И. С. Потемкин

Функциональные узлы цифровой автоматики

МОСКВА

ОГЛАВЛЕНИЕ

ББК 32.965
П 64
УДК 681.32

Рецензент Е. П. Угрюмов

Потемкин И. С.

П 64 Функциональные узлы цифровой автоматики. — М.: Энергоатомиздат, 1988. — 320 с.: ил.

ISBN 5-283-01478-9

Рассмотрены основные функциональные узлы цифровой автоматики: триггеры, регистры, счетчики, мультиплексоры, ПЛМ, сумматоры, схемы контроля и т. п. Приведены принцип работы, варианты функциональных схем, примеры микросхем, реализующих узел, области применения. Особое внимание уделено системам синхронизации и способам задания и обеспечения таких временных параметров, при которых сбои из-за состязаний отсутствуют.

Для инженерно-технических работников и других специалистов, осваивающих цифровую технику.

П $\frac{2404000000-070}{051(01)-88}$ 259—88 ББК 32.965

Производственное издание

Потемкин Игорь Семенович

ФУНКЦИОНАЛЬНЫЕ УЗЛЫ ЦИФРОВОЙ АВТОМАТИКИ

Редактор *Ю. Н. Колотов*
Редактор издательства *В. И. Петухова*
Художественные редакторы *Т. А. Дворецкова, Г. И. Панфилова*
Технический редактор *Т. Ю. Андреева*
Корректор *М. Г. Гулина*
ИБ № 1057

Сдано в набор 27.01.88. Подписано в печать 22.04.88. Т-11619. Формат 84×108¹/₃₂. Бумага типографская № 2. Гарнитура литературная. Печать высокая. Усл. печ. л. 16,8. Усл. кр.-отт. 16,8. Уч.-изд. л. 18,52. Тираж 60 000 экз. Заказ № 19. Цена 1 р. 40 к.

Энергоатомиздат, 113114, Москва, М-114, Шлюзовая наб., 10

Владимирская типография Союзполиграфпрома при Госкомиздате СССР 600000, г. Владимир, Октябрьский проспект, д. 7

ISBN 5-283-01478-9

ПРЕДИСЛОВИЕ

Внедрение микропроцессорной, и вообще цифровой, техники в устройства управления промышленными объектами требует от специалистов самого различного профиля быстрого освоения этой области знания. Книга посвящена логическому проектированию функциональных узлов цифровой аппаратуры и организации взаимодействия этих узлов. Под функциональным узлом понимается схема обозримой сложности, реализующая законченную функцию, например — дешифратор, регистр, счетчик, мультиплексор и т. п. Для функциональных узлов характерна универсальность: различные цифровые устройства и целые процессоры построены из узлов весьма небольшого числа типов.

В отличие от большинства публикаций в этой книге предотвращение сбоев из-за переходных процессов, гонок рассматривается не как дополнительное мероприятие по повышению надежности, а как вопрос элементарной грамотности проектирования функциональной схемы с позиций соблюдения четко сформулированных норм. Большое внимание уделяется выбору параметров системы синхронизации. Все эти вопросы недостаточно отражены в массовой литературе по цифровой технике, оставаясь предметом узкоспециализированных публикаций, не освещающих проблемы в целом и, как правило, трудных для понимания начинающими специалистами. В результате в опытных образцах цифровой аппаратуры доля ошибок и сбоев, вызванных гонками, весьма велика, что влечет длительную наладку и доводку схемы. Между тем склонность к сбоям из-за гонок закладывается в значительной степени на этапе проектирования функциональной схемы.

Книга компоновалась в большой мере как обобщенный ответ на множество практических вопросов, возникающих у тех, кто начал самостоятельно разрабатывать схемы цифровой аппаратуры; отсюда иногда не совсем традиционный подбор рассматриваемых тем.

Для помощи читателю ряд тем, особенно в первых главах, представлен в форме упражнений: попытка самостоятельно построить схему, даже если она не увенчалась успе-

хом, порождает серию целенаправленных вопросов, в результате предлагаемое решение понимается и усваивается намного лучше. Разбираются типичные ошибки начинающих схемотехников. Тем, кто желает всерьез овладеть логическим проектированием, настоятельно рекомендуется не торопиться «заглотать» сразу всю книгу, а останавливаться, оглядываться и выполнять встречающиеся упражнения. В качестве материала почти для всех упражнений используются реальные схемы широко применяемых функциональных узлов, чтобы совместить изучение приемов логического проектирования с освоением практических схем узлов.

Особое внимание уделено выявлению самих принципов построения различных узлов с тем, чтобы читатель мог самостоятельно строить требуемые модификации узла (например, при работе на базовых матричных кристаллах) или приспосабливать готовый вариант схемы для нестандартных условий применения. Автор стремится убедить читателя, что число основных принципов, которые нужно один раз понять и запомнить, невелико, а конкретных схем, которые можно на их основе построить самостоятельно, — великое множество.

Применительно к микросхемам средней интеграции в книге рассматриваются те их возможности и ограничения, которые определяются примененным в них вариантом логической схемы, но не вытекают из паспортных данных и не отражены в распространенных справочниках. При этом освещается смысл ряда паспортных данных, регламентирующих протекание переходных процессов, которые пока еще приводятся только в подробных справочниках или ТУ на микросхемы. Объясняется, как использовать эти данные для более полной реализации логических возможностей микросхем.

Главы 1 и 2 по содержанию и стилю изложения ориентированы на начинающих. Схемотехники, имеющие некоторый опыт, могут «пробежать» их «по диагонали», останавливаясь лишь на менее знакомых разделах. Таковыми, вероятно, окажутся § 1.5, 1.8—1.10, 2.3, 2.8—2.11. Глава 3 — переходная: уменьшается число упражнений, возрастает плотность потока технических сведений. В гл. 4 собраны наиболее сложные узлы комбинационного типа, и при освоении азов цифровой техники ее в первом чтении можно опустить.

Глава 5 открывает вторую часть книги, где рассматриваются схемы, процессы в которых существенно зависят от

времени. Она служит введением в эту часть, поэтому схемотехники, имеющие опыт, могут перейти сразу к базовым главам второй части книги — гл. 6 и 7. Несмотря на то что формально они посвящены в основном лишь триггерам и способам их применения, эти главы целесообразно просмотреть даже достаточно опытным схемотехникам, поскольку вводимые в них положения используются в дальнейшем, а некоторые из них, к сожалению, не очень широко известны. Для начинающих обязательно не только понимание, но и запоминание основных положений гл. 5—8, потому что *нарушение именно этих положений чаще всего приводит к ошибкам в логических схемах, вызывающим гонки и как следствие — нерегулярные, трудно диагностируемые сбои.* В первом чтении начинающие могут опустить лишь § 6.8 и 6.9.

Глава 9 посвящена счетчикам — узлам, без которых редко обходятся даже простейшие схемы цифровой автоматики. Столь же популярны и сдвигающие регистры, описанные в § 10.1. Остальная часть гл. 10, содержащая в основном схемы контроля, несколько сложнее других глав, и начинающие могут вернуться к ней при втором чтении.

Глава 11 — это эксперимент по возможно менее строгому изложению процесса синтеза цифрового автомата с рекомендациями по решению наиболее часто встречающихся практических вопросов.

В список литературы включены лишь источники, которые в наибольшей степени могли бы послужить второй ступенькой для расширения кругозора по конкретному вопросу. Без такого ограничения начинающий читатель, брошенный в густые библиографические дебри, практически оказывается вовсе лишенным литературной поддержки.

В работе над книгой большую и очень разностороннюю помощь оказала автору Е. А. Потемкина, за что он приносит ей искреннюю благодарность. Автор также благодарен рецензенту Е. П. Угрюмову и редактору Ю. Н. Колотову за их большую и квалифицированную работу с рукописью, а главное — за освещение с иных точек зрения совершенно очевидных, как автору ранее казалось, положений. Это благотворно повлияло на содержание книги.

Автор с благодарностью примет любые замечания, направленные на устранение недостатков этой книги, которые следует высылать по адресу: 113114, Москва, М-114, Шлюзовая наб., д. 10, Энергоатомиздат.

Автор

ВВЕДЕНИЕ

В процессе разработки функциональных схем цифровых устройств отчетливо выделяются два характерных этапа. На первом этапе, который можно назвать структурным проектированием, заданный неформально (например, объяснением на обычном русском языке) алгоритм разработчик представляет в виде последовательности некоторых операторов, таких, как хранение чисел, их сравнение или сложение, счет, дешифрация кода, коммутация каналов передачи данных и т. п. При этом он старается использовать ограниченный набор по возможности общепринятых операторов.

Эти операторы — довольно крупные концептуальные единицы, поэтому большинство реально встречающихся алгоритмов удается представить небольшим их числом. Структура алгоритма при этом оказывается вполне обозримой, что позволяет за приемлемое время рассмотреть несколько вариантов структуры и выбрать наиболее подходящую. Но поскольку операторы общеприняты, т. е. хорошо знакомы разработчику и его коллегам, работа на языке этих операторов не вызывает затруднений и может носить коллективный характер. Общепринятость операторов не означает, что их список как-либо стандартизован. Это, скорее, неформальное соглашение. Список, в принципе, открыт, но изменяется очень медленно — по оператору в несколько лет.

На основе выбранной структуры можно сформулировать технические требования к схемам, реализующим отдельные операторы, — задать их разрядность, быстродействие, особенности работы, и перейти ко второму этапу — построению самих логических схем, реализующих требуемые операторы. Схемы, реализующие общепринятые операторы, называют *функциональными узлами* или просто *узлами*.

В алгебре логики разработаны процедуры, позволяющие формальным способом построить (*синтезировать*) схему узла, если на него существует формальное задание. Однако эти процедуры, как и любой универсальный инструмент, хороши в среднем, вообще, но малоэффективны

в каждом конкретном случае в том смысле, что почти всегда для конкретного применения удается изобрести что-то существенно лучшее. Функциональные узлы не составляют исключения, и практически все реально используемые удачные схемы не синтезировались формально, а были когда-то изобретены. Поэтому для результативной работы с цифровыми схемами разработчику требуется три вида знаний: наиболее удачных реализаций схем-изобретений; наиболее результативных неформальных инженерных приемов работы с логическими схемами; правил формальных преобразований алгебры логики. В отечественной литературе по цифровой схемотехнике, как и в системе обучения студентов-схемотехников, в силу укоренившихся традиций принято основное внимание уделять лишь третьему, формальному компоненту инженерных знаний, почти полностью игнорируя первые два — именно то самое *know how*, которое в наибольшей степени и отличает профессионализм от дилетантства. Одна из целей книги — снабдить читателя набором удачных конкретных схем узлов, а также пусть и не строго формализованными, но результативными инженерными приемами работы с логическими схемами.

В 70-е годы наша промышленность освоила массовое производство микросхем средней интеграции, реализующих все широко используемые функциональные узлы. Создалось впечатление (не без помощи рекламы изготовителей микросхем), что для аппаратной реализации любого алгоритма достаточно построить его структурную схему, а затем лишь заменить операторы соответствующими корпусами микросхем. Интерес к логическому синтезу, в том числе и к схемам функциональных узлов, заметно упал. Типичной стала фигура схемотехника, совершенно не представляющего, что за схема находится внутри пластмассового корпуса. Счетчик, и все. А какой — это уже не важно. Результат не замедлил сказаться: несмотря на значительно большую логическую эффективность микросхем средней интеграции по сравнению с дискретными компонентами, сроки наладки и доводки устройств оставались огромными. Одна из самых существенных причин заключается в том, что способы стыковки узлов определяются их конкретными схемами, которые весьма разнообразны, а выбор схемы зависит от требований к данному узлу. Игнорирование сведений о содержимом корпусов микросхем и общее невнимание к вопросам схемотехники не позволяют проектировать сразу работоспособные логические устройства.

Новую волну тех же проблем вызвало появление микропроцессоров. Подогреваемая широкой рекламой уверенность в том, что микропроцессорные наборы нужно только программировать и совсем не нужно паять, привела к тому, что во многих организациях простейшую микропроцессорную систему на одной-двух платах не удается заставить работать в течение многих месяцев и даже лет. Восприятие схемотехники как чего-то второстепенного хорошо иллюстрирует состав издаваемой литературы: около трех десятков выпущенных за последние годы книг посвящены вопросам архитектуры и программирования микропроцессов и только одна — Дж. Коффрона — микропроцессорной схемотехнике. И даже она молчаливо базируется на отнюдь не нулевой схемной эрудиции читателя.

Как ни странно, но до сих пор не осознается тот факт, что понимание внутренней логики микросхемы особенно важно именно для специалистов по автоматике или радиотехнике, поскольку цифровые микросхемы изначально создавались для выполнения строго определенных функций в составе ЭВМ. В условиях автоматики и радиотехники они часто выполняют функции, не запланированные в свое время их разработчиками, и грамотное использование микросхем в этих случаях прямо зависит от понимания логики их работы. Хорошее знание тонкостей функционирования схем узлов становится жизненно необходимым при поиске неисправностей, когда нужно определить, имеется ли неисправность в данном узле или же на его вход поступают комбинации сигналов, на которые схема узла не рассчитана. Составление тестов, а тем более разработка самопроверяемых схем также требуют очень хороших знаний принципов работы узлов.

Возродился интерес к логическому синтезу в связи с появлением нового класса микросхем — *программируемых логических матриц (ПЛМ)*, поскольку проектирование устройств на их основе требует от разработчика исчерпывающего раскрытия каждого функционального узла вплоть до уровня простейших логических операций типа НЕ, И, ИЛИ. Пытаться строить самопроверяемые цифровые блоки на базе ПЛМ, используя лишь общие соображения о функциях узлов, стало абсолютно невозможно.

Резкий подъем интереса к рациональному проектированию узлов и блоков на их основе в последние годы вызван освоением промышленностью *матричных БИС (МаБИС)* или *базовых матричных кристаллов (БМК)*. Чтобы любая

БИС была достаточно дешевой, она должна выпускаться огромными тиражами, поэтому в невыгодном положении оказываются разработчики крупносерийной, но еще не массовой аппаратуры, и притом такой, для которой быстродействия универсальных микропроцессоров не хватает. МаБИС оказалась здесь удачным компромиссом: базовый матричный кристалл, пока связи в нем еще не проведены, универсален, поэтому он может иметь широкий спрос и выпускаться миллионными тиражами, а стоимость проведения связей, необходимых для построения схемы на таком кристалле, при использовании систем автоматизированного проектирования не слишком высока. На повестке дня — широкое распространение *кремниевых мастерских (silicone mills)* — предприятий, специализирующихся на проведении межэлементных связей на базовых кристаллах в соответствии с функциональными схемами, полученными от заказчика. Для изготовления партии блоков на МаБИС заказчик должен представить требующееся ему цифровое устройство в виде функциональной схемы на уровне простых узлов или даже логических элементов.

Процесс выявления ошибок в схеме на МаБИС очень трудоемок, поскольку для воздействия и наблюдения доступно лишь то небольшое число точек всей схемы, которое подключено к выводам из корпуса. Поэтому, как правило, прежде чем реализовать цифровой блок на кристалле МаБИС, проводятся его макетирование и отладка на обычных микросхемах малой и средней интеграции в виде платы (ТЭЗа). После этого блок без каких-либо изменений в функциональной схеме переводится на кристалл БИС (принцип ТЭЗ—БИС). Такая технология, а также стремление возможно лучше использовать уже имеющиеся навыки инженеров-схемотехников приводят к тому, что рекомендуемый набор логических фрагментов МаБИС — *библиотека стандартных ячеек* — включает в себя почти тот же привычный набор логических элементов и узлов, что и распространенные серии микросхем. Поэтому многие приемы проектирования схем на МаБИС аналогичны приемам проектирования на основе обычных микросхем малой и средней степени интеграции. Ожидается, что матричные БИС позволят создавать недорогие специализированные логические блоки для тех областей, в которых микропроцессор не обеспечивает требуемого быстродействия. Широкое применение ПЛМ и БМК потребует не менее широкого распространения знаний о методах проектирования логических схем.

ГЛАВА 1

ВВЕДЕНИЕ В АЛГЕБРУ ЛОГИКИ

1.1. Функции алгебры логики

Алгебра логики — это формальный аппарат описания логической стороны процессов в цифровых устройствах. Для овладения дисциплиной можно рекомендовать [1], а для более глубокого изучения отдельных вопросов — [2]. Кратко основные положения алгебры логики изложены в [3 и 4]. В данной главе сведения об алгебре логики приведены лишь в объеме, необходимом для понимания последующих глав, а также наиболее распространенной литературы по цифровой технике.

Алгебра логики имеет дело с *логическими переменными*, которые могут принимать только два значения, называемые различными авторами ИСТИНА и ЛОЖЬ, TRUE и FALSE, ДА и НЕТ, 1 и 0. Наиболее распространено последнее обозначение. При этом 1 и 0 нельзя трактовать как числа, над ними нельзя производить арифметические действия. Это просто короткая, удобная форма обозначения понятий ДА и НЕТ, точно так же, как номер трамвая — это краткая форма наименования его маршрута: если друг за другом идут трамваи № 2 и 1, то это не значит, что они вместе пойдут по маршруту № 3 или 21.

Логические переменные хорошо описывают состояния таких объектов, как реле, тумблеры, кнопки и т. п., т. е. объектов, которые могут находиться в *двух четко различимых состояниях:* включено — выключено. К этим объектам относятся и полупроводниковые логические элементы, на выходе которых может быть лишь один из двух четко различимых уровней напряжения. Чаще более высокий, или просто ВЫСОКИЙ, уровень принимается за логическую единицу, а более низкий, или просто НИЗКИЙ, — за логический нуль.

Функции алгебры логики принимают значения 1 или 0 в зависимости от значений своих аргументов. Если это функция нескольких аргументов, то аргументы образуют некоторое множество комбинаций своих возможных зна-

чений. Одна из форм задания логической функции — *табличная,* когда перечисляются все возможные комбинации значений аргументов и против каждой комбинации записывается значение функции. В выражении: Я поеду в горы, если не будет аврала на работе и удастся достать путевку в альплагерь, — предполагается, что поездка в горы G есть функция двух аргументов: аврала a и путевки p.

Упражнение. Описать функцию G с помощью таблицы. Полезно сначала попытаться сделать это самостоятельно и только потом продолжить чтение.

Решение. Нужно перебрать все возможные комбинации аргументов и для каждой комбинации четко определить исходя из смысла ситуации значение функции G:

при $a=0$ $p=0 \rightarrow G=0$;
при $a=0$ $p=1 \rightarrow G=1$;
при $a=1$ $p=0 \rightarrow G=0$;
при $a=1$ $p=1 \rightarrow G=0$.

Все это принято записывать в виде *таблицы истинности* (табл. 1.1), которая полностью, строго однозначно задает логическую функцию.

