Учебно-методические пособия Учебно-научного центра ОИЯИ Дубна

УНЦ-2011-48

В. А. Калинников

ОСНОВЫ СХЕМОТЕХНИКИ ЭВМ

2011

Учебно-научный центр ОИЯИ

В. А. Калинников

ОСНОВЫ СХЕМОТЕХНИКИ ЭВМ

Учебный курс

Дубна 2011

К17

Калинников В. А.

К17 Основы схемотехники ЭВМ. — Дубна: ОИЯИ, 2011. — 165 с.

ISBN 978-5-9530-0306-3

В пособии рассмотрен круг вопросов, связанных с изучением, проектированием и применением цифровых элементов, узлов и устройств, которые являются основой для реализации различных средств обработки информации в ЭВМ. Описано использование в схемотехнике стандартных элементов, типовых функциональных узлов и микросхем программируемой логики. Приведены структуры и схемотехника запоминающих устройств, микропроцессоров, и изложена методика автоматизированных методов проектирования цифровых устройств узлов.

Kalinnikov V. A.

The Bases of Circuit Techniques of Computer. — Dubna: JINR, 2011. — 165 p.

ISBN 978-5-9530-0306-3

This manual describes the circle of questions connected with studying, designing and application of digital elements and devices which are a basis for realization of various means of processing of information in the computer. The application in circuit technique of standard electronic elements, typical functional parts and microcircuits of programmed logics is described. The structures and circuit technique of memories, microprocessors and the technique of the automated methods of digital devices design are described.

ISBN 978-5-9530-0306-3

Оглавление

1. История развития ЭВМ

С увеличением объёма вычислений появился первый счётный переносной инструмент – «счёты». В начале 17-го века возникла необходимость в сложных вычислениях, потребовались счётные устройства, способные выполнять большой объём вычислений с высокой точностью. В 1642 г. французский математик Блез Паскаль сконструировал первую механическую счётную машину «Паскалину». В 1830 г. английский учёный Чарльз Бэббидж предложил идею первой программируемой вычислительной машины – «аналитическая машина». По замыслу ученого, она должна была приводиться в действие силой пара, а программы кодироваться на перфокартах. Реализовать эту идею не удалось, так как некоторые детали машины в то время невозможно было сделать. Американский инженер Герман Холлерит был первым, кто смог реализовать идею перфокарт. Он изобрёл машину для обработки результатов переписи населения. В своей машине он впервые применил электричество для расчётов.

В 1930 г. американский учёный Ванневар Буш изобрел «дифференциальный анализатор» – первый в мире компьютер. Большой толчок в развитии вычислительной техники дала вторая мировая война. Военным понадобился компьютер, которым стал «Марк-1» – первый в мире цифровой компьютер, изобретённый в 1944 г. профессором Айкнем. В нём использовалось сочетание электрических сигналов и механических приводов. Размеры компьютера – 15 × 2,5 м. Он состоял из 750000 деталей и мог перемножить два 23-разрядные числа за 4 секунды.

В 1946 г. группой инженеров по заказу военного ведомства США был создан первый электронный компьютер – «Эниак» с быстродействием 5000 операций сложения и 300 операций умножения в секунду. Размеры компьютера: длина – 30 м, объём – 85 $м^3$ и вес – около 30 тонн. В нем использовалось 18000 электронных ламп. Первая машина с хранимой программой «Эдсак»

была создана в 1949 г., а в 1951 г. создали машину «Юнивак» – первый серийный компьютер с хранимой программой. В этой машине впервые была использована магнитная лента для записи и хранения информации.

2. Направления развития и поколения ЭВМ

2.1. Аналоговые вычислительные машины

В аналоговых вычислительных машинах (АВМ) все математические величины представляются как непрерывные значения каких-либо физических величин. В качестве машинных переменных выступает напряжение электрической цепи, а изменения их величин происходят по тем же законам, что и изменения заданных функций. В этих машинах используется метод математического моделирования, то есть создаётся модель исследуемого объекта, а результаты вычислений выводятся в виде зависимостей электрических напряжений от времени на экран осциллографа или фиксируются измерительными приборами. Основным назначением АВМ является решение линейных и дифференциальных уравнений.

Достоинства АВМ: высокая скорость решения задач, соизмеримая со скоростью прохождения электрического сигнала; простота конструкции; лёгкость подготовки задачи к решению; наглядность протекания исследуемых процессов; возможность изменения параметров исследуемых процессов во время самого исследования.

Недостатки АВМ: малая точность получаемых результатов (до 10%); алгоритмическая ограниченность решаемых задач; ручной ввод решаемой задачи в машину; большой объём задействованного оборудования, растущий с увеличением сложности задачи.

2.2. Электронные вычислительные машины

В отличие от АВМ в электронно-вычислительных машинах (ЭВМ) математические величины представляются в виде последовательности цифр. В современных ЭВМ осуществляется принцип программного управления, а

числа представляются в виде двоичных кодов, то есть в виде комбинаций «1» и «0».

Достоинства ЭВМ: высокая точность вычислений; универсальность; автоматический ввод информации, необходимой для решения задачи; разнообразие задач, решаемых ЭВМ; независимость количества оборудования от сложности задачи.

Недостатки ЭВМ: сложность подготовки задачи к решению (разработка специальных методов решения задач и программирования); недостаточная наглядность протекания процессов и сложность изменения их параметров; сложность эксплуатации и технического обслуживания.

2.3. Поколения развития ЭВМ

Можно выделить четыре основных поколения ЭВМ (табл.2.1).

Первое поколение. ЭВМ на электронных лампах (БЭСМ, «Стрела»), быстродействие порядка 20000 операций в секунду, для каждой машины – свой язык программирования.

Второе поколение. В 1960 г. в ЭВМ («Минск-2», «Урал-14») применили транзисторы, изобретенные в 1948 г. Эти ЭВМ стали более надежными, долговечными и обладающими большой оперативной памятью. В качестве носителей информации использовались магнитные ленты.

Третье поколение. В 1964 г. появились первые интегральные схемы (ИС), на базе которых затем стали создавать ЭВМ. Интегральная микросхема – это кристалл полупроводника, площадь которого около 10 мм^2. Одна ИС способна заменить до 1000 транзисторов. В этих ЭВМ появилась возможность обрабатывать параллельно несколько программ.

Четвертое поколение. В ЭВМ стали применяться большие интегральные схемы (БИС). Это привело к снижению стоимости производства компьютеров. В 80-х годах центральный процессор ЭВМ уже был выполнен на одном кристалле, то есть на БИС («Иллиак», «Эльбрус»).

Таблица 2.1. Основные поколения ЭВМ

ХАРАКТЕРИСТИКИ	ПОКОЛЕНИЯ ЭВМ			
	I	II	III	IV
Годы применения	1946-1960	1960-1964	1964-1970	1970-1980
Основной элемент	Эл. лампа	Транзистор	ИС	БИС
Количество ЭВМ в мире (шт.)	Сотни	Тысячи	Десятки тысяч	Миллионы
Размеры ЭВМ	Большие	Значительно меньше	Мини-ЭВМ	МикроЭВМ
Быстродействие (усл)	1	10	1000	10000
Носитель информации	Перфокарта Перфолента	Магнитная лента	Диск	Гибкий диск

В настоящее время развитие вычислительной техники идет в направлении создания ЭВМ с искусственным интеллектом. Под искусственным интеллектом понимают: возможность ЭВМ делать логические выводы из представленных фактов; возможность ввода информации в ЭВМ при помощи голоса; возможность ввода в ЭВМ различных изображений и др. Это еще больше расширит возможности ЭВМ и станет помощником человеку во всех областях науки и техники.

3. Понятие сигнала, типы сигналов

3.1. Типы сигналов

Под термином «сигнал» понимается упорядоченное (каким-либо образом) отображение процесса, то есть характер изменения в пространстве, во времени или по любой другой переменной физических величин, физических свойств или физического состояния объекта исследований в реальном масштабе времени. Следовательно, сигнал является отображением общей измерительной информации.

Все сигналы разделяют на две крупные группы: детерминированные и случайные. Обычно выделяют два класса детерминированных сигналов: периодические и непериодические. К периодическим сигналам относят гармонические и полигармонические сигналы, а к непериодическим – почти периодические и апериодические (или переходные) сигналы.

Рассмотрим типы сигналов, которым соответствуют определенные формы их математического описания.

Аналоговый сигнал (analog signal) является непрерывной функцией непрерывного аргумента, то есть определен для любого значения аргументов. Источниками аналоговых сигналов, как правило, являются физические процессы и явления, непрерывные в динамике своего развития во времени, в пространстве или по любой независимой переменной, при этом регистрируемый сигнал подобен (аналогичен) порождающему его процессу.

Рис.3.1. Аналоговый сигнал

Пример графического отображения данного сигнала приведен на рис.3.1, где как сама функция, так и ее аргументы могут принимать любые значения в пределах некоторых интервалов $y_1 \le y \le y_2$, $t_1 \le t \le t_2$. Если интервалы значений сигнала или его независимых переменных не ограничены, то по умолчанию они принимаются в интервале от $-\infty$ до $+\infty$.

Множество возможных значений сигнала образует континуум – непрерывное пространство, в котором любая сигнальная точка может быть определена с точностью до бесконечности. Примером сигналов, аналоговых по своей природе, может быть изменение напряженности электрического, магнитного, электромагнитного поля во времени и в пространстве.

В природе все сигналы аналоговые, именно поэтому первые электронные устройства были аналоговыми. Они преобразовывали физические

величины в пропорциональные им напряжения или ток и выполняли над ними какие-то операции. Однако аналоговые сигналы и работающая с ними аналоговая электроника имеют большие недостатки, связанные именно с природой аналоговых сигналов. Дело в том, что аналоговые сигналы чувствительны к действию всевозможных «паразитных» сигналов – шумов, наводок и помех. Шум – это внутренние хаотические слабые сигналы любого электронного устройства (транзистора, резистора и т. д.). Наводки и помехи – это сигналы, приходящие на электронную систему извне и искажающие полезный сигнал (например, электромагнитные излучения).

Дискретный сигнал (discrete signal) по своим значениям также является непрерывной функцией, но определенной только по дискретным значениям аргумента. По множеству своих значений он является конечным и описывается дискретной последовательностью отсчетов $y(n\Delta t)$, где $y_1 \le y \le y_2$, Δt – интервал между отсчетами (шаг дискретизации), $n = 0,1,2,...,N$. Величина, обратная шагу дискретизации ($f = 1/\Delta t$), называется частотой дискретизации.

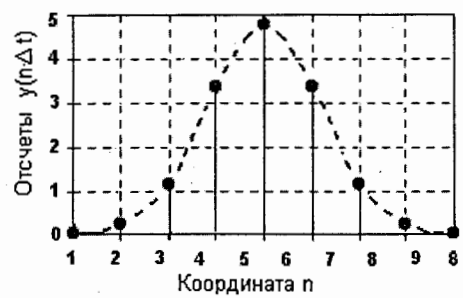


Рис.3.2. Дискретный сигнал

Если дискретный сигнал получен дискретизацией (sampling) аналогового сигнала, то он представляет собой последовательность отсчетов, значения которых в точности равны значениям исходного сигнала по координатам $n\Delta t$. Пример дискретизации аналогового сигнала, приведенного на рис.3.1, представлен на рис.3.2.

Цифровой сигнал (digital signal) квантован по своим значениям, дискретен по аргументу и описывается квантованной решетчатой функцией

$$y_n = Q_k[y(n\Delta t)],$$

где Q_k – функция квантования с числом уровней квантования k, при этом интервалы квантования могут быть как с равномерным распределением, так и с неравномерным, например, логарифмическим. Задается цифровой сигнал в виде дискретного ряда (discrete series) числовых данных – числового массива по последовательным значениям аргумента при Δt = const, но в общем случае сигнал может задаваться и в виде таблицы для произвольных значений аргумента.

Рис.3.3. Цифровой сигнал

По существу, цифровой сигнал по своим значениям (отсчетам) является формализованной разновидностью дискретного сигнала при округлении отсчетов последнего до определенного количества цифр, как это показано на рис.3.3. Цифровой сигнал конечен по множеству своих значений. Процесс преобразования бесконечных по значениям аналоговых отсчетов в конечное число цифровых значений называется квантованием по уровню, а возникающие при квантовании ошибки округления отсчетов (отбрасываемые значения) – шумами (noise) или ошибками (error) квантования (quantization). В дискретных системах и в ЭВМ сигнал всегда представлен с точностью до определенного количества разрядов, следовательно, он всегда является цифровым.

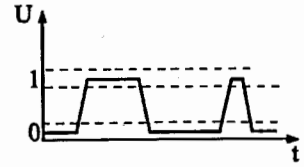


Рис.3.4. Логический сигнал

Логический сигнал. Логический сигнал является частным случаем цифрового сигнала, когда значение функция принимают только фиксированные значения «0» либо «1». На рис.3.4 показан вид логического сигнала.

3.2. Искажение полезных сигналов, шумы и наводки

Все операции, производимые электронными устройствами над сигналами, можно условно разделить на три большие группы: обработка (или преобразование), передача и хранение. Во всех этих случаях полезные сигналы искажаются паразитными сигналами (шумами, помехами, наводками). Кроме того, при обработке сигналов (например, при усилении, фильтрации) также искажается и их форма из-за несовершенства (неидеальности) электронных устройств, а при передаче сигналов на большие расстояния или при их хранении амплитуды этих сигналов ослабляются.

В случае аналоговых сигналов все это существенно ухудшает полезный сигнал, так как все его значения разрешены (рис.3.5). Следовательно, каждое преобразование, каждое промежуточное хранение, каждая передача по кабелю ухудшает аналоговый сигнал, иногда вплоть до его полного уничтожения. Отметим, что все шумы, помехи и наводки принципиально не поддаются точному расчету, поэтому точно описать поведение любых аналоговых устройств абсолютно точно невозможно. К тому же со временем параметры всех аналоговых устройств изменяются из-за старения элементов, вследствие чего характеристики этих устройств не остаются постоянными.

В отличие от аналоговых, цифровые сигналы, имеющие всего два разрешенных значения, гораздо лучше защищены от действия шумов, на-

водок и помех. Небольшие отклонения от разрешенных значений никак не искажают цифровой сигнал, так как у них всегда существуют зоны допустимых отклонений (рис.3.5). Именно поэтому цифровые сигналы допускают более сложную и многоступенчатую обработку, более длительное хранение без потерь и гораздо более качественную передачу, чем аналоговые. К тому же поведение цифровых устройств можно абсолютно точно рассчитать и предсказать.

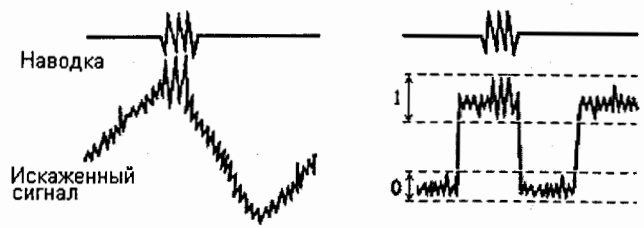


Рис.3.5. Искажение аналогового (слева) и цифрового сигнала (справа) шумами и наводками

Цифровые устройства гораздо меньше подвержены старению, так как небольшое изменение их параметров никак не отражается на их функционировании. Кроме того, цифровые устройства проще проектировать и отлаживать. Все эти преимущества обеспечивают бурное развитие цифровой электроники.

Однако у цифровых сигналов есть крупный недостаток. Дело в том, что на каждом из своих разрешенных уровней цифровой сигнал должен оставаться хотя бы в течение некоторого минимального временного интервала, иначе его невозможно будет распознать. Аналоговый сигнал может принимать любое свое значение бесконечно малое время. Иначе говоря: аналоговый сигнал определен в непрерывном времени (в любой момент времени), а цифровой – в дискретном времени (только в выделенные моменты времени). Поэтому максимально возможное (достижимое) быстродействие аналоговых устройств всегда принципиально больше, чем цифровых устройств. Аналоговые устройства могут работать с более быстро изменяющимися сигналами, чем цифровые. Скорость обработки и передачи информации аналоговым устройством всегда может быть сделана выше, чем скорость ее обработки и передачи цифровым устройством.

Кроме того, цифровой сигнал передает информацию только двумя уровнями, а аналоговый передает информацию еще и каждым текущим значением своего уровня, то есть он «более емкий» с точки зрения передачи информации. Поэтому для передачи полного объема полезной информации, который содержится в одном аналоговом сигнале, чаще всего приходится использовать несколько цифровых сигналов. К тому же, как уже отмечалось, в природе все сигналы аналоговые, поэтому для их преобразования в цифровые сигналы и обратно требуется применение специальной аппаратуры (аналого-цифровых и цифроаналоговых преобразователей).

4. Булевы операции как основа цифровой техники

4.1. Основы булевой алгебры

В основе обработки цифровых сигналов лежат *булевы операции*. Посредством этих операций можно из одного или нескольких сигналов на входе получить новый сигнал, который в свою очередь может быть подан на вход одной или нескольким таким операциям. По сути, именно булевы операции в сочетании с запоминающими элементами (например, *триггерами*) реализуют все возможности современной цифровой техники.

Булевой алгеброй называется непустое множество A с двумя бинарными операциями: $\wedge$ – *конъюнкция* (бинарная «И»), $\vee$ – *дизъюнкция* (бинарная «ИЛИ»); с унарной операцией $\neg$ – *отрицание* (унарная операция «НЕ») и двумя выделенными элементами: «0» («*Ложь*») и «1» («*Истина*»), такое, что для всех a, b и c из множества A верны следующие аксиомы:

1) ассоциативности – $a \vee (b \vee c) = (a \vee b) \vee c$, $a \wedge (b \wedge c) = (a \wedge b) \wedge c$;

2) коммутативности – $a \vee b = b \vee a$, $a \wedge b = b \wedge a$;

3) поглощения – $a \vee (a \wedge b) = a$, $a \wedge (a \vee c) = a$;

4) дистрибутивности – $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$,

$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

Таблица 4.1. Булевы отношения для двух логических переменных

Булевы отношения двух логических переменных		Свойства
$a \vee b = b \vee a$	$a \wedge b = b \wedge a$	Коммутативность
$a \vee (b \vee c) = (a \vee b) \vee c$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$	Ассоциативность
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	Дистрибутивность
$a \vee \bar{a} = 1$	$a \wedge \bar{a} = 0$	Комплементность
$\overline{(a \vee b)} = \bar{a} \wedge \bar{b}$	$\overline{(a \wedge b)} = \bar{a} \vee \bar{b}$	Закон де Моргана
$a \vee (a \wedge b) = a$	$a \wedge (a \vee c) = a$	Закон поглощения
$a \vee (\bar{a} \wedge b) = (a \vee b)$	$a \wedge (\bar{a} \vee b) = (a \wedge b)$	Блейка-Порецкого
$a \vee a = a$	$a \wedge a = a$	Идемпотентность
$\bar{\bar{a}} = a$		Инволютивность отрицания
$a \vee 0 = a$ $a \wedge 1 = 1$ дополнение 0 есть 1 $\bar{0} = 1$	$a \wedge 0 = 0$ $a \wedge 1 = a$ дополнение 1 есть 0 $\bar{1} = 0$	Свойства констант
$(a \vee b) \wedge (\bar{a} \vee b) = b$	$(a \wedge b) \vee (\bar{a} \wedge b) = b$	Склеивание
«↔» – «тогда и только тогда, когда»		Эквивалентность
«→» – «следовательно»		Импликация
«⊕» – «исключающее ИЛИ»		Сложение по модулю два
«↓» – отрицание *дизъюнкции* $(x \downarrow y = \overline{(x \vee y)} = \bar{x} \wedge \bar{y})$		Стрелка Пирса
«|» – отрицание *конъюнкции* $(x \mid y = \overline{(x \wedge y)} = \bar{x} \vee \bar{y})$		Штрих Шеффера

Простейшим и наиболее широко применяемым примером такой алгебраической системы является множество A, состоящее всего из двух элементов $A \in$ {Ложь, Истина}. В математических выражениях «*Ложь*» отождествляется с логическим нулём, а «*Истина*» – с логической единицей. Тогда операция ¬ приобретает смысл вычитания из единицы; ∨ – не-

модульного сложения, ∧ – умножения. На данном множестве A можно задать четыре *унарные* и шестнадцать *бинарных* отношений, представленных в табл.4.1. Однако все они могут быть получены через суперпозицию трёх выбранных операций «И», «ИЛИ», «НЕ».

4.2. Логические элементы и логические функции

4.2.1. Логические функции

Логическая функция – это функция, аргументами которой являются логические переменные, принимающая только два значения: «0» или «1». В свою очередь, сама логическая переменная тоже может принимать только два значения: «0» или «1».

Логический элемент – это устройство, реализующее заданную логическую функцию. Логическая функция Y = f(X1,X2,X3,...,Xn) может быть задана таблицей, которая называется «таблицей истинности» (рис.4.1). Количество строк в таблице соответствует числу возможных наборов значений аргументов функции и равно $N = 2^n$, где n – общее число входных переменных. Число различных значений функций от n логических переменных равно $N = 2^{2^n}$.

X1,X2,X3,...,Xn	Y=f(X1,X2,X3,...,Xn)
Набор значений аргументов X1,..,Xn	Значение функции Y (0 или 1)

Рис.4.1. Таблица истинности логической функции $Y=f(x_1,,,x_n)$

4.2.2. Логические функции одной переменной

Рассмотрим логические функции от одной переменной. Таблица истинности для функции от одной переменной Y=f(X) содержит всего две строки, а общее число функций равно 4. Это функции:

1. *Функция константа* «0» (Y=0). Техническая реализация этой функции состоит в соединении выхода Y с нулевым потенциалом. Таблица истинности функции «константа 0» имеет следующий вид

X	Y=f(X)
0	0
1	0

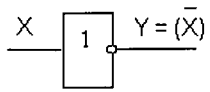
2. *Функция повторения* Y=f(X)=X. Для технической реализации этой функции необходимо соединить между собой выводы X и Y. Таблица истинности «функции повторения» имеет вид

X	Y=f(X)
0	0
1	1

3. *Функция отрицания* Y=f(X)=NOT(X) – это «НЕ» или «инверсия» (NOT(X)» или «НЕ(X)). Техническая реализация этой функции – инвертор на любом транзисторе или логическом элементе, или транзисторный ключ. Таблица истинности «функции отрицания» и обозначение логического элемента «НЕ» приведены на рис.4.2.

X	Y=f(X)
0	1
1	0

а)



б)

Рис.4.2. Таблица истинности и обозначение логического элемента «НЕ»

4. *Функция константа* «1» (Y=1). Техническая реализация этой функции – соединение вывода Y с источником питания. Таблица истинности функции константа «1» имеет вид

X	Y=f(X)
0	1
1	1

Важнейшей функцией от одной переменной является *отрицание* («НЕ»), остальные функции являются тривиальными.

4.2.3. Логические функции от двух переменных

Таблица истинности функции от двух переменных Y=f(X1,X2) содержит четыре строки, а общее число функций равно 16. Рассмотрим несколько основных функций от двух входных переменных.

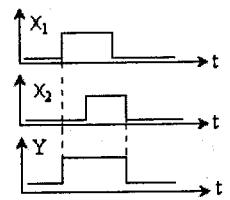
1. *Логическое* «ИЛИ» (логическое сложение, дизъюнкция)

$$Y = X1 + X2 = X1 \vee X2.$$

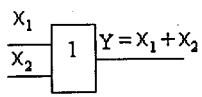
Техническая реализация этой функции – два параллельно соединенных ключа.

X_1	X_2	$Y=f(X_1,X_2)$
0	0	0
0	1	1
1	0	1
1	1	1

а)



б)



в)

Рис.4.3. Логический элемент «ИЛИ»

На рис.4.3 приведена таблица истинности элемента «ИЛИ», его временная диаграмма и логическое обозначение соответственно.

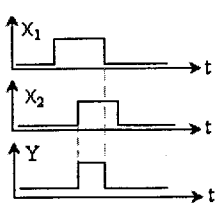
2. *Логическое* «И» (логическое умножение, схема совпадений)

$$Y = X1 \cdot X2 = X1 \,\&\, X2.$$

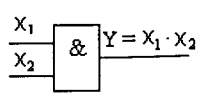
Техническая реализация этой функции – два последовательно соединенных логических ключа.

X_1	X_2	$Y=f(X_1,X_2)$
0	0	0
0	1	0
1	0	0
1	1	1

а)



б)



в)

Рис.4.4. Логический элемент «И»

На рис.4.4 приведена таблица истинности логического элемента «И», его временная диаграмма и функциональное обозначение на схемах.

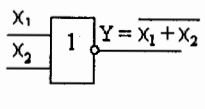
3. *Функция стрелка Пирса* («ИЛИ-НЕ»)

$$Y = NOT(X1+X2).$$

Таблица истинности функции «ИЛИ-НЕ» и его функциональное обозначение приведены на рис.4.5.

x_1	x_2	$Y=f(x_1,x_2)$
0	0	1
0	1	0
1	0	0
1	1	0

а)



б)

Рис.4.5. Логический элемент «ИЛИ-НЕ»

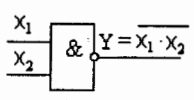
4. *Функция штрих Шеффера* («И-НЕ»)

$$Y = X1|X2 = NOT(X1X2).$$

Таблица истинности функции «И-НЕ» и его функциональное обозначение приведены на рис.4.6.

x_1	x_2	$Y=f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

а)



б)

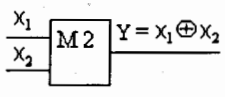
Рис.4.6. Логический элемент «И-НЕ»

5. *Функция сложение по модулю 2* («Исключающее ИЛИ»)

$$Y = X1 \oplus X2 = NOT(X1X2).$$

x_1	x_2	$Y=f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

а)



б)

Рис.4.7. Логический элемент «Исключающее ИЛИ»

Таблица истинности элемента «Исключающее ИЛИ» и его функциональное обозначение приведены на рис.4.7.

4.3. Приведение логических функций к функционально полному базису

Существуют такие наборы логических функций, с помощью которых можно выразить любые другие их комбинации. Эти наборы называются *функционально полными* или *базисами*. Наиболее известный такой базис – набор функций «И», «ИЛИ», «НЕ». Функция *штрих Шеффера* также является базисной, как и функция *стрелка Пирса*. Поэтому с помощью логических элементов «ИЛИ-НЕ» или «И-НЕ» можно собрать любую логическую схему.

Преобразование логических функций к базису ИЛИ-НЕ или И-НЕ

Логические схемы состоят из логических элементов (ЛЭ), осуществляющих логические операции, а упрощение структуры этих схем осуществляется на основе *эквивалентных преобразований* булевой алгебры. В качестве примера упростим выражение логической функции

$$y = a \vee b \vee c \vee \overline{(a \vee \bar{b} \vee c)}.$$

Используя правило *де Моргана* для выражения в скобках, получаем

$$y = a \vee b \vee c \vee \bar{a} \vee (\bar{b} \vee c) = 1 \vee b \vee c \vee \bar{b} \vee c = 1 \vee c = 1.$$

Существуют также специальные методы упрощения логической функции, например, «Карты Карно», «метод Куайна – Мак-Класки» и др.

Любую логическую функция от *n* переменных можно представить с помощью набора трех элементарных функций: инверсии, дизъюнкции и конъюнкции, так как этот набор является функционально полным. Поскольку базис *штрих Шеффера* и *стрелка Пирса* также являются функционально полными, то любую функцию можно также записать в базисе «ИЛИ-НЕ» или в «И-НЕ», то есть в дизъюнктивной нормальной форме (ДНФ) или конъюнктивной нормальной форме (КНФ). Алгоритм

перехода от таблицы истинности логической функции к ее записи в виде ДНФ следующий:

- выбрать в таблице истинности такие наборы входных переменных, которые обращают функцию в единицу;
- записать *минтермы* для выбранных наборов входных переменных, при этом если значение входной переменной в наборе единичное, то она записывается в прямой форме, если же значение переменной нулевое, то – в инверсной форме;
- полученные минтермы объединить между собой знаками дизъюнкции.

Алгоритм перехода от таблицы истинности логической функции к ее записи в виде КНФ следующий:

- выбрать в таблице истинности такие наборы входных переменных, для которых функция принимает нулевые значения;
- записать минтермы для выбранных наборов, при этом если значение входной переменной в наборе нулевое, то она записывается в прямой форме, если значение переменной единичное, то – в инверсной форме.

Рассмотрим пример приведения логической функции от двух переменных в базисе ДНФ и КНФ:

$$y = x_1 \cdot x_2 + \bar{x}_1 \cdot \bar{x}_2 . \qquad (4.1)$$

Первое инвертирование (4.1) с учетом *теоремы де Моргана* приводит к выражению

$$\bar{\bar{y}} = \overline{x_1 \cdot x_2 + \bar{x}_1 \cdot \bar{x}_2} = \overline{x_1 \cdot x_2} \cdot \overline{\bar{x}_1 \cdot \bar{x}_2} .$$

Второе инвертирование с учетом закона *двойного отрицания* приводит к искомой форме в базисе функций *штрих Шеффера*

$$\bar{\bar{y}} = y = \overline{\overline{x_1 \cdot x_2} \cdot \overline{x_1 \cdot x_2}} . \qquad (4.2)$$

Четырехкратное инвертирование (4.2) дает искомую форму в базисе функций *стрелка Пирса*

$$\bar{\bar{y}} = x_1 \cdot x_2 + \bar{x}_1 \cdot \bar{x}_2 = \overline{x_1 \cdot x_2} \cdot \overline{\bar{x}_1 \cdot \bar{x}_2} = (\bar{x}_1 + \bar{x}_2) \cdot (x_1 + x_2),$$

$$\bar{\bar{y}} = y = \overline{(\bar{x}_1 + \bar{x}_2) \cdot (x_1 + x_2)} = \overline{(\bar{x}_1 + \bar{x}_2)} + \overline{(x_1 + x_2)},$$

$$\bar{\bar{y}} = y = \overline{\overline{(x_1 + x_2)} + \overline{(x_1 + x_2)}},$$

$$\bar{\bar{y}} = y = (x_1 + x_2) + (x_1 + x_2).$$

5. Двоичное кодирование

Одиночный логический сигнал не информативен, так как может принимать только два значения «0» и «1». Поэтому, когда нужно обрабатывать или передавать большие объемы информации, применяют несколько параллельных цифровых сигналов, которые рассматриваются одновременно. В этом случае речь идет о *двоичных кодах*, образованных двоичными сигналами.

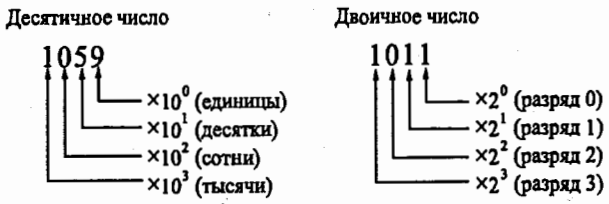


Рис.5.1. Десятичное и двоичное кодирование

Каждый из двоичных сигналов (рис.5.1), входящих в код, называется *разрядом*. Чем больше разрядов – тем большее значение может принимать данный код. Система счисления в таком коде называется *двоичной* или системой с основанием «2». Для преобразования любого десятичного числа в двоичный код надо последовательно делить это число на 2, каждый раз записывая остаток. Например, преобразуем десятичное число 13 в двоичный код:

$13_{10} \rightarrow 13/2 = 6,$ → остаток «1»,
$6/2 = 3,$ → остаток «0»;
$3/2 = 1,$ → остаток «1»;
$1/2 = 0,$ → остаток «1».

Запись ответа начинается с младшего значащего разряда, то есть $13_{10}=1101_2$.

Для преобразования двоичного кода в десятичное число каждое значение двоичного разрядного кода надо умножать на последовательные степени числа 2, то есть $1101_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 13_{10}$.

Для систем, имеющих только два состояния, обычно применяются *двоичные числа*. Однако запись таких чисел может быть очень длинной, поэтому обычно используют либо *восьмеричную* или *шестнадцатеричную* систему счисления. Разница заключается в основании, например, для записи двоичного числа в восьмеричной системе надо разбить его на группы по три бита, начиная с младшего разряда, и затем для каждой группы определить его восьмеричный эквивалент. Например:

$835_{10} = 1101000011_2 = (1\ 101\ 000\ 011_2) = 1503_8$.

Для шестнадцатеричной системы все двоичные разряды разбиваются на группы по четыре разряда, начиная с младшего, а затем уже каждая группа кодируется одним 16-ричным символом. Каждая такая группа называется *полубайтом*, а две группы – *байтом*. Каждое четырехразрядное двоичное число может принимать 16 различных значений. В качестве первых 10 символов берут числа от 0 до 9, а затем используются 6 заглавных букв латинского алфавита, то есть 0,1,2,…,9,A,B,C,D,E,F (рис.5.2).

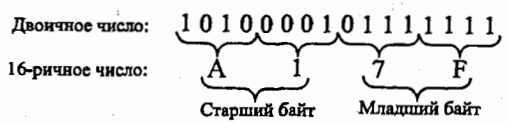


Рис.5.2. Представление двухбайтового числа

Для перевода 16-ричного числа в десятичное необходимо умножить значение младшего разряда на единицу, первого на 16, второго на 256 и так далее. Например, переведем 16-ричное число A17F в десятичное число:

$$A17F = F\cdot16^0 + 7\cdot16^1 + 1\cdot16^1 + A\cdot16^3 =$$
$$15\cdot1 + 7\cdot16 + 1\cdot256 + 10\cdot10 + 4096 = 41343.$$

6. Логические элементы в устройствах цифровой электроники

6.1. Реализация битовых операций

Реализация битовых операций в логических устройствах, в принципе, может быть любой: механической, электрической, магнитной и электромагнитной, а также в виде их комбинаций, например, электромеханической. Наиболее распространена электронная реализация битовых операций на базе транзисторов и микросхем.

6.2. Типы, структура, входные и выходные каскады микросхем

6.2.1. Резисторно-транзисторная логика

В настоящее время реализация битовых операций выполняется, в основном, на базе цифровых интегральных микросхем (ИС): это резисторно-транзисторная логика (РТЛ), диодно-транзисторная логика (ДТЛ), транзисторно-транзисторная логика (ТТЛ), комплементарная логика (КМОП- логика), эмиттерно-связанная логика (ЭСЛ).

Своё название РТЛ-логика получила благодаря реализации логических функций с помощью резисторных цепей и усиления сигнала с помощью транзистора. Схема двухвходового логического элемента «ИЛИ», выполненного в РТЛ-схеме, показана на рис.6.1.

Схема работает следующим образом. При логическом «0» на входах (*положительная, или позитивная, логика*), то есть при отсутствии напряжения на всех входах, первый транзистор закрыт и на выход через резистор поступает напряжение, близкое к напряжению питания, *логическая* «1». При поступлении логической «1» на какой-либо вход первый транзистор, работающий в ключевом режиме, открывается, и выходное напряжение падает почти до нуля. Таким образом, на выходе будет логический «0», а сама схема в положительной логике выполняет функцию «ИЛИ-НЕ», а в отрицательной (*негативной*) – «И-НЕ». Так как такой каскад является инвер-

тирующим, то к нему добавляется дополнительный выходной каскад (инвертор), чтобы получить логическую операцию «ИЛИ».

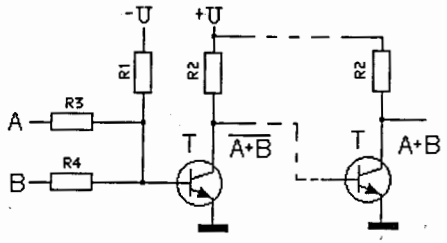


Рис.6.1. Резисторно-транзисторная логика

К достоинствам РТЛ-схем следует отнести их конструктивную простоту и низкую стоимость. Недостатки: высокая рассеиваемая мощность на включенном транзисторе и на резисторах; нечёткий уровень сигналов (уровень логической «1» принимает значения от ~ 0,9 В до напряжения питания); крайне низкое быстродействие и помехоустойчивость; низкая нагрузочная способность выходов (обычно не более трёх входов от других элементов).

Этот тип логики благодаря своей низкой стоимости был очень распространен в 1960 годах. Однако из-за высокого энергопотребления в настоящее время полностью вышел из употребления.

