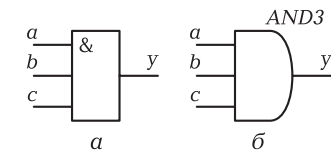


Рис. 1.5. Обозначения элементов И:

а — российский стандарт; б — американский стандарт

Таблица 1.6

$a$	$b$	$c$	$y = a \cdot b \cdot c$	$z = \overline{a \cdot b \cdot c}$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Из таблицы истинности и аксиом также можно сделать вывод, что один из входов, например  $a$ , будет активным и изменения сигнала с этого входа будут передаваться на выход, если на все остальные входы подаются сигналы, равные 1.

Использование логического элемента И для выполнения простейших операций обработки сигналов — выделения и объединения — поясняют временные диаграммы (рис. 1.6).

Выделение некоторой части информационного сигнала называется **стробированием**. **Строб** — это управляющий сигнал, который определяет интервал времени, когда происходит передача информационного сигнала. Схема стробирования — это управляемая схема, которая пропускает или не пропускает информационный сигнал.

Для схемы И один из входов ( $a$ ) можно использовать как информационный, а два других входа ( $b, c$ ) как управляющие. Схема стробирования на элементе И пропускает положительные импульсы при положительных управляющих сигналах. Сигнал со входа  $a$  будет передаваться на выход  $y$  только при  $b = c = 1$  (интервал времени  $t_1 \dots t_2$ ). На рис. 1.6 также показана операция объединения отрицательных импульсов посредством элемента И (интервал времени  $t_3 \dots t_4$ ).

**Логический элемент И—НЕ.** Он выполняет операцию *Штрих Шеффера*, или отрицание конъюнкции, что описывается формулой:  $z = \overline{a \cdot b \cdot c}$ . Условные графические обозначения элементов И—НЕ отличаются от обозначений элемента И наличием кружка на выходе, что означает инверсию выходного сигнала (рис. 1.7).

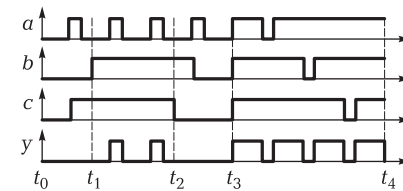


Рис. 1.6. Временные диаграммы, поясняющие работу элемента И

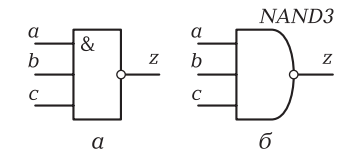


Рис. 1.7. Обозначения [а, б] элементов И—НЕ

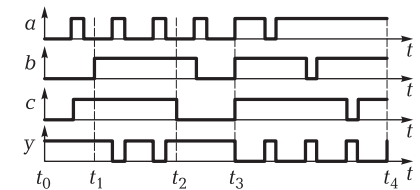


Рис. 1.8. Временные диаграммы, поясняющие работу элемента И—НЕ

Таблица истинности в табл. 1.6 (столбец  $z = \overline{a \cdot b \cdot c}$ ) позволяет сформулировать правило определения выходного сигнала независимо от числа входов: на выходе элемента И—НЕ имеем 0, если на все входы подан сигнал 1.

Элементы данного типа использовались как основные в интегральных схемах типа ТТЛ. В сериях интегральных схем малой степени интеграции, например 1533, содержатся элементы И—НЕ, имеющие 2, 3, 4 и 8 входов.

Схема стробирования на элементе И—НЕ, как и схема И, пропускает положительные импульсы при положительных управляющих сигналах, но выдает инвертированный выходной сигнал (рис. 1.8, интервал времени  $t_1 \dots t_2$ ). При объединении отрицательных импульсов на выходе формируются положительные импульсы (интервал времени  $t_3 \dots t_4$ ).

**Логический элемент ИЛИ.** Выполняет операцию *дизъюнкции*, или *логическое сложение*. Логические элементы ИЛИ могут иметь два входа и более.

Для операции ИЛИ справедливы следующие аксиомы:

$$a \vee 0 = a; \quad a \vee 1 = 1; \quad a \vee a = a; \quad a \vee \bar{a} = 1.$$

Функциональное назначение элемента ИЛИ в российском стандарте (рис. 1.9, а) обозначают цифрой 1, а в американском надписью, например, OR3 — элемент ИЛИ с тремя входами (рис. 1.9, б).

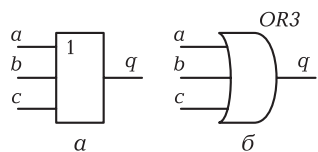


Рис. 1.9. Обозначения элементов ИЛИ:

*a* — российский стандарт; *б* — американский стандарт

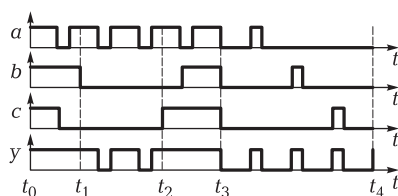


Рис. 1.10. Временные диаграммы, поясняющие работу элемента ИЛИ

Элемент ИЛИ, имеющий три входа, описывает логическая функция:  $q = a \vee b \vee c$ .

Таблица истинности логического элемента ИЛИ (табл. 1.7) позволяет сформулировать правило для определения выходного сигнала, справедливое для логических элементов с любым числом входов: выходной сигнал элемента ИЛИ равен нулю, если все входные сигналы равны нулю.

Работу элемента ИЛИ поясняют временные диаграммы (рис. 1.10). Если один из входов (*a*) использован как информационный, а два других входа (*b*, *c*) как управляющие, то сигнал с входа *a* будет передаваться на выход *q* только при  $b = c = 0$ . Схема стробирования на элементе ИЛИ пропускает отрицательные импульсы при отрицательных управляющих сигналах (интервал времени  $t_1 \dots t_2$ ).

Таблица 1.7

<i>a</i>	<i>b</i>	<i>c</i>	$q = a \vee b \vee c$	$v = \overline{a \vee b \vee c}$
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

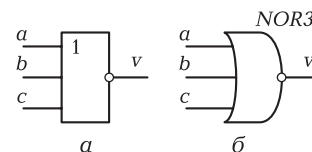


Рис. 1.11. Обозначения элементов ИЛИ—НЕ:

*a* — российский стандарт; *б* — американский стандарт

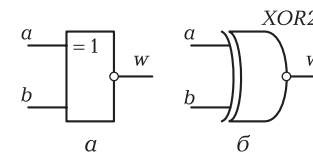


Рис. 1.12. Обозначения элементов Иключающее ИЛИ:

*a* — российский стандарт; *б* — американский стандарт

На рис. 1.10 также показана операция объединения положительных импульсов посредством элемента ИЛИ (интервал времени  $t_3 \dots t_4$ ).

**Логический элемент ИЛИ—НЕ.** Выполняет операцию *стрелка Пирса*, или *отрицание дизъюнкции*. Выходной сигнал элемента ИЛИ—НЕ (рис. 1.11) записывается в виде:  $v = \overline{a \vee b \vee c}$ .

На выходе элемента ИЛИ—НЕ имеем сигнал 1, если на всех входах сигнал равен нулю (см. табл. 1.7 — столбец  $v = \overline{a \vee b \vee c}$ ).

**Логический элемент Иключающее ИЛИ.** Функциональное назначение элемента Иключающее ИЛИ в российском стандарте (рис. 1.12, *a*) обозначают знаком равенства и цифрой 1, а в американском надписью XOR2 — элемент Exclusive OR — исключающее ИЛИ с двумя входами (рис. 1.12, *б*).

Данная операция называется также «Сумма по модулю два» и обозначается знаком « $\oplus$ » или перечеркнутым знаком дизъюнкции  $w = a \vee b = a \oplus b$ . Название «Сумма по модулю 2» означает, что сумма не может превышать число 2, а результат суммирования должен представлять один разряд (табл. 1.8).

Для операции Иключающее ИЛИ справедливы следующие аксиомы:

$$a \oplus 0 = a; \quad a \oplus 1 = \bar{a}; \quad a \oplus a = 0; \quad a \oplus \bar{a} = 1.$$

Данный элемент используют как управляемый инвертор (рис. 1.13). При подаче на вход *a* данных, а на вход *b* управляющего сигнала при  $b = 0$  получим передачу на выход *w* сигнала *a* без изменения  $w = a$ , а при  $b = 1$  — передачу сигнала с инверсией.

Операция суммирования по модулю 2 (свертка по модулю 2) широко применяет-

Таблица 1.8

<i>a</i>	<i>b</i>	$w = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

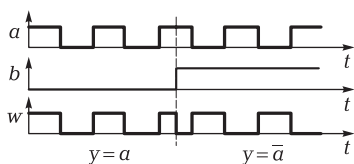


Рис. 1.13. Временные диаграммы, поясняющие работу элемента Иключающее ИЛИ

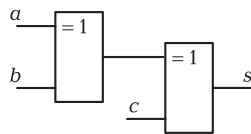


Рис. 1.14. Трехвходовой сумматор по модулю 2

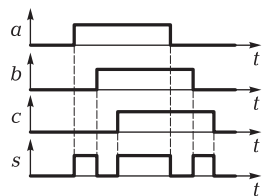


Рис. 1.15. Временные диаграммы, поясняющие работу сумматора по модулю 2

ся в системах контроля правильности передач и хранения данных. Для передаваемого слова данных формируется признак контроля по четности путем суммирования всех разрядов по модулю 2. Выполнение такого же суммирования принимаемых данных позволяет определить их достоверность.

Для операции Сумма по модулю 2 любой входной сигнал является активным при любых комбинациях сигналов на других входах. Каждое изменение сигнала на одном из входов при постоянных сигналах на других входах вызывает изменение выходного сигнала

на противоположное. Действительно, прибавление единицы к числу всегда изменяет признак четности на противоположный.

Сумматор по модулю 2, имеющий три входа (рис 1.14), описывает логическая функция  $s = a \oplus b \oplus c$ . Его работу поясняют таблица истинности (табл. 1.9) и временные диаграммы (рис. 1.15).

Выходной сигнал схемы Сумма по модулю 2 с любым количеством входов определяет правило: выход схемы Сумма по модулю 2 равен 1, когда число входов, имеющих состояние 1, нечетно.

Таблица 1.9

a	b	c	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Охарактеризуйте основные системы счисления, используемые в цифровых вычислительных устройствах.
2. Поясните метод преобразования числа из двоичной системы счисления в десятичную систему.
3. Охарактеризуйте метод преобразования числа из десятичной системы счисления в двоичную систему.
4. Каковы способы кодирования чисел со знаком?
5. Поясните способ преобразования прямого кода в дополнительный.
6. Дайте определение логических функций дизъюнкции и конъюнкции.
7. Назовите функционально полные системы логических элементов.
8. Запишите аксиомы для базовых логических операций.
9. Сформулируйте правила для определения выходного сигнала для базовых логических элементов.
10. Запишите основные законы алгебры логики.



# ЭЛЕМЕНТНАЯ БАЗА ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

Электронные вычислительные машины подразделяются на поколения в зависимости от элементной базы, используемой для их построения. Вычислительные машины первого поколения «Стрела», БЭСМ, «Урал», «Минск» были построены на электронных лампах. Самой быстродействующей в Европе в свое время была разработанная в СССР в 1953 г. ламповая ЭВМ БЭСМ (Большая электронная счетная машина).

Изобретение в 1948 г. в США биполярного транзистора, а в 1952 г. в Японии полевого транзистора явилось началом эры ЭВМ второго поколения, элементную базу которых составляли транзисторы и полупроводниковые диоды.

Интенсивное развитие ЭВМ и цифровых вычислительных устройств происходило с появлением и совершенствованием интегральных микросхем малой и средней степени интеграции, которые использовались для построения третьего поколения ЭВМ.

Последующие поколения вычислительных машин и систем строятся на больших и сверхбольших интегральных схемах и обладают все более и более высокими параметрами и характеристиками.

Функциональные возможности интегральных схем непрерывно растут. В зависимости от степени интеграции, которая определяется количеством элементов, входящих в интегральную схему, различают интегральные схемы:

- малой степени интеграции (МИС), содержащие до 50 элементов;
- средней степени интеграции (СИС) — до 500 элементов;
- большой степени интеграции (БИС) — до 10000 элементов;
- сверхбольшой степени интеграции (СБИС) — более 10000 элементов.

Современные сверхбольшие интегральные схемы содержат миллионы элементов.

Интегральные микросхемы выпускаются в виде серий. *Серией* логических элементов называется набор логических элементов, предназначенный для построения цифровых устройств, функционально объединяемый общими электрическими, конструктивными и технологическими параметрами, использующий одинаковый способ представления информации и одинаковый тип межэлементных связей — одинаковую схемную логику. Серии содержат элементы для выполнения логических операций, запоминающие элементы, элементы, реализующие функции узлов ЭВМ. Серии элементов в своем составе имеют не только многовходовые логические элементы, но и более сложные узлы вычислительных устройств — триггеры, счетчики, регистры, запоминающие устройства, а также элементы для усиления, восстановления и формирования сигналов стандартной формы.

Обозначение интегральных микросхем отечественного производства содержит ряд букв и цифр. Первые буквы обозначения определяют тип корпуса микросхемы. Первая буква К означает, что эти микросхемы общего или коммерческого применения, вторая буква означает материал и тип корпуса: Р — пластмассовый, М — керамический. Отсутствие буквы К означает микросхемы специального применения. Типы корпусов приведены в справочниках. Следующие элементы обозначения — три или четыре цифры — обозначают серию, следующие две буквы — функциональное назначение схемы, а последние цифры — номер разработки.

Обозначение 1533ЛА3 соответствует микросхеме специального применения серии 1533 (технология ТТЛШ), корпус — металлический, тип элементов — двухвходовые И — НЕ. Микросхема КР1554ЛЕ1: применение — общее, корпус — пластмассовый, серия 1554 (технология КМОП), тип элементов — двухвходовые ИЛИ — НЕ.

Наиболее широкий функциональный состав имеют серии элементов малой и средней степени интеграции типа ТТЛШ серии КР1533 (транзисторно-транзисторная логика с диодами Шоттки), элементы типа КМОП серии КР1554 и элементы эмиттерно-связанной логики (ЭСЛ) серий 100, 500.

Применение интегральных схем малой степени интеграции в настоящее время ограничено. Чаще всего они используются как буферные элементы, предназначенные для связи больших интегральных схем с другими устройствами. Схемы логических элементов, разработанные для реализации в интегральных схемах малой степени интеграции различных серий, являются основой для схемотехники больших интегральных схем.

## 2.1. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ НА БИПОЛЯРНЫХ ТРАНЗИСТОРАХ

Базовый логический элемент серии ДТЛ (диодно-транзисторная логика). Это один из наиболее простых, реализует логическую функцию И—НЕ, содержит логическую схему И на диодах и инвертор на одном транзисторе (рис. 2.1).

Данная схема использовалась сначала для построения логического элемента на дискретных компонентах — диодах и транзисторах, впоследствии была реализована в интегральном исполнении, послужила основой для разработки схем элементов других серий.

Для определения логической функции элемента рассмотрим работу схемы.

Если на оба входа схемы подано положительное напряжение  $+E$  (логическая 1), транзистор будет открыт, а напряжение коллектор — эмиттер мало и составляет приблизительно 0,2 В (логический 0). Диоды  $VD1$  и  $VD2$  будут закрыты, в базу транзистора  $VT1$  будет поступать ток, протекающий от источника питания  $+E$  через резистор  $R_1$  и через открытые диоды  $VD3$ ,  $VD4$ .

Во всех остальных случаях, если хотя бы на один из входов схемы подается напряжение, равное нулю (логический 0), транзистор будет закрыт и на его выходе будет сигнал логическая 1. Ток базы транзистора в этом случае будет равен нулю, так как ток, протекающий через резистор  $R_1$ , будет замыкаться на землю через входной диод, на катод которого подано нулевое напряжение. На коллекторе транзистора появится высокий потенциал, соответствующий логическому сигналу 1.

Логический элемент выполняет операцию И-НЕ (табл. 2.1).

Напряжение, при котором происходит переход транзистора из состояния отсечки в состояние насыщения и наоборот, зависит от

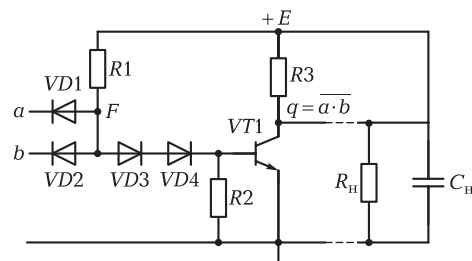


Рис. 2.1. Базовый элемент ДТЛ

потенциала точки  $F$ , который определяется при закрытых входных диодах. Этот потенциал равен сумме напряжений на трех  $p-n$ -переходах, смещенных в прямом направлении. Необходимо учесть переходы диодов  $VD3$ ,  $VD4$  и переход база-эмиттер транзистора. Падение напряжения на  $p-n$ -переходе, смещенном в прямом направлении обозначают  $U^*$ . Для  $p-n$ -переходов, выполненных на базе кремния, напряжение  $U^*$  составляет (0,6...0,7) В. Таким образом, состоянию насыщения транзистора соответствует потенциал точки  $F$   $UF = 3U^*$ . Диоды  $VD1$  и  $VD2$  называются компенсирующими, или смещающими, поскольку они увеличивают потенциал точки  $F$  и смещают пороговый уровень входного открывающего напряжения в большую сторону. Переход транзистора из состояния насыщения в состояние отсечки произойдет при открывании одного из входных диодов, например  $VD1$ . Для этого входное напряжение  $U_A$  должно быть меньше потенциала точки  $F$  и составлять  $2U^* \approx 1,3$  В. Входное напряжение, меньшее этого значения, будет соответствовать сигналу 0, а напряжение, превышающее это значение, — сигналу 1.

Устройство, подключенное к выходу логического элемента, называется **нагрузкой**. Это может быть линия связи и другой элемент. Нагрузка, как правило, имеет емкостной характер. На схеме показаны входное сопротивление и входная емкость цепи, подключенной к элементу:  $R_n$  и  $C_n$ .

При работе логического элемента на емкостную нагрузку ухудшается его быстродействие, так как возникает задержка сигнала, обусловленная переходными процессами при заряде и разряде конденсатора  $C_n$ . Длительность процессов зависит от емкости  $C_n$  и от токов, которые заряжают или разряжают эту емкость при изменениях выходного сигнала элемента.

Наибольшую длительность в данной схеме имеет процесс заряда конденсатора от источника питания через резистор  $R_3$ , возникающий при изменении выходного сигнала от 0 к 1. Впоследствии будет показано, что для уменьшения продолжительности этого процесса необходимо использовать двухтактный выходной каскад, в котором вместо резистора  $R_3$  включен транзистор.

**Базовый логический элемент ТТЛ.** Схема базового логического элемента ТТЛ построена на основе рассмотренной схемы элемента ДТЛ. В данной схеме (рис. 2.2) вместо диодной логической схемы использован многоэмиттерный транзистор  $VT1$ . Такая замена упро-

Таблица 2.1

$a$	$b$	$q$
0	0	1
0	1	1
1	0	1
1	1	0

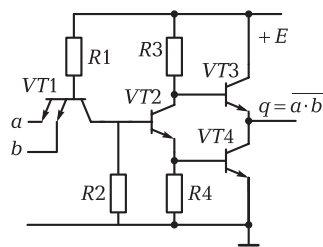


Рис. 2.2. Базовый элемент ТТЛ

стила технологию изготовления элемента. Изготовить один многоэмиттерный транзистор оказалось проще, чем несколько диодов.

Переходы транзистора VT1 между базой и эмиттерами работают подобно входным диодам предыдущей схемы, а переход между базой и коллектором выполняет функцию компенсирующего диода. Число входов может быть увеличено добавлением эмиттеров в транзистор VT1.

Базовый логический элемент ТТЛ имеет мощный двухтактный выходной каскад на транзисторах VT3 и VT4, которые открываются в противофазе. Управление двухтактным каскадом выполняет транзистор VT2, формирующий на коллекторе и эмиттере противофазные сигналы. Построение выходного каскада базового логического элемента по двухтактной схеме повышает быстродействие элемента при его работе на емкостную нагрузку.

Если на оба входа логического элемента поданы логические сигналы 1, то эмиттерные переходы транзистора VT1 будут закрыты (подобно элементу ДТЛ), а ток, протекающий через резистор R1, откроет транзисторы VT2 и VT4. При этом потенциал на коллекторе транзистора VT2 будет низким. В результате транзистор VT3 будет закрыт. Выходное напряжение в этом случае соответствует сигналу 0.

Изменение сигнала с 1 на 0 (хотя бы на одном из входов) изменит состояния транзисторов на противоположные и создаст на выходе сигнал 1.

Скорость переключения транзисторов снижается из-за переходных процессов, возникающих при рассасывании зарядов, накопленных в области базы транзистора. Накопление зарядов происходит, когда транзистор находится в состоянии насыщения. Для повышения максимальной частоты переключений необходимо предотвратить насыщение открытого транзистора и накопление зарядов в области базы.

В более поздних сериях микросхем рассмотренная схема элемента была модифицирована. Параллельно переходам коллектор — база транзисторов были подключены диоды Шоттки. В элементах ТТЛШ (ТТЛ с диодами Шоттки) диод препятствует возникновению насыщения транзистора при его открывании. Подключение диодов уменьшает время задержки распространения сигнала

почти в 3 раза. В настоящее время используются схемотехника ТТЛШ.

**Буферный элемент с открытым коллектором.** Буферные элементы предназначены для работы на внешнюю нагрузку, потребляющую большой ток, поэтому выходные каскады буферных элементов обладают повышенной нагрузочной способностью. К выходам буферных элементов могут подключаться различные элементы (обмотки реле и двигателей), различные индикаторы.

Буферные элементы используют для выдачи сигналов на общие шины. В цифровых системах к проводникам общей шины нередко подключается много приемников цифровых сигналов, поэтому буферный элемент должен формировать на выходе мощный сигнал.

Простейший **буферный элемент с открытым коллектором** — это инвертор типа ДТЛ (рис. 2.3). Выходной каскад имеет только один транзистор VT1, коллектор которого является выходом элемента. Отличительная особенность схемы элемента с открытым коллектором — отсутствие резистора в цепи коллектора транзистора.

Для получения на выходе такого элемента логического нуля необходимо открыть транзистор VT1, подавая на вход сигнал логической 1. Диод VD1 будет закрыт, а ток, протекающий через резистор R1 и диоды VD2, VD3, откроет транзистор VT1, и на его коллекторе будет низкий потенциал, близкий к нулю.

Для получения на выходе логического сигнала 1 необходимо закрыть транзистор, а к коллектору транзистора подключить внешний резистор Rн, соединенный с источником питания. Такой резистор называют подтягивающим, или способствующим повышению выходного напряжения логического элемента и стремлению его к напряжению источника питания E.

Несколько буферов с открытым коллектором можно подключить параллельно к общему проводнику, который имеет один подтягивающий резистор, соединенный с источником питания (рис. 2.4). Условное графическое обозначение логического элемента с открытым коллектором содержит ромб, подчеркнутый снизу.

**Внимание!** Подобное соединение базовых элементов с двухтактным выходным каскадом недопустимо, оно приводит к выходу элементов из строя.

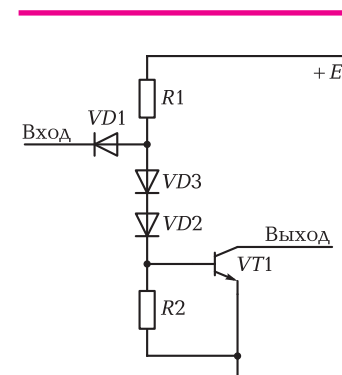


Рис. 2.3. Инвертор типа ДТЛ

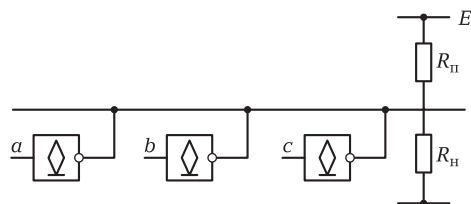


Рис. 2.4. Подключение элементов с открытым коллектором к общей шине

Если подать на входы всех элементов сигнал 0, то все транзисторы выходных каскадов в элементах будут закрыты, на выходах всех элементов и на общем проводнике будет высокий потенциал, создаваемый источником питания  $E$ . Напряжение на общем проводнике в этом случае, определяемое формулой  $U_Y = ER_n(R_n + R_n)$ , должно соответствовать сигналу логической 1. Сигнал логической единицы существует на проводнике общей шины по умолчанию, его называют «пассивный высокий».

Если одно из устройств должно выдать сигнал на общий проводник, то выходной транзистор этого устройства должен передавать нули, замыкая выходной сигнал на землю. Сигнал логического нуля является активным низким (табл. 2.2).

Каскады с открытым коллектором используются на выходе большого количества интегральных схем, предназначенных для подключения к общей шине. Это микросхемы памяти, дешифраторы, шинные формирователи, буферные регистры.

**Буферный элемент с тремя состояниями выхода.** Логический элемент с тремя состояниями выхода имеет двухтактный выходной каскад, в схему которого добавлены цепи, позволяющие закрыть оба транзистора этого каскада. Один из вариантов схемы (рис. 2.5) представляет собой логический элемент ТТЛ, в котором транзисторы  $VT3$ ,  $VT4$  могут быть закрыты, если в цепь базы транзистора  $VT5$  подать высокий потенциал, открывающий транзистор. В результате диоды  $VD1$ ,  $VD2$  создадут на базах транзисторов  $VT3$ ,  $VT4$  низкие закрывающие потенциалы.

Если оба транзистора выходного каскада закрыты, то выход элемента оказы-

Таблица 2.2			
$a$	$b$	$c$	$y$
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
0	1	1	0

вается никуда не подключенным: ни к шине питания, ни к шине земли. Такое состояние выхода называют *отключенным*, *высокоомным* или *Z-состоянием*.

Рис. 2.5. Схема элемента с тремя состояниями выхода

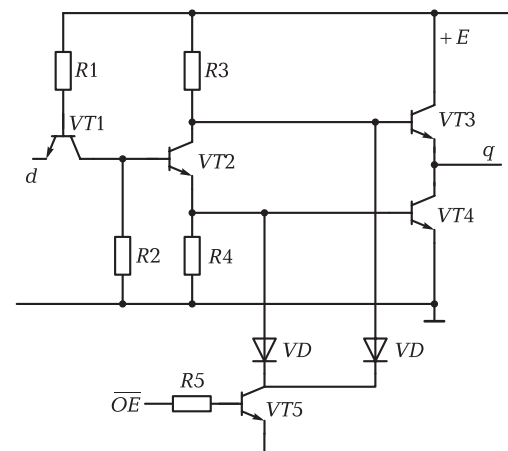


Рис. 2.6. Подключение элементов с тремя состояниями выхода к общей шине

вается никуда не подключенным: ни к шине питания, ни к шине земли. Такое состояние выхода называют *отключенным*, *высокоомным* или *Z-состоянием*.

Для перехода в отключенное состояние элемент данного типа имеет дополнительный управляющий инверсный вход  $\overline{OE}$  (*Output Enable* — разрешение выхода). При  $\overline{OE} = 0$  передачи сигнала разрешена, а при  $\overline{OE} = 1$  выход находится в отключенном состоянии.

Логические элементы, имеющие выходной каскад данного типа, обозначают перечеркнутым ромбом (рис. 2.6). Устройства, имеющие на выходе буферный элемент с тремя состояниями выхода, можно подключать к общей шине параллельно. При этом передача сигналов на выход должна быть разрешена только для одного буферного элемента, а все остальные должны быть в отключенном состоянии. В данной схеме на первый элемент подается сигнал  $\overline{OE} = 1$ , поэтому на выход будет передаваться инвертированный сигнал  $a$ .

Блоки вывода ПЛИС могут быть запрограммированы как буферы с открытым коллектором или с тремя состояниями выхода.

## 2.2. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ НА ПОЛЕВЫХ ТРАНЗИСТОРАХ

Особенностью интегральных схем, выполненных на полевых транзисторах, являются малая потребляемая мощность, высокая помехоустойчивость, возможность работы в широких интервалах питающих напряжений. Микросхемы на полевых транзисторах обеспечивают высокую технологичность изготовления. Полевые транзисторы, в сравнении с биполярными, имеют меньшие размеры и проще в изготовлении, что позволяет разместить на единице площади кристалла больше элементов. Это важно при построении функционально сложных устройств с малым потреблением мощности.

Наиболее удачные схемные решения имеют интегральные схемы на полевых транзисторах с различными типами каналов. Сочетания транзисторов различных типов называют **комплементарными** (от англ. *complement* — дополнение). По мере совершенствования технологии, направленной на повышение быстродействия, самыми массовыми становятся микросхемы на полевых транзисторах типа на комплементарных полевых транзисторах со структурой металл — оксид — полупроводник (КМОП). Микросхемы этого типа являются элементной базой современных СБИС и процессоров.

Для построения логических элементов семейства КМОП используются полевые транзисторы с изолированным затвором и ин-

дуцированным каналом  $n$ - и  $p$ -типа. Важной особенностью транзисторов с индуцированным каналом является отсутствие тока стока при нулевом напряжении между затвором и истоком. При  $U_{ЗИ} = 0$  транзистор с индуцированным каналом закрыт. Для того чтобы открыть транзистор с каналом  $n$ -типа, необходимо на его затвор подать положительное относительно истока напряжение и наоборот, транзистор с  $p$ -каналом открывается отрицательным напряжением между затвором и истоком. В схеме КМОП инвертора (рис. 2.7) нижний транзистор  $VT2$  имеет тип проводимости канала  $n$ , а верхний  $VT1$  — канала  $p$ .

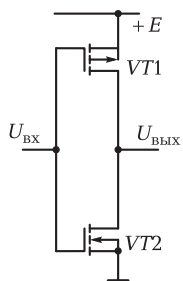


Рис. 2.7. Инвертор КМОП

Входной сигнал подается на соединенные затворы двух транзисторов, а выходной сигнал снимается со стоков, которые также соединены.

Если на вход элемента подается напряжение  $U_{ВХ} = +5$  В (логическая 1), то напряжение затвор — исток транзистора  $VT2$  составляет +5 В. Вследствие того что на его истоке и подложке нулевой потенциал, транзистор  $VT2$  будет открыт. Напряжение затвор — исток транзистора  $VT1$  в этом случае равно нулю, так как напряжение на его истоке и подложке составляет +5 В. Транзистор  $VT1$  будет закрыт. В этом случае выходное напряжение будет близко к нулю (логический 0).

При подаче на вход напряжения  $U_{ВХ} = 0$  (логический 0) имеем противоположную ситуацию. Транзистор  $VT2$  будет закрыт нулевым напряжением затвор — исток. Напряжение затвор — исток транзистора  $VT2$ , равное разности потенциалов затвора (на котором напряжение равно 0) и истока (на котором напряжение +5 В), в этом случае будет отрицательным и равным -5 В. Отрицательное напряжение затвор — исток открывает транзистор  $VT1$ , имеющий канал типа  $p$ . При открытом транзисторе  $VT1$  и закрытом транзисторе  $VT2$  получим выходное напряжение близкое к напряжению питания (логическая 1).

Применение полевых транзисторов с изолированным затвором обеспечивает высокое входное сопротивление микросхем КМОП. Вследствие высокого входного сопротивления микросхемы КМОП чувствительны к статическому электричеству. Пробой изоляции под затвором, в результате которого транзистор выходит из строя, происходит при напряжении около 30 В. Современные элементы типа КМОП имеют защиту входов от статического электричества, которая осуществляется с помощью диодов или стабилитронов, соединяющих входы с проводниками питания микросхемы.

В схеме инвертора один транзистор всегда открыт, а другой закрыт. В результате элемент в статическом режиме практически не потребляет ток. Только во время переключения потребляется ток от источника питания. При переключениях элемента из одного логического состояния в другое происходят процессы заряда и разряда межэлектродных емкостей транзисторов, которые сопровождаются потреблением тока от источника питания. Энергопотребление КМОП-элементов крайне низко, оно зависит в основном от частоты переключения.

Последовательное и параллельное соединение полевых транзисторов позволяет построить схемы для реализации логических функций. Схема логического элемента И — НЕ, выполненного по



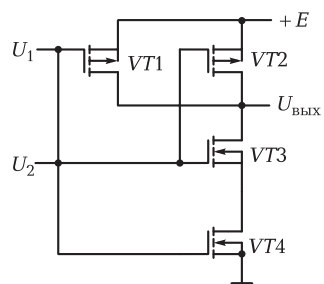


Рис. 2.8. Элемент И—НЕ серии КМОП

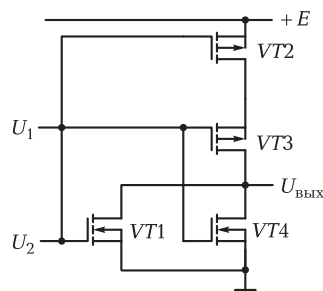


Рис. 2.9. Элемент ИЛИ—НЕ серии КМОП

технологии КМОП, состоит из двух групп транзисторов  $VT1$ ,  $VT4$  и  $VT2$ ,  $VT3$  (рис. 2.8). Каждая группа управляется одним входным сигналом  $U_1$  или  $U_2$ .

Сигнал логический 0 появляется на выходе схемы при единственном сочетании состояний транзисторов, когда оба нижних по схеме транзистора ( $VT3$ ,  $VT4$ ) открыты, а оба верхних ( $VT1$ ,  $VT2$ ) закрыты. Этим состояниям соответствуют входные сигналы  $U_1 = U_2 = +5$  В (логическая 1). При других комбинациях входных сигналов на выходе формируется сигнал логическая 1. Работа схемы соответствует логической функции И—НЕ.

Для схемы элемента ИЛИ—НЕ серии КМОП (рис. 2.9) характерной является ситуация, когда оба верхних транзистора открыты, а нижние закрыты и на выходе формируется логический сигнал 1. Для этого необходимо, чтобы все входные напряжения были равны нулю. Таким образом схема выполняет операцию ИЛИ—НЕ.

### 2.3. ХАРАКТЕРИСТИКИ И ПАРАМЕТРЫ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Основными параметрами логических элементов, которые необходимо учитывать при их использовании, являются напряжение источника питания, уровни напряжений логического 0 и логической 1, помехоустойчивость, нагрузочная способность, потребляемая мощность, быстродействие.

**Напряжение источника питания** логических элементов ТТЛ и ТТЛШ обычно составляет  $5 \text{ В} \pm 10 \%$ . Большая часть микросхем

на КМОП-структурах устойчиво работает при напряжении питания (3...15) В, некоторые — при напряжении  $9 \text{ В} \pm 10 \%$ . Большие интегральные схемы, разрабатываемые в настоящее время, рассчитаны на питающие напряжения меньшей величины: 3,3, 2, 1,8 В. Уменьшение питающего напряжения позволяет уменьшить потребляемую мощность и разместить на кристалле большее число элементов. Современные ПЛИС и микроконтроллеры могут быть рассчитаны на подключение нескольких источников питающих напряжений, например, источник 2 В для питания быстродействующего ядра, а источник 5 В для питания блоков ввода и вывода, совместимых с элементами ТТЛ.

**Уровни логических сигналов** определяют границы диапазонов напряжений, соответствующих сигналам 0 и 1. При выборе значений входных и выходных напряжений, соответствующих сигналам 0 и 1, необходимо учитывать, что нагрузками анализируемого логического элемента в большинстве случаев являются подобные ему элементы. Поэтому диапазоны напряжений, соответствующие логическим сигналам 0 и 1, должны быть одинаковы для входных и выходных сигналов и выбраны таким образом, чтобы, с одной стороны, выходной каскад элемента мог их сформировать, а с другой — входная цепь элемента однозначно их определила.

Сигналу 0 соответствуют напряжения на входе логического элемента в диапазоне от 0 до  $U_{\text{вх max}}^0$ , где  $U_{\text{вх max}}^0$  — максимальное входное напряжение, которое воспринимается элементом как сигнал 0.

Подобным образом сигналу 1 соответствуют напряжения на входе элемента, лежащие в диапазоне от  $U_{\text{вх min}}^1$  до напряжения питания  $E_{\text{пит}}$ , при этом  $U_{\text{вх min}}^1$  — это минимальное напряжение, которое, тем не менее, воспринимается элементом как сигнал 1.

Логические уровни сигналов выбираются по амплитудной передаточной характеристике таким образом, чтобы срабатывание логического элемента было надежным.

**Амплитудная передаточная характеристика** логического элемента отображает зависимость выходного напряжения элемента от входного напряжения  $U_{\text{вых}} = F(U_{\text{вх}})$ .

На рис. 2.10 приведены типовые передаточные характеристики для инвертирующих логических элементов типов ТТЛ и КМОП, например для инверторов.

Характеристики содержат участки, близкие к горизонтальным, соответствующие статическим состояниям, когда при входном сигнале низкого уровня на выходе формируется сигнал высокого уровня и наоборот. Также можно выделить участки, соответствующие



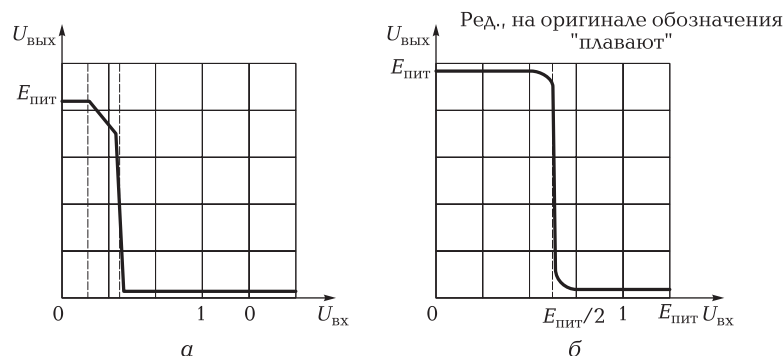


Рис. 2.10. Передаточные характеристики элементов ТТЛ (а) и КМОП (б)

переходу выходного напряжения от одного статического уровня к другому.

Для элементов ТТЛ при напряжении источника питания  $E_{\text{пит}} = 5$  В переход из одного статического состояния в другое происходит при входном напряжении порядка 1,3 В. В соответствии с этим значением выбираются пороговые напряжения:  $U_{\text{вх max}}^0 = 0,4$  В,  $U_{\text{вх min}}^1 = 2,4$  В. Это означает, что логическому сигналу 0 соответствуют напряжения в диапазоне (0...0,4) В, а логическому сигналу 1 — напряжения в диапазоне (2,4...5) В.

**Помехоустойчивость** элемента характеризуют максимальным напряжением помехи, которое не изменяет выходной сигнал. Выбранные уровни логических сигналов обеспечивают надежную работу устройства и высокую помехоустойчивость. Для элементов ТТЛ статическая помехоустойчивость составляет не менее 0,4 В. Это означает, что входной сигнал, отличающийся от  $U_{\text{вх max}}^0$  на 0,4 В и равный 0,8 В, будет восприниматься элементом как логический 0.

Передаточная характеристика логического элемента типа КМОП симметричная, переход из одного логического состояния в другое происходит при входном сигнале, соответствующем половине питающего напряжения. В то же время отклонения выходных напряжений от нулевого значения и напряжения источника питания незначительны и составляют всего нескольких десятков милливольт. При изменении питания передаточная характеристика изменяет масштаб. Передаточная характеристика элемента КМОП позволяет выбрать диапазоны сигналов для 0 и 1, обеспечивающие высокую помехозащищенность:  $U_{\text{пор}}^0 = 0,3E_{\text{пит}}$ ,  $U_{\text{вх min}}^1 = 0,7E_{\text{пит}}$ . Это означает,

что при напряжении питания  $E_{\text{пит}} = 5$  В логическому 0 соответствуют напряжения в диапазоне (0...1,5) В, а сигналу 1 — напряжения в диапазоне (3,5...5) В. Статическая помехоустойчивость для микросхем серий КМДП составляет не менее 30 % напряжения питания.

**Нагрузочную способность** логического элемента характеризует коэффициент разветвления по выходу, определяемый количеством входов однотипных элементов, которые можно подключить к выходу данного элемента. Этот коэффициент равен отношению максимального выходного тока логического элемента к входному току такого же элемента.

Коэффициент разветвления по выходу для базовых логических элементов серий ТТЛШ составляет 20, а для буферных элементов, имеющих более мощные выходные каскады — 40 и более.

В схемах на основе КМОП-транзисторов, имеющих высокое входное сопротивление, входы последующих схем в статическом режиме практически не нагружают выходов предыдущих. Это дает возможность иметь очень большой коэффициент разветвления по выходу, достигающий 100 и более, однако практически этот коэффициент ограничен из-за снижения быстродействия, обусловленного влиянием паразитных емкостей.

Достоинствами ИС КМОП являются малая потребляемая мощность и высокая помехозащищенность в сочетании с высоким быстродействием и нагрузочной способностью. Потребляемый ток в статическом режиме пренебрежимо мал, практически равен нулю.

**Динамические параметры** базовых элементов оценивают в первую очередь быстродействием. Количественно быстродействие можно характеризовать предельной рабочей частотой, т. е. максимальной частотой переключения триггера, выполненного на этих базовых элементах. Предельная рабочая частота микросхем «старого» типа ТТЛ серии К155 составляет 10 МГц, а микросхем серий К176 и К561 на КМОП-структурах — лишь 1 МГц. Микросхемы более поздних выпусков серии ТТЛШ 1533 имеют предельную рабочую частоту до 70 МГц, а КМОП серии 1554 — до 150 МГц.

Быстродействие элемента также характеризует задержка распространения сигнала. Сигнал на выходе логического элемента задерживается относительно входного сигнала. При этом задержка переднего фронта импульсного сигнала может отличаться от задержки заднего фронта, и в результате длительность импульса на входе оказывается отличной от длительности импульса на выходе. Среднее время задержки распространения сигнала является более

универсальным параметром микросхем, так как, зная его, можно рассчитать быстродействие любой сложной логической схемы суммированием средних значений времени задержки для всех последовательно включенных элементов. Для микросхем серии K1533 среднее время задержки составляет 10 нс, а для микросхем серии KP1554 — 7 нс.

**Надежность** элементов является важнейшим показателем микросхем. Ее характеризуют интенсивностью отказов. Средняя интенсивность отказов микросхем со средним уровнем интеграции составляет  $\lambda = 10^{-7}$  1/ч. Надежность цифровых устройств на микросхемах значительно превышает надежность аналоговых устройств на дискретных элементах.

## 2.4. ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ

Основой для построения цифровых устройств в настоящее время являются программируемые логические интегральные схемы (ПЛИС). Использование ПЛИС сокращает время проектирования, позволяет тестировать новые проекты в процессе разработки, а после изготовления изменять логику работы системы. Устройство, построенное на основе ПЛИС, имеет малые габаритные размеры и вес, низкую стоимость, высокую производительность.

Разработка и производство ПЛИС стали возможными благодаря совершенствованию процесса изготовления микросхем и уменьшению важного параметра — разрешения, определяемого минимальными значениями ширины линии и расстояния между линиями, которые реализованы в технологическом процессе.

Первые образцы ПЛИС изготавливались при разрешении приблизительно 2 мкм, что позволяло разместить на одном кристалле до 30000 транзисторов и обеспечивало работу при тактовой частоте до 10 МГц. В современных ПЛИС достигнуто уменьшение размеров элементов в 100 раз, а разрешение характеризует величина менее 20 нм. В результате максимальное количество транзисторов на кристалле исчисляется миллиардами, а предельные тактовые частоты возросли до нескольких гигагерц.

В настоящее время разработано большое количество ПЛИС различных типов. Простейшими являются ПЛИС типов PLA (Programmable Logic Array — программируемая логическая матрица) и GA (Gate Array — вентильная матрица или базовый матричный кристалл), предназначенные для построения в кристалле несложных

комбинационных схем. Кристалл ПЛИС данных типов содержит матрицы из элементов И и ИЛИ. Указанные типы ПЛИС, содержащие небольшое число ячеек, к настоящему времени морально устарели и применяются для реализации относительно простых устройств.

В настоящее время при проектировании цифровых устройств используют ПЛИС типов FPGA (Field Programmable Gate Array — программируемая матрица логических элементов) и CPLD (Complex Programmable Logic Device — сложное программируемое логическое устройство). Основными элементами кристалла ПЛИС указанных типов являются функциональные блоки различной сложности, состоящие из комбинационных схем, и элементы памяти.

Упрощенная структура ПЛИС типа FPGA представлена на рис. 2.11.

Основную часть площади кристалла занимают конфигурируемые логические блоки (КЛБ), или макроячейки, которые содержат различные элементы для построения цифровых схем — мультиплексоры, универсальные логические элементы, триггеры.

Соединение КЛБ между собой обеспечивают линии, коммутируемые транзисторными ключами (К). Ключи управляются сигналами от запоминающих элементов — триггеров, в которых хранятся данные о конфигурации микросхемы. Триггеры, предназначенные для хранения конфигурации, могут располагаться в непосредственной близости от элементов, которыми эти триггеры управляют. В ре-

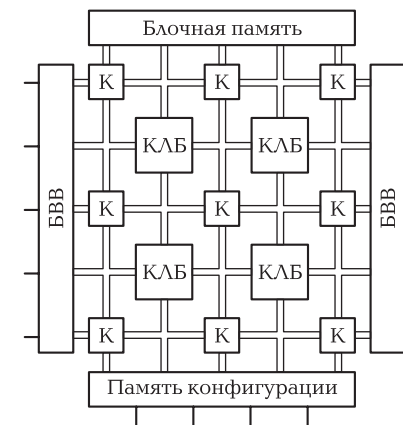


Рис. 2.11. Структура ПЛИС:

К — ключ; КЛБ — конфигурируемые логические блоки; БВВ — блок ввода-вывода

зультате память конфигурации оказывается распределенной по всему кристаллу. Зачастую ее не отображают на структурных схемах и называют «теневое запоминающее устройство».

По периферии кристалла расположены блоки ввода-вывода (БВВ), к которым подключены внешние выводы микросхемы. Блоки ввода-вывода можно программировать на ввод или вывод сигналов.

Для программирования ПЛИС необходимо записать файл конфигурации, называемый также файлом прошивки, в память конфигурации. Для записи этого файла предназначены специальные выводы на корпусе микросхемы. Этот файл формируется как результат разработки желаемого устройства с использованием специального комплекса программ, который называется *система автоматизированного проектирования* (САПР).

Коды, записанные в память конфигурации, поступают на все элементы, расположенные на кристалле, и управляют взаимосвязями между элементами, а также структурами и параметрами этих элементов. В результате программирования ПЛИС превращается в аппаратную реализацию разработанного устройства, которое обычно называют *проект*.

Основными параметрами ПЛИС являются функциональные возможности кристалла, а также тактовая частота, логическая емкость, количество выводов. Для оценки логической емкости ПЛИС используют число эквивалентных вентилях, определяемое как среднее число элементов И—НЕ, необходимых для реализации такого же проекта, какой может быть реализован на анализируемой ПЛИС.

Фирма *Altera*, один из лидеров по производству ПЛИС, выпускает несколько серий или семейств микросхем, различающихся сложностью и ценой. Рассмотрим некоторые семейства.

**Семейство MAX3000** содержит несколько наиболее дешевых ПЛИС, предназначенных для реализации сравнительно простых цифровых устройств, несложных цифровых систем управления и контроля для различных объектов.

Число логических эквивалентных вентилях ПЛИС этой серии находится в диапазоне 600 ... 5000, общее количество выводов 44 ... 208. Микросхемы этой серии могут быть многократно запрограммированы с помощью простого программатора непосредственно в схеме. Память конфигурации в микросхемах этого семейства изготовлена по технологии *EEPROM* (запоминающее устройство с электрическим стиранием), что является несомненным достоинством. Конфигурация ПЛИС (прошивка), записанная в эту память, сохраняется при отключении питания.

**Семейство FLEX10K** является в настоящее время популярной элементной базой для построения сложных устройств обработки данных. Число логических эквивалентных вентилях ПЛИС этой серии находится в диапазоне 10 000 ... 25 000, а общее количество выводов 84 ... 386. Выпускаются ПЛИС с различными значениями напряжения питания: 5, 3,3 и 2,5 В. Кроме того, ПЛИС этого семейства имеют встроенные блоки памяти емкостью 2048 или 4096 бит.

**Семейство APEX20K** включает в себя микросхемы очень высокой степени интеграции и предназначено для построения вычислительных систем на кристалле, в которых на одном кристалле содержатся не только матрица из функциональных блоков, но и микропроцессорная система. Микросхемы типа система на кристалле (*System On Chip — SOC*) содержат на кристалле матрицу КЛБ, подобно *FPGA*, а также узлы ЭВМ — модули памяти, процессорные ядра, микроконтроллеры. Системы на кристалле используют в сложных современных цифровых устройствах — компьютерах, смартфонах, цифровых фотоаппаратах.

Микросхемы **семейства APEX20K** имеют корпуса с общим количеством выводов от 144 до 984. Напряжение питания может быть равным 1,8 В и 2,5 В. Для совместимости с устройствами, имеющими уровни логических сигналов 3 В или 5 В, на дополнительный вывод питания должно быть подано напряжение 3,3 В. Двухнаправленный ввод-вывод обеспечивается на частотах до 370 МГц. Каждый из пользовательских выводов может быть индивидуально переведен в отключенное состояние. Двухнаправленный ввод-вывод обеспечивается на частотах до 250 МГц.

**Семейство Stratix V** является одной из разработок нового поколения СБИС фирмы *Altera*. Это семейство включает в себя микросхемы самой большой на сегодняшний день логической емкости, около 1 000 000 эквивалентных логических элементов. Микросхемы выполнены по технологии 28-нм., отличаются высоким быстродействием, высокой степенью интеграции и сниженным на 30 % энергопотреблением по сравнению с ПЛИС предыдущих поколений.

Микросхемы этого семейства содержат аппаратные блоки обработки, хранения и передачи данных: блоки встроенной памяти, контроллеры интерфейсов и внешней памяти, аппаратные блоки цифровой обработки сигналов, устройства реконфигурации кристалла. Предназначены для построения сложных систем обработки данных в реальном времени с высоким быстродействием, например, систем обработки видеосигналов.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимают под поколениями ЭВМ?
2. Что такое серия интегральных микросхем?
3. Назовите буквенно-цифровые элементы, определяющие условные обозначения микросхемы.
4. Назовите основные технические характеристики цифровых микросхем.
5. Охарактеризуйте свойства логических элементов ТТЛ и КМОП.
6. Поясните назначение логических элементов с открытым коллектором.
7. Опишите работу логических элементов с тремя состояниями выхода.
8. Опишите достоинства элементов типа КМОП.
9. Укажите преимущества проектирования устройств на основе ПЛИС.
10. Укажите основные элементы кристалла ПЛИС.
11. Перечислите основные параметры ПЛИС.

## Глава 3

### СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ MAX + PLUS II

Для программирования ПЛИС, выпускаемых фирмой *Altera*, предназначена система *MAX + plus II (Multiple Array Matrix Programmable Logic User)* — пользовательская система программирования упорядоченных структур логики. Система *MAX + plus II* предназначена для автоматизации процесса проектирования, оптимизации и тестирования устройства, размещения элементов на кристалле ПЛИС, программирования ПЛИС. За реализацию этих функций отвечают различные модули программы.

После запуска программы (файл *max2win.exe*) открывается окно менеджера проекта *MAX + plus II Manage*, отображающее структуру системы проектирования, которая состоит из 11 интегрированных модулей, предназначенных для полного проектирования устройства от ввода исходных данных и контроля описания до программирования микросхемы ПЛИС. Вызов любого модуля системы осуществляется из пункта меню *MAX + plus II* (рис. 3.1).

Система *MAX + plus II* позволяет выполнить моделирование и тестирование изучаемых устройств. Ее рекомендуется установить на компьютер и использовать по мере изучения курса. Моделирование проектируемых устройств в САПР позволит более полно изучить их функционирование. Кроме того, работа в системе *MAX + plus II* позволит освоить методы проектирования цифровых устройств на базе ПЛИС.

Система *MAX + plus II* позволяет также выполнить программирование кристалла ПЛИС. Однако для выполнения данной функции необходимо учитывать большое число параметров, характеризующих конкретную аппаратную реализацию устройства, что выходит за рамки курса.

Разрабатываемое в САПР устройство называют *проект*. Каждому проекту соответствует несколько файлов с одинаковыми имена-

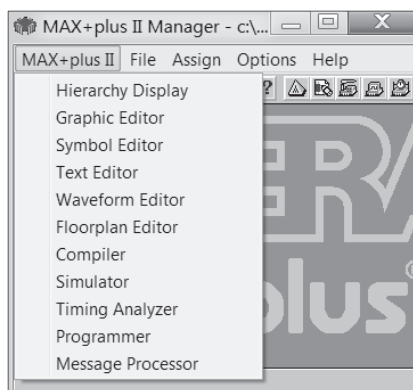


Рис. 3.1. Окно менеджера проектов системы проектирования MAX + plus II

ми и различными расширениями. Ввод исходных данных может быть выполнен несколькими способами: в виде схемы, в виде описания или в виде временных диаграмм.

### 3.1. ПРОЕКТИРОВАНИЕ УСТРОЙСТВА, ЗАДАННОГО В ВИДЕ СХЕМЫ

Этапы разработки проекта по схеме. Создание проекта с вводом исходных данных в виде схемы содержит следующие этапы.

1. Создание нового графического файла — *File/New* (*Graphic Editor file*);  
*File/Save As* (имя. *gdf*).  
Открытие существующего файла — *File/Open*.
2. Ввод схемы. Ввод элементов — двойной щелчок на схеме — окно *Enter Symbol*, выбор библиотеки (*//prim*), выбор элемента. Соединение элементов схемы проводниками или присваиванием одинаковых имен.
3. Установка ведущего файла — *File/Project/Set Project To Current File*. Команды *File/Open* и *Set Project To Current File* позволяют запускать ранее созданные проекты для продолжения работы.
4. Компиляция — *MAX + Plus II/Compiler*.
5. Создание символа — активизировать окно со схемой, вызвать команду *File/Create Default Symbol* (в простых проектах может отсутствовать).

6. Запуск сигнального редактора. *MAX + Plus II / Waveform Editor*.
7. Запись файла временных диаграмм с именем проекта — *Save As*.
8. Выбор контрольных точек. Из меню по правой кнопке — *Enter Nodes from SNF*.
9. Ввод тестовых входных сигналов. Панель редактирования сигналов.
10. Моделирование. Запуск имитатора — *MAX + Plus II/simulator*.

Рассмотрим пример исследования простейшей логической схемы. Рекомендуется выполнять изучаемые действия в системе *MAX + Plus II*. Для файлов проекта необходимо создать отдельный каталог, имя которого и путь к нему не содержат русских букв.

Задано разработать проект комбинационной схемы *ks1* в соответствии с рис. 3.2. Выполнить моделирование и рассмотреть выходные сигналы каждого элемента. Составить подробное описание работы схемы.

Создание нового графического файла. Для создания нового графического файла проекта выбирается команда из главного меню *File/New*, и в окне *New* выбирается тип файла — графический (расширение *.gdf*). В результате будет запущен графический редактор и создан файл без имени (*Untitled\_1*).

Открыть уже существующий файл позволяет команда *File/Open*. В открывшемся окне следует указать тип файла.

Для нового файла необходимо выбрать уникальное, не повторяющееся в других проектах имя (пробелы в именах недопустимы). Для проектов, представленных в виде схемы, будем использовать имена, начинающиеся с буквы *s*, после которой указан номер рисунка со схемой, а после символа подчеркивания обозначено функциональное назначение схемы. Такая система обозначения проектов исключит повторение имен и облегчит поиск файлов в общем каталоге.

Для графического файла, содержащего схему *ks1*, изображенную на рис. 3.2, выберем имя *s32\_ks1* и запишем данный файл командой *File/Save As* в созданный ранее каталог. Выбранное имя будут иметь все файлы данного проекта, а также символ, созданный впоследствии.

Ввод схемы. Ввод схемы содержит ввод элементов и их соединение.

**Ввод элементов** — это размещение на схеме символов элементов или условных графических

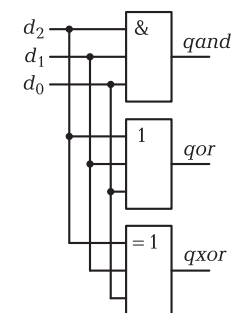


Рис. 3.2. Комбинационная схема *ks1*



отображений, которые содержатся в библиотеках. Для ввода символа необходимо двойным щелчком левой клавиши на свободном месте схемы открыть окно *Enter Symbol*, в котором показаны доступные библиотеки *Symbol Libraries*, выбрать библиотеку (например, *prim*), выбрать символ в списке *Symbol Files*, нажать *OK*. Можно ввести имя символа в строке *Symbol Name* окна *Enter Symbol*.

Библиотека примитивов (*..\prim*) содержит простейшие базовые логические элементы цифровых устройств, а также терминалы (контакты) для сигналов ввода и вывода.

Библиотека макрофункций (*..\mf*) содержит элементы средней степени интеграции. Библиотека (*..\lpm*) содержит модули блоки высокой степени интеграции с возможностью изменения параметров. Библиотека (*..\edif*) содержит модели логических элементов старого типа ТТЛ, применение которых в настоящее время ограничено. Описание всех элементов приведено в *Help*.

Для ввода символов разных типов требуется многократно открывать окно *Enter Symbol*. Если схема содержит несколько однотипных элементов, то из библиотеки следует ввести только один символ, а затем использовать копирование и вставку (комбинация клавиш *Ctrl + C* и *Ctrl + V*). Для копирования элемента также можно поместить курсор на тело элемента и нажать левую клавишу манипулятора, выбранный элемент будет иметь красный контур. Не отпуская левую кнопку, нажать на клавиатуре клавишу **Ctrl**, при этом около выбранного элемента появится символ «+». Не отпуская нажатые клавиши, переместить копию элемента в нужную зону схемы. Отпустить нажатые клавиши. Данный способ особенно удобен при вводе терминалов ввода или вывода.

Все элементы для построения заданной схемы содержатся в библиотеке *prim*. Это терминалы *Input* и *Output*, 3-входные элементы *and3*, *or3*. Элемент Исключающее ИЛИ в библиотеке имеется только с двумя входами — это элемент *xor*.

Для ввода имени терминала необходимо поместить курсор на текст *PIN\_NAME*, выбрать эту зону двойным щелчком левой клавиши мыши, ввести с клавиатуры имя терминала (старое имя будет автоматически стерто), нажать клавишу *Enter* на клавиатуре. Если существует другой терминал, расположенный ниже, то у него будет автоматически выбрана зона имени для ввода.

**Соединение элементов** в графическом редакторе может быть физическим или логическим.

**Физическое соединение** элементов выполняется посредством проводников, или шин. Для физического соединения выводов символов необходимо подвести курсор к выводу, нажать левую клави-

шу манипулятора и, удерживая ее, провести линию до другого, соединяемого вывода. Курсор в этом случае принимает вид крестика. Проводники изображаются тонкой линией, а шины — толстой, тип линии устанавливается выбором из меню *Options/Line Style*. Для фиксации положения проводника на схеме необходимо на короткое время отпустить левую клавишу.

**Логическое соединение** проводников, или шин выполняется присваиванием одинаковых имен. Для ввода имени необходимо выбрать цепь (проводник, или шину), поместить курсор на тело цепи и щелкнуть левой клавишей манипулятора. Выбранная цепь будет иметь красный цвет, а курсор примет вид мигающего черного квадрата, что является приглашением к вводу текста. С клавиатуры необходимо ввести имя цепи и нажать клавишу **Enter**.

**Имя нельзя присвоить выводу элемента**, к которому не подключена никакая цепь. Обязательно нужно изобразить хотя бы короткий проводник, или шину, которым можно присвоить имя.

Для передачи параллельных двоичных кодов используют шины. **Обозначение шины** после имени содержит индексы старшего и младшего разрядов, разделенные двумя точками, в квадратных скобках, например:  $d[2..0]$  — обозначение 3-разрядной шины, которая содержит проводники:  $d_2, d_1, d_0$ .

Для ввода и редактирования схем предназначены инструменты панели, которая расположена в левой части экрана графического редактора (рис. 3.3). Назначение кнопки, выбранной указателем, указывается на английском языке в нижней части экрана в строке состояния.

Выбор объекта 1 (элемента или фрагмента схемы) служит для его перемещения и копирования. Это основной режим при вводе и редактировании схем. Переход в этот режим из любого другого выполняет клавиша **Ecs**. В этом режиме можно выделять символы, проводники или фрагменты схемы при нажатой левой клавиши мыши, а затем перемещать, удалять — **Delete**, копировать — **Ctrl + C** и вставлять — **Ctrl + V** выделенные элементы. Правая клавиша мыши открывает контекстное меню функций, доступных для выделенного элемента.

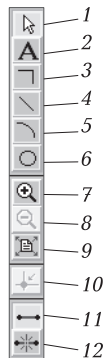


Рис. 3.3. Панель инструментов редактирования схем:

1 — выбор объекта; 2 — ввод поясняющего текста; 3 — ввод ортогональных линий; 4 — ввод линий с произвольным углом; 5, 6 — ввод дуг и окружностей соответственно; 7, 8 — изменение масштаба изображения; 9 — отображение всего листа схемы; 10 — изменение статуса точки пересечения цепей; 11, 12 — включение и выключение режима



Проводники изображаются тонкой непрерывной линией, а шины — широкой линией. Тип линии выбирают из списка на панели инструментов *Options/Lint Style*.

При вводе схем рекомендуется выбирать предельно крупный масштаб, позволяющий контролировать соединения. Уменьшение масштаба используется при изменении размещения элементов.

Для изменения статуса точки пересечения цепей на противоположный 10 сначала указателем манипулятора выделяют точку пересечения проводников. Каждое нажатие кнопки устанавливает или удаляет точку соединения пересекающихся проводников. При пересечении проводников без соединения точка не ставится, а при соединении проводников ставится точка.

Резиновые проводники (*Rubberbanding*) 11, 12 — включение или выключение режима — обеспечивает неразрывность цепей (шин) при перемещениях элементов. Режим, включаемый кнопкой 12, позволяет «разобрать» схему на отдельные элементы и построить из них новую схему.

Заданная схема, введенная в графическом редакторе, приведена на рис. 3.4. Место расположения логических элементов и терминалов на схеме значения не имеет. Компактное размещение позволяет увеличить масштаб при печати схем.

Входные сигналы схемы представлены в виде 3-разрядной шины, для которой на схеме изображен терминал *INPUT*. Обозначение терминала содержит имя шины и диапазон индексов в квадратных скобках: *d[2..0]*. Индекс — это номер проводника в шине. Индекс младшего разряда кода обычно принимают равным нулю. Подключение входов элементов к шине выполнено как логическое. К каж-

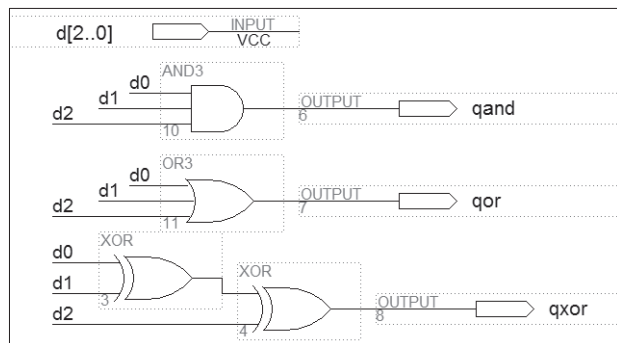


Рис. 3.4. Схема s32\_ks1 в графическом редакторе

дому входу элемента подключен короткий проводник, имеющий имя. Ввод имени осуществляется после изображения проводника и появления мигающего квадрата. На схеме показан способ обозначения проводников, входящих в шину, имя проводника состоит из имени шины и индекса, записанных без пробела. Проводники, имеющие имена *d0*, *d1*, *d2*, считаются подключенными с соответствующим разрядом шины с именем *d*.