Таблица 1.1

Аргументы		Функция	Аргументы		Функция
a	p	G	a	p	G
0	0	0	1	0	0
0	1	1	1	1	0

Для двух аргументов число комбинаций значений аргументов равно $2 \cdot 2 = 2^2 = 4$. Если число аргументов увеличить на единицу, то число строк таблицы удвоится и станет равным $2 \cdot 2 \cdot 2 = 2^3 = 8$. Действительно, для перебора всех комбинаций придется перечислить четыре возможные комбинации a и p, когда третья переменная равна 0, а потом те же четыре комбинации a и p при третьей переменной, равной 1. Вообще для n аргументов число комбинаций их значений, т. е. число строк m таблицы, равно $2 \cdot 2 \cdot 2 \cdot 2 \ldots$, и так n раз, т. е. равно 2^n. При $n=4$ $m=16$; при $n=6$ $m=64$; при $n=10$ $m=1024$.

Чтобы не ошибаться при перечислении комбинаций аргументов, нужно сразу приучиться перечислять их единообразно, в виде последовательности чисел, представленных в двоичной системе счисления. Например, комбинации трех

переменных нужно перечислять в следующем порядке: 000, 001, 010, 011, 100, 101, 110, 111, итого восемь двоичных чисел — от 0 до 7.

Практическим препятствием для повсеместного использования табличной формы задания функций является быстрый рост числа строк таблицы. При пяти входных переменных столбик таблицы займет всю страницу, а таблица от семи переменных (128 строк) становится уже совершенно необозримой. К счастью, на первых порах работы с логическими схемами в основном приходится иметь дело с функциями лишь двух — четырех аргументов, а если аргументов больше, то таблицу обычно удается разбить на части.

1.2. Булевский базис

Набор трех логических функций: НЕ, И, ИЛИ называют *булевским* или *булевым базисом* в честь английского математика конца XIX в. Джорджа Буля, исследовавшего эти функции. Алгебру, в которой различные логические функции выражаются через эти три функции, называют *булевой алгеброй.*

Функция **НЕ** — это функция одного аргумента (другие названия: *отрицание, инверсия*). Функция обычно обозначается чертой над аргументом:

$$Y = \overline{a},$$

где Y — логическая функция; a — аргумент. Встречаются и другие обозначения: $Y = \text{НЕ}a$; $Y = \neg a$. Функция отрицания равна 1, когда ее аргумент равен 0, и наоборот:

$$\text{ПОГАШЕН} = \overline{\text{ГОРИТ}}.$$

Если утверждение ГОРИТ истинно, то утверждение ПОГАШЕН будет ложно, и наоборот. Отрицание отрицания аргумента равно самому аргументу: НЕ (НЕ ВКЛЮЧЕН) = = ВКЛЮЧЕН, или если

$$Y = \overline{a}, \text{ то } \overline{Y} = \overline{\overline{a}} = a.$$

Электронный логический элемент, реализующий функцию НЕ в виде определенных уровней напряжения или тока, называют *инвертором.* Инвертор на функциональных схемах изображается, как показано на рис. 1.1, *а*. Вход — слева, выход — справа. На выходной (или входной, как на рис. 1.1, *б*) линии, в месте соединения ее с прямоугольником, изображается кружок — *символ инверсии.*

Стрелки на входных и выходных линиях как в случае инвертора, так и в случае изображения других логических элементов не ставят: это запрещено ЕСКД. Все условное изображение инвертора или любого другого логического элемента на схеме может быть повернуто на 90° так, что вход будет сверху, а выход — снизу, как показано на рис. 1.1, *в*. Другие углы поворота и направления входов

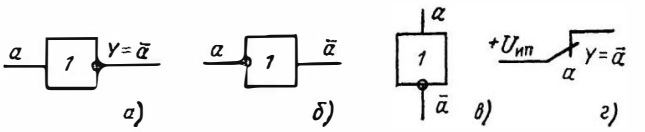


Рис. 1.1. Инвертор:

а, *в* — предпочтительные изображения; *б* — допустимое изображение; *г* — реализация НЕ с помощью размыкающего контакта реле

и выходов не допускаются ЕСКД. Подробнее об условных графических обозначениях логических элементов см. в [5]. В релейно-контактной логике функцию НЕ реализует размыкающий контакт (рис. 1.1, *г*), т. е. такой контакт реле, который замкнут, пока в обмотке нет токового сигнала *a*, и размыкается при подаче тока *a*.

Функция **И** — это функция двух или большего числа аргументов (другие названия: *конъюнкция*, логическое умножение, совпадение, *AND*). Обозначение:

$$Y = a\&b; \quad Y = a \wedge b; \quad Y = a \cdot b; \quad Y = ab.$$

Функция И равна 1 тогда и только тогда, когда все ее аргументы равны 1. Союз «и» естественного языка, как правило, выражает именно это отношение, например: ЛИФТ ПОЙДЕТ, если ДВЕРЬ ЗАКРЫТА **И** КНОПКА НАЖАТА, или в аналитической записи: $L = d \cdot k$, где L — выходной сигнал на двигатель лифта; d — входной сигнал закрытия двери; k — входной сигнал нажатой кнопки.

В релейно-контактной технике функция И реализуется последовательным включением замыкающих контактов, управляемых сигналами-аргументами (рис. 1.2, *а*). Ток в цепи пойдет только тогда, когда все контакты находятся в единичном состоянии. Если хотя бы один контакт находится в нулевом состоянии (разомкнут), то ток не пойдет и функция будет равна 0.

Значения функции И для всех комбинаций значений аргументов a и b приведены в табл. 1.2. Там же приведены

значения и других часто используемых логических функций, о которых речь будет ниже.

Элемент, реализующий функцию И, называют *элемент И* или *конъюнктор*. Элемент И часто используют для управ-

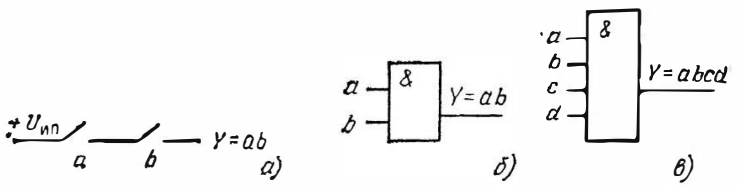


Рис. 1.2. Конъюнктор:
а — реализация операции И на контактах реле; *б*, *в* — условные изображения двух- и четырехвходовых конъюнкторов

ления потоком информации. При этом на один его вход поступают логические сигналы, несущие некоторую информацию, а на другой — управляющий сигнал: пропускать — 1, не пропускать — 0. Элемент И, используемый таким образом, называют *вентиль*.

Таблица 1.2

Аргументы		Функции					
a	*b*	И	ИЛИ	И-НЕ	ИЛИ-НЕ	М2	=
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Условное изображение элемента И в схемах показано на рис. 1.2, *б*.

В преобразованиях алгебры логики применяют следующие соотношения, которые легко доказать с помощью таблиц истинности или на основе аналогии с релейно-контактными схемами:

$$a \cdot 0 = 0; \quad a \cdot 1 = a; \quad a \cdot a = a; \quad a \cdot \bar{a} = 0, \qquad (1.1)$$

где a — аргумент, который может принимать любое значение: 0 или 1.

Функцию И можно построить от любого числа аргументов. На рис. 1.2, *в* показано условное изображение четырехвходового конъюнктора.

Функция **ИЛИ** — это функция двух или большего числа аргументов. Функция ИЛИ равна 1, если хотя бы один из ее аргументов равен 1 (другие названия: *дизъюнкция, OR*). Обозначение: $Y=a \vee b$. Используемые иногда обозначение «+» (знак «плюс») и название функции «логическое сложение» неудачны, и пользоваться ими не следует, поскольку в сложных случаях дизъюнкция будет смешиваться с друмя другими операциями: арифметическим суммированием и сложением по модулю 2. В русском языке функция дизъюнкции выражается союзом «или» во фразах типа: Мы попадем на тот берег, если речка мелкая или мост цел. Формальная запись $ТБ=РМ \vee МЦ$. В логических выражениях принято использовать как латинские, так и русские буквы, как одиночные, так и их сочетания. Значения функции ИЛИ от двух аргументов приведены в табл. 1.2. Полезно сравнить таблицы функций И и ИЛИ: у первой единственная единица находится в самом низу столбца, там, где все аргументы равны 1; у второй единственный 0 находится в самом верху столбца, где все аргументы равны 0. Эти

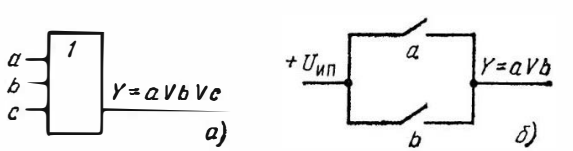


Рис. 1.3. Дизъюнктор:
а — условное изображение; *б* — реализация ИЛИ на контактах

свойства сохраняются при любом числе аргументов обеих функций. В § 1.9 будет показано, что эта антипохожесть не случайна.

Условное изображение на схеме элемента, реализующего функцию ИЛИ — *дизъюнктора* — показано на рис. 1.3, *а*. В релейно-контактных схемах функция ИЛИ реализуется параллельным включением контактов (рис. 1.3, *б*). Полезно запомнить следующие соотношения:

$$a \vee 0 = a; \quad a \vee 1 = 1; \quad a \vee a = a; \quad a \vee \bar{a} = 1. \qquad (1.2)$$

1.3. Произвольные функции и логические схемы

Поскольку значениями логических функций могут быть только 0 или 1, то любые логические функции можно использовать как аргументы других логических функций, т. е.

Таблица 1.3

N	Аргументы a	b	c	Функция Y	N	Аргументы a	b	c	Функция Y
0	0	0	0	0	4	1	0	0	0
1	0	0	1	1	5	1	0	1	0
2	0	1	0	0	6	1	1	0	1
3	0	1	1	1	7	1	1	1	1

строить из простых функций более сложные. Пусть в табл. 1.3 задана произвольная функция Y трех аргументов, и ее нужно выразить с помощью простых функций НЕ, И, ИЛИ.

По правилам построения таблицы функция $Y=1$ тогда, когда:

конъюнкция строки № 1 равна 1, т. е. $\bar{a}\bar{b}c=1$;

конъюнкция $\bar{a}bc=1$ (строка 3);

конъюнкция $ab\bar{c}=1$ (строка 6);

$abc=1$ (строка 7).

Все это можно записать в виде одного общего *аналитического выражения*

$$Y=\bar{a}\bar{b}c\vee\bar{a}bc\vee ab\bar{c}\vee abc. \qquad (1.3)$$

Выражение (1.3) действительно равно 1 только при любой из четырех перечисленных комбинаций значений аргументов. При всех других комбинациях значений все четыре его конъюнкции равны 0. Следовательно, функция, записанная в виде (1.3), эквивалентна функции, заданной табл. 1.3, причем выражена она через функции НЕ, И, ИЛИ.

Полученное аналитическое выражение для функции Y называют *совершенной дизъюнктивной нормальной формой* (*СДНФ*). СДНФ состоит из *элементарных конъюнкций*, соединенных знаками дизъюнкции. Конъюнкцию называют *элементарной,* если в нее не входит по нескольку одинаковых букв. Число элементарных конъюнкций в СДНФ обязательно равно числу единичных значений функции в таблице истинности. В каждую элементарную конъюнкцию СДНФ входят обязательно все аргументы функции, каждый из которых может быть представлен либо в прямой, либо в инверсной форме.

Описанная процедура получения СДНФ есть процедура перехода от табличной формы задания функции к ее аналитической форме. Для обратного перехода — от анали-

тической формы к табличной — достаточно подставить в аналитическую форму по очереди все комбинации значений аргументов и вычисленные значения функции записать в виде таблицы. В случае СДНФ эта процедура очень проста, поскольку каждая из входящих в СДНФ конъюнкций представляет всегда одну строку таблицы, для которой значение функции равно 1. Строка таблицы однозначно определяется по сочетанию инверсных и прямых аргументов: $\bar{a}\bar{b}c$ значит 001, $\bar{a}bc$ значит 011 и т. д.

Поскольку процедуру построения СДНФ в принципе можно применить к таблице, содержащей любое число аргументов при любом расположении единичных значений функции, то можно сделать важный вывод: *с помощью набора функций НЕ, И, ИЛИ можно выразить любую логическую функцию*, сколь бы сложной она ни была. Свойство некоторого набора функций выражать через себя любую функцию называется свойством *полноты* этого набора. Такой полный набор называют *логическим базисом* или просто *базисом*.

Свойство полноты имеет большую практическую ценность: оно позволяет промышленности массово выпускать ограниченный набор логических элементов, из которых разработчики аппаратуры могут строить любые логические схемы. На самом деле выпускаются несколько расширенные по сравнению со строго минимальными *избыточные* наборы, или базисы, позволяющие за счет расширенной номенклатуры экономить суммарное число используемых элементов.

Упражнение. Построить логическую схему, реализующую функцию Y, заданную табл. 1.3, используя элементы НЕ, И, ИЛИ с любым требуемым числом входов.

Решение. Для построения схемы обычно удобнее аналитическая форма представления функции, чем табличная. В данном случае — это выражение (1.3). Схема, реализующая (1.3), показана на рис. 1.4, *а*. Она состоит из трех ярусов. В первом ярусе расположены инверторы (элементы *1—3*), на которых получаются требуемые формулой инверсии аргументов. Очевидно, что максимальное число инверторов не превышает числа аргументов. Во втором ярусе расположены элементы И, реализующие входящие в формулу элементарные конъюнкции. Число входов каждого элемента И равно числу аргументов реализуемой им элементарной конъюнкции, а число самих конъюнкторов — числу элементарных конъюнкций в формуле, а для случая

СДНФ — также и числу единиц в таблице истинности. В третьем ярусе схемы стоит элемент ИЛИ, объединяющий выходы всех конъюнкторов. Число его входов равно числу элементарных конъюнкций формулы. Трехъярусные схемные фрагменты НЕ-И-ИЛИ широко распространены, поэтому на функциональных схемах сложных устройств используется более компактное их условное обозначение, показанное на рис. 1.4, *б*.

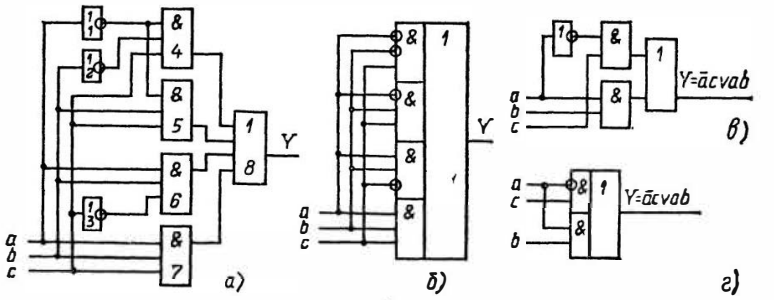


Рис. 1.4. Примеры логических схем:
а — схема, построенная непосредственно по СДНФ; *б* — сокращенное условное изображение схемы по рис. *а*; *в* — схема, полученная в результате минимизации; *г* — сокращенное обозначение схемы *в*

По логической схеме можно воспроизвести аналитическую форму реализуемой ею логической функции.

Упражнение. По схеме на рис. 1.4, *а* написать соответствующее ей логическое выражение и результат сравнить с (1.3). В случае затруднений выполните цепочку преобразований: таблица — СДНФ — схема — СДНФ — таблица над несколькими другими функциями различного числа аргументов, расставляя единицы или нули в столбце *Y* таблицы истинности случайным образом, например подбрасывая монету.

1.4. Минимизация функций

Запись функции в форме СДНФ не единственно возможная и, как правило, не самая короткая. Построенная по СДНФ логическая схема также часто оказывается не самой экономичной. Как правило, они *избыточны* и поддаются *минимизации*.

Наиболее очевидным способом минимизации СДНФ яв-

ляется выполнение преобразований, аналогичных преобразованиям обычной алгебры. Это возможно, поскольку для операций И и ИЛИ справедливы законы *ассоциативности* (сочетательный) и *дистрибутивности* (распределительный). Другими словами, аргументы можно группировать, а общие аргументы — выносить за скобки:

$$abc = a(bc) = (ab)c; \quad a \vee b \vee c = a \vee (b \vee c) = (a \vee b) \vee c; \tag{1.4}$$

$$ab \vee ac = a(b \vee c); \tag{1.5}$$

$$a \vee (bc) = (a \vee b)(a \vee c). \tag{1.6}$$

Справедливость выражений легко проверить, построив и сравнив между собой таблицы истинности для левой и правой частей равенств. Построение и сравнение таблиц — универсальный метод проверки (или доказательства) логических равенств. Две функции равны, если их таблицы одинаковы, и наоборот.

Если в (1.5) операцию конъюнкции заменить обычным умножением, а операцию дизъюнкции — сложением, то получится соотношение, хорошо известное в обычной алгебре. Это и дало повод в некоторых работах называть конъюнкцию логическим умножением, а дизъюнкцию — логическим сложением.

При минимизации целью преобразований является такая группировка членов формулы, которая позволяет после вынесения в соответствии с (1.5) общих конъюнктивных членов получить в скобках выражения вида $(a \vee 1)$ или $(a \vee \bar{a})$, которые в соответствии с (1.2) равны 1. Это сокращает число конъюнкций исходной формулы и число входящих в них аргументов. Аналогично пытаются выделить из формулы и другие соотношения вида (1.1) и (1.2), полезные для ее упрощения.

Упражнение. Минимизировать выражение (1.3).

Решение. Выражение позволяет вынести за скобки общие конъюнктивные члены, например

$$Y = \bar{a}\,\bar{b}c \vee \bar{a}bc \vee ab\bar{c} \vee abc = \bar{a}c(\bar{b} \vee b) \vee ab(\bar{c} \vee c) = \bar{a}c \vee ab. \tag{1.7}$$

Для минимизации были использованы соотношения из (1.2) и (1.1):

$$\bar{b} \vee b = 1; \quad \bar{a}c \cdot 1 = \bar{a}c; \quad \bar{c} \vee c = 1; \quad ab \cdot 1 = ab.$$

Выражение (1.7) существенно проще, чем (1.3). Выражение типа (1.7), представляющее собой дизъюнкцию нескольких элементарных конъюнкций, называют *дизъюнк-*

тивной нормальной формой или *ДНФ*. Совершенная дизъюнктивная нормальная форма есть частный случай ДНФ. Поскольку минимизация СДНФ часто удается, число элементарных конъюнкций (*длина* ДНФ) и число входящих в конъюнкции аргументов (*ранг* каждой конъюнкции) реально используемых ДНФ обычно меньше, чем у исходной СДНФ. Соответственно и схема на рис. 1.4, *в*, реализующая (1.7), экономичнее схемы, показанной на рис. 1.4, *а* и построенной непосредственно по СДНФ (1.3).

Построив для (1.7) таблицу истинности, можно убедиться, что она совпадает с табл. 1.3. При построении таблиц истинности ДНФ нужно учитывать особенности обращения с теми элементарными конъюнкциями, в которые входят не все аргументы, например конъюнкция ab не содержит c. Это означает, что, когда $a=b=1$, Y не зависит от c, т. е. если $ab=1$, то $Y=1$ как при $c=0$, так и при $c=1$. Поэтому ситуация, когда элементарная конъюнкция $ab=1$, должна быть зафиксирована сразу в двух строках таблицы: $ab\bar{c}$ и abc.