6.2.2. Диодно-транзисторная логика

В диодно-транзисторной логике реализация логических функций выполняется с помощью диодных цепей. Упрощённая схема двухвходового ДТЛ-элемента показана на рис.6.2. В этой схеме если хотя бы на один из входов подан уровень логического «0», то ток в схеме течет через резистор R1 и через открытый диод, где присутствует логический «0». На аноде открытого диода будет напряжение ~0,7 В, что недостаточно для открывания транзистора и для того, чтобы перевести его в режим насыщения. Транзистор закрыт, и на выходе формируется уровень логической «1».

Если на все входы поступает уровень логической «1», то ток течет через R1 в базу транзистора, образуя на анодах падение напряжения 1,4 В. Поскольку напряжение логической «1» больше этой величины, то входы диодов обратно смещены и не участвуют в работе схемы. Транзистор открыт и переведен в режим насыщения, что соответствует уровню логического «0».

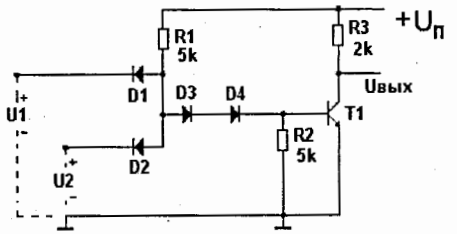


Рис.6.2. Диодно-транзисторная логика

Основное преимущество ДТЛ над более ранней РТЛ-технологией – возможность создания большого числа входов. К недостаткам следует отнести достаточно большое время задержки сигнала, которое связано с медленным процессом утечки заряда с базы транзистора в режиме насыщения при подаче на один из входов низкого логического уровня.

6.2.3. Транзисторно-транзисторная логика

На рис.6.3 показана упрощенная схема транзисторно-транзисторной логики. Основное отличие логических элементов ТТЛ от ДТЛ состоит в том, что в ТТЛ входные диоды заменены одним многоэмиттерным транзистором (МЭТ). В первом приближении *p-n* переходы база-эмиттер многоэмиттерного транзистора Т1 можно считать входными диодами, как у схем ДТЛ.

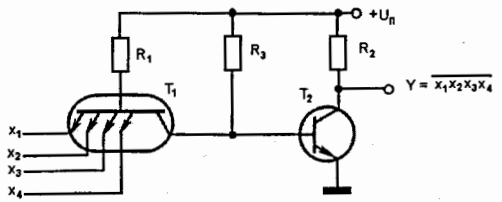


Рис. 6.3. Транзисторно-транзисторная логика

Схема работает следующим образом. При уровне логической «1» на входах транзистор Т1 закрыт, а Т2 открыт и насыщен. На выходе схемы будет логический «0». Когда на один из входов схемы поступает логический «0», многоэмиттерный транзистор Т1 становится транзистором, включенным по схеме с общей базой. При таком переходе транзистор Т2 из открытого (насыщенного) состояния переходит в закрытое, при этом накопленный в базе Т2 заряд быстро рассасывается через открытый многоэмиттерный транзистор Т1, что существенно уменьшает время переключения всего ТТЛ-элемента по сравнению с ДТЛ.

Особенностью многоэмиттерного транзистора в открытом состоянии является отсутствие прямого взаимодействия эмиттеров между собой, так как их разделяют участки базы. Можно считать, что многоэмиттерный транзистор во включенном состоянии – это структура из нескольких транзисторов, имеющих общий коллектор.

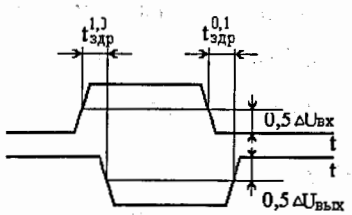


Рис.6.4. Временная диаграмма работы ТТЛ-схемы

На рис.6.4 показана временная диаграмма работы ТТЛ-схемы, где задержка включения и выключения схемы сравнительно мала благодаря работе транзистора Т1. Параметры сигналов для ТТЛ-микросхем следующие: входные напряжения: $U_{«1»}$пор = 2,4 В и $U_{«0»}$пор = 0,4 В; выходные напряжения: $U_{«0»}$вых <= 0,4 В и $U_{«1»}$вых >= 2,4 В.

6.2.4. Комплементарная логика на МОП-транзисторах (КМОП-логика)

В 1963 г. Фрэнк Вонлас изобрёл КМОП-микросхемы, которые в настоящее время являются наиболее распространенным классом семейства ло-

гических элементов. На КМОП-логике построены все микросхемы большой (БИС) и сверхбольшой степени интеграции (СБИС). Это связано с тем, что КМОП-технология позволяет создавать микросхемы с малым энергопотреблением и большой степенью интеграции элементов на одном кристалле.

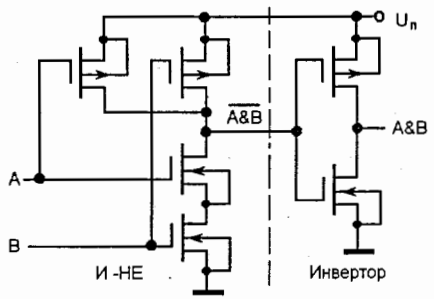


Рис.6.5. КМОП-логика

На рис.6.5 показана КМОП-схема логического элемента «2И». Рассмотрим работу этой схемы, построенной по технологии КМОП. Если на оба входа А и В подан высокий уровень, то оба транзистора снизу на схеме открыты, а оба верхних закрыты, выход соединён с «землёй».

Если хотя бы на один из входов подан низкий уровень, соответствующий транзистор сверху будет открыт, а снизу закрыт. Таким образом, выход будет соединён с напряжением питания и отсоединён от земли. Следовательно, данная схема реализует логическую функцию «2И-НЕ». Так как следующий каскад является инвертором, то эта схема реализует логическую функцию «2И».

В КМОП-схемах нет никаких нагрузочных сопротивлений, поэтому в статическом состоянии в схеме протекают только токи утечки через закрытые транзисторы, в связи с чем энергопотребление очень мало. При переключениях электрическая энергия тратится в основном на *перезаряд* емкостей затворов транзисторов и проводников схемы. Потребляемая мощность, пропорциональная частоте этих переключений, сравнительно неве-

лика. КМОП-схемы по параметрам электрических сигналов не отличаются от ТТЛ-логики.

Главным недостатком КМОП-микросхем является их уязвимость по отношению к статическому электричеству. Достаточно коснуться рукой вывода микросхемы и её целостность уже не гарантируется.

В настоящее время по параметрам быстродействия КМОП-микросхемы приближаются к ТТЛ-микросхемам. На практике в схемотехнике используют следующий подход – там, где необходимо экономить потребление тока, применяют КМОП-микросхемы, а там, где важнее скорость и не требуется экономия потребляемой мощности, применяют ТТЛ-микросхемы.

6.2.5. Микросхемы с эмиттерно-связанной логикой (ЭСЛ)

ЭСЛ-микросхемы являются самыми быстродействующими из всех типов логик. Это быстродействие обеспечивается за счет целого ряда особенностей.

1. Главная особенность, повышающая быстродействие ЭСЛ, заключается в том, что схема ее логического элемента основана на дифференциальном усилителе (рис.6.6), два транзистора которого переключают ток и не попадают в режим насыщения (благодаря чему значительно сокращается время перехода транзисторов из открытого состояния в закрытое).

2. Величина тока, задаваемая генератором стабильного тока (ГСТ), протекающего через транзисторы и сопротивления резисторов в коллекторных нагрузках (рис.6.6.), выбраны таким образом, чтобы исключить *режим насыщения* транзисторов в открытом состоянии независимо от разброса коэффициента усиления каскада.

3. В ЭСЛ-микросхемах введены два противофазных выхода – прямой и инверсный. Поэтому в ЭСЛ-микросхемах отсутствуют промежуточные инверторы, которые в ТТЛ- и КМОП-микросхемах вносят дополнительную задержку и снижают общее быстродействие.

4. С целью уменьшения времени перезаряда паразитных емкостей за счет уменьшения выходного сопротивления логических элементов в ЭСЛ-схемы введены мощные эмиттерные повторители с сопротивлениями нагрузки малой величины (50 Ом).

5. Сопротивления нагрузки выходных эмиттерных повторителей подключаются не к шине отрицательного питания, а к отдельному источнику смещения. Напряжение источника смещения меньше, чем на шине отрицательного питания, поэтому мощность, рассеиваемая этими сопротивлениями нагрузки, снижена на порядок.

6. Уменьшение времени задержки в ЭСЛ-микросхемах достигается также за счет уменьшения длительности фронтов выходных импульсов и за счет уменьшения перепада напряжения на фронтах импульсов:

$U_{«1»вых} = -0{,}96$ В, $U_{«0»вых} = -1{,}65$ В.

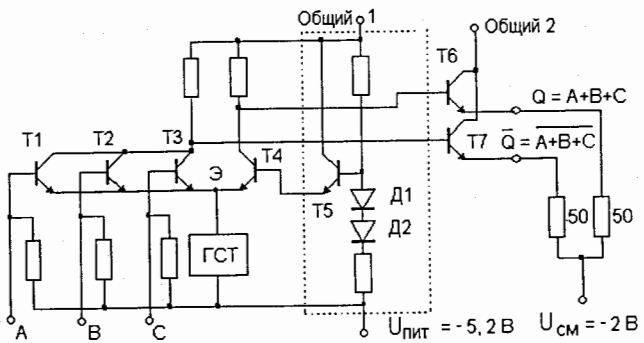


Рис.6.6. Схема ЭСЛ логического элемента «ИЛИ»

Основная область применения ЭСЛ-микросхем – быстродействующая цифровая электроника.

6.3. Выходные каскады и согласование связей между микросхемамн

На рис.6.7 показаны три типа выходных каскадов микросхем. Это схемы с двумя состояниями на выходе (рис.6.7, *а*), схемы с общим коллектором (рис.6.7, *б*) и схемы с тремя состояниями на выходе (рис.6.7, *в*).

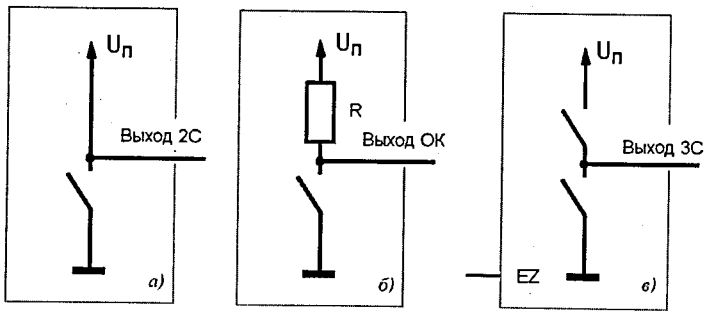


Рис.6.7. Типы выходных каскадов у микросхем

Для стандартных типов ТТЛ-микросхем значение входного тока составляет порядка 1,0 мА, а их выходные каскады обладают хорошей нагрузочной способностью, поэтому сопряжение между собой ТТЛ-элементов не представляет никакой проблемы. В том случае, когда выходной каскад микросхемы не может обеспечить требуемый ток для группы схем, подключенных к этому выходу (рис.6.8, *а*), используют дополнительные логические элементы (рис.6.8, *б*).

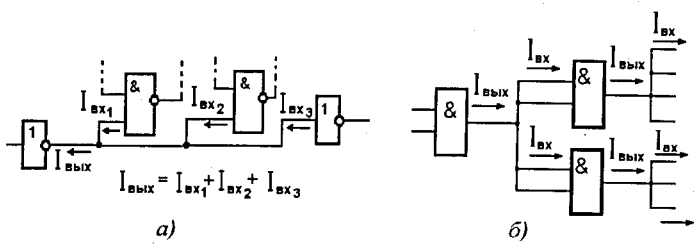


Рис.6.8. Нагрузочная способность выходных каскадов (*а*) и наращивание числа входов (*б*)

Другая возможность увеличить коэффициент разветвления схемы по выходу – применение микросхем с открытым коллектором (рис.6.9, *а*). В этом случае выбором внешнего резистора можно подобрать требуемое значение выходного тока. Кроме того, этот каскад позволяет выполнять по выходу операцию «проводного ИЛИ», как это показано на рис.6.9, *б*, что приводит к сокращению общего количества логических модулей.

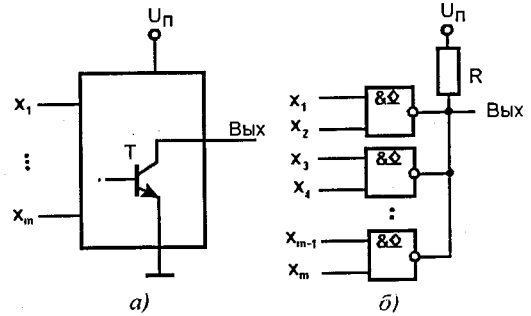


Рис.6.9. Выходной каскад с открытым коллектором

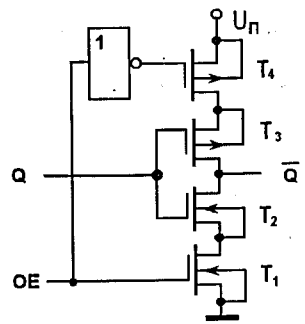


Рис.6.10. Выходной каскад с тремя состояниями

На КМОП-схемах достаточно просто создавать выходные каскады *с тремя состояниями*, где третье состояние – отключено. На рис.6.10 показан выходной каскад КМОП-схемы с тремя состояниями. При логической «1» на входе ОЕ схема работает в обычном режиме. При логическом «0» на входе ОЕ транзисторы Т1 и Т4 заперты и выход Q отключен от питания.

6.4. Цепи питания микросхем

6.4.1. Паразитные связи микросхем

Одной из главных задач при разработке цифровой электроники на базе микросхем является борьба с шумами и помехами, которые приводят к сбоям функционирования логики. Типовой проблемой здесь является наличие токовых импульсов при логическом переключении элемента из высокого состояния в низкое, и наоборот. Эти импульсы по цепям питания и по земляным

шинам распространяются к другим элементам и вызывают сбои в режиме работы схем. Для борьбы с этими помехами нужно: обеспечить «хорошую землю» за счет снижения сопротивления земляных шин, сокращение путей прохождения токов в схеме, фильтрацию питающего микросхемы напряжения.

6.4.2. Оптоэлектронная развязка

В тех случаях, когда не удается «отстроиться» от помех, идущих по земле, применяют методы гальванической развязки, то есть логические элементы не имеют общих связей ни по питанию, ни по земле. Это реализуется в методе оптронной развязки (рис.6.11). Здесь передача сигнала от источника приемнику выполняется оптическим методом с использованием на выходе светодиода, а на входе фототранзистора.

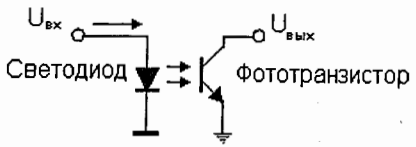


Рис.6.11. Оптоэлектронная развязка

6.4.3. Фильтрация питающих напряжений

Для фильтрации напряжения между шинами питания и землей включают конденсаторы. Установка фильтрующих конденсаторов создает короткий путь, по которому протекают токовые импульсы перезарядки, возникающие в моменты переключения логики. Эти конденсаторы должны иметь малое сопротивление для высокочастотных сигналов и малые паразитные индуктивности.

6.5. Элементы задержки, формирователи импульсов

При разработке цифровой электроники в большинстве случаев используются три модели представления цифровых сигналов: *логическая модель* (рис.6.12, *а*), модель с временными задержками – *физическая* (рис.6.12, *б*), модель с учетом электрических эффектов – *реальная* (рис.6.12, *в*).

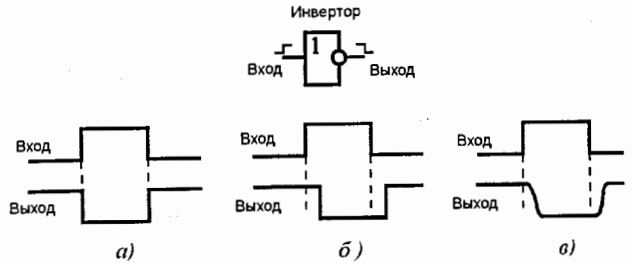


Рис.6.12. Три модели описания цифровых сигналов

Первую модель используют при разработке цифровых схем, работающих с низкой скоростью, то есть когда быстродействие не принципиально. Вторая модель учитывает задержки срабатывания логических элементов, что необходимо учитывать для всех быстродействующих схем, а также при разработке логических устройств, в случае одновременного изменения нескольких входных сигналов. Третья модель используется при реальном проектировании конкретной цифровой схемы. Она учитывает реальные задержки, входные и выходные токи, переходные сопротивления и ёмкости.

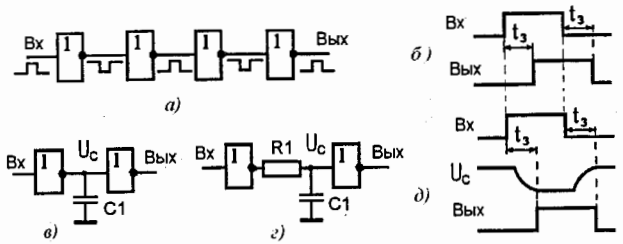


Рис.6.13. Типовые схемы цифровой задержки сигналов

При необходимости получения еще большей длительности задержки сигнала используют схемы с включенной *RC-связью* (цепочкой). Так, если в схеме только с конденсатором (рис.6.13, *в*) задержка составляет около 100 нс, то в схеме с *RC-связью* (рис.6.13, *д*) эта задержка из-за большой постоянной времени $\tau \geq 1/RC$ (номинал резистора может быть порядка сотен Ом) может быть достаточно значительной.

6.6. Элементы индикации

Для индикации в цифровой электронике используются светодиоды. На рис.6.14 показаны две типовые схемы включения светодиодов в схемах индикации.

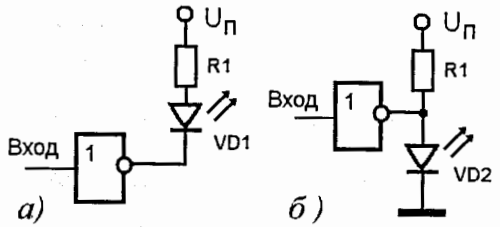


Рис.6.14. Типовые схемы цифровой индикации

В первой схеме (рис.6.14, *а*) светодиод «горит», когда на выходе инвертора логический «0», а во второй схеме диод «горит», когда на выходе логическая «1». Сопротивление R1 выбирается в зависимости от характеристик используемого типа светодиода.

7. Функциональная устойчивость и риски сбоя в логических схемах

7.1. Риски сбоя в комбинационных схемах

Изменение значений выходных сигналов логического элемента происходит с некоторым запаздыванием по отношению к входным сигналам, которое учитывается задержкой в моделях элементов. Каждый элемент характеризуется некоторой средней задержкой, значение которой может меняться в зависимости от режима работы элемента, комбинации входных сигналов, температуры, отклонения в технологии изготовления элемента и т. д. Временное рассогласование входных сигналов элемента может привести к появлению ложного сигнала на выходе логического элемента. Такая возможность появления ложных сигналов носит название «риска сбоя».

Риском сбоя называется возможность появления на выходе цифрового устройства сигнала, не предусмотренного алгоритмом его работы, что может привести к ложному срабатыванию. Отметим, что риск сбоя представ-

ляет собой только *возможность* ложного срабатывания, то есть при наличии риска сбоя может отсутствовать ложное срабатывание. Следовательно, риск сбоя представляет собой наихудший случай. Различают следующие виды рисков сбоя: статические, динамические, логические и функциональные.

7.1.1. Статические риски сбоя

На рис.7.1 показана работа элементов «И» и «ИЛИ» при подаче на их входы двух последовательных во времени наборов $X_1 = x_1x_0 =$ «01» и $X_2 = x_1x_0 =$ «10». Значение выходного сигнала y_1 для элемента «И» на этих наборах должно оставаться постоянным и равным «0», а значение y_2 – равным «1». Это условие выполняется в случае разброса моментов переключения переменных во времени x_1 и x_0, как показано на рис.7.1, *а* и рис.7.1, *в*.

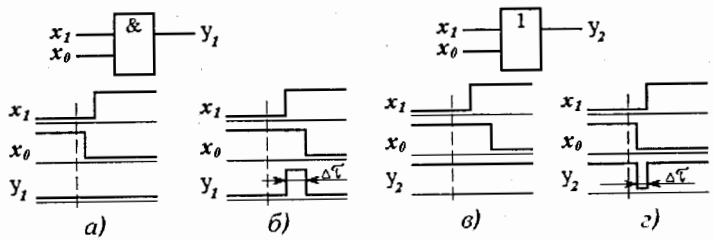


Рис.7.1. Разброс во времени моментов переключения

Если разброс моментов переключения соответствует значениям на рис.7.1, *б* и рис.7.1, *г*, то очевидно, что на выходе схемы «И» появится логический сигнал «1» длительностью $\Delta\tau_1$, а на выходе схемы «ИЛИ» – сигнал «0» длительностью $\Delta\tau_2$. Эти ложные сигналы и являются *рисками сбоя,* причем видно, что они могут присутствовать, а могут и отсутствовать. Все дальнейшие примеры будут иллюстрироваться временными диаграммами для наихудшего случая, когда риск сбоя обязательно имеет место.

Статическими называются риски сбоя в случае, если y(X1) = y(X2), где y – булева функция; X1, X2 – наборы входных сигналов до и после события изменения перехода соответственно. Если сигналы на выходе схемы для

двух смежных входных наборов остаются одинаковыми, а во время переходного процесса возможно появление ложного сигнала противоположного значения, то это называется *статическим риском сбоя* (рис.7.1, *б*, рис.7.1, *г*).

Риск сбоя называется *статическим в нуле* S_0, если $y(X_1) = y(X_2) = «0»$. Риск сбоя называется *статическим в единице* S_1, если $y(X_1) = y(X_2) = «1»$. Таким образом, на рис.7.1, *б* имеет место статический риск сбоя в нуле S_0, а на рис.7.1, *г* – статический риск сбоя в единице S_1.

При достаточно большой разности $\Delta\tau$ помеха будет иметь длительность, превышающую время переключения элемента, и амплитуду, равную номинальному сигналу. Это уже полноценный сигнал, на который могут реагировать последующие элементы (входы синхронизации, установки в «0» или «1», загрузки данных и т. п.). Такие помехи являются опасной ситуацией для цифровых схем, тем более что их практически невозможно увидеть на осциллографе. Они могут сужаться до полного исчезновения, но могут и расширяться, проходя через логические цепи.

7.1.2. Динамические риски сбоя

На рис.7.2, *а* приведена схема, реализующая функцию $y = x_2 x_1 + x_0$. Пусть входной набор сигналов $X_1 = x_2x_1x_0 = «010»$ изменяется на входной набор $X_2 = x_2x_1x_0 = «101»$. На рис.7.2, *б* приведены временные диаграммы, соответствующие наихудшему случаю разброса моментов переключения переменных x_2, x_1 и x_0. Поскольку $y(X_1) = 0$, а $y(X_2) = 1$, из рис.7.2, *б* видно, что на выходе схемы имеет место многократное переключение вместо идеального алгоритмического перехода «01».

Пусть входной набор $X_1 = x_2x_1x_0 = «011»$ изменяется на входной набор $X_2 = x_2x_1x_0 = «100»$, тогда из рис.7.2, *в* видно, что вместо идеального алгоритмического перехода «10» на выходе имеет место многократное переключение.

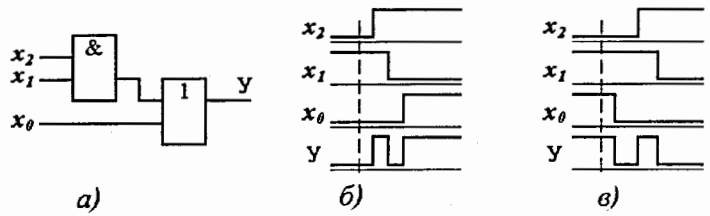


Рис.7.2. Временная диаграмма работы элемента при динамическом риске сбоя

Риск сбоя называется *динамическим*, если $y(X_1) \neq y(X_2)$, где y – булева функция. Динамический риск сбоя предполагает возможность многократного изменения сигнала на выходе при переходе от входного набора X_1 к набору X_2, когда выходной сигнал меняется на противоположный. Риск сбоя называется динамическим типа D_+ при переходе на выходе «01», если $y(X_1) = 0$, а $y(X_2) = 1$. Риск сбоя называется динамическим типа D_-, если $y(X_1) = 1$, а $y(X_2) = 0$. Таким образом, на рис.7.2, *б* имеет место динамический риск сбоя D_+, а на рис.7.2, *в* – динамический риск сбоя D_-.

Из временных диаграмм работы схемы видно, что динамический риск сбоя является следствием статического риска сбоя. Наличие динамических рисков сбоя в цифровой схеме также может привести к нарушению закона ее функционирования.

7.1.3. Логический риск сбоя

Логическим риском сбоя называется статический риск сбоя, появляющийся при смене *соседних* наборов входных сигналов, который может быть устранен изменением логической структуры, реализующей булеву функцию.

Рассмотрим переход набора входных сигналов от $X_1 = x_2x_1x_0 = «110»$ к $X_2 = x_2x_1x_0 = «010»$ для функции y, представленной картой Карно (рис.7.3, *а*). Логическое выражение для данной функции будет $y = x_2 x_1 + \bar{x}_2 \bar{x}_0$, а ее схемная реализация приведена на рис.7.3, *б*. Обратим внимание, что в дан-

ном случае осуществляется переход между соседними наборами, причем x_1= 1 и x_0= 0 являются константными сигналами. Следовательно, для этого случая можно записать $y = x_2 + \bar{x}_2$. Очевидно, что здесь y(X_1) = y(X_2) = 1. Однако на элементе «ИЛИ» возможен статический риск сбоя в единице (это отражено на рис.7.3, *в*).

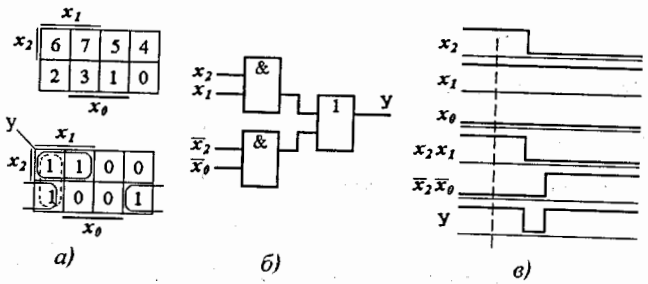


Рис.7.3. Логический риск сбоя

Можно устранить этот риск сбоя, видоизменив аппаратную реализацию данной функции. Для этого введем в карте Карно дополнительный контур, показанный штриховой линией на рис.7.3, *а*. Тогда уравнение функции будет иметь вид $y = x_2 x_1 + \bar{x}_2 \bar{x}_0 + x_1 \bar{x}_0$. Так как при переходе от набора X_1 к X_2 простая импликанта $x_1 \bar{x}_0$ все время равна «1», то риск сбоя, выявленный выше, не будет проявляться на выходе данной схемы. Эта ситуация отражена на временной диаграмме, представленной на рис.7.4, *а*, а ее схемная реализация показана на рис.7.4, *б*.

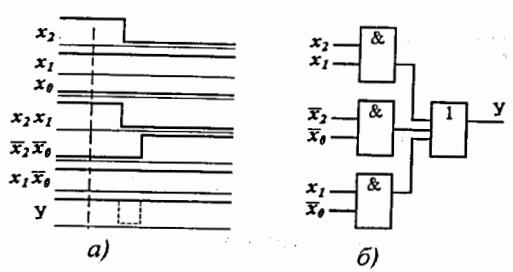


Рис.7.4. Устранение логического риска сбоя

7.1.4. Функциональный риск сбоя

Риски сбоя, проявляющиеся при *многоместной* (не соседней) смене наборов и определяемые характером самой функции, называются функциональными. Такие риски сбоя не могут быть устранены изменением логической структуры, реализующей булеву функцию.

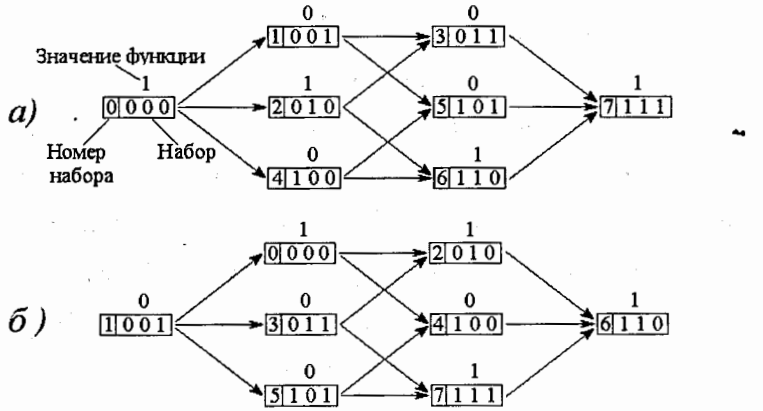


Рис.7.5. Функциональный риск сбоя

Рассмотрим поведение выходного сигнала функции при многоместной смене наборов. Пусть набор «0» переходит в набор «7». Так как все переменные изменяются, а моменты их изменения в общем случае неизвестны, то возможны различные переходы от набора «0» к набору «7» (рис.7.5, *а*). Существует единственный путь смены наборов: 0→2→6→7, при котором не будет статического риска сбоя, так как y(X_1 = 0) = y(X_2 = 7) = 1. Во всех остальных случаях будет статический риск сбоя в единице S_1. Причем никакими аппаратными средствами устранить его нельзя, так как значения выхода на промежуточных наборах определяются *характером самой функции.*

По тем же причинам при переходе от набора X_1 = «1», на котором y(X_1) = 0, к набору X_2 = «6», на котором y(X_2) = 1, возможен следующий путь смены наборов: 1→0→4→6, когда имеет место *динамический* риск сбоя D_+, так-

же определяемый характером самой функции. Во всех остальных случаях смены наборов (рис.7.5, *б*) будет чисто алгоритмический переход «01».

7.2. Функциональная устойчивость, состязания в логических схемах

Функциональная устойчивость определяется стабильностью реализации цифровым устройством заданного алгоритма работы при наличии разброса задержек выполнения операций в логических элементах, задержек сигналов в линиях связи и электромагнитных наводок и паразитных сигналов.

В схемотехнике проблема функциональной устойчивости может быть сведена к устранению опасных состязаний (гонок) сигналов устройства. Проблема устранения гонок в схемотехнике является очень серьезной задачей, так как большинство трудно обнаруживаемых ошибок связано именно с гонками. Эффект состязаний сигналов, приводящий к неустойчивой работе цифрового устройства, известен давно. Наиболее наглядным примером является эффект «дребезга» контактов реле или кнопок в устройствах.

Состязаниями или гонками сигналов называется процесс их распространения в различных цепях цифрового устройства при существовании разбросов временных задержек этих цепей. Цепь – совокупность логических и других элементов и линий связи между ними. *Алгоритмическим переходом* называется изменение сигнала на выходе какой-либо схемы, предусмотренное алгоритмом ее работы. *Неалгоритмическим переходом* называется изменение выходного сигнала, не предусмотренное алгоритмом ее работы. Опасными называются такие состязания, которые могут привести к неалгоритмическому переходу в цифровой схеме при заданных условиях ее работы, а неопасными называются такие состязания, которые не могут привести к неалгоритмическому переходу.

Задержка логической схемы – сумма задержек срабатывания логических элементов и задержек распространения сигналов по цепям связи между ними. Важнейшим параметром, характеризующим инерционность логического элемента, является среднее время задержки выходного сигнала по отношению к входному $t_{зд.ср.}$. Трудность учета задержек при проектировании зависит от соотношения значений задержек в самих логических элементах и задержек в цепях связи $t_{св.}$ между ними.

Ситуации, когда задержки $t_{св}$ превышают $t_{зд.ср.}$, возникают при использовании не очень быстродействующих элементов, и когда сигналы передаются между блоками на достаточно большое расстояние. Специфическим фактором, приводящим к разбросу $t_{зд.ср.}$, является длительность фронта (спада) входного сигнала при наличии разброса порогов срабатывания логических элементов. Такое явление часто называют гонками по входу (рис.7.6). Опасностью возникновения гонок по входу объясняются ограничения на максимальную длительность переходов «01» и «10» входных сигналов, приводимые в паспортах многих микросхем.

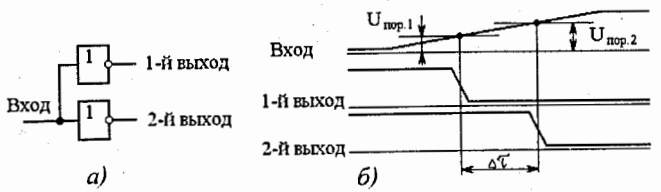
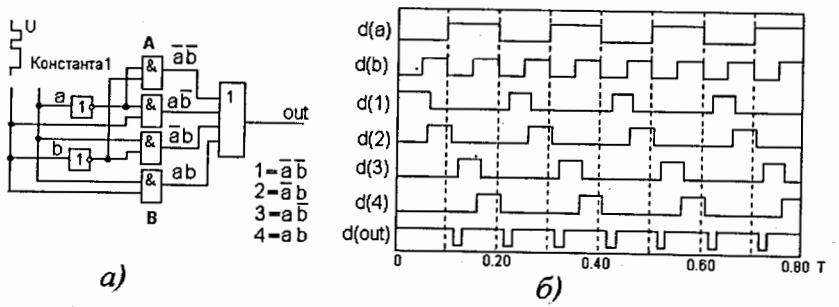


Рис.7.6. Гонки по входу

На практике изменение набора входных переменных комбинационной схемы часто происходит при несогласованных во времени изменениях различных входных переменных. При этом схема фактически находится под действием нескольких последовательно сменяющихся наборов, причем каждый из них дробится еще на несколько наборов при прохождении через схему из-за несогласованных значений $t_{зд.ср.}$ элементов. В результате изменение сигнала на каждом выходе схемы и внутреннем узле в действительности происходит не мгновенно, а образует некоторый сложный динамический процесс. Кроме того, встречаются участки, где сигнал разветвляется и

получившиеся два сигнала распространяются по двум независимым направлениям, а потом оба сигнала встречаются на входах одного элемента.



а) *б)*

Рис.7.7. Прохождения сигнала через логические элементы схемы

Например, в схеме на рис.7.7, *а* константа «1» в тракте А проходит четное число вентилей, а в тракте В – нечетное. Анализ подобной схемы методами *алгебры логики* без учета задержки показывает, что выход схемы будет равен «1» при любом значении входа. Если же учесть суммарную задержку при прохождении сигнала по обоим трактам, то может оказаться, что задержка по тракту А не равна задержке по тракту В. Это обстоятельство вызовет на выходе ложное срабатывание. В этом легко убедиться, анализируя временную диаграмму, приведенную на рис.7.7, *б*.

Таким образом, в различных частях комбинационной схемы, в зависимости от числа последовательно включенных элементов, переходный процесс после смены входного набора будет заканчиваться в разное время. Это приведет либо к деформированию длительности выходных сигналов, либо к появлению рисков сбоя. Если под воздействием входных сигналов схема из одного состояния может перейти в различные состояния в зависимости от задержек в элементах схемы, то в этом случае состязания называют критическими. Критические состязания приводят к появлению ложного срабатывания, не предусмотренного алгоритмом работы схемы.

Для нахождения критических состояний используется *динамический анализ комбинационной схемы*. В этом анализе изменение входного набора

рассматривается как неодновременные изменения различных входных переменных, образующих этот набор. Поэтому последовательность входных наборов можно рассматривать как набор входных последовательностей из нулей и единиц, действующих независимо друг от друга на разных входах. Затем производится анализ переключательного процесса.

Переключательным процессом называется последовательность уровней «1» и «0», которая на любом конечном наблюдаемом интервале времени содержит конечное число переходов «01» и «10». Примеры переключательных процессов приведены на рис.7.8. Длиной переключательного процесса называется общее число изменений сигнала в нем. Например, для процесса x_4 на рис.7.8 длина равна 3.