При вводе заданной схемы выясняется, что 3-входовый элемент *xor* в библиотеке *prim* отсутствует, поэтому требуемая операция реализована на 2-входовых элементах.

Установка ведущего файла. После ввода и редактирования схемы необходимо запустить команду *File/Project/Set Project To Current File* (установить проект по текущему файлу). Эта команда указывает компилятору системы проектирования имя проекта, с которым необходимо работать. Это имя отобразится в титульной (самой верхней) строке окна *Manager*.

Компиляция. *Компилятор* — программа, которая проверяет правильность ввода данных проекта, приводит к единообразному представлению различные формы описания, размещает разрабатываемое устройство на ячейках определенной микросхемы. Компилятор запускается командой *MAX + plus II/Compiler*, затем следует нажать кнопку *Start*.

По результату компиляции выдается сообщение о возможных ошибках. Двойной щелчок по строке *Error* позволяет выделить на схеме место ошибки.

Создание символа. После удачной компиляции можно создать символ. Для этого необходимо активизировать (открыть) окно графического редактора, в котором изображена схема, и вызвать команду *File/Create Default Symbol*. Имя символа будет совпадать с именем проекта и появится в списке *Symbol Files* окна *Enter Symbol*. Если активно какое-либо из служебных окон, то команда *File/Create Default Symbol* будет недоступна.

Вставляя символ в другие схемы, можно использовать разработанное устройство в других проектах, структуру которых называют *иерархической*.

Команды *File/Open* и *Set Project To Current File* позволяют запускать ранее созданные проекты для продолжения работы.

Запуск сигнального редактора. Сигнальный редактор предназначен для тестирования устройств. Он позволяет создать временные диаграммы входных сигналов устройства, которые называются **тестовые векторы**, и получить временные диаграммы выходных сигналов, являющиеся результатом моделирования схемы.

Сигнальный редактор запускается командой: *MAX + Plus II/Waveform Editor*.

Запись файла временных диаграмм с именем проекта. Открывшийся после запуска сигнального редактора безымянный файл необходимо сохранить командой *Save As*, при этом имя проекта и расширение \*.scf установятся автоматически. В окне сигнального редактора будет создано окно для отображения временных диаграмм.

**Выбор контрольных точек.** Необходимо установить курсор в столбце *Name* временных диаграмм, нажать правую клавишу мыши и в открывшемся меню выбрать *Enter Nodes from SNF*. В результате откроется окно для выбора контрольных точек при моделировании, в котором следует нажать кнопку *List*. В левой части окна появится список доступных узлов и групп, для которых на схеме изображены терминалы (*Available Nodes & Groups*). Из этого списка можно выбрать требуемые узлы либо переслать все узлы из левого списка в правый, нажав кнопку со стрелкой, расположенную между списками, и нажать кнопку *OK*.

В результате в окне сигнального редактора будут созданы временные диаграммы (рис. 3.5). Левый столбец *Name* содержит обозначения терминалов и имена сигналов. Входной терминал изображается как гнездо и обозначен буквой *I* (*Input*), а выходной терминал изображен как штекер с буквой *O* (*Output*). Для терминала шины указан диапазон разрядов. Расположение временных диаграмм в окне сигнального редактора можно изменять, установ-

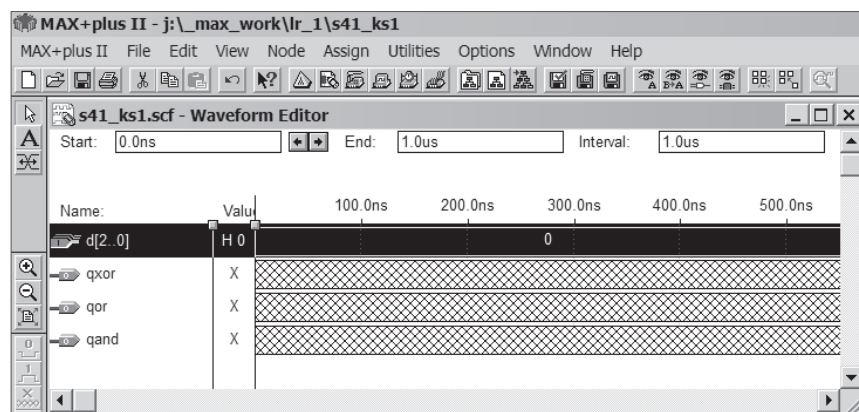
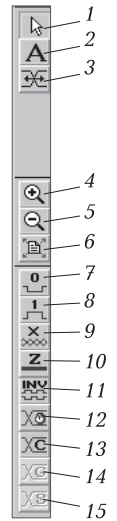


Рис. 3.5. Окно редактора временных диаграмм

Рис. 3.6. Панель редактирования сигналов:

1 — выбор временно#й диаграммы; 2 — ввод поясняющего текста; 3 — перемещение фронтов импульсов; 4, 5 — изменение масштаба временно#х диаграмм; 6 — просмотр диаграмм во весь экран; 7—10 — установка статических уровней (0, 1, X, Z); 11 — инвертирование выделенного сигнала; 12 — подача синхросигнала; 13 — подача на проводник или шину импульса; 14 — подача фиксированного кода на шину; 15 — ввод имени состояний сигнала



ливая указатель манипулятора на символ порта и перемещая его при нажатой левой клавише. В столбце *Value* отображены значения сигналов. В созданном окне первоначально отображаются неопределенные значения выходных сигналов, равные *x*.

**Ввод тестовых входных сигналов.** Важным моментом при моделировании является создание тестовых сигналов, которые должны обеспечить по возможности полную проверку правильности функционирования устройства и получение вполне определенного предсказуемого результата, основанного на знании работы устройства.

Рассматриваемая в примере логическая схема имеет равнозначные входы, свойства которых одинаковы. Для ее тестирования необходимо выполнить полный перебор всех возможных комбинаций входных сигналов. Для двоичного кода, имеющего три разряда, количество комбинаций равно 8. Для ввода и редактирования сигналов предназначены инструменты, показанные на рис. 3.6.

Выбор временной диаграммы или ее части для перемещения, копирования, удаления или установки определенного логического уровня сигнала осуществляется кнопкой 1. Переход в этот режим выполняет клавиша *Esc*.

Перемещение фронтов импульсов курсором манипулятора возможно при нажатой левой клавише. В инвертирование выделенной главы временной диаграммы. Режим, включаемый кнопкой 3, позволяет формировать (рисовать) импульсы курсором мыши при нажатой левой клавише мыши.

Подача синхросигнала в виде прямоугольных импульсов с программируемыми параметрами на выделенную диаграмму или ее часть осуществляется кнопкой 12.

Для формирования временных диаграмм для входных сигналов необходимо щелчком левой клавиши мыши выделить шину, или проводник установив курсор на терминал, а затем нажать кнопку, соответствующую требуемому сигналу.

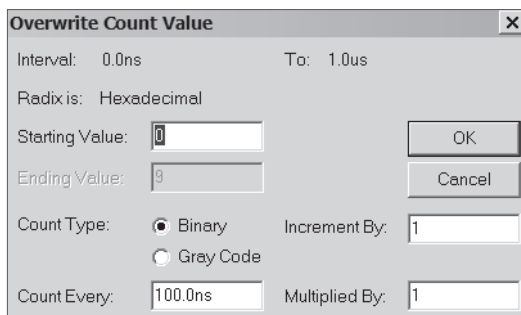


Рис. 3.7. Окно параметров импульсов

Если выбран терминал, к которому подключен проводник, то становятся активными только те кнопки, которые можно использовать в этом случае, как показано на [рис. 3.6](#). Они позволяют подать на проводник постоянные логические сигналы (кнопками 7 или 8), либо импульсы (кнопками 12 или 13).

Если выбран терминал шины, то активны кнопки 13 и 14 для подачи на шину импульсов или постоянного кода. Сигнал в форме прямоугольных импульсов выбирается кнопкой 13, а параметры импульсов устанавливаются в окне *Overwrite Count Value* ([рис. 3.7](#)), где можно задать: начальное значение логического сигнала — *Starting Value*, конечное значение — *Ending Value*, приращение кода для шины — *Increment By*, а также масштабный множитель *Multiplied By*, увеличивающий масштаб по оси времени. Если в схеме несколько шин, то для первой из них можно выбрать параметры по умолчанию, а для других выбрать больший масштабный множитель для увеличения периода импульсов.

Параметр *Interval*, по умолчанию равный 1 мкс, определяет длительность моделирования. Его можно изменить, выбирая пункт меню *File/End Time*. Параметр *Count Every* определяет период тактовых импульсов счетчика.

Двойной щелчок по символу терминала шины открывает окно для выбора системы счисления, в которой отображаются коды на данной шине. Правая кнопка открывает для выделенной шины контекстное меню, позволяющее отобразить сигналы на шине в виде диаграмм для отдельных проводников, разгруппировать шину (функция *Ungroup*). Для возвращения к отображению состояний шины в виде кодов необходимо выделить проводники шины и выбрать функцию *Enter Group*.

В рассматриваемом примере необходимо подать на шину *d* коды от счетчика. Для этого необходимо выделить терминал шины (см. [рис. 3.5](#)), выбрать кнопку 13 (см. [рис. 3.6](#)) и в открывшемся окне (см. [рис. 3.7](#)) нажать *OK* для выбора всех параметров по умолчанию. За полный цикл счетчика происходит перебор всех возможных комбинаций двоичных кодов.

**Запуск имитатора.** Последний этап моделирования — запуск имитатора из меню *MAX + plus II/Simulator* или кнопкой на панели инструментов. В результате моделирования появятся временные диаграммы для выходных сигналов.

По результатам моделирования можно проверить правильность функционирования и работоспособность устройства, а также описать логику работы.

Для рассматриваемого примера временные диаграммы для выходных сигналов, полученные в результате моделирования, соответствуют заданным входным сигналам и теоретическим временным диаграммам, что указывает на правильность функционирования схемы.

На выходе элемента И сигнал равен 1, если на все входы поданы сигналы, равные 1. На выходе элемента ИЛИ сигнал равен 0, если на все входы поданы сигналы 0. На выходе элемента Сумма по модулю 2 сигнал равен 1, если сумма сигналов, равных 1, на всех входах нечетна.

Для того чтобы один из входов элемента (например,  $d_0$ ) был активным и передавал изменения сигнала на выход, на остальные входы необходимо подать: для элемента И — 1; для элемента ИЛИ — 0; для элемента Сумма по модулю 2 — любое значение. Элементы И и ИЛИ можно использовать в схемах стробирования сигналов, а элемент Сумма по модулю 2 в качестве управляемого инвертора.

## 3.2. ОПИСАНИЕ АППАРАТНЫХ СРЕДСТВ НА ЯЗЫКЕ VERILOG HDL

Языки описания аппаратных средств (*Hardware Description Language* — HDL) позволяют использовать описание проектируемого цифрового устройства как исходные данные, для системы автоматизированного проектирования, запрограммировать ПЛИС и получить аппаратную реализацию этого проекта. В настоящее время при разработке новых устройств вместо того, чтобы разрабатывать схему и вводить ее в графическом редакторе, используют ее описание. Описание проектов на HDL-языках — эффективный

способ подготовки исходных данных при проектировании устройств на элементной базе ПЛИС, который может быть использован на всех этапах разработки цифровых устройств: при проектировании, моделировании и тестировании аппаратуры.

Современные системы автоматизированного проектирования устройств на ПЛИС позволяют использовать в качестве средств описания проектов специализированные и универсальные языки описания аппаратных средств. Специализированные языки предназначены для проектирования устройств в ПЛИС определенной фирмы (например, язык *AHDL (Altera HDL)* позволяет проектировать устройства для ПЛИС фирмы Altera). Универсальные языки позволяют разрабатывать проекты для ПЛИС различных фирм. Такими языками являются *VHDL (Very high-speed IC Hardware Description Language)* — язык для описания быстродействующих аппаратных средств) и *Verilog HDL*.

Существует несколько принципиальных отличий языков описания аппаратных средств от традиционных языков программирования.

Первое отличие состоит в том, что традиционные языки программирования позволяют писать программы для вычислений, моделирования, управления различными системами, а *HDL*-языки предназначены для составления описания, по которому компьютер, на котором установлена САПР, позволяет запрограммировать ПЛИС и получить в этой микросхеме желаемое устройство или, как говорят, его аппаратную реализацию. В настоящее время разработано большое количество описаний различных цифровых устройств. Наиболее интересны описания процессоров, позволяющие получить в ПЛИС аппаратную реализацию процессора для решения определенного круга задач.

Вторая характерная особенность *HDL*-языков заключается в том, что они описывают процессы, которые происходят в цифровой схеме параллельно. Сигналы в разных точках цифровой схемы могут изменяться одновременно, и эта особенность отображается средствами языка.

Также следует отметить, что переменные, с которыми работают *HDL*-языки, имеют вполне определенный физический смысл, они являются сигналами, которые действуют в цифровой схеме.

**Этапы разработки проекта по описанию.** Для ввода описания и разработки проекта необходимо выполнить следующие действия, подобные рассмотренным ранее.

1. Создание нового файла — *File/New (Text Editor File)*, *File/Save As* (имя. v), или открытие существующего *File/Open*. Имя файла

обязательно должно совпадать с именем модуля, указанным в описании, и иметь расширение .v.

2. Ввод описания или вставка из другого текстового редактора.
3. Установка ведущего файла — *File/Project/Set Project To Current File*.
4. Компиляция — *MAX + plus II/Compiler*.
5. Создание символа — активизировать окно с описанием, вызвать команду *File/Create Default Symbol*.
6. Запуск сигнального редактора — *MAX + Plus II/Waveform Editor*.
7. Запись файла диаграмм с именем проекта — *Save As*.
8. Выбор контрольных точек — из меню по правой кнопке *Enter Nodes from SNF*.
9. Ввод тестовых входных сигналов. Панель редактирования сигналов.
10. Моделирование. Запуск имитатора — *MAX + plus II/simulator*.

Имена проектов, для ввода которых использованы примеры описаний на языке *Verilog*, будут начинаться с буквы v, после которой указан номер примера, содержащего данное описание.

Далее будут рассмотрены основы универсального языка описания аппаратных средств *Verilog HDL* (для краткости — *Verilog*) и применение этого языка при проектировании цифровых устройств в САПР *MAX + PlusII*.

**Основы синтаксиса языка Verilog. Алфавит языка** — набор символов, допустимый в описаниях. Этот набор содержит латинские буквы, цифры и специальные символы.

Регистр имеет значение!

Для ввода описаний обычно устанавливают нижний регистр и впоследствии его не переключают.

**Комментарий** — строка с пояснениями, начинающаяся с двух символов косая черта //. Комментарии не влияют на результат компиляции описания. В комментариях можно использовать русские буквы и любые символы. Фрагмент в скобках /\* \*/ — также комментарий. Такие скобки можно использовать при отладке для временного исключения фрагментов описания из компиляции.

Описания, как правило, начинаются со строк комментария.

**Имя модуля или сигнала** (другое название — идентификатор) должно начинаться с буквы или символа подчеркивания; оно может содержать цифры, знаки подчеркивания и доллара. Имя сигнала выбирает пользователь, в простых схемах обычно используют короткие имена, а в сложных — сочетания букв, определяющие назначение сигнала. В имени сигнала недопустимы пробелы. Если требуется в имени разделить группы букв, имеющие



определенные смысловые значения, то используют знак подчеркивания.

Пример имен сигналов: *a*, *b*, *c*, *d*, *q*; пример имени модуля: *log\_schema\_var1*.

**Константы** в языке *Verilog* можно записать в различных системах счисления. Полная запись константы содержит четыре элемента. Первый из них — число, определяющее количество разрядов в константе. Второй элемент — одинарная кавычка. Третий элемент — буква, определяющая основание системы счисления: *b* — для двоичной системы, *o* — для восьмеричной, *d* — для десятичной, *h* — для шестнадцатеричной. Четвертым элементом записи являются цифры в указанной системе счисления. Отрицательные числа задаются знаком «минус» в самом начале записи перед разрядностью числа. В двоичной системе допускаются символы *z* и *x*. Для записи констант в описаниях сложных устройств отдают предпочтение двоичной системе, которая наглядно отображает все разряды числа.

Краткая запись константы содержит только число. Это соответствует по умолчанию десятичной системе счисления. Запись констант в десятичной системе проще воспринимается и используется при описании простых схем.

Например, запись *1'b1* соответствует единице в двоичной системе, а запись *1* — единице, записанной в десятичной системе. Первая запись, в которой явно указана разрядность, более строгая, ее применяют в сложных описаниях. Рассмотрим запись константы 5. Запись в двоичной системе имеет вид: *3'b101*, а в десятичной — 5.

**Сигналы языка Verilog.** Все переменные языка *Verilog* (аргументы и функции) имеют вполне определенный физический смысл, они являются сигналами.

Сигналы могут принимать одно из четырех значений: 0, 1, *z*, *x*. Значение *z* формируют элементы с теми состояниями выхода, когда источник сигнала отключен, а значение сигнала *x* означает, что сигнал не определен и может принимать любое значение 0 или 1. Чаще всего значения сигнала — это 0 и 1.

Сигналы в схеме бывают двух типов: цепь (*wire* — провод) и регистр (*reg* — регистр). **Сигнал типа цепь** — *wire* моделирует провод, к которому непрерывно прилагается воздействие от источника сигнала, который называют драйвер (*driver*). По умолчанию все сигналы в описаниях устанавливаются типа *wire*. Выходные сигналы комбинационных схем имеют тип *wire*.

**Сигнал типа регистр** — *reg* моделирует элемент памяти. Тип *reg* необходимо указать для выходных сигналов схем с элементами памяти.

Сигналы имеют различную разрядность. Если в описании разрядность не указана, то по умолчанию принимается сигнал в 1 бит.

Код, состоящий из нескольких разрядов, который параллельно передается через группу проводников, описывают как вектор или шину.

В описании вектора перед его именем в квадратных скобках через двоеточие указывают диапазон индексов, нумерующих разряды или проводники шины. Первый индекс должен соответствовать старшему разряду вектора, а индекс младшего разряда должен быть равен 0. При такой нумерации вес разряда и индекс будут совпадать. Например, описание 4-разрядной шины с именем *d* имеет вид: *[3:0] d*.

Какой-либо из проводников шины называют **элементом вектора**. Для использования в описании одного проводника шины (элемента вектора) необходимо указать имя вектора, а затем индекс проводника в квадратных скобках. Так, например, запись *d[3]* означает сигнал третьего проводника шины *d*.

По умолчанию вектор — беззнаковое скалярное (*scalared*) целое, в котором разрешен доступ к отдельным битам. Если перед указанием диапазона записать *signed*, то будем иметь число в дополнительном коде со знаком.

**Параллельные операторы языка Verilog.** **Параллельные операторы** используются для описаний комбинационных схем. Они начинаются с ключевого слова *assign*, что означает — назначать. Все сигналы параллельных операторов, входные и выходные, имеют тип *wire*, который не требуется указывать в описании, так как этот тип устанавливается по умолчанию.

Параллельные операторы работают непрерывно, изменение выходного сигнала происходит при изменении любого из входных. Такой режим работы обеспечивает возможность описания процессов, протекающих в схемах параллельно.

После ключевого слова *assign* допускается запись всех арифметических и логических операторов, а также объединения и условного присваивания. Окончание всех операторов обозначает разделитель — точка с запятой. Символы пробела, табуляции, возврата каретки транслятор игнорирует, если они не нарушают целостность ключевых слов.

**Арифметические операторы** имеют два аргумента, которые перечислены в табл. 3.1 в порядке убывания приоритета. Порядок операций можно изменить, используя круглые скобки.

В языке *Verilog* существует различные типы логических операторов.

Таблица 3.1

Арифметические		Поразрядные и свертки		Отношения и сравнения	
Умножение	*	И	&	Равно?	==
Деление	/	ИЛИ		Не равно?	!=
Сложение	+	НЕ	~	Больше ?	>
Вычитание	-	Исключающее ИЛИ	^	Меньше ?	<
Остаток от деления	%	И — НЕ	~&	Больше или равно ?	>=
		ИЛИ — НЕ	~	Меньше или равно ?	<=
Объединение	{a, b, c}	Исключающее ИЛИ — НЕ	~^	Идентично?	===

**Поразрядные операторы** содержат два операнда, разделенные символом операции, в соответствии с табл. 3.1. Входные и выходной сигналы могут быть векторами. Векторы могут иметь различную разрядность. Если разрядность одного операнда меньше, чем другого, то недостающие разряды заполняются нулями.

**Операторы свертки** содержат один операнд, используют те же символы, что и поразрядные операторы (кроме оператора инверсии «~»). Они выполняются над единственным многоразрядным операндом побитно, шаг за шагом, формируя на выходе одноразрядный результат. Символ операции записывается перед именем вектора. Этот тип операторов позволяет формировать признаки данных. Так, например, если  $d$  — многоразрядный вектор, то признак  $p1 = \&d$  будет иметь один бит, равный единице, если все биты входного вектора равны единице, признак  $p2 = |d$  будет равен нулю, если все биты вектора равны нулю, а  $p3 = ^d$  будет признаком для контроля на четность.

**Операторы отношения и сравнения** сравнивают два операнда и выдают значение в виде однобитовой логической переменной, которая обычно используется в условных операторах. При выполнении условия выдается 1. Если хотя бы один операнд не определен, то и результат примет неопределенное значение  $x$ . В случае неопределенной разрядности операндов, более короткий дополняется слева нулями. Операторы сравнения используются в описаниях компараторов кодов.

**Оператор объединения** позволяет из исходных векторов сформировать вектор суммарной разрядности. Исходные векторы записываются в фигурных скобках через запятую. Объединенный вектор может быть записан справа и слева от знака равенства в операторе присваивания. Использование операторов объединения в поведенческих описаниях сумматора и инкрементора будет приведено в гл. 5.

**Оператор условного присваивания** содержит три операнда, выполняет присваивание выходному сигналу одного из двух значений в зависимости от условия. Запись оператора условного присваивания содержит следующие элементы:

- ключевое слово *assign*;
- имя выходного сигнала;
- знак равенства;
- условие — сигнал или выражение;
- знак вопроса;
- ветвь «ДА»;
- двоеточие;
- ветвь «НЕТ»;
- знак «точка с запятой».

Например, запись  $assign\ q = s ? a : b$ ; читается так: «если  $s = 1$  то  $q = a$ , иначе  $q = b$ ». В качестве условия  $s$  в данном операторе может использоваться выражение, формирующее логический сигнал 1 или 0, а вместо переменных  $a$  и  $b$  могут быть операторы, в том числе и другой условный оператор.

Все рассмотренные ранее операторы — параллельные, они являются основой описаний комбинационных схем.

**Состав описания на языке Verilog.** Описание комбинационных схем содержит следующие основные элементы.

- **Комментарий** — это пояснение, начинающееся с двух символов «косая черта» (знак деления на клавиатуре), содержащее русские буквы. Комментарий на компиляцию не влияет и может отсутствовать.
- **Заголовок** сначала это ключевое слово *module*, затем через пробел имя модуля и перечисление всех входных и выходных сигналов, заключенное в круглые скобки. Завершение заголовка и всех последующих строк точка с запятой.
- **Описание входных и выходных сигналов или портов** — после ключевых слов *input* описываются входные сигналы, а после ключевых слов *output* — выходные сигналы.
- **Ключевое слово assign и операторы**, описывающие устройство, — строк данного типа может быть несколько, они пред-



ставляют собой описание устройств по логическим уравнениям, или поведенческое описание.

*Описание по логическим функциям* составляется следующим образом. Началом оператора всегда является ключевое слово *assign*. Затем записывается логическое уравнение, в котором используются символы логических операций принятые в языке *Verilog*. Операция И обозначается знаком амперсанд &, операция ИЛИ — вертикальной чертой |, операция НЕ — знаком тильда ~, а операция исключающее ИЛИ — знаком ^.

*Поведенческое описание* — это алгоритм функционирования устройства или зависимость выходных сигналов от входных сигналов. При поведенческом описании устройство представлено как «черный ящик». Поведенческое описание обычно содержит операторы условного присваивания, объединения, сравнения, имеет более высокий уровень абстракции, оно, как правило, короче и проще в восприятии.

**Завершение** — ключевое слово *endmodule*, после которого не должно быть никаких знаков препинания.

**Последовательные операторы языка Verilog.** Последовательные операторы описывают устройства с элементами памяти, которые изменяют свое состояние (или срабатывают) при действии сигналов синхронизации. Обязательно должен быть указан тип *выходных сигналов* — *регистр* — *reg*.

Последовательные операторы начинаются с ключевого слова *always* (всегда), содержат символ @, список чувствительности в круглых скобках и оператор условного перехода *if*, оператор варианта *case* или блоки *begin* — *end*.

Ключевое слово *always* обозначает повторение присваивания при возникновении события. Символ @ эквивалентен понятию «событие».

**Список чувствительности** — это перечисление через запятую или знак дизъюнкции всех сигналов, изменение которых должно вызывать повторение процесса присваивания. В списке чувствительности может быть указано, по какому именно событию должно выполняться срабатывание (по фронту или спаду импульсного сигнала). Фронт сигнала обозначается *posedge*, спад — *negedge* (от слов *positive*, *negative* и *edge* — край).

Если оператор содержит только слово *always*, а список чувствительности отсутствует, то срабатывание оператора будет происходить при изменении любого из входных сигналов оператора.

**Оператор условного перехода if** позволяет выполнить один из двух возможных операторов в зависимости от значения условия. Запись оператора *if* содержит следующие элементы:

- ключевое слово *always*, символ @, список чувствительности;
- ключевое слово *if*;
- условие — переменная или выражение, результат которого 1 или 0;
- ветвь «ДА» выполняется, если результат вычисления условия дает результат 1;
- знак «точка с запятой»;
- ключевое слово *else*;
- ветвь «НЕТ» — выполняется, если условие дает результат 0, *x* или *z*;
- знак «точка с запятой».

Заметим, в записи оператора слово *then* отсутствует. Выходной сигнал, которому присваиваются значения, должен быть определен как регистр. При записи условия равенства в операции сравнения следует записывать два знака равенства «==». Ветвь «ДА» выполняется, если условие истинно и в результате его вычисления получено некоторое число (например, 1). Ветвь «НЕТ» выполняется, если результат вычисления условия равен 0 или содержит значения *x*, *z*.

**Оператор варианта case** относится к последовательным операторам, используется совместно с присваиванием *always* и позволяет выбрать один из возможных путей исполнения алгоритма в зависимости от значения селектора. Переменная, которой присваивается значение, должна быть определена как регистр.

Запись оператора *case* содержит следующие элементы:

- ключевое слово *always* и список чувствительности;
- ключевое слово *case*;
- селектор — переменная или выражение, записанное в круглых скобках;
- первый вариант значения селектора;
- знак «двоеточие»;
- первый вариант оператора;
- знак «точка с запятой»;
- второй вариант значения селектора;
- знак «двоеточие»;
- второй вариант оператора;
- знак «точка с запятой»;
- ключевое слово *endcase*.

Операторы *case* и *if* могут быть вложенными.

Более подробное рассмотрение особенностей использования последовательных операторов будет описано при рассмотрении устройств, содержащих элементы памяти (см. гл. 6...8).

Пример описания комбинационной схемы по логическим функциям. При проектировании устройств на основе ПЛИС для ввода исходных данных нередко вместо схем используют описания на языке *Verilog*. Составим описание комбинационной схемы *ks1* (см. рис. 3.2), содержащей логические элементы И, ИЛИ. Сумма по модулю 2. Описание заданной комбинационной схемы (пример 3.1) составлено в соответствии с рассмотренными правилами языка *Verilog*.

```
//Пример 3.1.                                1
//Комбинационная схема ks1                  2
module v31_ks1 (d,qand,qor,qxor);           //3
input [2:0] d;                               //4
output qand, qor, qxor;                     //5
assign qand = d[2] & d[1] & d[0];            //6
assign qor = d[2] | d[1] | d[0];/           //7
assign qxor = d[2] ^ d[1] ^ d[0];           //8
endmodule                                   //9
```

Начало описания (как правило, комментарий, который начинается с двух символов косая черта (знак деления на клавиатуре)) допускает русские буквы и не влияет на компиляцию. В приведенном примере комментарий занимает две строки и содержит номер примера и обозначение описанного устройства. Номера строк для удобства ссылок записаны как комментарии.

Строка 3 — заголовок, содержащий ключевое слово *module*, за ним следует имя устройства и перечисление всех входных и выходных сигналов в круглых скобках через запятую.

Имя устройства должно начинаться с буквы и содержать латинские буквы, цифры и знак подчеркивания. В имени не допускаются русские буквы и пробелы. Определенная система выбора имен файлов упрощает их поиск в общем каталоге. Для рассматриваемого описания выбрано имя *v31\_ks1*, означающее «пример на *Verilog* 3.1, комбинационная схема 1». Выбранное имя необходимо присвоить файлу, содержащему это описание.

Строки 4 и 5 содержат описание входных и выходных сигналов, которые называют также «порты ввода-вывода». После ключевого слова *input* описан входной сигнал, который в данном примере описан как 3-разрядный вектор или шина, содержащая несколько проводников, предназначенная для передачи параллельного кода. Перед именем вектора в квадратных скобках через двоеточие указан диапазон номеров разрядов. После слова *output* указаны выходные сигналы. Никаких указаний о типах и разрядности сигналов нет. Эти сигналы по умолчанию будут назначены как одноразрядные типа *wire*.

В строках 6...8 для выходного сигнала каждого элемента записаны параллельные операторы присваивания с ключевым словом *assign*. Операторы записаны по логическим функциям.

Для получения строки описания на язык *Verilog* необходимо записать логическую функцию и в этой записи заменить символы логических операций, принятые в алгебре логики, на символы языка *Verilog* (см. табл. 3.1). Индексы переменных в языке *Verilog* необходимо указывать в квадратных скобках после имени сигнала. Для первого логического элемента И запишем логическую функцию и соответствующую строку описания на *Verilog*:

$$qand = d_2 \cdot d_1 \cdot d_0;$$

$$assign\ qand = d[2] \& d[1] \& d[0].$$

Для второго и третьего сигналов аналогично можно записать логические функции и соответствующие строки описания:

$$qor = d_2 \vee d_1 \vee d_0; \quad qxor = d_2 \oplus d_1 \oplus d_0;$$

$$assign\ qor = d[2] | d[1] | d[0]; \quad assign\ qxor = d[2] \wedge d[1] \wedge d[0].$$

Завершением описания является ключевое слово *endmodule*, после которого не должно быть никаких знаков препинания.

Язык *Verilog* позволяет записать строки 6...8 сокращенно, используя операторы свертки, которые выполняют логическую операцию над многоразрядным вектором бит за битом и выдают результат в виде одного бита.

Запись оператора свертки содержит ключевое слово *assign*, затем оператор присваивания, в котором после знака равенства записан символ логической операции, а затем имя вектора. Вместо *assign qand = (d[2] & d[1] & d[0])* в строке 6 можно записать *assign qand = & d*. Подобная замена операторов для строки 7 имеет вид: *assign qor = |d*, а для строки 8: *assign qxor = ^d*.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы назначение и функциональные возможности системы *MAX + plus II*.
2. Перечислите этапы проектирования при вводе проекта в виде схемы.
3. Как создать новый графический файл?
4. Как выполняется ввод элементов схемы в графическом редакторе?

5. Как выполняется перемещение, копирование, удаление и изменение положения элементов схемы в графическом редакторе?
6. Как выполняется ввод и редактирование примитивов *INPUT* и *OUTPUT*.
7. Как выполняется физическое соединение цепей в графическом редакторе?
8. Как выполняется логическое соединение цепей в графическом редакторе?
9. Перечислите возможности редактора временных диаграмм.
10. Как выбрать контрольные точки?
11. Как создать тестовые входные сигналы?
12. Как изменить время моделирования?
13. Как определить временную задержку сигнала в схеме?
14. Перечислите этапы проектирования при вводе проекта в виде описания.
15. Как создать новый файл для ввода описания на языке *Verilog*?

## Глава 4

### АНАЛИЗ И СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ

Цифровые логические схемы подразделяются на два класса: комбинационные схемы, не содержащие элементов памяти, и последовательностные схемы, которые содержат элементы памяти.

#### 4.1. АНАЛИЗ КОМБИНАЦИОННЫХ СХЕМ

**Комбинационные схемы** состоят из логических элементов. Выходные сигналы этих схем однозначно определяются комбинацией логических сигналов на их входах в текущий момент времени и не зависят от сигналов, которые подавались ранее. Комбинационные схемы используются для построения разнообразных цифровых устройств. Это арифметические и логические устройства, мультиплексоры, дешифраторы, различные преобразователи кодов.

Комбинационная схема может быть задана различными способами, чаще всего это принципиальная схема. При разработке комбинационной схемы для реализации в ПЛИС и для ввода исходных данных может потребоваться ее описание на языке *Verilog*, а для проверки правильности функционирования — временные диаграммы и таблица истинности, поэтому началом разработки является анализ заданного устройства.

**Анализ комбинационной схемы** — это ее полное описание, которое содержит таблицу истинности, логические функции, принципиальную схему, описание на языке *Verilog* и временные диаграммы.

В качестве примера рассмотрим выполнение анализа комбинационной схемы *ks2* (рис. 4.1). Для заданной схемы составить таблицу истинности, логические функции, описание на языке *Verilog* и временные диаграммы. Рассмотрим выполнение задания.

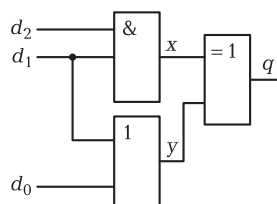


Рис. 4.1. Комбинационная схема ks2

**Составление таблицы истинности по заданной схеме.** Для составления таблицы истинности целесообразно дополнительно обозначить на схеме внутренние узлы  $x$  и  $y$  и добавить для этих узлов столбцы в таблицу. Сначала необходимо заполнить эти столбцы, учитывая логику работы соответствующих элементов (табл. 4.1). Сигнал  $x$  формируется логическим элементом И, на входы которого поданы сигналы  $d_2$ ,  $d_1$ , поэтому в соответствии с правилом для элемента И имеем:  $x = 1$ , если  $d_2 = d_1 = 1$ , иначе  $x = 0$ . Сигнал  $y$  формируется элементом ИЛИ, на входы которого поступают сигналы  $d_1$ ,  $d_0$ . В соответствии с правилом для операции ИЛИ:  $y = 0$ , если  $d_1 = d_0 = 0$ , иначе  $y = 1$ . Сигналы  $x$  и  $y$  позволяют определить выходной сигнал  $q$  по правилу для операции. Сумма по модулю 2:  $q = 1$ , если сумма единиц сигналов  $x$  и  $y$  нечетна.

**Составление логических функций по заданной схеме.** Сначала необходимо записать уравнения для вспомогательных сигналов  $x$ ,  $y$ , а затем — формулу для выходного сигнала  $q$ :

$$x = d_2 \cdot d_1; y = d_1 \vee d_0; q = x \oplus y = (d_2 \cdot d_1) \oplus (d_1 \vee d_0).$$

Использование скобок гарантирует необходимый порядок выполнения операций.

**Описание комбинационной схемы ks2 на языке Verilog по логическим функциям.** Описание на языке Verilog заданной комби-

национной схемы (см. рис. 4.1) выполнено с учетом правил, изложенных в гл. 3. В данном примере (пример 4.1) первые две строки — комментарий, строка 3 — заголовок, содержащий имя модуля и перечисление всех входных и выходных переменных. Имя модуля `v41_ks2` означает *пример на Verilog 4.1*, комбинационная схема 2. Такое же имя должен иметь файл, содержащий это описание.

```
//Пример 4.1. Комбинационная          1
// схема ks2                             2
module v41_ks2 (d, q);                  //3
input [2:0]d;                             //4
output q;                                 //5
assign q = (d[2]&d[1]) ^ (d[1] | d[0]);    //6
endmodule                                //7
```

В строках 4 и 5 выполняется назначение входных и выходных сигналов (портов ввода-вывода). После ключевого слова `input` описывается входной сигнал — 3-разрядный вектор, перед именем вектора в квадратных скобках через двоеточие указан диапазон номеров разрядов. После слова `output` указан выходной сигнал, который по умолчанию будет назначен как одноразрядный типа `wire`.

Строка 6 содержит параллельный оператор, начинающийся с обязательного ключевого слова `assign`, который записан по логической функции путем замены символов логических операций, принятых в алгебре логики на символы языка Verilog:

$$q = (d_2 \cdot d_1) \oplus (d_1 \vee d_0);$$

$$\text{assign } q = (d[2] \& d[1]) \wedge (d[1] | d[0]).$$

Завершением описания является ключевое слово `endmodule`.

**Построение временных диаграмм по таблице истинности.** Теоретические временные диаграммы построены с использованием таблицы истинности и логических функций (рис. 4.2).

Рекомендуется выполнить моделирование рассмотренной схемы. Ввод исходных данных проекта можно выполнить двумя в виде схемы и в виде описания. При вводе проекта в виде схемы

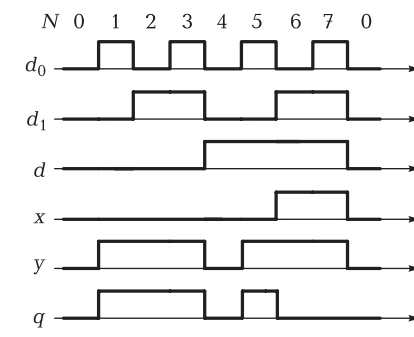


Рис. 4.2. Временные диаграммы, поясняющие работу схемы ks2

необходимо по схеме, изображенной на рис. 4.1 создать графический файл с именем `s41_ks2.gdf` и выполнить этапы, описанные ранее. При вводе в виде описания необходимо по примеру 4.1. создать текстовый файл с именем `v41_ks2.v` и выполнить уже перечисленные этапы. Результаты моделирования должны показать, что оба способа ввода данных — в виде схемы и в виде описания — дают одинаковый результат.

4.2. СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ

Под синтезом понимают проектирование и разработку заданного устройства по определенным правилам, обеспечивающим оптимальное решение поставленной задачи. Синтез комбинационной содержит несколько этапов:

- анализ технического задания;
- составление таблицы истинности;
- запись логических функций;
- минимизация логических функций;
- разработка схемы;
- описание устройства на языке, построение временных диаграмм.

Исходные данные на проектирование могут быть заданы в виде словесного описания выполняемой функции или в виде таблицы истинности. Выполнение перечисленных этапов будет рассмотрено на примерах.

4.2.1. Синтез комбинационной схемы ks3

**Задание.** Выполнить синтез комбинационной схемы `ks3`, заданной таблицей истинности (табл. 4.2), составить ее описание на языке `Verilog`.

Таблица 4.2		
Номер набора	Аргументы $d_1 d_0$	Функции $q_3 q_2 q_1 q_0$
0	0 0	0 1 1 0
1	0 1	0 0 1 1
2	1 0	1 1 1 1
3	1 1	1 0 0 0

Рассмотрим выполнение синтеза по пунктам.

**Анализ технического задания.** Началом синтеза является четкое и однозначное определение закона функционирования синтезируемой схемы. Необходимо определить входные и выходные сигналы, а также изобразить символ разрабатываемого устройства (рис. 4.3). В соответствии с ГОСТом символы логических элементов изображаются в сетке 2,5 мм, входы — слева, выходы — справа, в среднем поле — символическое отображение функционального назначения устройства.

**Составление таблицы истинности.** В общем случае таблицу истинности составляет разработчик на основе анализа технического задания. В данном случае таблица истинности задана. Для функции двух переменных она содержит четыре строки. Заданы четыре выходных сигнала.

**Составление логических функций по таблице истинности.** Комбинационная схема может быть описана логическими функциями, представленными в совершенной дизъюнктивной нормальной форме (СДНФ) или в совершенной конъюнктивной нормальной форме (СКНФ). Названия функций определяет правила их записи.

Логическая функция в СДНФ представляет собой дизъюнкцию вспомогательных функций — минтермов.

**Минтерм** (конституента единицы) — это произведение всех входных сигналов, записанных без инверсии или с инверсией, которое равно 1 только на заданном наборе аргументов. Каждый минтерм соответствует одной строке таблицы истинности, в которой функция равна 1. Если в таблице истинности количество строк, для которых функция равна 1, сравнительно небольшое, то используется запись логической функции в СДНФ. Для этого необходимо выполнить следующие действия:

- 1) записать несколько произведений всех аргументов, число которых равно числу единиц логической функции в таблице истинности;
- 2) соединить произведения знаками дизъюнкции;
- 3) записать под каждым произведением двоичный набор, для которого функция равна 1; над аргументом, которому в наборе соответствует нуль, поставить инверсию.

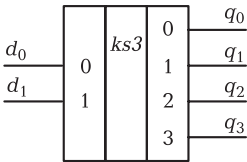


Рис. 4.3. Условное обозначение комбинационной схемы `ks3`



Другая форма записи логической функции — **СКНФ** — представляет собой *конъюнкцию*, или произведение вспомогательных функций — *макстермов*.

**Макстерм** (или конституента нуля) — это дизъюнкция всех входных сигналов, записанных без инверсии или с инверсией, которая равна 0 только на заданном наборе аргументов. Каждый макстерм соответствует строке таблицы истинности, в которой функция равна 0. Если количество строк, для которых функция равна 0, является небольшим, то целесообразно выбрать запись в СКНФ.

Для получения логических уравнений в СКНФ записываются дизъюнкции всех аргументов, которые заключаются в скобки и соединяются знаками умножения. Число сомножителей равно числу строк в таблице истинности, для которых функция равна 0. Под каждой дизъюнкцией записывается двоичный набор, на котором функция равна 0. Над аргументами, равными 1, в дизъюнкции ставится инверсия.

Рассмотрим на примере получение уравнений по таблице истинности (см. табл. 4.2).

Выделяя для каждой функции столбец, запишем выражения в СДНФ:

$$\begin{aligned} q_3 &= d_1 \cdot \bar{d}_0 \vee d_1 \cdot d_0; & q_2 &= \bar{d}_1 \cdot \bar{d}_0 \vee d_1 \cdot \bar{d}_0; \\ q_1 &= \bar{d}_1 \cdot \bar{d}_0 \vee \bar{d}_1 \cdot d_0 \vee d_1 \cdot \bar{d}_0; & q_0 &= \bar{d}_1 \cdot d_0 \vee d_1 \cdot \bar{d}_0. \end{aligned}$$

Выражения в СКНФ будут иметь вид:

$$\begin{aligned} q_3 &= (\bar{d}_1 \vee \bar{d}_0) \cdot (d_1 \vee d_0); & q_2 &= (d_1 \vee \bar{d}_0) \cdot (\bar{d}_1 \vee \bar{d}_0); \\ q_1 &= \bar{d}_1 \vee \bar{d}_0; & q_0 &= (d_1 \vee \bar{d}_0) \cdot (\bar{d}_1 \vee d_0). \end{aligned}$$

**Минимизация логических функций.** Логические функции, записанные по таблице истинности, в ряде случаев могут быть упрощены и записаны в минимальной дизъюнктивной нормальной форме (МДНФ) или в минимальной конъюнктивной нормальной форме (МКНФ). Минимизация логических функций позволяет уменьшить аппаратные затраты при построении комбинационной схемы, при этом также повышаются надежность и быстродействие проектируемого устройства.

Минимизация может быть выполнена аналитически с использованием теорем и законов алгебры логики. Так, например, логическую функцию  $q_3$  можно минимизировать, используя логическую функцию склеивания:

$$q_3 = d_1 \cdot \bar{d}_0 \vee d_1 \cdot d_0 = d_1 \cdot (\bar{d}_0 \vee d_0) = d_1.$$

Аналитические методы минимизации не всегда позволяют определить, является ли полученное уравнение минимальным.

Простым и эффективным методом минимизации логических уравнений является *метод карт Карно* (другое название — диаграммы Вейча — Карно), позволяющий осуществить операции склеивания и найти минимальные дизъюнктивную или конъюнктивную формы.

Карта Карно представляет собой прямоугольную таблицу. Для функции  $n$  переменных карта Карно состоит из  $2^n$  клеток, каждая из которых соответствует одной строке таблицы истинности и, соответственно, определенному набору входных переменных. В клетки записываются соответствующие значения функции. Если таблица истинности содержит значения для нескольких функций, то для каждой функции строится отдельная карта Карно.

Разметка строк и столбцов карты Карно выполняется таким образом, чтобы для любой клетки было легко определить соответствующий ей набор аргументов. Если для обозначения строк таблицы выбрать старший двоичный разряд кода набора, а для обозначения столбцов — младший разряд, то объединение этих значений будет соответствовать коду набора клетки, расположенной на пересечении выбранных строки и столбца. Выбранный порядок нумерации обозначают в верхней левой части таблицы ( $d_1 \setminus d_0$ ). Номера наборов, соответствующие определенным клеткам, показаны на **рис. 4.4**. Первая цифра набора определяется номером строки, а вторая — номером столбца.

Для отображения значений аргументов, соответствующих клеткам карты Карно, также используют квадратные скобки, которые соответствуют значению аргумента, равному 1, а отсутствие скобки означает равенство аргумента нулю. На **рис. 4.4** приведена двойная разметка строк и столбцов карты, несмотря на ее избыточность. Показаны значения аргументов для строк и столбцов, а также квадратные скобки.

Для того чтобы с помощью карты Карно задать функцию, необходимо в каждую клетку с определенным набором аргументов занести значение функции из соответствующей строки таблицы истинности. Клетка, в которой записана 1, отображает минтерм, а в которой 0 — макстерм.

Минимизация логических функций методом карт Карно заключается в выделении на

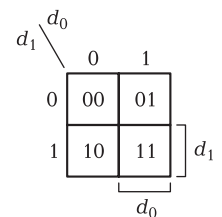


Рис. 4.4. Нумерация клеток в карте Карно для двух переменных

карте контуров и записи выражений, которые описывают эти контуры. Контуры должны быть правильными. Число единиц (или нулей) в правильном контуре должно быть равно степени числа 2 (2, 4, 8...). При этом требуется, чтобы контуры были максимальны по числу охватываемых единиц, а число всех контуров было минимальным. Контуры могут перекрываться.

При поиске МДНФ выделяются правильные контуры, охватывающие соседние единицы карты. Выражение, описывающее контур, содержащий две 2 единицы, в котором пропадает (склеиваются) одна переменная, заменяет два минтерма, соответствующие этим единицам. Такая замена минимизирует исходную функцию. Если контур содержит четырех единицы, то вместо четырех минтермов можно записать одно произведение, в котором пропадает две переменных, и т.д.

Важно отметить, что коды наборов, соответствующие соседним клеткам карты Карно, отличаются только в одном разряде, что показывает **рис. 4.4**. Это свойство является основой метода минимизации, выполняемого посредством выделения контуров. Для нумерации клеток таблиц для большого количества переменных используется *код Грея*, значения которого для соседних клеток различаются только в одном разряде. Это коды 00, 01, 11, 10. Данную последовательность кодов можно получить при обходе клеток карты Карно для двух переменных (см. **рис. 4.4**) по часовой стрелке.

В рассматриваемом примере для минимизации функций, заданных таблицей истинности (см. табл. 4.2), необходимо для каждой функции составить карту Карно и выделить на них контуры (**рис. 4.5**).

Для функции  $q_3$  можно выделить контур, охватывающий две единицы, для которого переменная  $d_0$  принимает значения 0 и 1 и пропадает (склеивается), а переменная  $d_1$  не изменяется,  $d_1 = 1$ . Поэтому МДНФ для функции  $q_3$  имеет вид:  $q_3 = q_1$ .

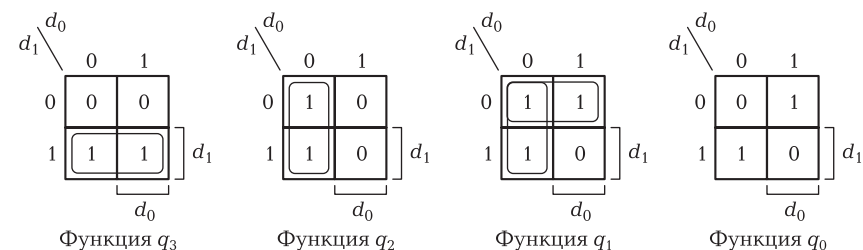


Рис. 4.5. Выделение контуров на картах Карно

Для функции  $q_2$ , пропадает переменная  $d_1$ , а переменная  $d_0 = 0$ , следовательно  $q_2 = \overline{d_0}$ .

Для функции  $q_1$  можно выделить два контура, которые перекрываются, что позволяет записать эту функцию в виде:  $q_1 = \overline{d_0} \vee \overline{d_1}$ . Используя *закон де Моргана*, данную функцию можно преобразовать к виду:  $q_1 = \overline{d_1 \cdot d_0}$ .

Для функции  $q_0$  выделить контуры, содержащие несколько единиц или нулей, не удастся. Форма записи этой функции, полученная по таблице истинности, уже является минимальной — МДНФ или МКНФ:

$$q_0 = \overline{d_1} \cdot d_0 \vee d_1 \cdot \overline{d_0}, \text{ или } q_0 = (d_1 \vee \overline{d_0}) \cdot (\overline{d_1} \vee d_0).$$

При поиске МКНФ функция представляется в карте Карно ее нулями, выделяются контуры, охватывающие нули. Контуры заменяются элементарными дизъюнкциями, которые соединяются знаками конъюнкции.

**Разработка схемы.** Полученные логические функции в МДНФ используют для построения схемы. Схема проектируемого устройства изображена на **рис. 4.6**. Минимизация логических функций позволила сократить аппаратные затраты. В приведенной схеме для выполнения функции  $d_3 = d_1$  использован буфер, который передает без изменения входной сигнал на выход. Заметим, в библиотеке *prim* подобный буфер имеет имя *wire*, подобно проводнику он передает логический сигнал без изменения.

**Описание комбинационной схемы на языке Verilog по логическим функциям.** Описание на языке *Verilog* рассматриваемой комбинационной схемы (см. **рис. 4.3**) выполнено по полученным логическим функциям подобно описаниям, рассмотренным в примерах 3.1 и 4.1.

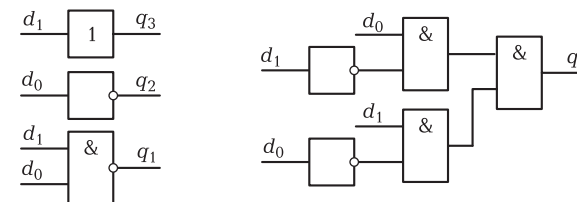


Рис. 4.6. Схема синтезированного устройства ks3

В примере 4.2 строки 1, 2 — комментарий, строка 3 — заголовок, содержащий имя модуля *v42\_ks3* и перечисление всех входных и выходных переменных.

```
//Пример 4.2. Комбинационная          1
// схема ks3                             2
module v42_ks3 (d, q);                  //3
input [1:0]d;                            //4
output [3:0] q;                          //5
assign q[3] = d[1];                      //6
assign q[2] = ~d[0];                     //7
assign q[1] = ~(d[1]&d[0]);                //8
assign q[0] = (~d[1]&d[0]) | (d[1] &~d[0]); //9
endmodule                               //10
```

В строках 4 и 5 выполняется назначение портов ввода-вывода. После ключевого слова *input* описывается входной сигнал — 2-разрядный вектор, перед именем вектора в квадратных скобках через двоеточие указан диапазон номеров разрядов. После слова *output* указан выходной сигнал как 4-разрядный вектор.

Строки 6...9 содержат параллельные операторы присваивания, начинающиеся с обязательного ключевого слово *assign*, которые записаны по логическим функциям для данного устройства.

**Внимание!** Индекс сигнала принадлежащего шине, записывается в квадратных скобках после имени.

Завершением описания является ключевое слово *endmodule* в строке 10.

**Построение временных диаграмм.** Теоретические временные диаграммы построены с использованием таблицы истинности и минимизированных логических функций (рис. 4.7).

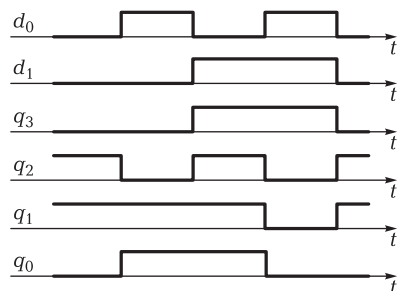


Рис. 4.7. Временные диаграммы работы устройства *ks3*

Для проверки правильности функционирования полученной схемы рекомендуется выполнить ее моделирование в САПР MAX + plus II.

## 4.2.2. Синтез формирователя признака числа

**Техническое задание.** Выполнить синтез комбинационной схемы *ks4*, которая формирует признак *q* принадлежности числа *d* заданному диапазону в соответствии с условием:  $q = 1$ , если  $d > 8$ .

Входной сигнал — 4-разрядное двоичное число *d*.

Выходной сигнал — признак *q*, содержит один разряд.

Заданному устройству соответствует условное обозначение (рис. 4.8).

**Составление таблицы истинности.** В рассматриваемом примере имеем логическую функцию четырех переменных, для которой количество наборов входных сигналов равно 16, поэтому таблица истинности также будет содержать 16 строк. В соответствии с заданием в устройстве должна быть определена только одна функция — заданный признак числа (табл. 4.3).

Таблица дополнена столбцами, содержащими минтермы и макстермы заданной функции. В каждой строке таблицы, в которой функция равна 1, записан минтерм, а в строке, где функция равна 0, — макстерм. В данном примере количество строк, в которых функция равна единице, меньше количества строк, в которых функция равна нулю.

**Составление логических функций по таблице истинности.** Логические функции в совершенной дизъюнктивной нормальной форме представляют собой дизъюнкцию минтермов, число которых равно числу единиц логической функции в таблице истинности. Каждый минтерм (см. табл. 4.3) является произведением всех аргументов, в котором над аргументом, равным в данном наборе нулю, ставится знак инверсии.

Индексы минтермов соответствуют десятичным эквивалентам кодов наборов. Логическая функция для сигнала *q* в СДНФ имеет вид:

$$q = d_3 \cdot \bar{d}_2 \cdot \bar{d}_1 \cdot d_0 \vee d_3 \cdot \bar{d}_2 \cdot d_1 \cdot \bar{d}_0 \vee \\ \vee d_3 \cdot \bar{d}_2 \cdot d_1 \cdot d_0 \vee d_3 \cdot d_2 \cdot \bar{d}_1 \cdot \bar{d}_0 \vee \\ \vee d_3 \cdot d_2 \cdot \bar{d}_1 \cdot d_0 \vee d_3 \cdot d_2 \cdot d_1 \cdot \bar{d}_0 \vee d_3 \cdot d_2 \cdot d_1 \cdot d_0.$$

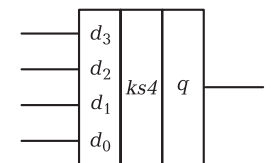


Рис. 4.8. Условное обозначение комбинационной схемы *ks4*

Таблица 4.3

$d_3$	$d_2$	$d_1$	$d_0$	$q$	Минтермы функции $q$	Макстермы функции $q$
0	0	0	0	0	—	$M_0 = d_3 \vee d_2 \vee d_1 \vee d_0$
0	0	0	1	0	—	$M_1 = d_3 \vee d_2 \vee d_1 \vee \bar{d}_0$
0	0	1	0	0	—	$M_2 = d_3 \vee d_2 \vee \bar{d}_1 \vee d_0$
0	0	1	1	0	—	$M_3 = d_3 \vee d_2 \vee \bar{d}_1 \vee \bar{d}_0$
0	1	0	0	0	—	$M_4 = d_3 \vee \bar{d}_2 \vee d_1 \vee d_0$
0	1	0	1	0	—	$M_5 = d_3 \vee \bar{d}_2 \vee d_1 \vee \bar{d}_0$
0	1	1	0	0	—	$M_6 = d_3 \vee \bar{d}_2 \vee \bar{d}_1 \vee d_0$
0	1	1	1	0	—	$M_7 = d_3 \vee \bar{d}_2 \vee \bar{d}_1 \vee \bar{d}_0$
1	0	0	0	0	—	$M_8 = \bar{d}_3 \vee d_2 \vee d_1 \vee d_0$
1	0	0	1	1	$m_9 = d_3 \cdot \bar{d}_2 \cdot \bar{d}_1 \cdot d_0$	—
1	0	1	0	1	$m_{10} = d_3 \cdot \bar{d}_2 \cdot d_1 \cdot \bar{d}_0$	—
1	0	1	1	1	$m_{11} = d_3 \cdot \bar{d}_2 \cdot d_1 \cdot d_0$	—
1	1	0	0	1	$m_{12} = d_3 \cdot d_2 \cdot \bar{d}_1 \cdot \bar{d}_0$	—
1	1	0	1	1	$m_{13} = d_3 \cdot d_2 \cdot \bar{d}_1 \cdot d_0$	—
1	1	1	0	1	$m_{14} = d_3 \cdot d_2 \cdot d_1 \cdot \bar{d}_0$	—
1	1	1	1	1	$m_{15} = d_3 \cdot d_2 \cdot d_1 \cdot d_0$	—

Для получения логических функций в СКНФ записываются в скобках все макстермы, и соединяются знаками умножения. Число сомножителей равно числу строк в таблице истинности, для которых функция равна нулю.

Для каждого макстерма учитывается двоичный набор, на котором функция равна нулю, над аргументами, равными единице, в дизъюнкции ставится инверсия. Логическая функция для сигнала  $q$  в СКНФ имеет вид:

$$q = (d_3 \vee d_2 \vee d_1 \vee d_0) \cdot (d_3 \vee d_2 \vee d_1 \vee \bar{d}_0) \cdot (d_3 \vee d_2 \vee \bar{d}_1 \vee d_0) \cdot (d_3 \vee d_2 \vee \bar{d}_1 \vee \bar{d}_0) \times \\ \times (d_3 \vee \bar{d}_2 \vee d_1 \vee d_0) \cdot (d_3 \vee \bar{d}_2 \vee d_1 \vee \bar{d}_0) \cdot (d_3 \vee \bar{d}_2 \vee \bar{d}_1 \vee d_0) \cdot (d_3 \vee \bar{d}_2 \vee \bar{d}_1 \vee \bar{d}_0) \times \\ \times (\bar{d}_3 \vee d_2 \vee d_1 \vee d_0).$$

**Минимизация логических функций.** Для минимизации логических функций используют метод карт Карно. Карта Карно для минимизации логической функции  $q$  должна содержать 16 клеток, что соответствует числу строк в таблице истинности (см. табл. 4.3).

Для обозначения строк таблицы выберем два старших двоичных разряда кода набора, а для обозначения столбцов — два младших разряда (рис. 4.9). Выбранный порядок нумерации обозначен в верхней левой части таблицы ( $d_3d_2/d_1d_0$ ). Нумерация строк и столбцов производится в коде Грея. В клетках карты Карно показаны коды наборов аргументов, соответствующие этим клеткам. Также расставлены квадратные скобки, которые для каждой из четырех переменных определяют области на карте, в пределах которых данная переменная равна 1.

При использовании карты для отображения значений заданной функции вместо показанных кодов необходимо записать в соответствующие клетки значения функции (0 или 1). Карта Карно, составленная для рассматриваемого примера (рис. 4.10), позволяет выделить три контура, каждый из которых содержит четыре клетки, и записать логическое уравнение в МДНФ:

$$q = d_3d_2 \vee d_3d_1 \vee d_3d_0 = d_3(d_0 \vee d_1 \vee d_2).$$

**Разработка схемы.** Схема устройства (рис. 4.11) строится по минимизированным логическим уравнениям.

**Описание комбинационной схемы на языке Verilog.** Наиболее удачным в данном случае является поведенческое описание, в основе которого лежит функциональная зависимость выходного

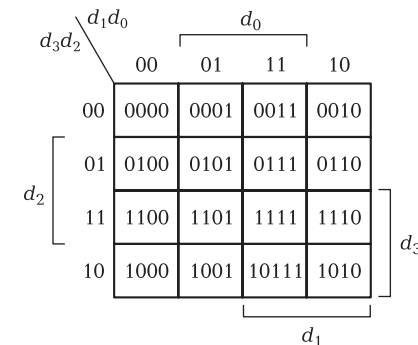


Рис. 4.9. Нумерация клеток карты Карно для четырех переменных

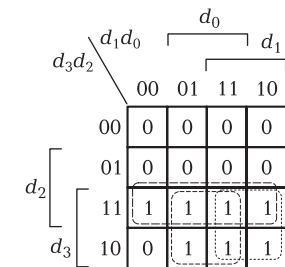


Рис. 4.10. Карта Карно, составленная для рассматриваемого примера

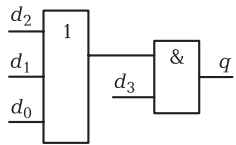


Рис. 4.11. Схема формирования признака *ks4*

сигнала от входных сигналов (пример 4.3).  
Укажем содержание строк.

```
//Пример 4.3. Комбина-
//ционная схема ks4
module v43_ks4 (d, q);
input [3:0] d;
output q;
assign q = d>8;
endmodule
```

Строки 1, 2 — комментарий, строка 3 — заголовок, в котором после ключевого слова *module* записано имя модуля *v43\_ks4* (пример на *Verilog 4.3* комбинационная схема *ks4*) и в скобках перечислены входные и выходные сигналы. Далее в строках 4, 5 описаны все сигналы. Входной сигнал *d* — вектор, выход *q* — по умолчанию, одноразрядный сигнал.

Описание поведения, другими словами — работы, предельно краткое, в строке 6 предписано назначать выходному сигналу *q* результат операции сравнения  $d > 8$ . В строке 4 указано, что входной сигнал *d* является 4-разрядным вектором, число 8 также будет представлено в виде четырех двоичных разрядов. Компилятор языка *Verilog* составит схему сравнения двух 4-разрядных чисел. Результат этого сравнения — сигнал разрядности 1 бит, принимающий значения 1 или 0, будет присвоен сигналу *q*.

Данный пример показывает, что описание цифровых устройств на *HDL*-языках существенно упрощает проектирование.

**Построение временных диаграмм.** Для полного тестирования несложных устройств комбинационного типа необходимо выполнить перебор всех возможных комбинаций входных сигналов. Временные диаграммы (рис. 4.12) составлены по таблице истинности.

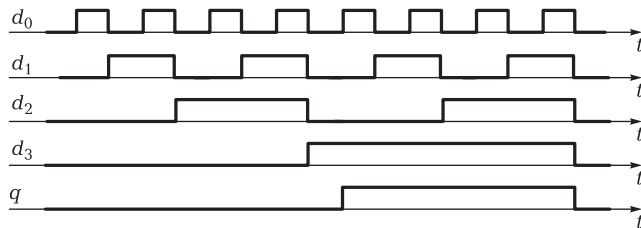


Рис. 4.12. Временные диаграммы работы комбинационной схемы *ks4*

По результатам моделирования можно сделать выводы о работоспособности устройства сопоставлением теоретических и реальных временных диаграмм.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Из каких этапов состоит анализ комбинационных схем?
2. Поясните, как составить логические функции по заданной схеме.
3. Как описать комбинационную схему на языке *Verilog* по логическим функциям?
4. Укажите значения сигналов, используемые в языке *Verilog*.
5. Перечислите элементы описания на языке *Verilog*.
6. Охарактеризуйте параллельные и последовательные операторы языка *Verilog*.
7. Как составить временные диаграммы по таблице истинности?
8. Из каких этапов состоит синтез комбинационных схем?
9. Как составить логических функций по таблице истинности?
10. Зачем используют минимизацию булевых выражений?