Выполняя последовательно шаги группировки типа показанных в (1.7) и сокращая ДНФ с помощью выражений (1.1) и (1.2), разработчик в конце концов может привести ДНФ к виду, когда дальнейшая группировка уже невозможна — это *тупиковая ДНФ*. Если выполнять группировку в другой последовательности, то можно получить иную форму тупиковой ДНФ этой же функции. Одна (или несколько) из множества тупиковых ДНФ данной функции имеет наименьшее по сравнению с остальными общее число букв, входящих во все ее элементарные конъюнкции. Это *минимальная* (минимальные) *ДНФ*.

Для некоторых целей (например, для реализации функции на ПЛМ простейшего типа, о которых речь будет в § 3.5) требуется получить не минимальную, а *кратчайшую ДНФ*, имеющую минимальное число элементарных конъюнкций. В общем случае эти два вида ДНФ не совпадают, о чем подробнее можно узнать в [1].

Целью минимизации логической функции в строгом понимании этого термина является нахождение минимальной ДНФ или одной из минимальных ДНФ, если их несколько. Какого-либо правила группировки, гарантированно приводящего любую ДНФ к ее минимальной форме, не существует. Приходится пробовать различные варианты и сравнивать результаты. Процедуру поиска заметно облегчают специально разработанные *методы минимизации*.

Минимизацию функций трех и даже четырех переменных можно эффективно выполнять, используя *геометрическое представление* логической функции. Все возможные комбинации значений n аргументов представляются как вершины n-мерного куба. Вершины, в которых функция равна единице, как-либо выделяются. На рис. 1.5, *а* показана гео-

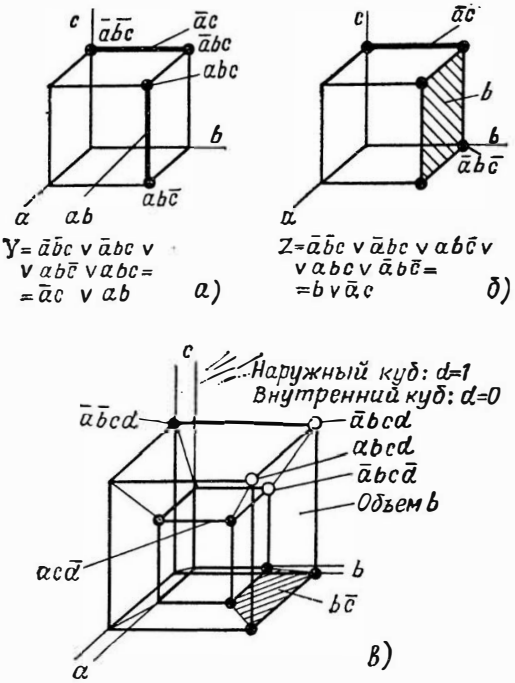


Рис. 1.5. Геометрическое представление логических функций трех аргументов *(а, б)* и четырех аргументов *(в)*

метрическая интерпретация функции, заданной табл. 1.3 и СДНФ (1.3). Единичные значения показаны жирными точками. Вершины куба соответствуют трехчленным конъюнкциям аргументов, т. е. трехчленным элементарным конъюнкциям СДНФ и ДНФ, ребра соответствуют двучленным элементарным конъюнкциям ДНФ, полученным в результате минимизации СДНФ, а грани — единичному или нулевому значению какой-либо одночленной элементарной конъюнкции минимизированной ДНФ. Из рисунка видно, что две соседние вершины, например $\bar{a}\bar{b}c$ и $\bar{a}bc$, можно *склеить*, за-

менив их ребром $\bar{a}c$, что находится в полном соответствии с процедурой вынесения за скобки общего множителя $\bar{a}c$ в (1.7). Аналогично можно склеить вершины abc и $ab\bar{c}$, заменив их ребром ab. В результате функция вместо четырех точек будет представлена двумя ребрами:

$$Y = \bar{a}c \vee ab,$$

что соответствует (1.7).

На рис. 1.5, *б* показана функция Z, СДНФ которой отличается от функции Y лишь наличием пятого члена $\bar{a}b\bar{c}$. Конфигурация точек куба становится такой, что четыре из них можно представить как грань b, а две — как ребро $\bar{a}c$ и сразу записать минимальную ДНФ: $Z = b \vee \bar{a}c$. То что вершина $\bar{a}bc$ входит и в состав ребра, и в состав грани, ошибки не порождает, поскольку это равносильно повторению одинаковых членов $\bar{a}bc$ в дизъюнкции Z, а из (1.2) следует, что $\bar{a}bc \vee \bar{a}bc = \bar{a}bc$. Таким образом, минимизация функции трех переменных с использованием геометрического представления сводится к тому, что на изометрическом чертеже куба точки единичных значений функции объединяются в группы так, чтобы сформировать сначала по возможности целые грани, затем — ребра, и лишь точки, не имеющие соседей, оставить в том виде, как они и были, — в форме трехчленных конъюнкций. Сложная логическая задача минимизации выполняется при этом почти на интуитивном уровне, за счет мощных врожденных механизмов восприятия трехмерного пространства. Как и в случае аналитических преобразований ДНФ, группировать точки, в которых функция равна единице, можно различными способами. При этом будут получаться различные формы аналитического представления этой функции, соответственно различные по конфигурации и сложности схемы, реализующие ее на элементах НЕ, И, ИЛИ.

Функции четырех переменных можно представлять в виде двух связанных кубов: одного — для нулевого значения четвертого аргумента, другого — для единичного. Кубы можно располагать один внутри другого, как это показано на рис. 1.5, *в*, где жирные точки соответствуют единичным значениям функции

$$Y = \bar{a}\bar{b}cd \vee \bar{a}b\bar{c}d \vee \bar{a}bcd \vee a\bar{b}cd \vee abc\bar{d} \vee ab\bar{c}d \vee abcd = b\bar{c} \vee ac\bar{d} \vee \bar{a}\bar{b}cd.$$

Как и в трехмерном кубе, минимизация выполняется выявлением «на глаз» замкнутых граней ($b\bar{c}$), ребер ($ac\bar{d}$), а если функция оказалась удачной, то и целых замкнутых объемов. Последнее имело бы место, если в состав СДНФ вошли бы еще три конъюнкции, обозначенные на рис. 1.5, *в* белыми кружочками справа вверху. Тогда в четырехмерном гиперкубе образовался бы замкнутый трехмерный объем b и минимальным стало бы выражение $Y' = b \vee ac\bar{d} \vee \vee \bar{a}\bar{b}cd$.

При пяти аргументах нужно рисовать четыре куба, метод становится громоздким и теряет наглядность.

При числе аргументов 3, 4 и даже 5 хорошо работает метод *минимизационных карт Карнау—Вейча* (или карт Карно) [2—4, 6], основанный на зрительном анализе разверток многомерных кубов на плоскости. Еще можно отметить метод минимизации Квайна — Мак-Класки [2—4, 6], дающий единообразную и рациональную методику выполнения перебора вариантов при поиске минимальной формы. Его используют при построении программ ЭВМ, автоматически минимизирующих функции, заданные в виде СДНФ или таблиц истинности.

1.5. О проблемах оптимизации логических выражений

Для минимизации функций и поиска наиболее экономичной схемы, реализующей заданное выражение, до сих пор не найдено эффективных алгоритмов, позволяющих решать эти задачи целенаправленно и быстро. Есть обоснованное мнение (см. [6]), что таких алгоритмов вообще не существует. Найти гарантированно минимальное выражение для произвольной функции можно лишь методом полного перебора вариантов различных способов группировки в процессе минимизации, а это реально осуществимо лишь при небольшом числе аргументов. С ростом их числа сложность минимизации и поиска экономичной схемы растет экспоненциально и очень скоро становится не под силу ни человеку, ни ЭВМ. Оценки объемов перебора при минимизации функций и поиске их кратчайших форм приведены в [1].

С точки зрения подходов к упрощению логических выражений, функции, с которыми имеет дело схемотехник, удобно разделить на три группы: функции малого числа аргументов; «объективные» функции многих аргументов; «субъективные» функции многих аргументов.

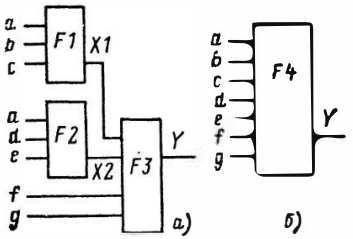
К первой группе можно отнести функции трех-четырех-пяти аргументов. Статистический анализ реальных схем цифровой аппаратуры показывает, что разработчики в подавляющем большинстве случаев

встречаются с необходимостью реализовывать именно такие функции. Благодаря малому числу аргументов таблицы таких функций короткие, вариантов группировки при минимизации не слишком много, хорошо работает наиболее наглядное геометрическое представление, и в общем, минимизация таких функций любым способом серьезных проблем не вызывает.

Ко второй группе — громоздких «объективных» функций — относятся функции более четырех-пяти аргументов, которые были построены не человеком, а отражают некоторую объективную природную зависимость. Например, это может быть один из разрядов таблицы поправок некоторого датчика, представленной двоичным кодом, причем каждый двоичный разряд выходного кода таблицы реализуется аппаратно как логическая функция n аргументов, где n — число двоичных разрядов на выходе датчика. Для этих функций характерно следующее. Во-первых, в них не заложено какой-либо простой логической закономерности, которую можно угадать и использовать для минимизации, т. е. таблица такой функции — это просто некая случайная последовательность единиц и нулей. Во-вторых, в силу значительного числа аргументов полный перебор вариантов при любом способе поиска минимальной или любой другой экономичной формы практически неосуществим. И, в-третьих, что самое существенное, из теоремы О. Б. Лупанова об оценке сложности функций (см. [6]) следует, что с ростом числа аргументов доля экономичных по оборудованию функций стремится к нулю, т. е. почти все функции оказываются неминимизируемыми. Следовательно, задача поиска минимальных форм «объективных» функций многих аргументов не только очень сложна и громоздка, но и с большой вероятностью безнадежна. Это учитывают изготовители элементов, выпуская для реализации функций многих переменных специальные микросхемы — ПЛМ и ПЗУ, о которых будет говориться в § 3.5 и 3.6. При использовании наиболее распространенных простых вариантов ПЛМ реализация функций будет самой экономичной по затратам аппаратуры, если функция приведена не к минимальной, а к кратчайшей форме. Если же при минимизации ДНФ самую короткую форму получить не удалось, то проигрыш оказывается небольшим, поскольку стоимость реализации на ПЛМ каждой элементарной конъюнкции ДНФ невелика — 1/50 стоимости корпуса. Одной из целей освоения промышленностью ПЛМ как раз и была экономия труда разработчиков схем. Что касается ПЗУ, то на них реализуется непосредственно таблица истинности, т. е. ни записи СДНФ, ни тем более ее минимизации не требуется вообще. Тем не менее в силу традиций математическая литература до сих пор имеет тенденцию завышать важность и распространенность решения задач минимизации произвольных функций большого числа аргументов.

К третьей группе функций относятся функции большого числа аргументов, составленные человеком. Особенность этих функций связана

Рис. 1.6. К вопросу о декомпозиции логических функций, представленных в виде одной или нескольких подфункций



с понятием декомпозиции. На рис. 1.6, *а* условно представлена функция $Y=F3\ (X1, X2, f, g)$, аргументами которой кроме f и g являются переменные $X1=F1(a, b, c)$ и $X2=F2(a, d, e)$, которые сами, в свою очередь, являются функциями уже только первичных аргументов a, b, c, d, e. Подставив в выражение для функции $F3$ дизъюнктивные формы $X1$ и $X2$, выраженные через аргументы a, b, c, d, e, можно получить выражение ДНФ Y непосредственно только через первичные аргументы a, b, c, d, e, f, g, что условно показано на рис. 1.6, *б*: $Y=F4(a, b, c, d, e, f, g)$. Если описанный процесс развертывать в обратную сторону, взяв за основу функцию $F4$, то можно сказать, что $F4(a, b, c, d, e, f, g)$ представима в виде структуры из более простых функций меньшего числа аргументов — $F1$, $F2$, $F3$ или что функция $F4$ подвергается *декомпозиции* вида

$$F4 = F3\,(f,\ g,\ F1\,(a,\ b,\ c),\ F2\,(a,\ d,\ e)).$$

Если декомпозиция некоторой заданной функции оказалась удачной, т. е. составляющие ее функции типа $F1$, $F2$ и т. д., на которые удалось разбить заданную функцию, достаточно просты, то аппаратурная реализация полученной структуры окажется экономичнее, чем непосредственная реализация заданной функции. Действительно, функции $F1$ и $F2$ в структурном представлении, показанном на рис. 1.6, *а*, реализуются каждая лишь по одному разу, а при представлении Y в виде единой функции $F4$ функции $F1$ и $F2$ скорее всего будут реализовываться несколько раз, поскольку они при раскрытии скобок войдут в состав нескольких элементарных конъюнкций функции $F3$.

Таким образом, декомпозицию функции можно выполнять с той же целью, что и минимизацию, чтобы получить более экономичное аппаратурное воплощение. Существуют формальные методы декомпозиции (см. [2, 6]), которые, к сожалению, как и методы минимизации, не являются целенаправленными алгоритмами с гарантированным хорошим исходом, а носят поисковый характер. При этом очевидно, что хороший вариант декомпозиции для данной функции можно найти, только если эта функция д е й с т в и т е л ь н о представляет собой структуру искомого класса и просто это неизвестно, пока данная структура еще не

найдена. Если же точки, изображающие единичные значения, разбросаны по поверхности гиперкуба функции так, что никаких структур не образуют, то любой поиск увенчаться успехом, естественно, не сможет. Сказать что-либо определенное о возможности хорошей декомпозиции любой случайной функции (например, третьего двоичного разряда таблицы поправок датчика) невозможно. Но вот что характерно, но не широко известно: *сложные функции, построенные человеком с целью описания создаваемых им сложных устройств, всегда имеют хорошо выраженную структуру.*

Причина этого явления вот в чем. Логическая функция (или схема) есть формальное отображение того соотношения между аргументами, которое на ранних этапах проектирования содержательно представлял себе разработчик. Поле внимания человека, измеряемое числом объектов или понятий, взаимодействие между которыми он может одновременно четко осознавать, очень мало́ и оценивается психологами обычно в 7 ± 2 единицы, т. е. не превышает числа возможных единичных значений средней функции трех-четырех аргументов. Именно поэтому в цифровых схемах широко распространены функции малого числа аргументов. Встречаясь с задачами, имеющими бо́льшую размерность, человек разбивает их на подзадачи, что можно проиллюстрировать тем же рис. 1.6, *а*. Функция *F3* является достаточно простой функцией верхнего яруса, а частью ее входных переменных являются функции (подзадачи) *F1* и *F2*. В общем случае это дерево может продолжаться и далее. Каждая функция осознается разработчиком отдельно, представляется своей таблицей, а затем с целью представления выходного значения *Y* как функции только первичных аргументов *a*, *b*,..., *g* все эти формулы или таблицы уже чисто формально, с помощью операций алгебры логики, сводятся в общую большую функцию или таблицу, иллюстрацией чего и является функция *F4* на рис. 1.6, *б*. Строить осмысленные функции многих аргументов как-либо иначе человек просто не может. В еще более сложных случаях различные промежуточные функции строятся параллельно несколькими разработчиками. Так получаются большие таблицы функций, описывающие поведение сложных автоматов, блоков управления и т. п. Эти функции и отнесены к третьей группе.

При реализации таких функций возникает естественная задача представления их выражений в наиболее компактной форме, требующей наименьших аппаратурных затрат. Для этого функции пытаются подвергнуть минимизации и (или) декомпозиции, и в силу достаточно простой общей структуры функции это, естественно, удается. Распространенное убеждение в том, что минимизация и декомпозиция любой функции практически всегда возможны, сопровождаемое иногда даже несколько недоверчивым отношением к упоминавшейся теореме О. Б. Лупанова, является, видимо, заключительным звеном следующей цепочки фактов: схемотехник в основном сталкивается с функциями, построен-

ными человеком; в силу способа своего построения все они имеют четкую внутреннюю структуру; в силу этой структуры функции результативно поддаются минимизации и декомпозиции; исходная причина успеха не осознается. Если на каждом этапе, когда несколько функций сводятся в одну общую, превичную информацию о структуре этой функции не игнорировать и не терять, как это повсеместно принято, а сохранять в каких-то сопровождающих документах, то выполнять минимизацию и декомпозицию построенных человеком сложных функций будет намного проще, поскольку схемотехнику уже не придется играть роль детектива.

1.6. Инверсные и недоопределенные функции

Сократить работу по минимизации иногда можно за счет работы не с самой заданной функцией, а с ее инверсией. Если число единиц в таблице истинности превышает половину числа комбинаций аргументов, то СДНФ для инверсии функции будет содержать меньше конъюнкций, чем СДНФ для прямой функции. При аппаратной реализации к выходу схемы, отрабатывающей инверсию заданной функции, нужно подключить инвертор.

Упражнение. Построить схему, реализующую функцию, заданную табл. 1.4.

Таблица 1.4

a	b	c	Y	$\bar{Y}$	a	b	c	Y	$\bar{Y}$
0	0	0	1	0	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	1	0
0	1	1	1	0	1	1	1	0	1

Решение. Совершенная дизъюнктивная нормальная форма требуемой функции

$$Y = \bar{a}\bar{b}\bar{c} \vee \bar{a}bc \vee a\bar{b}\bar{c} \vee \bar{a}bc \vee a\bar{b}\bar{c} \vee abc. \qquad (1.8)$$

Поскольку столбец Y содержит шесть единиц из восьми максимально возможных, то таблица для $\bar{Y}$ должна содержать лишь две единицы, что и отражено в табл. 1.4. Для $\bar{Y}$ СДНФ будет значительно проще:

$$\bar{Y} = a\bar{b}c \vee abc. \qquad (1.9)$$

Последнее выражение более обозримо, чем (1.8), и легко минимизируется:

$$\overline{Y} = a\bar{b}c \vee abc = ac\left(\bar{b} \vee b\right) = ac, \qquad (1.10)$$

откуда $Y = \overline{ac}$. Схема состоит из двухвходового элемента И и включенного последовательно с ним инвертора.

В выражении (1.10) нет аргумента b, т. е. от b результат не зависит. Функция, которая по первоначальному предположению зависела от трех переменных, фактически оказалась функцией двух, т. е. *$Y(a, b, c)$ существенно зависит* только от a и c, а b является *фиктивным аргументом*. Если построить куб, то легко убедиться, что функция $\overline{Y} = ac$ является его ребром, параллельным оси b. Независимость функции от b на кубе становится в буквальном смысле очевидной.

Фиктивные аргументы на практике встречаются: при первичном рассмотрении достаточно сложной задачи может создаться впечатление, что результат зависит от n переменных и для всех комбинаций этих переменных, как обычно, составляется таблица. Но взаимосвязи между аргументами могут оказаться такими, что от некоторых из них результат будет фактически независим. Грамотно выполненные преобразования выявят этот факт.