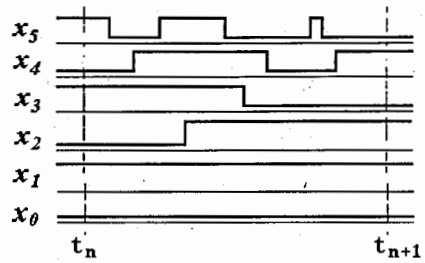


Рис.7.8. Сложный переключательный процесс логической схемы

Переключательный процесс называется сложным, если его длина больше или равна 2. В противном случае он называется простым переключением. Для векторного процесса соответствующим понятием является *вектор длин*. Компоненты этого вектора – длины процессов, являющихся компонентами векторного процесса. Например, векторный процесс, компонентами которого являются шесть процессов на рис.7.8, имеет вектор длин (5, 3, 1, 1, 0, 0). Векторный переключательный процесс считается простым переключением, если все его компоненты – простые переключения, совершаемые одновременно. В противном случае векторный процесс считается сложным.

8. Комбинационные логические устройства

8.1. Классификация логических устройств

Устройства, оперирующие с двоичной (дискретной) информацией, подразделяются на два класса: *комбинационные* и *последовательностные*. Комбинационные устройства (КУ) характеризуются отсутствием элементов памяти. Сигналы на их выходах в любой момент времени однозначно определяются сочетанием сигналов на входах и не зависят от предыдущих сигналов. Схемными признаками таких устройств является отсутствие цепей обратной связи с выхода на вход. К комбинационным устройствам относятся сумматоры, дешифраторы, шифраторы, преобразователи кодов, мультиплексоры, демультиплексоры, схемы сравнения и преобразования кодов.

В последовательностных устройствах (или автоматах с памятью) выходной сигнал определяется не только набором сигналов, действующих на входах в данный момент времени, но и внутренним состоянием устройства, которое зависит от того, какие наборы сигналов действовали во все предшествующие моменты времени. Следовательно, можно говорить, что последовательностные устройства обладают памятью, то есть хранят сведения о прошлом работы устройства.

Рассмотрим пример комбинационного устройства. Пусть КУ формирует на выходе логический сигнал, определяющий совпадение сигналов на входах. В таком устройстве на выходе формируется логическая «1» только тогда, когда на обоих входах действует логическая «1» либо на обоих входах действует логический «0». Если же на одном из входов действует логическая «1», а на другом – логический «0», то на выходе устройства образуется логический «0». Логическое устройство, в котором значение формируемой на выходе логической функции определяется лишь значениями ее аргументов в данный момент времени, является комбинационным.

Рассмотрим другой пример. Счетчик подсчитывает число импульсов. В каждый момент времени его состояние соответствует числу поступивших на вход импульсов, а результирующая выходная информация будет определяется тем, какое было его состояние до интервала времени счета и сколько импульсов поступило на вход в данном интервале. Следовательно, данное устройство является последовательностным.

8.2. Сумматор

Сумматором называется функциональное устройство, предназначенное для сложения двоичных чисел. Построение двоичных сумматоров обычно начинается с сумматора *по модулю 2*.

X	Y	Out
0	0	0
0	1	1
1	0	1
1	1	0

Рис.8.1. Таблица истинности сумматора по модулю 2

На рис.8.1 приведена таблица истинности (ТИ) такого сумматора. Ее можно получить, исходя из правил суммирования в двоичной арифметике. В соответствии с этой таблицей истинности реализуем схему сумматора по модулю 2. Эта схема приведена на рис.8.2.

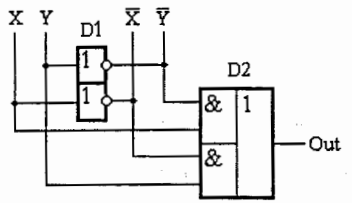


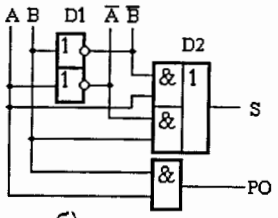
Рис.8.2. Схема сумматора по модулю 2

Сумматор по модулю 2 выполняет суммирование без учета переноса. В *полном двоичном сумматоре* необходимо учитывать значение переноса. Это требует дополнительных схем, формирующих значение переноса в следующий двоичный разряд. Таблица истинности такой схемы, называе-

мой *полусумматором*, приведена на рис.8.3, *а*, а ее принципиальная схема на рис.8.3, *б*. Схема полусумматора формирует перенос в следующий разряд, но не может учитывать перенос из предыдущего разряда, поэтому она и называется полусумматором.

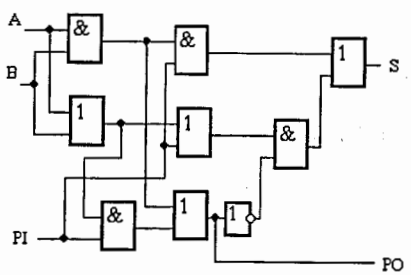
A	B	S	PO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

а)



б)

Рис.8.3. Таблица истинности и схема полусумматора



PI	A	B	S	PO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

а) *б)*

Рис.8.4. Схема полного двоичного сумматора

На рис.8.4, *а* приведена принципиальная схема *полного двоичного одноразрядного сумматора*, построенного по его таблице истинности. Таблицу истинности полного двоичного одноразрядного сумматора можно получить из правил суммирования двоичных чисел. В этой таблице (рис.8.4, *б*) в качестве входов использованы одноразрядные числа A и B; перенос на входе обозначен буквой PI (используется буква I, сокращение от английского слова *input* – вход), а перенос на выходе – PO (сокращение от английского слова *output* – выход). Обозначение полного двоичного сумматора показано на рис.8.5.

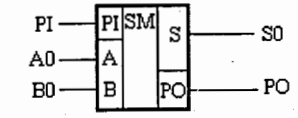


Рис.8.5. Обозначение полного двоичного сумматора

Для того чтобы получить многоразрядный сумматор, достаточно соединить входы и выходы переносов соответствующих двоичных разрядов. Схема четырехразрядного сумматора приведена на рис.8.6, а на рис.8.7 показано его обозначение.

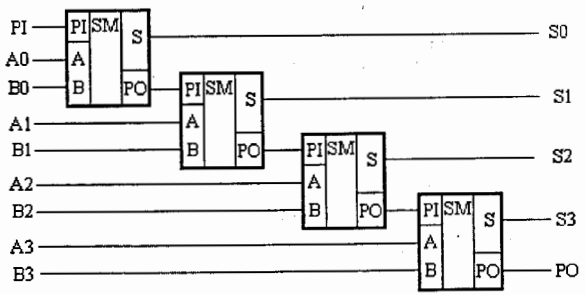


Рис.8.6. Схема четырехразрядного двоичного сумматора

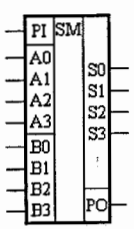


Рис.8.7. Обозначение четырехразрядного двоичного сумматора

8.3. Шифратор

Шифратором называется комбинационное устройство, преобразующее какой-либо код, подаваемый на входные шины, в соответствующий (требуемый) код на выходах. Поэтому задачей шифратора является сформировать нужный код. На ввод шифратора могут подаваться различные сигналы, например, логический «0» через контакты кнопок клавиатуры управления

или сигналы с других устройств. Во всех случаях в шифраторе происходит преобразование одного сигнала в *n*-разрядный код.

На рис.8.8 показана схема шифратора, который преобразует номер сработавшей кнопки в двоичный код на выходе. Если ни одна кнопка не нажата, то на выходах 1-2-4-8 шифратора устанавливается сигнал с уровнем логической «1». При нажатии на одну из кнопок на выходе 1-2-4-8 появляется сигнал двоичного кода, соответствующий номеру нажатой кнопки.

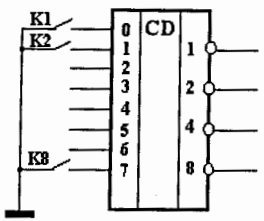


Рис.8.8. Шифратор

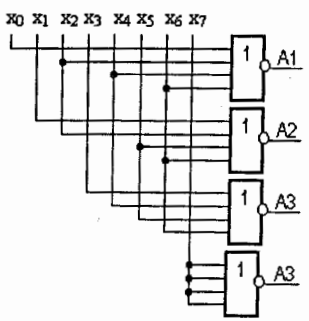


Рис.8.9. Схема четырехразрядного шифратора

Для построения такого шифратора можно использовать логические схемы «ИЛИ» – по одной на каждый выход. При этом схема разбивается на *n* простых фрагментов. К входу элементов «ИЛИ» каждого выходного разряда должны быть подключены те входы шифратора, в двоичном представлении номера которых есть единица в данном разряде. Например, к элементу «ИЛИ» (рис.8.9) младшего разряда формируемого выходного кода должны быть подключены все четные входы, поскольку у всех четных

номеров, и только у них, в младшем разряде содержится единица. Функциональная схема такого шифратора показана на рис.8.9.

8.4. Дешифратор

Дешифратор (или декодер) – это КУ с несколькими входами и выходами, у которого определенным комбинациям входных сигналов соответствует активное состояние одного из выходов (рис.8.10). Дешифратор преобразует двоичный или двоично-десятичный коды в унитарный код (то есть в код, у которого только один из разрядов равен логической «1»).

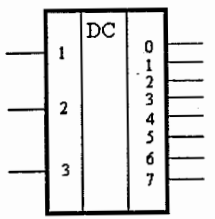


Рис.8.10. Дешифратор

По принципу построения дешифраторы делятся на *одноступенчатые* и *многоступенчатые*. Одноступенчатые дешифраторы выполняют прямое преобразование входных сигналов, поданных в двоичном (параллельном) коде, в выходной сигнал на одном из выходов. Если дешифратор имеет *n* входов и *m* выходов и использует все возможные наборы входных переменных ($m = 2^n$), то такой дешифратор называют *полным*, если же используются только часть наборов, то такой дешифратор называют *неполным*.

Дешифраторы в основном используют для обращения к различным цифровым устройствам, номер которого (его адрес) представлен двоичным кодом. Формально описать работу дешифратора можно, задав список функций, отрабатываемых каждым из его выходов Y_i, например:

$$Y_0 = \overline{a_4 a_2 a_1};\ Y_1 = \overline{a_4 a_2} a_1;\ Y_2 = \overline{a_4} a_2 \overline{a_1};\ Y_3 = \overline{a_4} a_2 a_1;\ \ldots\ Y_7 = a_4 a_2 a_1.$$

Матричные дешифраторы состоят из логических схем совпадения «И», каждая из которых имеет *n* входов. На входы подаются все возмож-

ные комбинации прямых и инверсных разрядов дешифрируемого числа X. Недостатком матричных дешифраторов является существенное увеличение число входов логических элементов с ростом его разрядности.

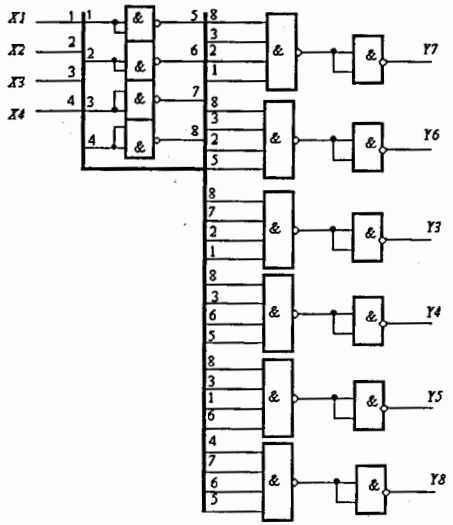


Рис.8.11. Схема дешифратора в базисе «И-НЕ»

В качестве примера построим дешифратор для перевода двоичных чисел от 3 до 8 в десятичный код в базисе «И-НЕ». Таблица истинности работы данного дешифратора представлена в табл.8.1.

Таблица 8.1. Таблица истинности дешифратора

Входы				Выходы
*X*4	*X*3	*X*2	*X*1	*Y*
0	0	1	1	*Y*3
0	1	0	0	*Y*4
0	1	0	1	*Y*5
0	1	1	0	*Y*6
0	1	1	1	*Y*7
1	0	0	0	*Y*8

Из табл.8.1 получаем функции для каждого выхода путем записи КНФ:

$Y3 = \overline{X4}X3\overline{X2}X1$; $Y4 = \overline{X4}X3\overline{X2}\overline{X1}$; $Y5 = \overline{X4}X3\overline{X2}X1$;

$Y6 = \overline{X4}X3X2\overline{X1}$; $Y7 = \overline{X4}X3X2X1$; $Y8 = X4\overline{X3X2X1}$.

Преобразуем полученные выражения к базису «И-НЕ»:

$Y3 = \overline{\overline{X4X3X2X1}} = \overline{\overline{X4}|\overline{X3}|\overline{X2}|X1}$; $Y4 = \overline{\overline{X4X3X2X1}} = \overline{\overline{X4}|X3|\overline{X2}|\overline{X1}}$;

$Y5 = \overline{\overline{X4X3\overline{X2}X1}} = \overline{\overline{X4}|X3|\overline{X2}|X1}$; $Y6 = \overline{\overline{X4X3X2\overline{X1}}} = \overline{\overline{X4}|X3|X2|\overline{X1}}$;

$Y7 = \overline{\overline{X4X3X2X1}} = \overline{\overline{X4}|X3|X2|X1}$; $Y8 = \overline{X4\overline{X3X2X1}} = X4|\overline{X3}|\overline{X2}|\overline{X1}$.

По полученным выражениям реализуем принципиальную схему дешифратора (рис.8.11) для перевода двоичных чисел в десятичный код в базисе «И-НЕ».

8.5. Мультиплексор

Мультиплексором (MS) называется комбинационное устройство, предназначенное для коммутации в желаемом порядке сигналов с нескольких входных шин на одну выходную. С помощью мультиплексора осуществляется временное разделение информации, поступающей по разным каналам.

Мультиплексоры имеют (как минимум) две группы входов и одну группу выходов. Входы делятся на: информационные, адресные и разрешающие. На первые подается информация, подлежащая передаче на выход. Адресные входы определяют нужный информационный вход. Если адресных входов n, то информационных входов 2^n. На разрешающий вход подается сигнал, разрешающий передачу информации с входа на выход. Наличие разрешающего входа позволяет синхронизировать работу мультиплексора с работой других устройств, а также наращивать его разрядность.

Каждый из входов данных имеет свой номер, задаваемый на адресных входах. Набор переменных, поступающих на адресные входы, задает двоичное число N_i вида $x_1x_2x_3...x_k$. При подаче сигнала на вход C выходная переменная мультиплексора Q повторяет переменную информационного входа D_{Ni} с номером N_i, задаваемым двоичным кодом на адресных входах. При отсутствии синхронизирующего сигнала (C = 0) связь между инфор-

мационными входами и выходом отсутствует (Q = 0). На рис.8.12 показано условное обозначение схемы мультиплексора.

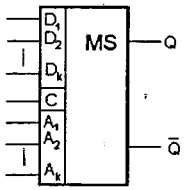


Рис.8.12. Условное обозначение схемы мультиплексора

Разработаем принципиальную схему синхронного мультиплексора на три адресных входа, работа которого определена таблицей истинности (ТИ) табл.8.2.

Таблица 8.2. Таблица истинности мультиплексора

Входы				*Выходы*
*A*1	*A*2	*A*3	*C*	*Q*
0	0	0	1	*D*0
0	0	1	1	*D*1
0	1	0	1	*D*2
0	1	1	1	*D*3
1	0	0	1	*D*4
1	0	1	1	*D*5
1	1	0	1	*D*6
1	1	1	1	*D*7

В соответствии с табл.8.2 составляем логическое выражение для выхода Q и строим принципиальную схему мультиплексора в базисе «И-ИЛИ-НЕ». Функция выхода в виде совершенной дизъюнктивной нормальной формы (СДНФ) получается как сложение конъюнкций каждой из строк ТИ, и имеет вид:

$$Q = C(D0\overline{A1}A2\overline{A3} \vee D1\overline{A1A2}A3 \vee D2\overline{A1}A2\overline{A3} \vee$$

$$\vee D3\overline{A1}A2A3 \vee D4A1\overline{A2A3} \vee D5A1\overline{A2}A3 \vee D6A1A2\overline{A3} \vee D7A1A2A3).$$

Используя полученную функцию, строим схему мультиплексора (рис.8.13.)

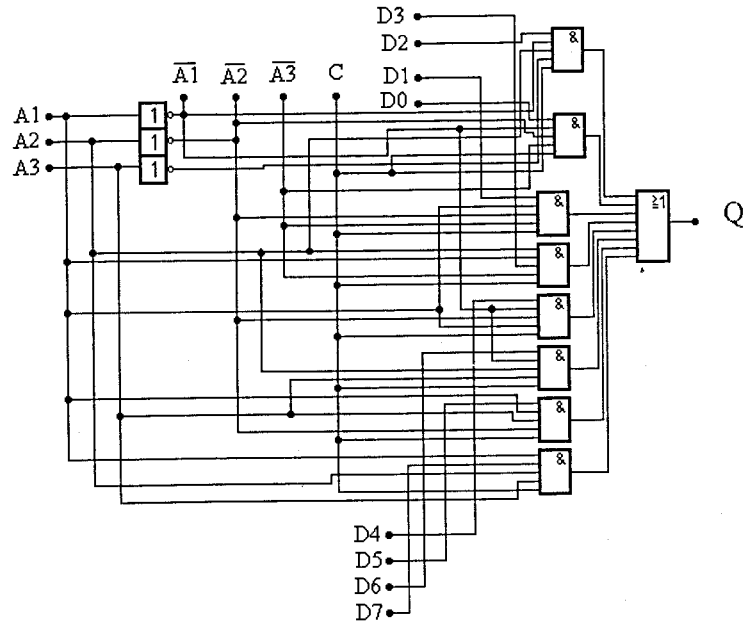


Рис.8.13. Принципиальная схема мультиплексора

8.6. Демультиплексор

Демультиплексор (DS) – это устройство (рис.8.14), предназначенное для коммутации входа данных D на один из выходов Q_1, Q_2, ..., Q_n, адрес которого выбирается при помощи управляющих входов A_1, A_2, ..., A_k (где $n = 2^k - 1$), при подаче сигнала синхронизации на вход C. Демультиплексор имеет принцип действия, обратный принципу действия мультиплексора. При *n*-разрядном адресе демультиплексор может иметь 2^n выходов.

Демультиплексор может использоваться как дешифратор. В этом случае на вход демультиплексора подается константа D = 1, а на адресные входы – принимаемая кодовая комбинация. В зависимости от значения переменных на адресных входах логическая «1» будет лишь на одном из выходов демультиплексора.

На рис.8.15, *а* показана схема демультиплексора, реализующего логические функции (рис.8.15, *в*), заданные таблицей истинности (рис.8.15, *б*).

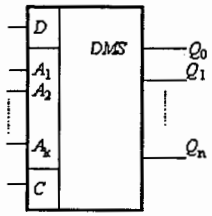
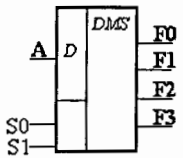


Рис.8.14. Демультиплексор



а)

S0	S1	F0	F1	F2	F3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

б)

$F_0 = A\overline{S_0}\,\overline{S_1}$

$F_1 = A\overline{S_0}S_1$

$F_2 = AS_0\overline{S_1}$

$F_3 = AS_0S_1$

в)

Рис.8.15. Реализация схемы демультиплексора

8.7. Преобразователи кодов

В цифровых устройствах часто возникает необходимость преобразования числовой информации из одного вида в другой, например, из двоично-десятичного кода в двоичный код. Эта процедура выполняется с использованием преобразователей кодов. Преобразователь кода – это комбинационное логическое устройство для автоматического перевода одной формы числа в другую. Наиболее распространены следующие два подхода к построению преобразователей кодов. Первый подход основан на синтезе m независимых одновыходных функций по заданной таблице истинности (таблице соответствия кодов). Второй подход – это построение преобразователя кодов по методу «*дешифратор-шифратор*». В качестве примера, иллюстрирующего второй подход построения преобразователей кодов, реализована схема «цифровой индикации».

На рис.8.16 показана принципиальная схема «цифровой индикации», выполненная на трехразрядном дешифраторе и восьмиразрядном шифраторе, а на рис.8.17, *а* приведена таблица истинности работы данного устройства.

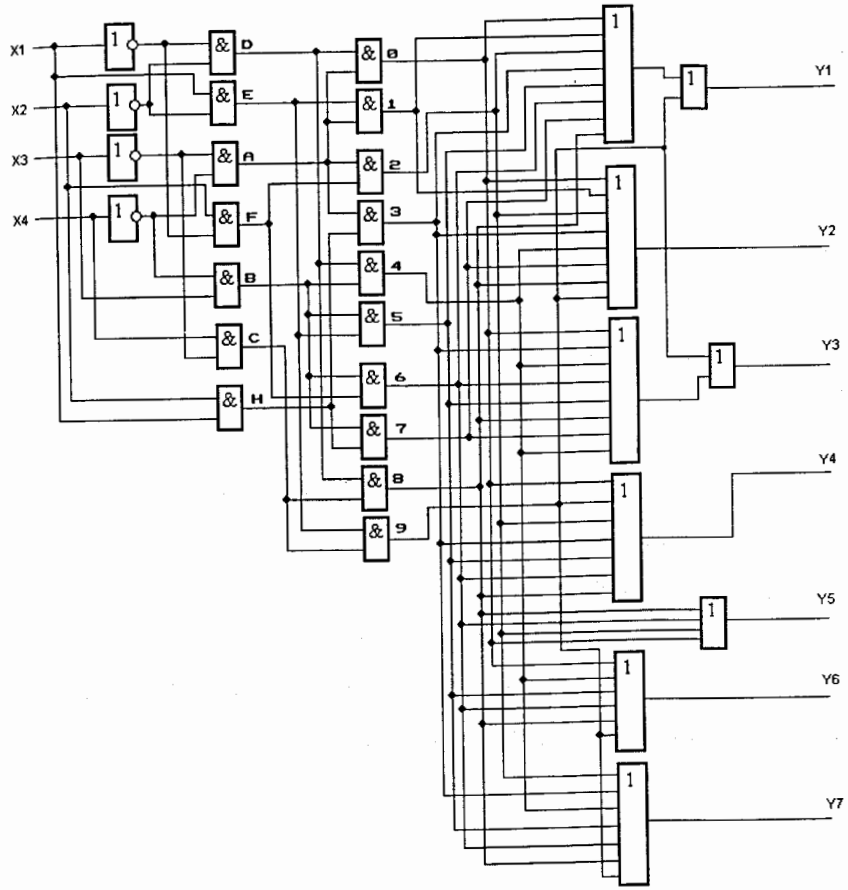
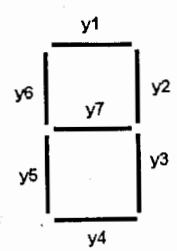


Рис.8.16. Схема устройства цифровой индикации

	X3 X2 X1 X0				y1	y2	y3	y4	y5	y6	y7
0	A	0	0	0 0 D	1	1	1	1	1	1	0
1		0	0	0 1 E	1	1	0	0	0	0	0
2		0	0	1 0 F	1	1	0	1	1	0	1
3		0	0	1 1 H	1	1	1	1	0	0	1
4	B	0	1	0 0 D	0	1	1	0	0	1	1
5		0	1	0 1 E	1	0	1	1	0	1	1
6		0	1	1 0 F	1	0	1	1	1	1	1
7		0	1	1 1 H	1	1	1	0	0	0	0
8	C	1	0	0 0 D	1	1	1	1	1	1	1
9		1	0	0 1 E	1	1	1	1	0	1	1

а)



б)

Рис.8.17. Таблица истинности схемы цифровой индикации

9. Последовательностные цифровые устройства

9.1. Триггер

9.1.1. Определение триггера

Триггер – это логическое устройство, предназначенное для хранения одного бита информации. Триггеры относятся к классу схем, значения выходных сигналов которых зависят не только от значений входных сигналов, но и от последовательности их изменения. Триггеры имеют два устойчивых состояния, которые устанавливаются импульсными или потенциальными сигналами. По способу записи информации триггеры подразделяются на *асинхронные* и *тактируемые*. В асинхронных триггерах момент переключения определяется моментом смены кодовой комбинации на информационных входах. В синхронных триггерах смена состояний осуществляется в строго определенные моменты времени действия тактирующих импульсов.

Отличительными особенностями триггеров являются: число внутренних устойчивых состояний, равное двум, чему соответствует одна переменная в прямой (Q) или инверсной форме ($\overline{Q}$); число выходов у триггера, также равное двум, один из них называют прямым (Q), другой инверсным ($\overline{Q}$), причем значения выходов равны соответствующим значениям внутренней переменной. Состояние триггера определяется по уровню напряжения на его прямом выходе. Если это напряжение уровня логической «1», то есть $Q = 1$, то говорят, что триггер находится в единичном состоянии или в триггер записана «1». Если же $Q = 0$ – триггер находится в нулевом состоянии (записан «0»).

9.1.2. Классификация триггеров

Классификация триггеров осуществляется по ряду признаков. Основным из них является признак *логического функционирования*, при использовании которого триггеры разделяют по виду *характеристического уравнения* (уравнению переходов). Еще одним важным классификационным признаком является *способ записи информации* в триггеры. Классификация триггеров по указанным признакам приведена на рис.9.1.

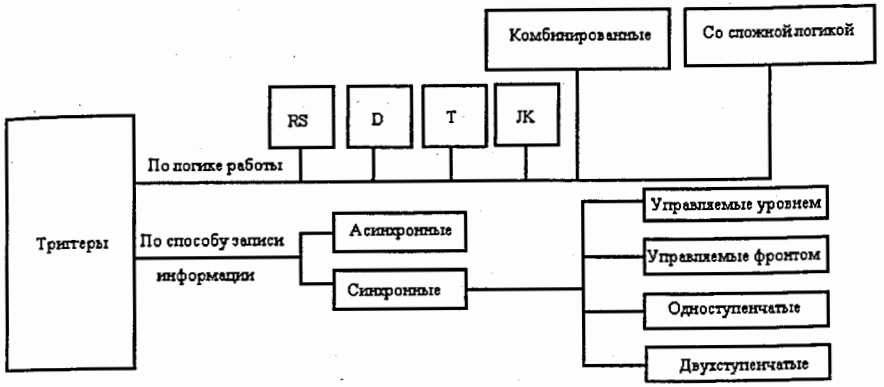


Рис.9.1. Классификация типов триггеров

По логическому функционированию различают триггеры: *RS-*, *D-*, *T-* и *JK*-типов. На рис.9.2 приведены условные обозначения таких триггеров. Кроме того, используются комбинированные триггеры, в которых совмещаются одновременно несколько типов, и триггеры со сложной входной логикой.

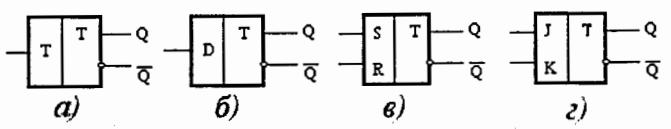


Рис.9.2. Графические обозначения *T-триггера* (*а*), *D-триггера* (*б*), *RS-триггера* (*в*) и *JK-триггера* (*г*)

T-триггер – это простейший триггер, имеющий только один информационный вход «Т». Он сохраняет свое состояние при подаче на его вход сигнала уровня «0» и изменяет состояние на противоположное при подаче входного сигнала уровня «1». Такой триггер еще называется *счетным* (со счетным входом).

Важным методом, используемым для описания функционирования триггера, является метод *таблиц состояний* (таблиц переходов), так как

таблица истинности состояний триггера отражает лишь динамику изменения состояния триггера и не учитывает свойство триггера запоминать единицу информации. Полная таблица состояний учитывает влияние значения предыдущего состояния триггера Q^k на его процесс управления, причем Q^k представляется как входная переменная. Таблицу состояний строят так же, как и таблицу истинности.

T^k	Q^k	Q^{k+1}
0	0	0
1	0	1
0	1	1
1	1	0

Рис.9.3. Таблица состояний *T-триггера*

По таблице состояний *T-триггера*, приведенной на рис.9.3, получаем выражение для характеристического уравнения *T-триггера*

$$Q^{k+1} = Q^k \overline{T}^k \vee \overline{Q}^k T^k, \qquad (9.1)$$
$$\overline{Q}^{k+1} = \overline{Q}^k \overline{T}^k \vee Q^k T^k.$$

Очевидно, что *T-триггер* реализует логическую функцию «*сумма по модулю 2*».

D-триггер (*D* – от слова *delay* – задержка) принимает информацию по входу «D», а его состояние повторяет входной сигнал, но с задержкой, определяемой тактовым сигналом С. Таблица состояний *D-триггера* показана на рис.9.4.

D^k	Q^k	Q^{k+1}
0	0	0
1	0	1
0	1	0
1	1	1

Рис.9.4. Таблица состояний *D-триггера*

Из таблицы состояний получаем характеристическое уравнение *D-триггера*

$$Q^{k+1} = D^k \overline{Q}^k \vee D^k Q^k = D^k, \qquad (9.2)$$
$$\overline{Q}^{k+1} = \overline{D}^k \overline{Q}^k \vee \overline{D}^k Q^k = \overline{D}^k.$$

Двухвходовые триггеры RS- и JK- типов устанавливаются (переключаются) в состояние «1» при подаче сигнала уровня «1» на один из входов, обозначаемый «S» (для *RS-триггеров*) или «J» (для *JK-триггеров*) и устанавливаются в состояние «0» при подаче сигнала уровня «1» на вход «R» (для *RS-триггеров*) или «K» (для *JK-триггеров*). Эти входные сигналы называются устанавливающими или переключающими сигналами. При отсутствии их на входах триггер сохраняет свое текущее состояние. Различия между *RS-* и *JK-триггерами* проявляются в их реакциях на одновременную подачу устанавливающих сигналов на оба установочных входа. Для *RS-триггера* такая комбинация входных сигналов является запрещенной, а *JK-триггер* меняет свое текущее состояние на противоположное. Таблица состояний *RS-* и *JK-триггеров* приведена на рис.9.5.

R^k / K^k	S^k / J^k	Q^k	Q^{k+1}	
			RS - тр.	JK - тр.
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	*	1
1	1	1	*	0

Рис.9.5. Таблица состояний *RS-* и *JK-триггеров*

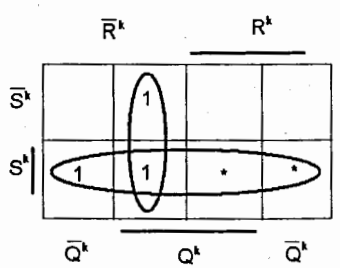


Рис.9.6. Карта Карно состояний *RS-триггера*

Из таблицы состояний (рис.9.5) можно получить выражение для характеристического уравнения *RS-триггера*. Для этого составляем карту Карно от входных переменных, при которых выходное состояние *RS-триггера* равно «1» ($Q^{k+1}=1$), и проводим соответствующие контуры (рис.9.6). В результате минимизации первый и второй контуры (горизонтальный и вертикальный) дают характеристическое уравнение *RS-триггера*

$$Q^{k+1} = S^k \vee \overline{R}^k \wedge Q^k, \qquad (9.3)$$

которое показывает, как меняется состояние триггера в зависимости от текущих значений состояния и входов.

По способу записи информации различают асинхронные (нетактируемые) и синхронные (тактируемые) триггеры. В асинхронных триггерах переход в новое состояние вызывается изменениями только входных информационных сигналов. Синхронные триггеры (рис.9.7), кроме информационных входов, имеют отдельный вход синхронизации, обычно обозначаемый буквой «С». Изменение состояния синхронного триггера может произойти при одновременном воздействии входных информационных сигналов и сигнала синхронизации.

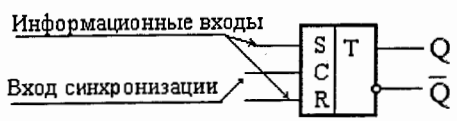


Рис.9.7. Синхронный *RSC-триггер*

По способу восприятия синхронизирующих сигналов триггеры делятся на управляемые уровнем и триггеры с динамическим управлением. Управление уровнем означает, что при одном уровне синхросигналов (С=1) триггер воспринимает входные информационные сигналы и реагирует на них, а при другом (С=0) не воспринимает и остается в неизменном состоянии (рис.9.8, *а*). При динамическом управлении (рис.9.8, *б*) разрешение на переключение триггера дается только в момент перепада синхросигнала (на фронте или срезе синхроимпульса). В остальное время действия синхросигнала, независимо от его уровня, триггер не воспринимает входные сигналы и, следовательно, остается в неизменном состоянии.

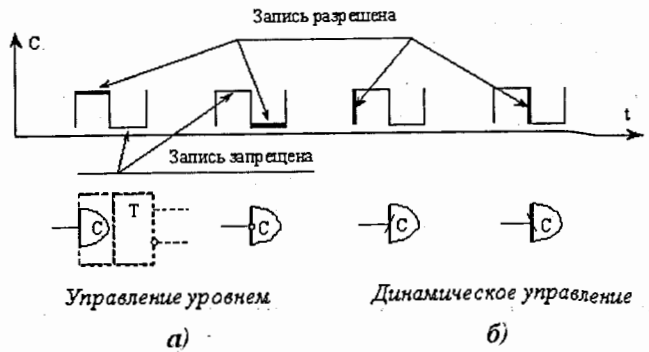


Рис.9.8. Временные диаграммы триггера, управляемого уровнем (*а*) и с динамическим управлением (*б*)

Синхронный вход в динамическом управлении может быть прямым или инверсным. При прямом входе разрешение на переключение триггера имеет место при изменении синхросигнала с уровня нуля до уровня единицы (положительным фронтом синхросигнала). Если у триггера используется инверсный синхронный вход, то его переключения возможны только при изменении синхросигнала с уровня единицы до уровня нуля (отрицательным фронтом синхросигнала). На диаграммах синхроимпульсов (рис.9.8) отмечено содержание процессов на отдельных участках, а под диаграммами приведены условные графические изображения синхронных входов для соответствующих типов триггеров.

По типу процесса переключения триггеры делятся на одноступенчатые и двухступенчатые. В одноступенчатом триггере переключение в новое состояние происходит за один такт, в двухступенчатом триггере – по этапам.

Двухступенчатый триггер состоит из ведущего (М) и ведомого (S) триггеров (рис.9.9, *а*). Переход в новое состояние происходит в обоих триггерах поочередно. Один из уровней синхросигнала разрешает прием информации

в *M-триггер*, при этом состояние *S-триггера* остается неизменным. Другой уровень синхросигнала разрешает передачу нового состояния *M-триггера* в *S-триггер* (рис.9.9, *б*).

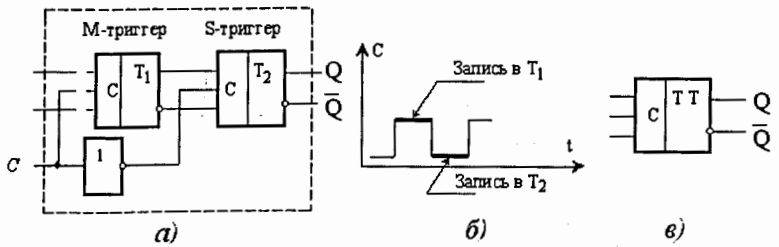


Рис.9.9. Структурная схема (*а*), временные диаграммы, поясняющие работу (*б*), и условное графическое изображение двухступенчатого триггера (*в*)

9.2. Схемотехника триггеров

9.2.1. Асинхронный RS-триггер

Схема асинхронного *RS-триггера* может быть построена на базе логического элемента (ЛЭ) из любого функционально полного набора. Однако оптимальное схемное решение получают при использовании функциональных наборов «И-НЕ» или «ИЛИ-НЕ».

Применив правило *де Моргана* к выражению характеристического уравнения *RS-триггера* (9.3), преобразуем его в базис «И-НЕ»

$$Q^{k+1} = \overline{S^k \vee \overline{R}^k \wedge Q^k} = \overline{\overline{S}^k \wedge \overline{\overline{R}^k \wedge Q^k}}, \qquad (9.4)$$
$$\overline{Q}^{k+1} = \overline{R^k \vee \overline{S}^k \wedge \overline{Q}^k} = \overline{\overline{R}^k \wedge \overline{\overline{S}^k \wedge \overline{Q}^k}}.$$

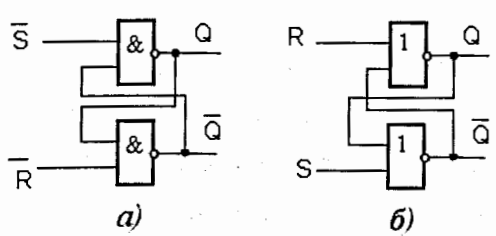


Рис.9.10. Схема асинхронного *RS-триггера*

Из анализа характеристического уравнения (9.4) очевидно, что простой *RS-триггер* можно сконструировать, соединив «крест-накрест» два элемента «И-НЕ», соответствующая схема приведена на рис.9.10, *а*.