## ПРОЕКТИРОВАНИЕ ЛОГИЧЕСКИХ УСТРОЙСТВ КОМБИНАЦИОННОГО ТИПА

По функциональному назначению можно выделить несколько классов комбинационных устройств. Это сумматоры, мультиплексоры и демультиплексоры, шифраторы и дешифраторы, преобразователи кодов, цифровые компараторы, арифметико-логические устройства (АЛУ). Логические устройства комбинационного типа не содержат элементов памяти и формируют выходные сигналы в соответствии с текущими уровнями входных сигналов.

Цифровые устройства, проектируемые на основе ПЛИС, обычно содержат типовые комбинационные схемы, разработанные для решения наиболее часто встречающихся задач. При проектировании сложных устройств, содержащих простые комбинационные схемы, сначала выполняют анализ функционирования и синтез комбинационных схем.

### 5.1. ПОЛУСУММАТОР

Простейшей арифметической операцией в цифровых устройствах является сложение. **Полусумматор** — устройство для суммирования двух одноразрядных двоичных чисел  $a$  и  $b$ . Результатами

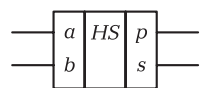


Рис. 5.1. Условное обозначение полусумматора

суммирования являются бит суммы данного разряда  $s$  и перенос в следующий старший разряд  $p$ . Условное обозначение полусумматора изображено на рис. 5.1.

Для синтеза схемы необходимо составить таблицу истинности (табл. 5.1). Если одно из слагаемых равно 1, то сумма  $s$  также равна 1. Если суммируются два слагаемых, равных 1, то сумма  $s$  равна 0, и возникает перенос  $p = 1$ .

Логические функции для выходных сигналов  $p$  и  $s$  целесообразно записать в СДНФ. Столбец для сигнала  $p$  содержит всего одну единицу, поэтому выражение для  $p$  содержит один минтерм. Функция  $s$  содержит два минтерма и является операцией Сумма по модулю 2:

$$p = a \cdot b; \quad s = \bar{a} \cdot b \vee a \cdot \bar{b} = a \oplus b.$$

Минимизация логических функций в данном примере не требуется.

Полученные формулы позволяют составить схему, один из вариантов, показан на рис. 5.2.

По логическим функциям может быть составлено описание полусумматора на языке *Verilog* (пример 5.1).

```
//Пример 5.1. Полусумматор,
//описание по функциям
module v51_hs (a, b, p, s); //3
    input a, b; //4
    output p, s; //5
    assign s = a ^ b; //6
    assign p = a & b; //7
endmodule //8
```

Строки 1, 2 — комментарий, строка 3 — заголовок, содержит ключевое слово *module*, затем имя устройства и перечисление всех входных и выходных сигналов в круглых скобках через запятую. При записи данной строки использованы обозначения выводов, изображенные на символе устройства. Выбранное в данном случае имя *v51\_hs* означает *Verilog, пример 5.1, полусумматор*. Все строки заканчиваются точкой с запятой.

В строках 4 и 5 после ключевого слова *input* перечисляются входные сигналы, разделенные запятой, а после слова *output* аналогично перечисляются все выходные сигналы. Все сигналы данного устройства однобитовые типа *wire*. Эти типы сигналов в описании указывать не требуется, они назначаются по умолчанию.

В логических операторах (строки 6, 7) сначала записано обязательное ключевое слово *assign*, а затем логическая функция, описывающая один из выходных сигналов сумматора  $p$  или  $s$ . Запись функ-

Таблица 5.1

$a$	$b$	$p$	$s$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

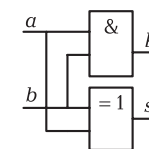


Рис. 5.2. Схема полусумматора

ции выполнена заменой символов логических операций, принятых в алгебре логики на символы языка *Verilog* (см. примеры 3.1 и 4.1).

Логическую функцию  $p = a \cdot b$  описывает строка 7: *assign p = a & b*, а функцию  $s = a \oplus b$  описывает строка 6: *assign s = a ^ b*.

Строка 8 — завершение описания — ключевое слово *endmodule*, после которого не должно быть никаких знаков препинания.

В языке *Verilog* предусмотрен другой вид описания устройств — это поведенческое описание, которое описывает алгоритмом функционирования устройства или зависимость выходного сигнала от входных сигналов. Поведенческое описание, как правило, короче и проще в восприятии. Для его составления необходимо определить алгоритм функционирования устройства.

Анализ таблицы истинности позволяет обнаружить такой алгоритм. В полусумматоре складываются два одноразрядных слагаемых (рис. 5.3), а результат представляют две одноразрядные переменные. Из таб. 5.1 следует, что сигналы  $p$  и  $s$  можно объединить в 2-разрядное двоичное число, равное сумме входных слагаемых, при этом перенос  $p$  является старшим разрядом, а сумма  $s$  — младшим.

Поведенческое описание полусумматора приведено в примере 5.2. Его отличия от предыдущего примера — имя и строка 6, описывающая работу полусумматора.

```
//Пример 5.2. Полусумматор
//поведенческое описание
module v52_hs (a, b, p, s); //3
input a, b; //4
output p, s; //5
assign {p,s} = a + b; //6
endmodule //7
```

В строке 6 сигналы  $p$  и  $s$  объединены в 2-разрядное слово (вектор) при помощи оператора объединения. Для обозначения этого оператора объединяемые сигналы записывают в фигурных скобках

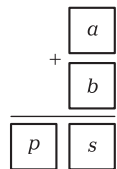


Рис. 5.3. Пояснение работы полусумматора

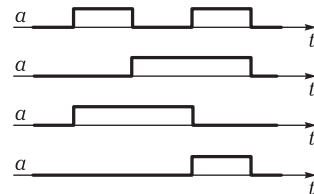


Рис. 5.4. Временные диаграммы, поясняющие работу полусумматора

через запятую. В результате выполнения оператора указанные сигналы будут представлять собой 2-разрядный код (вектор). В строке 6 этому вектору присваивается значение, равное сумме слагаемых.

Анализ работы полусумматора и таблица истинности позволяют построить теоретические временные диаграммы (рис. 5.4), которые могут быть использованы при создании тестовых сигналов и при моделировании устройства.

## 5.2. ИНКРЕМЕНТОР

Операция прибавления единицы к числу называется **инкремент**, а устройство, выполняющее эту операцию, называется **инкрементор**.

В качестве примера рассмотрим 4-разрядный инкрементор (рис. 5.5), который прибавляет сигнал входного переноса  $c$  к входному 4-разрядному коду  $d$  и выдает выходной 4-разрядный код  $q$  и выходной перенос  $p$ .

Схема инкрементора должна в каждом разряде содержать полусумматор, так как во всех разрядах многоразрядного двоичного числа суммируются только два слагаемых (рис. 5.6).

В младшем разряде числа к коду младшего разряда  $d$  прибавляется сигнал входного переноса  $c$ , а ко всем остальным разрядам числа прибавляется перенос из предыдущего младшего разряда. Наличие сигнала входного переноса расширяет функциональные возможности инкрементора. Инкремент выполняется только при  $c = 1$ , а при  $c = 0$  инкремента нет.

Инкрементор, построенный в виде комбинационной схемы, позволяет получить результат операции непосредственно после по-

Рис. 5.5. Обозначение инкрементора

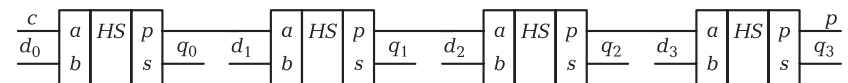
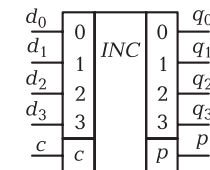


Рис. 5.6. Схема 4-разрядного инкрементора

дачи входных сигналов в отличие от схем с использованием счетчиков, где результат появляется лишь в следующем такте синхронизации.

При проектировании инкремента для реализации в ПЛИС используют его описание на языке *Verilog*.

Анализ постановки задачи и форматов данных позволяет использовать поведенческое описание с оператором объединения. Бит выходного переноса  $p$  и 4-разрядный выходной код  $q$  образуют 5-разрядное число (рис. 5.7).

Для описания инкремента в примере 5.3 выбрано имя *incr*, которое указано в заголовке (строка 2). Сигналы входного и выходного кодов  $d$  и  $q$  описаны как векторы (параллельные коды), для каждого сигнала перед его именем в квадратных скобках через двоеточие указаны значения старшего, а затем младшего индексов. Сигналы  $c$  и  $p$  по умолчанию будут одноразрядными.

```
//Пример 5.3. Инкремента
module v53_incr (d, c, q, p);           //2
input [3:0] d;                         //3
input c;                               //4
output [3:0] q;                        //5
output p;                              //6
assign {p,q} = d + c;                  //7
endmodule                             //8
```

Описание работы инкремента выполняет всего одна строка (строка 7) с оператором объединения, в котором выходные сигналы

Рис. 5.7. Форматы данных инкремента

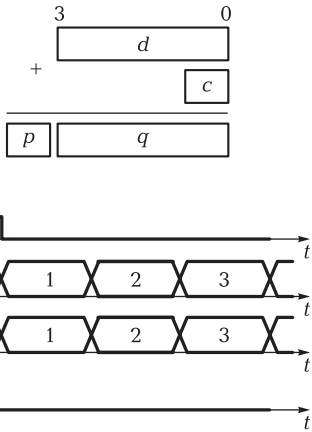


Рис. 5.8. Временные диаграммы, поясняющие работу инкремента

$p$  и  $q$  записаны в фигурных скобках через запятую и образуют объединенный вектор в соответствии с рис. 5.7.

Работу инкремента поясняют временные диаграммы (рис. 5.8). При  $c = 1$  на выходе инкремента формируется код:  $q = d + 1$ . Прибавление единицы к максимальному для 4-разрядного числа входному коду  $d = F$  дает на выходе код  $q = 0$  и перенос  $p = 1$ . При значении входного переноса  $c = 0$  инкремента нет:  $q = d$ .

### 5.3. СУММАТОР

Комбинационный сумматор является составной частью большинства цифровых вычислительных устройств. Выполним синтез одноразрядного сумматора в соответствии с методикой, рассмотренной в гл. 4.

Анализ постановки задачи. **Сумматор** — устройство, предназначенное для арифметического суммирования одного разряда кодов двух чисел, при этом производится суммирование трех цифр: первого слагаемого  $a$ , второго слагаемого  $b$  и переноса из соседнего младшего разряда  $c$  (рис. 5.9).

Выходами схемы являются сумма для данного разряда  $s$  и перенос в следующий разряд  $p$ . Сумматор, имеющий три входа, в отличие от полусумматора, называют также **полным сумматором**. Включая параллельно несколько подобных схем, можно получить сумматор с произвольным числом разрядов.

**Составление таблицы истинности.** Таблица истинности для сумматора описывает логическую функцию трех переменных, следовательно, количество наборов входных сигналов или строк таблицы равно 8. При составлении таблицы истинности (табл. 5.2) учитываем, что значения переноса и суммы образуют двухразрядное двоичное число, равное количеству входных сигналов, равных единице. Если в строке таблицы только один из входов равен 1, то  $p = 0$ ,  $s = 1$ ; если два сигнала равны 1, то  $p = 1$ ,  $s = 0$ ; если три сигнала равны 1, то  $p = 1$ ,  $s = 1$ .

В дополнение к таблице истинности приведены минтермы и макстермы функций  $p$  и  $s$ , записанные в соответствии с правилами, рассмотренными в гл. 4.

**Составление логических функций.** Логические функции в СДНФ представляют собой дизъюнкцию всех минтермов, число которых равно числу единиц логической функции

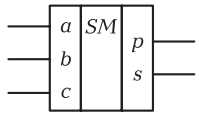


Рис. 5.9. Обозначение сумматора

Таблица 5.2									
$a$	$b$	$c$	$p$	$s$	Минтермы функции $p$	Макстермы функции $p$	Минтермы функции $s$	Макстермы функции $s$	
0	0	0	0	0	—	$M_0 = a \vee b \vee c$	—	$M_0 = a \vee b \vee c$	
0	0	1	0	1	—	$M_1 = a \vee b \vee \bar{c}$	$m_1 = \bar{a} \cdot \bar{b} \cdot c$	—	
0	1	0	0	1	—	$M_2 = a \vee \bar{b} \vee c$	$m_2 = \bar{a} \cdot b \cdot \bar{c}$	—	
0	1	1	1	0	$m_3 = \bar{a} \cdot b \cdot c$	—	—	$M_3 = a \vee \bar{b} \vee \bar{c}$	
1	0	0	0	1	—	$M_4 = \bar{a} \vee b \vee c$	$m_4 = a \cdot \bar{b} \cdot \bar{c}$	—	
1	0	1	1	0	$m_5 = a \cdot \bar{b} \cdot c$	—	—	$M_5 = \bar{a} \vee b \vee \bar{c}$	
1	1	0	1	0	$m_6 = a \cdot b \cdot \bar{c}$	—	—	$M_6 = \bar{a} \vee \bar{b} \vee c$	
1	1	1	1	1	$m_7 = a \cdot b \cdot c$	—	$m_7 = a \cdot b \cdot c$	—	

в таблице истинности. **Минтерм** — произведение всех аргументов, записанное в соответствии с двоичным набором, на котором функция равна единице, над аргументами, которые в наборе равны нулю, в произведении ставится инверсия.

Логические функции для сигналов  $p$  и  $s$  в СДНФ имеют вид:

$$p = \bar{a} \cdot b \cdot c \vee a \cdot \bar{b} \cdot c \vee a \cdot b \cdot \bar{c} \vee a \cdot b \cdot c;$$

$$s = \bar{a} \cdot \bar{b} \cdot c \vee \bar{a} \cdot b \cdot \bar{c} \vee a \cdot \bar{b} \cdot \bar{c} \vee a \cdot b \cdot c.$$

Для получения логических функций в СКНФ в скобках записываются все макстермы и соединяются знаками умножения. Число сомножителей равно числу строк в таблице истинности, для которых функция равна нулю. Каждый макстерм учитывает двоичный набор, на котором функция равна нулю, над аргументами, равными единице, в дизъюнкции ставится инверсия. Логические функции для сигналов  $p$ ,  $s$  в СКНФ имеют вид:

$$p = (a \vee b \vee c) \cdot (a \vee \bar{b} \vee c) \cdot (a \vee b \vee \bar{c}) \cdot (\bar{a} \vee b \vee c);$$

$$s = (a \vee b \vee c) \cdot (a \vee \bar{b} \vee \bar{c}) \cdot (\bar{a} \vee b \vee \bar{c}) \cdot (\bar{a} \vee \bar{b} \vee c).$$

**Минимизация логических функций.** Для минимизации логических функций необходимо построить карты Карно трех переменных для функций  $s$  и  $p$  (рис. 5.10). Число клеток в каждой карте Карно равно 8, что соответствует числу строк в таблице истинности (см. табл. 5.2).

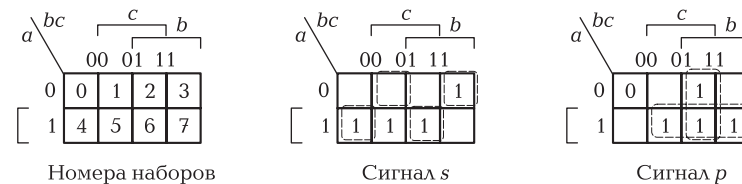


Рис. 5.10. Карты Карно трех переменных

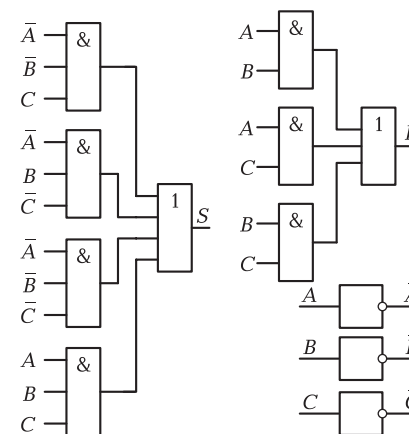


Рис. 5.11. Схема сумматора

В данном примере логическую функцию для сигнала  $s$  упростить не удастся, контуров на карте Карно нет, форма записи данной функции МДНФ. Для сигнала  $p$  имеем три контура, содержащие по две клетки, что позволяет упростить уравнение. Логические функции в МДНФ, описывающие сумматор, имеют следующий вид:

$$p = a \cdot b \vee a \cdot c \vee b \cdot c;$$

$$s = \bar{a} \cdot \bar{b} \cdot c \vee \bar{a} \cdot b \cdot \bar{c} \vee a \cdot \bar{b} \cdot \bar{c} \vee a \cdot b \cdot c.$$

Разработка схемы. Схема устройства (рис. 5.11) построена по минимизированным логическим функциям. В соответствии с постановкой задачи схема содержит три входных и два выходных терминала ( $a$ ,  $b$ ,  $c$ ,  $p$ ,  $s$ ).

Для формирования выходного сигнала  $s$  необходимы инверсные значения входных сигналов, для их формирования используются инверторы (в САПР это элементы NOT). В схеме использовано ло-

гическое соединение проводников. Проводники, имеющие одинаковые имена, соединены.

**Описания сумматора на языке Verilog.** При проектировании устройства для реализации на базе ПЛИС с использованием САПР исходные данные могут быть представлены в виде описания на HDL-языках, при этом схема устройства не нужна. Компиляторы САПР при обработке описаний схем выполняют оптимизацию схем. Проектирование схем в виде описаний становится все более и более перспективным. Описание одноразрядного сумматора по логическим уравнениям составлено в соответствии с постановкой задачи и символом элемента (пример 5.4).

```
// Пример 5.4. Описание сумматора 1
// по логическим функциям
module v54_sum (a,b,c,p,s);
input a,b,c;
output p,s;
assign p = a&b|a&c|b&c;
assign s = ~a&~b&c|~a&b&~c|
a&~b&~c|a&b&c;
endmodule
```

Строки 1 и 2 — комментарий — содержат русские буквы, начинаются с двух символов косая черта (знак деления на клавиатуре). Остальные строки для удобства ссылок пронумерованы, номера записаны как комментарии. При вводе описания в САПР нумерация строк не нужна.

Строка 3 — заголовок — содержит ключевое слово *module*, затем имя устройства и перечисление всех входных и выходных сигналов в круглых скобках через запятую. При записи данной строки использованы обозначения выводов, изображенные на символе устройства (см. рис. 5.9).

В строках 4, 5 описаны входные и выходные сигналы. После ключевого слова *input* перечисляются входные сигналы, разделенные запятой. Перечисление заканчивает точка с запятой. После слова *output* аналогично перечисляются все выходные сигналы.

В примере типы сигналов не указаны, все сигналы по умолчанию будут назначены как однобитовые типа *wire*. Входы и двунаправленные сигналы всегда должны иметь тип *wire*. Если сигналы должны иметь другие параметры (тип и разрядность), то после описания портов дополнительно приводится описание типов сигналов.

Строки 6 и 7 описывают выходные сигналы сумматора. Они содержат параллельные логические операторы непрерывного назначения, записанные в соответствии с логическими функциями. По-

рядок выполнения операций определяет их приоритет, изменение порядка не потребовалось, поэтому скобки в записи операторов отсутствуют. Строка 7 записана с переходом на новую строку, язык *Verilog* допускает такую запись. Символы пробела, табуляции и перевода строки допускаются в любом месте описания, кроме ключевых слов и имен переменных, которые разрывать нельзя.

Строка 8 — завершение описания — ключевое слово *endmodule*, после которого не должно быть никаких знаков препинания.

Другой вид описания устройств — поведенческое описание по алгоритму функционирования приведен в примере 5.5.

```
// Пример 5.5. Поведенческое
// описание сумматора
module v55_sum (a,b,c,p,s);
input a,b,c;
output p,s;
assign {p,s} = a + b + c;
endmodule
```

Алгоритм работы устройства поясняет рис. 5.12. В устройстве суммируется три слагаемых, а результат представляют две одно-разрядные переменные. Из таблицы истинности (см. табл. 5.2) следует, что сигналы *p* и *s* можно объединить в 2-разрядное число, равное сумме этих слагаемых. Алгоритм функционирования описывает строка 6, смысл которой в том, что 2-разрядное число, полученное объединением сигналов *p* и *s*, равно сумме всех входных слагаемых.

Поведенческое описание представляет собой устройство «черный ящик» и описывает алгоритм его функционирования или зависимость выходного сигнала от входных сигналов. Это описание, в основе которого лежит алгоритм, как правило, короче и проще в восприятии. Целесообразность использования поведенческого описания проявляется особенно наглядно при проектировании комбинационных схем средней и высокой сложности.

**Разработка и описание тестовых сигналов.** Проверка работоспособности называется *тестированием*, для выполнения которого на входы устройства подают тестовые сигналы.

Разработка тестовых сигналов выполняется с учетом функциональных особенностей устройства. Характерной особенностью сумматора является равнозначность всех его вхо-

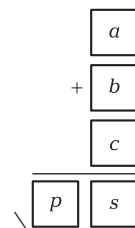


Рис. 5.12. Пояснение к примеру 5.5



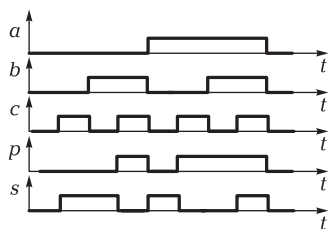


Рис. 5.13. Временные диаграммы работы сумматора

дов, входные сигналы сумматора — слагаемые — можно менять местами. Поэтому для полного тестирования сумматора необходимо выполнить перебор всех возможных комбинаций сигналов в любом порядке. Каждой комбинации входных сигналов будут соответствовать вполне определенные значения выходных сигналов  $p$  и  $s$ . По таблице истинности можно построить теоретические временные диаграммы (рис. 5.13).

## 5.4. ПАРАЛЛЕЛЬНЫЙ СУММАТОР С ПОСЛЕДОВАТЕЛЬНЫМ ПЕРЕНОСОМ

Для суммирования двоичных чисел, представленных параллельным двоичным кодом, используют схемы многоразрядных сумматоров, содержащие необходимое число одноразрядных сумматоров (рис. 5.14).

Если входной перенос отсутствует, то в младшем разряде используют полусумматор, который суммирует входные слагаемые и формирует сумму данного разряда и перенос в следующий разряд. В остальных разрядах использованы сумматоры для суммирования входных слагаемых и переноса из предыдущего разряда. В данной схеме используется последовательный перенос от разряда к разряду.

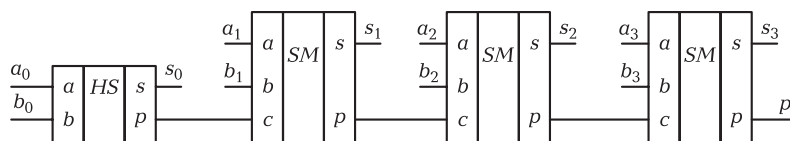


Рис. 5.14. Параллельный сумматор с последовательным переносом

ду, поэтому схема имеет ограниченное быстродействие, которое ухудшается с увеличением разрядности. Существуют схемы с ускоренным переносом, обеспечивающие повышение быстродействия.

Для построения схемы данного устройства необходимо использовать схемы одноразрядных полусумматора и сумматора. Процесс проектирования сокращается при использовании описания параллельного сумматора на языке Verilog (пример 5.6).

```
// Пример 5.6. Поведенческое опи-
// сание параллельного сумматора
module v56_4sum (a,b,s,p);           //3
input [3:0] a,b;                     //4
output [3:0] s;                       //5
output p;                             //6
assign {p,s} = a + b;                 //7
endmodule                             //8
```

Строки 1, 2 — комментарий. Строка 3 — заголовок — содержит имя описания (указывающее, что это 4-разрядный сумматор) и перечисление всех входных и выходных сигналов.

Строка 4 описывает входные сигналы  $a$  и  $b$  как 4-разрядные векторы (параллельные коды). Сигналы имеют одинаковую разрядность, поэтому они описаны в одной строке. Перед именами сигналов в квадратных скобках через двоеточие указан индекс старшего разряда, а затем индекс младшего разряда.

Рис. 5.15. Пояснение работы сумматора

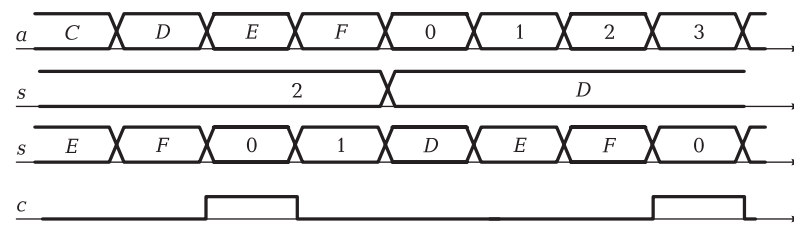
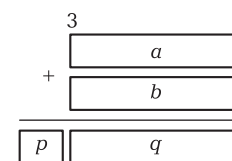


Рис. 5.16. Временные диаграммы, поясняющие работу параллельного сумматора

Строка 5 подобным образом описывает выходной сигнал *s*. Отдельной строкой описан выходной сигнал *p*. По умолчанию он будет назначен как одноразрядный сигнал типа *wire*.

Поведение или алгоритм функционирования описывает строка 7, смысл которой поясняет [рис. 5.15](#).

Для тестирования устройства необходимо разработать тестовые сигналы, один из вариантов которых показан на [рис. 5.16](#).

### 5.5. МУЛЬТИПЛЕКСОРЫ

Мультиплексор подключает один из информационных входов к выходу в соответствии с адресом, поданным на адресный вход. **Мультиплексоры** — это цифровые многопозиционные переключатели или коммутаторы. Слово «мультиплексирование» означает «переключение».

Мультиплексор — это комбинационная схема, имеющая *m* адресных входов,  $2^m$  информационных входов и один выход. На выход передается сигнал с того информационного входа, чей адрес в данный момент присутствует на адресных входах. Таким образом, мультиплексоры позволяют выбирать (селектировать) определенный канал.

**Мультиплексор 2 в 1.** Он является простейшим, имеет два входных сигнала: *x*, и *y*, и передает на выход *q* один из них в зависимости от значения адреса *a* ([рис. 5.17](#)).

Его работу описывают таблица истинности (табл. 5.3), составленная из следующих соображений. Сигнал *a* является адресом, он управляет подключением выхода к одному из входов. Поэтому при *a* = 0 выходной сигнал равен сигналу *y*, а при *a* = 1 — сигналу *x*. Таблице истинности и описанной логики работы соответствует логическое уравнение:

$$q = x \cdot a \vee y \cdot \bar{a}.$$

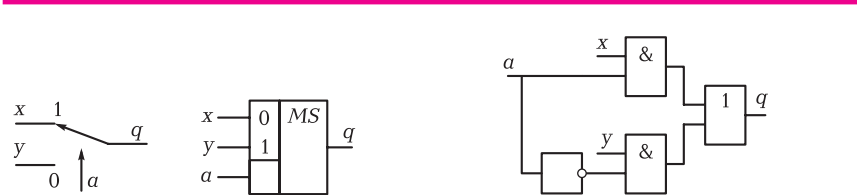


Рис. 5.17. Мультиплексор 2 в 1

Рис. 5.18. Схема мультиплексора 2 в 1

По уравнению несложно составить схему мультиплексора на логических элементах ([рис. 5.18](#)).

При реализации в ПЛИС мультиплексор можно синтезировать применительно к конкретной задаче, в соответствии с которой выбирается разрядность адреса, количество информационных входов и количество входов разрешения.

Описание мультиплексора на языке *Verilog* также можно выполнить двумя способами. Описание по логическому уравнению приведено в примере 5.7. Строка 1 — комментарий, содержащий русские буквы.

```
//Пример 5.7.                                     1
module v57_mux (a, x, y, q);//2
input a, x, y;                                     //3
output q;                                           //4
assign q = x & a | y & ~a; //5
endmodule                                           //6
```

Заголовок описания — строка 2 — содержит ключевое слово *module*, затем имя модуля и список всех входных и выходных сигналов (портов) в круглых скобках. Пробелы в именах недопустимы, а символ подчеркивания разрешается использовать. Конец данной строки и всех последующих строк и операторов — точка с запятой.

Описание портов — строки 3, 4. После ключевого слова *input* перечисляются входные порты, разделенные запятой. Перечисление заканчивается точкой с запятой. После слова *output* аналогично перечисляются все выходные порты.

В примере типы сигналов не указаны, все сигналы по умолчанию будут назначены как одноразрядные типа *wire*. Выходы комбинационных схем обычно принимают тип *wire*.

Описание схемы — строка 5 — начинается с ключевого слова *assign* (назначать). При записи логической функции использованы принятые в языке *Verilog* символы поразрядных операторов И, ИЛИ, НЕ. Завершение описания — строка 6 — это ключевое слово *endmodule*, после которого не должно быть никаких знаков препинания.

Описание записывают в файл, имя которого должно совпадать с именем модуля, а расширение должно быть *v*, что означает язык описания *Verilog*.

Таблица 5.3			
<i>a</i>	<i>x</i>	<i>y</i>	<i>q</i>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Поведенческое описание мультиплексора (пример 5.8) отличается от предыдущего именем и описанием схемы, в которой содержится оператор условного присваивания с ключевым словом *assign*. Формат данного оператора был рассмотрен в гл. 3. Строка 5 означает: если  $a = 1$ , то  $q = x$ , иначе  $q = y$ .

```
//Пример 5.8.                                     1
module v58_mux (a, x, y, q); //2
input a, b, s;                                     //3
output q;                                           //4
assign q = a ? x: y;                               //5
endmodule                                           //6
```

Сигналы для тестирования мультиплексора (рис. 5.19) составлены по табл. 5.3.

Рекомендуется выполнить моделирование мультиплексора применительно к реализации его в ПЛИС. Исходные данные проекта могут быть введены в виде схемы или в виде одного из вариантов описания на языке *Verilog*. Результаты моделирования в любом случае должны соответствовать теоретическим временным диаграммам.

Мультиплексор для коммутации шин. Шина представляет собой совокупность проводников, по которым одновременно передаются сигналы. Мультиплексор, переключаящий шины, должен подключить к выходным проводникам шины  $q$  проводники входной шины  $x$  или шины  $y$  в зависимости от адреса на входе  $a$  (рис. 5.20).

При построении схемы  $n$ -разрядного мультиплексора  $2$  в  $1$  необходимо использовать  $n$  одноразрядных мультиплексоров, на которые подается общий управляющий сигнал. Каждый проводник шины должен коммутироваться отдельным одноразрядным мультиплексором. Так, например, для коммутации 4-разрядных шин необходимо использовать четыре одноразрядных мультиплексора (рис. 5.21), каждый из которых выполнен по рассмотренной ранее схеме (см. рис. 5.18).

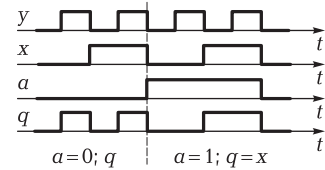


Рис. 5.19. Тестовые сигналы для мультиплексора

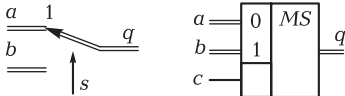


Рис. 5.20. Мультиплексор для шин

Таблица 5.4		
Количество разрядов	Количество наборов $m$	Количество функций $N$
1	$2^1 = 2$	$2^2 = 4$
2	$2^2 = 4$	$2^4 = 16$
3	$2^3 = 8$	$2^8 = 256$
4	$2^4 = 16$	$2^{16} = 65\,536$
5	$2^5 = 32$	$2^{32} = 4\,291\,367\,296$

Процесс проектирования мультиплексора для коммутации шин упрощается при использовании его описания на языке *Verilog* (пример 5.9).

```
//Пример 5.9.                                     1
module v59_mux_bus (a, x, y, q); //2
input a;                                           //3
input [3:0] x, y;                                 //4
output [3:0] q;                                   //5
assign q = a ? x: y;                             //6
endmodule                                         //7
```

Строки 4, 5 содержит описание векторов (шин), остальные строки совпадают с примером 5.8. Описание переключения многоразрядных сигналов  $x$  и  $y$  под управлением одноразрядного сигнала  $a$  выполнено посредством оператора условного присваивания в строке 6.

Универсальный логический элемент на основе мультиплексора. Аппаратная реализация универсальных логических элементов в ПЛИС сопряжена с определенными сложностями, заключающимися в резком возрастании количества вариантов функций при

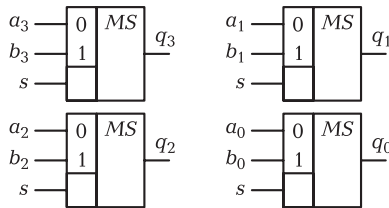


Рис. 5.21. Схема мультиплексора для 4-разрядных шин

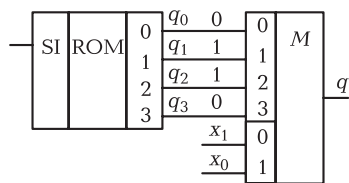


Рис. 5.22. Универсальный логический элемент

Мультиплексоры позволяют реализовать логические функции, число переменных которых равно или несколько превышает разрядность адресного кода. Выполняемая функция должна быть представлена в СДНФ. При этом переменные поступают на адресные входы, а информационные входы используются как настроечные, на них подаются константы нуля и единицы в соответствии с реализуемой функцией.

На рис. 5.22 изображена схема универсального логического элемента двух переменных. Значения входных переменных или аргументов подаются на адресные входы мультиплексора, а на входы данных — значения функции на всех возможных наборах. В схеме используется одноразрядное ПЗУ, программируемое последовательным кодом по входу SI. При значениях выходных сигналов ПЗУ, показанных на схеме, данный элемент будет выполнять функцию  $q = x_1 \oplus x_0$ .

Мультиплексор — один из базовых элементов в схемотехнике ПЛИС. Рассмотренная схема используется в качестве основного логического элемента в ПЛИС и называется *логическая таблица*, или *look up table (LUT)*.

## 5.6. ДЕШИФРАТОРЫ

Система электрических сигналов, используемая для передачи сообщений, называется **кодом**. Устройства, преобразующие одну разновидность кода в другую, называются **преобразователями кодов**. К преобразователям кодов принадлежат **дешифраторы** и **шифраторы**. Условные обозначения дешифраторов и шифраторов содержат буквы DC и CD (от слов *decoder* и *coder*).

Самым распространенным является дешифратор двоичного кода в единичный код, в котором на выходе только один из разрядов

увеличении разрядности. Логическая функция  $n$  переменных имеет  $m$  наборов, где  $m = 2^n$ . Для каждого набора существует  $N$  вариантов значений функции, где  $N = 2^m$ . С ростом разрядности  $n$  число  $N$  растет очень сильно (табл. 5.4). Даже для 4-входового элемента число функций велико, и реализовать аппаратно полный набор логических элементов для всех функций затруднительно.

равен 1. Подобный код называется также унарный или «1 из  $N$ ». Полным дешифратором называется устройство, имеющее  $N$  входов и  $2^N$  выходов, причем каждой комбинации значений входных сигналов соответствует сигнал, равный 1 только на одном выходе.

Дешифратор, дополненный разрешающими входами, называется *дешифратором-демультиплексором*. На рис. 5.23 приведен символ дешифратора — демультиплексора, имеющего 2-разрядный входной код ( $d_1, d_0$ ), один разрешающий вход  $e$ , и 4-разрядный выходной код ( $q_3, q_2, q_1, q_0$ ).

При построении таблицы истинности дешифратора — демультиплексора (табл. 5.5) учитывается физика его работы. Для того чтобы входные сигналы  $d_1, d_0$  вызвали изменения выходных сигналов  $q_3 \dots q_0$ , необходимо подать разрешающий сигнал  $e = 1$ . В этом случае на одном из выходов, номер которого определяет адрес, формируется сигнал, равный 1. При  $e = 0$  все выходные сигналы запрещены и равны нулю.

В столбце *hex* приведены значения выходного кода в шестнадцатеричной системе, что будет полезно при моделировании устройства и оценке экспериментальных данных, которые чаще представляются в шестнадцатеричной системе с теоретическими.

Таблица истинности позволяет записать логические функции для каждого выходного сигнала:

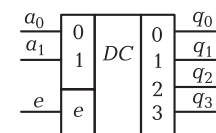


Рис. 5.23. Дешифратор-демультиплексор

Таблица 5.5

$e$	$d_1$	$d_0$	$q_3$	$q_2$	$q_1$	$q_0$	hex
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	0	0	0	1	1
1	0	1	0	0	1	0	2
1	1	0	0	1	0	0	4
1	1	1	1	0	0	0	8

$$q_0 = e \cdot \bar{d}_1 \cdot \bar{d}_0; \quad q_1 = e \cdot \bar{d}_1 \cdot d_0; \quad q_2 = e \cdot d_1 \cdot \bar{d}_0; \quad q_3 = e \cdot d_1 \cdot d_0.$$

По данным формулам несложно составить схему (рис. 5.24) и описание на Verilog (пример 5.10). В схеме дешифратора-демультиплексора для формирования каждого выходного сигнала используется отдельный логический элемент.

```
//Пример 5.10. Дешифратор-
// демультиплексор
module v510_dc_dmux (e, a, q);
input e;
input [1:0] d;
output [3:0] q;
assign q[3] = d[1] & d[0];
assign q[2] = & d[1] & ~d[0];
assign q[1] = ~d[1] & d[0];
assign q[0] = ~d[1] & ~d[0];
endmodule
```

Наличие дополнительных входов разрешения, которых может быть несколько, позволяет разрешать или запрещать работу устройства, расширяет функциональные возможности. В названии схемы отражена возможность ее применения в качестве демультиплексора, что поясняет рис. 5.25.

Демультиплексоры выполняют функцию, обратную мультиплексорам, они производят коммутацию одного информационного входного сигнала на  $2^m$  выходов, где  $m$  — число адресных входов. При этом в качестве информационного входа демультиплексора используется разрешающий вход  $e$ .

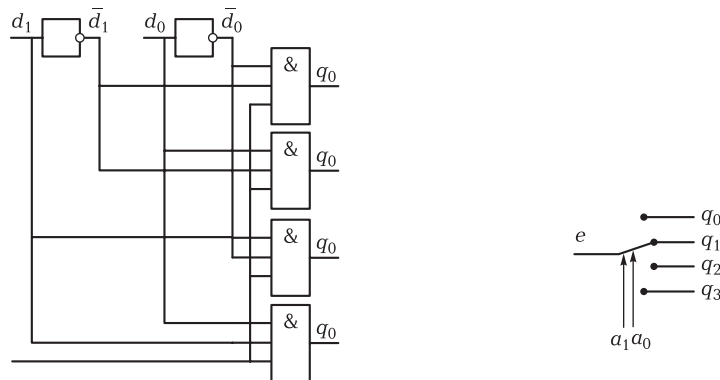


Рис. 5.24. Схема дешифратора 2 в 4

Рис. 5.25. Пояснение работы демультиплексора

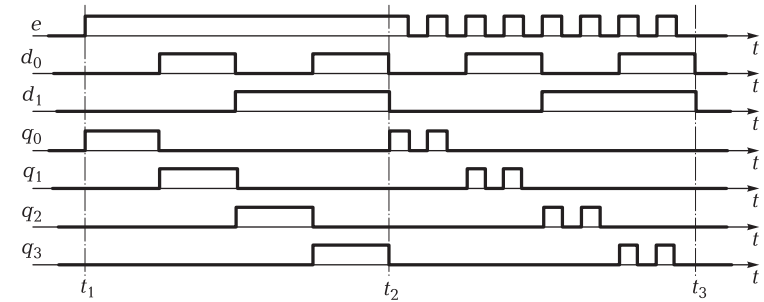


Рис. 5.26. Временные диаграммы работы дешифратора-демультиплексора

Работу дешифратора-демультиплексора поясняют временные диаграммы (рис. 5.26). Началу диаграмм соответствует сигнал  $e = 0$ , при этом все выходные сигналы равны нулю. Интервал времени  $t_1...t_2$  иллюстрирует работу устройства в качестве дешифратора. В этом случае сигнал, равный 1, появляется на одном выходе, адрес которого определяет входной код  $d$ . Интервал времени  $t_2...t_3$  иллюстрирует работу устройства в качестве демультиплексора, сигнал, поданный на вход  $e$ , передается на один из выходов, адрес которого определяет код  $d$ .

Выходы и разрешающие входы дешифраторов, выполненных в виде интегральных микросхем, могут быть инверсными. Так, например, дешифратор-демультиплексор с инверсными выходами и инверсным разрешающим входом  $\overline{CS}$  формирует при  $\overline{CS} = 0$  нулевой логический сигнал на одном из выходов, в соответствии с адресом на входах. При  $\overline{CS} = 1$  все выходные сигналы равны 1.

## 5.7. КОМПАРАТОРЫ КОДОВ

**Компаратор кодов** — схема сравнения кодов чисел. Основными отношениями между двумя двоичными кодами чисел (словами)  $a$  и  $b$ , через которые можно выразить все остальные, являются «равно» (обозначим как  $e$ ) и «больше» (обозначим как  $g$ ). Признак  $e = 1$  означает равенство чисел  $a = b$ , следовательно,  $e = 0$  означает «не равно». Признак  $g = 1$  означает «больше»:  $a > b$ , а  $g = 0$  — «не больше» или «меньше или равно». Если имеем признаки  $g = 0$  и  $e = 0$ , то получим отношение «меньше». В интегральном исполнении существуют компараторы кодов, имеющие три выхода: «меньше», «равно», «больше».



Признака равенства  $e$  чисел  $a$  и  $b$  можно получить при помощи комбинационных схем, которые выполняют поразрядную операцию исключающее ИЛИ (Сумма по модулю 2) и из результатов сравнения отдельных разрядов формируют общий результат. Признаки равенства отдельных разрядов определяют очевидные формулы:

$$e_3 = \overline{(a_3 \oplus b_3)}; \quad e_2 = \overline{(a_2 \oplus b_2)}; \quad e_1 = \overline{(a_1 \oplus b_1)}; \quad e_0 = \overline{(a_0 \oplus b_0)}.$$

Признак  $e$  равенства четырехразрядных чисел  $a$  и  $b$ , принимающий значение 1 при равенстве чисел, можно записать в виде формулы

$$e = \overline{(a_3 \oplus b_3)} \cdot \overline{(a_2 \oplus b_2)} \cdot \overline{(a_1 \oplus b_1)} \cdot \overline{(a_0 \oplus b_0)}.$$

Полученную формулу можно записать в другом виде, используя закон де Моргана:

$$e = \overline{(a_3 \oplus b_3) \vee (a_2 \oplus b_2) \vee (a_1 \oplus b_1) \vee (a_0 \oplus b_0)}.$$

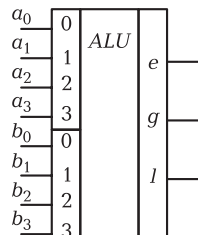
Данная логическая функция позволяет составить схему, формирующую признак равенства кодов, а также описание на языке Verilog (см. примеры 3.1 и 4.1).

Рассмотрим построение схемы сравнения положительных чисел, представленных в прямом коде без знака, которая формирует признак  $g$ . Процесс сравнения кодов чисел следует начинать со старшего разряда, что упростит разработку схемы.

Признак  $g$ , соответствующий отношению  $a > b$  будет равен 1, если в старшем разряде числа  $a$  записана 1, а в старшем разряде числа  $b$  записан 0, при этом остальные разряды на результат уже не влияют.

Если старшие разряды равны, то значение признака будут определять следующие разряды:  $g = 1$ , если следующий разряд числа  $a$  — единица, а числа  $b$  — ноль. Рассуждая таким образом можно рассмотреть все разряды чисел и записать логическую функцию:

$$g = a_3 \cdot \overline{b_3} \vee \overline{(a_3 \oplus b_3)} \cdot a_2 \cdot \overline{b_2} \vee \overline{(a_3 \oplus b_3)} \cdot \overline{(a_2 \oplus b_2)} \times \\ \times a_1 \cdot \overline{b_1} \vee \overline{(a_3 \oplus b_3)} \cdot \overline{(a_2 \oplus b_2)} \cdot \overline{(a_1 \oplus b_1)} \cdot a_0 \cdot \overline{b_0}.$$



Приведенные формулы позволяют составить схему, а также описание компаратора кодов, формирующего признак  $g$  — «больше».

Функциональные возможности языка Verilog позволяют составить компактное и наглядное по-

Рис. 5.27. Компаратор кодов

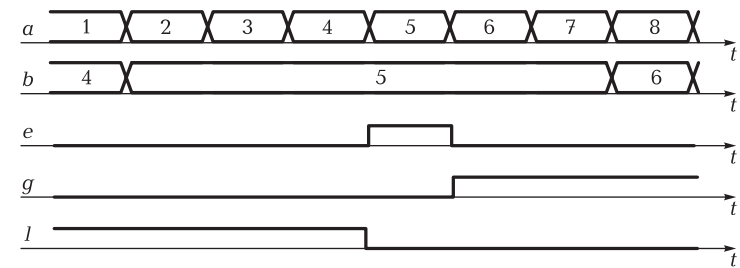


Рис. 5.28. Временные диаграммы, поясняющие работу компаратора кодов

веденческое описание компараторов кодов, в которых используются операторы сравнения, описанные в гл. 3. Операторы сравнения формируют одноразрядные сигналы — результат сравнения векторов (параллельных кодов).

Поведенческое описание компаратора, формирующего три основных признака отношения кодов — «меньше», «равно» и «больше» (пример 5.11), составлено в соответствии с обозначением (рис. 5.27). Входные сигналы компаратора — это 4-разрядные коды  $a$  и  $b$ , а выходные — признаки отношения кодов: равно  $e$ , больше  $g$ , меньше  $l$ .

```
//Пример 5.11. Полный
//компаратор
module v514_compar (a, b, e, g, l);
    input [3:0] a, b;
    output e, g, l;
    assign e = a == b;
    assign g = a > b;
    assign l = a < b;
endmodule
```

Теоретические временные диаграммы, поясняющие работу компаратора кодов, описанного в примере 5.11, приведены на рис. 5.28.

## 5.8. АЛУ КОМБИНАЦИОННОГО ТИПА

Ядром любого процессора является **арифметико-логическое устройство**, выполняющее математическую обработку данных. Входами АЛУ являются многоразрядные операнды  $a$ ,  $b$  и код операции  $k$ , а выходами — код результата  $q$  и признаки  $F$  (рис. 5.29).

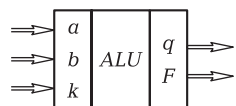


Рис. 5.29. Условное обозначение АЛУ

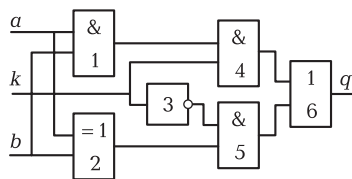


Рис. 5.30. Схема логического устройства

Рассмотрим проектирование простого одноразрядного логического устройства комбинационного типа, имеющего входы данных  $a$ ,  $b$  и вход кода операции  $k$ . Пусть сигнал на выходе  $q$  определяется следующим образом:

$$q = a \cdot b \text{ при } k = 0; q = a \oplus b \text{ при } k = 1.$$

Для выполнения заданных операций в устройстве (рис. 5.30) использованы отдельные логические элементы (элементы 1 и 2), на которые подаются параллельно входные сигналы. Но в качестве выходного сигнала посредством мультиплексора (элементы 3...6) выбирается сигнал только одного из логических элементов в соответствии с кодом операции.

Наиболее целесообразным методом проектирования АЛУ комбинационного типа является его описание на языке *Verilog* по алгоритму функционирования (поведенческое описание).

Рассмотрим проектирование АЛУ в виде описания на языке *Verilog* в соответствии с следующим заданием.

Разрядность входных и выходных сигналов — 4.

Количество выполняемых операций — четыре операции.

Типы и коды операций: 0 — суммирование, 1 — поразрядная логическая операция И, 2 — поразрядная логическая операция ИЛИ, 3 — поразрядная логическая операция Сумма по модулю 2.

Формируемые признаки результата: устройство должно формировать признак переноса  $p$  при выполнении суммирования и вычитания.

Обозначение 4-разрядного АЛУ приведено на рис. 5.31. Устройство имеет 4-разрядные входы  $a$ ,  $b$ , 4-разрядный выход  $q$ , а также двухразрядный код операции  $k$ .

В описании АЛУ (пример 5.12) строка 2 — заголовок — содержит имя описания и перечисление всех входов и выходов. В строке 3 описаны 4-разрядные входные векторы  $a$ ,  $b$ , в строке 4 описан 2-разрядный входной вектор кода операции  $k$ , а в строке 5 — выходной вектор  $q$ . В строке 6 указан сигнал переноса  $p$ , для которо-

го описание диапазона отсутствует, следовательно, этот сигнал — одноразрядный. Все сигналы по умолчанию будут иметь тип *wire*. С сигналами типа *wire* обычно используют оператор непрерывного назначения с ключевым словом *assign*, а также арифметические, логические операторы и оператор условного присваивания.

```
//Пример 5.12. АЛУ
module v512_alu (a,b,k,q, p); //2
input [3:0] a, b; //3
input [1:0] k; //4
output [3:0] q; //5
output p; //6
assign {p,q} = //7
(k == 0) ? (a + b); //8
(k == 1) ? (a & b); //9
(k == 2) ? (a | b); //10
endmodule //11
```

В строках 7...10 записан оператор условного присваивания, содержащий вложенные условные операторы, который в зависимости от кода операции выполняет определенные действия в соответствии с алгоритмом, представленном на рис. 5.32. Для улучшения восприятия оператор разделен на несколько строк, что не влияет на компиляцию.

Результат суммирования или вычитания 4-разрядных чисел содержит в общем случае пять разрядов, так как может появиться перенос. Сигнал, формируемый данным оператором, записан (в строке 7) как объединение сигналов переноса  $p$  и выходного вектора  $q$ , они записаны через запятую в фигурных скобках. Этот сигнал — вектор суммы или разности, старший бит которого равен 1, если возникает перенос.

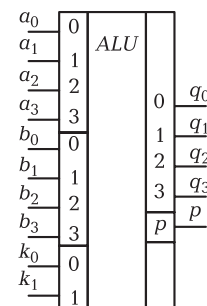


Рис. 5.31. Символ 4-разрядного АЛУ

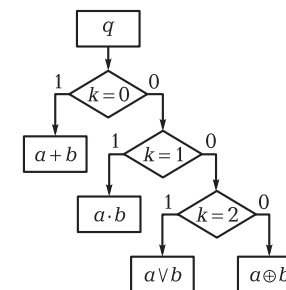


Рис. 5.32. Алгоритм выбора операции

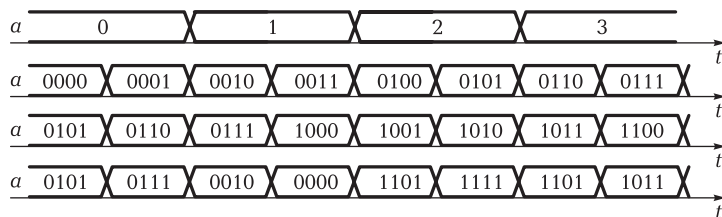


Рис. 5.33. Временные диаграммы, поясняющие работу АЛУ

Для проверки работоспособности выполняют тестирование устройства, при котором на входы устройства подают тестовые сигналы. Разработка тестовых сигналов выполняется с учетом функциональных особенностей устройства.

При построении временных диаграмм (рис. 5.33) выполнен полный перебор кодов операции и выбраны некоторые комбинации входных сигналов, для которых вычислены показанные на диаграммах значения выходного сигнала  $q$ .

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сформулируйте функциональное назначение, укажите входные и выходные сигналы полусумматора.
2. Как составляется таблица истинности и записываются логические функции для полусумматора?
3. Составьте поведенческое описание полусумматора на языке *Vereilog*.
4. Составьте описание сумматора по логическим уравнениям.
5. Составьте поведенческое описание сумматора на языке *Vereilog*.
6. Как составить тестовые временные диаграммы для сумматора?
7. Сформулируйте функциональное назначение и приведите схему инкрементатора.
8. Изобразите схему дешифратора, имеющего три входа и восемь выходов.
9. Постройте схему демультиплексора с двумя выходами.
10. Как составляется таблица истинности и логические функции для мультиплексора?
11. Составьте поведенческое описание мультиплексора на языке *Vereilog*.
12. Составьте описание дешифратора по логическим уравнениям.
13. Как составляется поведенческое описание компараторов кодов?

## Глава 6

### ТРИГГЕРЫ

**Триггером** называется устройство с двумя устойчивыми состояниями, переходящее из одного состояния в другое под воздействием внешних сигналов. Триггер способен сохранять устойчивое состояние сколь угодно долго после снятия внешнего воздействия. Он используется в качестве элемента памяти для хранения 1 бит информации.

В отличие от комбинационных логических цепей цифровые устройства, содержащие элементы памяти — триггеры, называются *последовательностными устройствами*, или *автоматами с памятью*.

Основу любого триггера составляет запоминающая ячейка, состоящая из двух инвертирующих элементов, соединенных в замкнутую цепь, в которой действует положительная обратная связь. За счет наличия положительной обратной связи схема имеет два устойчивых состояния.

По способу записи информации различают *асинхронные* и *синхронные триггеры*. В асинхронных триггерах запись нуля или единицы возможна в любой момент времени, при этом входной информационный сигнал является управляющим, при его изменении происходит запись. В синхронных триггерах момент записи определяет сигнал синхронизации  $s$  (от слова *clock*).

По моменту реакции триггера на сигнал синхронизации различают триггеры со *статическим управлением* — по уровню синхросигнала, и с *динамическим управлением* — по фронту или спаду синхросигнала.

По виду входных сигналов различают следующие типы триггеров: *RS*, *D*, *JK*, *T*.

## 6.1. АСИНХРОННЫЙ RS-ТРИГГЕР С ПРЯМЫМИ УСТАНОВОЧНЫМИ ВХОДАМИ

**RS-триггером** называется запоминающий элемент с прямыми установочными входами  $R$  и  $S$ , выполненный на элементах ИЛИ—НЕ (рис. 6.1). Установку триггера в состояние 1 выполняет сигнал, поданный на вход  $s$  (от слова *Set* — установка), а в состояние 0 — сигнал с входа  $r$  (от слова *reset* — сброс). Схема имеет два выхода: прямой выход  $q$  и инверсный выход  $\bar{q}$ .

Для того чтобы триггер обладал свойством хранения состояния, в его схеме должна существовать положительная обратная связь. Такая связь возникает за счет того, что сигнал обратной связи возвращается в исходную точку схемы после двух операций инверсии и имеет такое же значение, как и исходный сигнал. В результате это значение сигнала фиксируется.

Для рассмотрения сигналов в схеме триггера вспомним логику работы элементов ИЛИ—НЕ. На выходе элемента ИЛИ—НЕ сигнал равен 1 только в одном случае, когда на все входы подан 0. Для того чтобы один из входов элемента был активным и передавал на выход изменения сигнала, необходимо подать на остальные входы сигналы 0. Поэтому для получения в триггере режима хранения данных необходимо, чтобы сигналы обратной связи передавались по замкнутому контуру (который напоминает цифру 8). Для схемы ИЛИ—НЕ это условие выполняется при подаче на управляющие входы сигналов, равных 0. Режим хранения в RS-триггере обеспечивается при  $r = s = 0$ . В таблице истинности (табл. 6.1) в этом режиме новое состояние триггера  $q_{n+1}$  повторяет исходное состояние  $q_n$ . Если принять исходное состояние триггера равным 0, то можно определить все сигналы в схеме в режиме хранения.

Сигнал  $q = 0$  поступает на вход элемента  $D2$ . Комбинация сигналов на входах элемента  $D2$  будет 00. Сигнал на выходе этого элемента равен 1, в результате на входах элемента  $D1$  получается комбинация сигналов 01, которая создает сигнал  $q = 0$ . В результате обхода контура получен сигнал  $q = 0$ , равный исходному, что подтверждает существование режима хранения.

Сигнал  $r = 1$  устанавливает  $q = 0$ , а сигнал  $s = 1$  устанавливает  $q = 1$ . Работу триггера поясняют временные диаграммы (рис. 6.2).

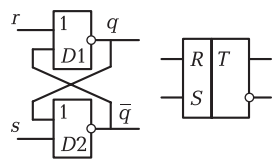


Рис. 6.1. Схема и обозначение RS-триггера

Таблица 6.1

$s$	$r$	$q_n$	$q_{n+1}$	Режим
0	0	0	0	Хранение
0	0	1	1	
0	1	0	0	Установка 0
0	1	1	0	
1	0	0	1	Установка 1
1	0	1	1	
1	1	0	x	Запрещенная комбинация
1	1	1	x	

Началу диаграмм — момент времени  $t_0$  — соответствует комбинация входных сигналов  $r = s = 0$ . Это режим хранения.

Установку триггера в состояние 1 выполняет сигнал  $s = 1$  при сохранении  $r = 0$  (момент времени  $t_1$ ). В этом случае сперва на выходе элемента  $D2$  установится  $\bar{q} = 0$ , а затем этот сигнал поступит на вход элемента  $D1$ , на выходе которого установится  $q = 1$ . Сигнал с выхода элемента  $D1$ , равный 1, поступит на второй вход элемента  $D2$ , в результате новое состояние триггера будет зафиксировано, и входной сигнал  $s$  можно вернуть в состояние 0.

В момент времени  $t_2$  вновь подается сигнал  $s = 1$ , при этом триггер, ранее установленный в состояние 1, сохраняет это состояние. Повторную установку триггера в то же самое состояние называют «подтверждение состояния».

В момент  $t_3$  выполняется установка триггера в состояние  $q = 0$  сигналом  $r = 1$  при сохранении сигнала  $s = 0$ . В этом случае сначала на выходе элемента  $D1$  установится сигнал  $q = 0$ , а затем на выходе элемента  $D2$  установится  $\bar{q} = 1$ .

При подаче сигнала 1 на оба входа  $r = s = 1$  (момент времени  $t_4$ ) возникает ситуация, когда триггер должен установиться в состояние 1 и в состояние 0 одновременно. Такая комбинация входных сигналов

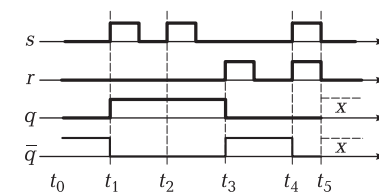


Рис. 6.2. Временные диаграммы, поясняющие работу RS-триггера

является запрещенной. Она вызывает появление на прямом и инверсном выходах триггера одинаковых сигналов, равных 0, что противоречит тождествам алгебры логики. После снятия запрещенной комбинации входных единичных сигналов (момент  $t_5$ ) схема переходит в неопределенное состояние  $x$ .

6.2.

АСИНХРОННЫЙ  $\overline{R}\overline{S}$ -ТРИГГЕР С ИНВЕРСНЫМИ УСТАНОВОЧНЫМИ ВХОДАМИ

Для построения  $\overline{R}\overline{S}$ -триггера элементы И—НЕ соединяются в замкнутую цепь (рис. 6.3). Установочные входы в этом триггере являются инверсными:  $\overline{S}$  — установочный вход единицы,  $\overline{R}$  — установочный вход нуля. Знак инверсии означает, что активным состоянием сигналов является 0. Схема имеет два выхода: прямой —  $q$ , инверсный —  $\overline{q}$ .

Работу триггера описывает таблица переходов (табл. 6.2). Рассмотрим сигналы в схеме, учитывая обратные связи и логику работы элементов. На выходе элемента И—НЕ только в одном случае сигнал равен 0, когда на оба его входа подан сигнал 1. Для того чтобы один из входов элемента был активным и передавал на выход изменения входного сигнала, необходимо на другой вход подать сигнал 1.

Таблица 6.2

$\overline{s}$	$\overline{r}$	$qn$	$qn + 1$	Режим
1	1	0	0	Хранение
1	1	1	1	
0	1	0	1	Установка 1
0	1	1	1	
1	0	0	0	Установка 0
1	0	1	0	
0	0	0	$x$	Запрещенная комбинация
0	0	1	$x$	

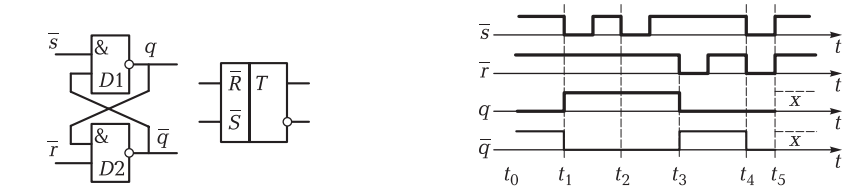


Рис. 6.3. Схема и обозначение  $\overline{R}\overline{S}$ -триггера

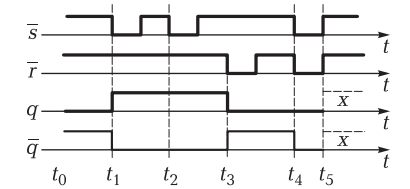


Рис. 6.4. Временные диаграммы, поясняющие работу  $\overline{R}\overline{S}$ -триггера

Режим хранения в  $\overline{R}\overline{S}$ -триггере обеспечивается при сигналах  $\overline{s} = \overline{r} = 1$  (момент времени  $t_0$  на временных диаграммах). При этом обеспечивается передача сигналов положительной обратной связи. Если принять  $q = 0$ , то комбинация входных сигналов элемента  $q_2$  с учетом выходного сигнала элемента  $q_1$  будет 10, поэтому на выходе элемента  $q_2$  в соответствии с логикой его работы, получим  $\overline{q} = 1$ . Этот сигнал, поступая на вход элемента  $D1$ , создаст комбинацию входных сигналов 11, в соответствии с которой на выходе этого элемента формируется  $q = 1$ . Сигнал, поступивший по цепи обратной связи, совпадает с исходным сигналом. В схеме действует положительная обратная связь, обеспечивающая сохранение устойчивых состояний. Подобным образом можно пояснить хранение сигнала  $q = 1$ .

Работу триггера поясняют временные диаграммы (рис. 6.4), исходное состояние (момент времени  $t_0$ ) соответствует режиму хранения.

Установку триггера в положение 1 выполняет отрицательный импульс на входе  $\overline{s} = 0$  (момент времени  $t_1$ ), устанавливающий на выходе элемента  $D1$  сигнал  $q = 1$ , который вызывает появление сигнала  $\overline{q} = 0$  на выходе элемента  $D2$ . Сигнал  $\overline{q}$ , в свою очередь, поступает по цепи обратной связи на вход верхнего элемента. Поэтому после окончания импульса на входе  $\overline{s}$  состояние схемы сохранится.

Повторная подача отрицательного импульса установки единицы  $\overline{s} = 0$  в момент времени  $t_2$  подтверждает ранее установленное состояние.

Установка триггера в положение 0 выполняется отрицательным импульсом  $\overline{r} = 0$  (момент времени  $t_3$ ), который устанавливает вначале выходной сигнал элемента  $D2$ , равный 1 ( $\overline{q} = 1$ ), а затем выходной сигнал элемента  $D1$ , равный 0 ( $q = 0$ ).



Запрещенная комбинация входных сигналов возникает при одновременной подаче отрицательных импульсов на входы  $\bar{s}$  и  $\bar{r}$  (момент времени  $t_4$ ). В момент действия этих импульсов на обоих выходах будут единичные сигналы. После окончания входных импульсов выходы случайным образом попадут в одно из двух устойчивых состояний, состояние триггера является неопределенным (момент времени  $t_5$ ). Случайным образом устанавливается одно из двух устойчивых состояний триггера и при включении питания.

Отметим, что данный триггер в технологии ТТЛ имеет наиболее удачное схемотехническое решение и традиционно используется в качестве элементарной запоминающей ячейки при построении более сложных схем.