Недоопределенной называют функцию, значения которой при некоторых комбинациях аргументов не определены или, как говорят, безразличны. Не определенные значения функции отмечают прочерками, как показано в столбце Y табл. 1.5. Недоопределенные функции появляются при формальном описании таких объектов, в которых при нормальной работе возникают и используются не все возможные комбинации аргументов. Так, если в электронном календаре вос-

Таблица 1.5

a	b	c	Y	$Y0$	$Y1$	$Y2$	$Y3$
0	0	0	—	0	0	1	1
0	0	1	1	1	1	1	1
0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	—	0	1	0	1

кресенье кодируется кодом 000, понедельник — 001 и т. д. до субботы 110, то при правильной работе календаря на вход блока, зажигающего световое табло СЕГОДНЯ ХИМЧИСТКА НЕ РАБОТАЕТ, никогда не будет поступать код 111. Поэтому значение функции при такой комбинации аргументов для того, кто задает таблицу, безразлично.

Упражнение. Построить схему, реализующую недоопределенную функцию *Y*, заданную табл. 1.5.

Подсказка. Значения *Y* при комбинациях аргументов 000 и 111 не определены. Недоопределенную функцию реализовать невозможно: не хватает данных для построения СДНФ. Функцию нужно *доопределить*.

Решение. Поскольку для заказчика схемы значения прочерков безразличны, то при двух прочерках возможны четыре способа доопределения:——=$\{00, 01, 10, 11\}$. Получившиеся при этом варианты функции представлены в табл. 1.5 в столбцах *Y0*—*Y3*. Все варианты одинаково хорошо выполняют задание, но, видимо, не все окажутся эквивалентными по аппаратурным затратам. Безразличное для заказчика скорее всего не будет безразличным для схемотехника.

На рис. 1.7, *а* показана геометрическая интерпретация заданной недоопределенной функции, на рис. 1.7, *б*—доопределение *Y0* (доопределение нулями, что чаще всего делают начинающие разработчики), а на рис. 1.7, *в* — доопределение *Y1*, наилучшее для заданной функции *Y*. На рис. 1.7, *г*, *д* показаны схемы, реализующие соответственно нулевое доопределение *Y0* и оптимальное *Y1*. Интересно сопоставить разницу в сложности схем и различие столбцов *Y0* и *Y1* (табл. 1.5): функции отличаются всего одной единицей.

С ростом числа прочерков число вариантов доопределения экспоненциально растет и поиск наилучшего решения становится практически неосуществимым. В таких случаях функцию доопределяют нулями, чтобы по крайней мере уменьшить число конъюнкций СДНФ.

Если неиспользуемых комбинаций аргументов много, то их можно применить для контроля работы аппаратуры. Для этого кроме заданной функции *Y* строится еще одна функция от тех же аргументов — функция обнаружения ошибки *ER*. В таблице *ER* единицы стоят там, где в основной функции *Y* стоят прочерки. В остальных позициях функции *ER* стоят нули. При ошибочном появлении на входе схемы какой-либо из неиспользуемых комбинаций на выходе *ER* вырабатывается сигнал 1. Такой тип контроля называется *семантическим*, т. е. смысловым. Он выявляет самые грубые ошибки — появление бессмысленных сигналов.

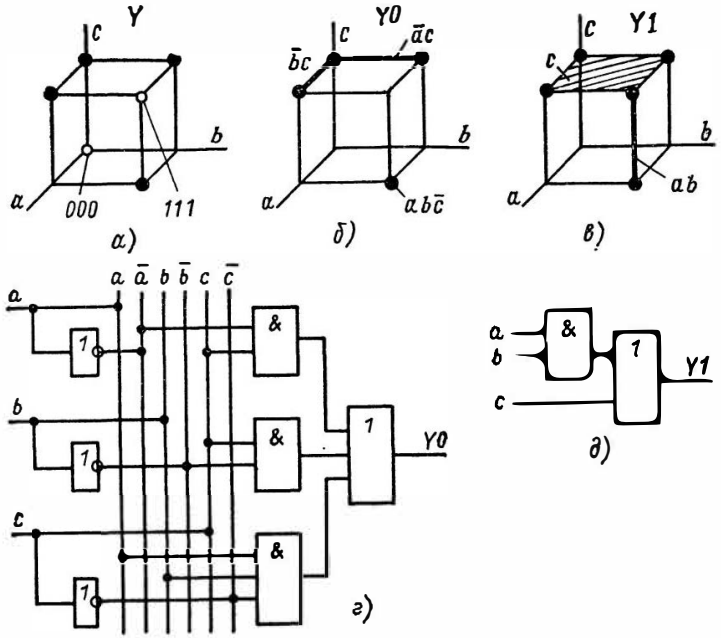


Рис. 1.7. Реализации недоопределенной функции

Если число прочерков существенно превосходит число определенных значений, то функцию называют *слабо определенной*. О методах работы с такими функциями см. [2 и 6].

1.7. Функции И-НЕ и ИЛИ-НЕ

Функция **И-НЕ** — это функция двух и более аргументов (другие названия: штрих Шеффера, функция Шеффера, *NAND*). Значения функции представлены в табл. 1.2. Легко видеть, что это инверсия функции И, т. е. отрицание конъюнкции. Любой 0 на входе дает 1 на выходе, все единицы на входе дают 0 на выходе. Специального символа для обозначения И-НЕ не применяют, используя комбинацию символов И и НЕ: $Y=\overline{ab}$. Обозначение функции на схемах показано на рис. 1.8, *а*. Используя только элементы И-НЕ, можно получить функции НЕ, И, ИЛИ, как показано на рис. 1.8, *б—г*. Правомерность такого представления можно доказать с помощью таблиц истинности. Освоить табличный

метод доказательства формул и проверки схем поможет упражнение.

Упражнение. Доказать, что схема по рис. 1.8, *г* выполняет функцию ИЛИ.

Решение. Типичная ошибка начинающего разработчика: при анализе схем он пытается все промежуточные ре-

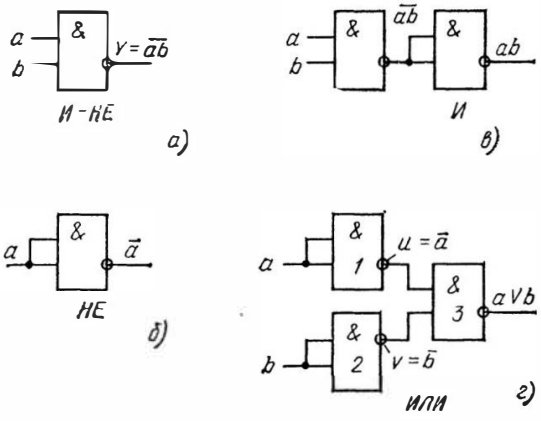


Рис. 1.8. Возможности элемента И-НЕ

зультаты держать в уме. Как уже отмечалось, емкость «сверхоперативной» памяти человека мала, поэтому, пока не появились навыки работы со схемами, единственным и очень верным средством является ведение в процессе работы хорошо организованных подробных записей промежуточных результатов. Пример удобной формы записи для вычисления значений громоздких функций — табл. 1.6.

Таблица 1.6

a b	Элемент 1 $u=\bar{a}$	Элемент 2 $v=\bar{b}$	Элемент 3		
			uv	$\overline{uv}=\overline{\bar{a}\,\bar{b}}$	$a \vee b$
1	2	3	4	5	6
0 0	1	1	1	0	0
0 1	1	0	0	1	1
1 0	0	1	0	1	1
1 1	0	0	0	1	1

В таблице для каждого элемента схемы (рис. 1.8, *г*) выделен столбец, в котором записываются значения выхода элемента для всех комбинаций входных сигналов. Для элементов, выполняющих сложные или просто непривычные функции, можно выделить два и больше столбцов, как это сделано для элемента 3. Полученный результат (столбец 5) сравнивается с ожидаемым результатом, в данном случае с таблицей функции ИЛИ (см. табл. 1.2), для удобства переписанной в столбец 6.

Способность функции И-НЕ выражать только через самое себя все функции булева базиса доказывает, что эта функция обладает логической *полнотой*. С помощью одной лишь функции И-НЕ можно построить любую сколь угодно сложную логическую функцию. Вторым ценным свойством функции И-НЕ оказалось то, что именно ее удалось эффективно реализовать средствами самой массовой интегральной технологии — ТТЛ. Поэтому уже четверть века функция И-НЕ наиболее распространена в цифровой автоматике. Выпускается И-НЕ в виде отдельных микросхем, на ее основе создано множество схем средней интеграции, И-НЕ широко используется в схемах с большой степенью интеграции (БИС). Аналога в русском языке эта функция не имеет, поэтому мышление в базисе И-НЕ непривычно и требует тренировки.

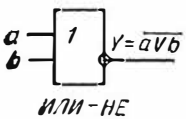


Рис. 1.9. Элемент ИЛИ-НЕ

Функция **ИЛИ-НЕ** — это функция двух и более аргументов (другие названия: функция Вебба, стрелка Пирса, *NOR*). Значения функции представлены в табл. 1.2. Данная функция является инверсией функции ИЛИ, в формулах обозначается как $Y=\overline{a \vee b}$, на схемах — как показано на рис. 1.9. Русский язык немного умеет оперировать с этой функцией, называя ее «ни — ни». Так, фраза *Y есть ни рыба, ни мясо* утверждает, что названный объект не принадлежит ни к одному из этих классов. Функция ИЛИ-НЕ, как и функция И-НЕ, обладает полнотой и тоже удобна для интегрального исполнения, особенно по технологии КМДП и ЭСЛ. Функция ИЛИ-НЕ — вторая по распространенности после И-НЕ функция в цифровой технике.

1.8. Сумма по модулю 2

Название в заголовке или его сокращенная форма М2 — это строгое название рассматриваемой функции от любого

числа аргументов. В случае двух аргументов эту функцию называют также функция *неравнозначности, исключающее ИЛИ, exclusive OR, XOR*. Обозначение в формулах: $Y=a \oplus b$. Таблица истинности функции представлена в табл. 1.2. На рис. 1.10, *а* показано обозначение этой функции на схемах, а на рис. 1.10, *б—г* — выражение ее через другие

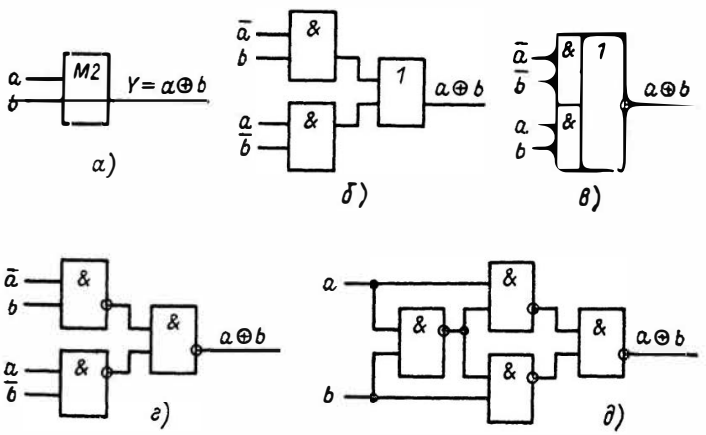


Рис. 1.10. Функция суммы по модулю 2:
а — обозначение на схемах; *б, в, г, д* — реализации функции в распространенных базисах: *б* — $Y=a \oplus b=\bar{a}b \vee a\bar{b}$; *в* — $Y=\overline{\bar{a}b} \vee \overline{ab}$; *г* — $Y=\overline{\overline{\bar{a}b} \cdot \overline{ab}}$; *д* — $Y=\overline{a \cdot \overline{ab}} \cdot \overline{b \cdot \overline{ab}}$ (эта схема не требует инверторов на входе)

функции. Название функции связано с тем, что $a \oplus b$ есть арифметическая сумма двоичных чисел a и b в пределах одного двоичного разряда: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=10$; в последнем случае возникает единица переноса в соседний старший разряд, а в разряде самих слагаемых получается нуль. Отсюда широкое применение этой функции при построении различных счетных и суммирующих устройств.

Упражнение. Доказать правильность схем по рис. 1.10, *б—г*, последовательно заполняя таблицы по примеру табл. 1.6.

Функция М2 обладает интересным свойством, которое полезно запомнить: *при инвертировании одного из аргументов вся функция инвертируется*, т. е.

$$\bar{a} \oplus b = a \oplus \bar{b} = \overline{a \oplus b}.$$

Инверсия суммы по модулю 2 для двух аргументов имеет и собственный смысл: это функция *равнозначности, или тождества* $a \equiv b$ (см. табл. 1.2). Она равна единице, если $a = b$. Следовательно, для построения схемы сравнения одноразрядных чисел можно использовать любую схему по рис. 1.10, проинвертировав один из аргументов или результат. Полезны следующие соотношения:

$$a \oplus 0 = a; \quad a \oplus 1 = \bar{a}; \quad a \oplus a = 0; \quad a \oplus \bar{a} = 1. \quad (1.11)$$

Первые два равенства позволяют превратить элемент М2 в *управляемый инвертор*. Если использовать один из входов М2 как управляющий и подавать на него 0 или 1, то информация, поступающая по второму входу, будет или пропускаться без изменения, или инвертироваться.

Функция М2 совместно с И и константой 1 образует так называемый *базис Жегалкина*, обладающий полнотой, т. е. позволяющий с помощью *полинома Жегалкина* представить любую логическую функцию [1, 2]. Однако базис Жегалкина как таковой в большинстве схем не используется, поскольку и мыслить с помощью функции М2 непривычно, и удачной микроэлектронной реализации она пока не имеет.

Исключение составляют схемы контроля и исправления ошибок, где функции М2 принадлежит ведущая роль благодаря ее специфическому свойству, которое иллюстрирует рис. 1.11. На нем представлена геометрическая интерпретация функции М2. Как видно из рисунка, при любом обходе куба по ребрам невозможно попасть из одного единичного значения этой функции непосредственно в другое единичное значение: они всегда разделены хотя бы одним нулевым. Поэтому, если из-за неисправности в схеме какой-либо аргумент функции М2 изменит свое значение, тут же обязательно проинвертируется значение и самой функции, а это легко обнаружить, поскольку при любом числе аргументов нужно контролировать всего лишь один выход М2.

Отсутствие рядом расположенных единиц объясняет также свойство функции М2 успешно противостоять всем попыткам минимизации ее в базисе НЕ, И, ИЛИ: ее единичные значения не образуют ни об-

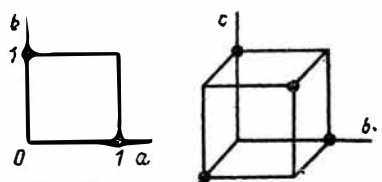


Рис. 1.11. Функция М2 для двух (*а*) и трех (*б*) переменных

щих ребер, ни граней, ни гиперграней. Отсюда практический вывод: если некоторая функция имеет в таблице много единиц и тем не менее плохо минимизируется, то она, возможно, близка по своей структуре к сумме по модулю 2 и будет иметь короткую запись, если используемый базис дополнить функцией *М2*. Функция *М2* действительно имеется в составе многих серий.

Неправомерность безусловного отождествления функций *М2*, *неравнозначности* и *исключающего ИЛИ* иллюстрирует предлагаемое упражнение.

Упражнение. Построить таблицы истинности для следующих функций трех аргументов: *1* — сумма по модулю 2; *2* — неравенство всех аргументов друг другу, т. е. несовпадение их; *3* — исключающее ИЛИ, альтернатива, т. е. один и только один из всех; *4* — дизъюнкция.

Решение приведено в табл. 1.7. Таблицы истинности всех четырех функций различны, что говорит о том, что при трех (и более — то-

Таблица 1.7

Входы			М2	Неравенство (несовпадение)	Исключающее ИЛИ (один и только один)	Дизъюнкция (ИЛИ)
a	*b*	*c*	$a \oplus b \oplus c$			
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	0	0	1

же) аргументах это — четыре совершенно различные функции. В то же время, будучи построенными для двух аргументов, таблицы трех первых функций полностью совпадают, чем и объясняются три почти одинаково распространенных названия функции М2: сумма по модулю 2, неравнозначность, исключающее ИЛИ. Однако эти же названия часто используют для обозначения и многовходовых функций М2, что, как следует из табл. 1.7, неверно.

Неоднозначность технических и даже логических терминов ведет начало от неоднозначности разговорного языка. Так, грамматика нашего языка (кстати, английского тоже) не различает функций исключающего ИЛИ и просто ИЛИ, т. е. дизъюнкции. Например, во фразе *Ревет ли зверь в лесу глухом, трубит ли рог, гремит ли гром,...* союз «ли» (краткая форма «или») имеет значение дизъюнкции, а в приветствии *Кошелек или жизнь*, как и в напутствии *Со щитом или на щите* тот же союз «или» выражает уже исключающее ИЛИ.

1.9. Формулы де-Моргана

Для работы в инвертирующих логических базисах, таких, как И-НЕ, ИЛИ-НЕ, очень удобными оказываются *формулы де-Моргана*

$$\bar{a}\bar{b} = \bar{a} \vee \bar{b}; \quad \overline{a \vee b} = \bar{a} \cdot \bar{b}. \tag{1.12}$$

Эти соотношения хорошо интерпретируются фразами естественного языка:

в комнате *ТЕПЛО*, если батареи *ВКЛЮЧЕНЫ* **И** окно *ЗАКРЫТО*,

$$T = B \cdot З;$$

в комнате **НЕ** *ТЕПЛО*, если батареи **НЕ** *ВКЛЮЧЕНЫ* **ИЛИ** окно **НЕ** *ЗАКРЫТО*,

$$\bar{T} = \bar{B} \vee З.$$

Обратите внимание, как И переходит в ИЛИ, когда отрицаются аргументы и сама функция, а смысл выражения при этом тем не менее не изменяется.

Формулы де-Моргана применимы при любом числе аргументов. Они иллюстрируют глубокую взаимную симметрию операций И и ИЛИ: если операция И избирательно реагирует на совпадение прямых сигналов, то операция ИЛИ так же избирательно реагирует на совпадение их инверсий. Элемент ИЛИ прозрачен для любого сигнала, элемент И — для любой инверсии. Пользуясь формулами де-Моргана, можно легко переводить логические схемы из базиса НЕ, И, ИЛИ, в котором человеку привычнее всего мыслить и составлять исходные логические выражения, в инвертирующие базисы, которые эффективнее всего реализуются интегральной технологией.

Упражнение. Построить на элементах И-НЕ схему, реализующую выражение

$$Y = a \vee bc \vee \overline{def \vee gh}.$$

Решение. Поскольку с помощью элементов И-НЕ неудобно реализовывать операцию ИЛИ, то в приведенном выражении нужно избавиться от операций дизъюнкции, используя формулу де-Моргана. Начать можно прямо с первых двух дизъюнкций:

$$Y = a \vee bc \vee \overline{def \vee gh} = \overline{\bar{a} \cdot \overline{bc} \cdot (def \vee gh)}.$$

Теперь часть полученного выражения уже легко реализуется на элементах И-НЕ (рис. 1.12, *а*), но выражение $def \vee gh$

пока еще имеет вид, неудобный для реализации на этих элементах. Его также можно преобразовать по формуле де-Моргана

$$Z = def \vee gh = \overline{\overline{def} \cdot \overline{gh}}.$$

Окончательная схема показана на рис. 1.12,*б*.