Для построения схемы асинхронного *RS-триггера* в базисе «ИЛИ-НЕ» преобразуем характеристическое уравнение (9.3) к выбранному базису. Для этого, используя правило *де Моргана*, перепишем его в виде

$$Q^{k+1} = S^k \vee \overline{R^k \vee \overline{Q}^k},$$
$$\overline{Q}^{k+1} = R^k \vee \overline{S^k \vee Q^k}.$$

Проинвертировав эти соотношения, получаем выражения для асинхронного *RS-триггера* в базисе «ИЛИ-НЕ»:

$$\overline{Q}^{k+1} = \overline{\overline{S^k \vee R^k \vee \overline{Q}^k}}, \qquad (9.5)$$
$$Q^{k+1} = \overline{R^k \vee \overline{S^k \vee Q^k}}.$$

R^k	S^k	Q^{k+1}
0	0	Q^k
0	1	0
1	0	1
1	1	*

а)

R^k	S^k	Q^{k+1}
0	0	*
0	1	0
1	0	1
1	1	Q^k

б)

Рис.9.11. Таблица состояний асинхронных *RS-триггеров*, построенных *а)* – на ЛЭ «ИЛИ-НЕ» и *б)* – на ЛЭ «И-НЕ»

Схема, реализующая эти выражения, приведена на рис.9.10, *б*. Из схем на рис.9.10 видно, что при замене одних ЛЭ другими схема триггера не меняется, меняются местами только входы (или выходы) схемы. Асинхронный *RS-триггер* на ЛЭ «ИЛИ-НЕ» управляется прямыми входными сигналами «R и S», а на ЛЭ «И-НЕ» – инверсными сигналами «$\overline{R}$ и $\overline{S}$». При одновременной подаче переключающих сигналов на оба входа (R=S=1 для триггера на ЛЭ «ИЛИ-НЕ») или (R=S=0 для триггера на ЛЭ «И-НЕ») триггер

распадается на два автономных инвертора. При этом на его обоих выходах будет сигнал уровня «0» (для триггера на ЛЭ «ИЛИ-НЕ») или уровня «1» (для триггера на ЛЭ «И-НЕ»), в связи с чем схема «теряет» триггерные свойства. Поэтому указанные комбинации входных сигналов являются запрещенными. Таблица переходов для асинхронных *RS-триггеров*, построенных на «ИЛИ-НЕ» и «И-НЕ» элементах, приведена на рис.9.11.

9.2.2. Синхронный RS-триггер

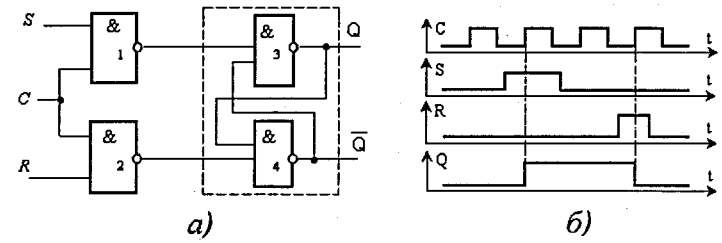
Для получения характеристического уравнения синхронного *RS-триггера* вводим в выражение (9.4) третью входную переменную – сигнал синхронизации «С». При С=1 триггер изменяет свое состояние в соответствии с логикой функционирования асинхронного триггера, а при С=0 состояние триггера остается неизменным, то есть

$$\begin{aligned} Q^{k+1} &= S^k \wedge C^k \vee Q^k(\overline{R}^k \vee \overline{C}^k), \\ \overline{Q}^{k+1} &= R^k \wedge C^k \vee \overline{Q}^k(\overline{S}^k \vee \overline{C}^k). \end{aligned} \qquad (9.6)$$

В базисе «И-НЕ» характеристическое уравнение синхронного *RS-триггера* имеет следующий вид:

$$\begin{aligned} Q^{k+1} &= \overline{\overline{S^k \wedge C^k} \wedge Q^k \wedge \overline{R^k \wedge C^k}}, \\ \overline{Q}^{k+1} &= \overline{\overline{R^k \wedge C^k} \wedge \overline{Q}^k \wedge \overline{S^k \wedge C^k}}. \end{aligned} \qquad (9.7)$$

Схема, реализующая эти уравнения, приведена на рис.9.12, *а*. Основой этой схемы является асинхронный *RS-триггер* на элементах 3 и 4, а элементы 1 и 2 образуют схему входной логики. При С=0 на выходах элементов 1 и 2 действуют единичные сигналы и асинхронный триггер, для которого эти сигналы являются входными, не изменяет своего состояния. Если С=1, то для сигналов S и R элементы 1 и 2 становятся инверторами и асинхронный триггер получает нулевой устанавливающий сигнал от входа, на котором действует единичный сигнал. Следовательно, устанавливающими (переключающими) сигналами для синхронного *RS-триггера* являются сигналы уровня логической «1». Временная диаграмма работы синхронного *RS-триггера* изображена на рис.9.12, *б*, а его таблица состояний – на рис.9.13.



а) *б)*

Рис.9.12. Схема синхронного *RS-триггера*

Режим работы	входы			выходы		
	C	S	R	Q	$\overline{Q}$	Влияние на выход Q
Хранение	⊓	0	0	Без изменений		Без изменений
Установка 1	⊓	0	1	0	1	Сброс или очистка в состояние 0
Установка 0	⊓	1	0	1	0	Сброс или очистка в состояние 1
Запрещенное состояние	⊓	1	1	1	1	Запрещено - не используется

Рис.9.13. Таблица состояний синхронного *RS-триггера*

9.2.3. Синхронный D-триггер

Синхронный *D-триггер* реализует задержку входного сигнала D с помощью тактирования, принимая сигнал только по разрешению тактового сигнала С. Из характеристического уравнения синхронного *D-триггера*

$$Q^{k+1} = \overline{C}^k \wedge Q^k \vee C^k \wedge D^k \qquad (9.8)$$

видно, что при наличии синхронизирующего сигнала (C^k=1) триггер переходит в состояние $D^k \rightarrow Q^{k+1} = C^k D^k = D^k$, а при его отсутствии ($C^k = 0$) триггер сохраняет свое состояние

$$Q^{k+1} = \overline{C}^k \wedge Q^k = Q^k. \qquad (9.9)$$

Таким образом, сигнал на выходе триггера в такте k+1 (Q^{k+1}) повторяет сигнал, который был на входе D в предыдущем такте k (D^k).

Схему синхронного *D-триггера* легко получить из схемы синхронного *RS-триггера* путем добавления инвертора, связывающего сигнал D с установочным R-входом синхронного *RS-триггера*. На рис.9.14, *а* показана функциональная схема синхронного *D-триггера*, а на рис.9.14, *б* схема синхронного *D-триггера*, управляемая по переднему фронту синхросигнала C.

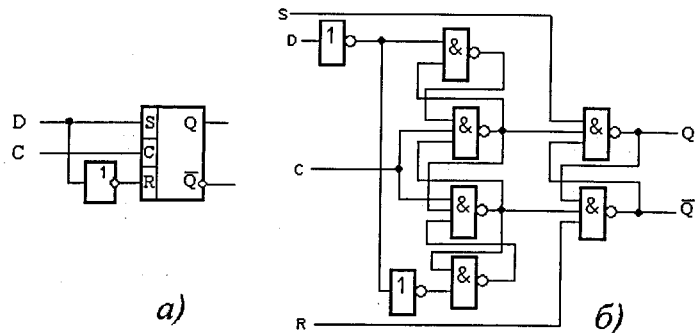


Рис.9.14. Схема *D-триггера* на базе синхронного *RS-триггера*

Условное графическое изображение синхронного *D-триггера* и его временные характеристики приведены на рис.9.15.

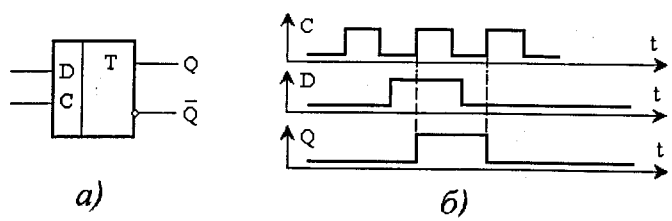


Рис.9.15. Условное графическое обозначение синхронного *D-триггера* (*а*) и его временная характеристика (*б*)

Серийно выпускаемые *D-триггеры* имеют два дополнительных асинхронных входа: вход предварительной установки «S» и вход очистки «R». Условное графическое обозначение *D-триггера* с двумя дополнительными входами показано на рис.9.16.

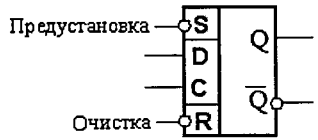


Рис.9.16. Обозначение *D-триггера* с дополнительными входами

Режим работы	Входы				Выходы	
	Асинхронный		Синхронный			
	S	R	C	D	Q	$\overline{Q}$
Асинхронная установка 1	0	1	X	X	1	0
Асинхронная установка 0	1	0	X	X	0	1
Запрещенное состояние	0	0	X	X	1	1
Установка 1	1	1	↑	1	1	0
Установка 0	1	1	↑	0	0	1

Рис.9.17. Таблица состояний *D-триггера* с дополнительными входами

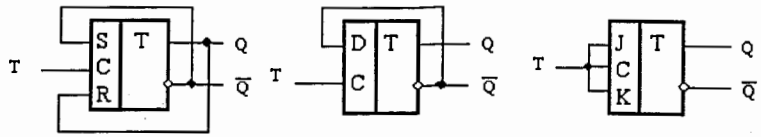
На рис.9.17 показана таблица состояний для *D-триггера* с дополнительными входами. Логический «0» на входе S инициирует установку логической «1» на выходе Q. Логический «0» на входе R инициирует установку логического «0» на выходе Q (*очистку* выхода Q). В активных состояниях входы S и R блокируют действия входов D и C. При разблокировании входы D и C действуют как в обычном *D-триггере*.

9.2.4. Т-триггер

Т-триггером называется триггер со счётным входом, в котором текущее состояние изменяется на противоположное каждый раз, когда на его вход приходит очередной тактовый сигнал. Обозначение *Т-триггера* произошло от первой буквы английского слова *toggle* – защёлка. *Т-триггер* имеет один вход T (счётный вход триггера) и два выхода Q и $\overline{Q}$.

Т-триггер легко может быть получен из синхронного *RS-триггера* (рис.9.18, *а*). Однако на практике *Т-триггер* получают из схем синхронного *D-* или *JK*-триггеров. Если обозначить вход синхронизации *D-триггера* через

T, а его инверсный выход соединить с входом D, то есть сделать $D=\overline{Q}$ (рис.9.18, *б*), то характеристическое уравнение *D-триггера* (9.2) примет вид, соответствующий уравнению *T-триггера* (9.1). Для преобразования *JK-триггера* в *T-триггер* достаточно объединить его входы J, K и C (рис.9.18, *в*).



а) *б)* *в)*

Рис.9.18. Преобразование *RS-(а)*, *D-(б)* и *JK-(в)* триггеров в *T-триггер*

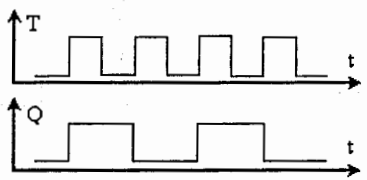


Рис.9.19. Временная диаграмма работы *T-триггера*

Характерной особенностью *T-триггеа* является то, что частота изменения выходных сигналов в два раза меньше частоты входных (рис.9.19). Это свойство *T-триггера* используется при построении на их основе делителей частоты следования импульсов и двоичных счетчиков. На рис.9.19 показана временная диаграмма работы *T-триггера*.

9.2.5. JK-триггер

JK-триггер – наиболее широко используемый универсальный триггер, обладающий характеристиками всех других типов триггеров. *JK-триггер* в отличие от *RS-триггера* не имеет запрещенных комбинаций входных сигналов, которые следует исключать при работе цифровых систем. На рис.9.20 показано условное графическое обозначение *JK-триггера*.

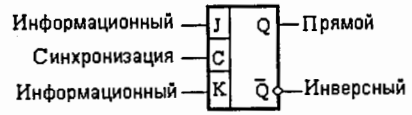


Рис.9.20. Условное графическое обозначение *JK-триггера*

Режим работы	Входы			Выходы		
	C	J	K	Q	$\overline{Q}$	Влияние на выход Q
Хранение	⊓	0	0	Без изменений		Без изменений - блокировка
Установка 1	⊓	1	0	0	1	Сброс или очистка в состояние 0
Установка 0	⊓	0	1	1	0	Установка в состояние 1
Запрещенное состояние	⊓	1	1	Переключается		Изменение состояния на противоположное

Рис.9.21. Таблица состояний работы *JK-триггера*

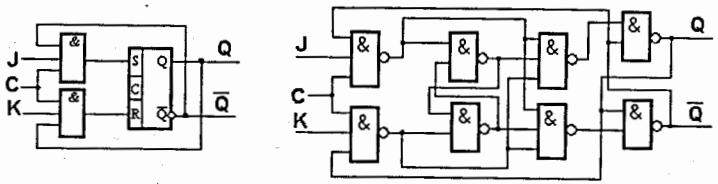
Рассмотрим таблицу состояний *JK-триггера* (рис.9.21), иллюстрирующую принципы его работы. Из таблицы видно, что когда на оба входа J и K подается уровень логического «0», триггер *блокируется* и состояния его выходов *не изменяются*. В этом случае триггер находится в режиме хранения. Вторая и третья строки (рис.9.21) описывают режимы, соответствующие установке триггера в состояние «0» и «1». Четвертая строка соответствует переключательному (счетному) режиму работы *JK-триггера*. Если на обоих входах J и K установлен уровень логической «1», то следующие друг за другом тактовые импульсы будут вызывать изменение уровней сигналов на выходах триггера от «1» к «0», от «0» к «1» и т. д.

Из таблицы состояний *JK-триггера* (рис.9.5) выводим характеристическое уравнение его работы

$$Q^{k+1} = \overline{J} \wedge \overline{K} \wedge Q^k \vee J \wedge \overline{K} \wedge \overline{Q}^k \vee J \wedge \overline{K} \wedge Q^k \vee J \wedge K \wedge \overline{Q}^k = \\ = Q^k(\overline{J} \wedge \overline{K} \vee J \wedge \overline{K}) \vee \overline{Q}^k(J \wedge \overline{K} \wedge Q^k \vee J \wedge K) = J\overline{Q}^k + \overline{K}Q^k, \qquad (9.10)$$

которое может быть реализовано с использованием двух элементов «И» и *RS-триггера* (рис.9.22, *а*) или в базисе «И-НЕ» (рис.9.22, *б*). *JK-триггер*

может иметь два дополнительных асинхронных входа – предварительной установки S и очистки R (рис.9.23).



а) *б)*

Рис.9.22. Схемная реализация *JK-триггера*

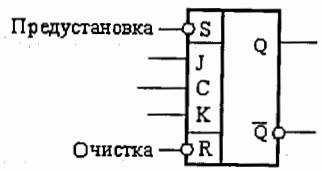


Рис.9.23. Условное графическое обозначение *JK-триггера*

Режим работы	Входы					Выходы	
	Асинхронные		Синхронные				
	S	R	C	J	K	Q	$\overline{Q}$
Асинхронная установка 1	0	1	X	X	X	1	0
Асинхронная установка 0	1	0	X	X	X	0	1
Запрещенное состояние	0	0	X	X	X	1	1
Хранение	1	1	⎍	0	0	Без изменений	
Установка 0	1	1	⎍	0	1	0	1
Установка 0	1	1	⎍	1	0	1	0
Переключение	1	1	⎍	1	1	Противоположное состояние	

Рис.9.24. Таблица состояний *JK-триггера* с дополнительными входами

На рис.9.23 показано условное обозначение *JK-триггера*, а на рис.9.24 – таблица состояний работы *JK-триггера* с дополнительными входами. Асинхронные входы S и R в активных состояниях блокируют действия синхронных входов. Активным состояниям асинхронных входов соответствуют первые три строки таблицы состояний *JK-триггера* (рис.9.24). В этих режимах синхронные входы блокированы и их состояния

не влияют на состояние выходов триггера, поэтому для входов J, K и C в этих строках поставлен знак «X» (любое состояние). Одновременная подача на оба асинхронных входа активного уровня сигнала (логического «0») соответствует запрещенному состоянию. При блокировании обоих асинхронных входов S и R уровнем логической «1» работу триггера контролируют синхронные входы, как показано в четырех нижних строках таблицы состояний *JK-триггера* (рис.9.24).

На основе *JK-триггеров* можно реализовать основные типы триггеров, например, *D-триггер*. Для преобразования в *D-триггер* достаточно объединить через инвертор вход J со входом K (рис.9.25).

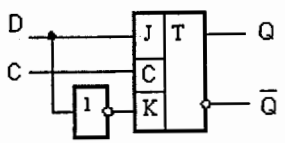


Рис.9.25. Преобразование *JK-триггера* в *T-триггер*.

9.3. Двухступенчатые триггеры

9.3.1. Двухступенчатые триггеры на базе синхронных RS-триггеров

Для надёжной работы триггерных ячеек в многоразрядных устройствах (регистрах, счётчиках) используются двухступенчатые триггеры, называемые *MS-триггерами* (*master/slave* – ведущий/ведомый или мастер/помощник). В этих триггерах входная и выходная ступени тактируются асинхронно, то есть прием информации разрешается поочередно. Это позволяет реализовать любые типы триггеров без режимов генерации (то есть без помех и сбоев) и дает возможность построения синхронных автоматов без опасных временных совпадений, нарушающих их работу.

Два варианта схемы такого триггера, состоящего из двух синхронных *RS-триггеров*, показаны на рис.9.26, а на рис.9.27 показана временная диаграмма их управления. В первом варианте двухступенчатого *RS-триггера* (рис.9.26, *а*) антисинхронное тактирование ступеней очевидно, поскольку пе-

резапись состояния *ТМ-триггера* (мастера) в *ТП-триггер* (помощник) производится следующим синхроимпульсом. Во втором варианте (рис.9.26, *б*) ступени идентичны по синхровходам, а для их антисинхронного управления включен инвертор. Теоретически в этом варианте возможны сбои – если триггер переключится быстрее, чем синхросигнал пройдет через инвертор, но на практике это не произойдет, так как временная задержка модуля инвертора меньше чем задержка переключения триггера.

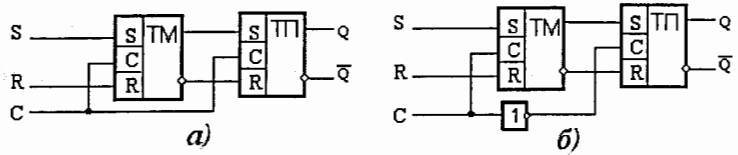


Рис.9.26. Схема двухступенчатого *RS-триггера* (*а*) и двухступенчатого *RS-триггера* с разнополярным управлением (*б*)

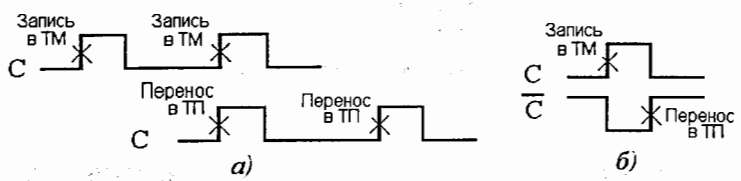


Рис.9.27. Временная диаграмма управления двухступенчатого *RS-триггера* (*а*) и двухступенчатого *RS-триггера* с разнополярным управлением (*б*)

9.3.2. Двухступенчатые JK-триггеры

Двухфазный способ управления полным тактовым импульсом применяется и для двухступенчатых *JK-триггеров*. Этот тип триггера, как и простой *JK-триггер*, имеет обратные связи с выходов на входы, исключающие неопределённое логическое состояние (рис.9.28).

На рис.9.28 показана схема двухступенчатого с полным тактовым импульсом *JK-триггера*, а на рис.9.29 – временная диаграмма переключающего импульса, на которой отмечены этапы работы составного триггера. Из диаграммы на рис.9.29 видно, что составным триггером «ТМ-ТП»

управляет полный (с фронтом и срезом) тактовый импульс «С». В момент t_1 триггер ТП изолирован от ТМ, а в момент t_2 разрешается приём данных входами ТМ триггера (рис.9.29). Следовательно, входная комбинация будет записана в триггер ТМ в момент прихода положительного перепада тактового импульса С. Когда придёт отрицательный перепад входного импульса С, на выходе инвертора он появится как положительный и в момент t_4 перепишет данные от выходов Q и $\overline{Q}$ в триггер ТП.

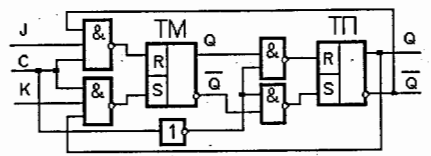


Рис.9.28. Схема двухступенчатого *JK-триггера*

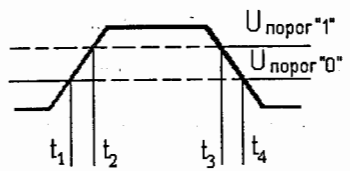


Рис.9.29. Схема управляющих фронтов тактирующего импульса

Основное свойство двухступенчатого триггера состоит в том, что входы приёма данных за период тактового импульса, то есть во время загрузки 1-го бита информации, не имеют сквозной связи с выходными цепями. Изоляция входов от выходов обеспечивает устойчивое переключение триггера, даже если частота тактовых импульсов нестабильна.

10. Последовательностные цифровые устройства

10.1. Счетчики и пересчетные устройства

10.1.1. Общие сведения

Счетчики – это последовательностные цифровые устройства, обеспечивающие хранение слова информации и выполнение над ним микрооперации счета. Микрооперация счета состоит в изменении хранимого зна-

чения слова информации (состояния счетчика) на «1». Счетчик, на котором реализуется микрооперация «C=C+1», называется суммирующим. Если на счетчике реализуется микрооперация «C=C−1», то это вычитающий счетчик. Счетчик, на котором реализуются обе указанные микрооперации, является реверсивным.

Пересчетным устройством называется последовательное цифровое устройство, обеспечивающее хранение слова информации и выполнение над ним микрооперации произвольной смены состояния. Вследствие этого определения «суммирующий», «вычитающий» и «реверсивный» неприменимы к пересчетным устройствам.

Счетчики и пересчетные устройства применяются: для образования адресов слов в запоминающих устройствах; адресов команд в устройствах управления; выработки специальных последовательностей управляющих сигналов; подсчета числа циклов сложения и вычитания; построения распределителей импульсов; в качестве делителей частоты и т. д. В процессе работы счетчик последовательно изменяет свои состояния. Длину списка разрешенных состояний называют модулем (основанием) счета или емкостью счетчика k. Это максимальное число единичных сигналов, которое может быть сосчитано счетчиком. При этом одно из возможных состояний счетчика принимается за начальное. Если счетчик начал работать от начального состояния, то каждый входной сигнал, кратный k, снова устанавливает счетчик в начальное состояние, а на выходе счетчика появляется сигнал k-ичного переноса (в суммирующем счетчике) или заема (в вычитающем счетчике).

10.1.2. Классификация счетчиков

Счётчики классифицируют: по значению модуля, по направлению счёта, по способу организации межразрядных связей, и по способу подачи тактового импульса.

По значению модуля счёта различают двоичные (модуль = 2^n), двоично-кодированные (с произвольным модулем, но кодированием состояний двоичными кодами) и счётчики с одинарным кодированием.

По направлению счёта счётчики делятся: на суммирующие (прямого счёта), вычитающие (обратного счёта) и реверсивные (с изменением направления счёта).

По способу организации межразрядных связей различают счётчики с последовательным, параллельным и комбинированным переносами. Параллельные счётчики называют синхронными, а последовательные – асинхронными.

10.2. Асинхронные счётчики

Цифровую схему, выполняющую функцию счёта, можно собрать на триггерах. Рассмотрим схему асинхронного двоичного счётчика по «модулю 16». Процедура двоичного и десятичного счёта показана рис.10.1.

Двоичный счет				Десятичный счет
D	C	B	A	
8	4	2	1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Рис.10.1. Таблица двоичного и десятичного счета

Используя 4 двоичных разряда (D, C, B и A), можно считать от «0000» до «1111» в двоичной системе счисления (то есть от 0 до 15 в десятичной системе). Столбец A соответствует самому младшему разряду, а столбец D самому старшему разряду. Следовательно, если нужен счётчик, считающий от «0000» до «1111», у него должно быть 16 различных выходных состояний, то есть нужен счётчик с модулем 16. Из рис.10.1 видно, что цифры (1 или 0) в столбце A изменяются на каждом шаге счёта, то есть триггер T1 переключается с приходом каждого нового тактового импульса. Триггер T2 (столбец B) переключается в два раза реже триггера T1, а каждый более старший разряд переключается в два раза реже предыдущего (столбцы C и D).

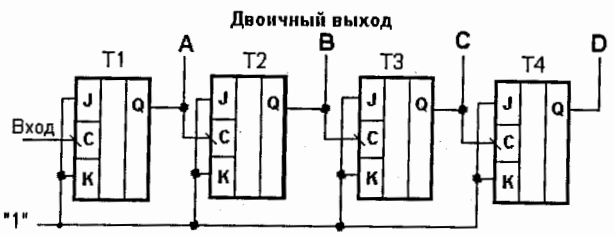


Рис.10.2. Схема двоичного счётчика по модулю 16

На рис.10.2 показана схема двоичного счётчика по модулю 16, составленная из четырех *JK-триггеров* в соответствии с таблицей двоичного (рис.10.1). Каждый *JK-триггер* работает в режиме переключения (J=K=1). Пусть в начальный момент состояние выходов счётчика соответствует двоичному числу «0000» (счётчик очищен). При поступлении тактового импульса «1» на синхронизирующий вход C триггера T1 этот триггер переключается (при прохождении среза импульса) и на выходе появляется двоичное число «0001». Тактовый импульс 2 возвращает триггер T1 в исходное состояние «0» (Q=0), что, в свою очередь, приводит к переключению триггера T2 в состояние «1» (Q=1). На выходе появится число «0010». Счёт продолжается: срез сигнала на выходе каждого триггера запускает следующий триггер.

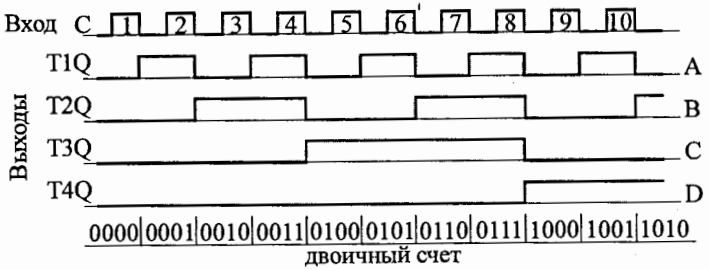


Рис.10.3. Временные диаграммы работы счетчика по модулю 16

На рис.10.3 показана временная диаграмма работы счётчика в процессе счёта до 10 (двоичное число «1010»). Синхронизирующему входу соответствует верхняя диаграмма, а ниже приведены диаграммы для выходов Q триггеров T1, T2, T3, T4. Внизу под диаграммами указаны двоичные числа, соответствующие различным состояниям счётчика. Анализируя диаграммы на рис.10.3, можно сделать вывод, что тактовые импульсы запускают только триггер T1, триггер T1 запускает триггер T2, триггер T2 запускает триггер T3 и т. д. Каждый триггер воздействует только на один (следующий за ним триггер), поэтому для переключения всех триггеров необходимо некоторое время. Например, на импульсе 8 (рис.10.3) тактовый импульс запускает триггер T1, вызывая его переключение в состояние «0». Это, в свою очередь, приводит к переключению триггера T2 из состояния «1» в состояние «0». Затем точно так же переключается триггер T3. В момент установки уровня логического «0» на выходе Q триггера T3 запускается триггер T4, который переключается из состояния «0» в состояние «1». Таким образом, изменение состояний последовательно распространяется по цепочке триггеров. Рассматриваемый счётчик называют асинхронным, поскольку предыдущий триггер вырабатывает для последующего тактовые импульсы. По направлению счёта этот счётчик является суммирующим (прямого счёта).

10.3. Асинхронные счётчики по модулю 10

Счётчик по модулю 10 считает от «0000» до «1001» (от 0 до 9 в десятичной системе). Для построения такого счётчика также необходимо четыре триггера, но он должен иметь обратные связи, останавливающие счёт при коде «9=1001». На рис.10.4 показана схема счётчика по модулю 10, в которую кроме четырех триггеров включён логический элемент «И-НЕ», для установки всех триггеров в нулевое состояние (очистки счетчика) с приходом десятого импульса.

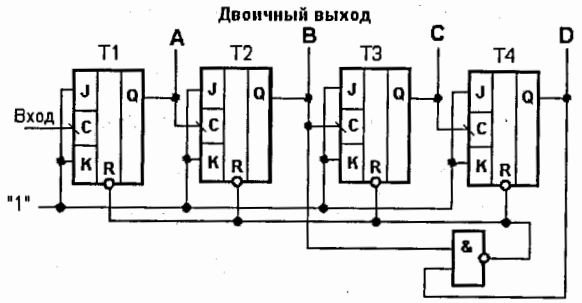


Рис.10.4. Схема асинхронного двоичного счетчика по модулю 10

Рассмотрим принцип работы данной схемы. Из таблицы на рис.10.1 видно, что за числом «1001» следует «1010» (10 в десятичной системе). При подаче логической «1», содержащейся в разрядах двоичного числа «1010», на входы элемента «И-НЕ» этот элемент подаст логический «0» на входы R всех триггеров. Таким образом, все триггеры установятся в состояние «0» и счетчик снова начнет считать от «0000» до «1010». Подобное использование логического элемента «И-НЕ» позволяет создавать счетчики с другими значениями модуля. Счетчик, изображенный на рис.10.3, называют также декадным (десятичным) счетчиком.

10.4. Синхронные счетчики

В синхронных счетчиках все триггеры получают тактовый импульс одновременно, поскольку их тактовые входы соединяются параллельно.

Такие триггеры переключаются практически одновременно. В асинхронных счетчиках каждый триггер вносит в процесс счета определенную задержку, поэтому младшие разряды результирующего кода появляются на выходах триггеров не одновременно (не синхронно с соответствующим тактовым импульсом). Например, для четырехразрядного асинхронного счетчика код «1111» появится на выходах триггеров уже после того, как поступит шестнадцатый тактовый импульс, то есть код «1111» сформируется не одновременно.

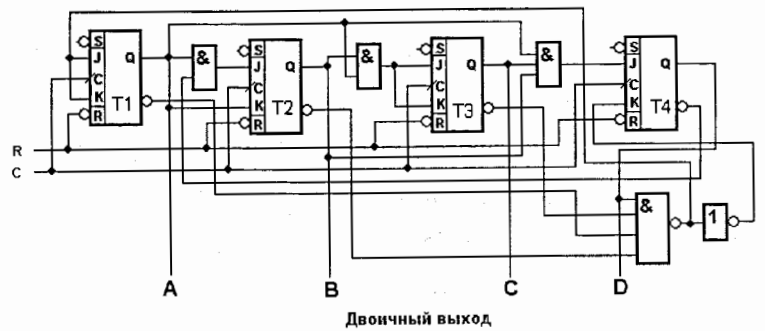


Рис.10.5. Схема синхронного двоичного счетчика по модулю 9

Таблица 10.1. Счетная последовательность импульсов

Строка	Номер тактового импульса	Двоичная счетная последовательность				Десятичные числа
		D	C	B	A	
1	0	0	0	0	0	0
2	1	0	0	0	1	1
3	2	0	0	1	0	2
4	3	0	0	1	1	3
5	4	0	1	0	0	4
6	5	0	1	0	1	5
7	6	0	1	1	0	6
8	7	0	1	1	1	7
9	8	1	0	0	0	8

На рис.10.5 показана схема четырехразрядного двоичного счетчика по модулю 9. Все синхронизирующие входы триггеров С соединены параллельно, то есть тактовые импульсы поступают непосредственно на син-

хронизирующий вход каждого триггера. Последовательность двоичных чисел, принимаемая счетчиком за один цикл счета, приведена в табл.10.1.

Рассмотрим принцип работы данного счетчика в течение одного цикла счета (табл.10.1). На каждом шаге цикла входной импульс поступает на синхронизирующий вход каждого триггера.

Импульс 1(строка 2) – переключается только триггер Т1, поскольку только у него на входах J и K действует уровень логической «1». Триггер Т1 переходит из состояния «0» в состояние «1». Результат: на выходе счетчика «0001».

Импульс 2 (строка 3) – переключаются два триггера Т1 и Т2, так как на входах J и K этих триггеров действует уровень логической «1». Триггер Т1 переходит из состояния «1» в состояние «0», а Т2 – из состояния «0» в состояние «1». Результат: на выходе счетчика «0010».

Импульс 3 (строка 4) – переключается один триггер Т1 (переходит из состояния «0» в состояние «1»), а Т2 не переключается, поскольку на его входах J и K действует уровень логического «0». Результат: на выходе «0011».

Импульс 4 (строка 5) – триггеры меняют свое состояние на противоположное, Т1 и Т2 переходят из «1» в «0», Т3 переключается из «0» в «1». Результат: на выходе счетчика «0100».

Импульс 5 (строка 6) – триггер Т1 переходит из состояния «0» в состояние «1». Результат: на выходе счетчика «0101».

Импульс 6 (строка 7) – переключаются два триггера, Т1 переходит из «1» в «0», а Т2 – из «0» в «1». Результат: на выходе счетчика «0110».

Импульс 7 (строка 8) – триггер Т1 переходит из состояния «0» в состояние «1». Результат: на выходе «0111».

Импульс 8 (строка 9) – все триггеры меняют свое состояние, переходя из «1» в «0». Результат: на выходе «1000».

Следует заметить, что в данном счетчике *JK-триггеры* используются как в режиме переключения (J=K=1), так и в режиме блокировки (J=K=0).

10.5. Вычитающие счетчики

Помимо суммирующих счетчиков (прямого счета), рассмотренных выше, существуют счетчики, которые считают в обратном направлении – вычитающие. Рассмотрим схему асинхронного вычитающего счетчика по модулю 8 (рис.10.6).

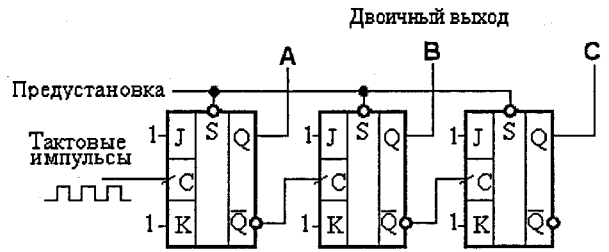


Рис.10.6. Схема асинхронного вычитающего счетчика по модулю 8

Таблица 10.2. Счетная последовательность импульсов

Номер тактового импульса	Двоичная счетная последовательность			Десятичные числа
	C	B	A	
0	1	1	1	7
1	1	1	0	6
2	1	0	1	5
3	1	0	0	4
4	0	1	1	3
5	0	1	0	2
6	0	0	1	1
7	0	0	0	0
8	1	1	1	7
9	1	1	0	6

Отличие данной схемы от схемы суммирующего счетчика состоит в способе переноса сигнала от триггера к триггеру. В суммирующем счетчике синхронизирующий вход каждого триггера связан с прямым выходом Q предыдущего триггера. В вычитающем счетчике синхронизирующий вход каждого триггера связан с инверсным выходом $\overline{Q}$ предыдущего триггера. В

счетчике, изображенном на рис.10.6, перед началом счета в обратном направлении предусмотрена предварительная его установка в состояние «111» (десятичное число 7) с помощью входа предустановки «S». Счетная последовательность двоичных чисел приведена в табл.10.2.

10.6. Самоостанавливающиеся счетчики

Вычитающий счетчик, схема которого показана на рис.10.6, является счетчиком циклического типа. Когда этот счетчик приходит в состояние «000», он снова начинает счет с двоичного числа «111». На практике в некоторых случаях нужны счетчики, которые останавливаются, когда исчерпывается вся счетная последовательность.