### 6.3. СИНХРОННЫЙ RS-ТРИГГЕР

Обработка логических сигналов в цифровых устройствах сопровождается переходными процессами и временными задержками. Эти явления могут привести к ошибкам, которые называют *гонками*, или *состязаниями*. Для устранения ошибок используют синхронизацию записи данных, при которой запись сигнала выполняется в определенный момент времени, когда все переходные процессы в цифровой схеме закончились. Синхронный RS-триггер позволяет осуществлять запись сигналов в определенные моменты времени.

Взаимодействие отдельных элементов цифровых устройств согласуется по времени с помощью импульсов синхронизации. Синхронные устройства имеют специальный вход для подачи импульсов синхронизации  $c$ .

Синхронный RS-триггер (рис. 6.5) содержит запоминающую ячейку (элементы  $D3, D4$ ) и цепь управления записью данных (элементы  $D1, D2$ ). Схема имеет вход синхронизации  $c$ , прямые установочные входы  $r$  и  $s$ , а также прямой и инверсный выходы  $q$  и  $\bar{q}$ .

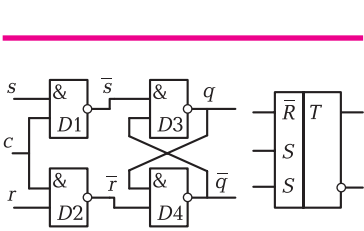


Рис. 6.5. Схема и обозначение синхронного RS-триггера

Работу синхронного RS-триггера поясняют таблица переходов (табл. 6.3) и временные диаграммы (рис. 6.6).

Режим хранения сигнала  $q = 0$  (моменты времени  $t_0, t_1, t_3$ ) обеспечивается либо при нулевом значении синхросигнала  $c = 0$ , при этом сигналы  $r$  и  $s$  не влияют, либо при нулевых сигналах на установочных входах

Таблица 6.3

$c$	$s$	$r$	$q_n$	$q_{n+1}$	Режим
0	x	x	0	0	Хранение
0	x	x	1	1	
1	0	0	0	0	
1	0	0	1	1	
1	1	0	0	1	Установка 1
1	1	0	1	1	
1	0	1	0	0	Установка 0
1	0	1	1	0	
1	1	1	0	X	Запрещенная комбинация
1	1	1	1	X	

$r = s = 0$ . В этом режиме на входах запоминающей ячейки формируются сигналы  $\bar{s} = \bar{r} = 1$ , что соответствует для данной ячейки режиму хранения.

Запись данных, поступающих на информационные входы  $r$  и  $s$ , происходит, если сигнал синхронизации  $c = 1$ . Момент записи сигналов, поданных на установочные входы  $r$  и  $s$ , определяет сигнал синхронизации  $c$ . Управление уровнем сигнала называется **статическим управлением**.

Установка триггера в состояние  $q = 1$  происходит при разрешающем сигнале  $c = 1$ , если сигнал  $s = 1$  (момент  $t_2$ ). Далее на диаграммах показан режим подтверждения состояния. Это повторная подача сигнала установки 1, не изменяющая состояния (момент  $t_4$ ).

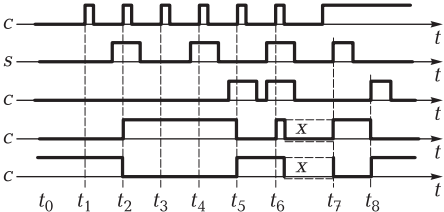


Рис. 6.6. Временные диаграммы, поясняющие работу синхронного RS-триггера

Установка триггера в состояние  $q = 0$  происходит при разрешающем сигнале  $c = 1$ , если сигнал  $r = 1$  при сохранении сигнала  $s = 0$  (момент  $t_3$ ).

Для данного триггера, как и для предыдущих, существует запрещенная комбинация входных сигналов:  $s = r = c = 1$ , при которой на входы запоминающей ячейки ( $\overline{R}\overline{S}$ -триггера) поступают сигналы  $\overline{s} = \overline{r} = 0$ . Оба выхода триггера в этом случае переходят в состояние 1. Дальнейшее непредсказуемое состояние на диаграммах обозначено как  $x$ .

При подаче постоянного сигнала  $c = 1$  триггер переходит в так называемый **прозрачный режим** (моменты  $t_7, t_8$ ). За время, пока  $c = 1$ , переключение триггера может быть многократным. В этом случае элементы схемы  $D1$  и  $D2$  работают как инверторы и устройство превращается в асинхронный  $RS$ -триггер с прямыми входами установки единицы  $s$  и установки нуля  $r$ .

6.4. СТАТИЧЕСКИЙ D-ТРИГГЕР

Основу схемы  $D$ -триггера (рис. 6.7) составляет ранее рассмотренный синхронный  $RS$ -триггер со статическим управлением (см. рис. 6.5), поэтому  $D$ -триггер также имеет статическое управление. Триггер имеет единственный информационный вход  $d$ . Благодаря инверсии, которую выполняет логический элемент  $D5$ , запрещенная комбинация сигналов на входах элементов  $D1$  и  $D2$  (когда все входные сигналы равны 1) становится невозможной.

Работу  $D$ -триггера поясняют таблица переходов (табл. 6.4) и временные диаграммы (рис. 6.8).

В режиме хранения (момент времени  $t_0$ ) принято состояние  $q = 0$ . Данный режим обеспечивается, если сигнал синхронизации  $c = 0$ . При этом изменение сигнала на входе  $d$  не передается на выход триггера, поэтому сигнал  $q$  не изменяется. В режиме хранения на

Таблица 6.4				
$c$	$d$	$q_n$	$q_{n+1}$	Режим
0	$x$	0	0	Хранение
0	$x$	1	1	
1	1	0	1	Установка 1
1	1	1	1	
1	0	0	0	Установка 0
1	0	1	0	

выходах элементов  $D1$  и  $D2$  формируются сигналы  $\overline{s} = \overline{r} = 1$ , и элементарный триггер, выполненный на элементах  $D3, D4$ , находится в режиме хранения.

Установка триггера в состояние  $q = 1$  выполняется при подаче сигналов  $d = 1$  и  $c = 1$  (момент времени  $t_1$ ). В результате формируется сигнал  $\overline{s} = 0$ , который установит выходные сигналы  $q = 1, \overline{q} = 0$ . Состояние триггера сохранится в следующем такте синхронизации (момент времени  $t_2$ ), так как при  $c = 1$  на входе триггера сохраняется сигнал  $d = 1$ .

Установка триггера в состояние  $q = 0$  выполняется при подаче сигналов  $d = 0$  и  $c = 1$  (момент времени  $t_3$ ).

Прозрачный режим триггера включается в момент времени  $t_4$ . Если установить сигнал  $c = 1$  и оставить его постоянным, то все изменения сигнала, поданного на вход  $d$ , передаются на выход  $q$  без изменения, а на выход  $\overline{q}$  с инверсией.

Переход триггера в режим хранения произойдет по спаду импульса на входе  $c$ , в этот момент времени произойдет установка выходного сигнала  $q = d$ .

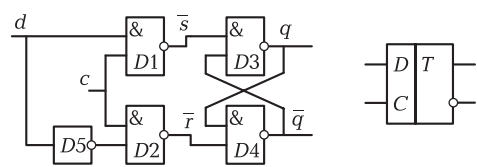


Рис. 6.7. Схема и обозначение статического  $D$ -триггера

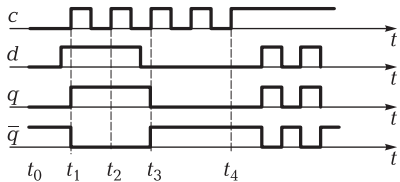


Рис. 6.8. Временные диаграммы, поясняющие работу  $D$ -триггера

Входное напряжение фиксируется только в момент изменения уровня сигнала синхронизации с высокого на низкий по спаду импульса  $s$ . Входные данные в этот момент как бы «защелкиваются», отсюда и название — *триггер-защелка*.

В название триггера входит начальная буква английского слова *delay* — задержка. Входной сигнал  $d$  передается на выход триггера  $q$  с задержкой. При этом выходной сигнал  $q$  сохраняется до прихода очередного тактового импульса.

Один информационный вход и отсутствие запрещенной комбинации являются достоинствами  $D$ -триггера, благодаря которым данный триггер находит широкое применение в цифровых устройствах.

6.5.

**D-ТРИГГЕР С ДИНАМИЧЕСКИМ УПРАВЛЕНИЕМ**

Недостатком триггеров со статическим управлением является то, что информационные сигналы на их входах не должны изменяться при постоянном сигнале  $s = 1$ . Если такое изменение произойдет, то оно передается на выход. Это называется *прозрачный режим*.

Прозрачный режим отсутствует в двухступенчатых триггерах, а также в триггерах с динамическим управлением, в которых переключение происходит в течение короткого интервала времени вблизи фронта или спада импульса синхронизации. Если триггер переключается фронтом синхроимпульса, то его вход называется прямым динамическим, а если триггер переключается спадом (задним фронтом) — инверсным динамическим.

Триггер с динамическим управлением не чувствителен к изменению информационных сигналов при постоянном значении сигнала синхронизации  $s = 1$  или  $s = 0$ . В триггерах с динамическим управлением записью информации синхроимпульс  $s$  активен лишь на коротком интервале времени в окрестности фронта или спада. Поэтому  $D$ -триггеры с динамическим управлением обладают высокой помехоустойчивостью.

Для построения триггеров с динамическим управлением используется схема трех триггеров (рис. 6.9). Здесь хранение информации осуществляет основной выходной  $\overline{R}\overline{S}$ -триггер (элементы  $D5$  и  $D6$ ) с инверсным управлением, а прием тактового  $s$  и информационного  $d$  сигналов — два входных коммутирующих триггера (элементы  $D1, D2, D3, D4$ ).

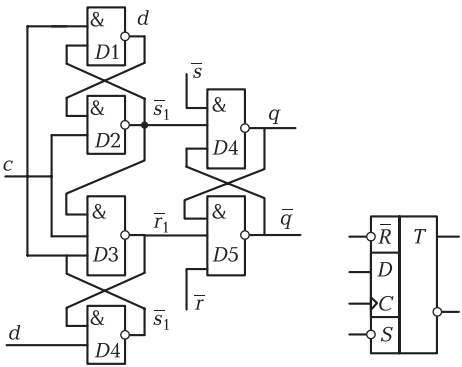


Рис. 6.9. Схема и обозначение динамического  $D$ -триггера

Режим хранения записанной информации обеспечивается при  $s = 0$ .

В этом случае в соответствии с логикой работы элементов И—НЕ  $D2, D3$  формируются сигналы  $\overline{s}1 = \overline{r}1 = 1$ . Элемент  $D4$  подает инверсное значение входного сигнала  $d$  на входы элементов  $D1$  и  $D3$ . Элемент  $D1$  повторяет значение сигнала  $d$ , как показано на схеме рис. 6.9.

Для записи 1 необходимо подать  $d = 1$ , а затем  $s = 1$ .

По фронту сигнала  $s$  на выходе элемента  $D2$  появляется сигнал  $\overline{s}1 = 0$ , который устанавливает основной триггер в состояние  $q = 1$ . Одновременно этот сигнал поступает на входы элементов  $D1$  и  $D3$ , на выходах которых будут сохраняться единичные сигналы. На изменения входного сигнала схема уже не будет реагировать.

Таблица 6.5				
$c$	$d$	$q_n$	$q_{n+1}$	Режим
$x$	$x$	0	0	Хранение
$x$	$x$	1	1	
$\uparrow$	1	0	1	Установка 1
$\uparrow$	1	1	1	
$\uparrow$	0	0	0	Установка 0
$\uparrow$	0	1	0	

Таблица 6.6	
Переход	$D$
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	0
$1 \rightarrow 1$	1

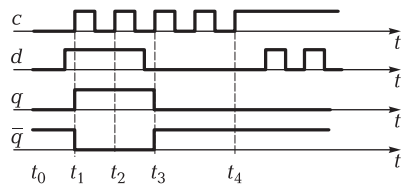


Рис. 6.10. Временные диаграммы для  $D$ -триггера с динамическим управлением

При  $d = 0$  по фронту сигнала  $c$  на выходе элемента  $D3$  формируется сигнал  $\bar{r}_1 = 0$ , который и устанавливает основной триггер в состояние  $q = 0$ .

Элементы  $D1...D4$  образуют два входных триггера, которые запоминают входной сигнал  $d$  в момент действия фронта сигнала  $c$  и сохраняют это значение, пока  $c = 1$ , что обеспечивает отсутствие прозрачного режима. Запись нового состояния возможна после перехода в режим хранения ( $c = 0$ ) и появления следующего фронта сигнала  $c$ .

Таблица истинности для данного триггера показывает, что срабатывание происходит по фронту импульса (табл. 6.5). Одно из обозначений изменения сигнала от значения 0 к значению 1, используемое в таблицах — вертикальная стрелка ( $\uparrow$ ).

Временные диаграммы, поясняющие работу данного триггера (рис. 6.10), отличаются от диаграмм для статического триггера (см. рис. 6.8) отсутствием прозрачного режима.

Для описания работы  $D$ -триггера по таблице истинности составляют таблицу переходов (табл. 6.6), которая позволяет определить значение входного сигнала  $d$ , обеспечивающее определенное изменение выходного сигнала триггера  $q$ . В соответствии с физикой работы триггера на вход  $d$  необходимо подать сигнал, значение которого равно требуемому значению выходного сигнала  $q$ .

## 6.6. JK-ТРИГГЕР

В  $JK$ -триггере имеются установочные входы:  $j$  — установка единицы,  $k$  — установка нуля. Отличие обозначений входов от принятых ранее обусловлено тем, что в данном триггере устранена запрещенная комбинация входных сигналов. В основе схемы  $JK$ -триггера используется  $RS$ -триггер с динамическим управлением (рис. 6.11).

В схему включены два двухвходовых элемента И, на входы которых подаются выходные сигналы триггера. Под действием сигналов обратной связи выполняется стробирование входных сигналов, и на установочные входы  $RS$ -триггера (входящего в состав  $JK$ -триггера) передаются только разрешенные для текущего состояния сигналы. Если триггер находится в состоянии  $q = 0$ , то разрешена передача сигнала установки 1, а если  $q = 1$ , то разрешается установка в 0. Сигналы обратной связи находятся в противофазе, поэтому один из входных элементов И будет всегда закрыт для прохождения сигналов управления.

На входы  $JK$ -триггера можно одновременно подавать единичные сигналы. Запрещенная комбинация входных сигналов в  $JK$ -триггере устранена.

Рассмотрим действие дополнительных связей, полагая  $q = 0$ .

Для данного состояния триггера разрешенным является переход в противоположное состояние  $q = 1$ . Элемент  $D1$  в этом случае выполнит формирование сигнала установки триггера в 1:  $s = j \cdot \bar{q} = 1$ .

Сигнал обратной связи поступает на элемент  $D1$  с инверсного выхода триггера.

На выходе элемента  $D2$ , на один из входов которого в этом случае подается сигнал  $q = 0$ , появится сигнал  $r = 0$ .

Аналогично можно показать, что при состоянии триггера  $q = 1$  элемент  $D2$  сформирует сигнал установки 0:  $r = k \cdot d = 1$ . Если триггер находится в состоянии 1, то разрешен сигнал установи 0.

При воздействии на входы  $c = j = k = 1$  появится сигнал 1 только на выходе того элемента, который устанавливает противоположное состояние триггера. Такая комбинация сигналов для  $JK$ -триггера не является запрещенной.

Работу триггера описывает таблица истинности (табл. 6.7), которая показывает, что при любых значениях  $c = 0$  или  $c = 1$  триггер находится в режиме хранения данных. Запись 0 или 1 происходит

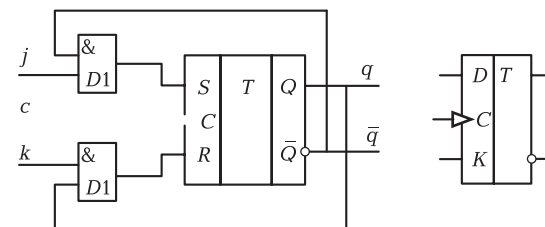


Рис. 6.11. Структура и обозначение  $JK$ -триггера

Таблица 6.7					
$c$	$j$	$k$	$q_n$	$q_{n+1}$	Режим
$x$	$x$	$x$	0	0	Хранение
$x$	$x$	$x$	1	1	
$\uparrow$	0	0	0	0	
$\uparrow$	0	0	1	1	
$\uparrow$	0	1	0	0	Установка 0
$\uparrow$	0	1	1	0	
$\uparrow$	1	0	0	1	Установка 1
$\uparrow$	1	0	1	1	
$\uparrow$	1	1	0	1	Инверсия состояния
$\uparrow$	1	1	1	0	

по спаду синхросигнала, что показано стрелкой в таблице, которая означает переход от высокого уровня к низкому.

По таблице истинности строится таблица переходов (табл. 6.8) в которой отражены все возможные комбинации управляющих сигналов  $j$  и  $k$  для заданных переходов триггера при активном изменении синхросигнала.

Работу триггера поясняют временные диаграммы (рис. 6.12), показывающие, что срабатывание триггера происходит по фронту синхроимпульса. На диаграммах показаны режимы: установка 1 (момент времени  $t_1$ ), установка 0 (момент времени  $t_2$ ), а также инверсия состояния (момент времени  $t_3$ ) и режим хранения (момент времени  $t_4$ ).

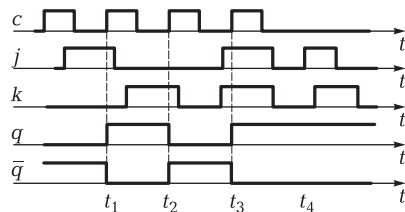


Рис. 6.12. Временные диаграммы, поясняющие работу JK-триггера

Таблица 6.8	
Переход	$j \ k$
$0 \rightarrow 0$	0 X
$0 \rightarrow 1$	1 X
$1 \rightarrow 0$	X 1
$1 \rightarrow 1$	X 0

Данный триггер является наиболее универсальным. При соответствующем подключении входной логики JK-триггер может выполнять функции триггера любого другого типа.

## 6.7. СЧЕТНЫЙ ТРИГГЕР

**Счетный триггер** (английское название — *toggle-flip-flop*) называют также T-триггером. Он изменяет свое логическое состояние на противоположное по каждому активному сигналу на входе  $t$ . В сериях выпускаемых микросхем T-триггеров, как правило, нет, но T-триггер может быть построен на основе триггеров типов D или JK с динамическим управлением.

В зависимости от логики построения цепей управления триггера различают асинхронные и синхронные T-триггеры. Асинхронные T-триггеры имеют один управляющий вход  $t$ , который является динамическим, следовательно, срабатывание триггера будет происходить по фронту (или по спаду) входных импульсов (рис. 6.13).

Для построения T-триггера на основе D-триггера необходимо инверсный выход  $\bar{q}$  соединить с информационным входом D (рис. 6.13, а). В результате каждый синхроимпульс будет изменять состояние триггера на противоположное.

Асинхронный счетный триггер можно построить также на базе JK-триггера. Согласно таблице истинности JK-триггера (см. табл. 6.7) данный триггер переходит в противоположное состояние каждый раз при одновременной подаче на входы J и K сигнала 1. Для построения T-триггера необходимо на входы J и K подать сигнал 1, как показано на рис. 6.13, б. Обозначение асинхронного T-триггера с прямым динамическим входом приведено на рис. 6.13, в.

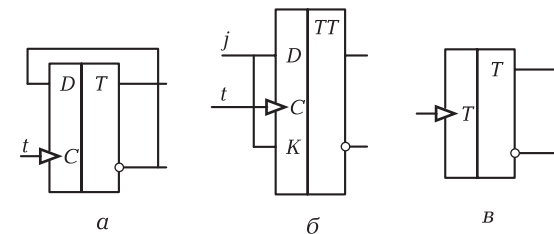


Рис. 6.13. Асинхронные счетные триггеры (а—в)



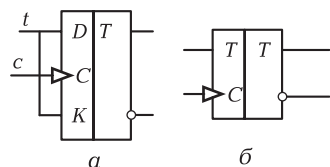


Рис. 6.14. Синхронные счетные T-триггеры (а, б)

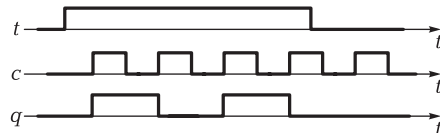


Рис. 6.15. Временные диаграммы, поясняющие работу T-триггера

Таблица 6.9

Переход	$t$
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	1
$1 \rightarrow 1$	0

Синхронный счетный триггер имеет два входа: вход синхронизации  $C$  и вход разрешения счета, который обозначают  $T$ . Этот триггер можно построить только на JK-триггере (рис. 6.14, а). На рис. 6.14, б приведено обозначение синхронного счетного триггера с прямым динамическим входом.

Работу синхронного счетного триггера поясняют временные диаграммы (рис. 6.15).

Сигнал  $t$  является управляющим. Процесс счета разрешается при  $t = 1$ . Каждый импульс входной частоты, поданный на вход  $C$ , вызывает изменение состояния триггера на противоположное по фронту импульса. Диаграммы показывают, что период повторения выходных импульсов счетного триггера вдвое больше периода входных импульсов. Триггер делит частоту входных импульсов на 2.

В таблице переходов для счетного триггера (табл. 6.9) содержатся значения входных сигналов  $t$ , обеспечивающие определенные переходы триггера. Для сохранения состояния триггера неизменным необходимо подать сигнал  $t = 0$ , а сигнал  $t = 1$  обеспечит изменение состояния на противоположное.

## 6.8. КОМБИНИРОВАННЫЕ ТРИГГЕРЫ В ПЛИС ALTERA

Библиотека `\prim` содержит комбинированные триггеры различных типов (рис. 6.16). В обозначениях триггера вначале указан его тип (JK, D, T), затем записаны буквы FF (flip-flop — триггер), затем буквы, указывающие на наличие дополнительных управляющих входов (ENA — Enable — разрешение работы). Символ треуголь-

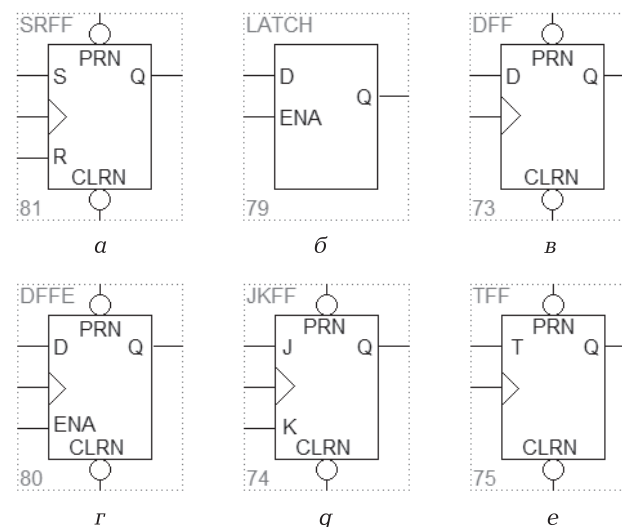


Рис. 6.16. Библиотечные триггеры (а—е)

гольника у входа  $C$  означает динамическое управление и срабатывание по переднему фронту импульса. Выводы, расположенные на обозначении слева ( $J, K, D, T, ENA$ ), являются управляющими синхронными входами, они определяют условия срабатывания по фронту синхроимпульса.

Выводы, расположенные на верхней и нижней сторонах обозначения, — это асинхронные установочные входы:  $PRN$  (Preset Negative) — инверсный вход установки единицы, а  $CLRN$  (Clear Negative) — инверсный вход очистки (установки нуля). Тип триггера указан в верхней части обозначения. На рисунке показаны: SRFF — RS-триггер (рис. 6.16, а), LATCH — D-триггер-защелка (рис. 6.16, б), DFF — D-триггер с динамическим входом (рис. 6.16, в), DFFE — D-триггер с динамическим входом и с дополнительным входом разрешения срабатывания (рис. 6.16, г), JKFF — JK-триггер (рис. 6.16, д), TFF — T-триггер (рис. 6.16, е).

**Внимание!** Триггеры, реализованные в ПЛИС, срабатывают, как правило, по фронту синхроимпульса (имеют прямое динамическое управление), отображаемое в обозначениях триггеров треугольником, направленным внутрь элемента. Для всех рассмотренных типов триггеров применительно к ПЛИС рассмотрено срабатывание по фронту синхроимпульса.

Существуют триггеры, выполненные в виде микросхем, срабатывающие по спаду синхроимпульса.

## 6.9. ОПИСАНИЕ ТРИГГЕРОВ НА ЯЗЫКЕ VERILOG

Для описания схем с элементами памяти используются последовательные операторы, которые содержат следующие элементы.

1. Ключевое слово *always* (всегда) — обозначает повторение присваивания при возникновении определенного события.
2. Символ *@* — соответствует понятию «событие».
3. Список чувствительности в круглых скобках — это перечисление через запятую или через знак дизъюнкции всех сигналов, изменение которых должно вызывать повторение процесса присваивания. В списке чувствительности можно указать срабатывание по фронту или спаду сигнала. Фронт сигнала обозначается *posedge*, спад — *negedge*.
4. Последовательные операторы, вычисляющие значения сигнала, — *if*, *case*, блоки *begin* — *end*, в которых может содержаться несколько последовательных операторов.

Выходные сигналы должны иметь тип *reg*, что обязательно нужно указывать.

Если оператор содержит только слово *always* и список чувствительности отсутствует, то срабатывание оператора происходит при каждом изменении любого из входных сигналов.

Формат строки описания с оператором *if* имеет следующий вид:

*always if* (Условие) Ветвь ДА ; *else* Ветвь НЕТ ;

Пример 6.1 содержит описания статического *D*-триггера — защелки, схема и обозначение которого приведены на [рис. 6.7](#).

```
//Пример 6.1. Статический
// D-триггер-защелка
module v61_latch (c, d, q);//3
input c, d;                //4
output q;                  //5
reg q;                     //6
always if (c) q = d;        //7
endmodule                  //8
```

Статический *D*-триггер, называемый защелка *latch*, в простейшем случае имеет входы *c*, *d*, выход *q* и передает на выход сигнал при наличии сигнала *c* = 1. По входу *c* управление осуществляется

потенциалом. Триггер имеет прозрачный режим. В американском обозначении вход *c* называется *latch*.

При описании цифровых схем с памятью обычно используют последовательные операторы, которые формируют сигнал типа *reg* — регистр.

Строки 1, 2 описания — комментарий, строка 3 — заголовок, содержащий имя модуля и перечисление всех сигналов.

Строки 4, 5 содержат описания входных и выходного сигналов.

Обратим внимание на строку 6. При использовании последовательных операторов для выходного сигнала должен быть указан тип «регистр».

Строка 7 — поведенческое описание триггера-защелки. Смысл этой строки в том, что изменение выходного сигнала триггера или, как говорят, срабатывание должно происходить, если *c* = 1, а также при изменениях сигнала *d*. В описании используется ключевое слово *always* без списка чувствительности. Такая запись обеспечит срабатывание по изменению любого сигнала.

Установку выходного состояния триггера описывает такой алгоритм.

Если *c* = 1, то входные данные «защелкиваются» и передаются на выход, в результате появляется сигнал *q* = *d*.

Выясним, какое действие должно выполняться, если *c* = 0. В соответствии с таблицей истинности (см. табл. 6.1) при *c* = 0 данный триггер должен находиться в режиме хранения, и его входной сигнал изменяться не должен.

Если *c* = 0, то никакие действия не требуется выполнять. Для описания этого случая ветвь *else* в операторе *if* отсутствует.

Рассмотрим описание *D*-триггера с динамическим управлением (см. [рис. 6.9](#)).

В описании поведения последовательностных схем с динамическим управлением необходимо учитывать, что эти схемы изменяют свое состояние по определенному фронту сигнала синхронизации. Триггеры, используемые в ПЛИС, обычно срабатывают по положительному фронту импульса. Срабатывание по положительному фронту сигнала *c* описывает конструкция языка *always @ (posedge c)* (пример 6.2).

```
// Пример 6.2
// Динамический D-триггер, 2
//вариант 1
module v62__d__din (c, d, q);
input c, d;
output q;
```

```

reg q; //7
always @ (posedge c) q = d; //8
endmodule //9

```

В данном триггере прозрачного режима нет. Запись сигнала  $d$  в триггер и его появление на выходе  $q$  должно происходить по фронту синхросигнала  $c$ . Смысл последовательного оператора, описывающего работу триггера (строка 8), состоит в следующем: «всегда по событию — положительный фронт сигнала  $c$  присваивать сигналу  $q$  значение сигнала  $d$ ».

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятию «триггер». Перечислите типы триггеров.
2. Укажите различия синхронных и асинхронных триггеров. В чем заключаются преимущества синхронных триггеров в сравнении с асинхронными?
3. Укажите различия статических и динамических триггеров.
4. Укажите на отличия между триггерами типов  $RS$ ,  $JK$ ,  $D$  и  $T$ .
5. Изобразите временные диаграммы, поясняющие работу статического и динамического  $D$ -триггеров.
6. Укажите функциональные особенности триггеров типа  $RS$ ,  $D$ ,  $JK$ ,  $T$ .
7. Опишите входные и выходные сигналы триггеров различных типов.
8. Как составить тестовые временные диаграммы для триггера  $D$ ,  $JK$ ,  $T$ ?
9. Укажите особенности описания схем с элементами памяти на языке *Verilog*.
10. Изобразите схемы счетных триггеров, реализованных на базе триггеров типа  $JK$  и  $D$ .

## Глава 7

### РЕГИСТРЫ

**Элементарной ячейкой** электронной памяти является триггер, способный сохранять 1 бит записанной в нем информации. Однако зачастую требуется запоминать двоичные слова большей разрядности, для этого используются регистры.

**Регистром** называется устройство из нескольких триггеров, предназначенное для записи, хранения и выдачи информации. Каждый разряд двоичного числа записывается в отдельном триггере, поэтому количество триггеров в регистре определяет разрядность записываемого числа.

Ввод и вывод информации могут выполняться параллельным или последовательным кодом, поэтому существуют различные типы регистров: параллельные, параллельно-последовательные, регистры сдвига.

Регистры обычно строятся на основе  $D$ -триггеров, которые могут быть с динамическим управлением — по фронту, или со статическим управлением — по уровню сигнала.

#### 7.1. ПАРАЛЛЕЛЬНЫЙ РЕГИСТР

В параллельных регистрах ввод и вывод двоичных слов производится одновременно для всех разрядов. Параллельный регистр строится на  $D$ -триггерах. К входам триггеров подключается шина входных данных, а к выходам триггеров — шина выходных данных. При записи информации в параллельный регистр все двоичные разряды записываются одновременно, поэтому входы синхронизации всех триггеров соединяют параллельно. Запись данных в регистр происходит при поступлении импульса синхронизации  $c$ .

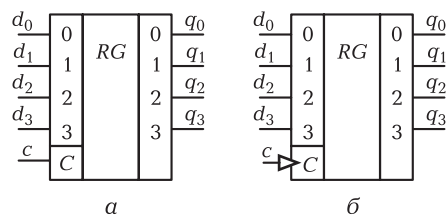


Рис. 7.1. Обозначения статического (а) и динамического (б) параллельных регистров с динамическим управлением

Параллельный регистр может быть построен на базе статических *D*-триггеров (в обозначении входов *C* будут отсутствовать треугольники), его называют регистр-защелка (рис. 7.1, а). Этот регистр имеет прозрачный режим, что используется в некоторых схемах.

Параллельный регистр с динамическим управлением прозрачного режима не имеет. Изображение треугольника в обозначении входа *C*, приведенное на рис. 7.1, б, соответствует регистру с прямым динамическим входом, в котором запись данных происходит по фронту импульса синхронизации *c*. Схема четырехразрядного параллельного регистра, построенного на *D*-триггерах с динамическим управлением, приведена на рис. 7.2.

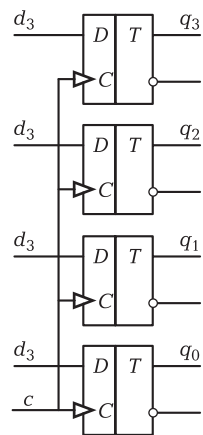


Рис. 7.2. Схема параллельного регистра

Существуют другие варианты схем параллельных регистров. В интегральном исполнении выпускаются буферные регистры, имеющие мощные формирователи выходных сигналов с тремя состояниями выхода или типа открытый коллектор. Они предназначены для подключения устройств к общим шинам, а также для работы на внешнюю низкочастотную емкостную нагрузку.

При проектировании схем, содержащих элементы памяти, целесообразно использовать описания регистров на языке *Verilog*.

В описаниях параллельных регистров, как и в описаниях триггеров (см. примеры 6.1 и 6.2), используются последовательные операторы с ключевым словом *always* (примеры 7.1 и 7.2). В этих примерах входные и выходные сигналы описаны как векторы опреде-

ленной разрядности (строки 4...6), кроме того, выходные сигналы описаны как регистр (строки 7).

```
//Пример 7.1. Статический
// параллельный регистр
module v71_stat_reg (c, d, q);//3
input c;
input [3:0] d;
output [3:0] q;
reg [3:0] q;
always if (c) q = d;
endmodule
```

```
Пример 7.2. Динамический
// параллельный регистр
module v72_din_reg (c, d, q);//3
input c;
input [3:0] d;
output [3:0] q;
reg [3:0] q;
always @ (posedge c) q = d;
endmodule
```

Поведение регистра-защелки описывает строка 8 в примере 7.1, которая имеет смысл: *Всегда, если c = 1, выполнять присваивание q = d.*

В операторе *always* символ @ и список чувствительности отсутствуют, поэтому срабатывание оператора будет происходить при любых изменениях сигнала *d* при условии *c = 1*. Данное описание соответствует регистру, который имеет прозрачный режим.

В примере 7.2 строка 8 описывает срабатывание динамического регистра по фронту и имеет смысл: *Всегда по событию — положительный фронт сигнала c — выполнять присваивание q = d.*

## 7.2. СДВИГАЮЩИЕ РЕГИСТРЫ

**Сдвигающий регистр** — цифровая схема, состоящая из последовательно включенных триггеров, содержимое которых можно сдвигать на один разряд влево или вправо подачей тактовых импульсов. Сдвигающие регистры служат для преобразования последовательного кода в параллельный код и наоборот. Также они позволяют умножать или делить коды чисел на 2. Число, код которого записан в сдвигающий регистр, при сдвиге влево удваивается, а при сдвиге вправо уменьшается вдвое.

Сдвигающий регистр, схема которого приведена на **рис. 7.3**, имеет последовательный вход *si* (*Serial Input*) и параллельный выход  $q_3...q_0$ . Кроме того, в данной схеме сигнал  $q_0$  можно использовать как последовательный выход. Соединение триггеров между собой, при котором сигнал передается с выхода старшего разряда на вход младшего разряда, позволяет получить сдвиг вправо.

Работу схемы поясняют временные диаграммы (**рис. 7.4**).

На вход *si* поступает последовательный код, значения которого будут записываться в триггер старшего разряда в моменты времени, соответствующие фронтам синхроимпульсов на входе *C*. По фронту первого синхроимпульса (момент  $t_1$ ) произойдет запись входного сигнала  $si = 1$  в триггер старшего разряда, в результате будет получен сигнал  $q_3 = 1$ . Изменение сигнала  $q_3$  произойдет с задержкой во времени относительно фронта сигнала *c*, а в момент фронта данного синхроимпульса сигнал  $q_3$  остается равным нулю. Сигнал *c* подается параллельно на все триггеры. На входах всех триггеров, кроме первого, к моменту поступления фронта первого импульса с присутствует сигнал, равный нулю, поэтому в момент  $t_1$  устанавливается сигнал  $q_3 = 1$ , а выходные сигналы остальных триггеров сохраняются нулевыми:  $q_2 = q_1 = q_0 = 0$ . После первого синхроимпульса в регистре с задержкой установится код 1000.

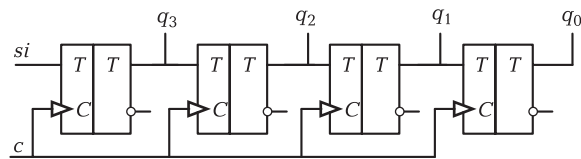


Рис. 7.3 Сдвигающий регистр

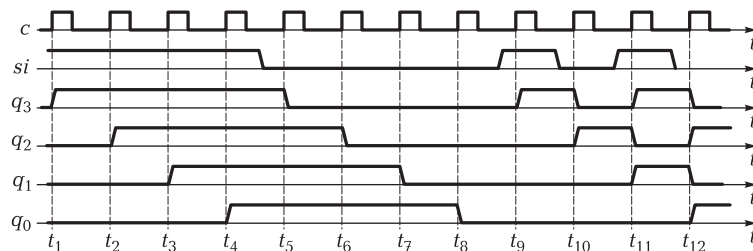


Рис. 7.4. Временные (а–в) диаграммы, поясняющие работу сдвигающего регистра

К моменту появления фронта второго импульса (момент  $t_2$ ) входные сигналы первого и второго триггеров равны единице. Первый триггер останется в состоянии 1 ( $q_3 = 1$ ), следующий триггер установится в 1 ( $q_2 = 1$ ). Остальные триггеры остаются в нулевом состоянии ( $q_1 = q_0 = 0$ ). После второго синхроимпульса в регистре будет установлен код 1100.

Каждый синхроимпульс записывает четыре разряда кодов: входной сигнал *si* и коды, содержащиеся в разрядах  $q_3, q_2, q_1$ , со сдвигом в разряды  $q_3, q_2, q_1, q_0$  соответственно.

После действия синхроимпульса в момент времени  $t_4$  код регистра принимает значение 1111, соответствующее зафиксированным на входе *si* четырем единичным значениям. В моменты времени  $t_5, t_6, t_7, t_8$  подаются четыре нулевых значения сигнала *si*, поэтому после действия синхроимпульса в момент времени  $t_8$  в регистре будет установлен код 0000. На интервале  $t_9...t_{12}$  сигнал *si* принимает последовательно значения 1010. Указанный последовательный код записывается в сдвигающий регистр младшими разрядами вперед.

Рассмотренный сдвигающий регистр позволяет осуществить прием информации в виде последовательного кода, а выдачу в виде параллельного или в виде последовательного кода. При параллельной выдаче информация снимается одновременно с выходов всех триггеров. Последовательная выдача осуществляется с выхода  $q_0$  в моменты действия тактовых импульсов.

В описании сдвигающего регистра на языке *Verilog* (пример 7.3) особый интерес представляет строка 7, описывающая работу регистра. Новое значение кода *q*, содержащееся в регистре, должно быть получено из предыдущего значения, как объединение сигналов *si, q3, q2, q1*. Оператор объединения векторов записывается как перечисление исходных векторов в фигурных скобках через запятую. Предыдущее значение  $q_0$  в данном случае теряется. Выполнение сдвига поясняет **рис. 7.5**.

```
//Пример 7.3. Сдвигающий
// регистр
module v73_shift_reg (c, si, q);           //3
input c, si;                             //4
output [3:0] q;                           //5
```

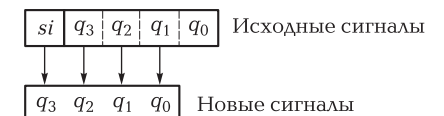


Рис. 7.5. Пояснение к примеру 7.3



```

reg [3:0] q; //6
always @ (posedge c) //7
q = {si,q[3:1]}; //8
endmodule //9

```

### 7.3. УНИВЕРСАЛЬНЫЙ СДВИГАЮЩИЙ РЕГИСТР

Регистры сдвига изготавливаются обычно как универсальные, имеющие параллельные и последовательные входы и выходы, что позволяет преобразовывать не только последовательный код в параллельный, но и выполнять обратное преобразование параллельного кода в последовательный. Последовательно-параллельный регистр используется, например, при реализации последовательного порта в микропроцессорной системе.

Универсальный сдвигающий 4-разрядный регистр (рис. 7.6) имеет отдельные выходы для ввода данных в виде параллельного и последовательного кода — это шина параллельного кода  $d_3...d_0$  и вход последовательного кода  $SI$  (Serial Input). Выбор входных сигналов определяет сигнал разрешения параллельной загрузки  $l$  (Load — загрузка). Выходные сигналы регистра формируются на шине  $q_3...q_0$ .

Схема универсального регистра (рис. 7.7) содержит четыре  $D$ -триггера, синхронизация которых осуществляется общим сигналом  $c$ . Входной сигнал каждого триггера формируется мультиплексором «2 в 1». При разрешении параллельной загрузки ( $l = 1$ ) входы триггеров подключаются к проводникам входной шины данных. В этом режиме по положительному фронту синхроимпульса с выполняется запись входного кода в регистр.

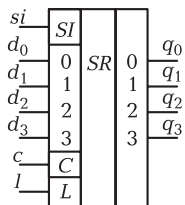


Рис. 7.6. Обозначение универсального сдвигающего регистра

При запрещении параллельной загрузки ( $l = 0$ ) выполняется последовательная загрузка. По положительному фронту синхроимпульса  $c$  произойдет сдвиг вправо. Код, хранящийся в регистре, переместится на один разряд в сторону младших разрядов.

Универсальный сдвигающий регистр используется для преобразования параллельного кода в последовательный и наоборот.

Описание универсального сдвигающего регистра (пример 7.4) выполнено по аналогии с предыдущими примерами. В строке 3 пере-

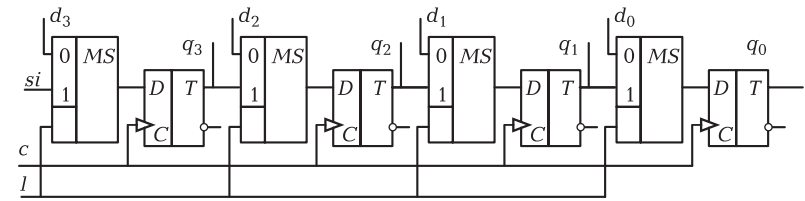


Рис. 7.7. Схема универсального сдвигающего регистра

числены все сигналы в соответствии с обозначением регистра (см. рис. 7.6), в строке 4 описаны одноразрядные входные сигналы, а в строке 5 — вектор входных данных. В строках 6, 7 описаны выходные сигналы, для которых указан тип регистра, что необходимо при формировании этих сигналов последовательным оператором.

```

//Пример 7.4. Универсальный //1
// сдвигающий регистр //2
module v74_univ_reg (c, si, d, l, q); //3
input c, si, l; //4
input [3:0] d; //5
output [3:0] q; //6
reg [3:0] q; //7
always @ (posedge c) //8
if (l) {q} = {d}; //9
else {q} = {si,q[3:1]}; //10
endmodule //11

```

Строка 8 — начало последовательного оператора — описывает его срабатывание по фронту сигнала  $c$ . Строка 9 содержит оператор *if*, в котором в качестве условия записан сигнал  $l$ . Если  $l = 1$ , то выполняется параллельная загрузка, описанная присваиванием  $\{q\} = \{d\}$ . Если  $l = 0$ , то выполняется сдвиг.

### 7.4. РЕГИСТРОВАЯ ПАМЯТЬ В ПРОЦЕССОРАХ

Регистры используют в процессорах для построения блоков памяти сравнительно небольшой емкости и предельно высокого быстродействия, которые выполняют функции сверхбыстродействующей памяти. Высокое быстродействие достигается благодаря использованию блоков памяти небольшой емкости, которые имеют достаточно простые дешифраторы адреса, имеющие высокое быстродействие. Кроме того, расположение элементов памяти в не-

посредственной близости от устройств обработки данных, обеспечивает быстрый физический доступ к информации, хранящейся в этих элементах.

В современных процессорах используются блоки регистров, предназначенных для хранения информации различного типа (данных, адресов, управляющих сигналов), необходимой при выполнении текущих команд. Быстродействие указанных регистров существенно влияет на быстродействие процессора. Из отдельных регистров составляются блоки регистровой памяти, называемые также регистровыми файлами.

Принцип построения блока регистров поясняет рис. 7.8. Данный блок содержит четыре регистра, каждый из которых предназначен для хранения параллельного кода определенной разрядности. В блоке использованы параллельные регистры с динамическим управлением, которые можно построить по схеме, показанной на рис. 7.1, б, или по описанию на языке *Verilog* из примера 7.2.

Процесс записи в цифровых устройствах, как правило, синхронизируют. В данном блоке регистров предусмотрена синхронная запись данных. Данные записи подаются на входную шину  $D$ , которая подключена параллельно к входам данных всех регистров. Однако запись данных произойдет только в один из регистров, адрес которого будет подаваться на входы дешифратора  $ax_1$ ,  $ax_0$ . Дешифратор работает в режиме демультиплексора, он передаст синхросигнал  $c$  на вход только одного выбранного регистра. Произойдет синхронная запись данных, поданных на шину  $D$  по фронту синхросигнала  $c$ .

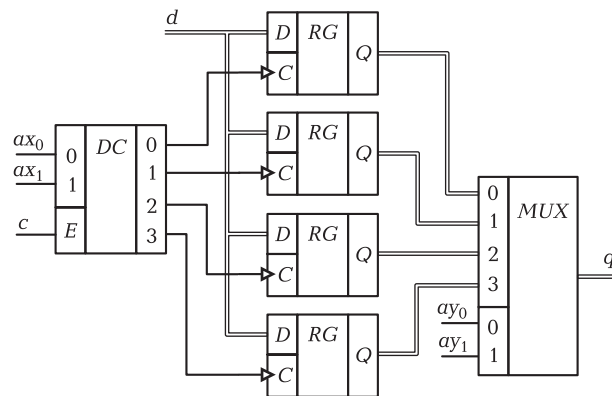


Рис. 7.8. Схема блока регистров

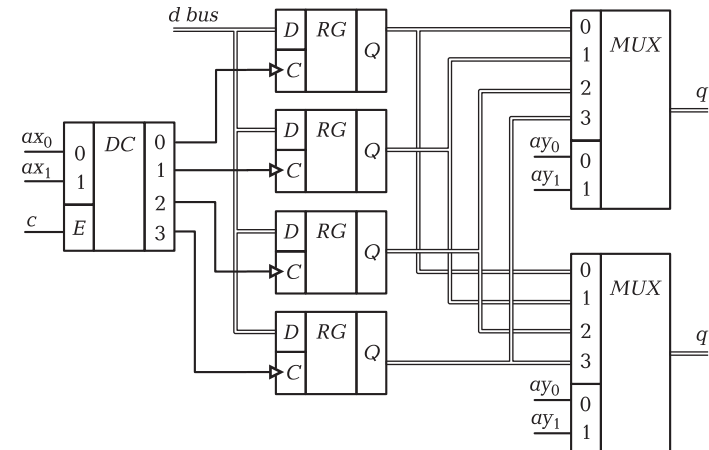


Рис. 7.9. Схема блока РОН

Чтение — асинхронное, оно выполняется по адресу  $ay$  посредством мультиплексора.

Существенное расширение функциональных возможностей процессоров достигается при использовании блоков регистров общего назначения (РОН), построенных по принципу двухадресной (двухортовой) памяти, в которых обеспечивается одновременное асинхронное чтение по двум адресам  $ax$  и  $ay$ . Для чтения данных к выходам регистров подключено два мультиплексора, которые коммутируют шины (рис. 7.9). Данные на выходах мультиплексоров  $qx$  и  $qy$  появляются непосредственно после подачи адресов  $ax$  и  $ay$ .

В блоке РОН используется синхронная запись данных, поданных на 8-разрядную шину  $d\_bus$ , в один из регистров в соответствии с адресом  $ax$ . Данные, предназначенные для записи, подаются по шине  $d\_bus$  параллельно на входы данных всех регистров. Запись данных произойдет только в один из регистров, на который в соответствии с адресом  $ax$  поступит синхросигнал с дешифратора-демультиплексора.

Важным моментом является получение на выходах блока РОН двух операндов одновременно, что позволяет сразу же выполнить двухадресную команду. Блок РОН, позволяющий выполнить чтение по двум адресам одновременно, является составной частью современных процессоров и микроконтроллеров.

1. Что такое регистры? Каковы области их применения, классификация?
2. Изобразите временную диаграмму записи в трехразрядный последовательный регистр двоичного кода, равного 101.
3. Поясните принцип функционирования параллельных регистров.
4. Определите построение сдвигающего регистра, поясните принцип работы.
5. Опишите работу универсального регистра в режиме параллельной загрузки.
6. Опишите работу универсального регистра в режиме сдвига.
7. Поясните способы описания регистров на языке *Verilog*.
8. Где находит применение сдвигающий регистр?

### СЧЕТЧИКИ

**Счетчик** — устройство, содержащее несколько триггеров, состояние которых определяется числом поступивших на вход устройства импульсов. Основное назначение счетчиков — подсчет импульсов и деление частот.

Счетчик — это цепочка последовательно включенных счетных триггеров, каждый из которых делит частоту повторения входных импульсов на два. Каждый входной импульс изменяет состояние счетчика, которое сохраняется до поступления следующего импульса. Различают суммирующий, вычитающий и реверсивный счетчики. Счетчики обозначают *СТ* (от англ. *counter*).

Каждый триггер счетчика хранит один разряд двоичного числа, которое однозначно определяет количество импульсов, поступивших на вход. Количество триггеров определяет модуль счетчика. Модуль счетчика  $M$  — максимальное количество его состояний. Для  $n$ -разрядного счетчика максимальный модуль  $M = 2^n$ , а период повторения импульсов переполнения содержит  $2^n$  периодов входной частоты. Это означает, что код счетчика может принимать  $2^n$  значений от 0 до  $2^n - 1$ .

Существуют счетчики с произвольным модулем, который меньше максимального. Для кодирования состояний счетчика используются различные коды. Наибольшее распространение получил двоичный код.

Различают асинхронные и синхронные счетчики.

Анализ натурального ряда чисел, записанного в двоичной системе счисления, позволяет построить теоретические временные диаграммы (рис. 8.1) и определить требования к схеме счетчика.

Для построения суммирующего счетчика необходимы триггеры, изменяющие свое состояние на противоположное по каждому входному импульсу. В суммирующем счетчике двоичный код на его

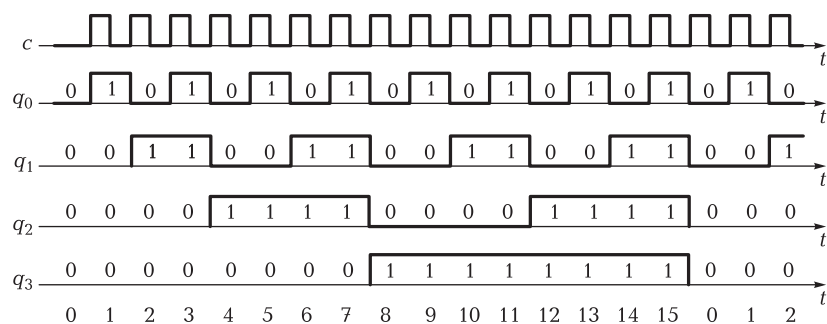


Рис. 8.1. Временные диаграммы работы суммирующего счетчика

выходах увеличивается на единицу с каждым входным импульсом. Если в исходном состоянии на всех триггерах счетчика установлены логические нули, то при поступлении импульсов на его вход должна формироваться последовательность кодов 0000, 0001, 0010, 0011, 0100, ..., 1111, 0000 и т.д.

В суммирующем счетчике каждый триггер изменяет свое состояние на противоположное, когда на него действует спад импульса от предыдущего триггера, если предыдущий триггер переходит из 1 в 0.

Частота на выходе каждого последующего триггера в 2 раза меньше, чем предыдущего, т. е. каждый триггер делит частоту входного сигнала на два. Это свойство используется в делителях частоты. Частота сигнала на выходе последнего триггера при  $n = 4$  меньше частоты входных импульсов в  $M = 2^n = 16$  раз. Максимальный код счетчика равен 15, с приходом 16-го импульса происходит переполнение счетчика, все разряды обнуляются и далее процесс повторяется.

Подобным образом можно описать процесс вычитания двоичных чисел. В вычитающем счетчике каждый импульс на входе уменьшает на единицу двоичный код на выходах. Процессу вычитания соответствует последовательность двоичных кодов: 1111, 1110, 1101, 1100, 1011, и т. д. Подобно **рис. 8.1** можно составить теоретические временные диаграммы и сделать вывод, что в вычитающем счетчике каждый триггер изменяет состояние на противоположное по фронту импульса от предыдущего триггера (если предыдущий триггер переходит из 0 в 1).

## 8.1. АСИНХРОННЫЕ СЧЕТЧИКИ

**Асинхронный счетчик** — цепочка счетных триггеров, вход каждого из которых подключен к выходу предыдущего триггера. В асинхронных счетчиках под воздействием входного импульса происходит переключение соответствующих разрядов последовательно от разряда к разряду. Для построения асинхронного суммирующего счетчика необходимо включить последовательно требуемое количество асинхронных триггеров с динамическим управлением (рис. 8.2). При соединении триггеров между собой необходимо учитывать вид сигнала, вызывающего изменение состояния триггера. Для того чтобы триггеры, имеющие прямой динамический вход, изменяли свое состояние по спаду входного сигнала предыдущего триггера, входы триггеров должны быть подключены к инверсным выходам предыдущих триггеров.

При поступлении тактовых импульсов на вход  $c$ , который называют *счетным входом*, работа счетчика будет происходить в соответствии с временными диаграммами (см. [рис. 8.1](#)), из которых следует, что до прихода первого импульса все триггеры находились в нулевом состоянии. Каждый импульс, поданный на вход  $c$ , изменяет выходной сигнал  $q_0$  на противоположный. Каждый спад сигнала  $q_0$  переключает следующий триггер в противоположное состояние. После прихода 16 импульсов все четыре триггера находятся снова в нулевом состоянии. Двоичный код, снимаемый с выходов триггеров  $q_3, q_2, q_1, q_0$  показывает, сколько импульсов поступило на счетчик.

Описанный счетчик называется *асинхронным*, или *последовательным*. В нем каждый последующий каскад изменяет состояние после предыдущего. Счетчики с последовательным переносом имеют низкое быстродействие. Максимальное время установления возникает при переключениях всех разрядов от кода 11...1 к коду 00...0. Сигнал на вход последнего триггера приходит лишь тогда, когда все предыдущие триггеры переключились. Это время пропорционально разрядности счетчика и времени переключения триггеров.

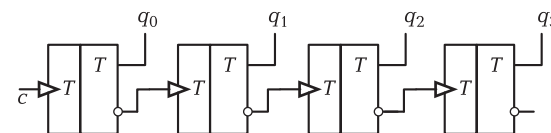


Рис. 8.2. Асинхронный суммирующий счетчик

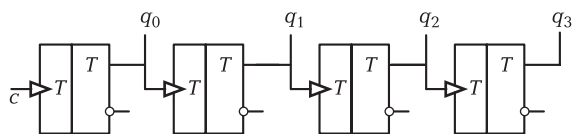


Рис. 8.3. Асинхронный вычитающий счетчик

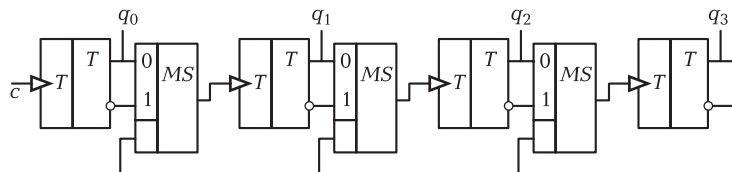


Рис. 8.4. Реверсивный счетчик

Асинхронный вычитающий счетчик (рис. 8.3) отличается от суммирующего тем, что прямой выход каждого триггера соединен с входом последующего.

Реверсивные счетчики могут работать как в режиме сложения, так и в режиме вычитания. Для изменения направления счета необходимо подключать либо прямой, либо инверсный выход предыдущего триггера, входящего в счетчик, к входу последующего (рис. 8.4). Реверсивные счетчики изменяют направление счета под воздействием управляющего сигнала  $s$ , который поступает на мультиплексоры  $MS$ , осуществляющие переключение межразрядных связей.

Достоинством асинхронных счетчиков является простота схемы: увеличение разрядности производится подключением необходимого числа триггеров. К недостаткам асинхронных счетчиков относятся сравнительно низкое быстродействие и его зависимость от числа разрядов, а также появление промежуточных значений выходных двоичных кодов при последовательном переключении триггеров в новое состояние.

## 8.2. СИНХРОННЫЕ СЧЕТЧИКИ

В синхронных счетчиках срабатывание всех триггеров происходит по фронту (или спаду) тактовых импульсов, которые одновременно подаются на входы с всех разрядов, а наличие или от-

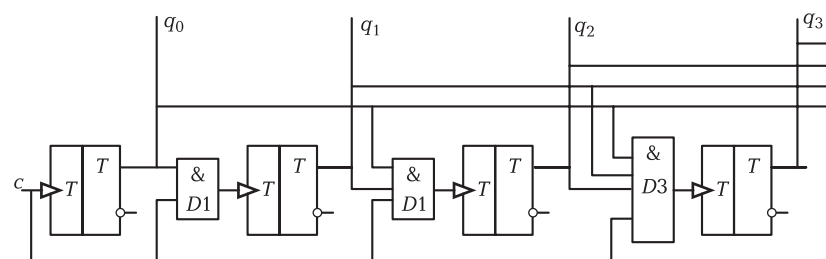


Рис. 8.5. Синхронный счетчик с параллельным переносом

сутствие переключения определяют управляющие входы:  $J$ ,  $K$  или  $T$ . Для повышения быстродействия в схемах синхронных счетчиков используют цепи ускоренного переноса.

Быстродействие увеличивается в счетчиках со сквозным переносом (рис. 8.5). Анализ временных диаграмм, поясняющих процесс счета (см. рис. 8.1), позволяет определить условия переключений триггеров суммирующего счетчика. Триггер младшего разряда переключается каждым импульсом синхронизации. Все последующие триггеры должны изменять состояние 0 на 1, если все предыдущие триггеры находятся в состояниях 1. Это требование осуществляется с помощью дополнительных логических элементов  $D_1 \dots D_3$ .

## 8.3. МНОГОФУНКЦИОНАЛЬНЫЕ СЧЕТЧИКИ В ПЛИС

Обычно счетчики имеют несколько входов управления для расширения функциональных возможностей. Многофункциональные счетчики имеют, как правило, режим параллельной загрузки, позволяющий загрузить параллельный код, поданный на дополнительные входы данных, устанавливающий начальное состояние всех триггеров. Для включения режима параллельной загрузки подобные счетчики имеют управляющий вход  $l$  или  $ldn$  (от англ. *load* — загрузка).

Счетчики, разработанные как библиотечные элементы ПЛИС, являются многофункциональными. В качестве примера на рис. 8.6 показан библиотечный элемент *4count* из библиотеки *mf* — 4-разрядный реверсивный счетчик с параллельной загрузкой.

Назначение выводов:  $A$ ,  $B$ ,  $C$ ,  $D$  — входные данные;  $QA$ ,  $QB$ ,  $QC$ ,  $QD$  — выходные данные;  $CIN$  — входной перенос;  $COU$  — выход-



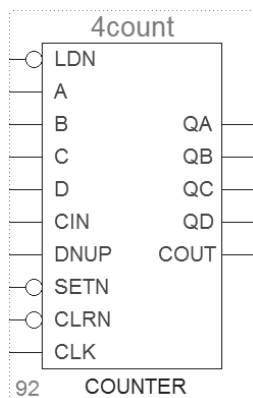


Рис. 8.6. Многофункциональный счетчик в ПЛИС

ной перенос; *LDN* (*Load Negative*) — сигнал, управляющий синхронной параллельной загрузкой входных данных, которая происходит по фронту синхроимпульса *CLK*, при *ldn* = 0; *SETN* — асинхронная параллельная загрузка; *CLRn* — асинхронная очистка; *DNUP* (*down-up* — вверх-вниз) — направление счета. Работа счетчика при различных комбинациях сигналов описана в *Help/Old-Style Macrofunctions/Counters*. Схема счетчика открывается двойным щелчком по символу.

Функциональные блоки большой степени интеграции с возможностью изменения параметров по желанию пользователя содержатся в библиотеке параметризованных модулей (*Library of Parameterized Modules — LPM*).

При вводе символа параметризируемого модуля отображается диалоговое окно настройки, позволяющее выбрать выходы модуля, которые будут отображены и активны, а также настроить параметры сигналов. Подробное рассмотрение библиотечных элементов выполнено в лабораторных работах.

## 8.4. СЧЕТЧИКИ С ПРОИЗВОЛЬНЫМ МОДУЛЕМ

Наибольшее распространение в цифровой схемотехнике имеют двоичные счетчики. Максимальную емкость двоичного счетчика определяет модуль  $M = 2^n$ , который выражается целой степенью числа 2. При использовании двоичного счетчика в качестве делителя частоты период повторения импульсов переполнения на выходе счетчика содержит  $2^n$  периодов входного сигнала (рис. 8.7).

Применительно к схеме рассмотренного 4-разрядного счетчика (см. рис. 8.6) период повторения импульсов переполнения  $T_{COUT}$  составит 16 периодов импульсов синхронизации  $T_{CLK}$ . Сигнал переполнения *COUT* соответствует максимальному значению кода и формируется в соответствии с логической функцией:  $COUT = QA \cdot QB \cdot QC \cdot QD$ . Используются обозначения сигналов, принятые для библиотечных элементов в САПР.

Для построения счетчика с произвольным модулем  $K$  используется способ исключения лишних состояний. Счетчики с произвольным модулем используются в качестве делителей частоты,

управляемых кодом, а также для формирования разнообразных импульсных сигналов.

Исключение в качестве лишних некоторого числа первых состояний достигается путем загрузки начального кода  $N$  при переполнении. Код счетчика в этом случае изменяется от значения  $N$  до  $2^n - 1$ , в результате период повторения импульсов переполнения составит  $2^n - 1 - N$  периодов входной частоты.

Управляемый делитель частоты с исключением последних состояний счетчика можно построить, если к выходу счетчика подключить компаратор, который формирует импульс принудительного сброса, если код на выходе  $N$  достиг определенного значения, заданного входным кодом  $N$ . Код счетчика в этом случае будет принимать значения от 0 до  $N$ , а период повторения импульсов сброса составит  $N + 1$  период синхросигнала.

Для построения счетчиков с заданным модулем в ПЛИС хорошо подходят библиотечные многофункциональные счетчики, однако более целесообразным является проектирование подобных устройств с описанием на HDL-языках.

## 8.5. ОПИСАНИЕ СЧЕТЧИКОВ НА ЯЗЫКЕ VERILOG

Функциональные возможности поведенческого описания устройств на языке *Verilog* особенно ярко проявляются при описаниях счетчиков и регистров.

Выполним описание 4-разрядного суммирующего счетчика, для которого заданы дополнительные управляющие входы *EN* и *R*. Вход *EN* выполняет разрешение счета, если *EN* = 1, а при *EN* = 0 счетчик должен находиться в режиме хранения. Сигнал *r* должен выполнять асинхронный сброс, который произойдет сразу же при появлении сигнала *r* = 0, по его отрицательному фронту (спаду).

Заголовок описания (пример 8.1) содержит имя модуля и перечисление всех сигналов. Все входные сигналы, перечисленные после слова *input* по умолчанию будут назначены как одноразрядные

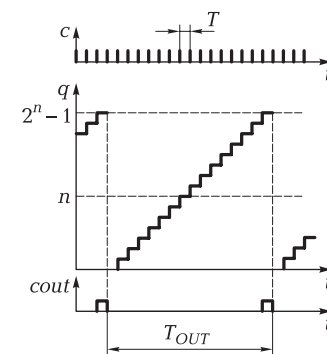


Рис. 8.7. Временные диаграммы, поясняющие работу счетчика

типа *wire*, выходной сигнал описан как 4-разрядный вектор, для которого указан тип, что является обязательным при описании устройств с элементами памяти.

```
//Пример 8.1. Суммирующий
//многофункциональный счетчик
module v81_cnt1 (en, c, r, q);
input c, r, en;
output [3:0] q;
reg [3:0] q;
always @ (posedge c or negedge r)
if (~r) q = 4'b0000;
else if (en) q = q + 1;
endmodule
```

В списке чувствительности оператора *always* указано срабатывание по фронту синхросигнала *c* или по спаду сигнала сброса *r*.