Можно сформулировать общую рекомендацию, полезную при переводе логической формулы из булева базиса

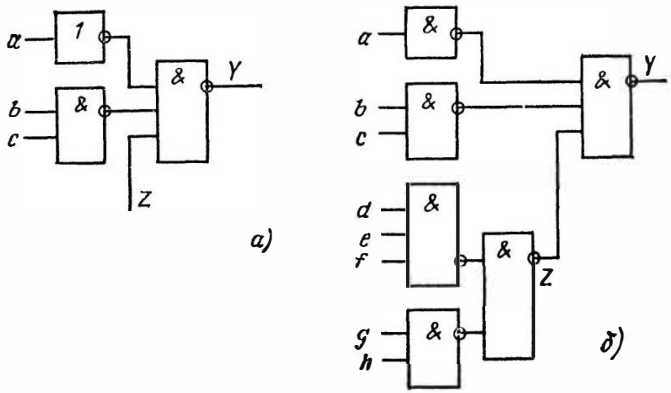


Рис. 1.12. Этапы построения функциональной схемы на И-НЕ

в базис инверсных функций И-НЕ, ИЛИ-НЕ: если все выражение или какой-либо его фрагмент не оканчивается операцией инвертирования, то к нему нужно применить формулы де-Моргана. Тогда в схеме не будет лишних инверторов.

Если в произвольной схеме, начав с элементов ее первого яруса (т. е. с элементов, связанных лишь со входом) и двигаясь далее от яруса к ярусу, к каждому элементу применить формулу де-Моргана, то можно доказать справедливость следующего важного правила:

Если в произвольной схеме, построенной на элементах НЕ, И, ИЛИ,

проинвертировать все входные сигналы,
все элементы И заменить на ИЛИ,
все элементы ИЛИ заменить на И,
все инверторы оставить без изменения,
то выходной сигнал схемы проинвертируется.

Примечательно здесь то, что столь кардинальная перестройка всей схемы не изменяет реализуемую ею функцию

каким-либо произвольным образом, а оставляет ее в общем той же самой, лишь инвертируя выход.

На рис. 1.13 показаны две логические схемы, причем схема *б* получена из схемы *а* с помощью описанной процедуры. Если в выражении для F^* к каждой из трех логичес-

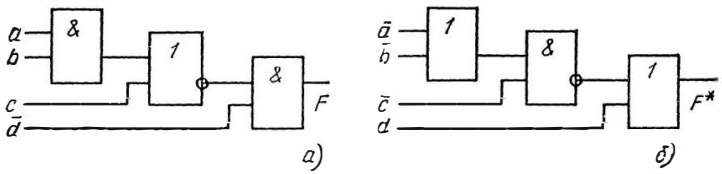


Рис. 1.13. Две взаимно двойственные схемы

ких операций ИЛИ, И, ИЛИ последовательно применять формулы де-Моргана, то получится следующая цепочка:

$$F^* = \overline{(\bar{a} \vee \bar{b}) \cdot \bar{c}} \vee d = \overline{\overline{ab} \cdot \bar{c}} \vee d = (ab \vee c) \vee d =$$

$$= \overline{\overline{ab \vee c} \cdot \bar{d}} = \bar{F}.$$

Выход оказался проинвертированным.

Следовательно, каждая логическая схема или любой ее фрагмент (узел) могут быть представлены в двух вариантах: *основном и двойственном* (дуальном). Последний по конфигурации схемы ничем не отличается от основного, только в нем И заменены на ИЛИ и наоборот, а все входы и выходы проинвертированы. Оба варианта абсолютно равноправны, и который из них считать «основным», определяется лишь вкусами разработчика.

Из сказанного следуют два вывода.

Во-первых, нет необходимости изучать построение одинаковых функциональных узлов отдельно на элементах И-НЕ и отдельно на элементах ИЛИ-НЕ: схемы в обоих базисах будут иметь абсолютно одинаковую конфигурацию.

Во-вторых, соотношение числа элементов И и ИЛИ в прямом и двойственном вариантах схемы различно. В то же время во многих сериях элементов входы И и ИЛИ не эквивалентны по затратам оборудования. Более того, в ряде серий существуют двухступенчатые элементы И-ИЛИ-НЕ, но отсутствуют двойственные им элементы ИЛИ-И-НЕ. Число требуемых инверторов на входах и выходах у прямого и двойственного вариантов также различно. Поэтому почти всегда прямой и двойственный варианты функционального узла отличаются и по объ-

ему затраченного оборудования, и по числу последовательно включенных элементов, т. е по значению задержки. Разработчик должен оценить оба варианта и выбрать более подходящий.

Формула де-Моргана позволяет выявить и в дальнейшем применять как стандартные четыре способа реализации ДНФ на элементах с инвертирующим выходом. Первый способ (рис. 1.14, *а*) основан на исполь-

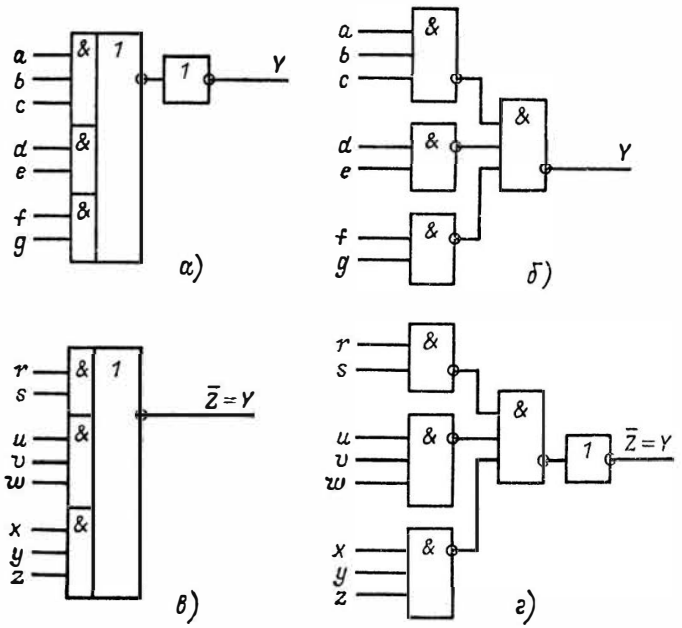


Рис. 1.14. Четыре способа реализации дизъюнктивных нормальных форм

зовании двухступенчатых элементов И-ИЛИ-НЕ, имеющихся во многих сериях элементов:

$$Y = abc \vee de \vee fg = \overline{\overline{abc} \vee \overline{de} \vee \overline{fg}}.$$

Второй способ базируется только на элементы И-НЕ, для чего исходная ДНФ преобразуется по формулам де-Моргана:

$$Y = abc \vee de \vee fg = \overline{\overline{abc} \vee \overline{de} \vee \overline{fg}} = \overline{\overline{abc} \cdot \overline{de} \cdot \overline{fg}}. \qquad (1.13)$$

Результирующая схема показана на рис. 1.14, *б*. Существенно, что, несмотря на использование инверсного базиса, в ней нет специальных инверторов. При грамотном проектировании в базисе И-НЕ дизъюнктивные формы реализуются не менее экономично, чем в базисе НЕ, И,

ИЛИ. Полезно запомнить, что *ДНФ, т. е. функцию двухступенчатой схемы И, ИЛИ, выполняет такая же по конфигурации двухступенчатая схема И-НЕ, И-НЕ.*

Третий и четвертый способы применяют тогда, когда из-за слишком большого числа единиц в таблице истинности СДНФ выгоднее записывать не для единичных, а для нулевых значений функции, т. е. вместо прямой функции Y синтезировать ее инверсию $\overline{Y}=Z=rs\vee uvw\vee xyz$. По третьему способу функция Z реализуется на ступенях И и ИЛИ элемента И-ИЛИ-НЕ. В результате прямая функция $Y=\overline{Z}=\overline{rs\vee uvw\vee xyz}$ оказывается реализованной всего на одном элементе И-ИЛИ-НЕ, в отличие от первого способа — без добавочного инвертора на выходе (рис. 1.14, *в*). Задержка отработки функции при этом получается минимально возможной, что полезно запомнить. Четвертый способ (рис. 1.14, *г*) использует при реализации $Z=\overline{Y}$ базис И-НЕ, и в этом случае для получения Y без инверсии требуется дополнительный по сравнению с рис. 1.14, *б* инвертор:

$$Y=\overline{\overline{Z}}=\overline{\overline{rs\cdot uvw\cdot xyz}}=\overline{rs\vee uvw\vee xyz}\,.$$

Таким образом, при использовании современных инвертирующих элементов существует четыре стандартных способа вложения в аппаратуру изначально заданных дизъюнктивных нормальных форм, и в каждом конкретном случае наилучшим может оказаться любой из них.

1.10. Положительная и отрицательная логика

Формулы де-Моргана объясняют ряд вопросов, связанных с соглашениями положительной и отрицательной логики. При кодировании логических переменных уровнями электрических сигналов обычно логическую единицу кодируют более положительным, ВЫСОКИМ уровнем напряжения, а логический нуль — более отрицательным, НИЗКИМ уровнем. Такой способ кодирования называют соглашением *положительной логики.* В принципе уровни сигналов можно трактовать и наоборот, т. е. использовать соглашение *отрицательной логики.* Физика работы логического элемента от этого, естественно, не изменится, но интерпретировать его поведение придется уже иначе. Так, элемент, формирующий на выходе ВЫСОКИЙ уровень при совпадении ВЫСОКИХ уровней на всех его входах и формирующий НИЗКИЙ уровень, если хотя бы на один его вход подан НИЗКИЙ, в положительной логике интерпретируется как элемент И (на выходе единица только при всех единицах на входе), а в отрицательной логике — как элемент ИЛИ (на выходе появ-

ляется нуль только при совпадении всех нулей на входе). Соответственно элементы И-НЕ превращаются в ИЛИ-НЕ, элементы И-ИЛИ-НЕ — в ИЛИ-И-НЕ, в общем, изменение трактовки выполняемой элементом функции происходит в полном соответствии с правилами де-Моргана.

Положительная логика для инженера удобнее: в соответствии с общепринятым направлением осей координат ему привычнее называть единицей более высокий уровень напряжения на экране осциллографа, а нулем — более низкий, чем наоборот. Поэтому положительная логика распространена шире. Точно так же психологически естественнее отождествлять с единицей сигналы активного, утверждающего смысла: РАЗРЕШЕНИЕ, ЗАПИСЬ, а с нулем — сигналы пассивного смысла: отсутствие разрешения и т. п.

К сожалению, иногда соглашение положительной логики вступает в конфликт с требованиями электротехники. Так, потребление по питанию ТТЛ-элементов, обслуживающих числовую магистраль, часто меньше при высоком уровне сигнала в магистрали и больше при низком. Поэтому если магистраль значительную часть времени находится в режиме ожидания, то именно неактивный, не утверждающий, логически нулевой сигнал рационально отождествить с высоким уровнем напряжения в магистрали, т. е. принять соглашение отрицательной логики. Обычно так и поступают.

Еще пример. Одновременное включение по сигналу разрешения или синхронизации большого числа транзисторов ТТЛ-микросхемы средней или большой интеграции порождает в общем выводе микросхемы мощный скачок тока с крутым фронтом, который при положительном напряжении питания вызывает импульсный подъем потенциала подложки кристалла по отношению к потенциалу общего провода печатной платы амплитудой до 0,1—0,2 В. Если этот процесс вызывается высоким уровнем управляющего сигнала, то подъем потенциала подложки приводит к уменьшению фактической амплитуды разрешающего сигнала на входах элементов микросхемы, что может вызвать сбой в работе. Если же разрешающий уровень низкий, то при подъеме потенциала подложки амплитуда сигнала, фактически действующая на элементы микросхемы, даже увеличивается. Поэтому в микросхемах средней и большой интеграции входы разрешения, синхронизации и т. п., несмотря на то что по ним поступают сигналы утверждающего, единичного смысла, часто управляются не высоким, а низким уровнем, т. е. работают по соглашению отрицательной логики.

Если подобных сигналов немного, то особых проблем не возникает. На схемах выводы таких сигналов помечают кружком инверсии, кроме того, символом инверсии помечают и названия сигналов, т. е. вместо, например, обозначения ЗАПИСЬ используют обозначение $\overline{\text{ЗАПИСЬ}}$.

Однако если в схеме сильно перемешаны соглашения и положительной, и отрицательной логики, то обозначение сигналов символами 1 и 0 начинает вызывать серьезные затруднения. В таблицах и текстах описаний применительно к каждому сигналу требуется уже специально оговаривать — как понимать значение символа 1 или 0: то ли как смысловое значение сигнала 1 — есть разрешение, 0 — нет разрешения, то ли как уровень напряжения сигнала 1 — ВЫСОКИЙ, 0 — НИЗКИЙ.

Чтобы устранить неоднозначность, нужно отделить смысловое описание работы схемы от формальных соотношений между электрическими уровнями входов и выходов. Поэтому при опасности разночтения в таблицах (и в тексте) при формальном описании функционирования узла уровни сигналов обозначают не единицей и нулем, а символами, независимыми от смыслового содержания сигналов: *L* (от *low*—низкий) и *H* (от *high* — высокий) (см. [5]). Иногда используют русские буквы: Н — НИЗКИЙ, В — ВЫСОКИЙ. Словесное наименование сигнала при этом всегда совпадает с его смысловым содержанием: РАЗРЕШЕНИЕ, ЗАПИСЬ независимо от уровня, каким этот сигнал подается. Обозначений типа $\overline{\text{ЗАПИСЬ}}$ применять уже не требуется и не следует. При необходимости в тексте дополнительно можно отметить, что данный сигнал имеет *активный низкий уровень (active low)*.

Если же основное назначение участка текста — разъяснить смысл работы схемы, то, наоборот, стараются избегать указания электрических уровней сигналов, и их текущие значения называют не высокий и низкий, и не единичный и нулевой, а активный и неактивный, подчеркивая тем самым его смысловое, концептуальное значение безотносительно к электрическому уровню. Применение такой системы обозначений уже не вызывает ошибок неоднозначности при совместном использовании в одном блоке соглашений и положительной, и отрицательной логики.

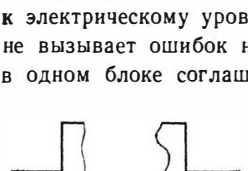


Рис. 1.15. Указатели полярности выводов

На чертежах схем при использовании смешанной логики вместо символов инверсии — кружков — применяют *указатели полярности выводов* (рис. 1.15). Указателями снабжаются те входы и выходы, для которых состоянию логическая единица, а значит, и утверждающему смыслу сигнала соответствует менее положительный уровень, т. е. те входы и выходы, для которых действует соглашение отрицательной логики (см. [5]).

В качестве примера использования символов *L* и *H* в табл. 1.8 дано описание функционирования элемента, реализующего функцию $Y=\overline{abc}$. Попутно иллюстрируется распространенный прием сокращения длины таблицы за счет

Таблица 1.8

Входы a	b	c	Выход Y	Входы a	b	c	Выход Y
L	×	×	H	×	×	L	H
×	L	×	H	H	H	H	L

введения символа × — *не зависит от значения аргумента.* Каждый крест заменяет сразу пару строк полной таблицы со значениями данного аргумента 0 и 1. Строка с двумя крестами заменяет 4 строки, с тремя — 8 и т. д. В результате табл. 1.8 оказалась вдвое короче своей полной формы. В последующих главах появятся примеры сокращения длины таблиц в десятки и даже сотни раз. Воспринимается такая сокращенная таблица даже лучше полной: она и меньше, и в ней четко выделяются те значения аргументов и их комбинаций, которые действительно определяют уровни выхода.

В таблицах справочников иногда вместо слов *не зависит от значения аргумента* пишут *значение аргумента безразлично.* Следует отличать безразличие в данном смысле от безразличия при рассмотрении в § 1.6 недоопределенных функций. Здесь это слово имеет смысл *обеспечить при всех значениях аргумента,* а применительно к недоопределенным функциям — *обеспечить при каком-то одном доопределении.* Это еще один пример неоднозначности используемой терминологии, который не должен вызывать сбоев у разработчика.

1.11. Этапы построения логической схемы

Если опыт построения *(синтеза)* логических схем у разработчика невелик, то можно рекомендовать ему следующую последовательность действий.

Этап 1. Составление таблицы истинности. Наиболее сложный, но очень часто встречающийся на практике способ задания схемы — это объяснение ее работы на понятийном уровне в виде набора фраз обычного языка (например, русского). Сложность этапа связана с тем, что задание описывается *неформальными* терминами, допускающими неоднозначную его трактовку. Основная цель этапа—*формализация* задания, в процессе которой нужно продумать значение функции для каждой комбинации значений аргументов, при необходимости поставить заказчику уточняющие

вопросы. Результат этапа — таблица истинности. Это уже задание, неоднозначное толкование которого невозможно. Наиболее трудно обнаруживаемые ошибки возникают именно на этапе формализации. Только если таблица из-за значительного числа переменных оказывается слишком громоздкой или если функция проста, привычна и смысл ее абсолютно ясен, можно начинать прямо с написания аналитической формулы.

Этап 2. Если функция определена не на всех наборах аргументов, то нужно ликвидировать неоднозначность таблицы. При малом числе неопределенных значений лучше рассмотреть несколько вариантов. Если же число или безразличных значений, или самих аргументов велико, то, возможно, придется доопределять функцию или всеми нулями, или всеми единицами — так, чтобы в результате уменьшить число членов СДНФ прямой функции или ее инверсии.

Этап 3. По полностью определенной таблице составить СДНФ. Если рассматривается несколько вариантов доопределения или если есть надежда, что инверсия функции будет реализовываться лучше, то в дальнейшей работе будут участвовать несколько вариантов СДНФ.

Этап 4. Минимизировать СДНФ любыми доступными методами. На этом этапе иногда требуется решимость, чтобы прекратить поиск лучшего варианта (которого, возможно, и не существует).

Этап 5. Реализовать получившиеся дизъюнктивные формы на логическом базисе заданной серии элементов. Попробовать варианты реализации на И-ИЛИ-НЕ и на И-НЕ, И-НЕ.

Этап 6. Оценить двойственный вариант логической схемы с учетом изменения числа входных и выходных инверторов.

Этап 7. Попытаться найти такую декомпозицию функции, чтобы каждый фрагмент полученного разложения зависел от меньшего числа аргументов, чем исходная функция. Попытаться выполнить это различными способами.

Этап 8. Выбрать из полученных на этапах 5, 6, 7 вариантов наиболее подходящий с точки зрения поставленной цели. Об оценке качества логических схем будет говориться в § 2.3.

Обычно по мере накопления опыта перечисленные этапы начинают взаимно проникать друг в друга, некоторые этапы опускаются вовсе, все больше оценок начинает выполняться очень быстро, почти на интуитивном уровне, и состав-

ление логических схем превращается в интересную, даже азартную игру.