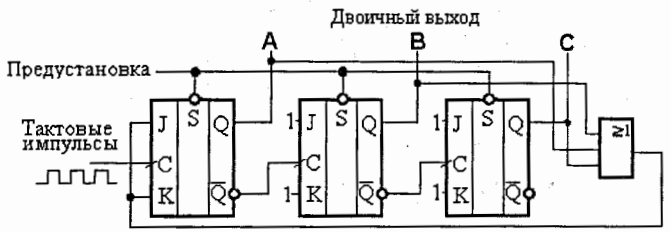


Рис.10.7. Схема самоостанавливающегося счетчика по модулю 8

Рассмотрим, какие изменения нужно внести в схему вычитающего счетчика, чтобы счет прекращался при достижении состояния 000. Из схемы на рис.10.7 видно, что для этого нужно ввести логический элемент «ИЛИ», который будет устанавливать на входах J и K триггера T1 уровень логического «0», когда на выходах (C, B, A) счетчика появится сигнал «000». Если нужно начать новый цикл счета с двоичного числа «111», на вход предустановки S следует подать уровень логического «0».

Используя один логический элемент или их комбинацию, можно останавливать счет в прямом и обратном направлении на любом наперед заданном двоичном числе. Для этого выход логического элемента надо присоединить к входам J и K первого триггера в асинхронном счетчике. При этом триггер T1 переводится в режим хранения и счет останавливается.

10.7. Счетчики – делители частоты

Одной из функций, которую выполняют счетчики в цифровых системах, является деление частоты. Пример простой системы с делителем частоты показан на рис.10.8. Эта система составляет основу цифровых часов. Периодический сигнал электросети с частотой 50 Гц, сформированный в виде последовательности прямоугольных импульсов, подается на вход системы, которая делит частоту на 50. На выходе схемы имеем последовательность прямоугольных импульсов с частотой 1 Гц (1 импульс в 1 секунду), то есть является таймером секунд.

Рис.10.8. Система с делителем частоты

На рис.10.9 схематически изображен декадный счетчик, а на рис.10.10 приведены временные диаграммы для его синхронизирующего входа C и выхода Q_D, соответствующего двоичному разряду 2^4.

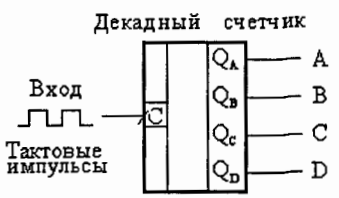


Рис.10.9. Схема декадного счетчика

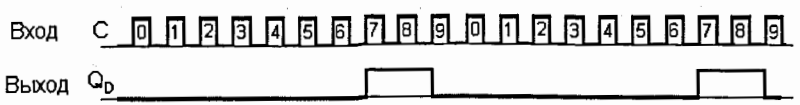


Рис.10.10. Временные диаграммы работы декадного счетчика

Из рис.10.10 видно, что 20 импульсов на входе счетчика преобразуются в два выходных импульса. Выполняется деление 20/2=10. Снимая

сигнал с входа Q_D, декадного счетчика получим счетчик-делитель на 10, так как частота выходного сигнала составляет 1/10 частоты на входе счетчика. Последовательно соединяя рассмотренный декадный счетчик (счетчик-делитель на 10) и счетчик по модулю 5 (счетчик-делитель на 5), получим схему, осуществляющую деления входной частоты на 50. Структура такой схемы показана на рис.10.11.

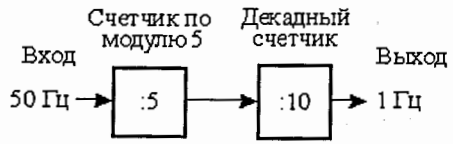


Рис.10.11. Структурная схема делителя частоты на 50

Последовательность прямоугольных импульсов с частотой 50 Гц поступает на вход счетчика-делителя на 5, а с его выхода с частотой 10 Гц подается на вход счетчика-делителя на 10. На выходе схемы получим сигнал с частотой 1 Гц. Функция деления частоты используется в таких цифровых устройствах, как частотомер, осциллограф и т. п.

10.8. Интегральные схемы счетчиков

На рис.10.12 приведена схема четырехразрядного двоичного счетчика-делителя на 2, на 6 и на 12 (аналог микросхемы К155ИЕ4). Если подать тактовые импульсы с частотой F на вход C1, то на выходе A получим частоту F/2. Тактовые импульсы с частотой F на входе C2 запускают делитель на 6, и на выходе D имеем частоту F/6. При этом на выходах B и C имеем импульсы с частотой F/3. На входы R1 и R2 подаются команды сброса.

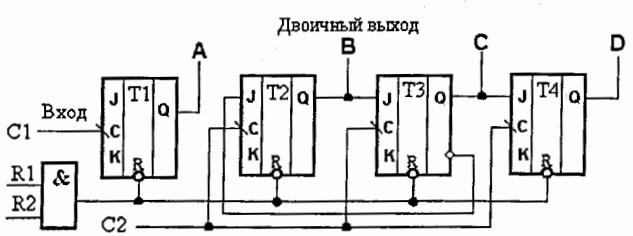


Рис.10.12. Схема четырехразрядного двоичного счетчика

Для построения счетчика с модулем деления 12 требуется объединить делители на 2 и на 6, соединив выход A со входом C2. На вход C1 подается входная частота F, на выходе D получаем последовательность импульсов с частотой F/12.

10.9. Проектирование счетчиков

В качестве примера разработаем схему суммирующего синхронного (параллельного) декадного счетчика по модулю 10 на *JK-триггерах*. Следует отметить, что синхронные счетчики обычно строятся на базе *RS-*, *JK-*, *D-* триггеров, синхронизируемых фронтом.

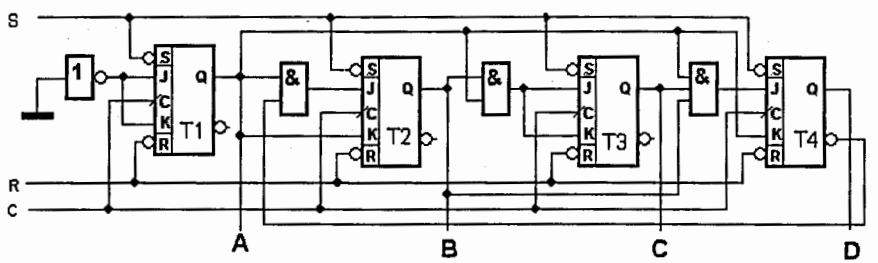


Рис.10.13. Схема счетчика по модулю 10 на *JK-триггерах*

Для реализации счетчика нам потребуется четыре триггера. Чтобы получить структуру с минимальным числом триггеров, принимаем m=4 (четырехразрядный счетчик). При этом $2^m - M = 2^4 - 10 = 6$ состояний счетчика будут нештатными. Схема такого счетчика приведена на рис.10.13. Она реализована на четырех триггерах и трех логических элементах «И». Счетчик является параллельным, так как все триггеры переключаются синхронно.

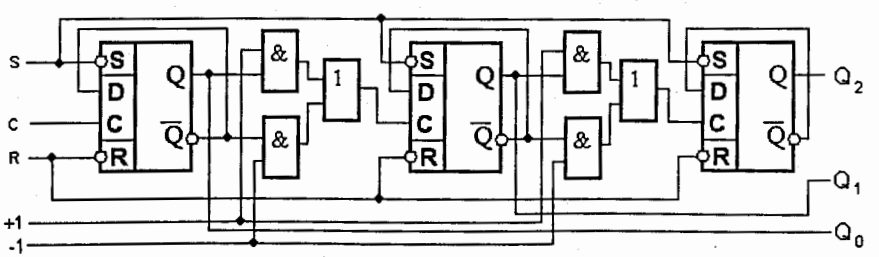


Рис.10.14. Схема реверсивного двоичного счетчика

На рис.10.14 показана принципиальная схема двоичного реверсивного счетчика, выполненного на трех *D–триггерах*. Изменение управления счета организовано на одноразрядном мультиплексоре, выполненном на двух элементах «И» и одном элементе «ИЛИ».

11. Регистры

11.1. Классификация регистров

Регистрами называются последовательностные цифровые устройства, выполняющие функции приема, хранения и передачи информации. Информация в регистре хранится в виде двоичного кода, то есть представлена комбинацией сигналов логического «0» и логической «1». Каждому разряду кода, записанному в регистр, соответствует свой разряд регистра, выполненный, как правило, на основе триггеров *RS*-, *D*- или *JK*-типа. Другими словами, регистр – это цепочка триггеров для запоминания одного двоичного числа. Общее количество триггеров равно наибольшей разрядности хранимого числа.

В цифровой электронике регистры выполняют ряд операций с информацией, представленной в виде многоразрядного двоичного кода. К этим операциям относятся: хранение, сдвиг в разрядной сетке, поразрядные логические операции и выдача числовых слов в определенном коде. Регистры являются одними из самых распространенных узлов в цифровых устройствах.

Основными классификационными признаками, по которым различают регистры, являются: способ записи информации (кода числа) в регистр, способ представления вводимой информации и способ тактирования.

По способу записи можно выделить регистры трех типов: параллельные (статические), последовательные (сдвигающие), параллельно-последовательные. *В параллельные регистры* запись числа осуществляется параллельным кодом, то есть во все разряды одновременно. *Последовательные регистры* характеризуются последовательной записью кода числа, начиная с младшего

или старшего разряда, путем последовательного сдвига кода тактирующими импульсами. *Параллельно-последовательные регистры* имеют входы как для параллельной, так и для последовательной записи кода числа. Поэтому ввод или вывод информации в них может осуществляться как в параллельном, так и в последовательном кодах.

По способу представления вводимой информации различают регистры однофазного и парафазного типа. В *однофазных* регистрах информация вводится по одному каналу (прямому или инверсному), а информация на выходе представлена в прямом или в обратном кодах. В парафазных регистрах ввод информации осуществляется по двум каналам одновременно (по прямому и инверсному входу), то есть одновременно в прямом и инверсном кодах. На выходе в парафазных регистрах информация также может быть представлена в прямом или в обратном кодах. Реализация парафазных регистров выполняется, в основном, на триггерах *CRS-типа*, а однофазных – на триггерах *CD-типа*.

По способу тактирования и в зависимости от типа используемых триггеров различают регистры многотактного и однотактного действия.

11.2. Параллельные регистры

Параллельный регистр используется для кратковременного хранения чисел, представленных в параллельном двоичном коде. Поэтому параллельные регистры называются еще регистрами памяти. В параллельных регистрах схемы разрядов не обмениваются данными между собой. Общими для разрядов являются только цепи управления (тактирования, сброса, установки, разрешения выхода или приема). В параллельных регистрах прием и выдача данных производится по всем разрядам одновременно, то есть время ввода числа в регистр равно времени ввода одного разряда в триггер.

Для построения *N*-разрядных параллельных регистров необходимо использовать *N* триггеров, каждый из которых должен иметь число входов, со-

ответствующее числу источников информации, подключенных к входу регистра. Каждый из триггеров в регистре имеет свой независимый информационный вход и свой независимый информационный выход, а тактовые входы С соединены между собой. Таким образом, параллельный регистр представляет собой многоразрядный, многовходовый триггер.

Однофазные параллельные регистры однотактного действия

Реализация однотактных параллельных однофазных регистров, в основном, выполняется на *CD-триггерах*. На рис.11.1 приведена функциональная схема такого регистра для записи двоичного числа $A(a_1,\ldots,a_n)$. Каждый триггер в регистре служит для хранения одного разряда числа, значит, для хранения *n*-разрядного двоичного числа необходимо иметь n *D-триггеров*. Запись числа $A(a_1,\ldots,a_n)$ осуществляется при поступлении переднего фронта синхроимпульса T, на входы D которых поданы соответствующие разряды числа A, без предварительной установки всех разрядов регистра в состояние «0», то есть однотактно.

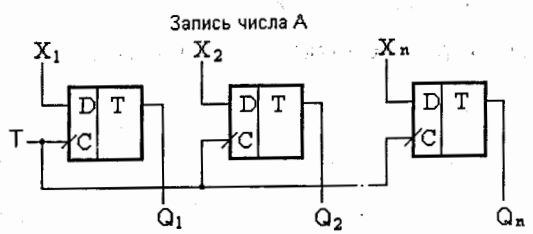


Рис.11.1. Структурная схема параллельного однофазного регистра на *D-триггерах*

Однофазные параллельные регистры двухтактного действия

Наиболее экономичными по числу компонентов являются однофазные регистры двухтактного действия, в качестве разрядов которых используются триггеры *CSR-типа* или двухтактные *D-триггеры*. Как следует из таблицы состояний *RS-триггера* (рис.9.11), для записи «1» необходимо подать «1» на вход S и «0» на вход R, а для записи «0» – наоборот, «1» на

вход R и «0» на вход S, то есть информация должна поступать на оба входа *RS-триггера*. Полученный регистр будет *парафазным,* причем вход S прямой, а вход R – инверсный. Для синхронной записи во все триггеры одновременно их тактовые входы необходимо объединить в одну шину.

Рассмотрим способ построения однофазных регистров двухтактного действия на *RS-триггерах*. На рис.11.2 приведена схема параллельного регистра двухтактного действия с однофазным режимом записи двоичного числа A. Входы регистра $X_1, X_2, \ldots, X_n$ (рис.11.2) соответствуют входам разрядов двоичного числа $A(a_1, a_2, \ldots, a_n)$. Первый тактирующий сигнал T_1 осуществляет установку всех разрядных триггеров в нулевое состояние ($Q_1 = Q_2 = \ldots = Q_n = 0$). По переднему фронту второго тактирующего сигнала T_2 обеспечивается запись числа A в данный регистр.

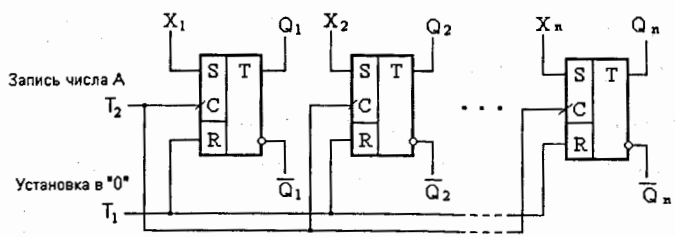


Рис.11.2. Структурная схема параллельного двухтактного однофазного регистра на синхронных *CSR-триггерах*

Парафазные параллельные регистры

Парафазные параллельные регистры (ППР) не отличаются таким широким многообразием, как однофазные. Все ППР по своему принципу действия являются однотактными и выполняются на триггерах *CRS-* или *CJK-типа*.

На рис.11.3, *а* приведена схема ППР для записи двоичного числа $A(a_1, a_2, \ldots, a_n)$, выполненная на *JK-триггерах*, а на рис.11.3, *б* аналогичная схема ППР, выполненная на *RSC-триггерах*. В этих схемах число A посту-

пает на регистры по двум каналам (прямому и инверсному) и заносится в них по тактирующему импульсу Т.

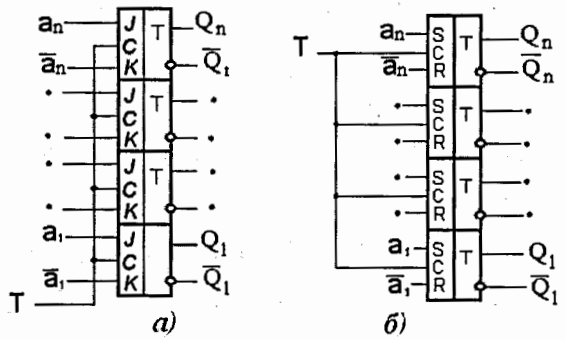


Рис. 11.3. Схема параллельного регистра на триггерах *JK-* и *RS-типа*

11.3. Сдвигающие регистры и их классификация

Сдвигающие (*Shift Register*) или последовательные регистры получили свое название от вида одной из наиболее распространенных операций, выполняемых в ЭВМ – операции сдвига кода числа. *Операция сдвига* – это перемещение (сдвиг) под действием внешних тактирующих импульсов содержимого регистра на определенное число разрядов вправо или влево. Чаще всего такое перемещение осуществляется на один разряд, и эту операцию можно записать в виде $Q_i^{n+1} = Q_{i-1}^n$, где Q_i^{n+1} – содержимое i-го (последующего) разряда регистра после цикла сдвига; Q_{i-1}^n – содержимое (i-1)-го (предыдущего) разряда до сдвига; i=1,2,3,..,n – число разрядов регистра сдвига. Время ввода числа в регистр последовательного типа равно m×T, где m – число разрядов вводимого числа, а T – период следования тактирующих сигналов, осуществляющих ввод (вывод) информации.

Наиболее общими классификационными признаками сдвигающих регистров (СР) можно считать следующие: способ управления сдвигом кода числа; вид электрической связи между разрядами; направление сдвига и способ приема и выдачи кода числа.

Способ управления сдвигом кода числа предполагает деление регистров по числу тактирующих сигналов, необходимых для выполнения операции сдвига информации на один разряд. По этому признаку все СР подразделяются на регистры *однотактного* и *многотактного* действия. Особенностью однотактных СР является то, что в них сдвиг кода числа на один разряд осуществляется за один такт. В многотактных СР сдвиг кода числа на один разряд осуществляется как минимум за два тактовых импульса.

По виду электрической связи между разрядами СР подразделяются на регистры с *однопроводной* (однофазной), *двухпроводной* (парафазной) и *смешанной* типами связей. Первые выполняются на триггерах *D-типа*, вторые на триггерах *RS-* или *JK-типа*, а третьи выполняются на триггерах *RS-* и *D-* или *JK-* и *D- типов*.

По направлению сдвига регистры подразделяются на три вида: регистры, осуществляющие сдвиг кода числа вправо (в сторону младших разрядов); регистры, сдвигающие код числа влево (в сторону старших разрядов); регистры, сдвигающие код числа как вправо, так и влево – реверсивные регистры сдвига. Последовательный сдвигающий «влево» регистр, построенный на *D-триггерах*, показан на рис.11.4.

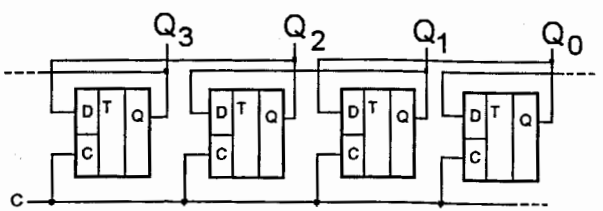


Рис.11.4. Сдвигающий «влево» регистр на триггерах *D-типа*

Иногда регистр должен иметь возможность сдвига информации в двух направлениях, параллельной записью и считыванием числа. В этом случае используются *реверсивные регистры* такие, как показано на рис.11.5. Этот регистр организован на базе *D-* триггеров.

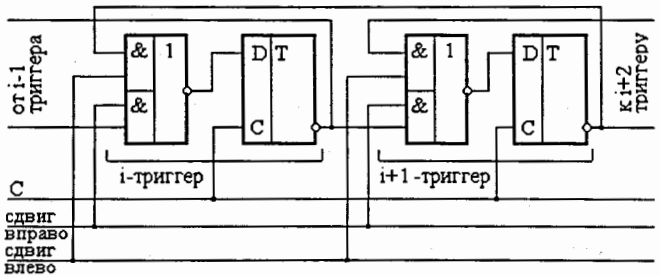


Рис.11.5. Реверсивный регистр на триггерах *D-типа*

Для построения *последовательного реверсивного регистра*, в котором код числа может сдвигаться как влево, так и вправо, необходимо между триггерами регистра включить устройства управления направлением сдвига. Эти устройства, в зависимости от значения управляющих сигналов «сдвиг вправо» и «сдвиг влево», должны переключать входы каждого триггера регистра либо к выходам предыдущего, либо к выходам последующего триггера (рис.11.5). Поскольку элемент «2И-ИЛИ-НЕ», выполняющий роль коммутатора, инвертирует значения сигналов, то для подачи на входы соответствующих разрядов сдвигаемого кода в прямом виде необходимо использовать инверсные выходы этих триггеров. Информацию на самый первый элемент «2И-ИЛИ-НЕ» с входа последовательного ввода необходимо подавать через инвертор. В практических схемах для упрощения процесса управления режимами направления сдвига вместо двух сигналов «сдвиг вправо» и «сдвиг влево» используется только один из этих сигналов. Второй сигнал формируется через инвертор.

По способу приема и выдачи кода числа регистры подразделяются на три вида: с последовательным приемом и последовательной выдачей информации; с параллельным вводом кода числа и параллельным выводом; с последовательно-параллельным вводом и последовательно-параллельным выводом кода числа. Последовательно-параллельные регистры являются

наиболее универсальными, так как они могут применяться в качестве первого и второго типа СР.

На рис.11.6 показан сдвигающий вправо последовательный регистр с параллельной записью кода, построенный на *D-триггерах*. Как и в последовательном регистре, ввод информации в последовательном коде осуществляется по входу A. Для тактирования сдвига синхровходы всех триггеров объединены. При этом для возможности введения кода числа в параллельном виде используются элементы «И-НЕ» (D1-1 ÷ Dn-2) в своих разрядах. Элемент D1-1 осуществляет функцию стробирования и инвертирования разряда B_i при единичном уровне управляющего сигнала «параллельная запись». В результате на вход S установки в «1» триггера T_i проходит инверсное значение разряда B_i параллельного кода числа только в том случае, если сигнал разрешения на линии «параллельная запись» имеет значение «1». Элемент D1-2 выполняет функцию инвертирования сигнала и передачи его на вход R сброса триггера T_i также по активному уровню сигнала разрешения «параллельной записи». В результате парафазный код всех разрядов параллельного кода проходит на соответствующий триггер только при активном уровне сигнала разрешения «параллельной записи». Поскольку элементы «И-НЕ» имеют инверсные выходы, то триггеры T_i должны иметь инверсные входы R и S.

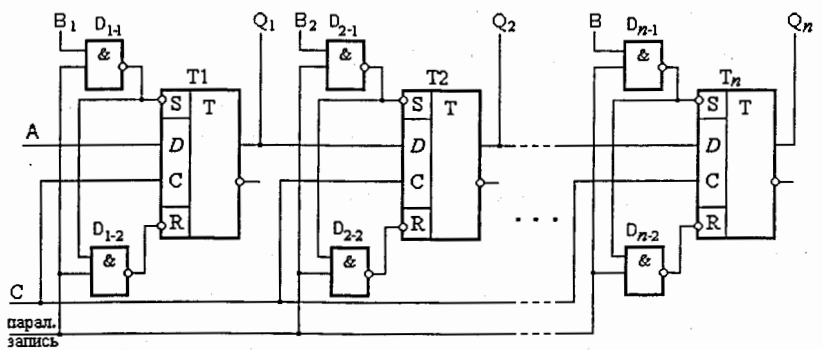


Рис.11.6. Сдвигающий вправо регистр с параллельной записью

Если выход последнего триггера соединить с входом первого, то получится *кольцевой регистр сдвига*. Записанная в его разряды информация под воздействием сдвигающих импульсов будет циркулировать по замкнутому кольцу. Кольцевой регистр иначе называется кольцевым счетчиком. Его коэффициент пересчета равен числу разрядов *n* последовательного кода. Единица, записанная в один из разрядов, периодически будет появляться в нем после того, как будут поданы *n* сдвигающих импульсов.

11.4. Функциональные требования при проектировании сдвиговых регистров

В основу приведенных классификационных признаков СР фактически положены функциональные требования, предъявляемые к ним. Это означает, что при их проектировании всегда задают: направление сдвига; способы приема и выдачи кода числа, представляющие собой функциональные требования для любого СР. Кроме того, к функциональным требованиям следует отнести функциональную надежность сдвигового регистра, которая характеризуется отсутствием опасных межкаскадных состязаний (гонок) при выполнении сдвига разрядов регистра на функционально надежных триггерах. К основным функциональным параметрам регистров сдвига можно отнести также их разрядность и быстродействие.

12. Запоминающие устройства

12.1. Определение запоминающих устройств

Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называются *запоминающими устройствами* (ЗУ) того или иного типа. Термин «запоминающее устройство» используется в тех случаях, когда речь идет о принципе построения устройства памяти (например, полупроводниковое ЗУ, ЗУ на жестком магнитном диске), а термин «память» используют, когда хотят подчеркнуть выполняемую данным устройством функцию или место расположения в составе оборудования ЭВМ (например, оперативная память, внешняя память). В случаях, когда эти отличия не принципиальны, термины «память» и «запоминающее устройство» используются как синонимы.

Запоминающие устройства играют важную роль в структуре ЭВМ. По некоторым оценкам, производительность компьютера на разных классах задач на 40 – 50% определяется характеристиками ЗУ, входящих в его состав.

12.2. Основные параметры запоминающих устройств

К основным параметрам, характеризующим запоминающие устройства, относятся: емкость памяти, организация памяти, быстродействие.

Емкость памяти – это максимальное количество данных, которое может храниться в ней. Емкость измеряется количеством адресуемых элементов (ячеек) ЗУ и длиной ячейки в битах. В настоящее время в ЭВМ практически все запоминающие устройства в качестве минимально адресуемого элемента используют 1 байт (1 байт $=2^3 = 8$ двоичных разрядов (бит)). Поэтому емкость памяти обычно определяется в байтах, килобайтах (1 кбайт $=2^{10} = 1024$ байта), мегабайтах (1 Мбайт $= 2^{20} = 2^{10}$ кбайт), гигабайтах (1 Гбайт $= 2^{30} = 2^{10}$ Мбайт) и т. д.

Организация памяти – это произведение числа хранимых слов на их разрядность. Хотя это произведение дает размер информационной ёмкости, оно является самостоятельным параметром. За одно обращение к ЗУ производится считывание или запись некоторой единицы данных, называемой *словом* (различной длины для устройств разного типа). Это определяет разную *организацию памяти*. Например, память объемом 1 мегабайт может быть организована как 1М слов по 1 байту, или 512К слов по 2 байта каждое, или 256К слов по 4 байта и т. д. В то же время в каждой ЭВМ используется свое понятие *машинного слова*, которое применяется при определении архитектуры компьютера, в частности при его программировании, и не

зависит от размерности слова памяти, используемой для построения данной ЭВМ. Например, компьютеры с архитектурой IBM PC имеют машинное слово длиной 2 байта.

Быстродействие памяти определяется продолжительностью операции обращения – временем, затрачиваемым на поиск нужной информации в памяти и на ее считывание, или временем на поиск места в памяти, предназначаемого для хранения данной информации, и на ее запись:

$$t_{обр} = \max(t_{обр\ сч}, t_{обр\ зп}),$$

где $t_{обр\ сч}$ – быстродействие ЗУ при считывании информации; $t_{обр\ зп}$ – быстродействие ЗУ при записи.

12.3. Классификация и структура устройств памяти

Запоминающие устройства выполняют функции запоминания или хранения массивов информации. Каждое информационное слово хранится в отдельном элементе памяти, называемой *ячейкой памяти*. Основная функция любой памяти состоит именно в выдаче информации из этих ячеек по внешнему запросу. Следовательно, эти устройства можно классифицировать таким образом (рис.12.1): по способу записи и хранения информации, по типу организации обращения к ячейкам памяти.

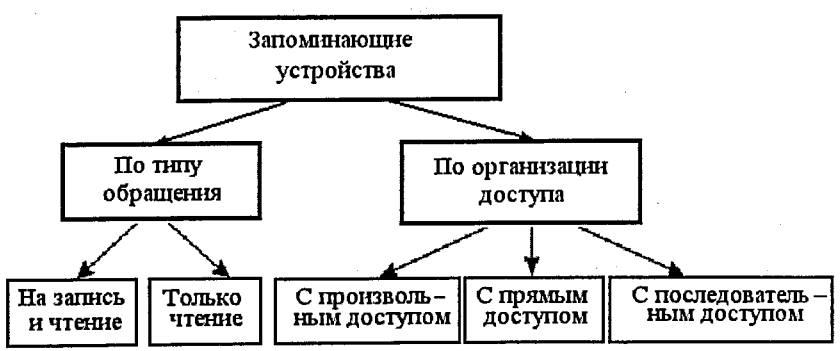


Рис.12.1. Классификация запоминающих устройств

По способу записи и хранения информации все ЗУ подразделяются на:

A. *Постоянное запоминающее устройство* ПЗУ, или ROM (*Read Only Memory*), в которое информация заносится только один раз и она не пропадает при выключении питания, то есть энергонезависимая память.

B. *Программируемая постоянная память* ППЗУ, или PROM (*Programmable ROM*), в которую информация может заноситься пользователем ограниченное число раз с помощью специальных методов. Информация в памяти также не пропадает при выключении питания (энергонезависима).

C. *Оперативное запоминающее устройство* ОЗУ, или RAM (*Random Access Memory*), в которое запись может производиться сколько угодно раз на протяжении всего срока службы микросхемы. Информация в памяти пропадает при выключении питания, то есть она энергозависима.

Память типа ПЗУ и ППЗУ используется в процессе работы процессора для хранения параметров, необходимых для запуска компьютера в работу, а также для хранения постоянных программных констант. В некоторых ЭВМ, предназначенных, например, для работы в системах управления по одним и тем же неизменяемым алгоритмам, все программное обеспечение может храниться в ПЗУ. Память типа ОЗУ используется процессором для хранения выполняемых программ, исходных данных, промежуточных и окончательных результатов.

По типу организации доступа все ЗУ делятся:

а) на ЗУ с *произвольным доступом* – время доступа не зависит от места расположения участка памяти (например, ОЗУ);

б) на ЗУ с *прямым* (циклическим) *доступом* – благодаря непрерывному вращению носителя информации (например, магнитный диск) возможность обращения к некоторому участку носителя циклически повторяется (время доступа зависит от взаимного расположения этого участка и головок чтения/записи и во многом определяется скоростью вращения носителя);

в) на ЗУ с последовательным доступом, где производится последовательный просмотр участков носителя информации, пока нужный участок не займет требуемое положение, например, напротив головок чтения/записи с магнитной ленты.

Существует множество промежуточных типов памяти, а также подтипов, но указанные типы – самые главные, принципиально отличающиеся друг от друга.

В общем случае элементы ЗУ имеют следующие входы и выходы:

1. *Адресные входы*, которые образуют *шину адреса*. Код на адресных линиях представляет номер ячейки памяти, к которой происходит обращение в данный момент времени. Количество адресных разрядов определяет количество ячеек памяти (для n адресных разрядов количество ячеек памяти равно 2^n).

2. *Выходы данных*, которые образуют *шину данных* памяти. Код на линиях данных представляет содержимое той ячейки памяти, к которой производится обращение в данный момент времени. Количество разрядов данных определяет количество разрядов ячеек памяти.

3. В случае оперативной памяти помимо входной адресной шины необходима *входная шина данных*, на которую подаются данные, записываемые в ячейку памяти. Возможен вариант совмещения входной и выходной шины данных, то есть организация *двунаправленной шины данных*. Направление передачи по этой шине определяется управляющими сигналами.

4. У большинства микросхем памяти имеются дополнительные входы, которые определяют режим работы. В частности, это вход выбора микросхемы «CS» (их может быть несколько, объединенных логической операцией И), а у оперативной памяти обязательно присутствует вход записи/чтения «WR» активный уровень на котором переводит микросхему в режим записи.

12.4. Постоянные запоминающие устройства

12.4.1. Однократно программируемые постоянно запоминающие устройства

Постоянные запоминающие устройства делятся на однократно программируемые ОПЗУ (например, биполярные ОПЗУ с «плавкими» соединениями) и многократно электрически программируемые МОП ПЗУ.

Однократно программируемые постоянные запоминающие устройства (ОПЗУ) это наиболее дешевые, емкие и быстродействующие большие интегральные схемы (БИС). Существует три основных способа построения ОПЗУ: использование плавких перемычек; использование прожигаемых кремниевых перемычек; программирование на одном из дополнительных (промежуточных) этапах изготовления (масочные ОПЗУ). Так, например, серийно выпускаемые микросхемы памяти ОПЗУ типа КР556РТ4 и КР556РТ5 используют прожигаемые кремниевые перемычки. Микросхема КР556РТ4 содержит 256×4, а КР556РТ5 – 512×8 ячеек программируемой памяти. Микросхемы состоят из матриц накопителя, дешифраторов адресов, из матрицы шин, обеспечивающих прожигание перемычек и мощных ключей для коммутации прожигающего тока.

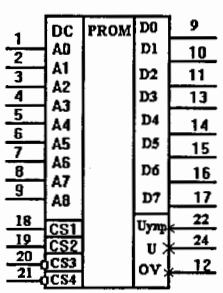


Рис.12.2. Условно-графическое обозначение микросхемы К556 РТ5 с однократной записью информации

На рис.12.2 показано условно-графическое обозначение ОПЗУ К556РТ5, где $U_{упр}$ – вход сигнала управления; U – вход напряжения пита-

ния; OV – вход общей шины питания и сигналов; $A_0 \div A_8$ – девятиразрядная шина адреса; $D_0 \div D_7$ – восьмиразрядная шина данных; CS1 ÷ CS4 – сигналы выбора кристалла. На рис.12.3 показан фрагмент матрицы накопителя ОПЗУ с прожигаемыми связями.

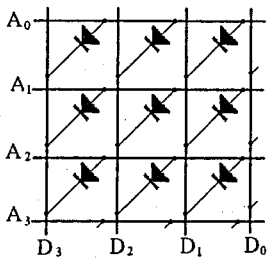


Рис.12.3. Фрагмент матрицы накопителя ОПЗУ

12.4.2. Перепрограммируемые ПЗУ с ультрафиолетовым стиранием

Перепрограммируемые постоянные запоминающие устройства с возможностью стирания записи ультрафиолетовым излучением или электрическим способом допускают многократное стирание записываемой информации. Это достигается благодаря использованию МОП-транзисторов с изолированными *плавающими затворами*. Из-за возможности многократного стирания записанной информации этот тип микросхем называют многократно-программируемыми ПЗУ (МПЗУ).

Конструкция полевого МОП-транзистора с плавающим затвором приведена рис.12.4, а на рис.12.5 показан фрагмент матрицы накопителя МПЗУ на полевых МОП-транзисторах с изолированными затворами.

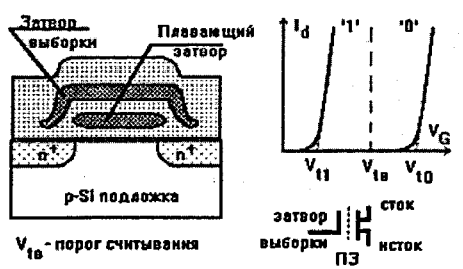


Рис.12.4. МОП-транзистор с плавающим затвором

МОП-транзистор с изолированным плавающим затвором – это *p*-канальный нормально закрытый МОП-прибор. Плавающий затвор представляет собой область кремния, окруженную со всех сторон диэлектриком, то есть он электрически не связан с другими электродами и его потенциал «плавает». Толщина нижнего диэлектрического слоя обычно составляет десятки ангстрем. Положительное смещение на верхнем затворе (относительно полупроводниковой подложки) вызовет накопление электронов на плавающем затворе (между плавающим затвором и обычным затвором – шиной программирования $U_{упр}$), то есть при прохождении через шину программирования импульса тока на «плавающем» затворе возникает наведенный заряд. В результате этого при программировании на изолированных (плавающих) затворах возникает отрицательный заряд, который благодаря высокому удельному сопротивлению двуокиси кремния может сохраняться неизменным в течение многих лет. Эти заряды закрывают или открывают каналы полевых транзисторов. Например, если плавающий затвор заряжен отрицательным зарядом, то транзистор с *n*-каналом закрыт, а если нет, то открыт.

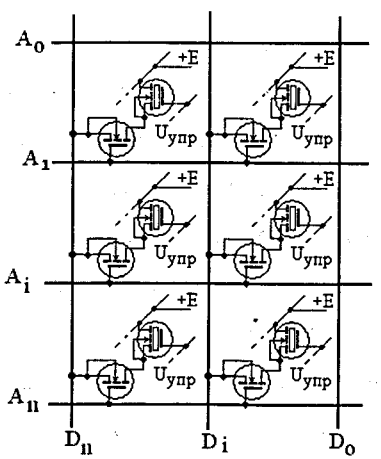


Рис. 12.5. Фрагмент матрицы накопителя МПЗУ на МОП-транзисторах с изолированными затворами: +E – напряжение питания; A_i – линия адресной шины; D_i – линии шины данных

МОП-транзистор с плавающим затвором используется в качестве элемента памяти со временем хранения равным, времени диэлектрической релаксации структуры, которое может быть очень велико и в основном определяется низкими токами утечки через барьер $Si\text{-}SiO_2$. Такой элемент памяти обеспечивает возможность непрерывного считывания без разрушения информации, причем запись и считывание могут быть выполнены в очень короткое время. Так, например, серийно выпускаемая память типа КР573РФ2 является МПЗУ емкостью 2048 байт со стиранием информации ультрафиолетовым излучением. Микросхемы большей емкости, подобные микросхеме КР573РФ2, – это КР573РФ4, КР573РФ6, КР573РФ8.