В качестве условия для оператора *if* записано инверсное значение сигнала *r*. При *r* = 0 счетчик сбрасывается, а при *r* = 1 и при *en* = 1 выполняется процесс счета.

Пример 8.2 описывает счетчик с параллельной загрузкой и асинхронным сбросом. Для подачи параллельного кода загрузки предусмотрена входная шина *d*, описанная как 4-разрядный вектор. При *r* = 0 произойдет сброс счетчика, при *r* = 1 и при *l* = 1 осуществляется загрузка кода *d* в счетчик. При *r* = 1 и *l* = 0 устанавливается режим прямого счета, т.е. будет выполняться прибавление единицы к коду счетчика по каждому фронту синхросигнала *c*.

```
//Пример 8.2. счетчик
// с параллельной загрузкой
module v82_cnt2 (q, d, l, c, r);
input l, c, r;
input [3:0] d;
output [3:0] q;
reg [3:0] q;
always @ (posedge c or negedge r)
if (~r) q = 4'b0000;
else if (l) q = d; else q = q + 1;
endmodule
```

В примере 8.3 описан счетчик с заданным модулем счета. В качестве входного сигнала указан 4-разрядный код *k*. Оператор *if* выполняет сброс счетчика при достижении равенства *q* = *k*.

```
//Пример 8.3. Счетчик с
//заданным модулем счета
module v83_cnt3 (q, k, c);
```

```
input c;
input [3:0] k;
output [3:0] q;
reg [3:0] q;
always @(posedge c)
if (q == k) q = 4'b0;
else q = q + 1;
endmodule
```

**Внимание!** Условие равенства обозначают два знака «равно». При этом модуль счета, определяемый количеством состояний счетчика, с учетом нулевого состояния, равен *k* + 1.

## 8.6. РАСПРЕДЕЛИТЕЛЬ ИМПУЛЬСОВ

**Распределитель импульсов** — схема, содержащая счетчик и дешифратор. Распределители импульсов используют для формирования различных импульсных сигналов с заданными параметрами, в качестве которых могут быть период повторения сигнала и интервалы времени, которым соответствуют единичные и нулевые значения сигнала. Выходные сигналы могут быть заданы в виде временных диаграмм.

Пусть задана временная диаграмма (рис. 8.8), в соответствии с которой выходной сигнал *у* изменяет свое значение в моменты времени, соответствующие фронтам импульсов сигнала синхронизации *с*. Интервалы времени, для которых сигнал *у* равен 1 или 0, заданы в виде количества периодов частоты синхронизации. Необходимо разработать распределитель импульсов, формирующий заданный сигнал.

В соответствии с заданной диаграммой период повторения импульсов равен восьми периодам тактовой частоты, поэтому требуется счетчик, содержащий три триггера. Для синтеза схемы дешифратора составим таблицу истинности, которая содержит восемь

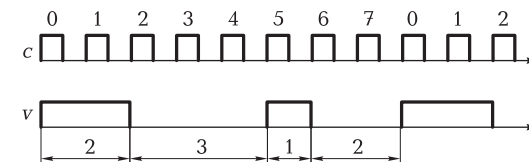


Рис. 8.8. Импульсный сигнал

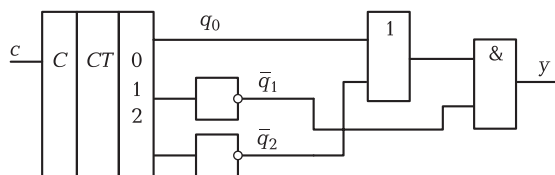


Рис. 8.9. Схема распределителя импульсов

Таблица 8.1

$q_2$	$q_1$	$q_0$	$y$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

строки (табл. 8.1). Периоды сигнала синхронизации на временной диаграмме необходимо пронумеровать и в таблице истинности учитывать, что заданный сигнал равен единице в нулевом, первом и пятом тактах. По таблице истинности составим логическую функцию, и выполним ее минимизацию.

$$y = \bar{q}_2 \bar{q}_1 \bar{q}_0 \vee \bar{q}_2 \bar{q}_1 q_0 \vee q_2 \bar{q}_1 q_0 = \\ = \bar{q}_2 \bar{q}_1 \vee \bar{q}_1 q_0 = \bar{q}_1 (\bar{q}_2 \vee q_0).$$

По уравнениям составлена схема, приведенная на рис. 8.9.

В описаниях распределителей импульсов на языке *Verilog* можно выделить две части. Одна из них посредством последо-

вательных операторов описывает счетчик, который может иметь произвольный модуль счета. Вторая часть описывает дешифратор с использованием параллельных операторов.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите области применения счетчиков.
2. Сформулируйте признаки классификации счетчиков.
3. По каким правилам организуются связи между триггерами суммирующего и вычитающего счетчиков?
4. Перечислите способы построения счетчиков с произвольным модулем счета.
5. Чем отличается двоичный счетчик от десятичного?
6. Постройте схему 4-разрядного суммирующего двоичного счетчика с модулем, равным 12.

7. Поясните принцип работы асинхронного суммирующего счетчика.
8. Опишите работу цепей переноса синхронного счетчика.
9. Поясните назначение входов и выходов библиотечного элемента *4count*.
10. Поясните назначение и принцип работы делителя частоты.

## ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

**Запоминающие устройства** (ЗУ), или **устройства памяти**, предназначены для записи, хранения и выдачи цифровой информации в процессе ее обработки.

В блоках памяти хранятся двоичные коды чисел, над которыми должны быть выполнены определенные действия, и коды команд, определяющие характер этих действий. Запоминающие устройства использовались сначала исключительно в ЭВМ, а в настоящее время они широко применяются в различных электронных устройствах: от систем управления до телевидения.

Для хранения одного бита информации нужен запоминающий элемент, а для хранения двоичного слова — **ячейка памяти** — упорядоченный набор запоминающих элементов. Совокупность ячеек памяти составляет ЗУ.

Количество запоминающих элементов в ячейке памяти называют **разрядностью**, или **длиной слова**. Наиболее употребительной разрядностью является 1 байт.

Основной параметр памяти — это ее объем или информационная емкость, т. е. количество кодов, которые могут в ней храниться, и разрядность этих кодов. Произведение числа хранимых слов на их разрядность, называемое организацией ЗУ, определяет максимальное количество хранимых бит.

На рис. 9.1 показана организация памяти  $16 \times 8$ , или 16 байт — 16 ячеек разрядностью 8 бит. Запоминающее устройство — набор ячеек для хранения данных, каждая из которых имеет определенный адрес. Разрядность адреса  $n$  опре-

	7	6	5	4	3	2	1	0
0000								
0001								
0010								
0011								
0100								
...								
1111								

Рис. 9.1. Организация памяти  $16 \times 8$

деляет максимальное количество адресуемых ячеек  $M$  в соответствии с очевидным выражением:  $M = 2^n$ .

Для емкости ЗУ обычно используют единицы, кратные числу  $2^{10} = 1024 = 1$  Кбайт. Емкость выражается в битах, байтах, килобайтах ( $1$  Кбайт  $= 2^{10} = 1024 \approx 10^3$  байт), мегабайтах ( $1$  Мбайт  $= 2^{20} \approx 10^6$  байт), гигабайтах ( $1$  Гбайт  $= 2^{30} \approx 10^9$  байт), терабайтах ( $1$  Тбайт  $= 2^{40} \approx 10^{12}$  байт).

Ширина выборки определяется числом разрядов, которые записываются в ЗУ или считываются из него за одно обращение.

Важным параметром запоминающих устройств является быстродействие, которое оценивают временами считывания, записи и длительностями циклов считывания- записи.

Основные параметры ЗУ находятся в противоречии. Так, например, увеличение информационной емкости сопровождается усложнением схем доступа к отдельным ячейкам памяти и снижением быстродействия.

В современных персональных компьютерах используют иерархическую память, содержащую несколько уровней:

- **регистровые ЗУ**, находящиеся в составе процессора или других устройств (т. е. внутренние для этих блоков), благодаря которым уменьшается число обращений к другим уровням памяти, реализованным вне процессора и требующим большего времени для операций обмена информацией;
- **кэш-память**, служащая для хранения копий информации, используемой в текущих операциях обмена. Высокое быстродействие кэш-памяти повышает производительность ЭВМ;
- **основная память** (оперативная, постоянная, полупостоянная), работающая в режиме непосредственного обмена с процессором и по возможности согласованная с ним по быстродействию. Исполняемый в текущий момент фрагмент программы обязательно находится в основной памяти;
- **внешняя память** для хранения больших объемов информации. Эта память реализуется на основе устройств с подвижным носителем информации (магнитные и оптические диски, магнитные ленты и др.);
- **специализированные виды памяти**, характерные для некоторых специфических архитектур (многопортовые, ассоциативные, видеопамять и др.).

По функциональному назначению устройства памяти разделяют на две группы: оперативные запоминающие устройства (ОЗУ) и постоянные запоминающие устройства (ПЗУ).

## 9.1. ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Постоянные запоминающие устройства предназначены для долговременного хранения данных, программ, констант, табличных функций и другой информации, которая записывается заранее и не изменяется в процессе работы компьютера. Они применяются в преобразователях кодов, знакогенераторах, микропрограммных устройствах управления. Общим свойством для всех микросхем постоянной памяти является энергонезависимость. Информация, хранящаяся в ПЗУ, не пропадает при выключении питания, но ее нельзя оперативно менять. Информация в ПЗУ записывается один раз либо в процессе их производства, либо непосредственно перед применением.

Основу ПЗУ (рис. 9.2) составляет матрица запоминающих элементов, предназначенных для хранения одного бита данных. Схемы запоминающих элементов для хранения одного бита данных в ПЗУ достаточно просты и содержат один транзистор или диод. Обращение к определенным запоминающим элементам обеспечивает дешифратор адреса.

Устройства памяти имеют типовые выводы, на которых действуют определенные адресные, информационные и управляющие сигналы. Назначение выводов и сигналов:

- *A (Address)* — входы адреса, разрядность которого определяет максимально возможное число данных (бит, байт, слов), которые хранятся в памяти и адресуются как единое целое;
- *D (Data)* — данные;
- *E (Enable — разрешить)* — сигнал разрешения работы устройства, при  $E = 1$  работа разрешена, а при  $E = 0$  — запрещена. В микросхемах старого типа применяли инверсный сигнал разрешения *cs (chip select)*.

Устройства постоянной памяти разделяются на несколько групп.

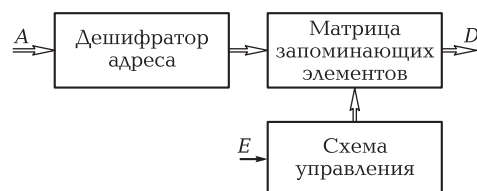


Рис. 9.2. Структура ПЗУ

**Постоянные запоминающие устройства — ПЗУ или ROM (*Read Only Memory* — память только для чтения).** Программируются однократно заводом-изготовителем.

**Внимание!** Одно и то же обозначение ПЗУ используется как общее для всех устройств постоянной памяти и как частное для рассматриваемого типа.

Запись информации в ПЗУ производится в процессе последней операции изготовления микросхемы — металлизации. Она выполняется при помощи маски, которая обеспечивает изготовление проводников. Поэтому такие устройства памяти получили название *масочных ПЗУ*. Масочные ПЗУ широко применяются при серийном производстве, так как являются самым дешевым видом ПЗУ. В микропроцессорных системах они используются для хранения данных, которые не потребуются изменять. Это программы начальной загрузки системы, стандартные программы вычислительных процессов, таблицы функций и констант.

**Программируемые ПЗУ — ППЗУ или PROM (*Programmable ROM*).** Программируются пользователем однократно электрическим способом. В этих микросхемах ПЗУ постоянное соединение проводников в запоминающей матрице заменяется плавкими перемычками, изготовленными из поликристаллического кремния.

Принцип записи информации в микросхемах ПЗУ основан на пережигании специальных внутренних перемычек. Каждая перемычка предназначена для хранения одного бита информации. Если перемычка цела, то это значит, что в данной ячейке хранится единственный бит информации. Если прожечь перемычку, то соответствующий бит примет нулевое значение.

При производстве микросхемы изготавливаются все перемычки, что эквивалентно записи во все ячейки памяти логических единиц. В процессе программирования на выводы питания и выходы микросхемы подается повышенное напряжение или низкий потенциал. При этом если на выход микросхемы подается повышенное напряжение питания (логическая единица), то через перемычку ток протекать не будет, она останется неповрежденной. Если же на выход микросхемы в режиме программирования подать низкий уровень напряжения (присоединить к общему проводу), то через перемычку будет протекать ток, который испарит ее, и при последующем чтении информации из этой ячейки будет считываться логический ноль.

ППЗУ используют при мелкосерийном производстве.

**Репрограммируемые ПЗУ — РПЗУ-УФ или EPROM (*Erasable PROM*).** Это многократно программируемые (репрограммируемые) с ультрафиолетовым стиранием и электрической записью.



Запоминающий элемент РПЗУ представляет собой МОП-транзистор, в котором затвор окружен оксидом кремния — диэлектриком с прекрасными изолирующими свойствами. Затвор со всех сторон окружен диэлектриком, как бы плавает внутри диэлектрика, поэтому его называют *плавающим затвором*.

В описанной ячейке при полностью стертом ПЗУ заряда в плавающем затворе нет, поэтому транзистор ток не проводит. При программировании микросхемы на программирующий электрод, находящийся над плавающим затвором, подается высокое напряжение, и на плавающем затворе индуцируются заряды. После снятия программирующего напряжения на плавающем затворе индуцированный заряд сохраняется, и, следовательно, транзистор остается в проводящем состоянии. Заряд на плавающем затворе может храниться десятки лет.

Стирание ранее записанной информации в репрограммируемых ПЗУ-УФ осуществляется ультрафиолетовым излучением. Для того чтобы оно могло беспрепятственно воздействовать на полупроводниковый кристалл, в корпус микросхемы РПЗУ встраивается окошко из кварцевого стекла. При облучении микросхемы изолирующие свойства оксида кремния теряются, накопленный заряд из плавающего затвора стекает в объем полупроводника, и транзистор запоминающей ячейки переходит в закрытое состояние. Время стирания микросхемы колеблется в пределах 10...30 мин. Количество циклов записи-стирания микросхем РПЗУ-УФ составляет от 10 до 100 раз, после чего вследствие разрушающего действия ультрафиолетового излучения микросхема выходит из строя.

В этих микросхемах в предыдущие десятилетия хранили программы BIOS персональных компьютеров. Из-за дороговизны корпуса с кварцевым окошком, а также вследствие такого недостатка, как сравнительно малое количество циклов записи-стирания, использование РПЗУ-УФ в настоящее время ограничено.

**Репrogramмируемые ПЗУ с электрическим стиранием РПЗУ-ЭС или EEPROM (Electrically EPROM).** Данные ПЗУ программируются и стираются многократно электрическими сигналами. РПЗУ-ЭС являются достаточно широко распространенными в настоящее время. В них используются такие же запоминающие ячейки, как и в РПЗУ, но они стираются электрическим потенциалом, поэтому количество циклов записи-стирания для этих микросхем достигает миллиона.

Время стирания ячейки памяти в таких микросхемах уменьшается до 10 мс.

Область применения электрически стираемых ПЗУ — хранение программ и данных, которые не должны разрушаться при выключении питания.

**FLASH-ПЗУ.** Микросхемы FLASH-ПЗУ отличаются от РПЗУ-ЭС тем, что производится стирание не каждой ячейки отдельно, а всей запоминающей матрицы или ее части (блока). Использование блочного стирания позволяет уменьшить сложность внутреннего устройства управления микросхем, поэтому FLASH-ПЗУ предоставляют максимальный объем матрицы запоминающих элементов. Кроме того, эти микросхемы обладают значительно меньшей стоимостью по сравнению с электрически стираемыми ПЗУ. В настоящее время FLASH-ПЗУ постепенно вытесняют все остальные виды постоянных запоминающих устройств за исключением электрически стираемых ПЗУ.

Достоинствами ПЗУ всех рассмотренных видов являются способность хранить информацию практически неограниченно долго и энергонезависимость; основной недостаток — низкое быстродействие.

Более высокое быстродействие имеют оперативные запоминающие устройства, теряющие свое содержимое при выключении питания.

## 9.2. ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Оперативные запоминающие устройства хранят данные, необходимые при выполнении текущей обработки данных, они могут обновляться в любой момент времени в высоком темпе работы процессора цифровой системы.

Структура оперативного запоминающего устройства (рис. 9.3) отличается от структуры ПЗУ наличием сигналов, обеспечивающих не только считывание, но и запись данных. В них используются следующие выводы сигналов:

- DI (Data Input) — шина входных данных;
- DO (Data Out) — шина выходных данных;

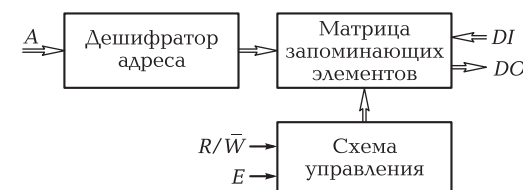


Рис. 9.3. Структура ОЗУ

- $R/\overline{W}$  (*Read or not Write*) — сигнал считывания данных (если сигнал равен 1) или записи данных (если сигнал равен 0).

Запоминающие устройства *RAM* (*Random-access memory* — оперативная память) подразделяются на статические *SRAM* (*Static RAM* — статическая оперативная память) и динамические *DRAM* (*Dynamic RAM* — динамическая оперативная память).

В статических ОЗУ элементами памяти являются триггеры. Они хранят свое состояние, пока схема имеет напряжение питания и новые данные не записываются. Основное достоинство статических ОЗУ — высокое быстродействие.

Основными требованиями к памяти являются максимально большая информационная емкость, высокое быстродействие (малое время обращения  $t_{обр} < 10$  нс), минимальная потребляемая мощность (менее 1 мкВт на 1 бит хранимой информации).

Статические ОЗУ используют для построения блоков кэш-памяти процессоров.

*Кэш-память* (от англ. *cache* — тайник) — это способ копирования и хранения блоков данных основной динамической памяти в процессе выполнения программы. Большинство прикладных программ имеет циклический характер и многократно использует одни и те же данные, поэтому наличие кэш-памяти уменьшает количество обращений к относительно медленной основной динамической памяти. Кэш-память строят на быстродействующих триггерных элементах памяти, она имеет небольшую емкость в сравнении с основной динамической памятью, но обеспечивает повышение быстродействия.

В динамических ОЗУ данные хранятся в виде зарядов миниатюрных конденсаторов, выполненных интегральным способом на кристалле кремния.

Каждый конденсатор хранит 1 бит информации. Входной сигнал через внутренние схемы коммутации поступает на этот конденсатор. Если его значение равно логической единице, то конденсатор заряжается, если логическому нулю — разряжается. Затем внутренний ключ отключает конденсатор от всех цепей, и заряженные конденсаторы в течение какого-то времени хранят свой заряд.

Но эти конденсаторы имеют очень маленькую емкость, поэтому свой заряд они сохраняют всего лишь несколько миллисекунд. Для того чтобы информация не потерялась, используют схему регенерации памяти.

Все ячейки памяти организуются как набор строк. Специальная схема периодически (каждые 2... 30 мс) считывает информацию из памяти строка за строкой. После считывания очередной строки

считанная информация опять записывается в те же ячейки памяти. Конденсаторы при этом подзаряжаются снова. Для нормальной работы динамического ОЗУ схема микропроцессорного устройства должна непрерывно обеспечивать такую регенерацию в течение всего времени работы системы. ОЗУ современных персональных компьютеров также построено по динамическому принципу. Современные микросхемы динамических ОЗУ имеют встроенную схему регенерации.

Плотность упаковки элементов памяти на кристалле динамических ОЗУ превышает в 4—5 раз этот же показатель для статических ОЗУ.

Динамические ЗУ характеризуются самой большой информационной емкостью и невысокой стоимостью, поэтому они используются как основная память компьютеров. Статические ЗУ в 4—5 раз дороже динамических, и приблизительно в столько же раз меньше их информационная емкость. Их достоинством является высокое быстродействие, а типовой областью применения — схемы кэш-памяти.

Быстродействие (производительность) ЗУ оценивают временами считывания, записи и длительностями циклов считывания-записи. Основными операциями в памяти являются запись и считывание определенной единицы информации, например байта. Эти операции называются также обращением к памяти. Память характеризуется информационной емкостью, физическим объемом, удельной емкостью и стоимостью, шириной выборки, потребляемой мощностью и быстродействием.

Важнейшие параметры ЗУ находятся в противоречии. Так, например, большая информационная емкость не сочетается с высоким быстродействием, а быстродействие, в свою очередь, не сочетается с низкой стоимостью. Поэтому системам памяти свойственна многоступенчатая иерархическая структура, и в зависимости от роли того или иного ЗУ его реализация может быть существенно различной.

### 9.3. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА В ПЛИС

Для реализации в ПЛИС модулей ОЗУ предусмотрено две возможности. Первую возможность предоставляет каждый логический блок, содержащий *D*-триггеры, из которых может быть построена регистровая память. Вторую возможность предоставляют отдельные блоки памяти *BlockRAM*, содержащиеся на кристалле ПЛИС. По-

является возможность использования ПЛИС для реализации различных устройств памяти внутри кристалла без использования внешних ЗУ.

Встроенные реконфигурируемые блоки памяти *FLEX 10K* и *ACEX 1K* большой информационной емкости имеются в ПЛИС фирмы *Altera*. Указанные серии являются популярными для реализации алгоритмов цифровой обработки сигналов, построения сложных устройств обработки данных.

Проектирование сложных цифровых устройств на ПЛИС в САПР *MAX + PLUS II* существенно упрощается при использовании готовых модулей типовых устройств, содержащихся в библиотеках. К числу этих модулей относятся параметризованные модули — **мегафункции**. Параметры этих модулей можно изменять. В библиотеке *mega-lpm* представлен модуль ПЗУ *LPM\_ROM* и три типа модулей ОЗУ. Модуль *LPM\_RAM\_DP* — двухпортовая память, позволяющая одновременно выполнять запись данных по одному адресу и считывание по другому адресу. Модуль *LPM\_RAM\_DQ* — характеризуется наличием отдельных шин для входных данных (шина *d*) и для выходных данных (шина *q*). В модуле *LPM\_RAM\_IO* использован выход с тремя состояниями для параллельной работы нескольких модулей на общую шину данных. Для большинства схем используют модули *LPM\_ROM* и *LPM\_RAM\_DQ* (рис. 9.4). Описание каждого модуля можно получить из *Help*, используя кнопку с изображением стрелки и вопросительного знака.

Параметризуемые модули допускают настройку сигналов и параметров с учетом конкретных требований. Настройка сигналов позволяет выбрать для отображения на схеме только те сигналы, которые необходимы при выбранном режиме работы. Настройка параметров позволяет указать разрядность и емкость памяти, а также выбрать асинхронные или синхронные режимы обращения к памяти.

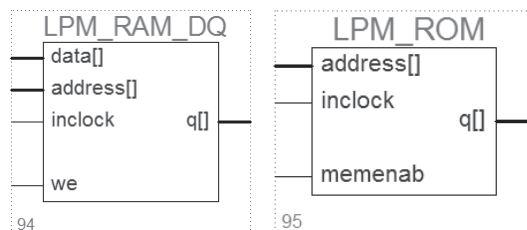


Рис. 9.4. Модули памяти в ПЛИС

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните назначение, перечислите типы запоминающих устройств.
2. Укажите основные параметры устройств памяти.
3. Поясните количественные меры информации — бит, байт, килобайт.
4. Перечислите элементы структуры постоянных запоминающих устройств.
5. Перечислите входные и выходные сигналы ОЗУ.
6. Почему данные в запоминающих устройствах хранятся в двоичном коде?
7. Охарактеризуйте принципы построения ПЗУ.
8. Какими достоинствами обладает динамическая память в сравнении со статической?
9. Укажите основные параметры запоминающих устройств.
10. Охарактеризуйте особенности флэш-памяти.

## СИНТЕЗ КОНЕЧНЫХ АВТОМАТОВ

Цифровое устройство, содержащее элементы памяти, называется **автоматом с памятью**, или **последовательностной схемой**, так как оно предназначено для выполнения целенаправленных действий без непосредственного участия человека, а последовательность выходных сигналов определяется последовательностью предыдущих состояний схемы и входными сигналами. В теории автоматов рассматриваются их классификация по ряду признаков. Для исследований функциональных возможностей автоматов используют математические модели абстрактных бесконечных автоматов с неограниченными ресурсами. При синтезе конкретных устройств используют другую модель — конечный автомат, все параметры которого конечны. Наибольший практический интерес представляют синхронные конечные автоматы, используемые как модели при синтезе цифровых устройств с памятью.

**Конечный автомат** — это цифровая схема, содержащая память и комбинационные схемы (рис. 10.1), которая под действием импульсов синхронизации  $s$  переходит из одного состояния  $q$  в другое и выдает выходные сигналы  $y$ , последовательность которых зависит от входных сигналов  $x$  и предыдущих значений сигналов  $q$ .

Рассмотрим назначение элементов структуры автомата.



Рис. 10.1. Структура синхронного конечного автомата

Память автомата предназначена для хранения текущего состояния и представляет собой набор из триггеров. Все триггеры подключены к общему источнику тактового сигнала  $s$ , поэтому состояния изменяются в каждом такте данного сигнала. В качестве элементов памяти обычно используют  $D$ -триггеры, что позволяет получить хорошие схемные решения автомата.

Комбинационная схема переходов формирует сигналы  $d$ , поступающие на входы триггеров памяти и вызывающие переход конечного автомата в следующее состояние, являющееся функцией текущего состояния и входного воздействия.

Комбинационная схема выходов формирует выходные сигналы  $y$ . В зависимости от способа формирования выходного сигнала различают два класса конечных автоматов. Если выходные сигналы зависят только от состояния  $q$  и связь, показанная на рис. 10.1 двойной штриховой линией, отсутствует, то конечный автомат называют **автоматом Мура** (Moore machine). Если выходные сигналы зависят от состояния  $q$  и от входных сигналов  $x$ , то конечный автомат называют **автоматом Мили** (Mealy machine). Функциональные возможности автоматов указанных классов идентичны. На практике чаще применяют автоматы Мура.

### 10.1. ПРИМЕР СИНТЕЗА КОНЕЧНОГО АВТОМАТА

Синтез конечного автомата содержит несколько этапов, формальное выполнение которых позволяет получить схему устройства, выполняющего заданные действия. Рассмотрим порядок выполнения синтеза автомата для управления светофором.

**Постановка задачи.** Началом синтеза является формулировка всех исходных данных, необходимых для проектирования. Должно быть составлено словесное описание проектируемого автомата, содержащее перечисление всех сигналов, а также описание логики формирования переходов и выходов.

**Задание.** Синтезировать автомат управления светофором на триггерах типа  $D$  в виде автомата Мура.

**Входные сигналы:**  $s$  — синхрои́мпульсы,  $s$  — сигнал «Старт»,  $k$  — сигнал от кнопки для включения сигнала переход.

**Выходные сигналы** (сигналы светофора для главной улицы):  $y$  — «Желтый»,  $g$  — «Зеленый»,  $r$  — «Красный»,  $p$  — «Переход».

**Словесное описание алгоритма работы.** При сигнале «Старт»  $s = 0$  должен быть постоянно включен сигнал «Желтый»  $y = 1$ .

При сигнале «Старт»  $s = 1$  автомат должен периодически переключать сигналы: «Желтый» — «Зеленый» — «Желтый» — «Красный».

Если нажата кнопка включения перехода ( $k = 1$ ), то после сигнала «Красный»  $r = 1$  должен включиться сигнал «Переход»  $p = 1$ , а затем должны периодически повторяться сигналы: «Желтый» — «Зеленый» — «Желтый» — «Красный».

**Алгоритм работы автомата.** При составлении алгоритма необходимо учитывать, что переключения сигналов светофора выполняется в результате изменения состояний автомата. В автомате Мура за состояния принимают исполняемые операторы алгоритма, которые обозначают прямоугольниками. Входные сигналы, определяющие условия смены состояний, отображают ромбами подобно условным операторам в программировании. В автомате Мура выходные сигналы являются функциями состояний, каждому состоянию соответствует определенный сигнал (рис. 10.2).

Для состояния  $q_0$  выбран желтый сигнал светофора. При  $s = 0$  автомат должен возвращаться в состояние  $q_0$ . При  $s = 1$  выполняется циклическое переключение состояний и выдача сигналов в соответствии с заданным словесным описанием алгоритма. Для автомата определено 5 состояний, которые обозначены как  $q_0...q_4$ . Для каждого состояния указан сигнал светофора.

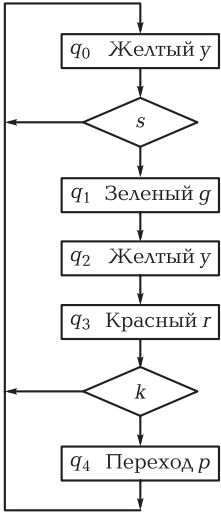


Рис. 10.2. Алгоритм работы конечного автомата

Для включения желтого сигнала предусмотрено два различных состояния, так как в этих ситуациях имеем различную предысторию состояний.

**Граф автомата.** Граф конечного автомата (другое название — *диаграмма состояний*) составляется по алгоритму, содержит вершины, соответствующие состояниям автомата, и дуги, отображающие переходы автомата из одного состояния в другое (рис. 10.3). Если переход является условным, то дуга обозначается соответствующим сигналом. Дуги, соответствующие безусловным переходам, не обозначаются вообще или обозначаются константой 1. На графе также отмечают выходные сигналы, соответствующие определенным состояниям.

**Выбор разрядности памяти.** Разрядность памяти  $n$  выбирается с учетом количества состояний автомата  $S$  и способа кодирования состо-

яний автомата. При двоичном кодировании разрядность должна обеспечивать выполнение условия:  $2^n \geq S$ , где  $2^n$  — ближайшее большее к  $S$  число, являющееся степенью числа 2. В рассматриваемом примере автомат имеет пять состояний, поэтому разрядность памяти  $n = 3$ . В схеме автомата память должна содержать три триггера. При построении схем на ПЛИС в качестве элемента памяти автомата предпочтение отдают  $D$ -триггеру.

Выберем коды состояний автомата:  $q_0 = 000$ ;  $q_1 = 001$ ;  $q_2 = 010$ ;  $q_3 = 011$ ;  $q_4 = 100$ .

Начало и конец алгоритма отображаются начальным состоянием  $q_0$ , при этом автомат будет циклическим, допускающим повторение рабочего цикла. Условия перехода из одного состояния в другое записываются на дугах графа в виде конъюнкций входных сигналов, принимающих единичное значение, когда данный переход происходит. При безусловном переходе дуга графа отмечается константой 1.

**Таблица переходов.** По графу составляется таблица переходов (табл. 10.1), которая описывает все переходы (или дуги) графа. Каждая строка таблицы соответствует одной дуге графа.

В первом столбце табл. 10.1 указаны входные сигналы, соответствующие описываемой дуге. Если переход, соответствующий дуге безусловный, то в этом столбце записывается константы 1. Второй столбец содержит двоичные коды исходного состояния автомата  $q$

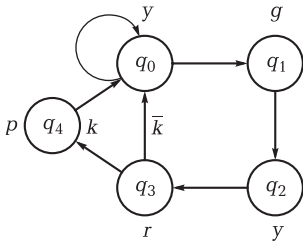


Рис. 10.3. Граф автомата управления светофором

Таблица 10.1			
Входные сигналы $x$	Исходное состояние $q(t)$	Новое состояние $q(t + 1)$	$d$
	$q_2 \ q_1 \ q_0$	$q_2 \ q_1 \ q_0$	$d_2 \ d_1 \ d_0$
$\bar{s}$ $s$	0 0 0	0 0 0	0 0 0
	0 0 0	0 0 1	0 0 1
	0 0 1	0 1 0	0 1 0
	0 1 0	0 1 1	0 1 1
	0 1 1	0 0 0	0 0 0
$k$	0 1 1	1 0 0	1 0 0
$\bar{k}$	1 0 0	0 0 0	0 0 0



( $t$ ) — вершину графа, из которой выходит дуга. Следующий столбец обозначают новое состояние автомата  $q(t+1)$  — вершину графа, в которую входит описываемая дуга.

Последний столбец содержит сигналы, которые необходимо подать на входы триггеров. Функция переходов для  $D$ -триггера имеет вид:  $q(t+1) = d(t)$ , в соответствии с которым входной сигнал в текущем такте должен быть равен требуемому состоянию в следующем такте. Поэтому столбец кодов для сигналов  $d$  является повторением столбца кодов для  $q(t+1)$ .

Первая строка таблицы описывает дугу, которая из вершины  $q_0$  возвращается в  $q_0$  при сигнале  $s = 0$ , поэтому в первом столбце этот сигнал записан с отрицанием. Вторая строка описывает переход из  $q_0$  в  $q_1$  при входном сигнале  $s = 1$ , а третья строка — переход из  $q_1$  в  $q_2$ , не зависящий от входных сигналов.

**Логические выражения для функций переходов.** По таблице переходов (см. табл. 10.1) составляются логические выражения для функций переходов.

Запишем выражение для сигнала  $d_2$ . Этому сигналу соответствуют крайние левые двоичные цифры в столбце  $d$ . Выделим строки, в которых  $d_2 = 1$ . Это всего одна строка, отмеченная в табл. 10.1 пунктиром. В соответствии с этой строкой автомат, находясь в состоянии  $q(t) = 011$ , при наличии сигнала  $k = 1$  должен сформировать  $d_2 = 1$ , чтобы обеспечить требуемое новое состояние, поэтому выражение для  $d_2$  имеет вид произведения сигналов, содержащихся в выделенной рамке:

$$d_2 = k \bar{q}_2 q_1 q_0.$$

Для записи выражения для сигнала  $d_1$  необходимо рассмотреть столбец двоичных цифр, соответствующих этому сигналу, и выделить строки, в которых сигнал равен 1. Сигнала  $d_1$  содержит две единицы в третьей и четвертой сверху строках таблицы, поэтому выражение для  $d_1$  будет содержать два произведения, соединенные знаком дизъюнкции.

$$d_1 = \bar{q}_2 \bar{q}_1 q_0 \vee \bar{q}_2 q_1 \bar{q}_0.$$

Сигналу  $d_0$ , который равен 1 во второй и четвертой строках таблицы, соответствует дизъюнкция двух произведений:

$$d_0 = s \bar{q}_2 \bar{q}_1 q_0 \vee d_1 \bar{q}_0.$$

**Логические выражения для функций выходов.** Выходные сигналы зависят от состояний автомата. В соответствии с обозначениями

Таблица 10.2

	$q$	$r$	$y$	$g$	$p$
$q_0$	000	0	1	0	0
$q_1$	001	0	0	1	0
$q_2$	010	0	1	0	0
$q_3$	011	1	0	0	0
$q_4$	100	0	0	0	1

ниями сигналов и состояниями автомата, приведенными на схеме алгоритма (см. рис. 10.2), можно составить таблицу выходов, которая описывает зависимость выходных сигналов от состояний  $q$  (табл. 10.2).

По таблице выходов записываются очевидные выражения для выходных сигналов:

$$y = \bar{q}_2 \bar{q}_1 q_0 \vee \bar{q}_2 q_1 \bar{q}_0 = \bar{q}_2 q_0;$$

$$g = \bar{q}_2 \bar{q}_1 q_0; r = \bar{q}_2 q_1 q_0; p = q_2 q_1 q_0.$$

**Схема конечного автомата.** Схема конечного автомата управления светофором (рис. 10.4) составлена по результатам выполнения

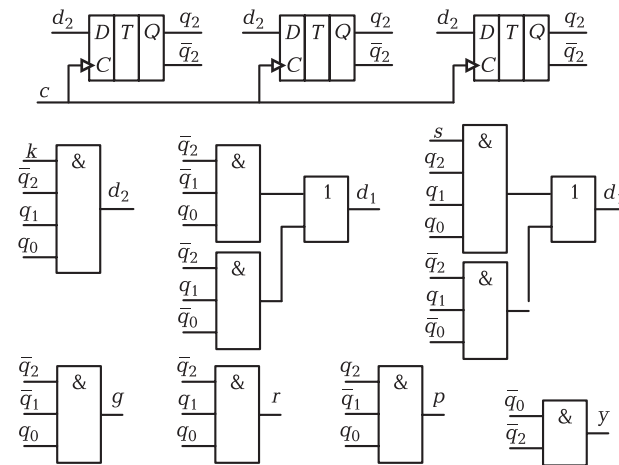


Рис. 10. 4. Схема автомата управления светофором

Таблица 10.3

$s$	$k$	Состояния	Выходы
0	0	000	$y$
0	1	000	$y$
1	0	000-001-010-011-000	$y-g-y-r-y-$
1	1	000-001-010-011-100-000	$y-g-y-r-p-y-$

предыдущих этапов. Схема содержит три триггера, образующие память, комбинационные схемы формирования сигналов перехода ( $d_2, d_1, d_0$ ) и комбинационные схемы формирования выходных сигналов ( $g, r, p, y$ ). Сигналы, имеющие одинаковые имена, являются соединенными. Такое соединение называется логическим.

**Тестирование автомата.** Для тестирования конечного автомата необходимо проверить правильность изменения состояний автомата при различных входных сигналах. Необходимо по графу составить таблицу изменений состояний автомата при всех возможных комбинациях входных сигналов и выделить подчеркиванием циклы повторяющихся состояний (табл. 10.3). Необходимо проверить формирование всех выходных сигналов, соответствие алгоритму моментов их появления. Последовательность состояний, формируемая при  $s = 1, k = 1$ , соответствует циклу алгоритма, в котором имеются все состояния. Такой цикл целесообразно использовать при отладке автомата.

## 10.2. СИНТЕЗ РЕВЕРСИВНОГО СЧЕТЧИКА ПО МОДУЛЮ 3

Методика проектирования цифровых устройств в виде конечных автоматов позволяет синтезировать счетчики с заданным модулем счета и любым порядком смены состояний, например двоично-десятичные, реверсивные, счетчики в коде Грея. В качестве элементов памяти чаще всего используются  $D$ -триггеры с динамическим управлением.

**Постановка задачи.** Синтезировать реверсивный счетчик с модулем счета 3 на триггерах типа  $D$  в виде автомата Мура. Направление счета определяет управляющий сигнал  $n$ .

**Граф автомата.** Граф автомата (рис. 10.5) должен иметь три состояния. Пусть направление счета определяет входной сигнал  $n$  (negative), при  $n = 1$  — вычитание, а при  $n = 0$  — суммирование.

**Выбор разрядности памяти и кодирование состояний автомата.** Необходимое число триггеров определяется из условия  $2^n \geq S$ , где  $S$  — количество состояний автомата, а  $n$  — количество триггеров. Для заданного счетчика, имеющего три состояния, потребуется два триггера, при этом одно состояние будет лишним. При кодировании состояния автомата необходимо исключить одно лишнее состояние, например 11. Для состояний автомата выберем коды:  $q_0$  — 00,  $q_1$  — 01,  $q_2$  — 10.

**Таблица переходов.** Таблица переходов автомата (табл. 10.4) составлена по графу, имеет столбцы:  $n$  — направление счета,  $q(t)$  — текущее состояние автомата,  $q(t+1)$  — новое состояние автомата,  $d$  — управляющие сигналы триггеров,  $d_1$  — старший разряд,  $d_2$  — младший разряд.

Первая строка записана следующим образом. Задан сигнал  $n = 0$ , что соответствует режиму суммирования. Из состояния  $q(t) = 00$  в этом случае должен произойти переход в следующее состояние  $q(t+1) = 01$ . Для выполнения такого перехода на входы триггеров необходимо подать сигналы  $d = q(t+1) = 01$ . Подобным образом записаны остальные строки.

**Логические выражения для функций переходов.** По таблице переходов (см. табл. 10.4) составляются выражения для функций переходов:

$$d_1 = \overline{nq_1q_0} \vee n\overline{q_1q_0}; \quad d_0 = \overline{nq_1q_0} \vee nq_1\overline{q_0}.$$

Таблица выходов в данном случае не составляется, коды состояний  $q$  являются выходными сигналами  $y = q$  и комбинационная схема выходов, показанная на рис. 10.1, отсутствует.

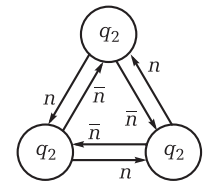


Рис. 10.5. Граф автомата счетчика по модулю 3

Таблица 10.4

$n$	$q(t)$		$q(t+1)$		$d$	
	$q_1$	$q_0$	$q_1$	$q_0$	$d_1$	$d_0$
0	0	0	0	1	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	0
1	0	0	1	0	1	0
1	1	0	0	1	0	1
1	0	1	0	0	0	0

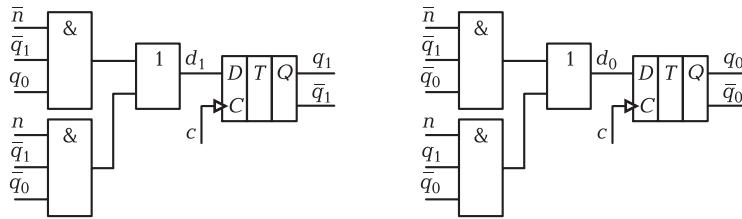


Рис. 10.6. Схема счетчика по модулю 3

Таблица 10.5

m	q
0	00-01-10-00-
1	00-10-01-00-

**Схема конечного автомата.** Логические выражения для функций переходов позволяют составить схему синтезированного счетчика (рис. 10.6).

**Тестирование автомата.** Для тестирования автомата необходимо по графу составить таблицу смены состояний графа при всех комбинациях управляющих сигналов (табл. 10.5). Данной таблице должны соответствовать временные диаграммы, полученные при моделировании автомата.

### 10.3. ОПИСАНИЕ КОНЕЧНЫХ АВТОМАТОВ НА ЯЗЫКЕ VERILOG

Функциональные возможности использования языка *Verilog* особенно ярко проявляются при описаниях конечных автоматов и сложных цифровых схем. Рассмотрим примеры описания счетчиков.

Для описания автомата на языке *Verilog* необходимо составить алгоритм и граф автомата. Основу описания автоматов составляют последовательные операторы *case* и *if* с ключевым словом *always*. В описаниях конечных автоматов используют последовательные операторы. Для таких операторов можно указать событие, по которому должно происходить срабатывание. Данный выбор обусловлен тем, что память конечных автоматов построена на синхронных триггерах с динамическим управлением, которые срабатывают по фронту импульсов синхронизации.

Существуют различные варианты использования указанных операторов. Наиболее наглядное описание получается, если для переходов автомата из одного состояния в другое используется

оператор варианта *case*, в котором селектором варианта является состояние графа *q*, а затем в качестве значения селектора указывается исходное состояние автомата, а в качестве выполняемого действия — присваивание нового состояния. Селектор оператора варианта — это переменная, записываемая в скобках после ключевого слова *case*.

Если переход автомата из одного состояния в другое зависит от входного сигнала, то необходимо использовать оператор *if*, в котором в качестве условия указан данный входной сигнал.

Описание комбинационной схемы выходов, которая не содержит элементов памяти, выполняют параллельными операторами с ключевым словом *assign*.

В примере 10.1 приведено описание автомата управления светофором, синтез которого выполнен в подразд. 10.1. В строке 2 указано имя модуля и перечислены все входные и выходные сигналы, в строке 3 определены входные сигналы, по умолчанию они будут одноразрядными типа *wire*. В строках 4, 5 указан выходной сигнал *q* в виде 3-разрядного вектора (или шины) типа *reg*. Этот тип сигнала необходим, чтобы использовать последовательные операторы в строках 8...13. Выходные сигналы *r*, *y*, *g*, *p* указаны в строке 6 как одноразрядные типа *wire*. Для формирования этих сигналов использованы параллельные операторы (строки 14...17).

```
//Пример 10.1. Автомат светофора
module v101_light (c,s,k,q,r,y,g,p);           //2
input c,s,k;                                   //3
output [2:0]q;                                 //4
reg [2:0]q;                                    //5
output r,y,g,p;                                //6
always @ (posedge c)                           //7
case (q)                                        //8
3'b000: if(s) q = 3'b001; else q = 3'b000; //9
3'b001: q = 3'b010;                             //10
3'b010: q = 3'b011;                             //11
3'b011: if(k) q = 3'b100; else q = 3'b000; //12
default: q = 3'b000; endcase                    //13
assign r = ~q[2] & q[1] & q[0];                 //14
assign y = ~q[2] & ~q[0];                       //15
assign g = ~q[2] & ~q[1] & q[0];                 //16
assign p = q[2] & ~q[1] & ~q[0];                 //17
endmodule                                       //18
```

В строке 13 исходное значение селектора указано как *default*. При выполнении этой строки будет выбран любой вариант, отсутствующий среди перечисленных ранее вариантов. Это не только

состояние 100, в котором может находиться автомат, но и все возможные другие состояния, возникающие при включении схемы, или в результате сбоя.

Селектором оператора *case* является исходное состояние (*q*). Коды, определяющие исходные и новые состояния автомата, записаны как константы в двоичной системе счисления. Первый элемент записи константы — число (десятичное), равное количеству разрядов; второй элемент — апостроф; третий элемент — буква, определяющая систему счисления (*b* — двоичная); четвертый элемент — число в указанной системе счисления.

В примере 10.2 приведено описание реверсивного счетчика, синтез которого выполнен в подразд. 10.2.

```
//Пример 10.2. Реверсивный счетчик
module v102_revers (c,n,q);           //2
input c,n;                           //3
output [1:0]q;                       //4
reg [1:0]q;                          //5
always @ (posedge c)                //6
case (q)                             //7
2'b00: if(n) q=2'b10; else q=2'b01; //8
2'b01: if(n) q=2'b00; else q=2'b10; //9
default: if(n) q=2'b01; else q=2'b00; //10
endcase                             //11
endmodule                           //12
```

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое конечный автомат?
2. Изобразите структуру конечного автомата, поясните назначение сигналов.
3. Перечислите этапы синтеза конечного автомата.
4. Поясните методику построения алгоритма и графа конечного автомата.
5. Как выбирается число триггеров для описания состояний автомата КА?
6. Как на графах переходов обозначаются состояния и переходы автомата?
7. Как происходит описание конечных автоматов на языке Verilog?

## Глава 11

# МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

**Микропроцессорная система** — это специализированная вычислительная, информационная или управляющая система, построенная на микросхемах. В качестве примеров микропроцессорных систем можно указать персональный компьютер, системы управления летательными объектами, электронику двигателя автомобиля и простейшие системы управления бытовыми приборами.

В каждой микропроцессорной системе можно выделить устройства обработки, хранения и обмена данными — это процессор, устройства памяти и устройства ввода-вывода (или периферийные устройства).

## 11.1. АРХИТЕКТУРЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Архитектура микропроцессорной системы отражает логическое построение системы, совокупность устройств и связей между ними.

В архитектуре Джона фон Неймана отдельные устройства микропроцессорной системы соединены между собой при помощи трех шин: шины данных (*Data Bus*), шины адреса (*Address Bus*), шины управления (*Control Bus*). Все вместе они образуют системную шину (рис. 11.1). Каждая шина представляет собой набор проводников для передачи сигналов между устройствами системы. Названия шин отражают тип передаваемых сигналов.

Главным элементом микропроцессорной системы является процессор, который выполняет арифметические и логические операции с числами, а также управляет процессом обработки. Процессор читает числа из устройств ввода-вывода или памяти, выполняет

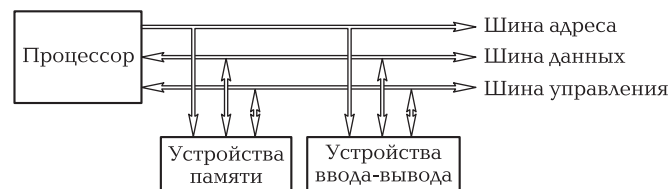


Рис. 11.1. Архитектура системы Джона фон Неймана

математическую обработку полученных данных и записывает результаты в память или устройства вывода.

Устройства ввода-вывода, называемые периферийными устройствами, являются функционально законченными модулями, подключенными к системной магистрали. В простейшем случае это буферные регистры ввода или вывода. Функциональный состав периферийных устройств современных микропроцессорных устройств исключительно широк.

Все обмены данными происходят через *шину данных*. Разрядность шины данных зависит от разрядности процессора. Это количество бит информации, которое процессор может обработать с помощью одной команды. В простых микропроцессорных системах шина данных имеет восемь разрядов, или 1 байт. Более мощные процессоры могут иметь шину данных разрядностью 2, 4 или 8 байт. Увеличение разрядности шины данных повышает производительность системы.

*Шина адреса* представляет собой набор проводников, по которым происходит передача адресов ячеек памяти, устройств ввода или устройств вывода, к которым в данный момент обращается процессор. Количество разрядов адресной шины ( $n$ ) определяет максимальное количество ячеек памяти ( $M$ ), которое может адресовать процессор  $M = 2^n$ , называемое **адресным пространством**.

Шина управления представляет собой набор проводников, передающих различные управляющие сигналы от процессора на все периферийные устройства и обратно. В шине управления обязательно присутствуют сигналы считывания или записи памяти или устройств ввода-вывода.

В архитектуре Джона фон Неймана для хранения программы и данных используется единое адресное пространство, или единая система адресации ячеек памяти и регистров периферийных устройств. Достоинствами такой архитектуры являются более простая внутренняя структура процессора, меньшее количество управ-

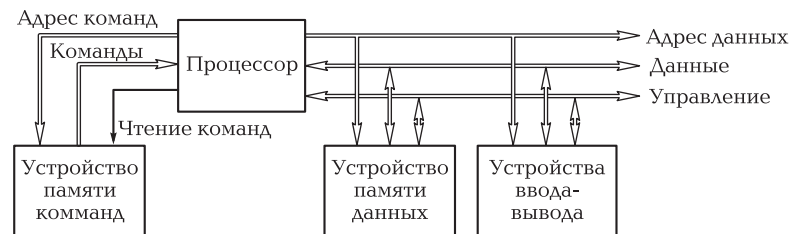


Рис. 11.2. Гарвардская архитектура микропроцессорной системы

ляющих сигналов, а также универсальность, возможность создания программ, которые сами себя модифицируют (например, компьютерные вирусы). В ЭВМ общего назначения и в персональных компьютерах неймановская архитектура используется достаточно часто.

Другой тип архитектуры — гарвардская (разработана в Гарвардском университете) — предусматривает использование независимых систем адресации для хранения программ и данных (рис. 11.2).

Гарвардская архитектура является более сложной, содержит отдельные шины и устройства для передачи команд и данных, однако она имеет ряд преимуществ.

Гарвардская архитектура позволяет использовать блоки памяти различных типов и различной разрядности для хранения команд и данных, а также позволяет простыми техническими средствами организовать конвейер команд и синхронизировать выполнение операций с адресами и данными.

Микроконтроллеры, построенные, как правило, по гарвардской архитектуре, содержат различные устройства памяти. Для программы, которая в процессе работы изменяться не должна, используют постоянное запоминающее устройство, а для хранения данных — оперативное запоминающее устройство.

## 11.2. ВИДЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ И ИХ ФИЗИЧЕСКАЯ РЕАЛИЗАЦИЯ

Элементной базой для построения вычислительных систем в XX в. служили микропроцессорные наборы, содержавшие комплекты микросхем, позволявших собрать систему с заданными параметрами и свойствами. В новых разработках микропроцессорные наборы, в основном снятые с производства, не используются. Современные технологии позволяют разместить на одном кристалле огромное



количество логических элементов, составляющих сложную систему, содержащую не только процессорное ядро, но и различные периферийные устройства. В настоящее время микропроцессорные системы разрабатывается СБИС.

**Процессоры высокой производительности.** Они предназначены для построения сложных систем обработки данных, содержат на одном кристалле СБИС быстродействующее процессорное ядро (или несколько ядер) большой разрядности (32...64 разряда), а также быстродействующие запоминающие устройства (кэш-память) и периферийные устройства.

В зависимости от числа операций системы команд процессоры подразделяются на две группы: CISC и RISC. Термин *CISC (Complex Instruction Set Computer)* означает компьютер со сложной системой команд, а термин *RISC (Reduce Instruction Set Computer)* — с сокращенной системой команд.

Основная идея *RISC*-архитектуры — это тщательный подбор таких команд, которые процессор может выполнить за один такт сигнала синхронизации. Основной выигрыш от такого подхода — упрощение аппаратной реализации процессора и возможность значительно повысить его производительность.

Выделяют также специализированные процессоры, предназначенные для решения определенного типа задач, например цифровые сигнальные процессоры *DSP (Digital Signal Processor)*. Назначение *DSP* состоит в том, чтобы получать текущие сигналы от аналоговой системы, обрабатывать данные и сформировать соответствующий отклик в реальном времени. Структура *DSP* имеет особенности, позволяющие им с наибольшей производительностью осуществлять алгоритмы обработки данных, которые используются во многих задачах обработки аналоговых и цифровых сигналов. Это кодирование, регулирование, цифровая фильтрация, вычисления определенных параметров сигналов.

**Микроконтроллеры.** Это микропроцессорные системы, содержащиеся на одном кристалле не только процессор, но также память и многочисленные периферийные устройства: последовательные и параллельные каналы передачи данных, аналого-цифровые и цифроаналоговые преобразователи, таймеры реального времени, генераторы программируемых импульсов и многое другое. Микроконтроллеры содержат все составные части вычислительной машины, поэтому их также называют *микроЭВМ*. Основное назначение микроконтроллеров — работа в системах автоматического управления различными устройствами и процессами (от сотовых телефонов и видеокамер до сложных объектов и технологических процессов).

**Система на кристалле** — *СнК (System On Chip — SoC)*. Это сложная система обработки данных, реализованная на основе СБИС, в состав которой входит не только матрица конфигурируемых логических блоков, но и микропроцессорная система. Для реализации СнК на основе ПЛИС используются кристаллы, которые имеют достаточно большой объем логических элементов, содержат блоки памяти и позволяют создавать достаточно сложные цифровые устройства. Проектирование СнК является современным, динамично развивающимся направлением развития цифровой схемотехники.

Процессорное ядро, содержащееся в СнК, может быть реализовано аппаратно, как часть кристалла, или программно в виде проекта, введенного в ПЛИС в виде файла конфигурации. Процессор, реализованный программно, называется *виртуальным*, или *синтезированным*. Он представляет собой проект, разработанный с использованием САПР, представляющий собой комплекс модулей, представленных в виде схем и описаний на *HDL*-языке. Современные системы проектирования устройств на основе ПЛИС позволяют разработать специализированные процессоры для решения задач определенных типов. Ввод проекта в САПР и программирование ПЛИС позволяет получить в этой ПЛИС аппаратную реализацию созданного процессора. В дополнение к процессору на том же кристалле могут быть созданы модули обработки данных и различные устройства ввода и вывода.

Разрабатывать универсальный процессор на основе ПЛИС не имеет смысла. Подобная задача под силу лишь большому коллективу профессионалов. Однако разработка несложных специализированных процессоров актуальна и целесообразна. При выполнении конкретной задачи определенного типа синтезированный процессор оказывается гораздо компактнее универсального процессора, что позволяет в большинстве случаев получить выигрыш в быстродействии, аппаратных затратах и потребляемой мощности. Кроме того, синтезируемые процессоры работают синхронно со всеми остальными устройствами, расположенными в этой же микросхеме, что существенно упрощает обмен данными.

### 11.3. СТРУКТУРЫ ПРОЦЕССОРОВ

Проектирование процессора на основе ПЛИС начинается с выбора структуры процессора и способа размещения операндов, участвующих в операции, и полученного результата. Современные процессоры имеют наборы регистров, которые располагаются вну-

три процессора в непосредственной близости от его АЛУ, что обеспечивает быстрый физический доступ к информации, хранящейся в них. Регистровую память процессора выделяют в отдельную группу и называют **сверхоперативной**. В регистрах процессора хранятся промежуточные результаты вычислений, которые нередко используются. Одни и те же регистры могут использоваться для хранения как данных, так и адресной информации. Такие регистры называются **регистрами общего назначения**.

По структуре процессоры отличаются огромным разнообразием. Тем не менее можно определить две характерные структуры построения, на основе которых строятся различные процессоры: это процессоры аккумуляторного типа и процессоры с блоком регистров общего назначения.

В *процессорах аккумуляторного типа* (рис. 11.3) используется АЛУ.

Достоинство процессоров данного типа — возможность использования одноадресных команд. Арифметические и логические команды используют в качестве одного из своих операндов исходное содержимое аккумулятора, а после выполнения операции сохраняют результат в том же аккумуляторе. Данная структура находит применение в микрокалькуляторах, а также для построения накапливающих сумматоров, которые позволяют прибавлять к ранее полученной сумме новые слагаемые.

К недостаткам процессора аккумуляторного типа можно отнести относительно низкое быстродействие, объясняемое тем, что каждый раз перед выполнением операции необходимо переслать в аккумулятор один из операндов, а после выполнения операции переслать результат из аккумулятора в ячейку памяти.

В *процессорах с блоком РОН* (рис. 11.4) исходные данные (операнды) и результат математических операций могут находиться в любом регистре. Каждый РОН может выполнять функции аккумулятора. Расширяются возможности по манипуляции данными.

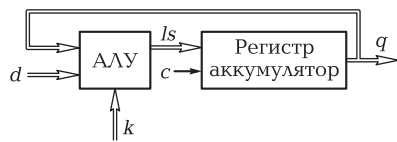


Рис. 11.3. Структура процессора аккумуляторного типа

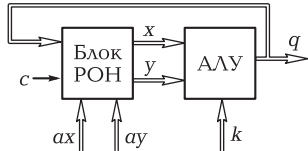


Рис. 11.4. Структура процессора с блоком РОН

Приведенные простые архитектуры необходимы для рассмотрения принципов проектирования вычислительных устройств на основе ПЛИС. Процессоры, выпускаемые в настоящее время, могут содержать несколько блоков регистров различного функционального назначения и обладать сложной системой команд с признаками как аккумуляторных процессоров, так и процессоров с регистрами общего назначения.

## 11.4. ОПИСАНИЕ АЛУ РЕГИСТРОВОГО ТИПА НА ЯЗЫКЕ VERILOG

В простых процессорах для хранения результата выполняемой операции используют всего один регистр, называемый **аккумулятором**. Сочетание комбинационной схемы *KS*, выполняющей арифметико-логические операции, и регистра *RG* позволяет получить АЛУ регистрового типа (рис. 11.5) — простейшее ядро процессора.

Комбинационная схема, представляющая собой АЛУ комбинационного типа, рассмотренное ранее, выполняет арифметические или логические операции, задаваемые кодом операции *k*, и формирует логический сигнал *ls*, который записывается в регистр синхронным импульсом *c* и определяет новое значение выходного кода *q*.

В настоящее время процессоры проектируют в виде описаний на языках *VHDL* или *Verilog* и реализуют в ПЛИС. Такие процессоры называют **встраиваемыми**, или **синтезируемыми**. Ввод данных в виде схем для задач подобного типа сложен или вообще невозможен.

Рассмотрим описание на языке *Verilog* АЛУ регистрового типа.

**Постановка задачи.** Требуется описать на языке *Verilog* 4-разрядное АЛУ регистрового типа на четыре операции. Операция, выполняемая устройством, задана двухразрядным кодом операции ( $k_1 k_0$ ) в соответствии с табл. 11.1.

Таблица 11.1	
Код операции $k_1 k_0$	Операция
00	$q := q + d$
01	$q := q * d$
10	$q := q \vee d$
11	$q := q \oplus d$

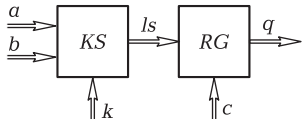


Рис. 11.5. Структура АЛУ регистрового типа

В соответствии со структурой проектируемого устройства (см. рис. 11.5), в описании (пример 11.1) для формирования сигнала *ls* используется параллельное присваивание *assign* (строка 10) подобно ранее рассмотренному примеру 5.12, а для формирования сигнала *q*, для которого указан тип *reg* — последовательное присваивание (строка 11).

```
//Пример 11.1. АЛУ
// аккумуляторного типа
module v75_alu (c, d, k, q, ls); //3
input c; //4
input [3..0] d; //5
input [1..0] k; //6
output [3..0] q; //7
output [3..0] ls; //8
reg [3..0] q; //9
assign ls = (k = 0)? q + d:
(k = 1)? q & d:(k = 2)? q | d: q^d; //10
always @ (posedge c) q = ls; //11
endmodule //12
```

Обработка данных в цифровых устройствах сопровождается временными задержками сигналов (гонками), которые могут вызывать ошибки. Для исключения ошибок, обусловленных гонками, используют синхронизацию.

В работе данного устройства можно выделить два действия, выполняемых последовательно. Первое из них — это подача данных на вход данных и кода операции и формирование логического сигнала *ls* в соответствии с заданной операцией. Второе действие — запись результата операции в регистр. Указанные действия должны выполняться последовательно во времени. Запись должна выполняться с задержкой относительно ввода данных

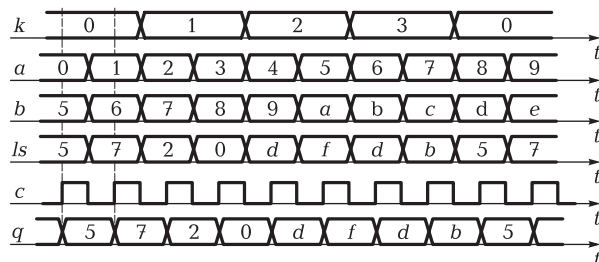


Рис. 11.6. Двухфазная синхронизация АЛУ регистрового типа

и выполняться в такой момент времени, когда переходные процессы закончились.

Выполнение отдельных действий в два этапа обеспечивает двухфазная синхронизация, при которой одно из действий выполняется в момент спада синхроимпульса, а другое — в момент фронта (рис. 11.6).

Временные диаграммы иллюстрируют работу устройства с двухфазной синхронизацией. Входные сигналы *k*, *a*, *b* формируются от суммирующих счетчиков. Изменение этих сигналов происходит по спаду синхроимпульсов. Запись результата выполнения операции (сигнал *ls*) происходит по фронту синхроимпульсов, как показано на диаграммах пунктиром. В моменты записи данных сигналы не содержат ошибок, обусловленных гонками.

## 11.5. СИСТЕМА КОМАНД ПРОЦЕССОРА

Важным моментом при проектировании процессора на основе ПЛИС является выбор набора выполняемых команд.

Процессор является программно управляемым цифровым устройством обработки данных. Программы, выполняемые процессором, состоят из команд, описывающих операции, которые необходимо выполнить.

Совокупность всех команд, которые может выполнить процессор, называется **системой команд**. Число команд, которые могут быть реализованы в процессоре, может составлять от десятков до нескольких сотен. Система команд процессора зависит от его структуры и аппаратной реализации. Каждая модель процессора имеет свою систему кодирования команд.

В команде можно выделить группы бит, называемые *полями*. Формат команды определяет ее деление на поля различного функционального назначения. Каждая команда характеризуется разрядностью. Существуют процессоры и микроконтроллеры с фиксированной разрядностью команд. В сложных процессорах команды разных форматов имеют различную разрядность.

В команде содержатся поле кода операции и адресная часть. Код операции определяет, какое действие должен выполнить процессор, а адресная часть показывает, где находятся исходные данные (операнды) и куда поместить результат. Код операции и адресная часть могут содержать несколько полей.