Нестандартные подходы к синтезу цифровых схем изложены в [7].

ГЛАВА 2

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ В БАЗИСАХ МИКРОСХЕМ

2.1. Серии логических элементов

Серией микросхем называют группу микросхем, выполненных по одинаковой или близкой технологии, имеющих сходные технические характеристики и предназначенных для совместной работы в составе цифровой аппаратуры.

Условное обозначение логической микросхемы состоит из следующих элементов: 1) буквы, в большой степени характеризующей стойкость микросхемы к воздействию окружающей среды и связанный с этим тип корпуса. Отсутствие буквы рассматривается как своего рода «нулевая буква»; 2) трех или четырех цифр, обозначающих номер серии. При этом предыдущую букву часто считают первым элементом наименования серии; 3) двух букв, характеризующих выполняемую функцию (функциональная группа); 4) одной или двух цифр, обозначающих тип микросхемы внутри функциональной группы, например число входов элемента И, тип триггера и т. п.; 5) буквы, характеризующей возможные вариации значений некоторых параметров, например выходной мощности. Этой буквы чаще всего не бывает.

Пример: К155ЛА2 — микросхема серии К155, выполняющая функцию И-НЕ, второго типа (в серии К155 этот тип имеет 8 входов).

Микросхемы заключены в стандартные корпуса, в основном с двумя типами выводов:

1) *перпендикулярными плоскости корпуса*, с шагом 2,5 мм, которые вставляются в отверстия монтажной платы и распаиваются на стороне платы, противоположной корпусу. Такие корпуса иногда называют корпусами *типа ДИП (DIP — dual in-line package* — корпус с двумя рядами выводов). В корпуса ДИП чаще всего заключаются *микросхемы широкого применения,* имеющие перед номером се-

рии буквы К или КМ. Обычный температурный диапазон микросхем с буквой К $-10 \div +70$ °С, а с буквами КМ—$45 \div +85$ °С. В микросхемах широкого применения более новых выпусков пластмассовый корпус часто обозначается буквами КР;

2) *плоскими (планарными)*, которые накладываются на плату и распаиваются на той же ее стороне, где находится и сам корпус; шаг выводов 1,25 мм. В таких корпусах обычно выпускаются серии без буквы перед номером. Типичный диапазон их рабочих температур $-60 \div +125$ °С. Подробнее об обозначениях микросхем, составе различных серий, условиях их эксплуатации и т. п. см. [8—10] или отраслевые справочники.

Габариты микросхемы определяет не кристалл кремния, а выводы из корпуса. Поэтому если элементы простые, то, чтобы использовать по возможности все выводы стандартного корпуса, в нем размещают несколько элементов. Простые логические элементы обычно размещаются в корпусах с 14 выводами, из которых один вывод — это питание и один вывод — общий провод всех логических входов, выходов и питания, кратко называемый *общий* или, менее строго — *земля*. Оставшиеся 12 выводов — логические. Примеры состава корпусов: $4\times$(2И) — четыре двухвходовых элемента И (заняты все 12 логических выводов), $2\times$(4 И-НЕ) — два четырехвходовых И-НЕ (не использованы только два вывода). Более сложные логические узлы размещаются в корпусах с 16, 24 и бóльшим числом выводов.

В настоящее время наиболее распространены три технологии изготовления логических элементов: ТТЛ (и ТТЛШ), КМДП и ЭСЛ.

Для технологии ТТЛ (*транзисторно-транзисторной логики*) и ТТЛШ (ТТЛ с диодами Шотки) самыми удобными для изготовления являются элементы И-НЕ. Именно они шире всего используются для построения более сложных узлов. Второй особенностью является простота изготовления двухступенчатых логических элементов И-ИЛИ-НЕ. В текстовых документах и справочниках логические возможности таких элементов принято обозначать в сокращенной форме. Принцип обозначения понятен из следующих примеров:

2-2-2И-3ИЛИ-НЕ — функция $Y=\overline{ab \vee cd \vee ef}$;

3-3И-2ИЛИ-НЕ — функция $Y=\overline{abc \vee def}$;

2-2-4И-3ИЛИ-НЕ — функция $Y=\overline{ab \vee cd \vee efgh}$.

В серии ТТЛ есть *расширяемые* элементы И-ИЛИ-НЕ,

к которым можно подключать *расширители,* увеличивая тем самым число входов ИЛИ (рис. 2.1). Обозначение метки *EX* происходит от *expand* — расширять. Сумма входов ИЛИ, имеющихся внутри корпуса и добавляемых за счет расширителей, обычно ограничена числом 8. Выпускаются расширители двух типов — с числом входов И, равным 4 и 8. На

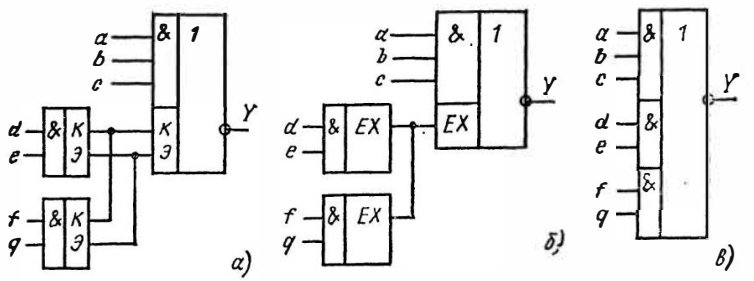


Рис. 2.1. Способы изображения расширяемого элемента И-ИЛИ-НЕ с расширителями:

а — для электрических схем; *б* — для функциональных подробных схем (отмечен факт использования расширителей); *в* — для функциональных схем общих (отмечена лишь результирующая функция комплекса $Y=\overline{abc \vee de \vee fg}$)

рис. 2.1 показаны различные по степени подробности способы изображения расширяемого элемента И-ИЛИ-НЕ с подключенными расширителями.

Элементы ТТЛ, а тем более их усовершенствованная модификация — ТТЛШ, имеют хорошее быстродействие, удовлетворительные электрические и эксплуатационные характеристики, хорошо отработаны технологически, следствием чего является достаточно высокая надежность. Им не свойственны труднообъяснимые эффекты при неисправностях, как у элементов КМДП. В одном устройстве допустимо использование элементов ТТЛ и ТТЛШ различных серий. Большинство микропроцессорных БИС и БИС памяти согласовано по питанию и уровням сигналов с элементами ТТЛ. Серии ТТЛ и ТТЛШ — наиболее распространенные и популярные у разработчиков цифровой автоматики. Подробнее о них см. [3, 8, 10, 28].

Комплементарные (взаимно дополняющие) МДП — структуры (КМДП), построенные на основе МДП-транзисторов с различным типом проводимости. В КМДП-схемотехнике операции И и ИЛИ реализуются комбинированием последовательного и параллельного включения МДП-транзисторов. Серьезных технологических преимуществ

у операции И перед операцией ИЛИ нет, поэтому в КМДП-сериях одинаково широко используются и элементы И-НЕ и ИЛИ-НЕ. Готовых двухступенчатых функций типа И-ИЛИ в КМДП-сериях в отличие от ТТЛ пользователю обычно не предоставляется, однако в состав серий, как правило, входят транзисторные сборки, позволяющие самостоятельно набирать любые функции непосредственно из МДП-транзисторов. Возможность строить функции из элементов И, ИЛИ и т. п. при этом также остается. Кроме того, по КМДП-технологии легко изготавливаются полупроводниковые двунаправленные ключи, способные подобно тумблеру разрывать электрическую цепь независимо от направления тока. Будучи даже проще, чем элементы И и ИЛИ, ключи наряду с обычными логическими элементами могут применяться при построении логических схем, как правило, в режиме монтажного ИЛИ. О монтажном ИЛИ см. § 2.5.

Элементы КМДП исключительно экономичны по потребляемой мощности, что на сегодня является их основным достоинством. Кроме того, они способны работать в широком диапазоне напряжений питания, например от 3 до 15 В, имеют хороший порог помехоустойчивости. Недостатком их являются пока еще заметно меньшее, чем у ТТЛ, быстродействие, а также то, что в случае неисправности или даже кратковременного нарушения электрических режимов в них возможно возникновение довольно сложных эффектов, требующих для своего обнаружения определенной квалификации. Основы схемотехники на МДП-транзисторах даны в [59]. Подробно микросхемы КМДП описаны в [10] и особенно в [11], справочные данные по КМДП-сериям можно найти в [8, 9], вопросы синтеза логических элементов непосредственно на кристалле подробно и квалифицированно рассмотрены в [12].

Эмиттерно-связанная логика (ЭСЛ) использует принцип переключения тока из одного транзистора в другой в паре транзисторов, работающих на общее эмиттерное сопротивление. Этот принцип обеспечивает работу транзисторов без насыщения, поэтому ЭСЛ-серии являются на сегодня наиболее быстродействующими. Однако реализация быстродействия ЭСЛ-элементов возможна лишь при высокой грамотности проектирования и технологической культуре изготовления монтажа. Поэтому на элементах ЭСЛ строят наиболее быстродействующие большие ЭВМ, а в цифровых устройствах автоматики ЭСЛ-серии практически не используют. Применению элементов ЭСЛ в небольших автономных устройствах автоматики мешает также несогласованность уровней

их сигналов с сериями ТТЛ, КМДП, большинством серий выпускаемых микросхем памяти, микропроцессоров, а также недостаточно широкий диапазон рабочих температур.

Принципы построения логических схем на элементах ЭСЛ те же, что и на элементах других технологий. Поэтому сведения, изложенные в книге, будут полезны при проектировании на любых сериях. Однако, учитывая ориентацию книги на устройства цифровой автоматики, для иллюстраций будут использованы серии ТТЛ и КМДП. Подробно методы работы с ЭСЛ-элементами изложены в [13], справочные сведения можно найти в [8, 9].

В табл. 2.1 приведены наборы микросхем отдельных логических элементов, выпускаемых в рамках некоторых широко распространенных серий ТТЛ, ТТЛШ, КМДП. Такие микросхемы называют микросхемами *малой интеграции,* а на арго схемотехников — *россыпью.* Из таблицы видно, что наиболее полно во всех сериях представлены элементы И-НЕ и ИЛИ-НЕ, поскольку технология микросхемотехники непосредственно создает именно их. Как уже отмечалось, в ТТЛ- и ТТЛШ-сериях технологически просто получаются также элементы И-ИЛИ-НЕ, позволяющие весьма экономично реализовывать ДНФ. Эти элементы представлены несколькими модификациями в соответствующих сериях. Существенно меньше выпускается элементов И и ИЛИ без инверсии; они неестественны для рассматриваемых технологий, и на кристаллах их получают уже схемным способом, подключая к выходу базового элемента инвертор. Будучи выпущенными в виде отдельных микросхем, они иногда даже имеют увеличенное время задержки, а в матричных БИС (о них см. [14, 15]) и при построении вообще любых схем, предназначенных для воплощения на кристалле, логические функции чистых И и ИЛИ не применяют.

Функцию сложения по модулю 2 также получают на кристалле за счет коммутации элементов И-НЕ и ИЛИ-НЕ. Построенные таким образом элементы М2 выпускают и в виде микросхем, что отражено в табл. 2.1.

Из сказанного следует, что при представлении или декомпозиции сложных функций применение совместно с функциями И-НЕ и ИЛИ-НЕ функций М2, И и ИЛИ выгодно при построении цифровых устройств на интегральных схемах, поскольку это экономит число корпусов, но при разработке схем матричных и других БИС применение функций М2, И, ИЛИ не обеспечивает экономии площади кристалла, так как эти функции все равно строятся из элементов И-НЕ, ИЛИ-НЕ.

Таблица 2.1

Основные параметры и выполняемая функция	Технология					
	ТТЛ		ТТЛШ		КМДП	
	Серия					
	134 К134	133 К155	530 К531	533 К555	164 К176	564 К561
Типовая средняя задержка, нс	80	19	5	20	200	100
Типовая средняя потребляемая мощность одним элементом в статике	2 мВт	20 мВт	35 мВт	4 мВт	1,5 мкВт	0,7 мкВт
6НЕ		ЛН1	ЛН1	ЛН1		ЛН1
4(2И-НЕ)	ЛБ1	ЛА3	ЛА3	ЛА3	ЛА7	ЛА7
3(3И-НЕ)		ЛА4	ЛА4	ЛА4	ЛА9	ЛА9
2(4И-НЕ)	ЛБ2	ЛА1	ЛА1	ЛА1	ЛА8	ЛА8
8И-НЕ	ЛА2	ЛА2	ЛА2	ЛА2		
4(2ИЛИ-НЕ)		ЛЕ1	ЛЕ1	ЛЕ1	ЛЕ5	ЛЕ5
3(3ИЛИ-НЕ)					ЛЕ10	ЛЕ10
2(4ИЛИ-НЕ)		ЛЕ2			ЛЕ6	ЛЕ6
4(2И)		ЛИ1		ЛИ1		
3(3И)			ЛИ3	ЛИ3		
2(4И)				ЛИ6		
4(2ИЛИ)		ЛЛ1	ЛЛ1	ЛЛ1		
4(2М2)		ЛП5	ЛП5	ЛП5	ЛП2	ЛП2
2-2И-2ИЛИ-НЕ+2-4И-2ИЛИ-НЕ	ЛР1					
2-2И-2ИЛИ-НЕ+3-3И-2ИЛИ-НЕ			ЛР11	ЛР11		
2-2-3-4И-4ИЛИ-НЕ	ЛР2		ЛР9			
2(2-2И-2ИЛИ-НЕ) один с расширением по ИЛИ		ЛР1				
4-4И-2ИЛИ-НЕ с расширением по ИЛИ	ЛР4	ЛР4				
2-2-2-3И-4ИЛИ-НЕ с расширением по ИЛИ		ЛР3				
2(4И—расширитель по ИЛИ)		ЛД1				
8И—расширитель по ИЛИ		ЛД3				

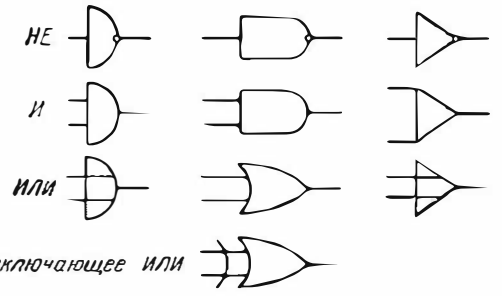


Рис. 2.2. Условные обозначения логических функций, встречающиеся в зарубежной и старой отечественной литературе

Серии цифровых микросхем в большой мере унифицированы в международном масштабе. Данные по зарубежным аналогам выпускаемых в нашей стране микросхем можно найти в [10]. На рис. 2.2 приведены условные изображения логических элементов, встречающиеся в зарубежной, а также использовавшиеся в отечественной литературе, до введения современных стандартов.

Таблица 2.1 иллюстрирует еще один существенный момент: сходство различных серий по набору предоставляемых ими функций. Это сходство распространяется и на серии ЭСЛ, оно захватывает микросхемы не только малой, но и средней степени интеграции. Причин в основном две. Во-первых, изготовители элементов на основе многолетней статистики уже почти нащупали те наборы, которые при весьма ограниченной номенклатуре позволяют строить экономичные схемы. Во-вторых, для того, чтобы разработчики охотно приняли и быстро освоили какую-то новую серию логических микросхем, набор ее элементов, система связей, в общем основные выразительные средства этого предлагаемого нового языка, должны быть аналогичными уже используемым сериям. Поэтому часто вновь выпускаемые серии логических микросхем в первую очередь повторяют номенклатуру старых и только потом начинают ее расширять.

Эта ситуация захватывает и матричные БИС. Разработчик схем, предназначенных для размещения на базовом матричном кристалле, обычно пользуется стандартными логическими элементами из определенного набора, называемого *библиотекой*. Стремясь сделать библиотечный набор возможно более оптимальным, его создатели учитывают множество факторов, в том числе и уровень преемственности набора, влияющий на быстроту освоения нового типа БИС инженерами. С проблемами, возникающими при разработке библиотек матричных кристаллов, можно познакомиться в [12, 14]. Существенно, что в результате учета всех важных факторов состав библиотек матричных БИС оказывается довольно похожим на наборы элементов распространенных серий микро-

схем, и схемотехник, свободно работающий на обычных микросхемах, достаточно легко переходит на базовые кристаллы матричных БИС. Хорошей преемственности библиотек способствует и еще одно обстоятельство: часто, прежде чем реализовать блок на матричной БИС, его макетируют на обычных микросхемах, и чем ближе наборы библиотеки БИС и серии микросхем, тем проще это делать.

Таким образом, весьма различные по технологии наборы обладают большой логической общностью. Отсюда следует, что методы логического проектирования в современных технических базисах и основные свойства функциональных узлов можно изучать, почти не привязываясь к какой-либо конкретной серии. В случаях, когда такая привязка желательна, используются наиболее распространенные серии: из ТТЛ — К155, из КМДП—К561. Иллюстрация положений логического синтеза на базе более новых серий, например К555 ТТЛШ, не дает преимуществ, поскольку принципы построения схем на ТТЛШ те же, что и на ТТЛ. Однако по полноте номенклатуры, доступности и распространенности новые серии пока еще уступают сериям К155 и К561, и с наибольшей вероятностью начинающие разработчики в первую очередь будут иметь дело именно с ними.

2.2. Временные характеристики логических элементов

На рис. 2.3 показана упрощенная картина реакции инвертирующего логического элемента на изменение входного сигнала. Процесс изменения напряжения от низкого *L* уровня к высокому *H* называют *фронтом* сигнала, а обратный процесс — *срезом*. Иногда говорят *положительный фронт и отрицательный фронт*, а если приходится говорить о фрон-

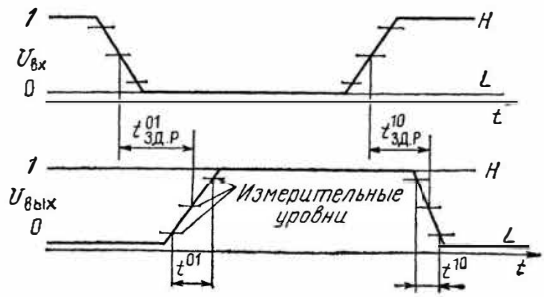


Рис. 2.3. Временные параметры логических элементов

тах и срезах одновременно, то говорят *фронты*. Длительность перехода элемента из 0 в 1 и наоборот измеряется между специально оговоренными в ТУ измерительными уровнями, отличающимися от номинальных уровней 0 и 1 на 0,1—0,3 амплитуды сигнала. *Время перехода из 0 в 1* обозначается t^{01}, *время перехода из 1 в 0* — t^{10}. Как правило, эти времена не одинаковы и оба приводятся в справочниках. Задержка между фронтами сигналов на входе и выходе логического элемента называется *задержкой распространения сигнала* и измеряется на оговоренном в ТУ уровне, близком к половине единичного уровня. Различают задержку распространения при переключении выхода элемента из 0 в 1 — $t^{01}_{\text{зд.р}}$ и задержку распространения при переключении выхода из 1 в 0 — $t^{10}_{\text{зд.р}}$. Как и фронты, эти задержки также обычно не равны друг другу.