Стирание информации в такой памяти осуществляется путем облучения кристаллов микросхем ультрафиолетовым излучением через специальное окно в корпусе микросхемы. Под действием облучения двуокись кремния частично ионизируется по всему объему и теряет свои изоляционные свойства там, куда проникает излучение. При этом заряд затвора стекает на подложку, и полевые транзисторы снова закрываются.

12.5. Оперативные запоминающие устройства

Современные оперативные запоминающие устройства (ОЗУ) выполнены на комплементарных МОП-микросхемах (КМОП), которые отличаются малой потребляемой мощностью. Это достигается за счет применения пары МОП-транзисторов с разным типом каналов: *n*-МОП и *p*-МОП. В КМОП-инверторе, как при низком, так и при высоком уровне сигнала на входе, один из транзисторов закрыт (рис.12.6), поэтому потребление энергии происходит только при переключении схемы с «1» в «0» или обратно. Аналогично на четырех МОП-транзисторах (2 *n*-МОП и 2 *p*-МОП, включенных параллельно и последовательно) можно построить и другие базовые логические элементы «И» и «ИЛИ» соответственно. На их основе строятся все другие более сложные логические схемы.

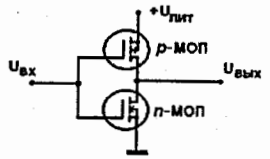


Рис.12.6. КМОП-инвертор

Оперативные запоминающие устройства подразделяются на:

- запоминающие устройства (ЗУ) с произвольной выборкой (ЗУПВ):
  - а) оперативные статические запоминающие устройства (СОЗУ);
  - б) динамические оперативные запоминающие устройства (ДОЗУ);
- запоминающие устройства с последовательным доступом:
  - а) регистры сдвига;
  - б) приборы с зарядовой связью (ПЗС).

12.6. Оперативные запоминающие устройства статического типа

Различают статические ОЗУ на *n*-МОП-структурах и *k*-МОП-структурах. Элементарной ячейкой статического ОЗУ с произвольной выборкой является триггер на транзисторах $T_1 \div T_4$ с ключами $T_5 \div T_8$ для доступа к шине данных (рис.12.7), где T_1, T_2 – это элементы нагрузки, а T_3, T_4 – элементы триггера. Сопротивление элементов T_1, T_2 легко регулируется в процессе изготовления транзистора путем подгонки порогового напряжения при легировании кремниевого затвора. Количество транзисторов на ячейку зависит от логической организации памяти микропроцессорной системы. Ячейка ЗУ содержит триггер, являющийся элементом памяти, и управляющие ключи для выбора ячейки, записи и считывания информации.

На рис.12.8 показана структурная схема КМОП-микросхемы ОЗУ КР537РУ10. Эта микросхема содержит 2048 восьмиразрядных слов оперативной памяти, где A – адресная шина; D – двунаправленная шина данных; DC_x – дешифратор строк; DC_y – дешифратор столбцов. Микросхема содержит матрицу-накопитель емкостью 2048x8 бит. Выбор каждого из 2048

слов осуществляется с помощью дешифраторов строк DC_X и столбцов DC_Y матрицы соответственно. Микросхема может работать в трех режимах: режим считывания, режим записи и режим хранения. Режим работы определяется сигналами, подаваемыми на входы схемы управления согласно временной диаграмме, показанной на рис.12.9, где A – сигналы адресной шины; D – сигналы шины данных; CE, CO – стробсигналы записи и считывания; W – сигнал разрешения записи и считывания.

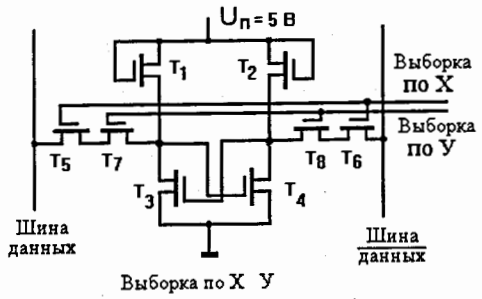


Рис. 12.7. Ячейка статического ОЗУ

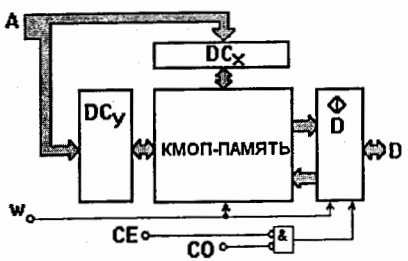


Рис.12.8. Структурная схема статической памяти КР537РУ10

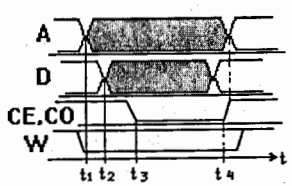


Рис.12.9. Временные диаграммы записи и считывания данных в КР537РУ10

12.7. Оперативные запоминающие устройства динамического типа

В отличие от статических ЗУ динамические ОЗУ строятся по *n*-МОП-технологии и характеризуются отсутствием транзисторов, работающих в качестве нагрузочных резисторов. В этом типе памяти информация хранится не в регистрах, а в виде заряда на конденсаторах. Такие схемы более компактны, имеют меньшую стоимость, но для сохранения данных необходима постоянная регенерация (перезапись) информации. Кроме того, они имеют более низкое быстродействие по сравнению со статическими ЗУ. Область применения динамической памяти гораздо уже, и в основном они применяются в качестве системной оперативной памяти компьютеров, где соображения стоимости выходят на первый план.

Рис.12.10. Запоминающая ячейка динамического ОЗУ

Простейшая динамическая ячейка ОЗУ показана на рис.12.10, где для хранения одного бита в ячейке памяти нужны всего один или два транзистора и накопительный конденсатор. Во время записи управляющий импульс шины адреса открывает транзисторы Т1 и Т2, а емкость С заряжается током разрядной шины. Для регенерации содержимого ячейки необходимо повторять запись или считывание через определенные интервалы времени. Регенерация может производиться также с помощью общего тактирующего устройства через определенные интервалы времени (каждые 1÷20 мс, в зависимости от уровня интеграции микросхемы). Для этого в микросхемах динамического ОЗУ есть один или несколько тактовых генераторов и логическая схема для восстановления информационного заряда, стекающего с конденсатора.

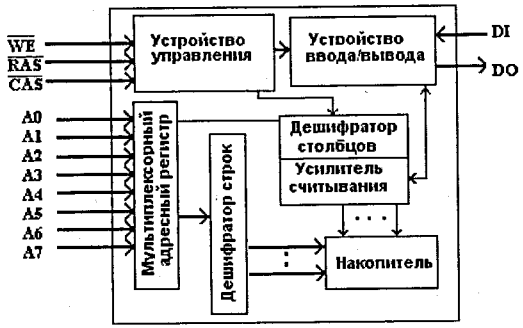


Рис.12.11. Структура ЗУ с произвольной выборкой (ЗУПВ) КР565РУ5 на 64К×1 бит

На рис.12.11 показана структурная схема микросхемы ЗУ с произвольной выборкой (ЗУПВ) КР565РУ5 емкостью 64×1, где: DO – информационный вход; DI – информационный выход; $A_1 \div A_7$ – адресные входы; WE – вход запись/чтение CS (*Chip Select*); CAS, RAS – входы выбора кристалла (*Column Address Strobe* и *Row Address Strobe*). Микросхема КР565РУ5 – это ЗУ с организацией хранения информации 65536 бит на 1 разряд. Накопительная матрица с однотранзисторными запоминающими элементами имеет размер 512×128. Для уменьшения количества ножек у ИС (16-входовый DIP корпус) применяется мультипликация адреса, что видно по наличию отдельных дешифраторов строк и столбцов (рис.12.11). Устройство управления включает два генератора тактовых сигналов и генератор сигналов записи и обеспечивает четыре режима работы: записи, считывания, регенерации и мультипликации адреса. Время регенерации памяти – 2 мс.

12.8. Микросхемы памяти в составе микропроцессорной системы

На рис.12.12 показано взаимодействие микросхем памяти К573РФ2 и К573РУ9, имеющих одинаковую организацию (2К×8), с системной магистралью. Дешифратор К555ИД7 посредством сигнала CS (выбор кристалла) позволяет выбрать положение конкретной микросхемы ЗУ в адресном

пространстве. Для данного случая – это адреса «0000h-07FFh» для ПЗУ(ROM) и «0800h-0FFFh» для ОЗУ(RAM).

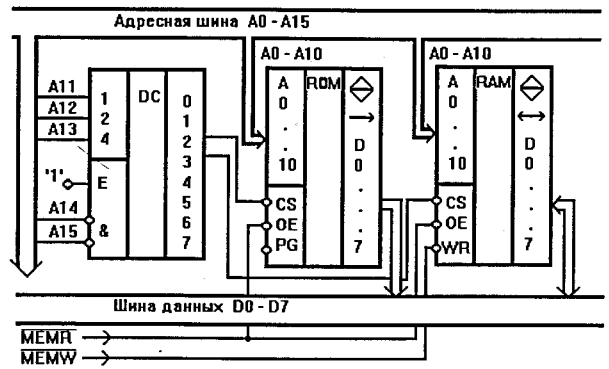


Рис.12.12. Микросхемы ОЗУ (К573РУ9) и ППЗУ (К573РФ5) в составе микропроцессорной системы

Типичные временные диаграммы циклов записи и чтения приведены на рис.12.13. Для записи информации в память надо выставить код адреса на адресных входах, выставить код записываемых в этот адрес данных на входах данных, подать сигнал записи WR и подать сигнал выбора микросхемы CS. Порядок выставления сигналов бывает различным, он может быть существенным или несущественным, например, можно выставлять или снимать CS раньше или позже выставления или снятия WR. Собственно запись обычно производится сигналом WR или CS, причем данные должны удерживаться в течение всего сигнала WR (или CS) и заданное время после его окончания.

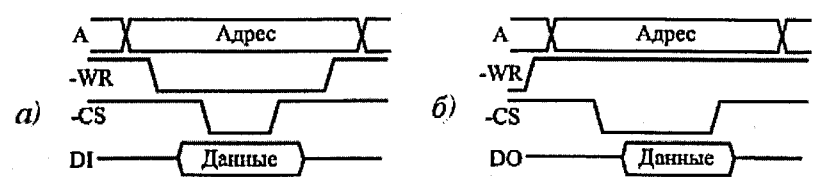


Рис.12.13. Типичные временные диаграммы записи в память (*а*) и чтения из памяти (*б*)

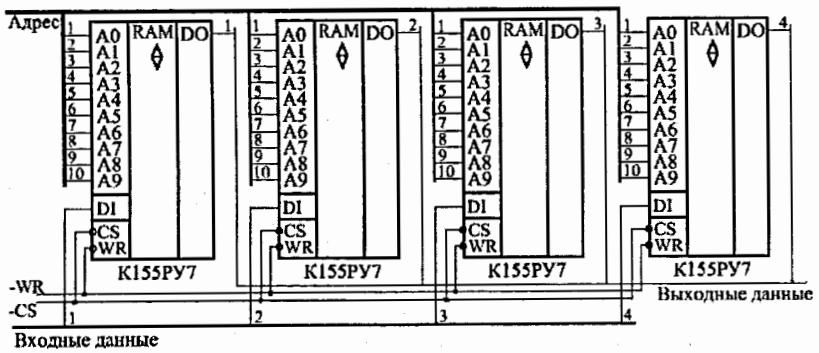


Рис.12.14. Увеличение разрядности шины данных

В случае микросхем памяти с двунаправленной шиной данных необходимо использовать источник записываемых данных с выходом, имеющим три состояния (3С), или с общим коллектором (ОК), чтобы избежать конфликта данных, записываемых в память, с данными, выдаваемыми из памяти в режиме чтения. На рис.12.14 показано объединение микросхем памяти К155РУ7 для увеличения разрядности шины данных.

12.9. ОЗУ для временного хранения информации

Главное применение микросхем оперативной памяти – это временное хранение цифровой информации, всевозможных массивов кодов, таблиц данных, одиночных чисел и т. д. Цель такого хранения информации состоит в том, чтобы в любой момент иметь возможность быстро ее прочитать для дальнейшей обработки или для другого использования. В зависимости от того, в каком порядке может записываться или читаться информация, существуют две разновидности ОЗУ: ОЗУ с параллельным, или произвольным, доступом и ОЗУ с последовательным доступом.

12.9.1. ОЗУ с параллельным, или произвольным, доступом

Параллельный, или произвольный, доступ наиболее прост и не требует никаких дополнительных элементов, так как именно на этот режим непосредственно рассчитаны микросхемы памяти. В этом режиме можно записывать информацию в любой адрес ОЗУ и читать информацию из любого адреса ОЗУ в произвольном порядке. Однако параллельный доступ требует формирования довольно сложных последовательностей всех входных сигналов памяти. То есть для записи информации необходимо сформировать код адреса записываемой ячейки и только потом подать данные, сопровождаемые управляющими сигналами CS и WR (рис.12.13.). Точно так же необходимо подавать полный код адреса читаемой ячейки при операции чтения. Этот режим доступа чаще всего применяется в компьютерах и контроллерах, где самыми главными факторами являются универсальность и гибкость использования памяти для самых разных целей.

12.9.2. ОЗУ с последовательным доступом

Последовательный доступ к памяти предполагает более простой порядок общения с памятью. В этом случае не надо задавать код адреса записываемой или читаемой ячейки, так как адрес памяти формируется схемой автоматически. Для записи информации надо всего лишь подать код записываемых данных и сопроводить его стробом записи. Для чтения информации надо подать строб чтения и получить читаемые данные. Автоматическое задание адреса при этом осуществляется внутренними счетчиками, меняющими свое состояние по каждому обращению к памяти. Например, десять последовательных циклов записи запишут информацию в десять последовательно расположенных ячеек памяти. Недостаток такого подхода – нет возможности записывать или читать ячейки с произвольными адресами в любом порядке.

Можно выделить три основных типа ОЗУ с последовательным доступом: память типа FIFO (*First In – First Out*, то есть первым вошел – первым вышел); память стекового типа LIFO (*Last In –First Out*, то есть последним вошел – первым вышел); память для хранения массивов данных.

Два первых типа памяти FIFO и LIFO подразумевают возможность чередования операций записи и чтения в памяти. При этом память FIFO выдает данные в том же порядке, в котором они были записаны, а память LIFO – в порядке, обратном тому, в котором они были записаны в память. Память FIFO можно сравнить со сдвиговым регистром. Память с принципом LIFO используется, в частности, в компьютерах (стек), где она хранит информацию о параметрах программ и подпрограмм. Для памяти FIFO требуется хранение двух кодов адреса (адрес для записи и адрес для чтения), для памяти LIFO достаточно одного кода адреса.

Память для хранения массивов предполагает, что сначала в память записывается целиком большой массив данных, а потом этот же массив целиком читается из памяти. Эта память также может быть устроена по двум принципам (FIFO и LIFO). В первом случае (FIFO) записанный массив читается в том же порядке, в котором и был записан, во втором случае (LIFO) – в противоположном порядке (начиная с конца). В обоих этих случаях для общения с памятью требуется хранить только один код адреса памяти.

ОЗУ типа FIFO

На рис.12.15 показана функциональная схема памяти типа FIFO на микросхемах с раздельными шинами входных и выходных данных. Адреса памяти задаются двумя счетчиками – счетчиком записи и счетчиком чтения, выходные коды которых мультиплексируются с помощью 2-канального мультиплексора. Запись данных осуществляется по стробу записи «Зап», чтение данных – по стробу чтения «Чт». Своим задним фронтом сигнал «Зап» переключает счетчик записи, а задний фронт сигнала «Чт» переключает счетчик чтения. В результате каждая следующая запись осуществляется в следующий по порядку адрес памяти. Точно так же каждое следующее чтение производится из следующего по порядку адреса памяти.

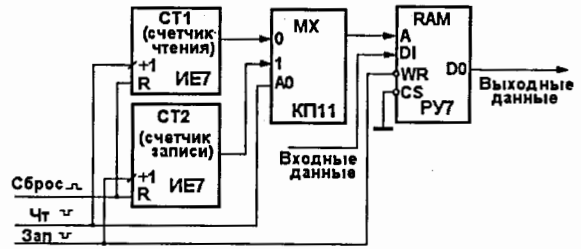


Рис.12.15. Функциональная схема памяти типа FIFO

Рассмотрим работу ОЗУ типа FIFO. Перед началом работы необходимо сбросить счетчики в «0» сигналом «Сброс». При отсутствии операций записи и чтения память находится в состоянии чтения (сигнал WR равен единице), а на адресные входы памяти подается код адреса записи со счетчика записи. При подаче строба записи «Зап» производится запись входных данных по адресу из счетчика записи. Входные (записываемые) данные должны выставляться раньше начала сигнала «Зап», а заканчиваться после этого сигнала. При подаче строба чтения «Чт» мультиплексор переключается на передачу адреса чтения, и на выходе памяти появляется информация, которая считывается из адреса чтения, задаваемого счетчиком чтения. Действительными выходные (читаемые) данные будут по заднему (положительному) фронту сигнала «Чт». Запись начинается с нулевого адреса памяти и производится по последовательно нарастающим адресам. Точно так же чтение начинается с нулевого адреса памяти и производится по последовательно нарастающим адресам. Операции записи и чтения могут чередоваться. Временные диаграммы циклов записи и чтения показаны на рис.12.16.

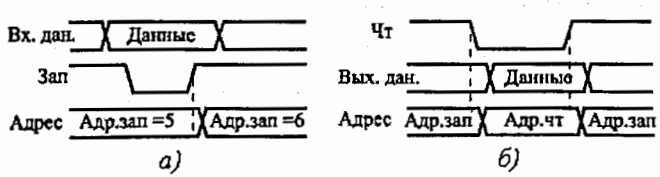


Рис.12.16. Временные диаграммы циклов записи (*а*) и чтения (*б*) для памяти типа FIFO

Условия правильной работы схемы следующие. Длительность сигнала «Зап» не должна быть меньше минимально допустимой длительности сигнала WR памяти. Длительность сигнала «Чт» не должна быть меньше суммы задержки переключения мультиплексора и задержки выборки адреса памяти. Период следования сигнала «Зап» не должен быть меньше суммы задержки переключения счетчика записи и длительности сигнала «Зап». Период следования сигнала «Чт» не должен быть меньше суммы задержки переключения счетчика чтения и длительности сигнала «Чт».

ОЗУ типа LIFO

Функциональная схема памяти типа LIFO (рис.12.17) проще по структуре, чем схема памяти FIFO, так как она содержит только один счетчик и не требует мультиплексирования. На рис.12.17 память типа LIFO использует двунаправленную шину входных/выходных данных.

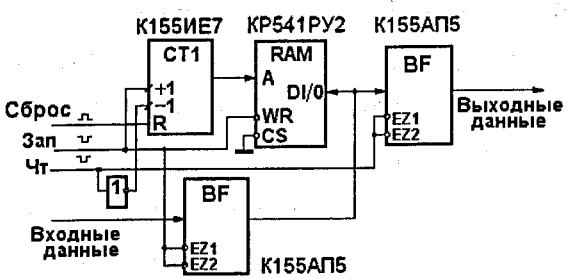


Рис.12.17. Функциональная схема памяти типа LIFO

В этой схеме необходим реверсивный счетчик адреса, с раздельными тактовыми входами прямого и обратного счета (например, ИЕ7). После проведения цикла записи по заднему фронту сигнала «Зап» счетчик увеличивает свой выходной код (адрес памяти) на единицу. Перед проведением цикла чтения по переднему фронту сигнала «Чт» счетчик уменьшает свой выходной код на единицу. Такая организация перебора адресов позволяет организовать чтение из памяти в порядке, обратном порядку записи в память. Например, пусть исходное состояние счетчика – 2 и мы производим три цикла

записи: первый – в адрес 2, второй – в адрес 3, третий – в адрес 4. После третьего цикла записи счетчик будет выдавать код 5. Затем проведем три цикла чтения: первый – из адреса 4 (перед чтением адрес уменьшился на единицу), второй – из адреса 3, третий – из адреса 2. После третьего цикла чтения счетчик будет выдавать код 2. Мы вернулись в исходное состояние, прочитав записанную информацию в обратном порядке.

Исходное состояние счетчика в данной схеме не важно, так как не важен текущий адрес памяти, в который производится запись и из которого потом производится чтение. Однако в случае, когда используется начальный сброс счетчика в нулевое состояние (по сигналу «Сброс»), можно проще организовать контроль за переполнением памяти LIFO. Для контроля переполнения можно использовать выходной сигнал переноса старшего счетчика (>15).

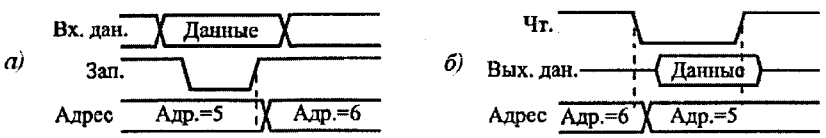


Рис.12.18. Временные диаграммы циклов записи (*а*) и чтения (*б*) для памяти типа LIFO

Временные диаграммы циклов записи и чтения приведены на рис.12.18. В цикле записи по сигналу «Зап» открывается входной буфер АП5 и входные (то есть записываемые) данные поступают на входы/выходы памяти. Одновременно по этому сигналу память переходит в режим записи. В результате в текущий адрес памяти записываются входные данные, после чего адрес увеличивается на единицу. Входные данные должны начинаться до начала сигнала «Зап» и заканчиваться после его конца. В цикле чтения по сигналу «Чт» адрес уменьшается на единицу, после чего открывается выходной буфер АП5, который выдает на выход схемы читаемую из памяти информацию. Применение выходного буфера не

обязательно, однако он предотвращает попадание на шину выходных данных информации, записываемой в память в цикле записи. Выходные данные действительны по заднему фронту сигнала «Чт».

ОЗУ для хранения массивов данных

Третий тип ОЗУ для временного хранения данных – память для хранения массивов данных. Рассмотрим схему такой памяти типа FIFO. Адреса памяти в данном случае задаются одним единственным счетчиком, который работает в режиме только прямого счета (рис.12.19). Перед началом работы необходимо сбросить счетчик (сигнал «Сброс»). Затем производится запись массива данных. При этом после каждого цикла записи по заднему фронту сигнала «Зап» выходной код счетчика увеличивается на единицу. После окончания записи всего массива снова надо сбросить счетчик в нуль (сигнал «Сброс»), а затем производить чтение массива, начиная с нулевого адреса. При этом после каждого цикла чтения по заднему фронту сигнала «Чт» выходной код счетчика снова увеличивается на единицу. В результате массив данных читается в том же порядке, что и был записан. Контроль за длиной записываемого и считываемого массива возлагается на внешнее (по отношению к приведенной схеме) устройство.

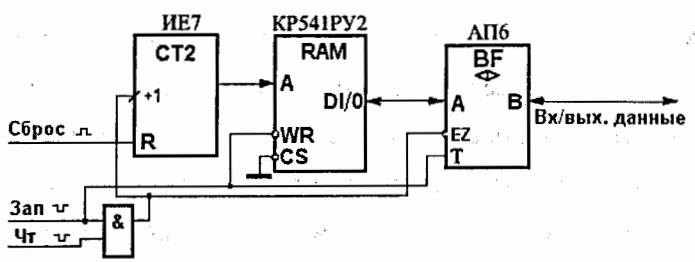


Рис.12.19. Функциональная схема памяти для хранения массивов данных

Для данной схемы должна использоваться нетактируемая память (например, КР541РУ2) с двунаправленной шиной входных/выходных данных. Следовательно, данные подаются на схему и читаются из схемы также

по двунаправленной шине данных. Между памятью и этой шиной включается двунаправленный буфер (типа АП6), который может понадобиться, например, для обеспечения большого выходного тока и малого входного тока со стороны внешней двунаправленной шины данных (типичная ситуация при построении микропроцессорных и компьютерных систем). Этот буфер открывается на передачу данных в память по сигналу «Зап» (сигнал «EZ» становится равным нулю, сигнал «T» также нулевой) и открывается для чтения данных из памяти по сигналу «Чт» (сигнал «EZ» становится равным нулю, сигнал «T» – единице).

Условия правильной работы схемы следующие. Длительность сигнала записи должна быть не меньше минимальной длительности сигнала WR памяти. Входные (записываемые) данные должны начинаться до начала сигнала «Зап», а заканчиваться после его окончания. Длительность сигнала чтения не должна быть меньше суммы задержки буфера и времени выборки адреса памяти. Действительными читаемыми данными будут данные, выбранные по заднему фронту сигнала «Чт». За период следования сигналов «Зап» и «Чт» схема должна успевать выполнить операцию записи и чтения, кроме того, должен успеть полностью переключиться счетчик адреса памяти.

12.10. ОЗУ как информационный буфер

Второе важнейшее применение микросхем оперативной памяти состоит в организации разнообразных информационных буферов, буферной памяти для промежуточного хранения данных, передаваемых между двумя устройствами или системами. Суть информационного буфера состоит в следующем: передающее устройство записывает передаваемые данные в буфер, а принимающее устройство читает принимаемые данные из буфера (рис.12.20). Такое промежуточное хранение позволяет лучше скоординировать работу устройств, участвующих в обмене данными, повысить их независимость друг от друга, согласовать скорости передачи и приема данных.

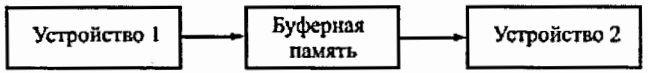


Рис.12.20. Включение буферной памяти

Пусть, например, в качестве первого устройства выступает компьютер, а в качестве второго – кабель локальной сети. Компьютеру значительно удобнее выдавать данные со скоростью, определяемой его собственным быстродействием, но в локальную сеть надо передавать данные со строго определенной скоростью, задаваемой стандартом на сети (например, 100 Мбит/с). Кроме того, компьютер по возможности не должен отвлекаться на контроль за текущим состоянием сети, за ее занятостью и освобождением. В данном случае буферная память просто необходима. Точно так же буферная память нужна при приеме данных, поступающих из локальной сети в компьютер.

Главное отличие буферной памяти от памяти для временного хранения информации, рассмотренной в разделе 12.9, состоит в том, что к информационному буферу всегда имеют доступ не одно внешнее устройство, а два (или даже более). Из-за этого иногда существенно усложняется как схема задания адреса микросхемы памяти, так и схема разделения потоков данных (записываемых в память и читаемых из памяти).

Рис.12.21. Двунаправленный информационный буфер

Информационные буферы бывают однонаправленными (входными или выходными) и двунаправленными, то есть входными и выходными одновременно (рис.12. 21). Например, буферная память сетевого адаптера двунаправленная, так как она буферирует и информацию, передаваемую в сеть из компьютера, и информацию, принимаемую из сети в компьютер.

Двунаправленные буферы всегда сложнее проектировать из-за большего количества потоков данных. Информационные буферы могут обеспечивать периодический обмен между устройствами или непрерывный обмен между ними. Примером буфера с непрерывным режимом обмена может служить контроллер видеомонитора, информация из которого постоянно выдается на видеомонитор, но может изменяться по инициативе компьютера. Информационные буферы с периодическим режимом обмена данными могут быть организованы по типу FIFO или по типу LIFO.

Информационные буферы с периодическим обменом

Информационные буферы с периодическим режимом обмена могут быть организованы по типу FIFO или по типу LIFO. В случае FIFO массив данных читается из памяти одним устройством в том же порядке, в каком он был записан в память другим устройством. Выпускаются даже специальные микросхемы быстродействующей буферной памяти типа FIFO, которые не имеют адресной шины и представляют собой, по сути, многоразрядный сдвиговый регистр. В отличие от обычной микросхемы сдвигового регистра, где записываемую информацию можно считать только тогда, когда она продвинется по всем ячейкам регистра, информацию с выходов буфера FIFO можно начинать читать с выходов сразу после записи.

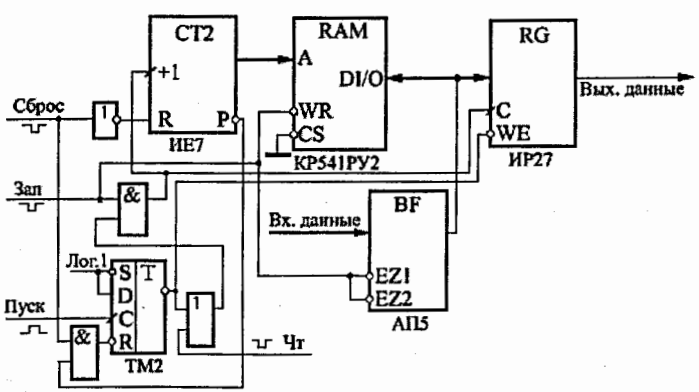


Рис.12.22. Однонаправленный буфер типа FIFO

На рис.12.22 показан простейший однонаправленный буфер с периодическим режимом обмена по принципу FIFO. Одно устройство записывает информацию в буфер, на другое устройство выдается информация из буфера. Память всегда записывается полностью, по всем адресам, и читается также полностью. Строб записи «Зап» поступает в режиме записи с частотой, необходимой для записи, строб чтения «Чт» поступает при чтении с частотой, необходимой для чтения. Шины данных для записи и чтения в память в случае, показанном на рисунке, отдельные. При таких условиях в схеме необходим всего лишь один счетчик для перебора адресов памяти, работающий только в режиме прямого счета и имеющий вход начального сброса в «0».

Перед началом работы устройство, производящее запись в память, сбрасывает счетчик в «0» сигналом «Сброс» и устанавливает режим записи в память. Затем начинается процесс записи: записываемые данные поступают с однонаправленного входного буфера (АП5) и записываются в память сигналом «Зап», который своим задним фронтом переключает адреса памяти. Полная процедура записи включает в себя столько циклов записи, сколько имеется ячеек у используемой памяти.

После окончания процедуры записи устройство, производившее запись, разрешает чтение из памяти, устанавливая в «1» триггер положительным фронтом сигнала «Пуск» («0» на инверсном выходе). При этом разрешается прохождение сигнала «Чт». Адреса памяти переключаются по заднему фронту сигнала «Чт» и по этому же фронту данные, читаемые из памяти, фиксируются в выходном регистре ИР27.

Выходной регистр выполняет две функции: не пропускает на выход данные, записываемые в память (по сигналу WE запрещается запись в триггер); обеспечивает одновременность изменения всех разрядов читаемых данных. После окончания чтения всего объема памяти, вырабатывается сигнал переноса счетчика P, который переводит всю схему в режим записи, переключая триггер в «0» («1» на инверсном выходе). После этого записывающее внешнее устройство снова может начинать процедуру записи в память.

Для правильной работы схемы необходимо, чтобы длительность сигнала «Зап» должна быть больше минимальной длительности сигнала WR памяти. Период следования сигналов «Зап» должен быть больше суммы длительности сигнала «Зап» и задержки переключения счетчика. Период следования сигналов «Чт» должен быть больше суммы времени выборки адреса памяти и задержки переключения счетчика.

Информационные буферы с непрерывным обменом

Буферы с непрерывным обменом – это буферы с непрерывным режимом работы. С одним из устройств такой буфер общается непрерывно, а с другим только в момент обращения со стороны этого устройства. В данном случае необходимо иметь два счетчика адреса памяти, выходные коды которых надо мультиплексировать с помощью мультиплексора.

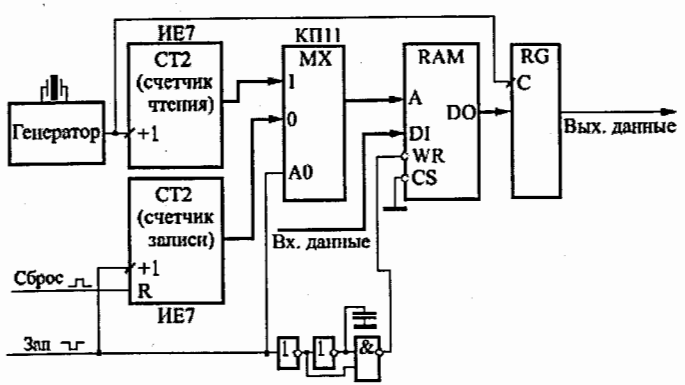


Рис.12.23 Передающий буфер с непрерывным режимом работы

На рис.12.23 показан простой однонаправленный и передающий буфер, то есть когда одно устройство только записывает в память информа-

цию (в нужные моменты), а на другое устройство постоянно выдается информация, читаемая из всех подряд адресов памяти. Счетчик чтения непрерывно перебирает адреса памяти с частотой тактового генератора и считываемая из памяти информация через регистр RG выдается на выход устройства. В режиме записи по сигналу «Зап» мультиплексор подает на адресные входы памяти код счетчика записи и управляющий сигнал WR. Такая последовательность сигналов позволяет записать в память входные данные по адресу записи со счетчика записи и не менять содержимое ячеек памяти с другими адресами.

Перед началом режима записи в память счетчик записи сбрасывается в «0» по сигналу «Сброс». После каждой операции записи по заднему фронту сигнала «Зап» код на выходе счетчика записи увеличивается на «1». То есть для того чтобы записать всю память, необходимо сбросить счетчик и произвести столько циклов записи, сколько ячеек имеется в памяти.

Условия правильной работы схемы следующие. Счетчики должны быть синхронными для быстрого переключения, а память нетактируемая, с раздельными входами и выходами для данных. Сигнал WR должен иметь достаточную длительность для записи информации в память. За время действия сигнала «Зап» должен успеть сработать мультиплексор и должна записаться информация в память. Выходной регистр должен срабатывать по фронту управляющего сигнала. Длительность периода тактового сигнала должна быть больше суммы задержки выборки адреса памяти и задержки переключения счетчика чтения. За период следования сигнала «Зап» информация должна успеть записаться в память и переключиться счетчик записи.

Недостаток приведенной организации буфера состоит в том, что при проведении цикла записи в память на выходе схемы будет не та информация, которая должна читаться из памяти в данный момент. Для преодоления этого недостатка надо производить запись в память только в те моменты, когда выходная информация буфера не важна. Например, если речь идет о буфере контроллера видеомонитора, то запись в память можно производить только во время кадрового гасящего импульса, когда на экране ничего не отображается.

13. Микроконтроллер

13.1. Введение

Структура ЭВМ содержит несколько основных компонентов, которым отведены определённые функции и которые выполняются определённым способом. Два таких компонента впервые были описаны в 1833 году Чарльзом Бебиджем в проекте «Аналитической машины». Бебидж ввёл название устройства, названного мельницей, в котором производятся действия над величинами, и «склад», где хранятся значения величин и результаты выполняемых операций. В современных ЭВМ это, соответственно, арифметико-логическое устройство (АЛУ) и оперативное запоминающее устройство (ОЗУ).

АЛУ является частью процессорного устройства компьютера, которое выполняет инструкции, а также управляет информацией, поступающей в машину. Все компоненты компьютера, в основном, работают по принципу последовательной обработки данных. Идёт ли речь о микрокомпьютере, персональном компьютере или о мощном суперкомпьютере – все они решают задачи в последовательной режиме, в каждый момент времени, анализируя и исполняя только одну инструкцию, после чего переходят к следующей. Даже решение простой задачи, например, сложить два числа или перейти от строчных букв к прописным, требует выполнения нескольких мелких процедур (микрокоманд).

13.2. Обобщенная структурная схема компьютера

На рис.13.1 представлено схематическое изображение типового компьютера, которое помогает понять его внутреннее устройство и прин-

ципы работы. Однако, по существу, данные элементы характерны для любой вычислительной системы.