Для определения места размещения операндов используют различные методы адресации. Наиболее употребительными из них являются четыре метода:

- **непосредственная адресация** — в команде содержится сам операнд;
- **регистровая адресация** — в команде указан регистр, содержащий операнд;
- **прямая адресация** — в команде указан адрес операнда, находящегося в памяти;
- **косвенная (непрямая) адресация** — в команде указан регистр, содержащий адрес операнда, который находится в памяти.

Программа, по которой работает процессор, представляет собой последовательность команд, записанных в виде двоичных кодов в соответствии с системой команд данного процессора на машинном языке. **Машинный язык** — это естественный язык процессора — набор двоичных чисел, которыми закодированы содержание команд, номера ячеек памяти или регистров, константы, содержащиеся в команде. Команды процессора являются двоичными числами, их можно записать в шестнадцатеричном коде. Однако использование машинного языка при программировании процессоров весьма неудобно, работа с большими массивами чисел сопровождается определенными сложностями, связанными с необходимостью запоминания кодов команд и с сложностью восприятия программы, представленной кодами.

В целях облегчения составления программ для процессоров разработаны специальные языки — ассемблеры, в которых коды машинного языка заменены специальными символами (буквенными или буквенно-цифровыми), что облегчает их запоминание и применение. Перевод с языка ассемблера на машинный язык производит специальная программа, которую также называют *ассемблером*. Это название переводится как «сборщик программ».

Для удобства запоминания и записи команд при программировании вместо кодов применяют мнемоники команд, являющиеся сокращениями английских слов. Название «мнемоника» происходит от имени богини памяти Мнемозины.

Язык ассемблера, называемый также **языком символического кодирования**, представляет собой машинный язык, в котором вместо кодов команд используют мнемоники, а для обозначения переменных допускается применение буквенно-цифровых имен. Язык ассемблера является машинно ориентированным языком, в котором учитываются аппаратные средства, система команд и методы адресации конкретного процессора. Каждый процессор имеет «свой ассемблер».

Для программирования процессоров кроме машинного языка и языка типа ассемблера, используют языки высокого уровня —

Паскаль или Си, облегчающие описание сложных алгоритмов. Программы, написанные на различных языках, компилируются посредством специальных программ — компиляторов — для получения программы на машинном языке.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое микропроцессорная система?
2. Назовите основные составные части микропроцессорной системы.
3. Охарактеризуйте неймановскую архитектуру микропроцессорной системы.
4. Охарактеризуйте гарвардскую архитектуру микропроцессорной системы.
5. Что такое процессор?
6. Охарактеризуйте *RISC*-архитектуру процессора.
7. Охарактеризуйте *CISC*-архитектуру процессора.
8. Опишите назначение АЛУ.
9. Нарисуйте и поясните структуру процессора.
10. Что такое микроконтроллер?
11. Назовите характеристики микропроцессора.
12. Назовите виды архитектур процессоров.

## РАЗРАБОТКА ПРОЦЕССОРА ДЛЯ РЕАЛИЗАЦИИ В ПЛИС

Современный уровень проектирования цифровых вычислительных устройств характеризуется широким использованием программируемых логических интегральных схем типа система на кристалле (System On Chip) для построения высокопроизводительных систем обработки данных. В указанных системах для управления вычислительным процессом используют синтезированные процессоры, которые разрабатывают в виде проектов в САПР. Разработка процессора в ПЛИС позволяет изучить работу процессора и методику его проектирования.

Для проектируемого процессора выбрана гарвардская *RISC*-архитектура, а также устройство обработки данных, содержащее АЛУ с блоком РОН. В целях получения высокого быстродействия использована двухфазная синхронизация для обработки команд и данных. В результате разрабатываемый процессор имеет предельно высокое быстродействие, любая команда выполняется за один такт частоты синхронизации.

Для процессора выбраны параметры: разрядность данных — 8, разрядность команд — 16. Состав системы команд ограничен и содержит некоторые арифметические и логические операции с непосредственной и регистровой адресацией, команды безусловного и условного переходов. При необходимости состав команд может быть расширен.

### 12.1. ФУНКЦИОНАЛЬНАЯ СХЕМА ПРОЦЕССОРА

В процессоре, построенном по гарвардской архитектуре (рис. 12.1), обработка команд и данных выполняется различными устройствами.

Команды хранятся в ПЗУ команд и представляют собой двоичные коды, которые после декодирования указывают процессору, что он должен делать.

Модуль управления переходами формирует адрес следующей команды  $ak$ , которую процессор должен выполнять. Для хранения этого адреса в модуле управления содержится специальный регистр, называемый программным счетчиком (на схеме рис. 12.1 не показан). Если программа не содержит команд переходов, то новый адрес получается из предыдущего адреса прибавлением единицы. При выполнении команды перехода, когда программа должна перейти на адрес, указанный в команде, схема управления формирует в программном счетчике адрес следующей команды. Адрес выполняемой команды по шине  $ak$  поступает на адресные входы ПЗУ. Выполняемая команда выдается с выхода ПЗУ на шину команд  $k$ .

Команда содержит всю информацию, необходимую для ее выполнения. Команда имеет определенный формат и в отдельных полях, в соответствии с методом адресации, содержит различную информацию: код операции, адреса регистров, адреса переходов или данные.

Заданная в команде операция выполняется в АЛУ. Код операции подается в АЛУ из команды, а данные поступают из блока РОН по шинам  $x$  и  $y$ . Тип выполняемой операции и адреса регистров определяют определенные разряды команды, подключенные к блоку РОН и АЛУ.

К выходу АЛУ подключена шина данных  $d\_bus$ , на которую выдается результат операции. Шина данных  $d\_bus$  подключена параллельно к входам всех устройств, которые предназначены

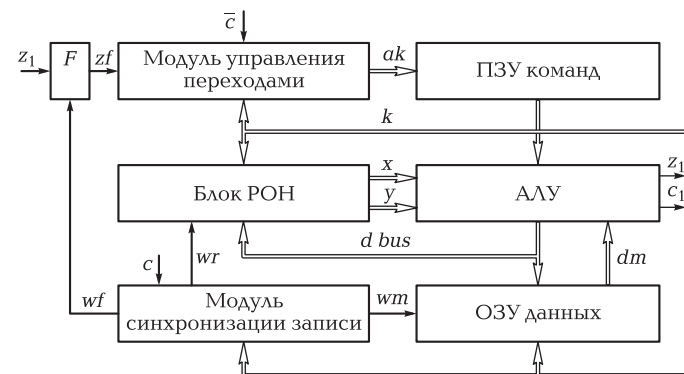


Рис. 12.1. Функциональная схема процессора



для сохранения результата операции — это регистр блока РОН, или ОЗУ.

**Внимание!** К шине данных могут быть дополнительно подключены другие многочисленные периферийные устройства.

Модуль синхронизации записи выдает синхроимпульс записи только на то устройство, которое указано в команде. Поэтому запись результата будет выполнена только в одно устройство, на которое модуль синхронизации записи выдаст синхроимпульс записи в соответствии с командой. Запись в регистр блока РОН синхронизирует импульс *wr* (*Write Register*), запись в ОЗУ — импульс *wm* (*Write Memory*), а запись в регистр признаков — *wf* (*Write Flags*).

Работу над проектом в САПР целесообразно начать с разработки и тестирования устройств обработки команд, а впоследствии подключать модули, выполняющие обработку данных, и проводить их тестирование.

Работа процессора сводится к циклическому выполнению определенных операций в определенные моменты времени, которые определяют импульсы синхронизации. Для выполнения команды за один такт в процессоре используется двухфазная синхронизация. Один такт синхросигнала содержит два состояния (два интервала времени), которым соответствуют значения синхросигнала  $c = 0$  и  $c = 1$ , и два изменения состояний: спад и фронт импульса (рис. 12.2).

Выполнение команды за один такт синхроимпульса становится возможным, если выполнять операции записи синхронно с фронтом и спадом импульсов, а все остальные операции выполнять асинхронно. Выполнение отдельных операций, составляющих команду, должно происходить в определенные моменты времени.

1. Спад сигнала *c*. Модуль управления (см. рис. 12.1) выполняет запись адреса новой команды в программный счетчик, который входит в состав модуля. Этот момент времени является началом цикла выполнения команды. С выхода ПЗУ команд на шину команд *k* выдается код выполняемой команды, который будет неизменным в течение всего периода синхросигнала.
2. Интервал  $c = 0$ . Устройства обработки данных выполняют операцию, предусмотренную в команде. На входы АЛУ и блока РОН в данном интервале будут поступать все необходимые данные для выполнения операции. На выходе АЛУ и на шине *d\_bus* будет сформирован результат выполнения команды. Обработку данных выполняют устройства комбинационного типа (АЛУ, мультиплексоры, дешифраторы), которые не связаны с импульсами синхронизации. Они работают асинхронно.

Формирование результата будет сопровождаться задержками сигналов в логических элементах.

3. Фронт сигнала *c*. Модуль синхронизации записи формирует импульс для записи данных, полученных на шине *d\_bus* при выполнении текущей команды, в одно из устройств памяти — РОН, ОЗУ, регистр признаков.

4. Интервал  $c = 1$ . Формирование адреса следующей команды в модуле управления с учетом возможных переходов.

Разработанная система синхронизации позволяет выполнить одну команду процессора за один такт импульсов синхронизации.

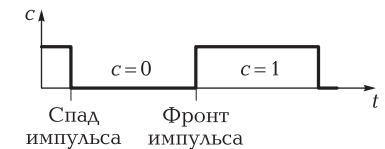


Рис. 12.2. Импульсы синхронизации

## 12.2. РАЗРАБОТКА СИСТЕМЫ КОМАНД

При разработке системы команд сначала выбирается метод адресации и формат команд, а затем кодирование команд определенного типа.

При разработке форматов для упрощения кодирования команд использованы поля, содержащие четыре двоичных разряда (одна тетрада). При этом каждому полю соответствует одна цифра в шестнадцатеричной системе счисления, а код команды содержит четыре цифры.

### 12.2.1. Команды с непосредственной адресацией

Команды с непосредственной адресацией содержат два операнда, один из которых находится в регистре общего назначения, а второй — в команде. После выполнения операции результат записывается в регистр общего назначения, при этом исходное значение первого операнда теряется.

Формат команды с непосредственной адресацией (рис. 12.3) содержит три поля. Первое поле содержит код операции *k1*, второе — адрес (или номер) регистра, в котором находится первый операнд *rx*, а третье поле содержит второй операнд — один байт данных — две шестнадцатеричные цифры *dd*.

Старший бит в коде операции выделен для записи признака команд с непосредственной адресацией. Пусть признаком команд

Код операции	Адрес регистра		Данные
15	12 11	8 7	0
<i>k1</i>	<i>rx</i>	<i>dd</i>	

Рис. 12.3. Формат команд с непосредственной адресацией

с непосредственной адресацией будет нулевое значение старшего бита. При этом максимальное количество команд с непосредственной адресацией равно восьми, а возможные значения кода *k* принадлежат диапазону 0...7.

Символ *rx* обозначает номер РОН, при этом максимальное количество адресуемых РОН — 16. При практической реализации проекта количество РОН будет выбрано небольшим с целью упрощения схемы.

Символы *dd* обозначают две шестнадцатеричные цифры или байт данных, в диапазоне от 00 до FF.

Выполним кодирование конкретных команд. Примем код операции *k1* = 0 для команды пересылки константы в регистр с непосредственной адресацией. Мнемонику данной команды запишем в виде *mvi* (от англ. *move* — пересылать, двигать, *immediate* — непосредственный, безотлагательный).

**Внимание!** Буква *i* в мнемониках команд процессоров фирмы *Intel* обычно обозначает непосредственную адресацию.

Запись команды пересылки с непосредственной адресацией на ассемблере имеет вид: *mvi rx, dd*. Команда выполняет присваивание содержимому регистра *rx* значения байта данных из команды *dd* (две шестнадцатеричные цифры). Команда *mvi r3, 2f*, например, выполнит пересылку числа  $2F_{16}$  в регистр *r3* и будет иметь код  $032F_{16}$ , или  $0000\ 0011\ 0010\ 1111_2$ . После выполнения команды содержимое регистра *r3* будет равно  $2F_{16}$ , независимо от кода, содержащегося в этом регистре до выполнения команды.

Закодируем команду суммирования с непосредственной адресацией. Примем код операции суммирования *k1* = 1. Мнемонику указанной команды можно записать в виде *addi* (от англ. *addition* — сумма, дополнение). Запись на ассемблере команды суммирования с непосредственной адресацией в общем случае имеет вид: *addi rx, dd*. Операцию, выполняемую командой суммирования, можно записать в виде:  $rx = rx + dd$ , что означает *новому содержимому регистра rx присвоить предыдущее значение кода этого регистра, сложенное с константой dd*. Команда *addi r2, 50* с кодом  $1250_{16}$ , прибавит к предыдущему коду регистра *r2* число  $50_{16}$ .

Таблица 12.1

Код	Мнемоника	Операция	Пояснение
0 <i>rx dd</i>	<i>mvi rx, dd</i>	$rx = dd$	Пересылка в регистр константы
1 <i>rx dd</i>	<i>addi rx, dd</i>	$rx = rx + dd$	Суммирование
2 <i>rx dd</i>	<i>subi rx, dd</i>	$rx = rx - dd$	Вычитание
3 <i>rx dd</i>	<i>andi rx, dd</i>	$rx = rx \cdot d$	Логическая операция И
4 <i>rx dd</i>	<i>ori rx, dd</i>	$rx = rx \vee dd$	Логическая операция ИЛИ
5 <i>rx dd</i>	<i>xori rx, dd</i>	$rx = rx \oplus dd$	Логическая операция Исключающее ИЛИ

Подобно суммированию работают команды логических операций, приведенные в табл. 12.1.

## 12.2.2. Команды с регистровой адресацией

В командах с регистровой адресацией операнды находятся в регистрах. При выполнении команд с регистровой адресацией, которые содержат два операнда, одновременно читается содержимое двух регистров по адресам *rx* и *ry* и сразу же выполняется операция, результат которой записывается в регистр *rx* вместо первого операнда. Исходное содержимое первого регистра теряется.

Формат команды содержит четыре поля по четыре разряда, из которых два поля используются для адресации двух регистров *rx* и *ry*, а оставшиеся два поля — для кодирования операции (рис. 12.4). Первое поле *k1*, которое называют *префиксом кода операции*, будет определять тип адресации, а второе поле *k2* будет кодировать конкретную команду.

Для команд с регистровой адресацией выберем код префикса *k1* = 8. Набор и коды команд с регистровой адресацией приведены в табл. 12.2.

Первый код операции		Адрес первого регистра		Адрес второго регистра		Второй код операции		
15		12	11	8	7	4	3	0
<i>k1</i>		<i>rx</i>		<i>ry</i>		<i>k2</i>		

Рис. 12.4. Формат команд с регистровой адресацией

Таблица 12.2			
Код	Мнемоника	Операция	Пояснение
8 <i>rx ry 0</i>	<i>mov rx, ry</i>	$rx = ry$	Пересылка в регистр константы
8 <i>rx ry 1</i>	<i>add rx, ry</i>	$rx = rx + ry$	Суммирование
8 <i>rx ry 2</i>	<i>sub rx, ry</i>	$rx = rx - ry$	Вычитание
8 <i>rx ry 3</i>	<i>and rx, ry</i>	$rx = rx \cdot ry$	Логическая операция И
8 <i>rx ry 4</i>	<i>or rx, ry</i>	$rx = rx \vee ry$	Логическая операция ИЛИ
8 <i>rx ry 5</i>	<i>xor rx, ry</i>	$rx = rx \oplus ry$	Операция Исключающее ИЛИ

Поясним выполнение команд с регистровой адресацией.

Команда *mov rx, ry* выполняет пересылку в регистр *rx* содержимого регистра *ry*. В ассемблерной записи команды после слова *mov* (переслать) на первом месте указан регистр, куда переслать (в *rx*), а на втором месте указан регистр, откуда переслать (из *ry*). Регистр *rx* называется **получателем**, или **приемником, данных**, а регистр *ry* — **источником данных**. Запись операции  $rx = ry$  означает: *присвоить регистру rx содержимое регистра ry*.

Команда *add rx, ry* выполняет суммирование содержимого регистров *rx* и *ry* и записывает сумму в регистр *rx*. В ассемблерной записи команды после слова *add* (суммировать) на первом месте указан регистр-получатель, куда будет записан результат (в *rx*), а на втором месте указан регистр-источник. Команда *add r4, r5*, например, выполнит суммирование содержимого регистров *r4* и *r5* с записью результата в регистр *r4*. В соответствии с табл. 12.2 данная команда имеет код 8451. Первая шестнадцатеричная цифра (8) определяет команды с регистровой адресацией, вторая цифра (4) — номер регистра первого слагаемого и получатель результата, третья цифра (5) — номер регистра второго слагаемого, или источник, последняя цифра (1) — тип выполняемой операции — суммирование.

### 12.2.3. Команды с косвенной регистровой адресацией

Команды с косвенной регистровой адресацией использованы для обращения к памяти. При косвенной регистровой адресации в команде указан регистр, содержащий адрес ячейки памяти, в которой находится операнд (рис. 12.5). Формат данных команд по-

Первый код операции		Регистр операнда	Регистр адреса	Второй код операции	
15	12 11	8 7	4 3	0	
<i>k1</i>		<i>rx</i>	<i>ry</i>	<i>k2</i>	

Рис. 12.5. Формат команд с косвенной регистровой адресацией

добен ранее рассмотренному формату команд с регистровой адресацией.

В простейшем случае необходимо предусмотреть всего две команды обращения к памяти: загрузка в регистр (к себе) операнда из памяти и запись из регистра (от себя) в память (табл. 12.3).

В соответствии с форматом команда содержит четыре поля, кодируемые шестнадцатеричными цифрами. Первая цифра кодирует метод адресации. Выберем для команд с косвенной адресацией код  $k1 = A_{16} = 1010_2$ . Вторая цифра содержит номер регистра *rx*, который содержит операнд. Третья цифра определяет номер регистра *ry*, в котором содержится адрес ячейки памяти, а четвертая — код *k2* (0 или 1), определяющий направление пересылки операнда.

Пересылку в регистр содержимого ячейки памяти называют **загрузкой** регистра (*load*) и обозначают мнемоникой *ld*, а пересылку в ячейку памяти содержимого регистра называют **записью** (*store*) и обозначают *st*. Команда загрузки регистра *rx* содержимым ячейки памяти, адрес которой предварительно был записан в регистр *ry*, имеет вид: *ld rx, (ry)*. Круглые скобки в ассемблерах процессоров фирмы *Intel* указывают, что в регистре содержится адрес. Заметим, в системах команд некоторых микроконтроллеров в подобных случаях используется символ @.

Для команды загрузки регистра из памяти примем  $k2 = 0$ , а для команды записи в память из регистра выберем код  $k2 = 1$  (см. табл. 12.3).

Команда *ld r1, (r2)*, например, будет иметь шестнадцатеричный код A120, или двоичный код 1010 0001 0010 0000. Она загрузит в ре-

Таблица 12.3		
Код	Мнемоника	Пояснение
<i>a rx ry 0</i>	<i>ld rx, (ry)</i>	Загрузка в регистр <i>rx</i> операнда из ячейки памяти, адрес которой содержится в <i>ry</i>
<i>a rx ry 1</i>	<i>st (ry), rx</i>	Запись в ячейку памяти, адрес которой содержится в <i>ry</i> , операнда из регистра <i>rx</i>

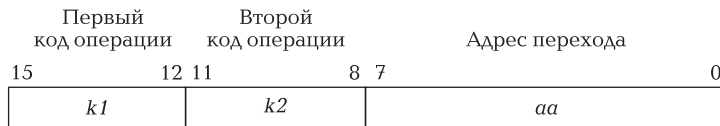


Рис. 12.6. Формат команд перехода с прямой адресацией

гистр *r1* содержимое ячейки памяти, адрес которой заблаговременно был записан в регистр *r2*.

Команда *st (r0), r1* будет иметь шестнадцатеричный код A101, или двоичный код 1010 0001 0000 0001. Она запишет в ячейку памяти, адрес которой заблаговременно был записан в регистр *r0* содержимое регистра *r1*.

12.2.4. Команды с прямой адресацией

При прямой адресации в команде содержится адрес. Прямая адресация используется для команд переходов, в этом случае необходимо выделить два 4-разрядных поля кодов операции (*k1*, *k2*) и 8-разрядный адрес (*aa*). Формат команд переходов приведен на рис. 12.6.

Минимальный набор команд переходов приведен в табл. 12.4.

12.2.5. Расширение набора команд

Разработанный небольшой набор команд предназначен для изучения работы и методики проектирования синтезированного процессора. Система команд является открытой, допускающей изменения и дополнения, содержит большое количество неиспользованных кодовых комбинаций.

Для дополнительных команд могут быть использованы префиксы (коды *k1*), принимающие в шестнадцатеричной системе значения 9, B, C, D, E. Система может быть дополнена командами, со-

Таблица 12.4

Код	Мнемоника	Пояснение
<i>F 0 a a</i>	<i>jmp aa</i>	Безусловный переход к адресу <i>aa</i>
<i>F 1 a a</i>	<i>jz aa</i>	Условный переход к адресу <i>aa</i> , если <i>fz</i> = 1

держащими один операнд (например, сдвиги), командами обращения к памяти и к регистрам с прямой адресацией. Дополнительные команды потребуются включить в описание АЛУ.

12.3. ПРОЕКТ ПРОЦЕССОРА В САПР

При разработке процессора в САПР целесообразно представлять модули верхних уровней иерархии в форме схем, наглядно отображающих структуру и связи, а для модулей нижнего уровня иерархии, менее сложных — использовать описания на языке *Verilog*.

Сначала целесообразно разработать и отладить устройства обработки команд, что позволит впоследствии подключать другие модули и проводить тестирование проекта, постепенно наращивая его сложность.

12.3.1. Модуль управления control

Схему управления переходами и программный счетчик, показанные на функциональной схеме (см. рис. 12.1), можно описать как единое устройство, формирующее адрес следующей команды — модуль управления. Для поведенческого описания модуля управления необходимо описать входные и выходные сигналы, а также сформулировать алгоритм функционирования (пример 12.1).

```
// Пример 12.1 Модуль
// управления control
module control (c, k, fz, ak);                                //1
input c, fz; input [15:0] k;                                  //2
output [7:0] ak; reg [7:0] ak;                                //3
always @ (negedge c)                                          //4
if ((k[15:8] == 8'b 11110000) |                                //5
    (k[15:8] == 8'b 11110001) & fz) //6
    ak = k[7:0];                                               //7
else ak = ak + 1;                                              //8
endmodule                                                       //9
```

В соответствии с функциональной схемой входные сигналы модуля — это синхросигнал *c*, команда (16-разрядный вектор) *k* и признак нуля *zf*, полученный в предыдущей команде. Выходным сигналом является адрес следующей команды (8-разрядный вектор) *ak*. Все указанные сигналы перечислены в заголовке и описаны в строках 1 ... 3. Для сигнала *ak* указан тип *reg*. Этот сигнал будет являться выходным сигналом программного счетчика.

Выходной сигнал, как было рассмотрено ранее, должен выдаваться по спаду импульса синхронизации. Это требование обеспечивает запись в строке 4.

Модуль *control* должен функционировать в соответствии с выбранными командами пересылки. Если в текущей команде выполняется условие пересылки, то адресом следующей команды является младший байт текущей команды, который является адресом пересылки.

Система команд содержит две команды переходов: безусловный переход, для которого старший байт команды (номера разрядов от 15 до 8) содержит код  $F0_{16} = 11110000_2$ , и условный переход с кодом операции  $F1_{16} = 11110001_2$ , который выполняется при  $z = 1$ . Эти команды учитывают условие оператора *if*. Выражение, записанное в строке 5, равно 1, если текущая команда — безусловный переход, а выражение в строке 6 равно 1 для команды условного перехода и при выполнении условия  $z = 1$ .

Если выполняется условный или безусловный переход, то в программный счетчик загружается адрес перехода из команды (строка 7). При отсутствии переходов адрес программного счетчика инкрементируется (строка 8).

Новое значение адреса будет записано в программный счетчик по спаду, и команда, хранящаяся в ПЗУ по этому адресу, будет присутствовать на шине команд  $k$  в следующем цикле.

### 12.3.2. ПЗУ команд

В разрабатываемом процессоре используются встроенные блоки памяти большой информационной емкости имеющиеся в ПЛИС *FLEX10K* и *ACEX 1K* фирмы *Altera*, поэтому для данного проекта необходимо выполнить назначение типа ПЛИС, выбирая из меню *Assign/Device*, и в открывшемся окне указать семейство *FLEX10K* и тип устройства *AUTO*.

Модуль памяти *LPM\_ROM* называют параметризуемым. Это означает, что параметры модуля можно изменять и настраивать модуль для определенного режима работы.

После ввода символа открывается окно редактирования портов и параметров *Edit Port/Parameters*, в разделе *Ports* которого предусмотрена возможность выбора портов (сигналов), которые будут использоваться в устройстве. Здесь же задаются наличие инверсии выбранных сигналов и системы счисления для отображения значений параметров (рис. 12.7).

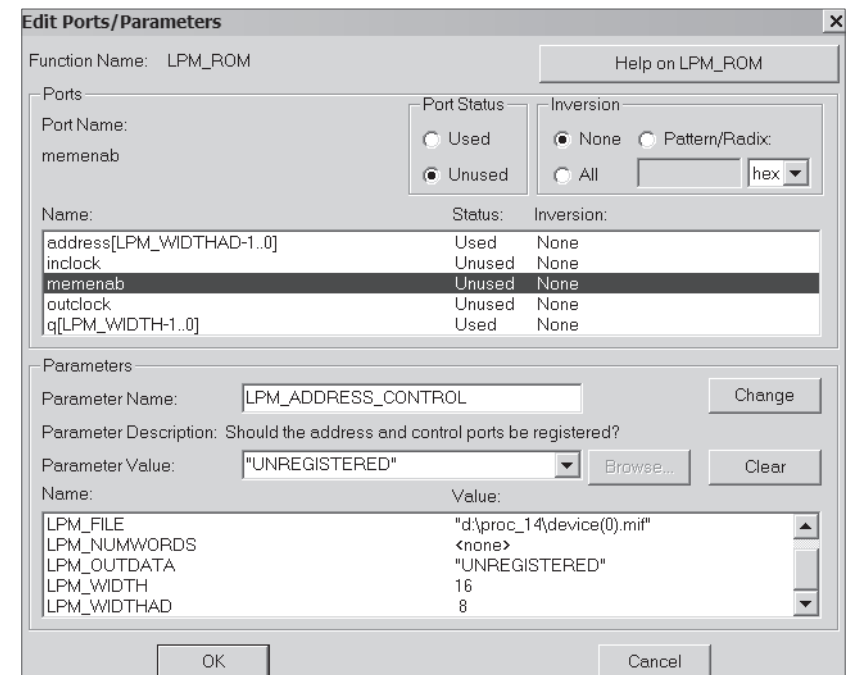


Рис. 12.7. Окно редактирования портов и параметров

Для разрабатываемого проекта необходимо, чтобы ПЗУ команд работало асинхронно и данные на выходе появлялись непосредственно после подачи адреса без промежуточной записи в регистр, поэтому входные и выходные сигналы синхронизации *inclock*, *outclock*, а также сигнал разрешения (и запрещения) работы памяти *memenab* указаны как неиспользуемые (*unused*).

В разделе параметров указаны разрядность данных *LPM\_WIDTH* — 16, разрядность адреса *LPM\_WIDTHHAD* — 8, а также отсутствие синхронизации адреса и данных *UNREGISTERED*. В этом разделе впоследствии, при моделировании, указывается путь к файлу содержимого памяти (с расширением *.mif*).

### 12.3.3. Блок РОН

В разрабатываемом процессоре используется блок регистров общего назначения (блок РОН), имеющий режим одновременного чтения содержимого двух регистров. Такие устройства памяти на-



зывают **двухадресными**, или **двухпортовыми**. Для записи используют один из этих адресов.

В соответствии с разработанными форматами команд процессор может содержать 16 РОН.

Для предварительного исследования работы процессора количество регистров уменьшено до 4.

Модули блока РОН разработаны в виде описаний на языке Verilog.

**Регистр *reg\_8\_bit*.** В блоке РОН используются 8-разрядные регистры с динамическим управлением, описанные в примере 12.2. Входной 8-разрядный код *d* записывается по фронту сигнала *c*, и формируется выходной код *q*. Работу регистра описывает оператор последовательного присваивания с ключевым словом *always*, символом @ и условием срабатывания в круглых скобках — по фронту сигнала — *posedge*. Выходной сигнал, формируемый оператором последовательного присваивания, указан как регистр *reg*.

```
//Пример 12.2. Модуль
// регистра reg_8_bit
module reg_8_bit (c, d, q); //1
input c; //2
input [7:0] d; //3
output [7:0] q; //4
reg [7:0] q; //5
always @ (posedge c) q = d; //6
endmodule
```

**Демultipлексор *d\_mux*** — выполняет управление записью данных (пример 12.3). Он подключает сигнал синхронизации с к одному из восьми выходов в соответствии с 3-разрядным адресом *ax*. Выходные сигналы демultipлексора, обозначенные буквами (*ca*, *cb*, *cc*, *cd*), будут подключены к входам синхронизации регистров.

```
// Пример 12.3. Модуль
// демultipлексора d_mux
module d_mux (ax, c, ca, cb, cc, cd);
input [1:0] ax;
input c;
output ca, cb, cc, cd;
assign ca = c & ~ax[1] & ~ax[0];
assign cb = c & ~ax[1] & ax[0];
assign cc = c & ax[1] & ~ax[0];
assign cd = c & ax[1] & ax[0];
endmodule
```

Описание 8-разрядного демultipлексора выполнено с использованием параллельных операторов с ключевым словом *assign*,

поэтому устройство будет работать как асинхронная комбинационная схема. Для каждого выходного сигнала записан отдельный оператор присваивания в соответствии с логикой функционирования демultipлексора. Так, например, *ca* = 1 при *c* = 1 и *ax* = 00<sub>2</sub>.

**Мультимплексоры шин *mux\_bus*** (пример 12.4) — позволяют выполнить асинхронное чтение двух регистров одновременно по адресам *ax* и *ay* с выдачей результата на шины *x* и *y* соответственно. содержимое двух регистров *rx* и *ry* в соответствии с 3-разрядными адресами *ax* и *ay*. Выходной код первого регистра обозначен *qa*, следующего — *qb*, а последнего — *qh*. К выходам регистров подключены два мультимплексора, один для формирования сигнала *x*, а другой — для сигнала *y*.

```
//Пример 12.4. Модуль
//мультимплексоров шин mux_bus
module mux_bus (a, da, db, dc, dd, q);
input [1:0] a;
input [7:0] da, db, dc, dd;
output [7:0] q;
```

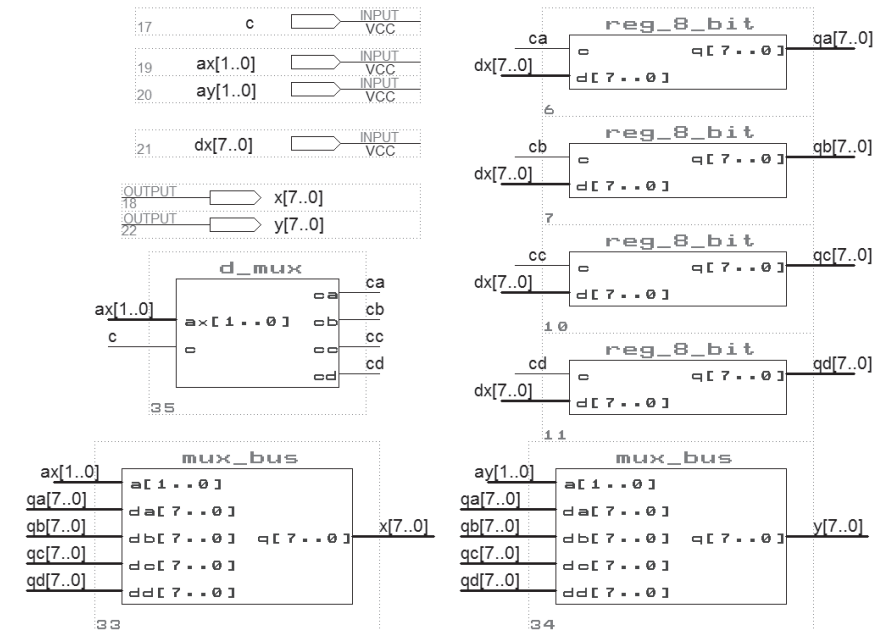


Рис. 12.8. Схема блока РОН в графическом редакторе

```

assign q = (a == 0) ? da:
(a == 1) ? db:
(a == 2) ? dc:dd;
endmodule

```

Для описания мультиплексора использованы вложенные операторы условного присваивания, выполняющие выбор варианта. В первом операторе после двоеточия вместо ветви «Нет» записано условие и следующий условный оператор и т.д.

Для модулей регистра, дешифратора и мультиплексора необходимо по приведенным описаниям создать проекты, выполнить компиляцию и создать символы, из которых строится схема блока РОН (рис. 12.8) с именем *block\_ron*.

### 12.3.4. Модуль АЛУ

Модуль АЛУ выполняет арифметические и логические команды обработки данных с непосредственной и регистровой адресацией. Кроме того, через данный модуль выполняются команды обращения к ОЗУ.

Входными сигналами модуля АЛУ (пример 12.5) в соответствии с функциональной схемой процессора (см. рис. 12.1) являются два 8-разрядных операнда, используемые при регистровой адресации *x* и *y*, 16-разрядная команда *k*, а также данные с выхода ОЗУ *dm*.

```

//Пример 12.5. Модуль АЛУ
module alu (x, y, k, q, fz);
input [7:0] x, y;
input [15:0] k;
output [7:0] q; reg [7:0] q;
output fz;
always
if (k [15] == 0) case (k [14:12])
0: q = k [7:0]; 1: q = x + k [7:0];
2: q = x & k [7:0]; 3: q = x | k [7:0];
4: q = x ^ k [7:0]; endcase
else if (k [15:12] == 8) case (k [2:0])
0: q = y; 1: q = x + y; 2: q = x & y;
3: q = x | y; 4: q = x ^ y; endcase
else if (k [15:12] == 10)
case (k [0]) 0: q = dm; 1: q = x; endcase
else q = 0;
assign fz = (q == 0);
endmodule

```

Выходные сигналы — 8-разрядный результат выполнения операции, поступающий на шину данных *d\_bus* и признак (флаг) нуля *fz*.

Модуль АЛУ должен быть выполнен в виде комбинационной схемы, которая формирует результат в соответствии с текущими значениями входных сигналов. Для описания подобных схем предназначены операторы параллельного типа *assign*, но можно использовать и последовательные операторы *always*, в которых список чувствительности отсутствует. Второй вариант имеет более широкие функциональные возможности, он позволяет использовать операторы *case* и *if*.

Если последовательный оператор содержит только слово *always*, а символ @ и список чувствительности отсутствует, то он будет срабатывать после изменения любого из входных сигналов подобно параллельному оператору. Выходной сигнал, формируемый данным оператором, необходимо описать как *reg*. Данный вариант будет использован для описания АЛУ, где необходимы операторы *case* и *if*, которые с параллельным присваиванием не работают.

В описании модуля АЛУ *alu* сначала, посредством оператора *if*, определяется тип адресации выполняемой команды, а затем, оператором *case*, операция, заданная в команде. Описание содержит три части, начинающиеся с оператора *if*, в которых описаны команды с непосредственной и команды с регистровой адресацией.

Первый оператор *if* выполняет условный переход к выполнению команд с непосредственной адресацией. Условием перехода на ветвь «Да» для этого оператора является признак команд с непосредственной адресацией — нулевое значение старшего бита команды. Вторую ступень дешифрации выполняет оператор варианта *case*, селектор которого — биты команды с 14 по 12, представлены в описании как 3-разрядный вектор, который может принимать значения от 0 до 7. Значения селектора в операторах *case* записаны в десятичной системе с целью упрощения восприятия описания.

При нулевом значении селектора выполняется команда загрузки регистра данными из команды. Для этого байт данных из команды передается на шину *d\_bus*, которая подключена к входам РОН. При этом на адресный вход блока РОН из команды будет подаваться адрес загружаемого регистра, и по фронту синхросигнала произойдет загрузка.

Второй оператор *if* описывает команды с регистровой адресацией, при условии, что префикс КОП\_1, содержащийся в разрядах команды с 15 по 12, равен 8. Выполняемую команду определяет вторая часть кода операции КОП\_2, расположенная в разрядах 2 — 0, используемая как селектор варианта во вложенном операторе *case*.

Третий оператор *if* описывает выполнение команд обращения к памяти.

### 12.3.5. Устройство синхронизации записи данных (*sync\_wr*)

Результат операции, выполненной в АЛУ, выдается на шину выходных данных *d\_bus*, к которой подключаются различные устройства памяти, принимающие данные: РОН, ОЗУ, порт вывода, другие периферийные устройства. Однако запись результата должна быть выполнена только в определенное устройство в соответствии с кодом команды.

Управление записью осуществляется подачей сигнала, синхронизирующего запись только на то устройство, для которого эти данные предназначены в текущей команде.

В этом случае все получатели подключены к источнику сигнала постоянно, и результат операции поступает параллельно на все входы. Однако запись результата происходит только в выбранное устройство.

Устройство синхронизации записи (пример 12.6) содержит подобно дешифратору набор логических схем И, на входы которых поступают импульсы синхронизации и кодовые комбинации, соответствующие командам, для которых должна выполняться запись. Выходами комбинационных схем являются сигнал записи результата в РОН (*wr\_ron*) и сигнал записи в память (*we\_mem*).

```
// Пример 12. 6. Модуль
// синхронизации записи
// данных sync_wr
module sync_wr (c, k, w_r, w_m);
input c; input [15:12] k;
output we_ron, we_mem;
assign we_ron = c & (~k[15] |
(~k[14]) & (~k[13]));
assign we_mem = c & (k[15] &
(~k[14]) & (k[13]) & (~k[12]));
endmodule
```

### 12.3.6. Разработка схемы процессора

Разработанные описания позволяют после удачной компиляции создать символы модулей и построить из них принципиальную схему процессора. Соединение модулей между собой выполняется

в соответствии с функциональной схемой (см. рис. 12.1). При подключениях сигналов необходимо учитывать форматы команд. Принципиальная схема процессора в графическом редакторе показана на рис. 12.9.

Модуль *control* выдает на ПЗУ команд *LPM\_ROM* адрес новой команды по спаду синхросигнала с учетом анализа предыдущей команды и признака результата *z*.

Операции обработки данных выполняются в АЛУ совместно с блоком РОН. В разработанной схеме блок РОН содержит четыре регистра, для адресации которых необходимы два разряда. В соответствии с принятыми форматами команд в качестве адреса *ax* используются разряды команды 9 и 8. Это младшие разряды поля *ix* (см. рис. 12.3, 12.4). В качестве адреса *au* используются разряды команды 5 и 4 — младшие разряды поля *yu*.

Результат выполнения всех команд формируется на выходе АЛУ и подается по шине данных *d\_bus* на блок РОН и модуль ОЗУ *LPM\_RAM\_DQ*. Выбор устройства, в которое выполняется запись данных, происходит по фронту синхросигнала с осуществляет модуль синхронизации записи *dc\_we*, на выходе которого формиру-

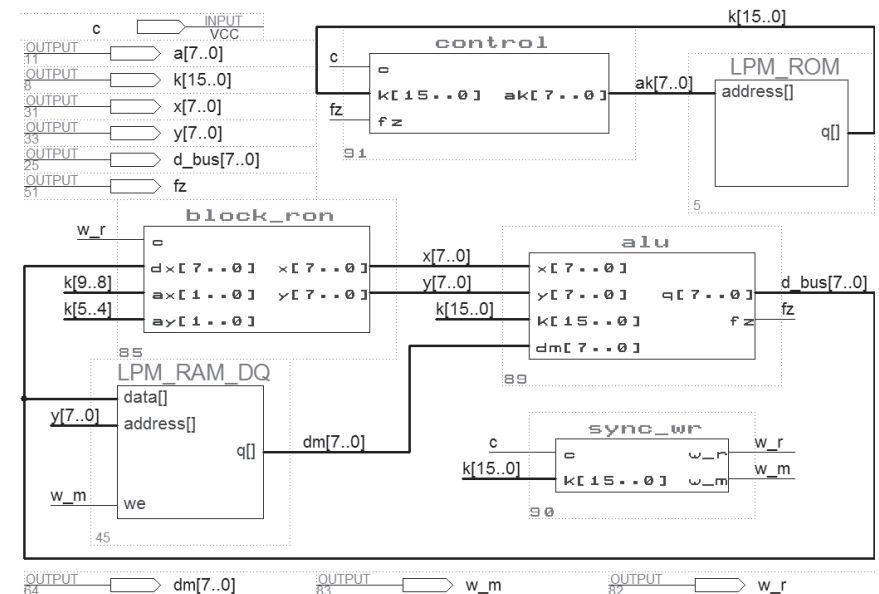


Рис. 12.9. Схема процессора в графическом редакторе

ются сигналы разрешения записи в регистр  $w_r$ , или в память  $w_m$  в соответствии с выполняемой командой.

12.4.

РАЗРАБОТКА ПРОГРАММ ДЛЯ  
СИНТЕЗИРОВАННОГО ПРОЦЕССОРА

**Программа 1.** Началом программирования является формулировка задачи и составление алгоритма в виде словесного описания, таблицы или блок-схемы. В соответствии с алгоритмом составляется ассемблерная программа, использующая мнемоники команд, что целесообразно для удобства восприятия. Затем записывается программа на машинном языке в виде последовательности двоичных кодов команд, предназначенных для записи в ПЗУ команд процессора. Результат разработки удобно представить в виде таблицы.

Для анализа работы процессора при выполнении команд с непосредственной и регистровой адресацией составим программу, содержащую команды указанных типов. Выполним пересылку констант в регистры процессора, а затем операцию суммирования.

В регистр  $r0$  перешлем константу 01, в регистр  $r1$  — константу 12, в регистр  $r2$  — константу 23, в регистр  $r3$  — константу 34. Затем выполним суммирование всех чисел с накоплением суммы в регистре  $r0$ . Разработанная программа представлена в табл. 12.5. В столбце  $q\_alu$  приведены теоретические значения суммы в шестнадцатеричной системе счисления выдаваемые на шину данных.

При моделировании необходимо выполнить инициализацию модуля  $LPM\_ROM$ , записать программу в ПЗУ команд. Для инициализации памяти после вызова симулятора, следует, не нажимая в окне симулятора кнопку *Start*, вызвать из меню *Initialize/Memory*

Таблица 12.5			
Адрес	Программа	Код	Шина $q\_alu$
00	$mvi\ r0,01$	0001	23
01	$mvi\ r1, 12$	0112	34
02	$mvi\ r2,23$	0223	45
03	$mvi\ r3,34$	0334	56
04	$add\ r0, r1$	8011	13
05	$add\ r0, r2$	8021	36
06	$add\ r0, r3$	8031	6A

Initialize Memory							
Memory Name:		LPM_ROM:5[altrom:srom]content					
Address:		Value:					
00	0001	0112	0223	0334	8011	8021	8031
07	0000	0000	0000	0000	0000	0000	0000
0E	0000	0000	0000	0000	0000	0000	0000

Рис. 12.10. Окно инициализации памяти

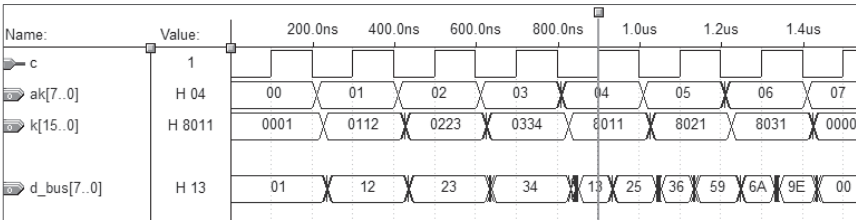


Рис. 12.11. Работа команд с непосредственной и регистровой адресацией

*Initialize*. В результате откроется окно, в котором представлены адреса (*Address*) и содержимое (*Value*) ячеек памяти (рис. 12.10).

Для изменения содержимого ячейки следует выделить ее, ввести новое значение, нажать клавишу **Enter**. Автоматически будет выделена следующая ячейка. Введенный массив команд необходимо записать кнопкой *Export File*. Будет создан текстовый файл с именем проекта и расширением *.mif*. Путь к нему можно указать в строке *LPM File* окна параметров модуля памяти для того, чтобы при повторных запусках моделирования происходила инициализации памяти с использованием содержимого указанного файла.

Результат моделирования (рис. 12.11) позволяет рассмотреть и подробно описать работу команд с непосредственной и регистровой адресацией.

Спад синхроимпульса определяет начало машинного цикла, сначала устанавливается адрес, а затем выдается команда на шину адреса команд  $ak$ .

В момент времени  $t = 0$  устанавливается адрес  $a = 00$ , по которому записан код команды 0001. Первая цифра кода (0) определяет операцию пересылки с непосредственной адресацией. Вторая цифра кода команды (0) является адресом регистра  $rx$ , а две последние цифры (в шестнадцатеричной системе) определяют пересылаемые данные, или константу (01). Код данных из младшего байта коман-

Таблица 12.6

Адрес	Программа	Код	Шина $q\_alu$
00	<i>mvi r0, 01</i>	0001	01
01	<i>mvi r1, 012</i>	0101	01
02	<i>add r0, r1</i>	8011	02
03	<i>add r1, r0</i>	8101	03
04	<i>jmp 02</i>	f002	0A

ды передается на выход АЛУ (на шину данных  $d\_bus$ ) и по фронту синхроимпульса будет записан в регистр  $rx$ .

На диаграммах отображаются помехи, сопровождающие изменения команд и данных. Помехи возникают после спада синхроимпульса. Однако запись данных будет выполнена по фронту синхроимпульса в момент времени, когда данные установились и являются корректными.

По адресу 04 записана команда суммирования, результат выполнения которой будет записан в регистр  $r0$  по фронту синхроимпульса в момент времени, отмеченный на диаграммах маркером. Записываемый код (13) соответствует теоретическому значению. После записи в регистр  $r0$  нового значения на выходе АЛУ появляется результат суммирования новой суммы и второго слагаемого (25), который никуда не записывается и ошибок в работе процессора не вызывает. Подобные данные называют «мусор».

**Программа 2.** Рассмотрим еще пример. Задано разработать программу, которая формирует на выходе АЛУ массив, содержащий числа Фибоначчи.

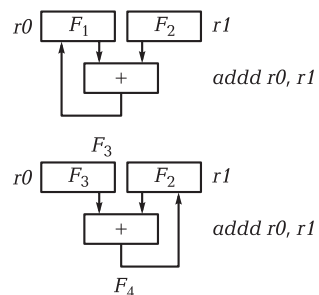


Рис. 12.12. Пояснение вычислений чисел Фибоначчи

Последовательность чисел Фибоначчи формируется следующим образом. Первые два числа равны 1. Каждое последующее число равно сумме двух предыдущих чисел. Данная последовательность имеет вид: 1, 1, 2, 3, 5, 8, 13...

Началом программы является *инициализация* — это запись первого и второго чисел последовательности, равных единице, в регистры  $r0$  и  $r1$  (табл. 12.6).

Для вычисления третьего числа Фибоначчи суммируются первое и второе числа  $F_3 = F_1 + F_2$ , а для вычисления чет-

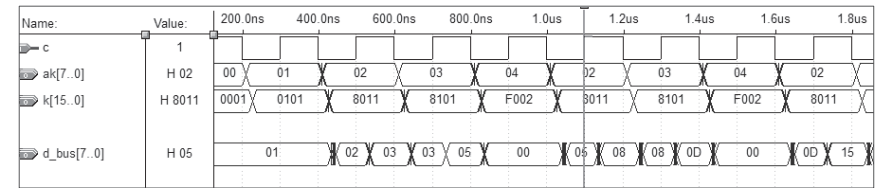


Рис. 12.13. Работа программы

вертого — второе и третье  $F_4 = F_2 + F_3$ . Вычисление следующих чисел выполняется в цикле. Тело цикла содержит вычисление двух чисел последовательно с получением результата попеременно в регистрах  $r0$  или  $r1$  (рис. 12.12).

Результат суммирования записывается на место слагаемого, которое не потребуется для дальнейших вычислений. Работу программы (рис. 12.13) иллюстрируют временные диаграммы.

Результат вычислений может быть записан в ОЗУ посредством дополнительных команд. Момент записи, соответствующий истинному результату, определяют фронты синхроимпульсов. Один из таких моментов отмечен на рис. 12.13. Количество выполненных циклов будет определяться заданной длительностью процесса моделирования (параметр *End Time*).

Разработанный синтезированный процессор может быть модифицирован применительно к решению конкретных задач, что обеспечит высокую производительность вычислительной системы.

В этом случае проект должен начинаться с анализа технического задания, формирования алгоритма, выбора системы команд и заканчиваться описанием эскизного проекта в САПР, с результатами моделирования отдельных блоков и устройств процессора.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните особенности Гарвардской архитектуры процессора.
2. Какие достоинства имеет процессор с блоком РОН?
3. Опишите порядок выполнения команд в процессоре.
4. Поясните особенности способов адресации.
5. Обоснуйте выбор форматов команд с непосредственной и регистровой адресацией.
6. Перечислите функциональные узлы, составляющие блок РОН.
7. Поясните назначение сигналов по схеме процессора.
8. Поясните функции модуля управления.



## УСЛОВИЯ ЭКСПЛУАТАЦИИ ЦИФРОВЫХ УСТРОЙСТВ

На условия эксплуатации любой радиоэлектронной аппаратуры (РЭА), в том числе и цифровых устройств, влияют внешние воздействующие факторы, которые подразделяются на климатические, механические и радиационные.

К **климатическим факторам** относятся: температура и влажность окружающей среды; тепловой удар; атмосферное давление; пыль или песок; солнечное облучение; наличие в атмосфере агрессивных веществ и озона, грибковых образований (плесень), микроорганизмов, насекомых, грызунов, взрывоопасных и легковоспламеняющихся веществ, дождя и брызг.

К **механическим факторам** относятся: воздействие вибраций, ударов, линейного ускорения, акустического удара.

К **радиационным факторам** относятся все виды космической, естественной и искусственной радиации.

Существенное влияние на работоспособность РЭА оказывает человеческий фактор. Квалификация специалиста сказывается на качестве работы РЭА на всех этапах ее жизненного цикла. При проектировании необходимо вводить «защиту от дурака».

*«Главное зло условий эксплуатации, выводящее из строя любую аппаратуру надежно и надолго — дурак. Но в технических регламентах этот фактор пока отсутствует, а потому и защита от дураков к числу обязательных требований не относится»* — Игорь Бреднев, уральский геофизик, XX в.

Внедрение автоматизации на всех этапах создания РЭА уменьшает влияние человеческого фактора.

### 13.1. КЛИМАТИЧЕСКИЕ ФАКТОРЫ

**Нормальными** климатическими условиями являются: температура  $+25 \pm 10^\circ\text{C}$ , относительная влажность 45...80 %, атмосферное давление 83...106 кПа (630...800 мм рт. ст.), отсутствие активных веществ в окружающей атмосфере.

Весь земной шар разделен на семь климатических зон:

- **очень холодный регион** (Антарктида) — со средней минимальной температурой ниже  $-60^\circ\text{C}$  и с сильным ветром;
- **холодная зона** (большая часть России и Канады, Аляска, Гренландия) — со средней минимальной температурой до  $-50^\circ\text{C}$ . Годовой перепад температур составляет до  $80^\circ\text{C}$ , среднесуточный — до  $40^\circ\text{C}$ . Атмосфера благоприятна для ионизации воздуха и, как следствие, для накопления статического электричества. Характерны обледенение, иней, ветер с мелкой снежной пылью;
- **умеренный климатический регион** (часть территории России, большая часть Европы, США, прибрежные территории Австралии, Южной Африки и Южной Америки) — годовое изменение температуры от  $-5$  до  $+35^\circ\text{C}$ , образование инея, выпадение росы, наличие тумана, изменение давления воздуха от 86 до 106 кПа;
- **влажная тропическая зона** — располагается вблизи экватора и включает в себя большую часть Центральной и Южной Америки, среднюю часть Африки, юг Индии, Индонезию, часть Юго-Восточной Азии. Среднегодовые температуры  $+20...+25^\circ\text{C}$  с перепадом за сутки не более  $10^\circ\text{C}$ . Высокая влажность и повышенная концентрация солей (особенно вблизи побережья морей и океанов) делает атмосферу коррозионно-агрессивной. Благоприятное сочетание температуры и влажности способствует существованию плесневых грибов;
- **зона с сухим тропическим климатом** (северная часть Африки, центральная Австралия, засушливые районы Средней Азии, Аравийский полуостров, часть Северной Америки) — характеризуется высокими температурами (до  $+55^\circ\text{C}$ ), низкой влажностью, интенсивным солнечным излучением (до  $1500 \text{ Вт/м}^2$ ), высоким содержанием пыли и песка в атмосфере. Последнее способствует отрицательному абразивному и химическому воздействию на аппаратуру;
- **умеренно холодная морская зона** — включает в себя моря, океаны и прибрежные территории, расположенные севернее  $30^\circ$  северной широты и южнее  $30^\circ$  южной широты;
- **тропическая морская зона** — остальная часть морей, океанов и прибрежных территорий.

Климат морских зон отличается сравнительно небольшими суточными перепадами температур, наличием высокой влажности и значительной концентрацией хлоридов в атмосфере.

**Температурные условия** влияют на место установки аппаратуры: необходимо, чтобы температура нагрева элементов находилась в допустимых для них пределах. Кроме того, многие конструктивные материалы при высоких температурах претерпевают структурные изменения (тепловое старение материалов). Известно, что при нагревании или охлаждении изменяются линейные размеры изделий. В результате в конструкциях возникают дополнительные напряжения, вызывающие деформацию деталей, ослабление крепления деталей, появление дополнительных погрешностей из-за увеличения зазоров, разрушение паяных швов, нарушение герметизации. Защитить полностью РЭА от изменения температуры не представляется возможным, поэтому выбор материалов и конструкции устройств должен производиться с учетом всех последствий влияния температуры.

**Тепловой удар** характеризуется резким изменением температуры окружающей среды. Наиболее сильно тепловой удар проявляется в элементах конструкции, где имеются локальные механические напряжения, способствуя образованию микротрещин, их росту и объединению. Для охлаждения аппаратуры предусматривают теплоизоляцию от внешних источников тепла и отвод тепловой энергии от элементов, выделяющих собственное тепло во время работы. Это кулеры и теплоотводящие радиаторы.

**Влажность** — один из наиболее агрессивных факторов, способствующий коррозии металлических деталей, старению неметаллов, изменению электроизоляционных характеристик изоляторов. Вода, содержащаяся в атмосфере, всегда загрязнена активными веществами (углекислыми и сернистыми солями кальция, магния, железа, хлористым кальцием, газами), что еще больше способствует проявлению коррозии.

Разрушение металлов и сплавов из-за химического и электрохимического воздействия внешней среды называется **коррозией**. Для защиты от коррозии необходимо применять металлы и сплавы, которые не подвергаются коррозии (нержавеющие стали, титановый сплав и др.) и наносить защитные металлические, неметаллические и лакокрасочные покрытия.

**Давление окружающей среды** зависит, прежде всего, от высоты над уровнем моря места, где эксплуатируется аппаратура. Содержание влаги в атмосфере с ростом высоты уменьшается. При снижении давления ухудшается отвод тепла конвективным теплообме-

ном, уменьшается электрическая прочность воздуха, что приводит к ионизации воздуха и образованию химически активных ионов и радикалов. Повышенное атмосферное давление оказывает в первую очередь механическое воздействие на элементы конструкции РЭА. В качестве защиты применяют специальные покрытия и отвод тепла.

**Пыль и песок**, содержащиеся в атмосфере, оседая на поверхности деталей, могут стать причиной возникновения неисправностей. Пыль содержит углекислые и сернокислые соли и хлориды, которые, взаимодействуя с влагой, ускоряют процессы коррозии. Кроме того, находящаяся в воздухе пыль способствует утечке зарядов и может вызвать пробой промежутка, находящегося между контактами с высоким потенциалом. Для защиты от песка и пыли необходимо предусматривать соответствующие уплотнители, герметизацию корпуса изделия, замену разъемов на пайки и специальные покрытия.

К **активным (агрессивным) веществам** в атмосфере относятся сернистый газ, хлористые соли, пары кислот, щелочей и др. Их содержание в атмосфере районов, находящихся в прибрежной зоне, значительно больше, чем во внутриконтинентальных районах. Различают три типа атмосферы: атмосфера сельской местности (содержание сернистого газа не более  $0,02 \text{ мг/м}^3$ ), атмосфера промышленного района (сернистый газ  $0,02 \dots 2 \text{ мг/м}^3$ , хлористые соли не более  $0,3 \text{ мг/м}^3$ ), морская атмосфера (хлористые соли  $2 \dots 2000 \text{ мг/м}^3$ ). Защита изделий от агрессивных воздействий сводится к герметизации и специальным покрытиям.

**Солнечное облучение** также активно воздействует на работоспособность схем. Спектр излучаемой солнцем энергии состоит из трех составляющих: ультрафиолетовая часть, видимая часть, инфракрасная часть. Аппаратура, находясь под воздействием прямых солнечных лучей, может сильно нагреваться, что вызывает такие же изменения в ней, что и при температурных воздействиях. Ультрафиолетовые излучения солнца вызывают химические изменения в ряде изоляционных материалов. Некоторые типы пластмасс меняют свой цвет, быстро стареют и становятся хрупкими. Лакокрасочные покрытия обесцвечиваются, растрескиваются и отслаиваются. Большинство металлов и керамических материалов под действием космической радиации практически не изменяет своих свойств. Неорганические материалы также очень слабо подвержены влиянию космической радиации. Защита изделий от солнечного облучения сводится к применению радиационно-стойких материалов, защитных экранов и покрытий.

**Грибковые образования (плесень)** относятся к низшим растениям. В процессе жизнедеятельности они выделяют лимонную, уксусную, щавелевую кислоты и другие химические вещества, под действием которых изменяются характеристики многих материалов. Эти вещества способствуют ускорению процессов коррозии, ухудшают электроизоляционные свойства материалов и т. д. Идеальные условия для развития грибковых образований: температура 25... 35°C, относительная влажность 80... 100 %, неподвижность воздуха, отсутствие света (особенно ультрафиолетовой и инфракрасной частей спектра). Специальные лакокрасочные покрытия позволяют предохранить РЭА.

## 13.2. МЕХАНИЧЕСКИЕ ФАКТОРЫ

В процессе транспортирования и эксплуатации аппаратура подвергается воздействию **вибраций**, представляющих собой сложные колебания, которые возникают при контакте конструктивных элементов с источником колебаний. Особо опасны вибрации, частота которых близка к собственным частотам колебаний узлов и элементов конструкции. Вибрации подвержена аппаратура, устанавливаемая на автомобильном и железнодорожном транспорте, на кораблях, самолетах и ракетах. Под действием силы в деталях несущих конструкций возникают механические напряжения, которые могут привести к поломке. При вибрации (даже при малых уровнях) с течением времени возникают разрушения элементов конструкции за счет явлений усталости, которые при знакопеременных нагрузках проявляются больше, чем при статических. При длительном действии вибрации разбалтывают винтовые соединения, расшатываются заклепочные, а сварные — разрушаются. Происходит обрыв проводов и элементов, закрепленных за выводы.

**Явление удара** в конструкции РЭА возникает в случаях, когда объект, на котором она установлена, претерпевает быстрое изменение ускорения. Удар характеризуется ускорением, длительностью и числом ударных импульсов. Удары возникают при приземлении самолета, маневрировании вагонов железнодорожного транспорта, падении приборов и т. д. Части конструкций смещаются со своих фиксированных мест или даже срываются с крепления. Удары разрушают сварные и заклепочные соединения. Элементы крепления деталей, выполненных из материалов, обладающих малой удельной вязкостью, например алюминиевое

литье, керамика, некоторые пластмассы, при воздействии ударных нагрузок нередко обламываются.

**Линейное ускорение** характеризуется ускорением (в единицах  $g$ ) и длительностью воздействия. Возникает при резком изменении скорости движения объектов, на которых установлена РЭА.

**Акустический шум**, проявляющийся в приборах, устанавливаемых вблизи работающих двигателей ракет, самолетов, на кораблях, автомобильном и железнодорожном транспорте, характеризуется давлением звука, мощностью колебаний источника звука, силой звука, спектром звуковых частот.

Акустический шум подвергает механическим нагрузкам практически в равной степени все элементы конструкции. Ударно-вибрационные нагрузки воздействуют на элементы конструкции через их точки крепления. Детали крепления элементов в определенной мере являются своего рода демпферами, ослабляющими действие источника вибраций. Поэтому при прочих равных условиях действие акустического шума является более разрушительным, чем действие ударно-вибрационных нагрузок. Для защиты РЭА от механических воздействий используют специальные материалы конструкции, различные амортизаторы и специальные крепления.

## 13.3. РАДИАЦИОННЫЕ ФАКТОРЫ

Среди существующих видов излучений наибольшую опасность представляют электромагнитные излучения и ионизирующие частицы высоких энергий. Полный спектр электромагнитных излучений охватывает диапазон длин волн от десятков тысяч метров до тысячных долей нанометра. Наиболее значимое воздействие на РЭА оказывают гамма-излучение и рентгеновское излучение (длина волн менее 10 нм). Эти виды излучения обладают значительной проникающей и ионизирующей способностью.

Существенное воздействие на конструкцию РЭА могут также оказывать заряженные частицы: альфа, бета и протоны, а также нейтроны, обладающие высокой проникающей способностью.

Наименее стойкими к облучению являются полупроводниковые приборы и интегральные микросхемы. В них при облучении существенно изменяются характеристики вследствие изменения параметров входящих в них резисторов, конденсаторов, диодов, транзисторов. Так же изменяются изолирующие свойства разделительных  $p-n$ -переходов, возрастают токи утечки, появляются много-

численные паразитные связи между элементами структуры микросхем, что в результате приводит к нарушению их функционирования. Необратимые дефекты в полупроводниках приводят к потере выпрямительных свойств диодов; транзисторы всех типов при облучении теряют усилительные свойства, в них возрастают токи утечки, пробивное напряжение снижается. Их радиационная стойкость составляет  $10^{12} \dots 10^{14}$  нейтронов на  $1 \text{ см}^2$  при облучении нейтронами и  $10^4 \dots 10^7$  рад при гамма-облучении.

**Ионизирующие излучения** (ИИ) — это любые излучения, взаимодействие которых со средой приводит к образованию электрических зарядов разных знаков.

Ионизирующее излучение характеризуется полем (пространственно-временным распределением ИИ в рассматриваемой среде), потоком ионизирующих частиц  $\Phi_N$ , плотностью потока ионизирующих частиц  $\phi_N$ , потоком энергии ИИ  $\Phi_{ИИ}$ , плотностью потока энергии ИИ  $\phi_{ИИ}$ , переносом ионизирующих частиц  $F_N$ , переносом энергии ИИ  $F_{ИИ}$ .

**Радиационная стойкость** изделия или материала — свойство аппаратуры выполнять свои функции и сохранять параметры в пределах установленных норм во время воздействия ИИ. Воздействие ИИ на изделие проявляется в виде радиационного и ионизационного эффектов, обратимого или необратимого радиационных дефектов, радиационного разогрева и других явлений.