При прохождении перепада сигнала по достаточно длинной цепочке инвертирующих элементов число элементов, переключающихся в 1 и 0, будет практически одинаково и равно половине общего числа элементов в цепочке. Поэтому для оценки задержки распространения всей цепочки можно пользоваться *средним временем задержки распространения сигнала* $t_{\text{зд.р.ср}}$, равным полусумме $t^{01}_{\text{зд.р}}$ и $t^{10}_{\text{зд.р}}$ умножая его на число элементов цепочки. В силу распространенности инвертирующих элементов такой способ оценки задержки широко используется. Термин «среднее время задержки» ни в коем случае не следует понимать как математическое ожидание времени задержки с учетом его разброса; в справочниках, если нет специальных оговорок, указаны максимально возможные времена задержек.

На рис. 2.4 показан процесс распространения фронта сигнала по цепочке элементов. Общее время T переходного процесса складывается из суммы задержек распространения всех элементов цепочки плюс половина фронта входного сигнала плюс половина фронта сигнала на выходе последнего элемента. Поскольку и задержки и длительности фронтов элемента — величины одного порядка, то при достаточно длинной цепочке слагаемым в одну длительность фронта можно пренебречь и оценивать время переходного процесса лишь суммой задержек распространения элементов. На практике так обычно и поступают, кроме случаев анализа сигналов обратной связи, если они проходят по очень короткой цепочке.

Большинство функциональных узлов имеет несколько

входов и выходов, что не позволяет охарактеризовать длительность переходных процессов этих узлов единственным числом. На рис. 2.4, *в* показан в общем виде некоторый функциональный узел *A* с двумя входами и двумя выходами. При постоянном уровне сигнала на входе *a* и изменении сигнала на входе *b* узел *A* придется характеризовать двумя

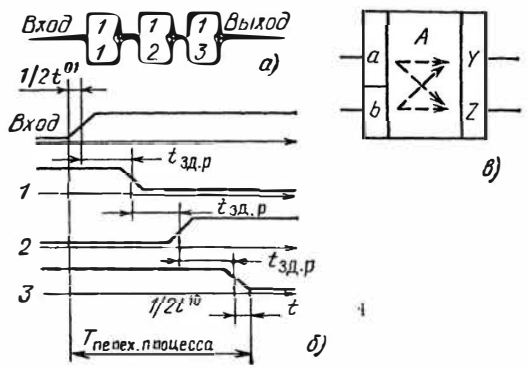


Рис. 2.4. Переходный процесс в группе элементов:
а — цепочка элементов; *б* — временна́я диаграмма переходного процесса в цепочке; *в* — тракты распространения сигналов в функциональном узле

значениями задержек распространения — по тракту bY и тракту bZ. Более того, при $a=0$ и $a=1$ число переключающихся элементов, т. е. *глубина трактов* bY и bZ в общем случае не одинакова, и тогда переходный процесс в узле *A*, вызванный изменением сигнала на входе *b*, придется характеризовать уже четырьмя значениями задержки:

от b до Y при $a=0$;
от b до Z при $a=0$;
от b до Y при $a=1$;
от b до Z при $a=1$.

Аналогичной четверкой будет характеризоваться и задержка сигнала от входа *a* до выходов *Y* и *Z*. Схему с большим, чем на рис. 2.4, *в*, числом входов и выходов нужно в общем случае характеризовать довольно сложной матрицей. К счастью, в реальных схемах число временны́х параметров так быстро не растет. Выручают схемная идентичность многих входов, позволяющая объединять их в группы с равной задержкой, и отсутствие влияния большинства входов на глубину чужих трактов. Поэтому редкие узлы характеризуются более чем шестью значениями задержки.

При оценке общей задержки цифрового блока нужно суммировать не какие-то унифицированные задержки функциональных узлов, а задержки узлов именно по тем трактам, которые срабатывают при выполнении узлом рассматриваемой функции. Соответственно в проектируемой схеме нужно уметь выделять тракты, срабатывающие в различных режимах, группировать идентичные входы, оценивать влияние трактов друг на друга. Для выпускаемых промышленностью микросхем средней и большой интеграции списки задержек приводятся в справочниках. Краткие справочники иногда ограничиваются лишь задержками по самому длинному тракту. Такая усеченная информация не дает возможности схемотехнику реализовать в полной мере потенциально возможное быстродействие микросхемы.

2.3. Оценка качества функциональных схем

Для сравнения между собой различных вариантов схем, реализующих одну и ту же функцию, нужно уметь как-то оценивать их качество. При этом оценка должна, с одной стороны, возможно лучше отражать окончательные показатели качества того реального блока, который будет построен на основе данной схемы, а с другой — выполняться только на базе той информации, которую несет функциональная схема, иначе выбор одной из конкурирующих схем на этапе логического проектирования будет невозможен.

Требования эти противоречивы, поскольку на качество блока влияют также и параметры этапа конструкторского проектирования — характер размещения элементов, трассировки связей и т. п., которые еще не известны на этапе разработки функциональной схемы. Поэтому разумным компромиссом является постановка вопроса не о точном вычислении значения качества, а лишь о приближенной его оценке, позволяющей если не выбрать гарантированно наилучшую функциональную схему, то хотя бы отсеять множество явно неперспективных и выделить небольшой список неразличимых по качеству на данном этапе с целью дальнейшего более внимательного их рассмотрения.

Наиболее распространена оценка схемы по двум параметрам — задержке T и аппаратурным затратам W, поскольку значения ряда других важных параметров цифрового блока—потребляемой мощности, частоты отказов, стоимости — при заданной элементной базе и априорно невысокой точности оценок в первом приближении допустимо счи-

тать пропорциональными аппаратурным затратам. В дальнейшем методика оценки качества схем иллюстрируется на примере именно этой пары параметров — T и W. Если же проектирование блока специально ориентировано на достижение еще каких-то целей (уменьшение потребляемой мощности, повышение надежности, улучшение контролепригодности и т. п.), то вместо (или вместе с) T и W в процедуру оценки качества схемы можно включить любые актуальные для разработчика параметры. Каких-либо принципиальных изменений в излагаемом подходе это не вызовет.

Способы оценки схем излагаются в предположении, что тип элементной базы — ТТЛ- или КМДП-микросхемы, вид базового матричного кристалла и т. п. — уже выбран: обычно эти вопросы решаются на уровне главного конструктора на этапе эскизного проектирования, поэтому схемотехник, разрабатывающий логические схемы на уровне функциональных узлов, тип элементной базы уже не выбирает.

При работе на микросхемах задержка T схемы достаточно объективно оценивается значением среднего времени задержки распространения $t_{зд.р.ср}$ входящих в нее элементов. В рамках одной серии обычно целесообразно полагать, что задержки всех логических элементов россыпи — И-НЕ, И, И-ИЛИ-НЕ, М2 — одинаковы и равны некоторой усредненной для данной серии величине τ. Это близко к истине, поскольку в микросхемах не только малой, но иногда и средней интеграции основная доля задержки приходится на мощный выходной каскад, перезаряжающий большую емкость печатного монтажа. Задержки внутренней, размещенной на кристалле части схемы намного меньше, поскольку там меньше размеры линий связи и, следовательно, их паразитные емкости. Для серий К155 и К555, например, значение τ можно принять равным 20 нс. Задержку более сложных микросхем средней интеграции, целесообразно округлять до значения, кратного целому τ или его половине.

Аппаратурные затраты W функционального узла оценивают различными способами. Хорошие результаты дает оценка величины W площадью, занимаемой узлом на плате или кристалле. При использовании микросхем площадь платы при прочих равных условиях приблизительно пропорциональна числу корпусов. Размеры корпусов различны, поэтому их приходится приводить к какому-то единому, принятому за единицу. В качестве масштаба можно использовать отношение площадей корпусов или чисел их выводов. Можно

оценивать величину W схемы и непосредственно суммарным числом выводов всех корпусов.

При оценке задержки схем, предназначенных для воплощения на кристалле матричной БИС, следует иметь в виду, что для целого ряда технологий задержка средней межэлементной связи на кристалле соизмерима с задержкой самого логического элемента. Поэтому действительное значение задержки на кристалле становится известным лишь после размещения на нем элементов и трассировки связей. Это приводит к тому, что методика оценки задержки схемы суммированием задержек логических элементов дает при работе на кристалле заметно бо́льшую погрешность, чем та же методика при работе на отдельных микросхемах. Может оказаться, что оценка вариантов на уровне функциональных схем так и не выделит какой-либо одной наилучшей схемы, а лишь сузит поле поиска, выявив небольшое число кандидатов для дальнейшего, более детального рассмотрения. Правда, эти опасения относятся к крупным узлам, имеющим длинные связи, и к быстродействующим БИС. В небольших узлах объемом в 20—40 элементов и на матричных БИС невысокого быстродействия задержки в связях заметно меньше задержек в элементах, и простые оценки работают удовлетворительно.

Аппаратурные затраты схем, предназначенных для размещения на матричных БИС, часто можно оценивать числом логических элементов или библиотечных ячеек, содержащих сразу несколько элементов наподобие корпусов микросхем. Структура и значения параметров некоторых типов базовых матричных кристаллов приведены в [15].

Некоторые приемы работы с реальными микросхемами и подход к оценке качества схем иллюстрирует следующее упражнение.

Упражнение. На логических микросхемах серии К155 (табл. 2.1) построить несколько вариантов схем, реализующих заданную минимальную ДНФ (2.1). Сравнить полученные результаты.

$$Y_{\text{а}} = \overline{ac} \vee \overline{bc} \vee \overline{d}. \qquad (2.1)$$

Схемная реализация этой формулы представлена на рис. 2.5, *а*. Поскольку на основании данных справочника задержки распространения элементов ЛР3 и ЛН1 серии К155 одинаковы, задержка всей схемы равна 3τ. Аппаратурные затраты состоят из пяти инверторов ЛН1, каждый из которых занимает 1/6 корпуса, и элемента И-ИЛИ-НЕ—ЛР3, за-

нимающего целый корпус. Итого для схемы по рис. 2.5, *а*

$$W_a = 5 \cdot 1/6 + 1 = 22/12 \text{ корпуса.}$$

Неиспользованные элементы частично занятого корпуса (в данном случае шестой инвертор) не учитываются, поскольку они могут быть использованы в других узлах. Подсчеты удобно производить в двенадцатых долях корпуса: 12 — это число логических выводов корпуса наименьшего

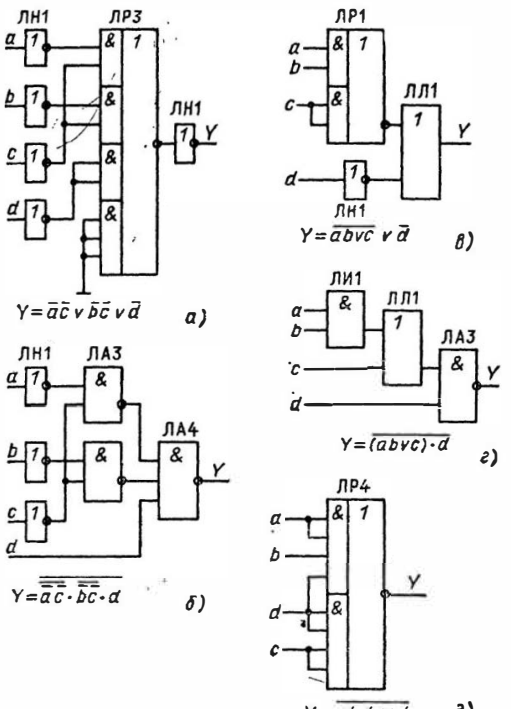


Рис. 2.5. Различные варианты схемной реализации выражения (2.1)

размера. Оценка оборудования в двенадцатых долях корпуса численно близка к оценке оборудования по сумме входов и выходов всех элементов.

Поскольку у ЛР3 часть входов осталась неиспользованной, можно попытаться реализовать Y с помощью элементов И-НЕ, набор которых имеет более мелкую градацию по числу входов. Применив к (2.1) правило де-Моргана, по-

лучим

$$Y_б = \bar{a}\bar{c} \vee \bar{b}\bar{c} \vee \bar{d} = \overline{\overline{\bar{a}\bar{c}} \cdot \overline{\bar{b}\bar{c}} \cdot d}. \qquad (2.2)$$

Реализация (2.2) представлена на рис. 2.5, *б*. Параметры схемы: $T_б = 3\tau$; $W_б = 3 \cdot 1/6 + 2 \cdot 1/4 + 1 \cdot 1/3 = 16/12$ корпуса. Схема оказалась заметно экономичнее.

Можно попытаться использовать формулы де-Моргана для уменьшения числа инверторов на входе. Тогда

$$Y_в = \bar{a}\bar{c} \vee \bar{b}\bar{c} \vee \bar{d} = (\bar{a} \vee \bar{b})\bar{c} \vee \bar{d} =$$

$$= \overline{\overline{\bar{a} \vee \bar{b}} \vee c} \vee \bar{d} = \overline{ab \vee c} \vee \bar{d}. \qquad (2.3)$$

Схема, реализующая (2.3), показана на рис. 2.5, *в*. Для нее $T_в = 2\tau$; $W_в = 11/12$ корпуса. Удалось выиграть и во времени, и в оборудовании. Еще одно применение формулы де-Моргана дает

$$Y_г = \overline{ab \vee c} \vee \bar{d} = \overline{(ab \vee c) \cdot d}. \qquad (2.4)$$

Схема (2.4) представлена на рис. 2.5, *г*: $T_г = 3\tau$; $W_г = 9/12$ корпуса. Схема оказалась очень экономичной, хотя и довольно медленной.

Если в (2.4) раскрыть скобки, то можно получить еще один вариант схемы (рис. 2.5, *д*)

$$Y_д = \overline{abd \vee c\bar{d}}; \quad T_д = 1\tau;$$

$$W_д = 12/12 \text{ корпуса}. \qquad (2.5)$$

Задержка этой схемы оказалась наименьшей из всех рассмотренных.

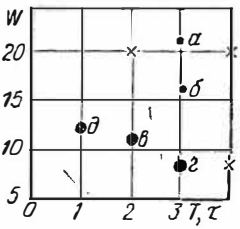


Рис. 2.6. Соотношение величин задержек T и аппаратурных затрат W схем, изображенных на рис. 2.5

На рис. 2.5 показаны не все возможные схемы. Продолжая преобразования логического выражения, читатель самостоятельно сможет построить еще ряд схем и сравнить с уже полученными. Примечательно, что даже весьма простые логические выражения типа (2.1) допускают далеко не одну схемную интерпретацию.

Еще примечательно то, что хотя за основу для построения схем была взята минимальная ДНФ, ее схемная реализация оказалась тем не менее самой неэкономичной из всех. Противоречия здесь нет. Минимальная ДНФ минимальна лишь в определенном узком смысле: это выражение, обязательно принадлежащее классу ДНФ и имеющее

минимальное суммарное число букв. Переводя на язык аппаратуры, можно сказать, что это соответствует минимуму суммы входов всех конъюнкторов, реализующих элементарные конъюнкции ДНФ в трехъярусной схеме НЕ-И-ИЛИ типа показанной на рис. 2.5, *а*. Затраты инверторов и дизъюнкторов этот критерий минимальности игнорирует. Базис россыпи серии К155 намного богаче булевского базиса НЕ, И, ИЛИ, оставаться в жестких рамках ДНФ никто от схемотехника не требует, а выйдя за эти рамки, возможно, удастся найти и другие, более компактные представления или декомпозиции функции. Поэтому практические оценки в единицах корпуса имеют не очень много общего с теоретическими оценками сложности, принятыми в булевой алгебре. Вид выражения, наиболее экономичного в каком-то техническом базисе, может существенно отличаться от вида минимальной ДНФ, и полученную в результате минимизации ДНФ схемотехник часто склонен рассматривать не как окончательный результат, а лишь как полуфабрикат, с которым можно еще поработать. Отсюда не следует, что минимизация не нужна вообще: чем компактнее выражение, тем легче обрабатывать его дальше.

С ростом сложности логической формулы число возможных вариантов быстро растет. При этом, к сожалению, в общем случае не существует алгоритма кроме полного перебора вариантов, который бы позволял целенаправленно строить хорошие схемы. Подход к автоматизированному построению схем, где перебор выполняется с помощью ЭВМ, изложен в [16], но написание программы для этого весьма трудоемко. А построение схемы человеком — это процесс поиска, процесс проб, оценок, сравнений, в общем — чисто изобретательская деятельность. Преобразования алгебры логики вопреки встречающемуся убеждению не есть алгоритм, который целенаправленно, шаг за шагом ведет к поставленной цели, в данном случае — получить хорошую схему. Аппарат алгебры логики есть *исчисление, грамматика,* где на каждом шаге указано лишь, что м о ж н о делать, но не указано — что же н у ж н о делать. В грамматике каждый шаг — это развилка: можно вынести за скобки или одну, или другую группу аргументов, преобразовать по де-Моргану или одну, или другую часть выражения. На этих развилках направление не указывается каким-либо правилом; куда идти — выбирает сам разработчик. Поэтому на результирующей схеме лежит хорошо заметная печать субъективности.

Огромную роль в процессе поиска хорошей схемы играет опыт. Запоминаются удачные фрагменты схем, формулируется множество эвристик — рекомендаций, которые хоть и не гарантируют успеха всегда, но часто к нему приближают. Вот несколько примеров подобных эвристик, которые начинающий разработчик может использовать: применив формулу де-Моргана, сократить число инверторов; использовать элементы И-ИЛИ-НЕ — они логически мощнее, чем И-НЕ, И, ИЛИ; подбирать

такие элементы, чтобы по возможности использовались все их входы; если выражение плохо минимизируется, попытаться применить элементы М2; вместо прямой функции реализовать ее инверсию.

Не существует в общем случае н четких признаков окончания процесса поиска хорошей схемы. В этом смысле разобранный пример не следует воспринимать как требование всегда проводить столь же тщательную обработку любого выражения. Это просто иллюстрация характера работы при логическом проектировании. На решение об окончании процесса поиска влияет опыт разработчика, замедление потока появления схем лучших, чем те, что уже получены, соотношение жесткости сроков разработки схемы и требований к ее качеству и т. д.

Все сказанное об особенностях поиска схемы относится не только к синтезу схемы по заданному логическому выражению. То же самое можно сказать и о процессе построения более сложных блоков из микросхем средней и большой интеграции. Слабо алгоритмизированный, поисковый, изобретательский стиль работы характерен для всех этапов функционально-логического проектирования цифровой аппаратуры.

Построив несколько вариантов схем (а в более общем случае — рассмотрев несколько вариантов решения любой задачи), разработчик должен выбрать какое-то одно решение, в некотором смысле наилучшее. Если фактор, по которому выполняется выбор, только один, например, минимальные аппаратурные затраты, то задача нахождения наилучшего объекта тривиальна. В рассматриваемом случае факторов, оценивающих качество схемы, два: задержка T и аппаратурные затраты W, и задача выбора наилучшего объекта принципиально усложняется.