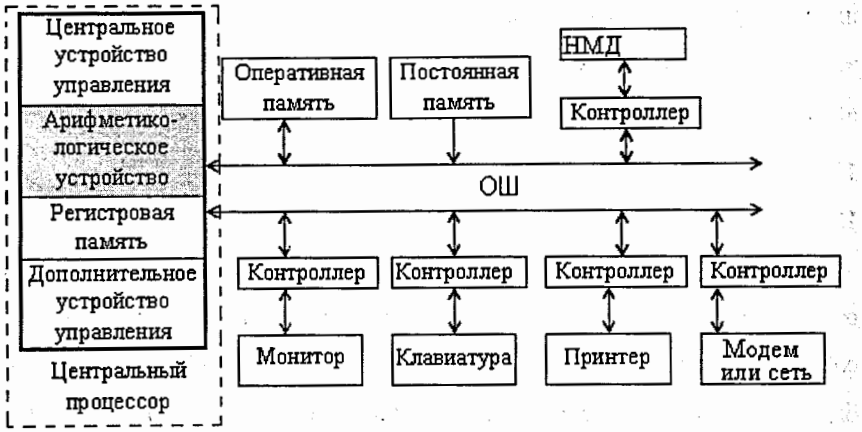


Рис.13.1. Схематическое изображение типового компьютера

Центральный процессор

Центральный процессор в современных компьютерах представляет собой *микропроцессор*, то есть выполнен на одной микросхеме (БИС, СБИС). В его состав входят: центральное устройство управления – комплекс средств автоматического управления процессами передачи и обработки информации; арифметико-логическое устройство (АЛУ) – устройство, осуществляющее обработку информации и выработку признаков управляющих сигналов; внутренняя память процессора – регистровая память и постоянная память устройства управления.

Оперативная и постоянная память

Модули оперативной и постоянной памяти связаны с микропроцессором напрямую при помощи общей системой шины (ОШ).

Устройства ввода-вывода

Устройства ввода-вывода (монитор, клавиатура, накопители на магнитных носителях и т. д.) связаны с микропроцессором через контроллеры

ввода-вывода, которые, в свою очередь, связаны с микропроцессором через системную ОШ.

Процессор является ядром ЭВМ (рис.13.1). Он осуществляет обработку данных и функции управления системой. К функциям управления системой относятся: инициирование операций ввода-вывода; управление доступом к основной памяти (работа с виртуальной памятью); обработка системных прерываний; организация многозадачных режимов работы.

Организация центрального процессора определяется архитектурой и принципами работы ЭВМ (состав и форматы команд, организация памяти). Структурно эти средства разбиваются на следующие устройства (рис.13.1): центральное устройство управления; арифметико-логическое устройство (АЛУ); внутреннюю память; управляющие устройства, связанные с конкретными устройствами вычислительной машины.

Центральное устройство управления принимает и расшифровывает команды, формирует адреса команд, формирует последовательности управляющих сигналов и обеспечивает координацию работы всех узлов посредством выработки синхронизирующих сигналов.

Управляющая память входит в состав центрального устройства управления и относится к классу постоянной памяти. Этот вид памяти используется для хранения микропрограмм. Ее отличает очень высокое быстродействие и небольшая емкость, определяющаяся количеством команд и микрокоманд в системе команд центрального процессора.

Внутренняя регистровая память входит в состав памяти первого уровня. Она непосредственно связана с АЛУ и другими блоками центрального процессора и имеет скорость работы, соизмеримую со скоростью работы блоков процессора (но меньше, чем у управляющей памяти). Память выполнена на триггерных элементах и имеет небольшую емкость.

13.3. Арифметико-логическое устройство

13.3.1. Классификация АЛУ

АЛУ предназначены для выполнения арифметических и логических операций на данными (операндами). Все АЛУ классифицируются следующим образом (рис.13.2): по способу действий над операндами; по виду обрабатываемых чисел; по организации действий над операндами; по типу структуры.

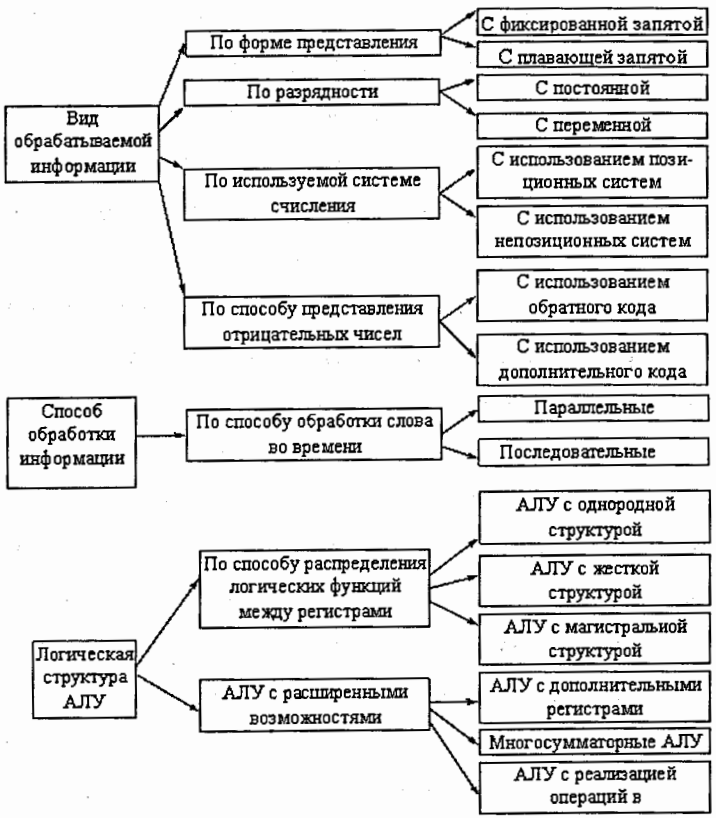


Рис.13.2. Классификация АЛУ

По способу действий над операндами АЛУ бывают последовательного и параллельного действия. В последовательных АЛУ действия над операндами производятся последовательно, разряд за разрядом, начиная с

младшего. В параллельных АЛУ все разряды операндов обрабатываются одновременно.

По виду обрабатываемых чисел все АЛУ могут производить следующие операции: с двоичными числами с фиксированной или плавающей запятой и с двоично-десятичными числами. При действии с двоично-десятичными числами АЛУ должны содержать схему десятичной коррекции. Схема десятичной коррекции преобразует полученный результат таким образом, чтобы каждый двоично-десятичный разряд не содержал цифру больше «9». При записи числа с фиксированной запятой запятая фиксируется после младшего разряда, если число целое, и перед старшим, если число меньше «1». При записи чисел с плавающей запятой выделяется целая часть, которая называется мантиссой, и показатель степени, который называется порядком и характеризует положение запятой.

По организации действий над операндами АЛУ делятся на блочные и многофункциональные. В АЛУ блочного типа отдельные блоки предназначены для выполнения действий с двоичными числами, с двоично-десятичными числами, с числами с фиксированной и с плавающей запятой. В многофункциональных АЛУ одни и те же блоки обрабатывают числа с фиксированной запятой, плавающей запятой и двоично-десятичные числа.

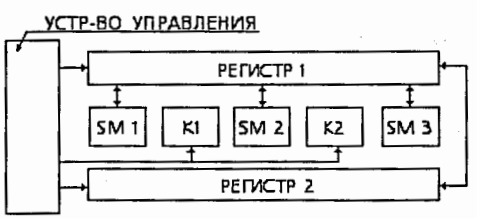


Рис.13.3. Схематическое изображение многофункционального АЛУ

На рис.13.3 показано схематическое изображение многофункционального АЛУ. Коммутаторы К1 и К2 объединяют сумматоры 1, 2 и 3 для действий над числами с фиксированной запятой. Для действий над числа-

ми с плавающей запятой коммутатор К2 объединяет сумматоры 2 и 3 для обработки мантисс, а коммутатор К1 отсоединяет первый сумматор от второго. Сумматор 1 обрабатывает порядки.

По структуре АЛУ делятся на АЛУ с непосредственными связями и многосвязные АЛУ. В АЛУ с непосредственной связью вход регистра приемника связан с выходом регистра источника операндов и регистра, в котором происходит обработка. Например, в схеме на рис.13.4 суммирование происходит следующим образом. Операнды подаются в регистр 1. Регистр 2 является накапливающим сумматором. Он суммирует слагаемые, поступающие в разные моменты времени, и передает результат в регистр 3.

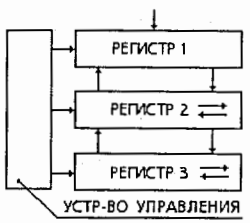


Рис.13.4. Схематическое АЛУ с непосредственной связью

В многосвязных АЛУ входы и выходы регистров приемников и источников информации подсоединяются к одной шине, а распределение входных и выходных данных происходит под действием управляющих сигналов.

13.3.2. Логическая структура АЛУ

Арифметико-логическое устройство функционально можно разделить на две части: микропрограммное устройство (устройство управления), задающее последовательность микрокоманд (команд) и операционное устройство, в котором реализуется эта заданная последовательность микрокоманд.

На рис.13.5 показана типовая структурная схема АЛУ и его связь с другими блоками в ЭВМ. АЛУ имеет следующий состав: регистры Рг1÷Рг7, в которых обрабатывается информация, поступающая из оперативной или пассивной памяти; N_1, N_2,..,N_r – логические схемы, реализующие

обработку слов по микрокомандам, поступающим из устройства управления процессора.

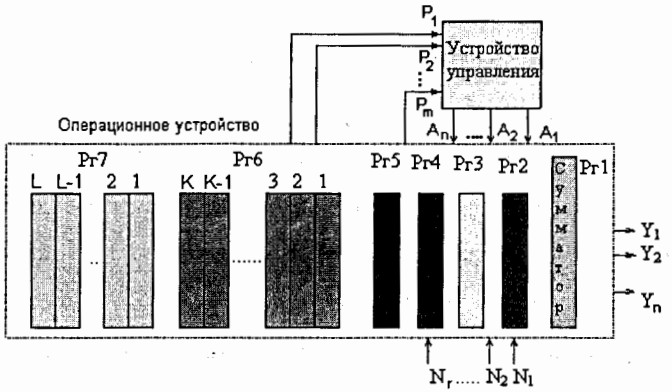


Рис.13.5. Структурная схема АЛУ

Закон обработки информации в АЛУ задает команда М, которая записывается в виде последовательности микрокоманд A_1, A_2,…, A_{n-1}, A_n. В АЛУ различают два вида микрокоманд: *внешние*, то есть такие микрокоманды, которые поступают в АЛУ от внешних источников и вызывают в нем те или иные преобразования информации (микрокоманды A_1, A_2,…, A_{n-1}, A_n, на рис.13.5), и *внутренние*, которые генерируются в операционном устройстве АЛУ и воздействуют на микропрограммное устройство, изменяя естественный порядок следования микрокоманд. Например, АЛУ может генерировать признаки в зависимости от результата вычислений j, w, Q и др. Здесь j – признак переполнения, w – признак отрицательного числа, Q – признак равенства «0» всех разрядов числа. На рис.13.5 внутренние микрокоманды обозначены P_1, P_2,..,P_m. Результаты вычислений из АЛУ передаются в ОЗУ по кодовым шинам записи Y_1, Y_2,.., Y_n.

Перечислим регистры, входящие в АЛУ, и их функции:

- Рг1 – сумматор (или сумматоры) – основной регистр АЛУ, в котором образуется результат вычислений;

- Рг2, РгЗ – регистры слагаемых, сомножителей, делимого или делителя;
- Рг4 – адресный регистр (или адресные регистры Рг5), предназначен для запоминания (иногда и формирования) адреса операндов и результата;
- Рг6 – К индексных регистров, содержимое которых используется для формирования адресов;
- Рг7 – L вспомогательных регистров, которые по желанию программиста могут быть аккумуляторами, индексными регистрами или использоваться для запоминания промежуточных результатов.

Часть операционных регистров является программно-доступной, они могут быть адресованы в команде программы для выполнения операций с их содержимым. К ним относятся: сумматор, индексные и некоторые вспомогательные регистры. Остальные регистры программно недоступные, так как они не могут быть адресованы в программе.

Сложность логической структуры АЛУ в процессоре или в микропроцессоре в определенной степени определяется количеством отличающихся друг от друга микроопераций, необходимых для выполнения всего комплекса задач, поставленных перед АЛУ. На входе каждого регистра собраны соответствующие логические схемы, обеспечивающие необходимые связи между регистрами, что позволяет реализовать заданный набор микроопераций. Выполнение операций над словами сводится к выполнению последовательности микрокоманд, которые управляют передачей слов в АЛУ и действиями по преобразованию слов. Порядок выполнения микрокоманд определяется алгоритмом выполнения операций. Следовательно, связи между регистрами АЛУ и функции, которые должны выполнять регистры, зависят в основном от принятой методики выполнения операций: арифметических, логических и специальной арифметики.

Перечень операций, выполняемых в АЛУ, зависит от назначения цифровой вычислительной машины и от функций, выполняемых АЛУ при обес-

печении работы остальных устройств машины. При представлении операций в виде последовательностей микроопераций АЛУ должно состоять из элементов, реализующих эти микрооперации.

Таким образом, структура АЛУ определяется набором микроопераций, необходимых для выполнения заданных арифметических, логических и специальных операций, а задачу построения АЛУ можно свести к задаче определения требуемого набора микроопераций, который позволяет составить микропрограмму любой из заданных операций. Такой набор можно получить, если записать микропрограммы всех операций, выполняемых в АЛУ, и выбрать из них все, входящие в микропрограммы хотя бы один раз. Однако если алгоритм операций выбирать произвольно, то количество микроопераций, входящих в полный набор, может оказаться слишком большим и, следовательно, АЛУ будет сложным. Для получения более простой схемы АЛУ алгоритмы арифметических и логических операций следует выбирать из условия получения минимального набора микроопераций. При этом необходимо учитывать требование по обеспечению заданного быстродействия АЛУ (слишком ограниченный набор микроопераций может привести к *длинным микропрограммам некоторых операций*, что увеличивает время выполнения данных операций).

13.4. Алгоритмы сложения, вычитания и умножения в АЛУ

13.4.1. Алгоритм сложения и вычитания

Структурная схема микропрограммы сложения двоичных слов, реализованная на АЛУ, показана на рис.13.6. Выполнение этого алгоритма состоит в следующем:

1. Первое слагаемое «а» записывается в Рг1, и анализируется его знак. Если знак отрицательный, то операнд инвертируется и передается на Рг3, если положительный – передается без инверсии через Рг2 на Рг3.

2. Второе слагаемое «b» также устанавливается на Рг1, и анализируется его знак. Если знак отрицательный, то операнд инвертируется и передается на Рг2, если положительный – сразу начинается суммирование операндов на Рг1.

3. После суммирования анализируется знак результата. Если результат отрицательный, то он инвертируется, если положительный – добавляется «+1» к младшему разряду результата и выполняется анализ признаков переполнения.

4. В случае переполнения разрядной сетки машины формируется признак переполнения φ, если переполнение отсутствует, то выполняется переход на конец микропрограммы сложения.

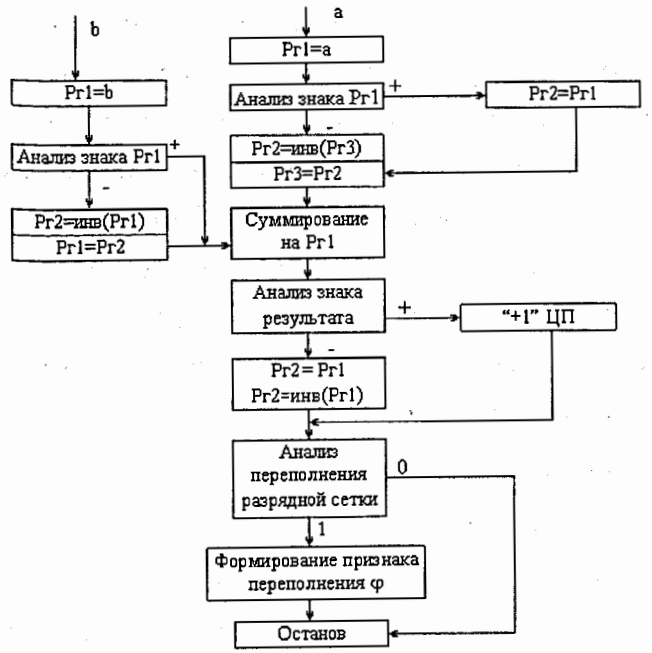


Рис.13.6. Структурная схема алгоритма сложения и вычитания

Для того чтобы структурная схема, показанная на рис.13.6, могла выполнять операцию вычитания, достаточно перед выполнением операции проинвертировать знак второго слагаемого.

13.4.2. Алгоритм умножения

Умножение двоичных чисел с фиксированной запятой можно свести к последовательности сдвигов и сложений. Наиболее удобен следующий алгоритм: умножение начинается с младших разрядов множителя, который сдвигается вправо, сумма частичных произведений также сдвигается вправо, множимое – неподвижно. На рис.13.7 показана графическая интерпретация этого алгоритма.

Рис.13.7. Схема алгоритма умножения двоичных чисел

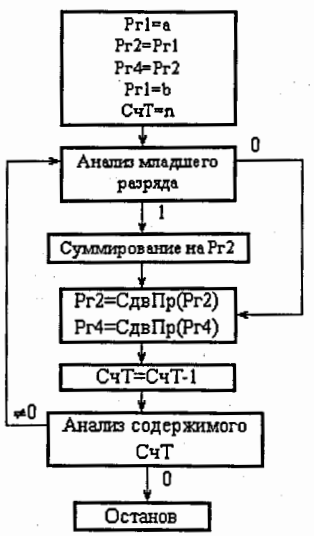


Рис.13.8. Схема алгоритма умножения двоичных чисел

Структурная схема микропрограммы умножения (рис. 13.8) включает в себя следующие шаги:

1) в начале операции все регистры устанавливаются в нулевое состояние;

2) множимое и множитель располагаются в определенных регистрах, предусматриваются также регистры, в которых образуется сумма частичных произведений;

3) анализируется младший разряд множителя (если он имеет значение «1», то к сумме частичных произведений прибавляется множимое);

4) производится сдвиг суммы частичных произведений и множителя на один разряд вправо;

5) действия 3 и 4 повторяются n раз (n – разрядность сомножителей).

13.5. Алгоритм конвейерного (матричного) умножения

Для повышения производительности процессора при выполнении операций его операционное устройство может строиться по блочному принципу. В блочных ОУ реализуется несколько функционально-независимых исполнительных устройств, выполняющих различные операции или различные группы операций. Например, три блока целочисленного сложения, два блока целочисленного умножения, по одному блоку деления, сложения и умножения с плавающей запятой и т. д. Эти устройства работают параллельно, каждый обрабатывая свои операнды. Управление этими устройствами осуществляется с помощью так называемых длинных командных слов (*Very Long Instruction Word – VLIW*). Командные слова включают инструкции для каждого исполнительного устройства, а также указатели на них.

Преимуществом блочных ОУ является более высокая производительность, достигаемая за счет распараллеливания вычислений. В то же время использование таких устройств не всегда эффективно, поскольку не всегда есть возможность загрузить все исполнительные устройства в каждом такте, и в результате часть из них простаивает. Более эффективными часто оказываются конвейерные операционные устройства, поскольку конвейеризовать вычисления в ряде случаев проще, чем распараллелить, что связано с повторением однотипных вычислений в алгоритмах.

Типичным примером конвейерных операционных устройств могут служить так называемые *матричные умножители*. Свое название они получили, во-первых, потому, что включают фактически матрицу операционных элементов (сумматоров), а во-вторых, поскольку одной из наиболее очевидных сфер их применения является умножение матриц. Рассмотрим процесс умножения двух двоичных 4-разрядных положительных чисел:

$$
\begin{array}{cccccccc}
 & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & \times & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & + & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 \\
 & & + & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & \\
 & + & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & \\
\hline
c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0
\end{array}
$$

По косвенной схеме умножения на устройстве с одним сумматором и набором регистров для реализации этого умножения необходимо в общем случае выполнить четыре шага, на каждом из них выполняются операции: умножение «А» на очередной разряд b_i, сложение $A \times b_i$ с текущей суммой частичных произведений и сдвиг новой полученной суммы на один разряд вправо. Таким образом, время на выполнение этого умножения будет

$$T_{умн} = 4(t_{\&} + 4 \times t_{sm} + t_{sh}),$$

где $t_{\&}$ – задержка на логическом вентиле (при умножении А на b_i), $4t_{sm}$ – задержка на сложение при использовании сумматора с последовательным переносом, t_{sm} – задержка одноразрядного полного двоичного сумматора, которую можно принять равной $2t_{\&}$, t_{sh} – задержка на один сдвиг, которую также можно приравнять $2t_{\&}$.

Если вычислять все произведения Axb_i, а результирующие переносы учитывать только при завершении сложения всех слагаемых (вместо того, чтобы рассчитывать их на каждом шаге), то процесс умножения можно значительно ускорить. Схема, реализующая этот алгоритм, приведена на рис.13.9 и носит название *умножителя Брауна*. Каждая линейка суммато-

ров представляет собой сумматор со *сквозным переносом* (ССП), который широко применяется в различных арифметических устройствах.

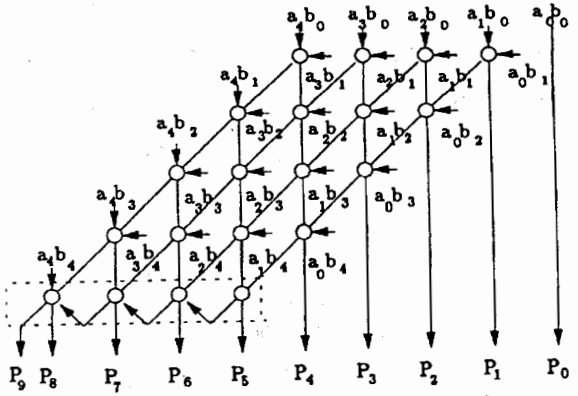


Рис.13. 9. Схема матричного умножителя

13.6. Структура АЛУ цифрового процессора

В качестве примера рассмотрим АЛУ цифрового процессора, предназначенного для решения задач цифровой обработки сигналов. На рис.13.10 показана блок-схема АЛУ такого процессора.

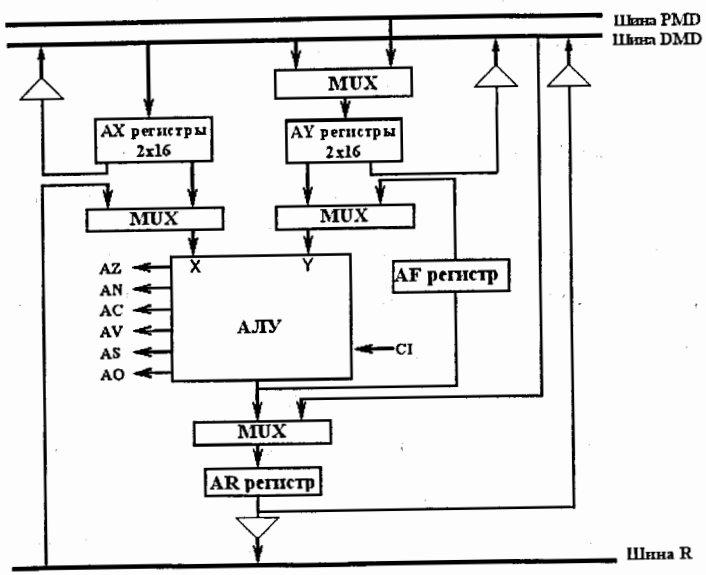


Рис.13.10. Блок-схема АЛУ цифрового процессора

АЛУ имеет три 16-битных регистра, доступных для программиста: X и Y – регистры операндов, AR – регистр результата. АЛУ использует входной сигнал переноса (CI), который означает *бит переноса* в регистре арифметического состояния. АЛУ генерирует шесть статусных сигналов: результат «0» (AZ); отрицательный (AN); перенос (AC); переполнение результата (AV); знак (AS); состояние частного (AQ). Входной порт X может принимать данные из двух источников: из блока регистров AX или с шины результата. Шина результатов (R) соединяет выходные регистры всех вычислительных устройств, позволяя им быть непосредственно операндами инструкций. Блок регистров AX, в свою очередь, состоит из двух регистров: AX0 и AX1. Эти регистры читаемы и могут быть записаны через шину DMD. Выход блока регистров AX таков, что один может обеспечивать операнд для АЛУ, в то время как другой может записываться в память через шину DMD.

Таблица 13.1. Стандартные функции АЛУ цифрового процессора

R=X+Y	Сложение X и Y
R=X+Y+CI	Сложение X и Y с *переносом*
R=X – Y	Вычесть Y из X
R=X – Y – CI – 1	Вычесть Y из X с *заемом*
R=Y – X	Вычесть X из Y
R= – X	Арифметическое отрицание X
R= – Y	Арифметическое отрицание Y
R=Y+1	Инкремент Y
R=Y – 1	Декремент Y
R=PASS X	Результат равен операнду X
R=PASS Y	Результат равен операнду Y
R=O (PASS 0)	Очистить результат
R=X AND Y	Логическое и (AND) X и Y
R=X OR У	Логическое или (OR) X и У
R=X XOR У	Исключающее логическое или (XOR) X и У
R=NOT X	Логическое отрицание X
R=NOT Y	Логическое отрицание У

Входной порт Y также может принимать данные из двух источников: из набора регистров AY или из регистра обратной связи AF. Блок регист-

ров AY состоит из двух регистров AY0 и AY1. Эти регистры читаемы и могут быть записаны через шину DMD, а также могут быть записаны через шину PMD. Выход блока регистров AY совпадает по своим возможностям с блоком регистров AX. Результат работы АЛУ загружается либо в регистр обратной связи AF, либо в регистр результата AR. Регистр обратной связи – внутренний регистр АЛУ, который позволяет использовать результат непосредственно как операнд Y. Регистр результата AR может записываться как на шину DMD, так и на шину результатов. Он также непосредственно загружаем с шины DMD. Набор инструкций позволяет осуществить чтение этих регистров с шины PMD, но при этом необходимо использовать устройство обмена между шинами DMD-PMD. Любые регистры, связанные с АЛУ, могут как читаться, так и писаться в одном цикле. Регистры читаются в начале цикла и записываются в конце. Новое значение, записанное в регистр, не может быть считано до начала следующего цикла. В табл.13.1 приведены стандартные функции, реализуемые в АЛУ.

13.7. АЛУ на базе цифровых ИМС

Развитие микроэлектроники способствовало появлению малогабаритных, высоконадёжных и экономичных устройств АЛУ на базе цифровых ИМС. Одним из представителей является ИМС АЛУ серии К1533.

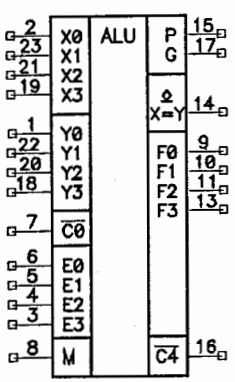


Рис.13.11. Функциональное обозначение микросхемы АЛУ серии К1533

На рис.13.11 показано функциональное обозначение АЛУ серии К1533, а в табл.13.2 приведены выполняемые им операции.

Таблица 13.2. Операции, выполняемые АЛУ серии К1533

Выбор функции	Положительная логика		
	Логические операции (M=1)	Арифметико-логические операции (M=0)	
		$\overline{C0}=1$	$\overline{C0}=0$
0000	$\overline{X}$	X	$X+1$
0001	$\overline{X \vee Y}$	$X \vee Y$	$X \vee Y+1$
0010	$\overline{X}Y$	$X \vee \overline{Y}$	$X \vee \overline{Y}+1$
0011	0	-1(дополнение до2)	0
0100	$\overline{XY}$	$X+X\overline{Y}$	$X+X\overline{Y}+1$
0111	$X\overline{Y}$	$X\overline{Y}-1$	$X\overline{Y}$
1000	$\overline{X} \vee Y$	$X+XY$	$X+XY+1$
1001	$\overline{X \oplus Y}$	$X+Y$	$X+Y+1$
1010	Y	$X \vee \overline{Y}+XY$	$X \vee \overline{Y}+XY+1$
1011	XY	$XY-1$	XY
1100	1	$X+X$`	$X+X$`$+1$
1101	$X \vee \overline{Y}$	$X \vee Y+A$	$X \vee Y+A+1$
1110	$X \vee Y$	$X \vee \overline{Y}+A$	$X \vee \overline{Y}+A+1$
1111	X	$X-1$	X

14. Программируемые логические интегральные схемы (ПЛИС)

14.1. Основные типы ПЛИС и их классификация

Программируемая логическая интегральная схема (ПЛИС) – электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования. Программируемые логические интегральные схемы появились в конце семидесятых годов прошлого столетия как альтернатива программируемым логическим матрицам (ПЛМ), от которых они отличаются как по архитектуре, так и по технологии изготовления.

В последние годы производители больших ПЛИС стали рекламировать свои изделия в качестве альтернативы *классическим процессорам цифровой обработки сигналов* (ЦПОС). На основе больших ПЛИС стало возможно реализовать высокопроизводительные системы параллельной обработки цифровой информации. Появились библиотеки стандартных узлов ЦПОС (например, цифровые фильтры, анализаторы и т. д.), пригодные для встраивания в общий проект на базе ПЛИС. Рассмотрим основные типы ПЛИС.

ПЛИС типа PLD и ПЛИС с программируемой макрологикой

Плис типа PLD (*Programmable Logic Device*) – программируемое логическое устройство, которое содержит две логические матрицы – матрицу «И» и матрицу «ИЛИ». На матрицу «И» подаются прямые и инверсные значения входных сигналов, а на матрицу «ИЛИ» подаются сформированные в матрице «И» логические произведения входных сигналов. Таким образом, устройство типа PLD реализует логическую сумму произведений от входных переменных.

ПЛИС PLD в зависимости от того, структура какой из логических матриц является фиксированной, а какой – программируемой, делятся на PROM (*Programmable Read Only Memory*), PAL (*Programmable Array Logic*) и PLA (*Programmable Logic Array*). В устройствах PROM логическая матрица «И» является фиксированной, а матрица «ИЛИ» – программируемой. Поскольку все комбинации входных сигналов жестко зашиты в матрице «И», использование PROM наиболее эффективно для реализации устройств с небольшим числом входов и большим числом логических произведений. Примерами таких ПЛИС могут служить отечественные схемы К556РТ1, К556РТ2, К556РТ21, принцип реализации которых основан на том, что любая комбинационная функция может быть приведена в виде логической суммы (операция «ИЛИ») от логических произведений (операций «И»).

Недостаток такой архитектуры – слабое использование ресурсов программируемой матрицы «ИЛИ». Поэтому дальнейшее развитие получили микросхемы, построенные по архитектуре программируемой матричной логики PAL. PAL – это ПЛИС, имеющие программируемую матрицу «И» и фиксированную матрицу «ИЛИ». К этому классу относится большинство современных ПЛИС с небольшой степенью интеграции. Например, отечественные PAL: КМ1556ХП4, КМ1556ХП6, КМ1556ХП8, КМ1556ХЛ8 и др. Эти устройства сочетают гибкость, свойственную PAL, с быстродействием, свойственным PROM.

В ПЛИС типа PLA обе логические матрицы являются программируемыми, что делает их наиболее гибкими из всех устройств типа PLD. Расплатой за это является снижение быстродействия и повышение цены.

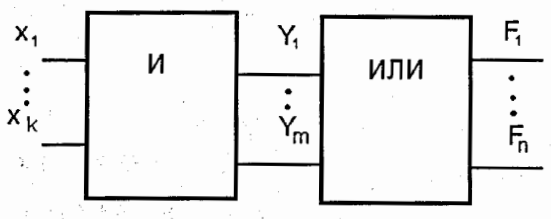


Рис.14.1. Функциональная схема ПЛИС типа PLA

На рис.14.1 показана функциональная схема ПЛИС типа PLA. Входные переменные $x_1,..,x_k$ поступают на вход матрицы «И», в которой образуется m термов (конъюнкций) от входных переменных. Затем эти термы подаются на вход матрицы «ИЛИ», в которой формируются выходные функции от полученных термов в матрице «И». Таким образом, ПЛИС реализует различные функции от комбинации входных переменных, которая определяется при её программировании.

ПЛИС с программируемой макрологикой представлена на рис.14.2. Этот тип ПЛИС содержит единственную программируемую матрицу «И-НЕ» или «ИЛИ-НЕ», но за счёт многочисленных инверсных обратных связей спо-

собен формировать сложные логические функции. К этому типу относятся, например, PLHS501 и PLHS502 фирмы SIGNETICS, имеющие матрицу «И-НЕ», а также XL78C800 фирмы EXEL, основанную на матрице «ИЛИ-НЕ».

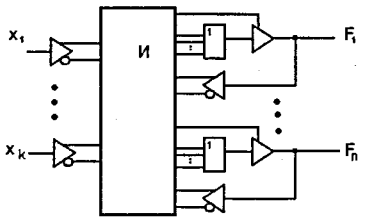


Рис.14.2. Функциональная схема ПЛИС с программируемой макрологикой

Все вышеперечисленные архитектуры ПЛИС содержат небольшое число программируемых ячеек. К настоящему времени они морально устарели и применяются для реализации относительно простых устройств, для которых не существует готовых интегральных схем средней степени интеграции.

Программируемые коммутируемые матричные блоки

Программируемые коммутируемые матричные блоки (ПКМБ) – это ПЛИС, содержащие несколько матричных логических блоков (МЛБ), объединённых коммутационной матрицей (рис.14.3). В зарубежной литературе они получили название CPLD (*Complex Programmable Logic Devices*).

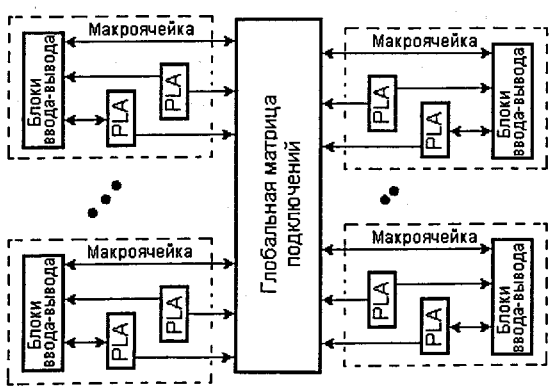


Рис.14.3. Матрица CPLD MAX 7000 фирмы Altera

Каждый МЛБ представляет собой структуру типа PLA, программируемую матрицу «И», фиксированную матрицу «ИЛИ» и макроячейки. Макроячейки содержат относительно крупные программируемые логические блоки, соединённые с внешними выводами и внутренними шинами (рис.14.4). ПЛИС типа CPLD, как правило, имеют высокую степень интеграции (до 10000 эквивалентных вентилей и до 256 макроячеек) и архитектуру, весьма удобную для реализации цифровых автоматов. Функциональность CPLD кодируется в энергонезависимой памяти, поэтому нет необходимости их перепрограммировать при включении. К этому типу относятся ПЛИС семейства MAX5000 и MAX7000 фирмы ALTERA, схемы XC7000 и XC9500 фирмы XILINX, а также большое число микросхем других производителей (Atmel, Vantis, Lucent и др.). На рис.14.3 показана функциональная структура CPLD матрицы MAX 7000 фирмы Altera, а на рис.14.4 – функциональная схема макроячейки этой матрицы.

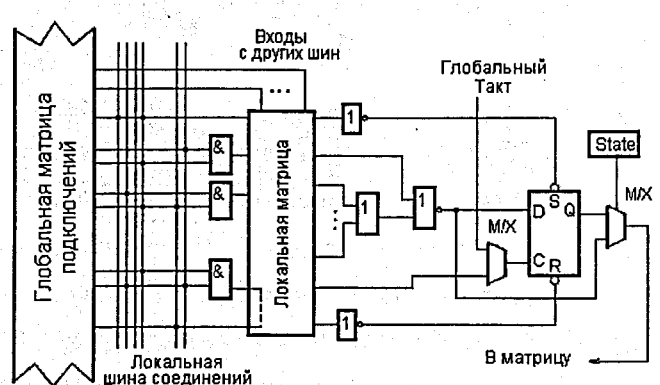


Рис.14.4. Макроячейка матрицы MAX 7000 фирмы Altera

Программируемые пользователем вентильные матрицы

Другой тип архитектуры ПЛИС – программируемые пользователем вентильные матрицы (ПВМ), состоящие из логических блоков (ЛБ) и коммутирующих путей – программируемых соединений или шин маршрутизации матрицы (рис.14.5). В зарубежной литературе такой тип ПЛИС полу-

чил название FPGA (*Field Programmable Gate Array*). Логические блоки FPGA состоят из одного или нескольких относительно простых логических элементов, в основе которых лежит таблица перекодировки LUT (*Look-up table*), программируемый мультиплексор, *D-триггер*, а также цепи управления. Таких простых элементов в FPGA может быть достаточно много. Например, у современных FPGA ёмкостью до 1 млн вентилей число логических элементов может достигать нескольких десятков тысяч.