Металлы наиболее устойчивы к воздействию ИИ. Наименьшей радиационной стойкостью обладают электротехнические стали и магнитные материалы. Наименее устойчивы полупроводниковые и органические материалы.

В резисторах ИИ вызывает обратимые или необратимые изменения сопротивления, увеличение уровня шумов, ухудшение влагостойкости. Основные причины: деградация электрофизических характеристик резистивного и электроизоляционных материалов. Наиболее устойчивы к воздействию ИИ керамические и проволочные резисторы. В конструкции этих резисторов используются лишь радиационно-стойкие материалы: металл, керамика, стекло. Менее устойчивы к ИИ металлопленочные и пленочные углеродистые резисторы. Тонкопленочные интегральные резисторы способны выдерживать потоки быстрых нейтронов без существенных изменений величины сопротивления и параметров надежности. Наибольшей стойкостью к ИИ обладают танталовые, никелевые, нихромовые тонкопленочные резисторы, покрытые пассивирующей защитной пленкой. В радиационно-стойкой РЭА рекомендуется применять резисторы с номиналом менее 10 кОм. Высокоомные

резисторы защищают заливкой эпоксидной смолой. При уменьшении размеров резистора его устойчивость к ИИ повышается.

Воздействие ИИ сказывается на параметры электрической прочности-конденсаторов. Причины этих изменений: преобразования в структуре диэлектрика, механические деформации, ионизация диэлектрика и окружающей среды, выделение газов. Наибольшей стойкостью к ИИ обладают конденсаторы с неорганическим диэлектриком: керамические, стеклоэмалевые, слюдяные. Конденсаторы с органическим диэлектриком (бумажные, полистироловые, лавсановые, триацетатные, фторопластовые) обладают пониженной устойчивостью к ИИ (резко падает сопротивление изоляции, изменения номинала емкости составляют единицы или десятки процентов). Общая причина этих изменений — разложение полимерных материалов. Электролитические конденсаторы при облучении ненадежны. Отмечены случаи разгерметизаций из-за разложения электролита. Из интегральных тонкопленочных конденсаторов наиболее устойчивы к ИИ конденсаторы с диэлектриком на основе  $\text{Ta}_2\text{O}_5$  и  $\text{Al}_2\text{O}_3$ .

Воздействие ИИ на полупроводниковые приборы служит причиной обратимых либо необратимых радиационных дефектов, являющихся следствием ионизации и структурных нарушений в кристаллах. Ионизирующее действие радиации приводит к генерации в объеме полупроводника избыточных зарядов. Минимизация размеров полупроводниковых приборов повышает их устойчивость к ИИ.

Основные радиационные эффекты в диодах: фототоки (на один-два порядка больше рабочих токов), изменение сопротивления полупроводника, времени жизни носителей заряда. Наибольшей устойчивостью к ИИ обладают высокочастотные диоды (с тонкой базой).

Радиационная стойкость биполярных транзисторов в основном определяется деградацией коэффициента передачи по току. Главная причина деградации параметров биполярных транзисторов при ИИ — радиационные дефекты в полупроводниковом материале. При облучении биполярных транзисторов, не имеющих на поверхности кристалла защитных покрытий, наблюдается обратимое возрастание начального коллекторного тока. Для повышения радиационной стойкости РЭА рекомендуется применять высокочастотные транзисторы с пассивирующими покрытиями поверхности кристалла и с низкой мощностью рассеяния, работающие в режиме больших токов.

Радиационная стойкость униполярных транзисторов определяется изменениями поверхностных и объемных состояний, обуслов-



ленными процессами в оксиде, покрывающем поверхность приборов. Униполярные транзисторы выдерживают уровни ИИ меньшие, нежели биполярные. Среди униполярных транзисторов наибольшей устойчивостью к воздействию ИИ обладают приборы с управляющим  $p-n$ -переходом и  $p$ -каналом.

Действие ИИ на интегральные микросхемы проявляется в обратимых нарушениях работоспособности, вызванных ионизационными эффектами, и в необратимой деградации параметров. Основные причины нарушения работоспособности: изменение параметров у входящих в них элементов (резисторов, транзисторов и др.), повреждение межсоединений, ухудшение качества изоляции. Значительный интерес для использования в условиях ИИ представляют интегральные схемы (ИС) на основе керамических элементов (керамические твердые схемы).

**Конструктивно-технологические методы повышения радиационной стойкости** следующие:

- обеспечение стойкости к ИИ активных и пассивных элементов;
- создание надежной электрической изоляции элементов в условиях воздействия ИИ;
- использование радиационно-стойких проводящих и диэлектрических пассивирующих материалов;
- ослабление первичного ИИ за счет рационального выбора конструкции корпуса и применение материалов, поглощающих энергию ИИ (экранировка).

## 13.4. КЛАССИФИКАЦИЯ АППАРАТУРЫ ПО УСЛОВИЯМ ЭКСПЛУАТАЦИИ

Характер и интенсивность воздействия внешних дестабилизирующих факторов зависят от методов использования и объекта установки радиоэлектронной аппаратуры. По виду объекта установки РЭА можно разделить на три группы: стационарные, транспортируемые и портативные (ГОСТ 16019—2001).

**Стационарная РЭА** — это аппаратура, эксплуатируемая в отапливаемых и неотапливаемых помещениях, помещениях с повышенной влажностью, на открытом воздухе под навесом, в производственных цехах. Условия эксплуатации и транспортирования такой аппаратуры характеризуются весьма широким диапазоном рабочих ( $-50...+50^{\circ}\text{C}$ ) и предельных ( $-50...+65^{\circ}\text{C}$ ) температур, влажностью до 90...98 %, вибрацией до 120 Гц при 4...6  $g$ , наличием

многократных (до 5  $g$ ) и одиночных (до 75  $g$ ) ударов, воздействием дождя до 3 мм/мин и соляного тумана с дисперсностью капель до 10 мкм и содержанием воды до 3 г/м<sup>3</sup>.

**Транспортируемая РЭА** — это аппаратура, устанавливаемая и эксплуатируемая на автомобилях и автоприцепах, железнодорожном и гусеничном транспорте, судах различных классов, на борту самолетов и вертолетов, на ракетах и спутниках. Специфика работы этого вида аппаратуры предопределяет повышенное воздействие механических факторов. Каждый вид транспорта имеет собственные вибрационные характеристики. Для предупреждения повреждения аппаратуры необходимо, чтобы вся она и отдельные ее части имели собственные частоты колебаний вне диапазона частот вибрации транспортного средства.

На РЭА, установленную на автомобильном транспорте, могут воздействовать вибрация частотой до 200 Гц и удары, вызванные неровной дорогой. При движении железнодорожного транспорта возможны внезапные толчки (при маневрировании — удары с ускорением до 40  $g$ ). Биение колес о стыки рельсов вызывают вибрацию с частотой до 400 Гц при ускорении до 2  $g$ . Особо жестким воздействиям подвергается конструкция РЭА, эксплуатируемая на гусеничном транспорте. Здесь вследствие «стука» гусениц частота вибраций может достигать до 7000 Гц с амплитудой  $\pm 0,025$  мм. Кроме того, постоянно воздействие акустического шума.

Для РЭА в морском исполнении характерными условиями работы является наличие вибраций, ударных нагрузок и агрессивной (морской) атмосферы. Вибрация на судне вызывается работой винтов, гребного вала, двигателей и гидродинамическими силами при движении судна по беспокойному морю. Диапазон частот вибраций на кораблях обычно не превышает 25 Гц с небольшой амплитудой вибраций.

На самолетах электронная аппаратура находится, как правило, в фюзеляже. При этом на нее воздействуют вибрационные нагрузки частотой до 500 Гц с амплитудой до 10 мм и акустический шум, уровень которого достигает 150 дБ при частоте 50...10000 Гц.

Ракетно-космическая РЭА подвергается значительным воздействиям линейных ускорений и ионизирующего излучения.

**Портативная РЭА** включает в себя аппаратуру и специализированные вычислители, находящиеся в распоряжении геолога, геофизика, топографа, строителя и др. Условия работы портативной РЭА должны соответствовать зоне комфорта человека, которая характеризуется температурой окружающей среды 18...24 $^{\circ}\text{C}$ , уровнем акустического шума 70...85 дБ, влажностью 20...90 % и высотой



над уровнем моря до 3000 м. Если температура становится меньше  $-17^{\circ}\text{C}$  или выше  $+43,5^{\circ}\text{C}$ , уровень шума достигает 120 дБ, влажность составляет меньше 1 %, а высота над уровнем моря больше 6000 м, то считается, что такие условия превышают физиологические возможности человека, но предельные условия для перемещения аппаратуры могут быть много выше. С точки зрения физических возможностей человека портативная аппаратура подразделяется на легкую (до 29 кг для мужчин и до 16 кг для женщин), среднюю (соответственно до 147 кг и 80 кг) и тяжелую (до 390 кг и 216 кг). На портативную аппаратуру может воздействовать вибрация частотой до 20 Гц с ускорением до 2 *g* и удары до 10 *g* при длительности 5... 10 мс.

Различают и специальные виды РЭА, эксплуатируемые, например, в условиях химического производства. Для них характерны сверхбольшие значения одного — трех внешних факторов, на устойчивость к которым и проектируется конструкция такой РЭА.

Каждой из групп аппаратуры соответствует совокупность климатических и механических факторов, которой она должна соответствовать.

Аппаратура в эксплуатационных условиях должна быть стойкой, прочной и устойчивой к внешним воздействующим факторам (для бытовой аппаратуры климатические и механические воздействия задаются по ГОСТ 11478—88, для средств вычислительной техники — ГОСТ 21552—84, для малых вычислительных машин — ГОСТ 20397—82, для подвижной аппаратуры — ГОСТ 16019—2001).

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие факторы влияют на условия эксплуатации радиоэлектронной аппаратуры?
2. Что является нормальным климатическим условием?
3. Что такое ионизирующие излучения?
4. Какие конструктивно-технологические методы повышения радиационной стойкости вам известны?
5. Охарактеризуйте понятия «стационарная РЭА», «транспортируемая РЭА», «портативная РЭА».

## Глава 14

### ТРЕБОВАНИЯ К КОНСТРУКЦИИ РЭА

Разрабатываемая РЭА должна отвечать тактико-техническим, конструктивно-технологическим, эксплуатационным, надежностным и экономическим требованиям. Все эти требования взаимосвязаны, и оптимальное их удовлетворение представляет собой сложную техническую задачу.

#### 14.1. ТАКТИКО-ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Эти требования обычно содержатся в техническом задании на аппаратуру и включают в себя такие характеристики, как вид входных данных, диапазон их изменения, быстродействие, объем памяти для регистрации данных, точность выполнения вычислительных операций и т. д.

В основном данные требования удовлетворяются на ранних этапах разработки аппаратуры, когда определяются состав изделия, его структура, математическое обеспечение, основные требования к отдельным устройствам.

#### 14.2. КОНСТРУКТИВНО-ТЕХНОЛОГИЧЕСКИЕ ТРЕБОВАНИЯ

К этим требованиям относятся: обеспечение функционально-узлового принципа построения конструкции РЭА, технологичность, минимальная номенклатура комплектующих изделий, минимальные габаритные размеры и масса, меры защиты от воздействия климатических и механических факторов, ремонтоспособность.

Функционально-узловой принцип конструирования заключается в разбиении принципиальной схемы изделия на такие функционально законченные узлы, которые могут быть выполнены в виде идентичных конструктивно-технологических единиц. Применение этого принципа конструирования позволяет автоматизировать процессы изготовления и контроля конструктивных единиц, упростить их сборку, наладку и ремонт.

Технологичность конструкции в существенной степени определяется рациональным выбором ее структуры, которая должна быть разработана с учетом автономного, отдельного изготовления и наладки основных элементов, узлов, блоков. Конструкция РЭА тем более технологична, чем меньше доводочных и регулировочных операций приходится выполнять после окончательной сборки изделий.

В технологичной конструкции должны максимально использоваться унифицированные, нормализованные и стандартные детали и материалы. Аппаратура считается также более технологичной, если в ней предусматривается минимальная номенклатура комплектующих изделий, материалов, полуфабрикатов.

Конструкция РЭА должна иметь минимальные габаритные размеры и массу, что особенно важно для бортовых и носимых приборов.

В конструкции аппаратуры необходимо предусматривать меры защиты от воздействия климатических и механических факторов.

К числу важных характеристик конструкции РЭА следует также отнести ремонтоспособность — качество конструкции к восстановлению работоспособности и поддержанию заданной долговечности. Для повышения ремонтоспособности в конструкции предусматривают:

- доступность ко всем конструктивным элементам для осмотра и замены без предварительного удаления других элементов;
- наличие контрольных точек для подсоединения измерительной аппаратуры при настройке и контроле;
- применение быстросъемных фиксаторов и т. д.

Конструкция аппаратуры тем ремонтоспособнее, чем меньшую конструктивную единицу она позволяет оперативно заменять.

### 14.3. ЭКСПЛУАТАЦИОННЫЕ ТРЕБОВАНИЯ

К эксплуатационным требованиям относятся: простота управления и обслуживания, различные меры сигнализации опасных режимов работы (выход из строя, обрыв заземления и т. д.), наличие

аппаратуры, обеспечивающей профилактический контроль и наладку конструктивных элементов (стенды, имитаторы сигналов и т. д.).

В последнее время развивается направление построения систем высокой надежности и живучести, имеющих в своем составе средства самодиагностики и автореконфигурации системы.

### 14.4. ТРЕБОВАНИЯ ПО НАДЕЖНОСТИ

Данные требования включают в себя обеспечение:

- вероятности безотказной работы;
- наработки на отказ;
- среднего времени восстановления работоспособности;
- долговечности;
- сохраняемости.

**Вероятность безотказной работы** есть вероятность того, что в заданном интервале времени при заданных режимах и условиях работы в аппаратуре не произойдет ни одного отказа.

**Наработкой на отказ** называется средняя продолжительность работы аппаратуры между отказами.

**Среднее время восстановления работоспособности** определяет среднее время на обнаружение и устранение одного отказа.

**Долговечностью прибора** — продолжительность его работы до полного износа с необходимыми перерывами для технического обслуживания и ремонта. Под полным износом при этом понимают состояние аппаратуры, не позволяющее ее дальнейшую эксплуатацию.

**Сохраняемость аппаратуры** — способность сохранять все технические характеристики после заданного срока хранения и транспортирования в определенных условиях.

### 14.5. ЭКОНОМИЧЕСКИЕ ТРЕБОВАНИЯ

К экономическим требованиям относятся:

- минимально возможные затраты времени, труда и материальных средств на разработку, изготовление и эксплуатацию изделия;
- минимальная стоимость аппаратуры после освоения в производстве.

Тесная связь предъявляемых к аппаратуре требований приводит к тому, что стремление максимально удовлетворить одному из них

ведет к необходимости снизить значение других. Так, желание увеличить надежность введением структурной избыточности неизбежно влечет за собой увеличение габаритных размеров, массы, мощности потребления, стоимости. В данном случае выходом служит дальнейшее повышение степени интеграции микросхем.

Соотношение между различными требованиями может быть установлено исходя из типа, назначения и характера эксплуатации проектируемых изделий.

Для больших универсальных ЭВМ наиболее важное требование — обеспечение максимального быстродействия, поскольку оно в существенной степени определяет их производительность. Наименее важное требование — обеспечение небольших габаритных размеров и массы.

Для встраиваемых приборов наиболее важные требования — высокая надежность и малые вес и габаритные размеры. Приборы для массового потребления должны, прежде всего, иметь малую стоимость. Достижение высокого быстродействия для этого класса приборов — желательное, но не обязательное требование. Обычно стремятся достичь относительного высокого быстродействия, доступного в определенной ценовой категории.

Бортовые изделия должны обладать высокой степенью надежности. При этом стоимость приборов в некоторых случаях не имеет существенного значения. Применение РЭА накладывает на их конструкцию дополнительные жесткие требования. О ремонте какого-либо бортового прибора в процессе эксплуатации не может быть и речи. Здесь должна быть обеспечена возможность быстрой замены вышедших из строя блоков запасными. Поэтому основным требованием к приборам, установленным на борту, является надежность. Не менее важные требования — способность работать практически во всех известных условиях эксплуатации, ремонтоспособность, малые габаритные размеры, масса, мощность потребления.

## 14.6. ПОКАЗАТЕЛИ КАЧЕСТВА КОНСТРУКЦИИ

Большое разнообразие РЭА требует от разработчиков знания наборов показателей, по которым можно сравнивать существующие модели РЭА. К таким показателям следует отнести следующие:

- сложность конструкции РЭА:  $C = K_1(K_2N + K_3M)$ , где  $N$  — число составляющих элементов;  $M$  — число соединений;  $K_i$  — масштабный и весовые коэффициенты соответственно;

- **число элементов, образующих РЭА:**  $N = \sum_{j=1}^{N_y} \sum_{i=1}^{K_n} n_{ji}$ , где  $N_y$  — число устройств в РЭА;  $K_n$  — число типов применяемых элементов;  $n_{ji}$  — число элементов  $i$  — типа, входящих в  $j$  — устройство;
- **объем РЭА:**  $V = V_N + V_c + V_k + V_{т.у.}$ , где  $V_N$  — общий объем интегральных микросхем и электрорадиоэлементов, образующих РЭА;  $V_c$  — объем, занимаемый всеми видами соединений;  $V_k$  — объем несущей конструкции, обеспечивающей прочность и защиту РЭА при транспортировании и эксплуатации;  $V_{т.у.}$  — объем теплоотводящего устройства;
- **коэффициент интеграции**, или коэффициент использования физического объема:  $q_u = V_N/V$ ; характеризует степень использования физического объема РЭА элементами, выполняющими полезную функциональную нагрузку, т. е. непосредственно определяющими электрическую схему РЭА;
- **общая масса РЭА**, определяемая как сумма масс, входящих в состав РЭА устройств:  $m = m_N + m_c + m_k + m_{т.у.}$ ;
- **общая мощность потребления:**  $P = \sum_{j=1}^{N_y} p_j$ , где  $p_j$  — мощность потребления  $j$ -устройства. (Известно, что 80 ... 90 % мощности потребления рассеивается в виде теплоты и определяет тепловой режим РЭА и соответствующие перегревы элементов конструкции.);
- **общая площадь**, занимаемая РЭА:  $S = \sum_{j=1}^{N_y} s_j$ , где  $s_j$  — площадь, требуемая для эксплуатации  $j$ -устройства РЭА;
- **собственная частота колебаний:**  $f_0 = (1/2)\sqrt{K/m}$ , где  $K$  — коэффициент жесткости конструкции,  $m$  — масса конструкции РЭА;
- **степень герметичности** конструкции, определяемая количеством газа, истекшем из определенного объема конструкции за известный отрезок времени:  $D = V_o P / T_{сл.}$ , где  $V_o$  — объем герметизированной части РЭА;  $T_{сл.}$  — срок службы РЭА;  $P$  — избыточное давление газа в конструкции РЭА;
- **вероятность безотказной работы РЭА**  $p(t)$  и **средняя наработка на отказ**  $T_{ср}$  — показатели надежности ЭА;
- **степень унификации РЭА:**  $K_{ун} = N_{ун}/N$ , где  $N_{ун}$  — количество унифицированных элементов;  $N$  — общее количество примененных в РЭА элементов;
- **коэффициент автоматизации** конструкторских работ:  $K_a = M_a/M$ , где  $M_a$  — количество конструкторских работ, выполнен-

ных с применением ЭВМ;  $M$  — общее число конструкторских работ при проектировании РЭА.

Важнейшим параметром, определяющим большинство эксплуатационных, конструкторских и экономических характеристик разрабатываемой РЭА, является **технологичность**.

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким требованиям должна отвечать разрабатываемая РЭА?
2. Какие характеристики включают в себя тактико-технические требования?
3. Что относят к конструктивно-техническим требованиям?
4. Что относят к эксплуатационным требованиям?
5. Что относят к требованиям по надежности?
6. Что относят к экономическим требованиям?
7. Что является важнейшим параметром разрабатываемой РЭА?

## Глава 15

### ОСНОВЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ЦИФРОВЫХ УСТРОЙСТВ

Различают неавтоматизированное, автоматизированное и автоматическое проектирование. Такие сложные объекты, как изделия современной электронно-вычислительной техники, вручную проектировать очень нелегко. Оптимальное решение может быть найдено только путем полного перебора всех возможных вариантов решений и выбора из них наилучшего. Автоматизированное проектирование подразумевает диалог человека с машиной. В этой системе конструктор сам устанавливает (вместо полного перебора) наиболее приемлемые пути решения, которые удовлетворяют требованиям технического задания (ТЗ).

#### 15.1. КЛАССИФИКАЦИЯ САПР

Системы автоматизированного проектирования (САПР, или CAD — *computer automation design*) в зависимости от характеристик и этапа жизненного цикла (ЖЦ) изделия, на котором они применяются, подразделяются на ряд групп.

По предметной области различают:

- **САПР системно-технического проектирования** — используется на этапе эскизного проекта при выборе системных и структурных решений;
- **САПР схмотехнического проектирования** — используется на этапе технического проекта при решении задач логического проектирования, моделирования, контроля, разработки диагностических тестов и др.;
- **САПР конструкторского проектирования** — используется на этапе технического проекта и рабочей конструкторской документации при решении следующих задач: покрытие функ-

ционально-логической схемы заданным набором микросхем, компоновка конструктивных модулей, размещение элементов  $i$ -го уровня конструктивной иерархии на следующий уровень, задача трассировки печатных и проводных соединений;

- **САПР технологического проектирования** — решает следующие задачи: разработка алгоритмов управления оборудованием с ЧПУ, разработка технологической документации.
- Для функционирования САПР необходимы специальные виды обеспечения.
- **Математическое обеспечение** (МО) — это совокупность математических методов, моделей и алгоритмов, представленных в заданной форме.
- **Техническое обеспечение** (ТО) — это совокупность взаимодействующих технических средств (компьютеры, графопостроители, плоттеры и т. д.).
- **Программное обеспечение** (ПО) — это совокупность программ, необходимых для обеспечения процесса проектирования (прикладное и системное ПО).
- **Информационное обеспечение** (ИО) — это совокупность сведений, необходимых для процесса проектирования. Включает СУБД, БД и базу знаний.
- **Лингвистическое обеспечение** (ЛО) — это совокупность языков проектирования. Включает в себя термины, определения, правила формализации естественного языка, методы сжатия и развертывания текстов.
- **Методическое обеспечение** (МТО) — это совокупность документов, устанавливающих состав и правила эксплуатации средств обеспечения системы.
- **Организационное обеспечение** (ОО) — это совокупность документов, определяющих состав проектной организации, связь между подразделениями, а также форму представления и порядок рассмотрения проектных документов.

## 15.2. САПР ПЕЧАТНЫХ ПЛАТ

**Печатная плата** (ПП) — это диэлектрическое основание, на котором определенным образом сформированы печатные проводники в строгом соответствии с электрической принципиальной схемой.

Помимо проводящего рисунка на печатной плате располагаются монтажные и переходные отверстия. Монтажные отверстия служат

для установки радиоэлементов со штыревыми выводами. Переходные отверстия служат для обеспечения электрической связи между печатными проводниками, расположенными с противоположных сторон печатной платы или на внутренних слоях многослойной печатной платы. Переходные отверстия выполняются металлизированными.

Для правильной ориентации печатной платы в процессе ее изготовления необходимо предусмотреть ключ. Как правило, он выполняется в левом нижнем углу.

Разработка очередных поколений элементной базы (интегральная, затем функциональная микроэлектроника) и ужесточение требований к электронным устройствам потребовали развития техники печатного монтажа и привели к созданию многослойных печатных плат (МПП), появлению гибких, рельефных печатных плат.

**Односторонние печатные платы.** Односторонние платы по-прежнему составляют значительную долю всех выпускаемых в мире печатных плат. В 90-е годы XX в. в США они составляли около 70 % объема выпуска плат в количественном исчислении, однако лишь около 10 % в стоимостном исчислении.

**Двухсторонние печатные платы.** Двухсторонние платы составляют в настоящее время значительную долю объема выпуска плат. Столь значительное внимание разработчиков к этому виду плат объясняется своеобразным компромиссом между их относительно малой стоимостью и достаточно высокими возможностями.

**Многослойные печатные платы.** Многослойные печатные платы (МПП) составляют 2/3 мирового производства печатных плат в ценовом исчислении, хотя в количественном выражении уступают одно- и двухсторонним платам.

По своей структуре МПП значительно сложнее двухсторонних плат. Они включают в себя дополнительные экранные слои (земля и питание), а также несколько сигнальных слоев.

Для обеспечения коммутации между слоями МПП применяются межслойные переходы (*vias*) и микропереходы (*microvias*).

Межслойные переходы могут выполняться в виде сквозных отверстий, соединяющих внешние слои между собой и с внутренними слоями, применяются также глухие и скрытые переходы.

**Глухой переход** — это соединительный металлизированный канал, видимый только с верхней или нижней стороны платы. Скрытые же переходы используются для соединения между собой внутренних слоев платы. Их применение позволяет значительно упростить разводку плат: например, 12-слойную конструкцию МПП можно свести к эквивалентной 8-слойной.



Специально для поверхностного монтажа разработаны микро-переходы, соединяющие между собой контактные площадки и сигнальные слои.

**Гибкие печатные платы.** Использование гибких диэлектрических материалов для изготовления печатных плат дает как разработчику, так и пользователю электронных устройств ряд уникальных возможностей. Это, прежде всего, уменьшение размеров и веса конструкции, повышение эффективности сборки, повышение электрических характеристик, теплоотдачи и в целом надежности.

Если учесть основное свойство плат — динамическую гибкость, становится понятным все возрастающий объем применения таких плат в автомобилях, бытовой технике, медицине, в оборонной и аэрокосмической технике, компьютерах, в системах промышленного контроля и бортовых системах.

**Процесс изготовления печатной платы.** Его можно условно разделить на следующие этапы:

- разработка узла и построение его принципиальной схемы;
- размещение принципиальной схемы на печатной плате;
- технологическая подготовка проекта;
- автоматизированное производство ПП;
- автоматизированный контроль ПП.

Основой первых трех этапов, которые можно объединить общим термином «конструкторско-технологическое проектирование», является использование современных САПР ПП. Применяются сквозные интегрированные САПР, которые имеют альтернативные алгоритмы реализации отдельных проектных процедур и позволяют гибкую настройку на требования технологий производства ПП. При этом термин «Интегрированные САПР» подразумевает пакеты, которые в комплексе выполняют функции CAD/CAM/CAE/PDM систем автоматизированного проектирования:

- *Computer-aided design (CAD-системы)* — решение конструкторских задач, оформление КД (САПР конструкторской разработки ПП);
- *Computer-aided manufacturing (CAM-системы)* — проектирование обработки изделий на станках с числовым программным управлением (ЧПУ) (САПР технологической проработки ПП);
- *Computer-aided engineering (CAE-системы)* — предназначены для инженерных расчетов.

Можно выделить следующие наиболее популярные программные продукты для решения задач конструкторско-технологического

проектирования: *Cadence Allegro*, *Cadence OrCAD*, *Altium P-CAD*, *Altium Designer*. В России это *P-CAD* и *Altium Designer*.

Особенно важным и исключительно ответственным этапом проектирования печатных плат является технологическая подготовка уже разработанного конструкторского проекта ПП к промышленному производству. Под этим подразумевается технологическая постобработка и генерация управляющих файлов для изготовления фотошаблонов, для сверления отверстий на станках с ЧПУ, для автоматического тестирования плат на автоматах и расстановки компонентов на автоматических линиях.

При конструкторско-технологическом проектировании ПП главным является требование целостности данных на всех этапах проекта: от разработки проекта до выпуска управляющих файлов для технологического оборудования с ЧПУ с использованием интегрированной САПР ПП.

Обобщенный алгоритм автоматизированной технологической подготовки производства ПП включает следующие этапы.

0. Разработка схемы устройства и ее размещение на плате в CAD.

1. Ввод исходных данных, полученных в системах проектирования печатных плат. При этом из CAD-программ данные конвертируются в специальные форматы.

2. Анализ и подготовка проекта с учетом критериев технолога:
- *DRC (Design Rule Check)* — анализ и проверка данных на соответствие результатов проектирования требованиям производства ПП. На этом этапе проверяются заданные требования на расстояния между проводниками, контактными площадками, на размер контактных площадок и т.д.;
  - редактирование топологии печатной структуры на уровне отдельных проводников, участков металлизации и контактных площадок;
  - определение и коррекция перекрывающихся или не функциональных элементов;
  - каплевидное сглаживание стыков проводников с контактными площадками;
  - вычисление суммарной площади металлизации ПП;
  - размещение нескольких плат на групповой заготовке.

3. Экспортирование сформированной технологической документации и управляющих файлов для оборудования с ЧПУ на участки автоматизированного производства ПП.

Такой подход к конструкторско-технологическому проектированию ПП позволяет существенно снизить процент брака.

## 15.3. КОНСТРУКТОРСКИЕ САПР

Средства автоматизированного проектирования (CAD) применяются для преобразования электрической цепи, описанной принципиальной схемой, в физическую компоновку или печатную плату. Эти средства обеспечиваются таблицами соединений схем, перечнями компонентов, правилами трассировки и другой информацией размещения с помощью систем ввода этих описаний или средствами CAD. В своем простейшем виде они позволяют проектировщику создавать схемы контактных площадок для выводов компонентов и форму печатной платы, а затем вручную соединять выводы компонентов медными трассами. Средства CAD в состоянии автоматически определять оптимальное положение каждого компонента печатной платы (авторазмещение) и затем автоматически соединять (автотрассировка) все выводы, соблюдая при этом правила компоновки. Завершается этот процесс с помощью свода правил, определяющих, какие компоненты должны быть размещены в группах или вблизи разъемов, а также определением того, какое пространство нужно обеспечить между соседними трассами, максимально допустимой длины между точками на схеме и т. п.

Выходными данными программных средств проектирования CAD являются информационные файлы, которые необходимы для производства, сборки и тестирования печатной платы. К этим файлам относятся тестовые таблицы соединений, файлы для фотоплоттера, ведомости материалов, файлы по монтажу и сборочные чертежи. Средства CAD состоят из программы трассировки схем, установки компонентов, средств тестирования и средств создания выходных файлов.

**Трассировщики.** Трассировщики ПП являются частью системы CAD. Они осуществляют соединения между компонентами, как это определено списками межсоединений. Трассировщик работает по спискам соединений печатной платы и по их размещению, после того как был завершен этап размещения компонентов. Трассировщики варьируются от функционирующих полностью в ручном режиме, в которых проектировщик определяет, где будут размещены трассы, с помощью графического дисплея с мышью или светового пера, до полностью автоматизированных, в которых специализированная программа берет список соединений и, используя правила размещения и правила создания пространств между компонентами, а также правила трассировки, принимает все решения, необходимые для полного соединения всех компонентов между собой.

*Основным преимуществом* ручной трассировки является то, что проектировщик может подогнать каждое соединение по своему вкусу. *Основным недостатком* ручной трассировки является то, что она выполняется достаточно медленно, иногда затрачивая несколько минут для полной трассировки и проверки одной цепи. Автоматизированные трассировщики решают эту задачу быстро. Однако возможность детального контроля формы каждой цепи ограничена способностью автотрассировщика следовать правилам трассировки соединений. Некоторые современные автотрассировщики в состоянии согласовать трассировку с очень сложными правилами. Существенной проблемой автотрассировщиков является то, что они не в состоянии найти способ успешной трассировки *всех* проводников. Когда такое происходит, проектировщик должен добавить больше места для проводников за счет дополнительных слоев или попытаться завершить трассировку в ручном режиме. Важной функцией хорошего автотрассировщика является наличие возможности дотрассировки в ручном режиме, которая зачастую бывает необходима. Почти все трассировщики имеют комплект средств проверки и это гарантирует, что окончательная трассировка согласуется со списком соединений и, что все правила по резервированию свободного пространства между компонентами были соблюдены.

**Трассировщик с сеткой.** Этот тип трассировщика (сеточный трассировщик) работает с размещением проводников на предварительно заданную координатную сетку. *Основными недостатками* трассировщиков с координатной сеткой являются сложность управления более чем одной шириной трасс без потери в плотности прокладки и необходимость задания конечных точек цепей, которые должны быть на сетке трасс. Соединения компонентов вне координатной сетки обычно необходимо выполнять вручную и проверять аналогичным образом.

**Трассировщик без сетки.** Расположение проводников в этой разновидности программы трассировки (безсеточный трассировщик) не зависит от координатной сетки. Программа размещает максимальное количество проводников, которые могут разместиться в имеющемся пространстве при соблюдении правил резервирования свободных промежутков, обеспечивающих надлежащие электрические характеристики при оптимизации технологичности процесса изготовления. Преимущество этого метода заключается в возможности оптимизации технологичности процесса изготовления за счет поддержания как можно больших промежутков между компонентами. Эта разновидность трассировщиков исполь-

зуется при проектировании цифровых конструкций высокой сложности.

**Трассировка с учетом формы.** Этот тип трассировщика (трассировщик с учетом формы) распознает формы уже установленных на монтажной поверхности элементов и прокладывает проводники в обход. Свободное пространство между проводниками и другими объектами, такими как сквозные отверстия, используемые для перехода на другой слой, и контактные площадки, сохраняется как трассировочное пространство.

**Средства проверки.** С помощью этих инструментов выполняется *проверка соответствия* трассировки печатной платы правилам резервирования свободного пространства между трассами и между отверстиями и трассами путем сравнения фактических промежутков, имеющих в готовых трафаретах, с теми, которые установлены правилами конструктора плат. Они также гарантируют, что все цепи полностью подключены и не соединены с объектами, с которыми соединять их не следует.

**Генераторы выходных файлов.** После трассировки печатной платы и проверки точности всех соединений система *CAD* сохраняет эту информацию. Затем эти данные следует преобразовать в формат, который можно использовать на технологическом оборудовании, таком как фотоплоттеры, тестеры и монтажное оборудование. Процедуры, генерирующие выходные файлы, выполняют это преобразование.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. На какие группы подразделяется САПР (по предметной области)?
2. Какие виды обеспечения необходимы для функционирования САПР?
3. Что такое печатная плата?
4. Что такое глухой переход?
5. На какие этапы можно разделить процесс изготовления печатной платы?
6. Перечислите преимущества и недостатки ручной трассировки.

## Глава 16

# АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ ПРОИЗВОДСТВА ИЗДЕЛИЙ РАДИОЭЛЕКТРОНИКИ

Изготовление изделий радиоэлектроники — сложный процесс, включающий в себя множество различных этапов, технологических операций, применяемых методик по управлению качеством и технологичностью, характеризующийся специфичностью производства на основе применения высокотехнологичного автоматизированного оборудования, состав которого постоянно модернизируется.

Условно этот процесс можно разделить на три этапа:

- изготовление печатных плат;
- монтаж печатных плат;
- сборка печатных плат в изделие.

Каждый из этапов включает в себя помимо непосредственных типовых операций автоматизированный контроль, который позволяет избежать возможность появления брака и снизить вероятность выхода изделия электроники из строя в процессе эксплуатации.

## 16.1. ПРОИЗВОДСТВО ПЕЧАТНЫХ ПЛАТ

Рассмотрим основные технологии изготовления печатных плат.

**Субтрактивная технология** предусматривает травление медной фольги на поверхности диэлектрика по защитному изображению в фоторезисте (светочувствительном покрытии). Эта технология широко применяется при изготовлении односторонних и двусторонних слоев МПП.

**Аддитивная технология** позволяют уменьшить ширину проводников и зазоров до 50 ... 100 мкм при толщине проводников 30 ... 50 мкм. От субтрактивных процессов этот метод принципиально отличается тем, что металл проводников не вытравливают, а наносят.

**Комбинированный позитивный метод** включает в себя следующие основные этапы производства.

1. **Изготовление фотошаблонов и подготовка информации.** К этому этапу относятся:

- *подготовка информации* — осуществляется разработка принципиальной схемы устройства, затем трассировка. Принципиальная электрическая схема преобразуется в схему разводки слоев. Затем следует доработка файлов и вывод файлов сверления и фрезерования;
- *изготовление фотошаблонов* — производится изготовление фотошаблонов, которые затем используются для формирования топологического рисунка внутренних и внешних слоев печатной платы при экспонировании. Различают позитивные и негативные фотошаблоны.

2. **Резка заготовок.** Листы стеклотекстолита нарезаются на заготовки. Очень важно правильно выбрать размеры заготовок, так как от этого зависит коэффициент использования материала. Обычно размер заготовок выбирается кратным листу стеклотекстолита (914,4×1220 мм). Резка заготовок может производиться на гильотинных ножницах (ручных или автоматических) или на роликовых ножницах.

3. **Изготовление базовых отверстий.** На этом этапе в заготовке изготавливается набор базовых отверстий. Обычно базовые отверстия круглой формы выполняются сверлением, а овальной — вырубкой.

4. **Ламинирование.** Следующий этап — нанесение пластичного фоточувствительного материала на заготовку. Заготовка очищается и приготавливается к нанесению фоторезиста. Этот этап проходит в чистой комнате с желтым освещением. Резист светочувствителен (обычно к ультрафиолету) и при долгом не использовании разрушается.

5. **Экспонирование.** К этому этапу относятся:

- *размещение фотошаблона* — на заготовке размещается фотошаблон. Изображение на фотошаблоне негативное по отношению к будущей схеме. Под темными участками фотошаблона медь не будет удалена. При использовании систем базирования точность совмещения определяется точностью изготовления базовых отверстий в заготовках и фотошаблоне. В случае ручного совмещения точность зависит от квалификации и усталости оператора. Наиболее точной системой совмещения является автоматическая оптическая система совмещения — система анализирует

расположение реперных знаков и выбирает оптимальное положение фотошаблона;

- *экспонирование фоторезиста* — участки поверхности, не защищенные фотошаблоном, засвечиваются. Фотошаблон снимается. После этого засвеченные участки могут быть удалены химически.

6. **Химическая обработка.** Эти операции производятся в установках химической обработки. Существует несколько типов установок: струйные, погружные, конвейерного типа и с ручной загрузкой. К этому этапу относятся:

- *проявление* — засвеченные участки фоторезиста удаляются, оставляя фоторезист только в тех областях, где будут проходить дорожки платы. Назначение фоторезиста — защитить медь под ним от воздействия травителя на следующем этапе;
- *травление* — заготовка травится для удаления ненужной меди. Резист, оставшийся на поверхности, предохраняет медь под ним от травления. Вся незащищенная медь удаляется, оставляя диэлектрическую подложку. После травления дорожки схемы созданы и внутренний слой имеет требуемый рисунок;
- *удаление резиста* — резист удаляется, открывая не вытравленную медь. Теперь заготовка представляет собой полностью готовый внутренний слой.

7. **Прессование.** На этом этапе плата собирается в пакет, состоящий из внутреннего и внешних слоев, проложенных препрегом (материалом, служащим в качестве клея). На границах пакета необходимо использование дополнительных слоев, служащих для защиты пластин пресса от попадания расплавленного препрега и простоты разборки пакета. Точность совмещения слоев при прессовании в основном определяется системой совмещения, а так же точностью используемой оснастки.

8. **Сверление отверстий.** Отверстия на плате служат двум целям: обеспечивать соединение между слоями и для монтажных целей. Платы сверлятся на станках с программным управлением, нередко называемым обрабатывающими центрами. Точность сверления определяется классом оборудования, а так же его настройкой.

9. **Металлизация отверстий.** Этот этап служит для покрытия отверстий тонким слоем металла. Для металлизации плата помещается в ванну, где плата полностью химически покрывается тонким слоем палладия. Сущность процесса химическая, и в результате покрываются как диэлектрические, так и металлические поверхности.



#### 10. Химическая обработка. К данному этапу относятся:

- *нанесение резиста* — далее плата покрывается резистом, он засвечивается через фотошаблон, засвеченные участки удаляются. Эти этапы аналогичны описанным ранее с одним отличием: резист удаляется с участков, где будет наноситься медь. Следовательно, изображение на фотошаблоне должно быть позитивным;
- *электролитическое нанесение меди* — медь наносится на поверхность отверстия до толщины 0,25 мм. Медь, осажденная ранее на поверхность отверстия, достаточно толстая, чтобы проводить ток для электролитического осаждения меди. Это необходимо для надежного электрического соединения сторон и внутренних слоев платы;
- *оловянно-свинцовое покрытие* — оловянно-свинцовое электролитическое покрытие выполняет две важные функции. Во-первых, оловянно-свинцовая смесь выступает резистом для последующего травления. Во-вторых, она защищает медь от окисления;
- *удаление резиста* — резист удаляется, оставляя оловянно-свинцовую смесь (припой) и нанесенную медь. Медь, покрытая припоем, выдержит процесс травления и образует собой рисунок платы;
- *травление меди* — на этом этапе припой используется как резист для травления. Незащищенная медь удаляется, оставляя на плате рисунок будущей схемы;
- *удаление припоя* — припой удаляется с поверхности меди, и плата очищается.

11. **Нанесение защитного покрытия.** Для защиты поверхности платы, где в дальнейшем не потребуются пайка, наносится маска. Существует несколько типов масок и методов ее нанесения. Нанесение через трафарет не обладает такой точностью, но материал маски более пластичен, и стоимость процесса ниже. Также после нанесения маски наносят маркировочные знаки шелкографией по аналогии с маской.

## 16.2. АВТОМАТИЗАЦИЯ СБОРКИ РЭА. СБОРКА ПЕЧАТНЫХ ПЛАТ

Особенностью современного производства электронных устройств является все более широкое применение больших и сверхбольших интегральных схем. При этом существенно возрастает

количество выводов каждой схемы, расстояния между выводами уменьшаются от 2,5 мм до 0,625 мм и менее.

Установка многовыводных корпусов БИС и СБИС на печатные платы технически и экономически более эффективна не в сквозные отверстия, а на контактные площадки, расположенные на поверхности печатных плат.

Этим объясняется все более широкий переход от монтажа компонентов в отверстия (*PTH — Plated Through Hole*) к технологии поверхностного монтажа (*SMT — Surface Mount Technology*).

В настоящее время в большинстве серийных электронных блоков применяют как поверхностный монтаж, так и монтаж в отверстия. Это связано с тем, что конструкции ряда компонентов не пригодны для поверхностного монтажа. Кроме того, в устройствах, работающих в условиях ударных и вибрационных перегрузок, предпочитают монтаж в отверстия из-за более надежного крепления компонентов.

### 16.2.1. Поверхностный монтаж

**Поверхностный монтаж печатных плат**, также называемый технология монтажа на поверхность (ТМП), *SMT- (surface mount technology)* и *SMD-технология* (от англ. *surface mount device* — прибор, монтируемый на поверхность), появился в 60-х годах XX в. и получил широкое развитие в конце 80-х годов XX в. Данная технология является наиболее распространенным на сегодняшний день методом конструирования и сборки электронных узлов на печатных платах. Основным ее отличием от «традиционной» технологии монтажа в отверстия является то, что компоненты монтируются на поверхность печатной платы, однако преимущества технологии поверхностного монтажа печатных плат проявляются благодаря комплексу особенностей элементной базы, методов конструирования и технологических приемов изготовления печатных узлов.

Монтаж микросхем на поверхностные контактные площадки без отверстий, так называемый **планарный монтаж**, во второй половине XX в. успешно применялся в специальной технике. Корпуса микросхем для планарного монтажа имели выводы по двум или четырем сторонам. Обрезка и формовка выводов осуществлялась перед установкой, после чего микросхема фиксировалась на клей или подпайкой и припаивалась специальными роликовыми или гребенчатыми паяльниками, либо на установке пайки волной.



С другой стороны, во время появления поверхностного монтажа существовала и другая технология — **технология гибридных модулей и микросхем**, в которых применялись компоненты с укороченными выводами или вообще без выводов, устанавливаемые на керамические подложки. Также такие компоненты применялись в СВЧ-технике, где длина выводов может оказывать существенное влияние на качество сигнала.

Технология поверхностного монтажа по сравнению с технологией монтажа в отверстия обладает рядом преимуществ как в конструкторском, так и технологическом аспекте:

- **снижение габаритных размеров и массы печатных узлов** — компоненты для поверхностного монтажа имеют значительно меньшие размеры по сравнению с элементной базой для монтажа в отверстия. Как известно, большую часть массы и габаритных размеров микросхемы составляет не кристалл, а корпус и выводы. Размеры корпуса продиктованы в основном расположением выводов (могут существовать и другие факторы, например требования по теплоотводу, но они значительно реже являются определяющими). Поверхностный монтаж позволяет применять компоненты с существенно меньшим шагом выводов благодаря отсутствию отверстий в печатной плате. Поперечные сечения выводов могут быть также меньше, поскольку выводы формуруются на предприятии-изготовителе компонентов и не подвергаются существенным механическим воздействиям от разупаковки до установки на плату. Кроме того, эта технология позволяет применять корпуса компонентов с контактными поверхностями, заменяющими выводы.

Современная технология поверхностного монтажа позволяет устанавливать компоненты с обеих сторон печатной платы, что позволяет уменьшить площадь платы и, как следствие, габариты печатного узла;

- **улучшение электрических характеристик** — за счет уменьшения длины выводов и более плотной компоновки значительно улучшается качество передачи слабых и высокочастотных сигналов;
- **повышение технологичности** — это преимущество является, пожалуй, основным, позволившим поверхностному монтажу получить широкое распространение. Отсутствие необходимости подготовки выводов перед монтажом и установки выводов в отверстия, фиксация компонентов паяльной пастой или клеем, самовыравнивание компонентов при пайке — все это позволяет применять автоматическое технологическое оборудование с производительностью, недостижимой при соответствующей

стоимости и сложности технических решений при монтаже в отверстия. Применение технологии оплавления паяльной пасты значительно снижает трудоемкость операции пайки по сравнению с ручной или селективной пайкой, и позволяет экономить материалы по сравнению с пайкой волной;

- **повышение ремонтпригодности** — современное ремонтное оборудование позволяет снимать и устанавливать компоненты без повреждений даже при большом количестве выводов. При монтаже в отверстия эта операция является более сложной из-за необходимости равномерного прогрева достаточно тепломеханических соединений. При поверхностном монтаже тепломеханическая прочность соединений меньше, а нагрев может осуществляться по поверхности горячим воздухом или азотом. Тем не менее некоторые современные компоненты для поверхностного монтажа являются настолько сложными, что их замена требует специального оборудования;
- **снижение себестоимости** — уменьшение площади печатных плат, меньшее количество материалов, используемых в компонентах, автоматизированная сборка — все это при прочих равных условиях позволяет существенно снизить себестоимость изделия при серийном производстве.

## 16.2.2. Типичная последовательность операций монтажа

В технологии поверхностного монтажа, как правило, применяются два метода пайки: **пайка оплавлением** припойной пасты и **пайка волной**. В зависимости от применяемого метода пайки последовательность операций различна.

Основное преимущество **метода пайки волной** — возможность одновременной пайки компонентов, монтируемых как на поверхность платы, так и в отверстия. При этом пайка волной является самым производительным методом пайки при монтаже в отверстия. В современных конструкциях доля монтажа в отверстия постоянно снижается, а развитие более экономной и качественной селективной пайки позволяет автоматизировать пайку компонентов, монтируемых в отверстия, без применения волны. Эти факторы приводят к тому, что производители все чаще отказываются от пайки волной, применяя метод оплавления для поверхностно-монтируемых компонентов и ручную или селективную пайку для компонентов, монтируемых в отверстия.

Пайка волной, как и селективная пайка, применяется при так называемой смешанной технологии, когда на плате одновременно присутствуют компоненты, монтируемые на поверхность и в отверстия. Полностью избавиться от монтажа в отверстия в большинстве современных устройств не удастся, тем не менее множество изделий уже собирается с применением только поверхностного монтажа.

Прежде чем привести типичную последовательность операций при использовании метода пайки оплавлением для сборки платы, не содержащей компонентов для монтажа в отверстия, рассмотрим состав и особенности паяльной пасты.

**Пайка оплавлением** основана на применении специального технологического материала — паяльной пасты. Она содержит три основных составляющих: припой, флюс (активаторы) и органические наполнители.

Припой в паяльной пасте содержится в виде частиц, имеющих, как правило, форму шариков. Размер шариков составляет несколько десятков микрометров, типичное значение 20...25 мкм. Форма шариков наиболее оптимальна с точки зрения нанесения пасты, поскольку они легко и предсказуемо проходят через отверстия трафарета и иглы дозаторов и приводят к минимальному износу оснастки. Кроме того, шарик, имея минимальную площадь поверхности при заданном объеме, обладает наилучшими характеристиками по окислению. Состав припойного сплава, применяемого в пастах, такой же, как и при других методах пайки. Обычно это эвтектический сплав олово-свинец, либо SAC-сплав ( $Sn - Ag - Cu$ ) при применении бессвинцовой технологии. Широкое распространение получили сплавы олово-свинец с добавлением 2 % серебра, обеспечивающие снижение миграции серебра с покрытия контактных поверхностей компонентов в материал припоя.

Содержание припойной фракции обычно составляет порядка 50 % по объему и 90...95 % по массе.

Флюсы служат для подготовки поверхности перед пайкой. Их наличие в паяльной пасте является преимуществом метода оплавления, поскольку позволяет отказаться от операции нанесения флюса. Флюсы различаются по активности и методу удаления остатков. Активные флюсы применяются при пайке компонентов и плат с плохой паяемостью, либо когда качество подготовки поверхностей критично по другим причинам. В бессвинцовой технологии из-за худшего смачивания поверхностей припоем применяются более активные флюсы, чем при использовании оловянно-свинцовых

припоев. Недостатком активных флюсов является необходимость их тщательного удаления после пайки. Остатки активных флюсов могут приводить к коррозии проводников платы в процессе эксплуатации, а также при повышенной влажности вызывать образование электролитов на поверхности плат, приводящих к гальваническим эффектам (например, росту медных дендритов).

По методу удаления остатков большинство флюсов подразделяются на не требующие отмывки, водосмываемые и смываемые растворителями. Если флюс не требует отмывки, это не означает, что его остатков на плате после пайки нет. Остатки таких флюсов не влияют на внешний вид изделия и не приводят к выходу изделия из строя при нормальных условиях эксплуатации. Такие флюсы применяются в бытовой и лабораторной аппаратуре и имеют низкую активность. В аппаратуре, эксплуатируемой при воздействии повышенной влажности и в широком диапазоне температур, применение таких флюсов нежелательно, и их остатки должны быть удалены растворителями.

Остатки водорастворимых флюсов могут удаляться горячей деионизованной водой. Эти флюсы могут быть активны. Иногда в состав паст с водосмываемыми флюсами вводятся поверхностно-активные вещества, улучшающие процесс отмывки. Флюсы, требующие отмывки, должны удаляться в течение строго определенного промежутка времени после пайки. Обычно это время составляет 8 ч.

Ввиду широкой распространенности и технологичности водосмываемых флюсов и флюсов, не требующих отмывки, флюсы, смываемые растворителем, практически не применяются.

В подавляющем большинстве случаев при сборке электроники применяются именно флюсы, не требующие отмывки, так как это позволяет уменьшить количество операций и снизить стоимость процесса.

Прочие органические наполнители вводятся в состав паяльных паст для регулирования их свойств, таких как тиксотропность, холодная и горячая осадка, клейкость и др.

**Последовательность операций при применении технологии поверхностного монтажа с использованием пайки оплавлением.** После разупаковки и очистки платы, как правило, выполняется следующая последовательность операций.

1. Нанесение паяльной пасты. Паяльная паста наносится на контактные площадки либо с помощью дозатора, либо через трафарет. При выполнении данной операции необходимо получение отпечатков, содержащих определенный объем пасты. Недостаток пасты

может приводить к отсутствию соединения, избыток — к перемычкам и низкой прочности соединения.

**Использование дозатора** — более гибкий, но менее точный и производительный метод, обычно применяющийся при опытном производстве. Пасты для дозирования поставляются в стандартных шприцах, совместимых с большей частью оборудования.

На шприц устанавливаются иглы различного диаметра, обеспечивающие нанесение определенного объема пасты. Некоторые автоматы установки компонентов начального уровня имеют возможность установки дозатора вместо установочной головки.

**Трафаретная печать** — наиболее распространенный метод нанесения пасты в серийном производстве. Паста наносится путем продавливания ракелем через апертуры (отверстия) в металлическом трафарете. Объем пасты определяется размером апертур и толщиной трафарета. Апертуры, как правило, выполняются несколько меньшими по размерам, чем контактные площадки (примерно на 5... 10 % с каждой стороны). В некоторых случаях для получения требуемого объема пасты применяются ступенчатые трафареты с переменной толщиной. Трафарет обычно выполняется из нержавеющей стали методом лазерной резки. Трафаретная печать выполняется на автоматах, полуавтоматах и вручную. Основными режимами, влияющими на качество печати, являются скорость, угол наклона и усилие ракеля. Скорость ракеля обычно задается характеристиками пасты. Типичное ее значение составляет порядка 20... 25 мм/с, однако современные пасты допускают печать со скоростью 150... 200 мм/с. Типичный угол наклона ракеля составляет 60°. Ракель должен двигаться таким образом, чтобы паста образовала катящийся валик.

Автоматы выполняют нанесение полностью автоматически, включая совмещения трафарета с платой, проход ракеля, отделение трафарета и его очистку. Полуавтоматы обеспечивают необходимые угол наклона и усилие на ракель, а движение ракеля осуществляется оператором вручную по направляющим.

2. Установка компонентов. Установка компонентов осуществляется, как правило, по программе на автоматах установки из стандартных упаковок, в которых компоненты поставляются заводом-изготовителем, но при единичном и мелкосерийном производстве может применяться ручная установка с помощью вакуумного пинцета или манипулятора, а также автоматизированная установка на полуавтомате (манипуляторе с указателем места установки компонента по программе).

Типичная производительность автоматов начального уровня лежит в пределах 1 500... 5 000 компонентов в час. Типичная производительность серийных автоматов составляет 10... 50 тысяч компонентов в час. Современные высокоскоростные автоматы обладают максимальной производительностью до нескольких сотен тысяч компонентов в час.

В современном оборудовании захват компонентов осуществляется вакуумной головкой. Для захвата тяжелых компонентов применяются специальные насадки. Для примера рассмотрим оборудование цеха поверхностного монтажа. Сборка печатных узлов с поверхностно-монтируемыми элементами производится на двух автоматизированных линиях, скомплектованных фирмой *ASSEMBLEON* и включающих в себя:

- автозагрузчик печатных плат;
- автомат трафаретной печати *ELA* фирмы *DEK*;
- высокоточный автомат установки компонентов *EMERALD*;
- экспедиционный конвейер, где производится дополнительный визуальный контроль;
- конвейерная печь конвекционного оплавления *HOTFLOW5*;
- автоматический разгрузчик печатных плат.

Автомат *DEK ELA* трафаретной печати обеспечивает беспрецедентную автоматизацию для изделий с большим количеством разнообразных компонентов, имея при этом высокие гибкость и точность. Компьютерное управление в сочетании с программированием в среде Windows NT делает автомат удобным в применении. Управление всеми основными параметрами печати осуществляется программно.

Сборочный автомат *EMERALD* устанавливает широкий спектр компонентов. Конструктивно автомат состоит из двух прецизионных установочных головок, имеющих высокоскоростные прецизионные приводы на основе шариковинтовой пары, камеру для обучения и коррекции по реперным знакам, системы технического зрения, обеспечивающих центрирование компонентов «на лету». Максимальная производительность 6 500 элементов в час.

3. Пайка оплавлением. Процесс оплавления припоя, содержащегося в паяльной пасте, выполняется в печах путем нагрева печатной платы с компонентами. Нагрев может осуществляться различными способами: инфракрасный (ИК), конвекционный нагрев и нагрев в паровой фазе. Наиболее широкое распространение получил конвекционный нагрев.

+ 1

Пайка оплавлением выполняется путем изменения температуры по заданному закону, называемому *температурным профилем пайки*.

Печь конвекционного оплавления припоя *HOTFLOW5* предназначена для односторонней или двухсторонней пайки оплавлением поверхностно монтируемых элементов на печатной плате.

*HOTFLOW5* имеет три регулируемые зоны нагрева и одну зону охлаждения, которые позволяют выбрать оптимальный профиль пайки. Передача тепла в печи осуществляется почти полностью конвекцией. Подача воздуха в печь осуществляется с помощью системы *MULTI-JET*, которая позволяет равномерно распределять воздух по всей рабочей зоне печи и обеспечить равномерный нагрев по всей площади платы. Для точного определения тепловых режимов используется система *SHUTTLE*, позволяющая контролировать тепловые зоны печи на экране ПК в графическом виде.

После операции пайки, в зависимости от типа применяемой пасты, плата может подвергаться отмывке и сушке.

**Последовательность операций при применении пайки волной.** При применении пайки волной обычно выполняется следующий набор операций.

1. Нанесение клея. Нанесение клея выполняется с помощью ручного или автоматического дозатора из специальных шприцов, в которых клей поставляется. Клей наносится в области расположения компонентов, монтируемых на поверхность, таким образом, чтобы обеспечить приклейку компонента к плате, но не покрыть клеем контактные площадки.
2. Установка компонентов, монтируемых на поверхность. Установка компонентов производится аналогично установке при применении пайки оплавлением. Точность установки компонента при использовании клея должна быть достаточно высокой, поскольку компоненты фиксируются клеем, и характерного для пайки оплавления самовыравнивания не происходит.
3. Полимеризация клея. Полимеризация клея завершает процесс фиксации компонентов. Обычно полимеризация производится в сушильных шкафах при повышенной температуре и необходимой вентиляции.
4. Установка компонентов в отверстия. Эта группа операций полностью аналогична установке компонентов при применении технологии монтажа в отверстия. Компоненты должны фиксироваться для обеспечения правильности их положения в про-

цессе пайки. При применении смешанной технологии с пайкой волной компоненты, монтируемые в отверстия, устанавливаются со стороны, противоположной компонентам, монтируемым на поверхность.

5. Нанесение флюса. Флюс наносится на поверхность платы со стороны пайки, где установлены поверхностно монтируемые компоненты.
6. Пайка волной. Пайка волной осуществляется аналогично методу монтажа в отверстия. Если микросхема имеет выводы по четырем сторонам, она устанавливается на плату под углом 45°. После выполнения пайки плата может подвергаться отмывке и сушке.

**Лазерная пайка.** Лазерная пайка не относится к групповым методам пайки, поскольку монтаж ведется по каждому отдельному выводу либо по ряду выводов. Однако бесконтактность приложения тепловой энергии позволяет повысить скорость монтажа до 10 соединений в секунду и приблизиться по производительности к пайке в паровой фазе и ИК-излучением.

По сравнению с другими методами лазерная пайка обладает рядом следующих преимуществ. Во время пайки печатная плата и корпуса элементов практически не нагреваются, что позволяет монтировать элементы, чувствительные к тепловым воздействиям. Возможна пайка плат с высокой плотностью компоновки элементов, с размерами контактных площадок до 25 мкм, без образования перемычек на соседние соединения или их повреждение.

При использовании лазерной пайки нет необходимости в предварительном подогреве многослойной печатной платы, что обычно необходимо делать при пайке в паровой фазе для предотвращения расслоения платы. Не требуется также создавать какую-либо специальную газовую среду. Процесс пайки ведется в нормальной атмосфере без применения инертных газов.

**Другие варианты технологии поверхностного монтажа.** Существуют и другие методы сборки печатных узлов, основанные на технологии поверхностного монтажа.

**Комбинированный монтаж** может выполняться в два этапа: сначала производится монтаж поверхностных компонентов с применением пайки оплавлением, затем установка и пайка компонентов, монтируемых в отверстия, вручную, волной или селективной пайкой. Данный метод является в настоящее время наиболее распространенным для сборки узлов по комбинированной технологии.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. На какие этапы можно разделить процесс изготовления изделий радиоэлектроники?
2. Какие технологии изготовления печатных плат вам известны?
3. Какие этапы производства включает в себя комбинированный позитивный метод производства ПП?
4. Какими преимуществами обладает технология гибридных модулей и микросхем?
5. Какие методы пайки применяются в технологии поверхностного монтажа?

## Глава 17

### КОНТРОЛЬ В СБОРОЧНОМ ПРОИЗВОДСТВЕ ПЕЧАТНЫХ ПЛАТ

На всех стадиях сборочно-монтажных операций выполняются операции контроля: входной контроль, операционный контроль, выходной контроль. Большинство операций относятся к сплошному контролю, т. е. проверке подвергаются все модули. Обнаруженные дефекты фиксируются в сопроводительной документации на узел для последующего устранения, для статистического учета и с целью выявления и устранения причин их появления.

#### 17.1. ВИЗУАЛЬНЫЙ КОНТРОЛЬ

*Контроль с помощью оператора* — самый распространенный способ. Оборудование — микроскоп с 2—10-кратным увеличением. Качество контроля зависит от квалификации оператора. Такой контроль применяется в лабораторных условиях или на опытном производстве. В сборочных линиях контроль осуществляют автоматические установки.

**Автоматическая оптическая инспекция (АОИ).** Автоматизированный контроль реализуется в ходе четырех основных этапов технологического процесса: нанесения припойной пасты, позиционирования компонентов, отверждения адгезива («клеящего» вещества) и проверки после пайки.

Автоматическая оптическая инспекция позволяет контролировать:

- нанесение припойной пасты (недостаточное, избыточное, неточное, позиционирование трафарета);
- качество позиционирования компонентов (отсутствие/наличие компонента, точность позиционирования, включая разворот по горизонтали и вертикали, несоответствие полярности



или номера вывода, дефект вывода, наличие посторонних предметов);

- качество паяного соединения (короткое замыкание, непропай, несмачиваемость, излишек или недостаток припоя).

Основой АОИ является формирование изображений объектов и анализ характерных особенностей их элементов. Двухмерное изображение объекта формируется оптическими матрицами. Изображение оцифровывается, и формируется матрица, несущая информацию об объекте. Сформированная картинка может сравниваться с эталонным изображением платы или с информацией о сборке на основании данных CAD-файлов. Такие системы позволяют выполнять 100%-ный контроль плат со скоростью до 150 000 компонентов в час, но чувствительны к смене материала платы и компонентов. Большинство АОИ хорошо обнаруживают дефекты расположения компонентов и с меньшим успехом различают дефекты нанесения припойной пасты или качество пайки.

Оптические системы на основе лазеров могут формировать трехмерное изображение объектов. Они применяются и для двумерного анализа сборок особенно в тех случаях, когда наблюдаемые элементы имеют малую высоту или небольшое различие по контрасту (отверстия, реперные точки).

**Рентгеновские контрольные технологические установки (РКТУ).** Для контроля качества внутренних слоев ПП и качества пайки некоторых типов компонентов применяется анализ изображений, полученных с помощью рентгеновских установок. Изображение внутренних слоев МПП и паяных соединений шариковых выводов корпусов типа BGA, скрытых под днищем микросхемы, может быть получено благодаря высокой проникающей способности рентгеновских лучей и разной способности материалов поглощать рентгеновские кванты.

Рентгеновские лучи позволяют получать изображения с разрешением от 0,5 до единиц микрон. Достаточно хорошо с помощью РКТУ идентифицируются дефекты пайки (непропаи и короткие замыкания), скрытые под корпусами микросхем. С помощью рентгеновского контроля можно обнаружить дефекты типа пустот внутри паяных соединений. Широкое применение рентгеновский контроль нашел в производстве МПП для обнаружения дефектов ширины внутренних проводящих дорожек, расслоения диэлектрика и других. Однако установки весьма дороги, для них характерна низкая скорость контроля, повышенные эксплуатационные расходы.

## 17.2. ЭЛЕКТРИЧЕСКИЙ КОНТРОЛЬ

Предназначен для проверки целостности печатных плат, что включает в себя проверку на обрыв цепи, короткое замыкание, правильность топологии.