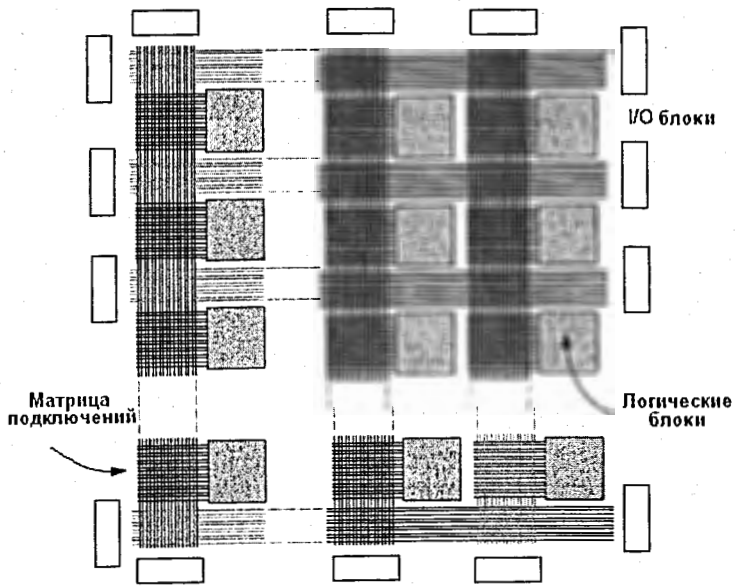


Рис.14.5. Структура матрицы FPGA

Некоторые семейства FPGA имеют встроенные реконфигурируемые модули памяти EAB (*embedded array block*), что делает их архитектуру весьма удобным средством реализации алгоритмов цифровой обработки сигналов, основными операциями в которых являются перемножение, умножение на константу, суммирование и задержка сигнала. К классу FPGA относятся такие ПЛИС: XC2000, XC3000, XC4000 Virtex и Spartan, фирмы XILINX; ACT1, ACT2 фирмы ACTEL, а также семейство FLEX8000 фирмы ALTERA и некоторые другие FPGA фирмы Atmel и Vantis.

По сравнению с CPLD FPGA выигрывают в следующих параметрах: в неограниченном количестве перепрограммирования; в логической емкости (на два-три порядка большей емкости в числе эквивалентных логических вентилей, чем CPLD); в малом энергопотреблении. Кроме того, FPGA, как правило, имеют статическое ОЗУ, которое почти не потребляет энергии при отсутствии переключений, и у их на порядок выше надежность, чем у CPLD.

14.2. Область применения ПЛИС

Одноуровневые ПЛИС типа PAL, благодаря простоте реализации и высокому быстродействию, широко используются в практических применениях. В цифровых устройствах, реализованных на PAL, все выходные сигналы имеют одинаковую временную задержку, что позволяет строить логические схемы с высоким быстродействием. Однако на их базе строятся только достаточно простые логические схемы.

Двухуровневые PLD для реализации логических функций «ИЛИ» используют матрицу второго уровня. Основной недостаток для данного типа ПЛИС заключается в том, что время формирования выходных сигналов на первом и втором уровнях в ПЛИС *различно*. Выравнивание выходных сигналов по времени требует дополнительных ресурсов матрицы. На практике для того чтобы обеспечить одинаковое время формирования выходных сигналов, на выходе логической схемы используют регистры, программируют выходы ПЛИС как регистровые. Это приводит к снижению быстродействия схемы, а также к необходимости формирования сигналов синхронизации и управления регистром, что в итоге усложняет реализацию.

ПЛИС типа PLD широко используются для построения различных по сложности и возможностям цифровых устройств. Это цифровая обработка сигнала, цифровая видео- аудиоаппаратура, высокоскоростная передача данных, криптография, проектирование и т. д.

Альтернативой ПЛИС являются заказные БИС (большие интегральные схемы), которые при мелкосерийном и единичном производстве существенно дороже, и компьютеры (микроконтроллеры), которые из-за программного способа реализации алгоритмов медленнее ПЛИС.

14.3. Структура ПЛИС типа PLD

Основой PLD служат программируемые матрицы «И» или «ИЛИ» (одноуровневые PLD) или последовательность матриц «И» и «ИЛИ» (многоуровневые PLD). В их структуру могут входить также блоки входных и выходных буферных каскадов. Входные буферы преобразуют однофазные сигналы в парафазные, а выходные буферы обеспечивают требуемую выходную нагрузочную способность выходов и разрешают или запрещают подключение выходов ПЛИС на внешние шины.

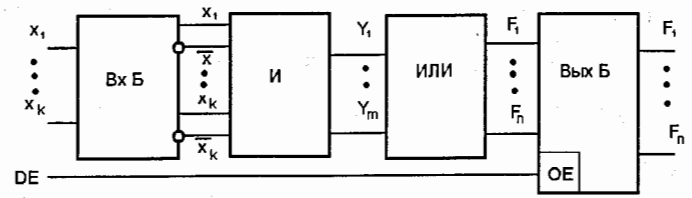


Рис.14.6. Функциональная схема двухуровневой PLD

На рис.14.6 показана структура двухуровневой PLD. Входные переменные $x_1,..,x_k$ через входной буфер поступают на вход матрицы «И», в которой образуется m термов (конъюнкций) от входных переменных, представленных в прямой или в инверсной форме. Затем термы подаются на вход матрицы «ИЛИ», в которой формируются выходные функции от полученных термов в матрице «И». Таким образом, ПЛИС реализует различные функции от комбинации входных переменных, которая определяется при её программировании. Для программирования используются отладочные среды, позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках (Verilog, VHDL).

Программируемые матрицы «И» и «ИЛИ» представляют собой коммутационные матрицы, в которых перемычками программируются минтермы дизъюнктивных нормальных форм функций этих элементов. Вначале перемычки выполнялись в виде пережигаемых тонких проводников. В настоящее время перемычки выполняются в виде МОП-транзисторов с плавающим затвором, как в электрически перепрограммируемом ПЗУ. Большие многоуровневые PLD отличаются только тем, что несколько ПЛИС собраны на одном кристалле и объединены программируемым полем связей.

14.4. Программируемая пользователем вентильная матрица

Программируемая пользователем вентильная матрица (ППВМ) является дальнейшей разработкой программируемых ПЗУ и программируемых ПЛИС. Основатели ППВМ Росс Фримен и Берни Вондершмит (фирма «Xilinx») изобрели первую коммерчески пригодную ППВМ типа XC2064 в 1985 году. Эта FPGA, имеющая программируемые вентили и программируемые соединения между вентилями, положила начало новой технологии и новому рынку. В 1990-х годах произошёл резкий скачок интереса к ППВМ, возросла их сложность и объёмы производства. Если в начале 1990-х годов ППВМ использовались в основном в области телекоммуникаций и сетей связи, то к концу десятилетия они нашли своё применение в потребительских товарах, в автомобильной промышленности и других отраслях.

ППВМ представляет собой матрицу маловходовых (от двух до пяти входов) логических элементов, триггеров и линий связи, соединяемых перемычками из полевых транзисторов, которые программируются изменением уровня электрического поля в затворах этих транзисторов. Затворы всех программирующих полевых транзисторов подключены к выходам триггеров одного длинного сдвигового регистра, который заполняется при программи-

ровании ППВМ. Прошивка обычно хранится в ПЗУ, расположенном рядом с ППВМ, и после включения питания или по сигналу сброса она автоматически переписывается в программирующий сдвиговый регистр. Этот процесс называется *конфигурированием*. Так как основу ППВМ составляют триггеры, хранящие прошивку, то они изготавливаются по технологии микросхем статического ОЗУ.

Современные ППВМ содержат блоки умножения и суммирования, которые широко применяются при обработке сигналов, а также логические элементы (LUT) и блоки их коммутации. Микросхемы типа ППВМ обычно используются для обработки сигналов, так как они имеют больше логических элементов и более гибкую архитектуру, чем CPLD. Программа конфигурирования ППВМ хранится в распределённой памяти, которая может быть выполнена как на основе энергозависимых ячеек статического ОЗУ (подобные микросхемы производят фирмы Xilinx и Altera) – в этом случае программа не сохраняется при выключении электропитания микросхемы, так и на основе энергонезависимых ячеек *памяти* – в этом случае программа сохраняется при исчезновении электропитания. Если программа хранится в энергозависимой памяти, то при каждом включении питания микросхемы необходимо заново конфигурировать её при помощи начального загрузчика, который может быть встроен в саму ППВМ. Основные производители ППВМ – Atmel, Altera, Lattice-semiconductor, Xilinx.

14.4.1. Архитектура ППВМ

Программируемая пользователем вентильная матрица – это полупроводниковое устройство, которое может быть конфигурировано производителем или разработчиком после изготовления. Она программируется с помощью исходного кода на языке проектирования (например, VHDL), на котором можно описать логику работы микросхемы. ППВМ могут быть модифицированы практически в любой момент в процессе их использования. Они состоят из конфигурируемых логических блоков, подобных переключателям с множеством входов и одним выходом (логические вентили). В цифровых схемах такие переключатели реализуют базовые двоичные операции AND, NAND, OR, NOR и XOR. В большинстве современных микропроцессоров функции логических блоков фиксированы и не могут модифицироваться.

Принципиальное отличие ППВМ состоит в том, что и функции блоков, и конфигурация соединений между ними могут меняться с помощью специальных сигналов, посылаемых схеме. В некоторых специализированных интегральных схемах (ASIC) используются логические матрицы, аналогичные по структуре ППВМ, однако они конфигурируются один раз в процессе производства, в то время как ППВМ могут постоянно перепрограммироваться и менять топологию соединений в процессе использования. Однако такая гибкость требует существенного увеличения количества транзисторов микросхемы.

ППВМ включают в себя три главных программируемых элемента: *коммутируемые логические блоки* (ПЛБ), *блоки ввода-вывода* (БВВ) и *внутренние связи маршрутизации*. ПЛБ являются функциональными элементами для построения логики пользователя, БВВ обеспечивают связь между контактами корпуса и внутренними сигнальными линиями. Программируемые ресурсы внутренних связей обеспечивают управление путями соединения входов и выходов ПЛБ и блоков ввода-вывода на соответствующие сети. Все каналы трассировки имеют одинаковую ширину (одинаковое количество проводников). Большинство блоков БВВ вписываются либо в одну строку (по высоте), либо в один столбец (по ширине) массива вентилей.

Логический блок классической ППВМ состоит из логического устройства LUT (Look Up Table) на четыре входа и триггера (рис.14.7). В по-

следние годы производители начали переходить на LUT с шестью входами в высокопроизводительных частях схемы, объясняя это необходимостью повышения производительности.

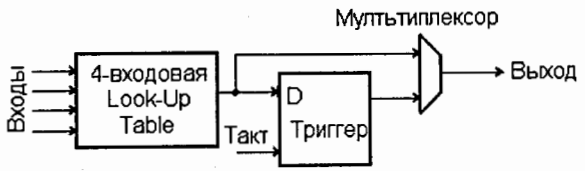


Рис.14.7. Типичный логический блок

Типичный логический блок имеет LUT на четыре входа и вход синхронизации (Такт). Выход блока только один, он может быть регистровым или не синхронизированным (определяется состоянием мультиплексора). Поскольку сигналы синхронизации в ППВМ (а часто и другие сигналы, распараллеленные на большое количество входов) трассируются особым образом специальными трассировочными цепями, управление этими сигналами делается отдельно. Для приведённого примера архитектура представлена на рис.14.7, расположение контактов логического блока показано на рис.14.8.

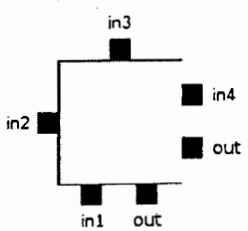


Рис.14.8. Расположение контактов логического блока

Входы на схеме расположены на отдельных сторонах логического блока, выходной контакт может трассироваться в двух каналах: либо справа от блока, либо снизу. Выходные контакты каждого логического блока могут соединяться с трассировочными сегментами в смежных каналах. Аналогично контактная площадка блока ввода-вывода (pad) может соединяться с трассировочным элементом в любом смежном канале.

В месте пересечения вертикальных и горизонтальных каналов создаются переключательные блоки. При такой архитектуре для каждого проводника, входящего в переключательный блок, существуют три программируемых переключателя (рис.14.9), которые позволяют ему подключаться к трём другим проводникам в смежных сегментах канала. Модель или топология выключателей, используемая в этой архитектуре, является планарной или доменной топологией переключательных блоков. В этой топологии проводник трассы номер 1 подключается только к проводнику трассы номер 1 в смежных каналах, проводник трассы номер 2 подключается только к проводникам трассы номер 2 и так далее. На рис.14.9 показаны соединения в переключательном блоке.

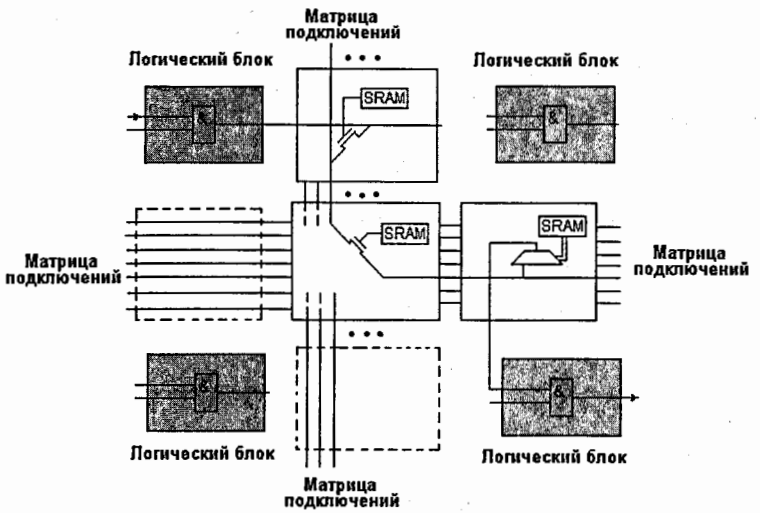


Рис.14.9. Организация связей между логическими блоками в ППВМ

Современные семейства ППВМ расширяют перечисленные выше возможности и включают *встроенную функциональность высокого уровня*. Имея эти общие функции, можно сократить площадь кристалла, к тому же эти функции будут работать быстрее, чем если их создавать на базе *примитивов*. Примерами таких функций являются мультиплексоры, блоки

цифровой обработки сигналов, встроенные умножители, быстрая логика ввода-вывода и встроенная память.

Типичным примером ППВМ могут служить микросхемы семейства Spartan фирмы XILINX, в которой множество конфигурируемых логических блоков объединяются с помощью матрицы соединений (рис.14.5). Характерными для ППВМ архитектур являются блоки ввода-вывода, позволяющие реализовать двунаправленный ввод/вывод, третье состояние и т. п.

Особенностью современных ПЛИС является возможность тестирования узлов с помощью порта JTAG (B-scan), а также наличие внутреннего генератора (Osc) и схем управления последовательной конфигурацией.

Фирма Altera пошла по пути развития ППВМ-архитектур и предложила в семействе FLEX10K так называемую двухуровневую архитектуру матрицы соединений. Логические элементы (ЛЭ) объединяются в группы – логические блоки (ЛБ). Внутри логических блоков ЛЭ соединяются посредством локальной программируемой матрицы соединений, позволяющей соединять любой ЛЭ с любым ЛЭ. Логические блоки связаны между собой и с элементами ввода/вывода посредством глобальной программируемой матрицы соединений (ГПМС). Локальная и глобальная матрицы соединений имеют непрерывную структуру, то есть для каждого соединения выделяется непрерывный канал.

Дальнейшее развитие ПЛИС идёт по пути создания комбинированных архитектур, сочетающих удобство реализации алгоритмов цифровой обработки сигналов (ЦОС) на базе LUT и реконфигурируемых модулей памяти, характерных для ППВМ-структур, и многоуровневых ПЛИС с удобством реализации цифровых автоматов на CPLD архитектурах. Так, например, ПЛИС APEX20K фирмы Altera содержат в себе логические элементы всех перечисленных типов, что позволяет применять ПЛИС как основную элементную базу для «систем на кристалле» (*system-on-chip* , SOC).

14.5. Особенности программирования ПЛИС

В настоящее время для больших ПЛИС в основном используются две технологии для хранения информации о конфигурации: статическое ОЗУ или электрически перепрограммируемое ПЗУ, причем создание файла конфигурации невозможно без автоматизированных систем проектирования. Такие системы выпускают все ведущие производители ПЛИС. При работе в подобных системах конфигурация схемы, которая должна быть получена внутри ПЛИС, или алгоритм ее работы задается либо на текстовом языке описаний (VDHL или Verilog), напоминающем язык программирования высокого уровня (например, Си++), либо на графическом уровне – в виде электрической схемы (ADHL). В дальнейшем все этапы работы, включая программирование или загрузку ПЛИС, выполняет автоматизированная система.

15. Автоматизация функционально-логического проектирования цифровых устройств на программируемых логических схемах фирмы «Altera» с использованием системы проектирования «MAX plus II»

15.1. Программируемые логические интегральные схемы

Все электронные изделия условно можно разделить на два типа. К первому типу относятся устройства, которые обрабатывают информацию, представленную в аналоговом виде. Ко второму типу относятся устройства, перерабатывающие информацию, представленную в цифровой форме. Еще недавно представители первого типа доминировали над цифровыми устройствами. Схемотехника цифровых устройств явилась огромным достижением науки конца XX века. Ее преимущество перед схемотехникой аналоговых устройств основано на следующих факторах:

- информация в цифровых устройствах не подвержена воздействию окружающей среды (температуры, влажности, напряжения питания и т. д.);

• переработку цифровой информации можно осуществлять с неограниченной точностью;

• микросхемы цифровых устройств допускают большую степень интеграции, включающую десятки миллионов транзисторов.

На первом этапе цифровая техника развивалась в направлении создания устройств переработки цифровой информации с помощью «жесткой» логики. Такие устройства предназначены для выполнения одной какой-либо операции. Создание устройств на «жесткой» логике привело к появлению универсальных микросхем, которые можно конфигурировать в соответствии с выбранным законом функционирования самим разработчиком. Эти микросхемы получили название «программируемые логические интегральные схемы (ПЛИС)». Одна микросхема ПЛИС может заменить несколько сотен корпусов микросхем традиционной «жесткой» логики.

Особенностями современных ПЛИС являются: низкая стоимость, высокое быстродействие, широкие функциональные возможности, многократность перепрограммирования, низкое энергопотребление, гибкость архитектуры.

15.2. Этапы процесса проектирования цифровых устройств

Процесс проектирования устройства обработки цифровой информации с использованием ПЛИС заключается в следующем: описание логики функционирования на языке используемого программного средства; выполнение автоматизированного синтеза; проведение моделирования и настройки выбранной ПЛИС с помощью программатора. Важной особенностью является тот факт, что время разработки, даже очень сложных схем, может составлять несколько часов, а для изменения алгоритма работы устройства достаточно перепрограммировать ПЛИС.

Любое сложное цифровое устройство состоит из элементарных логических устройств (вентилей), реализующих какую-либо функцию алгебры

логики. С помощью логических вентилей строятся как комбинационные схемы, так и последовательностные. Современное схемотехническое проектирование любых схем невозможно без применения компьютерных методов расчета и проектирования электронных схем. Рассмотрим принципы проектирования основных схемотехнических элементов и синтеза их функционирования с использованием САПР «MAX+plusII» фирмы «Altera».

15.3. Проектирование логических элементов с использованием САПР «MAX plus II»

Задание на разработку функциональной схемы

Провести моделирование и отладку схемы счетчика, выполненного на *JK-триггерах*: Задана схема счетчика по модулю 5 (рис.15.1), где С – тактовый вход, R – сброс всех триггеров в «0», Q0, Q1,Q2 – информационные выходы.

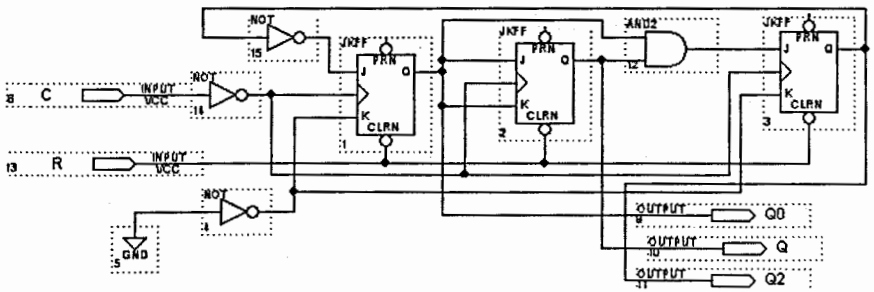


Рис.15.1 Схема счетчика по модулю 5

Проведение работы

Шаг 1. Создание файла проекта.

После запуска пакета системы автоматизированного проектирования (САПР) «MAX+plusII» открывается главное окно программы (рис.15.2). Создадим проект схемы в графическом редакторе. Для этого создадим новый файл (крайний левый значок панели инструментов).

В открывшемся диалоговом окне «New» выберем пункт «Graphic Editor file» и нажмем кнопку «ОК», при этом автоматически откроется окно

графического редактора. Сохраним новый файл проекта (через меню «File > Save») под именем *counter5* (расширение будет присвоено автоматически). Имя файла проекта привяжем к имени проекта. Для этого выберем пункт «Set Project to Current File» (в подменю «Project» меню «File» главного меню рабочего окна). Для создания графического проекта нам понадобятся библиотеки (\Maxplus\Max2lib\Prim), макрофункций (\~\Max2lib\Mf) и параметризированных мегафункций (\~\~\Mega_Lpm).

Рис.15.2 Главное окно САПР «MAX+plus II»

Шаг 2. Создание схемы проекта

Для размещения элементов в окне редактора используется диалоговое окно «Enter Symbol», которое можно открыть через меню «Symbol» основного меню редактора (пункт «Enter Symbol») или с помощью контекстного меню (двойным щелчком левой клавиши мыши по свободному пространству открытого окна редактора). После этой операции выбранный элемент будет размещён именно в нужном месте.

Выбрать и установить необходимый элемент можно двумя способами:

- Набрать имя элемента (примитива, мега- или макрофункции) в окне «Symbol Name» диалогового окна «Enter Symbol» и нажать «OK»;
- Выбрать необходимую библиотеку в окне «Symbol Libraries» диалогового окна «Enter Symbol» и двойным щелчком левой клавиши мыши открыть её (рис.15.3). Затем аналогичным образом выбрать необходимый элемент в окне «Symbol File».

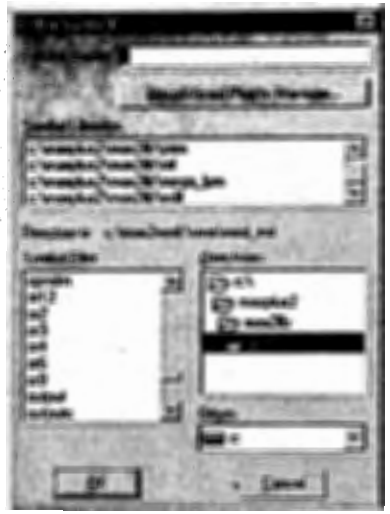


Рис.15.3. Окно «Symbol File»

Выводы элементов можно соединить сигнальными линиями. Для проведения сигнальной линии необходимо совместить указатель курсора (конец стрелки) с выводом элемента, при этом курсор автоматически превращается в инструмент рисования ортогональных линий (перекрестие). После этого проводится необходимая линия (при нажатой и удерживаемой левой кнопке мыши). За один приём можно провести два ортогональных отрезка. Если этого недостаточно, то процедуру можно повторять, начиная с конца проведённой линии или с вывода другого элемента. Логика разрабатываемого проекта может быть также реализована с использованием примитивов. Разместим требуемые элементы заданной схемы (рис.15.1) и выполним необходимые соединения.

Шаг 3. Компиляция и создание символа проекта

Следующий этап – компиляция и создание символа проекта для включения его в файл проекта верхнего уровня. Перед компиляцией мож-

но выполнить проверку корректности введённого проекта. Проверка осуществляется через подменю «Project» (меню «File» главного меню рабочего окна) путём выбора пункта «Save&Check» или щелчком левой кнопки мыши на пиктограмме соответствующего инструмента основной панели инструментов.

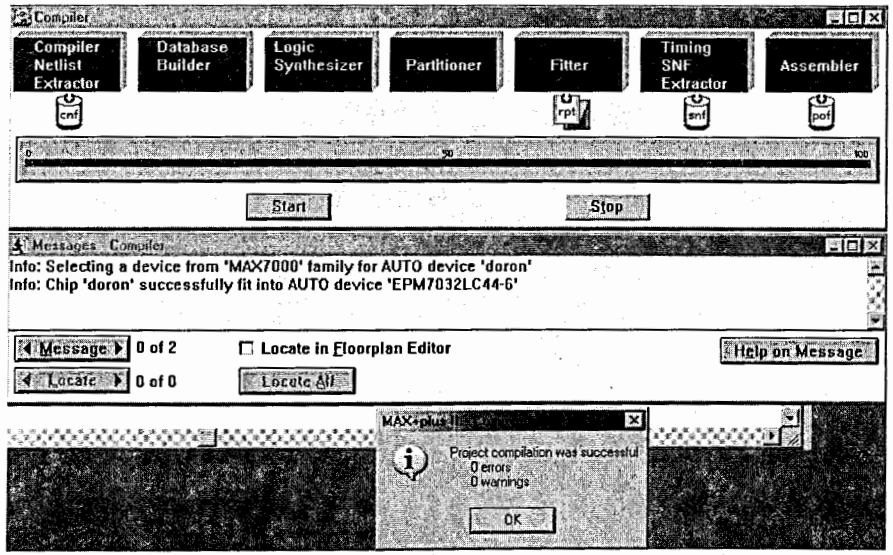


Рис.15.4. Итоги компиляции

Компиляция осуществляется также через подменю «Project» путём выбора пункта «Save&Compile» или опять же с помощью соответствующего инструмента основной панели инструментов. После компиляции процессор сообщением проинформирует нас о том, что для реализации данного проекта могут быть использованы ПЛИС семейства «MAX7000», а именно ПЛИС типа «EPM7032LC44» (в корпусе PLCC с 44 выводами).

Создание символа проекта осуществляется через подменю «Project», в котором следует выбрать пункт «Create Default Symbol» – этот пункт становится доступным только после закрытия окна компилятора. Созданный символ будет помещён в каталог проекта. Использование созданных

символов, так же как и элементов других библиотек, производится через диалоговое окно «Enter Symbol».

Шаг 4. Создание проекта и символа комбинационной схемы

Для создания проекта этого устройства используем редактор временных диаграмм («Waveform Editor»). Для этого следует нажать кнопку открытия нового файла на панели инструментов, в открывшемся диалоговом окне «New» отметить пункт «Waveform Editor file», в соседнем окне выбрать расширение .WDF (расширение .SCF используется для моделирования) и нажать «ОК», после чего открывается окно редактора (рис.13.5).

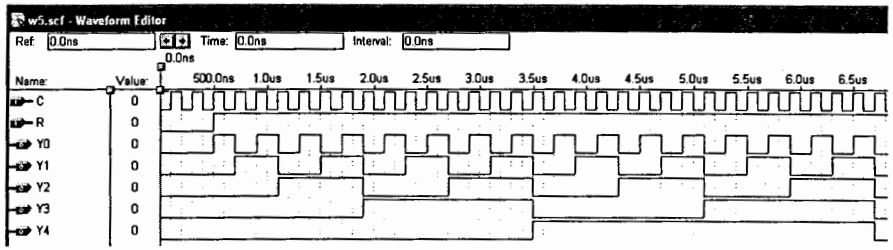


Рис.15.5. Временные диаграммы

Окно редактора имеет четыре поля, разделённых вертикальными линиями. Первое поле слева («Name») предназначено для ввода имени узла, во втором поле («Type») отображается тип ввода (INPUT, OUTPUT, BIDIR), в третьем поле («Value») показаны состояния выводов, соответствующие положению специальной вертикальной визирной линии, которая при открытии окна установлена в начало горизонтальной оси, размеченной в единицах времени. Для перемещения визира в нужное место необходимо совместить курсор с вершиной визира (при этом курсор примет крестообразную форму) и при нажатой левой кнопке мыши перетащить визир в нужное место. Четвёртое поле предназначено для задания требуемых состояний выводов, при этом используются инструменты с панели инструментов редактора, которая расположена вертикально вдоль левой стороны окна. Активизация панели инструментов происходит только в том случае, если выделен один из

узлов. Чтобы выделить узел, необходимо щёлкнуть левой кнопкой мыши на имени узла, можно также выделить любой участок вдоль горизонтальной оси, при этом границы выделяемых участков привязываются к сетке.

Параметры сетки устанавливаются следующим образом: с помощью пункта «End Time» (меню «File») задаётся максимальное значение временного интервала с указанием единиц измерения, а с помощью пункта «Grid Size» (меню «Options») – шаг сетки. В верхней части экрана расположены окна для точного отсчёта интервалов времени.

В верхнее поле «Node Name» необходимо ввести имя узла, в окне «I/O Type» (рис.15.5) указать тип вывода и нажать «ОК». Кроме того, имена узлов можно вводить другим способом. Для этого необходимо щёлкнуть левой кнопкой мыши в поле «Name», при этом автоматически открывается диалоговое окно «Insert Mode».

Размещённые узлы можно редактировать, перемещать, удалять, размножать (с обязательным редактированием имени или типа, если это необходимо). Для редактирования используется то же диалоговое окно «Insert Mode». Остальные операции осуществляются аналогично тому, как это делается в графическом редакторе.

В нашем случае необходимо задать возможные состояния восьми входов и четырех выходов, соответствующих каждому состоянию входа. Для этого вся горизонтальная ось должна быть разбита на 32 дискрета. Например, установив значение «End Time» 320 нс, а «Grid Size» – 10 нс, получим 32 дискрета с шагом сетки 10 нс. После этого последовательно выделяем узлы и дискреты и, с помощью панели инструментов задаём требуемые состояния входов и выходов (рис.13.5).

Шаг 5. Назначение типа ПЛИС и выводов ПЛИС

Тип ПЛИС, необходимый для реализации проекта, может быть выбран автоматически или назначен вручную. При создании файла проекта

по умолчанию установлен режим автоматического выбора минимальной по объёму ПЛИС, в которой может быть реализован данный проект. При необходимости тип ПЛИС может быть назначен вручную с помощью пункта «Device» (меню «Assign» основного меню системы). После назначения устройства проект необходимо перекомпилировать.

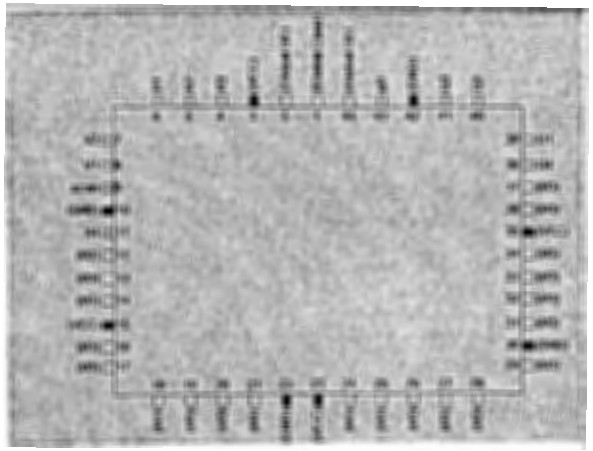


Рис.15.7. Устройство, реализованное на ПЛИС

Выводы ПЛИС первоначально назначаются автоматически (рис.13.7). После завершения работы над проектом необходимо закрепить или переназначить выводы ПЛИС, для того чтобы при возможной последующей доработке (отладке) проекта в составе всего изделия компилятор не мог изменить их назначение. Эта операция выполняется с помощью редактора «Floorplan Editor» (рис.15.8), который запускается либо через меню «MAX+plus II» (в основном меню), либо через панель инструментов.

Все результаты работы над проектом фиксируются в файле *counter5.rpt*, который представляет собой обычный текстовый файл и содержит подробное описание реализованного проекта, в том числе и описание назначения выводов, необходимое для разработки принципиальной схемы целевого устройства. Файл *counter5.rpt* можно открыть через меню «File» или нажатием соответствующей кнопки на панели инструментов.

Для завершения работы над проектом необходимо провести функциональное моделирование. Для этого необходимо создать исходный файл с расширением .SCF, задать тестовые (эталонные) состояния входов, выбрать проверяемые выходы и запустить «Simulator».

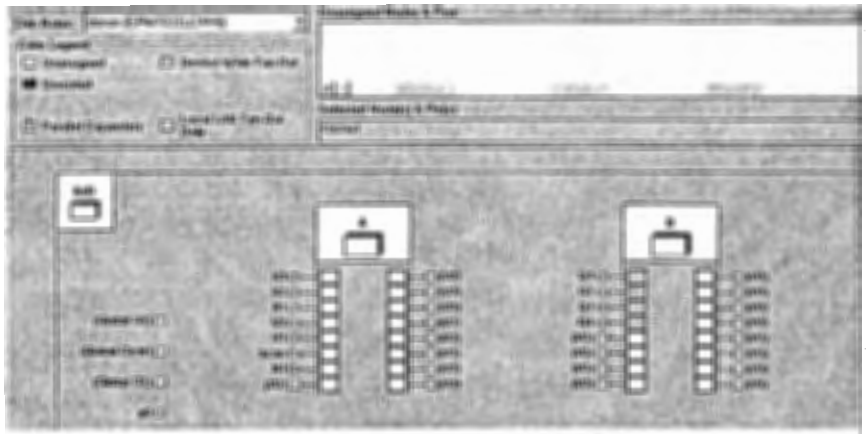


Рис.15.8. Редактор «Floorplan Editor»

Шаг 6. Завершение работы

Для создания исходного файла необходимо открыть новый файл, в диалоговом окне «New» отметить пункт «Waveform Editor file», установить в соседнем поле окна расширение .SCF и нажать «OK», а затем сохранить этот файл через меню «File-Save» основного меню. После этого необходимо связать его с проектом. Для этого нужно войти в меню «Node» и выбрать пункт «Enter Nodes from SNF». После нажатия кнопки «List» в левой панели диалогового окна «Enter Nodes from SNF» появляется список доступных узлов («Available Nodes&Groups»), выделенный синим цветом, который нужно перенести на правую панель («Selected Nodes&Groups»), для чего нужно нажать кнопку со стрелкой, расположенную между панелями. После нажатия кнопки «OK» в окне редактора появляется готовый шаблон (заготовка) для задания тестовых состояний входов. Перед началом ввода

тестовых состояний необходимо в порядке, описанном выше, задать длительность интервала моделирования и установить шаг сетки с учётом временных параметров реальных сигналов, которые будут подаваться на вход ПЛИС в целевом устройстве.

Сами тестовые состояния входов вводятся так же, как и при создании проекта. «Simulator» может быть запущен либо через меню «MAX+plus II» (в основном меню), либо через панель инструментов, при этом открывается диалоговое окно с кнопками «Start» и «Open SCF». Для начала моделирования необходимо нажать кнопку «Start», а для просмотра результатов – «Open SCF».

Заключение

В данном учебном пособии рассмотрен круг вопросов, связанных с изучением, проектированием и применением цифровых элементов, узлов и устройств, которые являются основой для реализации различных средств обработки информации в ЭВМ. В пособии дан только необходимый минимум знаний, который должен иметь разработчик цифровой схемотехники. Для более глубокого и детального изучения автор предлагает использовать книги из списка рекомендованной литературы.

Рекомендуемая литература

1. Угрюмов Е.П. Цифровая схемотехника. БХВ-Петербург, 2004.
2. Новиков Ю.В. Основы цифровой схемотехники. М.: Мир, 2001.
3. Хоровиц П., Хилл У. Искусство схемотехники. М.: Мир, 1986.
4. Тарабрин Б.В. Интегральные микросхемы. Справочник. М.: Радио и связь, 1983.

100 г

Учебное издание

Калинников Владимир Александрович

Основы схемотехники ЭВМ

УНЦ-2011-48

Редактор *М. И. Зарубина*

Получено 14.09.2011. Подписано в печать 21.12.2011.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 10,5. Уч.-изд. л. 8,28. Тираж 100 экз.
Заказ № 57543.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@jinr.ru
www.jinr.ru/publish/