В основе электрического контроля лежит наличие контакта в системе зонд — проводник платы — зонд или зонд — проводник платы — компонент — проводник платы — зонд.

При тестировании электрическим методом платы устанавливаются на адаптеры, построенные по принципу «поля контактов». Для обнаружения коротких замыканий и обрывов используется низкое напряжение (10 В). Высоким напряжением (500 В) тестируется изоляция на утечку и пробой. Наличие тестовых контактов в переходных отверстиях позволяет с высокой точностью локализовать обрывы. Тестирование плат при помощи этого метода занимает несколько секунд. Самой ответственной частью тестеров является тестовый контакт, так как именно от качества контактирования зависит достоверность информации. Тестовые контакты содержат подпружиненную контактирующую часть. Для соединения с переходными отверстиями, выводами штырьковых компонентов, тестовыми площадками предусмотрены различные формы контактирующих соединений: коронка, игла, воронка и др. Слабое место в тестерах такого типа — адаптерная часть, индивидуальная для каждой разновидности платы. Учитывая, что номенклатура изделий на больших предприятиях велика, стоимость всех адаптеров может оказаться выше стоимости самой тестовой системы.

Лучшее решение для производства с большой номенклатурой — применение оборудования, работающего по методу «летающих пробников». Тестеры имеют несколько головок с приводами по осям X, Y, Z, на каждой из которых установлен пробник. Головки поочередно контактируют с платой с подачей и измерением сигнала, для перехода от одной платы к другой достаточно изменить программу тестирования. Программы перемещения пробников методом трансляции из систем CAD значительно сокращают время подготовки тестовой обработки. Вместе с тем метод «летающих пробников» не обеспечивает высокой производительности тестирования, хотя цена на оборудование достаточно высока.

**Тестирование многослойных ПП.** Имеет определенные сложности. Обычные способы («поле контактов», «летающие пробники») позволяют найти цепи с имеющимися короткозамкнутыми слоями или проводниками, однако они не определяют их точного местопо-

ложения. Если учесть, что стоимость некоторых МПП достаточно велика, то можно говорить о рентабельности оборудования, позволяющего локализовать и устранять такие дефекты. Для точного определения места межслоевого короткого замыкания применяется оборудование, работающее по методу «векторного поиска». Суть его в том, что на область предполагаемого дефекта подается напряжение питания, после чего с помощью очень точных миллиомметров, микровольтметров и миллиамперметров отслеживается зависимость изменения величины протекающего тока от положения пробника на ПП.

**Платы для высокочастотных схем.** Дорожку в такой плате нельзя рассматривать как простой проводник. В таком проводнике необходимо контролировать волновое сопротивление (импеданс), которое измеряется рефлектометрическим методом. Происходит наблюдение за формой волнового сопротивления линии передачи по всей ее длине, и при этом измеряется коэффициент отражения импульсов с малым временем нарастания. Рефлектометрические приборы представляют собой сложное измерительное оборудование и применяются, как правило, в лабораторных условиях.

**Методы тестирования сборок.** Методы тестирования радиоэлектронных изделий на стадии производства подразделяются на два класса: внутрисхемное и функциональное. Каждый из методов отличается способом контактирования с тестируемым изделием.

*Внутрисхемное тестирование* выполняет проверку отдельных компонентов на плате или фрагментов схем. Применяются методы исключения влияния параллельных цепей. При проверке резистора, например, измеряется именно его сопротивление, а не сопротивление цепи, к которой он подключен. Внутрисхемное тестирование подразделяется, в свою очередь, на аналоговое и цифровое.

При аналоговом внутрисхемном тестировании обычно проверяется:

- наличие коротких замыканий и обрывов;
- номиналы дискретных компонентов (резисторов, конденсаторов, индуктивностей, дискретных полупроводниковых приборов);
- наличие и правильность установки микросхем.

Влияние параллельных цепей исключается установкой блокирующих напряжений, применением метода многопроводного измерения, точным подбором напряжения и частоты тестирования. Этот метод тестирования позволяет обнаружить до 80 % дефектов сборки, поэтому аналоговое внутрисхемное тестирование нередко называют анализом производственных дефектов.

При цифровом внутрисхемном тестировании цифровые микросхемы проверяются на соответствие таблице истинности. Для исключения влияния параллельно установленных микросхем (например, при использовании шинной технологии) на вход тестируемой микросхемы подаются импульсы большого уровня с ограниченной длительностью.

- *Функциональное тестирование* предназначено для проверки работоспособности модуля и, при необходимости, его регулировки и настройки. Контакт с изделием осуществляется обычно через краевой разъем. Тестовое оборудование, применяемое при функциональном тестировании, выполняет:
- подачу питающего напряжения с возможностью изменения его в автоматическом режиме, от минимального до максимально допустимого;
- подачу цифровых и аналоговых входных сигналов в широком диапазоне частот и напряжений;
- измерение параметров выходных сигналов;
- эмуляцию нагрузок;
- обмен данными с тестируемым устройством;
- обработку результатов измерений и вывод их на дисплей и принтер в удобном для пользователя виде;
- накопление и обработку статистической информации.

### 17.3. РЕМОНТ ПЕЧАТНЫХ ПЛАТ

Операция ремонта узлов выполняется вручную, включается в процесс сборки после стадии пайки узла и соответствующей операции контроля. Ремонт узла заключается, как правило, в замене дефектного компонента или корректировки дефектного паяного соединения в соответствии с рекомендациями стандартов на ремонтные операции. Операция ремонта узла должна быть экономически целесообразной, поскольку процесс замены дефектных компонентов на уже собранной плате чрезвычайно трудоемок и чреват внесением дополнительных дефектов. Поэтому должны учитываться многие факторы, в том числе стоимость узла, дефектного компонента, трудозатраты на ремонт и др. Дешевые сборки целесообразнее выбрасывать, нежели ремонтировать.

Демонтаж сложных компонентов поверхностного монтажа является прецизионной операцией из-за высокой плотности монтажа. Тепло, необходимое для отпайки компонента, может оказать воздействие на соседние чувствительные к нагреву компоненты и по-

вредить саму ПП. Учет на стадии проектирования требований по обеспечению ремонтпригодности изделия налагает определенные ограничения на процессы сборки и монтажа и в некоторой степени снижает плотность монтажа.

При демонтаже компонентов в корпусах сложной конфигурации доминирующим способом теплопередачи становится конвекция. Приспособление для демонтажа забракованных компонентов оснащено нагревательными капиллярами для разогрева мест пайки со сменными наконечниками, рассчитанными на различные формы и размеры компонентов. Капилляры с наконечниками сконструированы таким образом, что струя горячего газа (воздуха) направляется на выводы компонента. Удаление дефектного и установка на его место исправного компонента производится с помощью вакуумного пинцета. В ряде случаев используется микроскоп, который обеспечивает контроль точности позиционирования компонента. Типичная операция по исправлению брака может занять до 30 мин и включает в себя следующие этапы:

- подготовка платы к демонтажу компонента (очистка паяных соединений; снятие теплоотвода; защита соседних компонентов; покрытие флюсом концов выводов компонента, припаянных на контактных площадках платы);
- разогрев паяных соединений (разогрев микросборки, разогрев выводов горячим газом);
- снятие компонента со знакоместа с помощью вакуумного пинцета;
- очистка платы, удаление остатков флюса, загрязнений и излишков припоя;
- защита подготовленного знакоместа, если замена компонента откладывается;
- замена компонента (нанесение флюса на концы выводов компонента и места пайки с последующим их облуживанием, позиционирование компонента с помощью вакуумного пинцета, оплавление припоя горячим газом, очистка платы после пайки с целью удаления продуктов разложения флюса).

Исправление брака, в сущности, сводится к повторному выполнению определенной части сборочно-монтажных операций. Необходимо тщательный контроль и управление процессом устранения брака, чтобы исключить возможность повреждения годного (заменяющего бракованный) компонента, а также соседних компонентов и элементов коммутационной платы. Надежной гарантией от проблем, связанных с ремонтом изделий, является обеспечение высокого качества процесса сборки и обязательный контроль процесса монтажа.

Для выполнения ремонтных и дополнительных работ по монтажу можно использовать паяльные станции и конвекционные системы, которые позволяют производить монтаж и демонтаж элементов в различных корпусах, а также многие другие ремонтные работы.

## 17.4. ИСПЫТАНИЯ РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

- Испытания РЭА представляют собой экспериментальное определение количественных и качественных характеристик изделий при различных воздействиях. К основным целям испытания, общим для всех видов РЭА, можно отнести:
- выбор оптимальных конструктивно-технологических решений при создании новых изделий;
- доводку изделий до необходимого уровня качества;
- объективную оценку качества изделий при их постановке на производство, в процессе производства и при техническом обслуживании;
- прогнозирование гарантированного срока службы.

Испытания служат эффективным средством выявления скрытых дефектов материалов и элементов конструкции, не обнаруженных методами технического контроля. По результатам испытаний изделий в производстве разработчик РЭА устанавливает причины снижения качества. Если эти причины установить не удастся, совершенствуют методы и средства контроля изделий и ТП их изготовления.

Испытания на механические воздействия. Механические испытания РЭА проводят в нормальных климатических условиях под электрической нагрузкой или без нее.

Наибольшее влияние на РЭА оказывает сочетание вибрационных нагрузок и одиночных ударов, испытания на воздействия которых проводят в первую очередь. Испытания по определению резонансных частот конструкции допускается проводить на отдельных типах (типоразмерах, типоминалах) изделий, имеющих одинаковую конструкцию. Испытание на проверку отсутствия резонансных частот конструкции изделия в заданном диапазоне частот не проводят, если оно обеспечивается их конструкцией, о чем должно быть указано в технических условиях (ТУ) на изделия.

Испытание на виброустойчивость допускается совмещать с испытанием на вибропрочность, проводя его в начале или в конце

испытаний на вибропрочность. Испытаниям на ударную прочность не подвергают изделия, у которых низшая резонансная частота превышает 1000 Гц. Ударная прочность и (или) устойчивость таких изделий обеспечивается их конструкцией.

Применяемые виды механических испытаний и их последовательность указываются в ПИ и зависят от назначения РЭА, условий эксплуатации, типа производства. Например, в программу определительных испытаний опытного образца и образцов установочной серии обычно включают все виды механических испытаний, а для образцов, изготавливаемых в серийном производстве, — только испытания, предусмотренные в ТУ. Надежная работа РЭА обеспечивается за счет конструктивных запасов по вибропрочности, виброустойчивости, резонансной частоте и другим характеристикам.

Испытание на климатические воздействия. Принята следующая последовательность операций испытания РЭА на климатические воздействия: предварительная выдержка изделий (стабилизация свойств); первоначальные измерения параметров и внешний осмотр; установка изделий в камеры, выдержка их в условиях испытательного режима, измерения параметров; извлечение из камер и выдержка для восстановления свойств изделий (конечная стабилизация свойств); внешний осмотр и заключительные измерения параметров изделий.

Климатические испытания проводят на стадии проектирования РЭА, в серийном производстве для отбраковки потенциально ненадежных изделий и для контроля стабильности производства (периодические испытания). Режимы и условия испытания РЭА устанавливают в зависимости от степени жесткости, которая, в свою очередь, определяется условиями дальнейшей эксплуатации РЭА. Изделия считают выдержавшими испытание, если они во время и после его проведения удовлетворяют требованиям, заданным в ТУ для данного вида испытаний.

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что позволяет контролировать автоматизированная оптическая инспекция?
2. Для чего предназначен электрический контроль?
3. На какие классы подразделяются методы тестирования радиоэлектронных изделий?
4. Какие этапы включает в себя операция по исследованию брака?
5. Что относится к основным целям испытания РЭА?

## Глава 18

### ЕДИНАЯ СИСТЕМА КОНСТРУКТОРСКОЙ ДОКУМЕНТАЦИИ

Проектирование и производство радиоэлектронной аппаратуры, обязательно сопровождается выпуском технической документации. Правила ее оформления сформулированы в ГОСТах.

Единая система конструкторской документации (ЕСКД) — комплекс государственных стандартов, устанавливающих взаимосвязанные нормы и правила по разработке, оформлению и обращению конструкторской документации, разрабатываемой и применяемой на всех стадиях жизненного цикла изделия (при проектировании, разработке, изготовлении, контроле, приемке, эксплуатации, ремонте, утилизации). Соблюдение стандартов обеспечивает:

- возможность обмена конструкторской документацией без ее переоформления;
- оптимальную комплектность конструкторской документации;
- механизацию и автоматизацию обработки конструкторских документов и содержащейся в них информации;
- высокое качество изделий;
- наличие в конструкторской документации требований, обеспечивающих безопасность использования изделий для жизни и здоровья потребителей, окружающей среды, а также предотвращение причинения вреда имуществу;
- возможность расширения, унификации и стандартизации при проектировании изделий и разработке конструкторской документации;
- возможность проведения сертификации изделий;
- сокращение сроков и снижение трудоемкости подготовки производства;
- правильную эксплуатацию изделий;
- оперативную подготовку документации для быстрой переналадки действующего производства;



- упрощение форм конструкторских документов и графических изображений;
- возможность создания единой информационной базы автоматизированных систем (САПР, АСУП и др.);
- гармонизацию с соответствующими международными стандартами;
- возможность информационного обеспечения поддержки жизненного цикла изделия.

Если в состав цифрового устройства входят узлы, имеющие программное обеспечение (микропроцессоры, микроконтроллеры), то, кроме конструкторской, необходимо оформить и программную документацию. Правила ее оформления представлены в Единой системе программной документации (ЕСПД).

## 18.1. ВИДЫ КОНСТРУКТОРСКИХ ДОКУМЕНТОВ

К **конструкторским документам** (КД) относятся графические и текстовые документы, которые в отдельности или в совокупности определяют состав и устройство изделия и содержат необходимые данные для его разработки, изготовления, контроля, приемки, эксплуатации, ремонта и утилизации.

К **графическим документам** относятся чертежи (детали, сборочный, общего вида, теоретический, габаритный, электромонтажный, монтажный, упаковочный), схемы, электронные модели деталей и сборочных единиц, электронная структура изделия.

К **текстовым документам** относятся следующие: пояснительная записка; технические условия; программа и методика испытаний; таблицы; расчеты; эксплуатационные документы; ремонтные документы; инструкции; перечень элементов; спецификация; ведомости спецификаций, ссылочных документов, покупных изделий, разрешения применения покупных изделий, держателей подлинников, технического предложения, эскизного проекта, технического проекта, электронных документов.

Конструкторские документы в зависимости от стадии разработки подразделяются на *проектные* (техническое предложение, эскизный проект и технический проект) и *рабочие* (КД опытного образца и КД серийного производства).

Наименования конструкторских документов в зависимости от способа их выполнения и характера использования приведены в табл. 18.1.

Таблица 18.1

Наименование документа	Определение
Оригиналы	Документы, выполненные на любом материале и предназначенные для изготовления по ним подлинников
Подлинники	Документы, оформленные подлинными установленными подписями и выполненные на любом материале, позволяющем многократное воспроизведение с них копий. Допускается в качестве подлинника использовать оригинал, копию или экземпляр документа, изданного типографским способом, завизированные подлинными подписями
Дубликаты	Копии подлинников, обеспечивающие идентичность воспроизведения подлинника, выполненные на любом материале, позволяющем снятие с них копий
Копии	Документы, выполненные способом, обеспечивающим их идентичность с подлинником (дубликатом) и предназначенные для непосредственного использования при разработке, в производстве, эксплуатации и ремонте изделий

## 18.2. СТАДИИ РАЗРАБОТКИ КД И ЭТАПЫ ВЫПОЛНЕНИЯ РАБОТ ПРИ ПРОЕКТИРОВАНИИ

Последовательность проектирования устройств регламентирована ГОСТ и приведена в табл. 18.2.

**Техническое предложение** содержит технические и технико-экономические обоснования целесообразности разработки документации изделия на основании анализа технического задания заказчика и различных вариантов возможных решений изделий, сравнительной оценки решений с учетом конструктивных и эксплуатационных особенностей разрабатываемого и существующих изделий и патентные исследования.

**Эскизный проект** содержит принципиальные конструктивные решения, дающие общее представление об устройстве и принципе работы изделия, а также данные, определяющие назначение, основные параметры и габаритные размеры разрабатываемого изделия.



Таблица 18.2	
Стадия разработки	Этапы выполнения работ
Техническое предложение	Подбор материалов. Разработка технического предложения. Рассмотрение и утверждение технического предложения
Эскизный проект	Разработка эскизного проекта. Изготовление и испытание макетов (при необходимости). Рассмотрение и утверждение эскизного проекта
Технический проект	Разработка технического проекта. Изготовление и испытание макетов (при необходимости). Рассмотрение и утверждение технического проекта
Рабочая конструкторская документация:	Разработка конструкторской документации, предназначенной для изготовления и испытания опытного образца (партии)
опытного образца (опытной партии) изделия, предназначенного для серийного (массового) или единичного производства (кроме разового изготовления)	Изготовление и предварительные испытания опытного образца (партии). Корректировка конструкторской документации по результатам изготовления и предварительных испытаний опытного образца (партии). Приемочные испытания опытного образца (партии). Корректировка конструкторской документации по результатам приемочных испытаний опытного образца (партии). Для изделия, разрабатываемого по заказу Министерства обороны, при необходимости — повторное изготовление и испытания опытного образца (партии)
серийного (массового) производства	Изготовление и испытание установочной серии. Корректировка конструкторской документации по результатам изготовления и испытания установочной серии, а также оснащения технологического процесса изготовления изделия. Для изделия, разрабатываемого по заказу Министерства обороны, при необходимости — изготовление и испытание головной (контрольной) серии

**Технический проект** содержит окончательные технические решения, дающие полное представление об устройстве разрабатываемого изделия, и исходные данные для разработки рабочей документации.

## 18.3. ВИДЫ ИЗДЕЛИЙ

**Изделием** называется любой предмет или набор предметов, подлежащих изготовлению на предприятии. Изделия, в зависимости от их назначения, подразделяются на изделия основного производства, предназначенные для поставки (реализации), и на изделия вспомогательного производства, предназначенные только для собственных нужд предприятия, изготавливающего их.

ГОСТ устанавливает следующие виды изделий: детали, сборочные единицы, комплексы, комплекты.

Определение видов изделий и их структура приведены в табл. 18.3.

Таблица 18.3	
Вид изделия	Определение
Деталь	Изделие, изготовленное из однородного материала, без применения сборочных операций, например: валик из одного куска металла, литой корпус; печатная плата; отрезок кабеля или провода заданной длины
Сборочная единица	Изделие, составные части которого подлежат соединению между собой на предприятии-изготовителе сборочными операциями (свинчиванием, сочленением, клепкой, сваркой, пайкой, опрессовкой, развальцовкой, склеиванием, сшивкой, укладкой и т. п.), например: микромодуль, сварной корпус, комплект составных частей персонального компьютера: материнская плата, память, жесткий диск и т. д.
Комплекс	<p>Два и более специфицированных изделия, не соединенных на предприятии-изготовителе сборочными операциями, но предназначенных для выполнения взаимосвязанных эксплуатационных функций.</p> <p>Каждое из этих специфицированных изделий, входящих в комплекс, служит для выполнения одной или нескольких основных функций, установленных для всего комплекса, например: вычислительная система; автоматическая телефонная станция.</p> <p>В комплекс, кроме изделий, выполняющих основные функции, могут входить детали, сборочные единицы и комплекты, предназначенные для выполнения вспомогательных функций, например: детали и сборочные единицы, предназначенные для монтажа комплекса на месте его эксплуатации; комплекс запасных частей, укладочных средств, тары и др.</p>

Вид изделия	Определение
Комплект	<p>Два и более изделия, не соединенных на предприятии-изготовителе сборочными операциями и представляющих набор изделий, имеющих общее эксплуатационное назначение вспомогательного характера, например: комплект запасных частей, комплект инструмента и принадлежностей, комплект измерительной аппаратуры, комплект упаковочной тары и т. п.</p> <p>К комплектам также относят изделия, предназначенные для выполнения вспомогательных функций при эксплуатации этих сборочных единиц или деталей, например: осциллограф в комплекте с укладочным ящиком, запасными частями, монтажным инструментом, сменными частями</p>

Различают следующие разновидности изделий: покупные изделия, изделия установочной серии, освоенное изделие, оригинальное (впервые разработанное) изделие, унифицированное изделие (применяемое в КД нескольких изделий), стандартное изделие (примененное по стандарту, определяющему его конструкцию, показатели качества, методы контроля, правила приемки и поставки), типовое, базовое, модернизированное, разъемное и неразъемное изделия, изделия единичного производства, повторяющегося единичного производства, разового изготовления, изделия серийного и массового производства, годное, дефектное, комплектное, новое, устаревшее, технологическое, обслуживаемое изделие (проведение технических обслуживаний предусмотрено в документации), не обслуживаемое изделие, ремонтируемое изделие (проведение ремонтов предусмотрено в документации, ремонтом называется комплекс операций по восстановлению исправного или работоспособного состояния изделий), неремонтируемое изделие, отремонтированное изделие, восстанавливаемое изделие, невосстанавливаемое изделие.

Кроме того, изделия могут быть составными частями других изделий: комплектующее изделие, покупное изделие, кооперированное изделие, взаимозаменяемое изделие, запасная часть, крепежное изделие и т. д.

Существует несколько видов моделей и макетов изделий: модель изделия, модель для испытаний, макет изделия, проектный макет изделия (проектный макет), рабочий макет изделия (рабочий макет).

## 18.4. ПРАВИЛА ОФОРМЛЕНИЯ ГРАФИЧЕСКОЙ ДОКУМЕНТАЦИИ

Основные правила сформулированы в ГОСТ 2.702—2011 «Единая система конструкторской документации. Правила выполнения электрических схем».

**Схема электрическая** — документ, содержащий в виде условных изображений или обозначений составные части цифрового устройства и их взаимосвязи. Подразделяют: структурные, функциональные, принципиальные, соединений, подключения, общие и схемы расположения. Они могут быть выполнены в бумажной и (или) электронной форме.

**Схема электрическая принципиальная** — наиболее подробная схема. Она обязательно показывает все использованные в устройстве элементы и все связи между ними. Схема позволяет полностью воспроизвести устройство.

**Структурная схема** — наименее подробная схема. Она предназначена для отображения общей структуры устройства, т. е. его основных блоков, узлов, частей и главных связей между ними. Из нее должно быть понятно назначение, режимы работы, взаимодействие его частей. Обозначения могут быть довольно произвольными.

**Функциональная схема** — гибрид структурной и принципиальной. Некоторые наиболее простые блоки, узлы, части устройства отображаются на ней, как на структурной схеме, а остальные — как на принципиальной. Функциональная схема дает возможность понять логику работы устройства, все его отличия от других подобных устройств.

В технической документации обязательно приводятся структурная или функциональная и принципиальная схемы. В научных статьях и книгах чаще всего ограничиваются структурной или функциональной схемой, приводя принципиальные схемы только некоторых узлов.

На структурной схеме изображают все основные функциональные части изделия (элементы, устройства и функциональные группы) и основные взаимосвязи между ними. Функциональные части на схеме изображают в виде прямоугольника или условных графических обозначений. Графическое построение схемы должно давать наиболее наглядное представление о последовательности взаимодействия функциональных частей в изделии. На линиях взаимосвязей рекомендуется стрелками обозначать направление хода про-

цессов, происходящих в изделии. При изображении функциональных частей в виде прямоугольников наименования, типы и обозначения рекомендуется вписывать внутрь прямоугольников.

На функциональной схеме изображают функциональные части изделия (элементы, устройства и функциональные группы), участвующие в процессе, иллюстрируемой схемой, и связи между этими частями. Функциональные части и связи между ними на схеме изображают в виде условных графических обозначений, установленных в стандартах ЕСКД. Отдельные функциональные части допускается изображать в виде прямоугольников. Графическое построение схемы должно давать наиболее наглядное представление о последовательности процессов, иллюстрируемых схемой. На схеме должны быть указаны: для каждой функциональной группы — обозначение, присвоенное ей на принципиальной схеме, и (или) ее наименование; если функциональная группа изображена в виде условного графического обозначения, то ее наименование не указывают; для каждого устройства, изображенного в виде прямоугольника, — позиционное обозначение, присвоенное ему на принципиальной схеме, его наименование и тип; для каждого устройства, изображенного в виде условного графического обозначения, — позиционное обозначение, присвоенное ему на принципиальной схеме, его тип; для каждого элемента — позиционное обозначение, присвоенное ему на принципиальной схеме и (или) его тип.

На принципиальной схеме изображают все электрические элементы или устройства, необходимые для осуществления и контроля в изделии заданных электрических процессов, все электрические связи между ними, а также электрические элементы (соединители, зажимы и т. п.), которыми заканчиваются входные и выходные цепи. Элементы и устройства, условные графические обозначения которых установлены в стандартах ЕСКД, изображают на схеме в виде этих условных графических обозначений.

Все узлы, блоки, части, элементы, микросхемы показываются в виде прямоугольников с соответствующими надписями. Все связи между ними, все передаваемые сигналы изображаются в виде линий, соединяющих эти прямоугольники. Входы и входы-выходы должны быть по возможности расположены на левой стороне прямоугольника, выходы — на правой стороне. Выводы и связи питания, как правило, не прорисовывают, если не используются нестандартные включения. Выводы неиспользованных элементов изображают короче. При необходимости на поле схемы помещают соответствующие пояснения. Допускается слияние нескольких

Таблица 18.4			
Позиционное обозначение	Наименование	Количество	Примечание
D1	Микросхема К155ТМ2 Бко.346.006ТУ1	1	—
D2	Микросхема К155ЛАЗ. Бко.348.006ТУ1	1	—
R1, R2	МЛТ-0,25-430 Ом±10 % ГОСТ...	2	—
SA1	Переключатель АБВП	1	—
C1,C2	Конденсатор КМ-За-Н30-0,22... ТУ	2	—

электрически не связанных линий взаимосвязи в шину, но при подходе к контактам каждую линию взаимосвязи изображают отдельной линией. При этом каждую линию помечают в месте слияния, а при необходимости — на обоих концах. На схеме рекомендуется указывать технические характеристики функциональных частей (рядом с графическими обозначениями или на свободном поле схемы).

Позиционные обозначения элементам следует присваивать в пределах изделия, начиная с единицы, в пределах группы элементов, которым на схеме присвоено одинаковое буквенное позиционное обозначение, например  $R_1, R_2, R_3$  и т.д.,  $C_1, C_2, C_3$  и т.д. Порядковые номера следует присваивать в соответствии с последовательностью расположения элементов или устройств на схеме сверху вниз в направлении слева направо (можно в зависимости от размещения элементов в изделии или направления прохождения сигналов).

Данные об элементах следует записывать в перечень элементов, оформляемый в виде табл. 18.4.

Около условных графических обозначений номиналов резисторов и конденсаторов (рис. 18.1) допускается применять упрощенный способ обозначения единиц измерений:

- для резисторов:
  - ✓ от 0 до 999 Ом — без указания единиц измерения;
  - ✓ от  $1 \cdot 10^3$  до  $999 \cdot 10^3$  Ом — в килоомах с обозначением единицы измерения строчной буквой к;

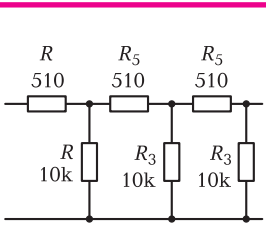


Рис. 18.1. Обозначение резисторов

- ✓ от  $1 \cdot 10^6$  до  $999 \cdot 10^6$  Ом — в мегаомах с обозначением единицы измерения прописной буквой М;
- ✓ свыше  $1 \cdot 10^9$  Ом — в гигаомах с обозначением единицы измерения прописной буквой Г;
- для конденсаторов:
  - ✓ от 0 до  $9999 \cdot 12^{-12}$  Ф — в пикофарадах без указания единицы измерения,
  - ✓ от  $1 \cdot 10^{-8}$  до  $9999 \cdot 10^{-6}$  Ф — в микрофарадах с обозначением единицы измерения строчными буквами мк.

На схеме следует указывать обозначения выводов элементов, нанесенных на изделие или установленные в их документации; характеристики входных и выходных цепей изделия (частоту, напряжение, силу тока, сопротивление, индуктивность и т. д.), а также параметры, подлежащие измерению на контрольных контактах, гнездах и т. д. Характеристики входных и выходных цепей изделия, а также адреса их внешних подключений рекомендуется записывать в таблицы.

При изображении на принципиальной схеме элементов, параметры которых подбирают при регулировании, на схеме и в перечне элементов проставляют звездочки (например,  $R_1^*$ ), а на поле схемы помещают сноску: «Подбирают при регулировании».

На **схеме соединений** следует изображать все устройства и элементы, входящие в состав изделия, их входные и выходные элементы (соединители, платы, зажимы и т. д.), а также соединения между этими устройствами и элементами. Расположение графических обозначений устройств и элементов, входных и выходных выводов на схеме соединений должно примерно соответствовать их действительному размещению в изделии. Провода, группы проводов, жгуты и кабели должны быть показаны на схеме соединений отдельными линиями, их следует нумеровать отдельно. При этом провода, входящие в жгут, нумеруют в пределах жгута, а жилы кабеля — в пределах кабеля. На схеме соединений следует указывать марку, сечение и, при необходимости, расцветку; количество и сечение жил и, при необходимости, количество занятых жил.

Если на схеме соединений не указаны места присоединений, то данные о проводах, жгутах и кабелях и адреса их соединений сводят в «Таблицу соединений».

На **схеме подключения** должны быть изображены изделие, его входные и выходные элементы (соединители, зажимы и т. п.) и подводимые к ним концы проводов и кабелей (многожильных проводов, электрических шнуров) внешнего монтажа, около которых

помещают данные о подключении изделия (характеристики внешних цепей и (или) адреса).

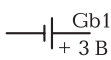

На **общей схеме** изображают устройства и элементы, входящие в комплекс, а также провода, жгуты и кабели, соединяющие эти устройства и элементы. Устройства и элементы на схеме изображают в виде прямоугольников. Допускается элементы изображать в виде условных графических обозначений или упрощенных внешних очертаний, а устройства — в виде упрощенных внешних очертаний. Расположение графических обозначений устройств и элементов на схеме должно примерно соответствовать действительному размещению элементов и устройств в изделии.

На **схеме расположения** изображают составные части изделия в соответствии с их действительным размещением в конструкции. Составные части изделия изображают в виде упрощенных внешних очертаний или условных графических изображений. Провода, группы проводов, жгуты и кабели (многожильные провода, электрические шнуры) изображают в виде отдельных линий. При выполнении схемы расположения допускается применять различные способы построения (аксонометрию, план, условную развертку, разрез конструкции и т. д.).

## 18.5. ОБОЗНАЧЕНИЯ ЭЛЕМЕНТОВ НА ЧЕРТЕЖАХ И СХЕМАХ

Любой элемент обозначается на схеме одной или двумя буквами (первая обязательно прописная), и порядковым номером на конкретной схеме. Например  $R25$  обозначает, что это резистор ( $R$ ), и на изображенной схеме — 25-й по счету. При модульном (блочном) построении аппаратуры, элементы каждого блока имеют свои порядковые номера. Условные графические обозначения некоторых элементов представлены в табл. 18.5.

Таблица 18.5

Наименование элемента	Графическое обозначение (варианты)
Элемент питания	
Узел	

Продолжение табл. 18.3

Наименование элемента	Графическое обозначение (варианты)
Контакт	X2
Штекер	XP2
Выключатель	S7
Общий провод	
Заземление	
Лампа накаливания	VL2
Резистор	R12
Переменный резистор	R2
Подстроечный резистор	R10
Терморезистор	R5
Конденсатор	C6
Конденсатор электролитический	C13
Подстроечный конденсатор	C4
Переменный конденсатор	C23
Катушка индуктивности	L1
Подстроечная катушка индуктивности	L1

Окончание табл. 18.3

Наименование элемента	Графическое обозначение (варианты)
Трансформатор	
Диод	VD1, VD5, CR8
Светодиод	VD7
Фотодиод	VD2
$n-p-n$ -транзистор	VT15
$p-p-p$ -транзистор	VT17
Транзистор полевой	n-канальный T4
Транзистор полевой со встроенным $n$ -каналом	VT18
Транзистор полевой с индуцированным $n$ -каналом	VT1

Микросхемы изображаются в виде прямоугольников, входы располагают слева, выходы — справа. Для обозначения полярности сигнала на схемах используется правило: если сигнал отрицательный, то перед его названием ставится знак «минус» (например,  $-WR$  или  $-OE$ ) или же над названием сигнала ставится черта. Для названий сигналов обычно используются латинские буквы, представляющие собой сокращения английских слов (например,  $WR$  — сигнал записи (от англ. *write* — писать)). Инверсия сигнала обознача-



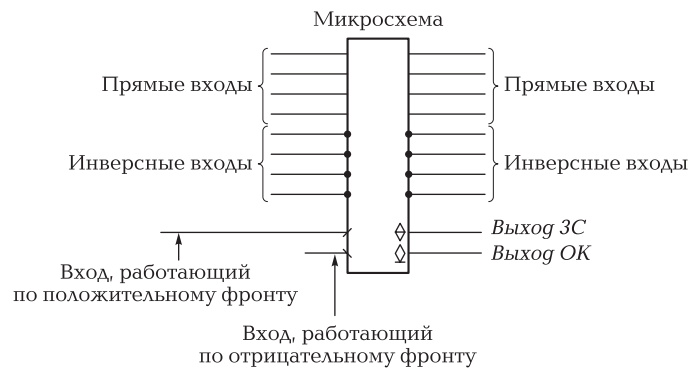


Рис. 18.2. Обозначение входов и выходов

ется кружочком на месте входа или выхода (рис. 18.2). Если микросхема выполняет функцию по фронту входного сигнала, то на месте входа ставится косая черта, причем наклон вправо или влево определяется тем, положительный или отрицательный фронт используется в данном случае.

На структурных и функциональных схемах шины обозначаются толстыми линиями или двойными стрелками. Количество сигналов, входящих в шину, указывается рядом с косой чертой, пересекающей шину. На принципиальных схемах шина тоже обозначается толстой линией, а входящие в шину и выходящие из шины сигналы изображаются в виде перпендикулярных к шине тонких линий с указанием их номера или названия (рис. 18.3).

При изображении микросхем указывается выполняемая ими функция (обычно в центре сверху), сокращенные названия входных и выходных сигналов (рядом с соответствующим выводом). Изображение микросхемы иногда делят на три вертикальные поля. Левое поле относится к входным сигналам, правое — к выходным сигналам. В центральном поле помещается название микросхемы

и символы ее функциональных особенностей. В табл. 18.6 приведены некоторые обозначения сигналов и функций микросхем. Цифровая микросхема обозначается буквами *DD* (от англ. *digital* — цифровой) с соответствующим номером, напри-

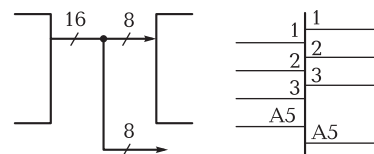


Рис. 18.3. Обозначение шин

Таблица 18.6

Обозначение	Название	Назначение
<i>A</i>	<i>Address</i>	Адресные разряды
<i>BF</i>	<i>Buffer</i>	Буфер
<i>C</i>	<i>Clock</i>	Тактовый сигнал (строб)
<i>CE</i>	<i>Clock Enable</i>	Разрешение тактового сигнала
<i>CT</i>	<i>Counter</i>	Счетчик
<i>CS</i>	<i>Chip Select</i>	Выбор микросхемы
<i>D</i>	<i>Data</i>	Разряды данных, данные
<i>DC</i>	<i>Decoder</i>	Дешифратор
<i>EZ</i>	<i>Enable Z-state</i>	Разрешение третьего состояния
<i>G</i>	<i>Generator</i>	Генератор
<i>I</i>	<i>Input</i>	Вход
<i>I/O</i>	<i>Input/Output</i>	Вход/Выход
<i>OE</i>	<i>Output Enable</i>	Разрешение выхода
<i>MS</i>	<i>Multiplexer</i>	Мультиплексор
<i>Q</i>	<i>Quit</i>	Выход
<i>R</i>	<i>Reset</i>	Сброс (установка в нуль)
<i>RG</i>	<i>Register</i>	Регистр
<i>S</i>	<i>Set</i>	Установка в единицу
<i>SUM</i>	<i>Summator</i>	Сумматор
<i>T</i>	<i>Trigger</i>	Триггер
<i>TC</i>	<i>Terminal Count</i>	Окончание счета
<i>Z</i>	<i>Z-state</i>	Третье состояние выхода

мер, *DD1*, *DD20.1*, *DD38.2* (после точки указывается номер элемента или узла внутри микросхемы).

Примеры оформления чертежей и документации приведены на рис. 18.4... 18.7.

Перф. примен. ВИАМ.ХХХХХХХ.ХХХ	Поз. обозначение	Наименование	Кол.	Примечание
	A1	Контроллер	1	
	A2	Плата питания	1	
	A3	Плата питания ТЭМ	1	
	A4	Плата управления ТЭМ	1	
	A5	Синтезатор частоты LC-06 РЭРД 467874.006 ТУ	1	
	A6	Плата питания УМ	1	
Справ. №	B1	Датчик температуры AD 221000 ST	1	
	L1	Дроссель ВИАМ.671342.098	1	
	X1	Вилка СНЦ144-13/11В011-NWP ЦСНК.430421.008 ТУ	1	
	X2	Вилка ОНЦ-БС-1-19/18-В1-2-8 ВР0.364.030 ТУ	1	
	X3	Вилка СНЦ144/5/15В011-NWP ЦСНК.430421.008 ТУ	1	
	X4-X6	Вилка СР-50-154 ФВ ВР0.364.030 ТУ	3	
	X7	Вилка СНЦ144-26/17В011-NWP ЦСНК.430421.008 ТУ	1	
	X8	Вилка ОНЦ-БС-1-19/18-В1-3-8 ВР0.364.030 ТУ	1	
	1	Лепесток 1-3-4,3к22-07 ГОСТ 22375-77	1	
	Взам. инф. №	Примечание- Допускается использовать датчик температуры AD 22100 STZ "Analog Devices"		
Подп. и дата				
Инф. № подл.	Разраб.		Лит.	
	Пров.		Лист	
Инф. № подл.	Н. контр.		Листов	
	Утв.		1	
Электронный модуль				
Перечень элементов				

Копировал

Формат А4

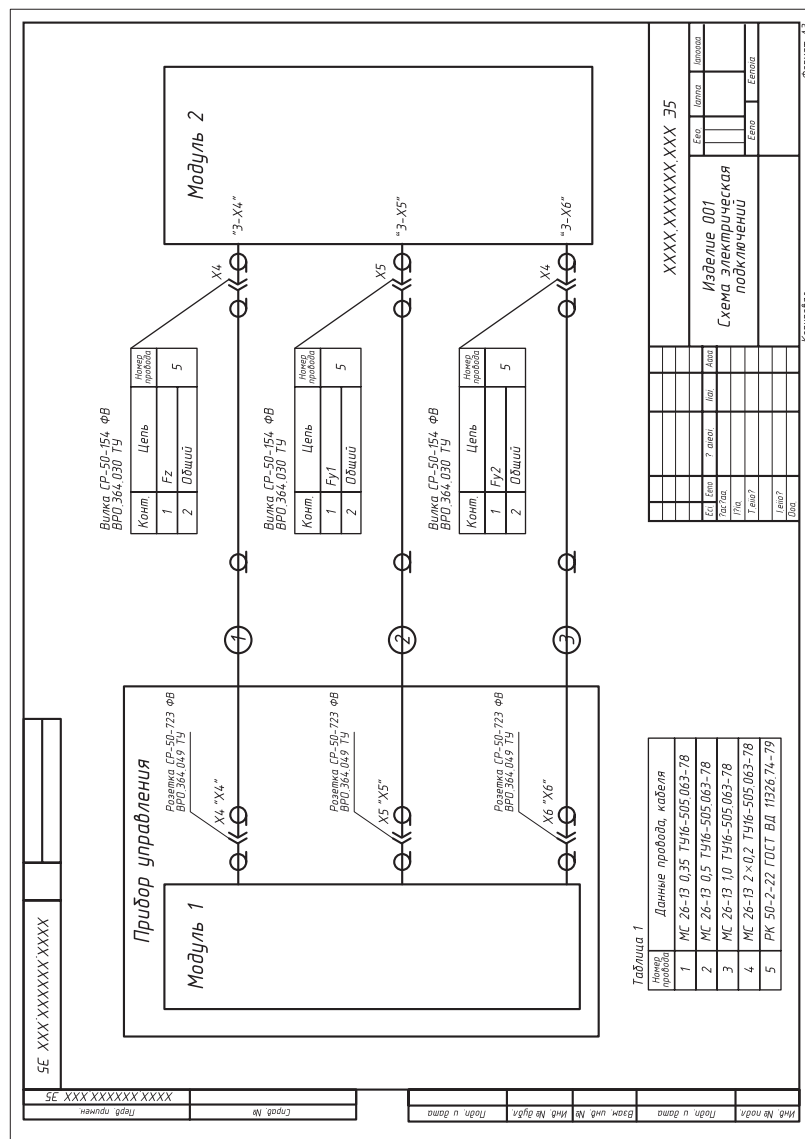


Рис. 18.6. Пример схемы подключений

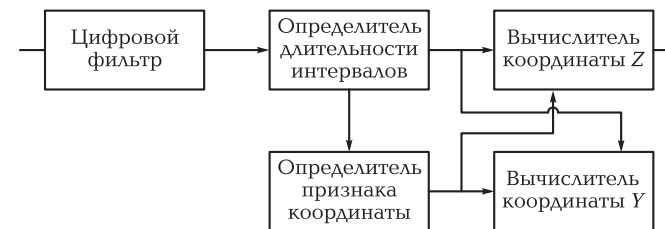


Рис. 18.7. Пример структурной схемы

## 18.6. ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ ЦИФРОВЫХ УСТРОЙСТВ

Испытание любых изделий, в том числе и цифровых устройств, и приемка изготовленной продукции регламентируется ГОСТ 15.309—98 «Система разработки и постановки продукции на производство. Испытания и приемка выпускаемой продукции. Общие положения», который устанавливает основные положения по проведению испытаний и приемки продукции серийного (массового) производства, выпускаемой предприятиями независимо от их формы собственности (далее — изготовители (поставщики)) и предназначенной для поставки или непосредственной продажи потребителю (заказчику).

Для контроля качества и приемки продукции производят приемосдаточные и периодические испытания.

Средства измерений также подвергают испытаниям в соответствии с требованиями ГОСТа. Они должны быть поверены и аттестованы по ГОСТу.

Для оценки эффективности и целесообразности внесения изменений в конструкцию выпускаемой продукции и (или) технологию ее изготовления, которые могут повлиять на технические характеристики продукции, проводят типовые испытания.

**Приемосдаточные испытания** проводят в целях контроля соответствия продукции требованиям стандартов, а также контрольному образцу или образцу-эталоны для определения возможности приемки продукции. Образец-эталон — образец продукции, предназначенный для сравнения с ним единиц продукции при изготовлении, испытаниях, приемке и поставке. Испытания проводят с применением сплошного или выборочного контроля.

**Периодические испытания** проводят для периодического подтверждения качества продукции и стабильности технологического процесса в целях подтверждения возможности продолжения изготовления продукции по действующей конструкторской и технологической документации.

В документах, по которым проводят испытания, устанавливают: требования к продукции, подлежащие контролю; категории и виды испытаний, включая состав проверок, их последовательность; планы контроля; методы испытаний, режимы испытаний; требования к средствам испытаний; требования по количеству единиц продукции, отбираемых для испытаний; порядок обработки данных, полученных при испытаниях, и критерии принятия решений по ним, а также порядок оформления и представления результатов испытаний.

Категории испытаний по составу могут включать в себя один или несколько видов или групп испытаний (механические, электрические, климатические, на надежность и др.) и (или) видов контроля (визуальный, измерительный и др.) и проводиться в один или несколько этапов испытаний.

Результаты испытаний считают положительными, а продукцию — выдержавшей испытания, если она испытана в объеме и последовательности, которые установлены для данной категории испытаний в стандартах на продукцию, а результаты подтверждают соответствие испытываемых единиц продукции заданным требованиям.

Результаты испытаний оформляют протоколом испытаний.

**Приемка продукции** — процесс проверки соответствия продукции требованиям, установленным в стандартах, КД, ТУ, договоре на поставку, изготовленной для ее поставки заказчику или непосредственной продажи покупателю.

Задача испытаний заключается в подтверждении соответствия качества продукции установленным требованиям и в демонстрации этого соответствия потребителю. Увеличение объема испытаний повышает достоверность их результатов, но приводит к росту затрат на изготовление продукции. Однако при уменьшении объема и интенсивности испытаний уменьшается достоверность их результатов, что может привести к еще более значительному росту затрат изготовителя на исправление обнаруженных потребителем дефектов, замене дефектной продукции, компенсации причиненного потребителю ущерба, уплате штрафов, потере фирмой репутации и доверия к качеству ее продукции. Поэтому объем и трудоемкость испытаний должны быть оптимальными.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое ЕСКД?
2. Что обеспечивает соблюдение стандартов?
3. Перечислите виды конструкторских документов.
4. Какова последовательность проектирования устройств?
5. Что называют изделием? Перечислите известные вам виды изделий.
6. Что такое приемка продукции? Какие испытания производят для контроля качества и приемки продукции?
7. Перечислите виды схем, которые содержатся в правилах выполнения электрических схем ЕСКД (ГОСТ 2.702—2011).

ЕДИНАЯ СИСТЕМА  
ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

При проектировании микропроцессорных систем разработчик, кроме аппаратной части, разрабатывает и ее программное обеспечение, что необходимо выполнить в соответствии с определенными правилами. Единая система программной документации — комплекс государственных стандартов, устанавливающих правила разработки, оформления и обращения программ и программной документации, что обеспечивает возможность:

- унификации программных изделий для взаимного обмена программами и применения ранее разработанных программ в новых разработках;
- снижения трудоемкости и повышения эффективности разработки, сопровождения, изготовления и эксплуатации программных изделий;
- автоматизации изготовления и хранения программной документации.

19.1. ПРАВИЛА ВЫПОЛНЕНИЯ СХЕМ  
АЛГОРИТМОВ, ПРОГРАММ,  
ДАННЫХ И СИСТЕМ

Схемы алгоритмов, программ, данных и систем (схемы) оформляются в соответствии с ГОСТ 19.701 — 90 «Единая система программной документации. Схемы алгоритмов программ, данных и систем. Обозначения условные и правила выполнения». Различают: схемы данных, схемы программ, схемы работы системы, схемы взаимодействия программ, схемы ресурсов системы.

+1

**Схема данных** отображает путь данных при решении задач и определяет этапы обработки, а также различные применяемые носители данных.

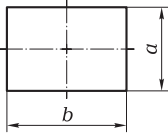
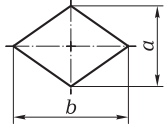
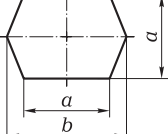
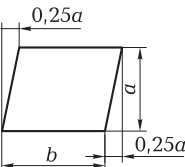
**Схема программ** отображает последовательность операций в программе.

**Схема работы системы** отображает управление операциями и поток данных в системе.

**Схема взаимодействия программ** отображает путь активации программ и взаимодействий с соответствующими данными.

**Схема ресурсов системы** отображает конфигурацию блоков данных и обрабатывающих блоков, которая требуется для решения задачи или набора задач.

В табл. 19.1 приведены графические обозначения символов.

Таблица 19.1		
Название символа	Обозначение и размеры в мм	Функция
Процесс		Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды или группу команд, изменяющих программу
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)



Название символа	Обозначение и размеры в мм	Функция
Документ		Ввод-вывод данных, носителем которых служит бумага
Неавтономная память		Ввод-вывод данных в случае использования запоминающего устройства, управляемого непосредственно процессором
Запоминающее устройство с прямым доступом		Символ отображает данные, хранящиеся в ЗУ прямым доступом.
Соединитель		Указание связи между прерванными линиями потока, связывающими символами
Пуск-останов (терминатор)		Начало, конец, прерывание процесса обработки данных или выполнения программы
Комментарий		Связь между элементом схемы и пояснением

## 19.2. ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ

В состав **программной документации** входят: техническое задание, спецификация, текст программы, описание программы, программа и методика испытаний, пояснительная записка.

В минимальный комплект входят текст программы и описание программы.

Рекомендуемая структура программного документа «Текст программы»: лист утверждения, титульный лист, аннотация (необязательна), содержание (необязательно), основная часть, регистрация изменений.

Основное, чем требуется руководствоваться при создании этого документа, — это то, что текст программы должен быть удобочитаемым. Текст каждого программного файла начинается с «шапки», в которой указывается: наименование программы, автор, дата создания программы, номер версии, дата последней модификации. Обязательными являются комментарии.

Рекомендуемая структура программного документа «Описание программы»: лист утверждения, титульный лист, аннотация, содержание, основная часть.

В основной части должны быть: общие сведения (обозначение и наименование программы; программное обеспечение, необходимое для функционирования программы; языки программирования, на которых написана программа); функциональное назначение (классы решаемых задач; назначение программы; сведения о функциональных ограничениях на применение); описание логической структуры (алгоритм программы; используемые методы; структура программы с описанием функций составных частей и связи между ними; связи программы с другими программами); используемые технические средства; вызов и загрузка (способ вызова программы с соответствующего носителя данных; входные точки в программу); входные данные (характер, организация и предварительная подготовка входных данных; формат, описание и способ кодирования входных данных); выходные данные (характер и организация выходных данных; формат, описание и способ кодирования выходных данных); регистрация изменений.

Испытание программ производится в соответствии с ГОСТ 19.301 — 2000 «Единая система программной документации. Программа и методика испытаний. Требование к содержанию и оформлению», который содержит номенклатуру показателей качества

программных средств и методические указания по определению количественных значений показателей качества.

Документ «Программа и методика испытаний» содержит следующие разделы:

- **объект испытаний** — наименование, область применения и обозначение испытываемой программы;
- **цель испытаний** — цель проведения испытаний;
- **требования к программе** — требования, подлежащие проверке во время испытаний и заданные в техническом задании на программу;
- **требования к программной документации** — состав программной документации, предъявляемой на испытания;
- **средства и порядок испытаний** — технические и программные средства, используемые во время испытаний, а также порядок проведения испытаний;
- **методы испытаний** — описания используемых методов испытаний. В методах испытаний должны быть приведены описания проверок с указанием результатов проведения испытаний (перечней тестовых примеров, контрольных распечаток тестовых примеров и т. п.).

В документе содержится описание того, что и как необходимо сделать, чтобы убедить Заказчика в правильности работы программы. Фактически, этот документ является определяющим для приемосдаточных испытаний.

### 19.3. ЭКСПЛУАТАЦИОННЫЕ ДОКУМЕНТЫ. РУКОВОДСТВО ОПЕРАТОРА

К **эксплуатационным документам** относятся: ведомость эксплуатационных документов, формуляр, описание применения, руководство системного программиста, руководство программиста, руководство оператора.

Руководство оператора должно содержать следующие разделы:

- **назначение программы** — сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации;
- **условия выполнения программы** — состав аппаратных и программных средств;
- **выполнение программы** — последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и за-

вершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды;

- **сообщения оператору.**

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое ЕСПД?
2. Перечислите схемы, содержащиеся в ЕСПД (ГОСТ 19.701—90).
3. Что входит в состав программной документации?
4. Какие разделы содержит документ «Программа и методика испытаний»?
5. Что относится к эксплуатационным документам?
6. Какие разделы должны содержаться в руководстве оператора?

## Список литературы

1. Амосов В. В. Схемотехника и средства проектирования цифровых устройств : учеб. пособие / В. В. Амосов. — СПб. : БХВ — Санкт-Петербург, 2007.
2. Бабич Н. П. Компьютерная схемотехника. Методы построения и проектирования : учеб. пособие / Н. П. Бабич. — Киев : МК-Пресс, 2004.
3. Волгов В. А. Детали и узлы радиоэлектронной аппаратуры / В. А. Волгов. — 2-е изд., перераб. и доп. — М. : Энергия, 2007.
4. Гаврилов С. А. Искусство схемотехники. Просто о сложном / С. А. Гаврилов. — СПб. : Наука и техника, 2011.
5. Дрожжин И. В. Автоматизация конструирования и производства цифровых устройств : учеб. пособие / И. В. Дрожжин, М. Б. Никифоров. — Рязань : РИЦ РГРТУ, 2014.
6. Елесина С. И. Документальное сопровождение разработки и производства радиоэлектронной аппаратуры : учеб. пособие / С. И. Елесина, М. Б. Никифоров. — Рязань : РИЦ РГРТУ, 2014.
7. Ивченко В. Г. Конструирование и технология ЭВМ. Конспект лекций / В. Г. Ивченко. — Таганрог : ТГРУ, 2001.
8. Кистрин А. В. Синтезированный процессор на основе ПЛИС. // Методические указания к курсовому проекту. — Рязань, РИЦ РГРТУ, 2014.
9. Китаев Ю. В. Основы цифровой техники : учеб. пособие / Ю. В. Китаев. — СПб. : СПбГУ ИТМО, 2007.
10. Конструкторско-технологическое проектирование электронной аппаратуры : учебник / [К. И. Билибин, А. И. Власов и др.] / под ред. В. А. Шахнова. — М. : Изд-во МГТУ им. Н. Э. Баумана, 2002.
11. Лаврентьев Б. Ф. Схемотехника электронных средств : учеб. пособие / Б. Ф. Лаврентьев. — М. : Издательский дом «Академия», 2010.
12. Лазутин Ю. Д. Технология электронных средств : учебник / Ю. Д. Лазутин, В. П. Корячко, В. В. Сускин. — М. : Изд-во МГТУ им. Н. Э. Баумана, 2013.
13. Мылов Г. В. Методологические основы автоматизации конструкторско-технологического проектирования гибких многослойных печатных плат / Г. В. Мылов, А. И. Таганов. — М. : Горячая линия-Телеком, 2014.
14. Новиков Ю. В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования / Ю. В. Новиков. — М. : Мир, 2001.
15. Соломахо В. Л. Справочник конструктора-приборостроителя / В. Л. Соломахо. — М. : Высш. шк., 2008.
16. Тупик В. А. Технология и организация производства радиоэлектронной аппаратуры / В. А. Тупик. — СПб. : СПбГЭТУ «ЛЭТИ», 2004.

17. Угрюмов Е. П. Цифровая схемотехника / Е. П. Угрюмов. — СПб. : БХВ — Санкт-Петербург, 2000.

18. Уэйкерли Дж. Ф. Проектирование цифровых устройств : учеб. пособие / Дж. Ф. Уэйкерли. — М. : Постмаркет, 2002.

## Нормативные документы

1. ГОСТ 19.401—78. Единая система программной документации. Текст программы. Требования к содержанию и оформлению.
2. ГОСТ 19.402—78. Единая система программной документации. Описание программы.
3. ГОСТ 19.505—79. Единая система программной документации. Руководство оператора. Требования к содержанию и оформлению.
4. ГОСТ 19.701—90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
5. ГОСТ 2.001—93. ЕСКД. Общие положения.
6. ГОСТ Р 15.201—2000. Система разработки и постановки продукции на производство.
7. ГОСТ 19.301—2000. Единая система программной документации. Программа и методика испытаний. Требования к содержанию и оформлению.
8. ГОСТ 2.051—2006. ЕСКД. Электронные документы. Общие положения.

## Оглавление

Уважаемый читатель! .....	3
Список основных обозначений и сокращений .....	4
Введение.....	6
<b>Глава 1. Арифметические и логические основы цифровой техники .....</b>	<b>8</b>
1.1. Системы счисления.....	8
1.2. Виды двоичных кодов.....	13
1.3. Выполнение арифметических операций в двоичной системе счисления .....	17
1.4. Основы алгебры логики.....	18
1.5. Анализ базовых логических элементов.....	20
<b>Глава 2. Элементная база цифровых вычислительных устройств.....</b>	<b>28</b>
2.1. Логические элементы на биполярных транзисторах .....	30
2.2. Логические элементы на полевых транзисторах .....	36
2.3. Характеристики и параметры логических элементов.....	38
2.4. Программируемые логические интегральные схемы.....	42
<b>Глава 3. Системы автоматизированного проектирования     MAX+ plus II .....</b>	<b>47</b>
3.1. Проектирование устройства, заданного в виде схемы.....	48
3.2. Описание аппаратных средств на языке <i>Verilog HDL</i> .....	57
<b>Глава 4. Анализ и синтез комбинационных схем .....</b>	<b>69</b>
4.1. Анализ комбинационных схем.....	69
4.2. Синтез комбинационных схем .....	72
4.2.1. Синтез комбинационной схемы <i>ks3</i> .....	72
4.2.2. Синтез формирователя признака числа.....	79
<b>Глава 5. Проектирование логических устройств     комбинационного типа .....</b>	<b>84</b>
5.1. Полусумматор.....	84
5.2. Инкрементор .....	87
5.3. Сумматор.....	89
5.4. Параллельный сумматор с последовательным переносом .....	94

5.5. Мультиплексоры.....	96
5.6. Дешифраторы .....	100
5.7. Компараторы кодов.....	103
5.8. АЛУ комбинационного типа.....	105

<b>Глава 6. Триггеры.....</b>	<b>109</b>
6.1. Асинхронный <i>RS</i> -триггер с прямыми установочными входами.....	110
6.2. Асинхронный <i>RS</i> -триггер с инверсными установочными входами.....	112
6.3. Синхронный <i>RS</i> -триггер .....	114
6.4. Статический <i>D</i> -триггер.....	116
6.5. <i>D</i> -триггер с динамическим управлением .....	118
6.6. <i>JK</i> -триггер .....	120
6.7. Счетный триггер.....	123
6.8. Комбинированные триггеры в ПЛИС <i>Altera</i> .....	124
6.9. Описание триггеров на языке <i>Verilog</i> .....	126

<b>Глава 7. Регистры.....</b>	<b>129</b>
7.1. Параллельный регистр .....	129
7.2. Сдвигающие регистры .....	131
7.3. Универсальный сдвигающий регистр .....	134
7.4. Регистровая память в процессорах.....	135

<b>Глава 8. Счетчики .....</b>	<b>139</b>
8.1. Асинхронные счетчики.....	141
8.2. Синхронные счетчики.....	142
8.3. Многофункциональные счетчики в ПЛИС .....	143
8.4. Счетчики с произвольным модулем .....	144
8.5. Описание счетчиков на языке <i>Verilog</i> .....	145
8.6. Распределитель импульсов .....	147

<b>Глава 9. Запоминающие устройства.....</b>	<b>150</b>
9.1. Постоянные запоминающие устройства.....	152
9.2. Оперативные запоминающие устройства .....	155
9.3. Запоминающие устройства в ПЛИС.....	157

<b>Глава 10. Синтез конечных автоматов .....</b>	<b>160</b>
10.1. Пример синтеза конечного автомата.....	161
10.2. Синтез реверсивного счетчика по модулю 3.....	166
10.3. Описание конечных автоматов на языке <i>Verilog</i> .....	168

<b>Глава 11. Микропроцессорные системы.....</b>	<b>171</b>
11.1. Архитектуры микропроцессорных систем .....	171
11.2. Виды микропроцессорных систем и их физическая реализация .....	173
11.3. Структуры процессоров .....	175

11.4. Описание АЛУ регистрового типа на языке <i>Verilog</i> .....	177
11.5. Система команд процессора .....	179
<b>Глава 12. Разработка процессора для реализации в ПЛИС</b> .....	182
12.1. Функциональная схема процессора .....	182
12.2. Разработка системы команд .....	185
12.2.1. Команды с непосредственной адресацией .....	185
12.2.2. Команды с регистровой адресацией .....	187
12.2.3. Команды с косвенной регистровой адресацией .....	188
12.2.4. Команды с прямой адресацией .....	190
12.2.5. Расширение набора команд .....	190
12.3. Проект процессора в САПР .....	191
12.3.1. Модуль управления <i>control</i> .....	191
12.3.2. ПЗУ команд .....	192
12.3.3. Блок РОН .....	193
12.3.4. Модуль АЛУ .....	196
12.3.5. Устройство синхронизации записи данных ( <i>sync_wr</i> )... ..	198
12.3.6. Разработка схемы процессора .....	198
12.4. Разработка программ для синтезированного процессора .....	200
<b>Глава 13. Условия эксплуатации цифровых устройств</b> .....	204
13.1. Климатические факторы .....	205
13.2. Механические факторы .....	208
13.3. Радиационные факторы .....	209
13.4. Классификация аппаратуры по условиям эксплуатации .....	212
<b>Глава 14. Требования к конструкции РЭА</b> .....	215
14.1. Тактико-технические требования .....	215
14.2. Конструктивно-технологические требования .....	215
14.3. Эксплуатационные требования .....	216
14.4. Требования по надежности .....	217
14.5. Экономические требования .....	217
14.6. Показатели качества конструкции .....	218
<b>Глава 15. Основы автоматизированного проектирования цифровых устройств</b> .....	221
15.1. Классификация САПР .....	221
15.2. САПР печатных плат .....	222
15.3. Конструкторские САПР .....	226
<b>Глава 16. Автоматизация технологических процессов производства изделий радиоэлектроники</b> .....	229
16.1. Производство печатных плат .....	229
16.2. Автоматизация сборки РЭА. Сборка печатных плат .....	232
16.2.1. Поверхностный монтаж .....	233
16.2.2. Типичная последовательность операций монтажа .....	235

<b>Глава 17. Контроль в сборочном производстве печатных плат</b> .....	243
17.1. Визуальный контроль .....	243
17.2. Электрический контроль .....	245
17.3. Ремонт печатных плат .....	247
17.4. Испытания радиоэлектронной аппаратуры .....	249
<b>Глава 18. Единая система конструкторской документации</b> .....	251
18.1. Виды конструкторских документов .....	252
18.2. Стадии разработки КД и этапы выполнения работ при проектировании .....	253
18.3. Виды изделий .....	255
18.4. Правила оформления графической документации .....	257
18.5. Обозначения элементов на чертежах и схемах .....	261
18.6. Программа и методика испытаний цифровых устройств .....	269
<b>Глава 19. Единая система программной документации</b> .....	272
19.1. Правила выполнения схем алгоритмов, программ, данных и систем .....	272
19.2. Программная документация .....	275
19.3. Эксплуатационные документы. Руководство оператора .....	276
Список литературы .....	278



*Учебное издание*

**Кистрин Алексей Васильевич,  
Никифоров Михаил Борисович**

## **Проектирование цифровых устройств**

**Учебник**

Редактор *И. В. Могилевец*  
Компьютерная верстка: *А. В. Бобылева*  
Корректор

Изд. № 101116968. Подписано в печать . Формат 60 × 90/16.  
Гарнитура «Балтика». Бумага офсетная. Печать офсетная. Усл. печ. л. 18,0.  
Тираж 2000 экз. Заказ №

ООО «Издательский центр «Академия». [www.academia-moscow.ru](http://www.academia-moscow.ru)  
129085, Москва, пр-т Мира, 101В, стр. 1.  
Тел./факс: (495) 648-0507, 616-00-29.

Санитарно-эпидемиологическое заключение № РОСС RU. АЕ51. Н 164592 от 29.04.2014.

Отпечатано с электронных носителей издательства.  
ОАО «Тверской полиграфический комбинат», 170024, г. Тверь, пр-т Ленина, 5.  
Телефон: (4822) 44-52-03, 44-50-34. Телефон/факс: (4822) 44-42-15  
Home page — [www.tverpk.ru](http://www.tverpk.ru). Электронная почта (E-mail) — [sales@tverpk.ru](mailto:sales@tverpk.ru)