На рис. 2.6 пять схем, которые были показаны на рис. 2.5, изображены в виде точек на плоскости TW (W дано в двенадцатых долях корпуса). Такой чертеж делает процесс сравнения различных вариантов существенно нагляднее. Нетрудно видеть, что схемы *а* и *б* хуже, чем схема *г*, поскольку при той же задержке их аппаратурные затраты больше. Допустим, что преобразовав (2.1) еще несколькими способами, удалось получить схемы, помеченные на рис. 2.6 крестиками. Все эти схемы также оказываются плохими, поскольку для каждой из них существует хотя бы одна схема, которая лучше нее или по T, или по W, или по обоим показателям сразу. Этого нельзя сказать только о группе схем, образующей «лево-нижнюю» границу всего множества схем, т. е. о схемах *д*, *в*, *г*. При переходе от *д* к *в* и от *в* к *г* на каждом шаге получается выигрыш в аппаратуре, но проигрыш в скорости. Эти объекты л у ч ш е любых других объектов, расположенных выше них и правее, но между

собой по характеристикам T и W они несравнимы. Такая группа объектов называется *множеством объектов, оптимальных по Парето.*

Для произвольной схемы число *Парето-оптимальных* объектов может быть любое. Среди этого множества можно выделить крайние объекты с наилучшими значениями какой-либо одной характеристики, в нашем случае — самую быструю схему *д* и самую экономичную *г*. Именно эти схемы будут соответствовать техническому заданию с требованием экстремальных значений по какой-либо одной характеристике. Однако значения второй характеристики у этих схем будут соответственно наихудшими, чем иллюстрируется неправомерность встречающейся иногда формулировки задания: построить схему с минимальной задержкой и минимальными аппаратурными затратами.

Далеко не всегда задание требует поиска объекта с наилучшими значениями лишь по одной характеристике. Часто задание формулируется как некоторый компромисс между требованиями и к одной, и к другой характеристике, тогда разработчик в качестве окончательного решения выбирает одну из промежуточных точек множества Парето. Распространенной формулировкой задания для цифровых схем является следующая: задержка не более заданной и при этом минимум аппаратурных затрат. Наилучшим решением для такого задания также будет одна из точек множества Парето. Например, если требуемая задержка ограничена величиной 2,5 τ, то, как видно из рис. 2.6, наилучшей будет схема *в*. Схема *г* не удовлетворяет по значению задержки, а схема *д* неоправданно громоздка.

Введение в обиход разработчика понятия Парето-оптимального множества удобно потому, что при всех реально используемых формулировках задания наилучшим решением всегда оказывается одна из точек множества Парето. Число Парето-оптимальных объектов обычно существенно меньше числа всех вариантов объекта, поэтому работать с множеством Парето заметно проще. Можно выделить Парето-оптимальное множество объектов и по трем характеристикам, добавив к T и W еще, например, потребляемую мощность. Подробнее с использованием множества Парето и проблемами многокритериальной оптимизации можно ознакомиться, например, по [16, 17].

Некоторая нечеткость ситуации заключается в том, что набор схем, в действительности принадлежащих множеству Парето, разработчику в процессе работы еще неизвестен, поэтому практически ему приходится пользоваться приближением этого множества, сформированным из

уже построенных схем. Это множество модифицируется по мере получения все лучших схем. В каждый момент времени работа идет с теми лучшими вариантами, которые уже получены. Необходимость сравнения объектов одновременно по нескольким критериям возникает на всех этапах разработки цифрового устройства.

2.4. Правила схемного включения элементов

Ограничение по нагрузочной способности элементов задается коэффициентом разветвления по выходу $K_{раз}$. Коэффициент равен максимальному числу входов той же серии, которые можно подключить к выходу данного элемента. При превышении нагрузки сверх допустимой значения выходных параметров элемента не гарантируются. Различные элементы различных серий имеют $K_{раз}=5\div20$, типовое значение — 10. Мощные элементы со специальным выходным каскадом имеют $K_{раз}\geqslant30$. Часто кроме $K_{раз}$ оговариваются максимально допустимый выходной ток и максимально допустимая емкость нагрузки, что необходимо знать при стыковке элементов различных серий.

Неиспользованные входы И в большинстве серий не должны оставаться ни к чему не подключенными (свободными). В ТТЛ- и ТТЛШ-сериях сигнал от свободного входа воспринимается элементом как логическая единица, и у элементов И-НЕ его можно было бы так и оставить свободным, однако возникающие при этом дополнительные заряды в базе замедляют переключение элемента по другим, работающим, входам. В сериях ТТЛ и ТТЛШ неиспользованные И-входы либо объединяют с другими (см. рис. 2.5, *д*), если при этом не превышена допустимая нагрузка источника сигнала, либо подключают к источнику логической единицы. В качестве последнего используют или элемент И-НЕ, вход которого заземлен, или резистор с сопротивлением 1 кОм, подключенный к источнику питания +5 В. К одному такому резистору обычно разрешается подключать до 20 неиспользованных входов И.

В КМДП-элементах ни в коем случае не должно быть неподключенных входов. На них может оказаться наведенным любой потенциал, соответственно ложный потенциал окажется и на выходе. В сериях КМДП неиспользованные входы можно подключать к источнику питания непосредственно, без резистора. Как и в ТТЛ-сериях, неиспользованные входы можно объединять с рабочими.

Неиспользованные входы ИЛИ в любых сериях должны быть соединены с логическим нулем, т. е. с общим проводом. В неиспользуемой секции ИЛИ элемента И-ИЛИ-НЕ все, или хотя бы один вход И должен быть подключен к общему проводу (см. рис. 2.5, *а*).

Если некоторые элементы, входящие в состав корпуса, не используются, то на входы неиспользуемых элементов ТТЛ-серий нужно подать такие сигналы, чтобы на их выходах была единица: в таком состоянии элемент потребляет меньше мощности, и его можно использовать как источник логической единицы. Неиспользованные элементы КМДП-серий можно фиксировать в любом состоянии, только не оставлять их в безразличном.

Обычный выходной каскад логического элемента ТТЛ показан на рис. 2.7. Он состоит из двух последовательно включенных транзисторов, которыми управляет логическая часть элемента. Один из выходных транзисторов всегда открыт, а другой — закрыт. Если открыт *VT1* и закрыт *VT2*, то общая точка транзисторов подключена к шине питания $U_{\text{ип}}$ и на выходе держится высокий уровень напряжения. Если открыт *VT2* и закрыт *VT1*, то выход подключен к общей шине и уровень напряжения на нем низкий. Включенные таким же образом два униполярных транзистора образуют выходной каскад КМДП-элемента. Открывшиеся транзисторы, имея малое сопротивление, быстро перезаряжают емкость нагрузки и паразитную емкость монтажа, поэтому выходной каскад такого типа широко распространен в цифровых элементах.

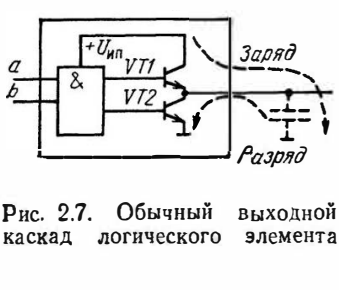


Рис. 2.7. Обычный выходной каскад логического элемента

Если выходной каскад образован последовательно включенными транзисторами, то выходы таких элементов объединять не допускается, что начинающие разработчики иногда стремятся сделать для получения «сборки» (т. е. логического ИЛИ) двух сигналов. Если один из объединенных по выходу элементов отрабатывает на своем выходе 1 и у него открыт *VT1*, а второй элемент отрабатывает 0 и у него открыт *VT2*, то образуется цепь короткого замыкания и выходные транзисторы будут повреждены. Тем более нельзя подключать выходы элементов к общему проводу или шине питания.

2.5. Элемент с открытым коллектором

Кроме логических элементов, имеющих обычный выходной каскад с двумя транзисторами, выпускаются элементы с *открытым (свободным) коллектором (open collector)*. Упрощенная его схема показана на рис. 2.8, *а*. Верхний транзистор обычной выходной пары отсутствует, и из корпуса выведен лишь коллектор нижнего транзистора *VT*. Этот транзистор открывается логической частью элемента при совпадении на входе всех единиц. Если хотя бы один из входных сигналов равен 0, транзистор закрыт. Такой вы-

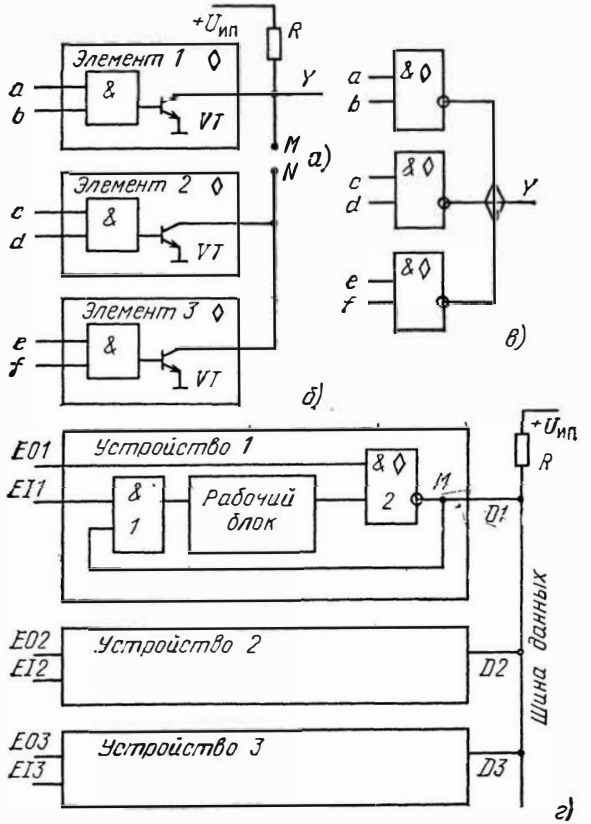


Рис. 2.8. Элементы с открытым коллектором:

а — логический элемент; *б* — организация вместе с элементом по рис. *а* монтажного ИЛИ; *в* — обозначение монтажного ИЛИ на схемах; *г* — организация шины данных

ходной каскад не способен сам по себе сформировать на выходе высокий уровень напряжения. Для этого к выводу корпуса *внешним монтажом* подключается резистор R, который в некоторой степени играет роль верхнего транзистора обычного выходного каскада. Этот резистор называют *подтягивающим* (*pull-up resistor*). В таком виде элемент выполняет функцию И-НЕ. Выход с открытым коллектором помечают специальным значком — ромбиком, показанным на рис. 2.8, *а*. Одно из назначений элементов с открытым коллектором — быть переходным звеном от логической части устройства к элементам вывода информации. Вместо резистора R (или наряду с ним) в коллекторую цепь выходного транзистора могут быть включены светодиод, индикаторная лампочка, обмотка реле, коаксиальный кабель, вход усилителя мощности. Конкретные схемы подключения некоторых из этих элементов приведены в [8, 10, 11]. Выходной транзистор многих элементов с открытым коллектором имеет повышенный допустимый ток, а иногда и повышенное коллекторное напряжение. Примеры элементов с открытым коллектором: К155ЛА7; К155ЛА11. Открытый коллекторный выход иногда имеют и микросхемы более сложных функциональных узлов, например дешифратор К155ИД1, многие микросхемы памяти.

Элементы с открытым коллектором в отличие от обычных логических элементов приспособлены для объединения по выходу. Если точку M на рис. 2.8, *а*, объединить с точкой N пары элементов на рис. 2.8, *б*, то уровень выхода будет равен 0, когда открыт выходной транзистор любого из объединенных элементов. Все три элемента совместно будут отрабатывать функцию И-ИЛИ-НЕ:

$$Y = \overline{ab \vee cd \vee ef}.$$

Такую схему называют *монтажное ИЛИ, проводное ИЛИ, wired OR.* Впрочем на логику работы полученной схемы можно посмотреть и несколько иначе: на объединенном выходе будет 1 тогда и только тогда, когда каждый элемент будет стремиться установить на своем выходе 1. Если же хотя бы один элемент установит на своем выходе 0, этот же 0 установится и на объединенном выходе. Такое рассуждение позволяет рассматривать объединение открытых коллекторов на общем подтягивающем резисторе как схему, реализующую функцию И: единица при всех единицах и нуль при любом нуле. Это дало повод для еще одного названия рассматриваемой схемы: *монтажное И.*

Максимальное число объединяемых элементов и максимальное значение подтягивающего сопротивления ограничиваются соотношением значения этого сопротивления и токов утечки выходных транзисторов: когда все они закрыты, падение напряжения на сопротивлении от суммарного тока утечки не должно снижать высокий выходной уровень ниже допустимого. По минимуму величина подтягивающего сопротивления ограничена максимально допустимым током открытого выходного транзистора. Постоянная времени положительного фронта равна произведению значения подтягивающего сопротивления на емкость нагрузки. Изображение монтажного ИЛИ на функциональных схемах показано на рис. 2.8, *в*.

Схему монтажного ИЛИ при работе на россыпи в ТТЛ- и КМДП-сериях с чисто логическими целями используют не часто в силу худших ее частотных свойств по сравнению с элементами, имеющими обычный выходной каскад. Однако принцип монтажного ИЛИ широко используется в БИС памяти и ПЛМ, а также для построения двунаправленных *числовых шин (числовых магистралей — data bus)*.

Для взаимного обмена данными различные устройства подключаются к связывающему их единому проводу — шине, как показано на рис. 2.8, *г*. По этой шине и происходит обмен. Основой каждого устройства является рабочий блок, выполняющий функции этого устройства, какие — в данном случае безразлично. Выход блока через элемент *2* с открытым коллектором подключен к числовой шине. При единичном значении сигнала *EO* (от *enable out* — разрешение выхода) выходной сигнал соответствующего рабочего блока пропускается на шину, при нулевом его значении выходной транзистор элемента *2* заперт и выход блока отключен от шины. Если единичное значение сигнала *EO* в каждый отрезок времени подавать только на одно устройство, то в течение этого отрезка уровни на шине будут отображать уровни выходного сигнала именно этого устройства.

Вход блока через конъюнктор *1* подключен к точке *M*, находящейся внутри корпуса устройства (или внутри монтажной платы). Точка *M* связана с шиной через тот же самый вывод из корпуса (контакт разъема платы), что и выход блока. Если на некоторый отрезок времени нулевым значением сигнала *EO* отключить от шины выход блока, а единичным значением сигнала *EI* (от *enable in* — разрешение входа) подключить к шине его вход, то логический блок будет воспринимать сигналы с шины. Входы

других устройств, подключенных к шине, сделаны по такой же схеме. Подавая в течение некоторого отрезка времени на устройство, например *1*, разрешение выхода *EO*, а на другое устройство, например *3*, — разрешение входа *EI*, можно передать сигнал от устройства *1* к устройству *3*. В другой отрезок времени, подав выбирающие сигналы *EO* и *EI* на другие устройства, можно и им дать возможность обменяться данными. Если обмен нужно вести многоразрядными числами, то соответственно увеличивается число проводов магистрали, а в каждом устройстве — число входных (*1*) и выходных (*2*) элементов.

Полученную числовую магистраль на базе элементов с открытым коллектором называют *двунаправленной,* поскольку и ввод информации в корпус (плату), и выдачу ее осуществляют через один и тот же вывод корпуса (контакт разъема), используя разделение во времени. Двунаправленные магистрали существенно сокращают требуемое число выводов из корпуса (контактов разъема платы) по сравнению с однонаправленными, где требуются отдельные выводы и для выхода, и для входа. Соответственно сокращается и число проводов магистрали. Поэтому двунаправленные магистрали широко применяются в цифровой аппаратуре.

Когда выходы всех устройств отключены от шины, т. е. шина находится в состоянии информационного нуля, на ней держится высокий уровень напряжения. Именно на этом фоне идут по шине информационные посылки подключенных устройств. Поэтому естественным соглашением для обмена по шинам с открытым коллектором является соглашение отрицательной логики, когда активный уровень НИЗКИЙ. Это соответствует и меньшей потребляемой мощности в промежутках между операциями обмена.

2.6. Элемент с тремя состояниями выхода

На рис. 2.9, *а* в условном виде показана упрощенная электрическая схема элемента *с тремя состояниями выхода (3-state output, Z-буфер)*. Иногда называют неверно: с тремя устойчивыми состояниями. Когда на входе элемента, обозначенном *Z*, уровень НИЗКИЙ, то транзистор *VT3* заперт и не влияет на работу схемы элемента, выполняющего обычным образом операцию И-НЕ. Если сигнал *Z* имеет высокий уровень, то низкий уровень коллектора открытого транзистора *VT3* передается через диоды на базы

выходных транзисторов *VT1* и *VT2* и запирает их оба. В результате связь логической части элемента с его выходом разрывается, элемент со стороны выхода приобретает *высокий импеданс (high impedance)*. Уровень потенциала на выходе уже не определен (*плавающий потенциал — floating level*). Он может быть любым в зависимости от соотношения токов утечки транзисторов *VT1* и *VT2*, но такое состояние потенциала в качестве значения входного сигнала не используется. *Третье состояние выхода* в отличие от 1 и 0 обозначается *Z*. На рис. 2.9, *б* показано условное

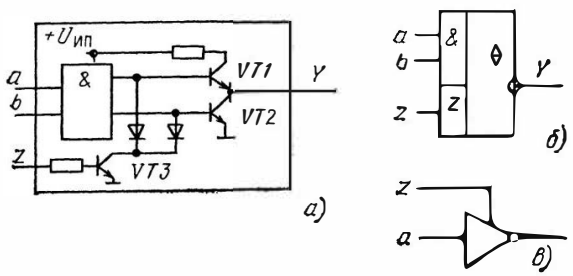


Рис. 2.9. Элемент с тремя состояниями выхода

обозначение элемента с тремя состояниями выхода. Символ такого выхода — ромб с поперечной чертой. На рис. 2.9, *в* показан способ изображения *Z*-буфера на функциональных схемах, распространенный в переводной литературе. Таблица 2.2 иллюстрирует табличный способ задания работы элемента с тремя состояниями выхода.

Таблица 2.2

Входы			Выход	Входы			Выход
Z	*a*	*b*		*Z*	*a*	*b*	
1	×	×	*Z*	0	×	0	*H*
0	0	×	*H*	0	1	1	*L*

Z-буфер характеризуется задержками распространения двух трактов: логические входы — выход при $Z=0$ и *Z*-вход — выход.

Элементы с тремя состояниями выхода разработаны специально для применения в качестве выходного управ-

ляемого буфера для подключения цифровых блоков к магистралям. Схема такой магистрали аналогична показанной на рис. 2.8, *г*, только в качестве элементов *2* использованы *Z*-буферы. Принцип обмена данными между устройствами такой же, как и по магистралям с открытым коллектором. Подключение магистрали к источнику питания через подтягивающий резистор не обязательно, так как *Z*-буфер задает на выходе и нижний, и верхний уровни сигнала.

Поскольку выходной каскад *Z*-буфера построен на основе двух последовательно включенных транзисторов, подача разрешающих сигналов на *Z*-входы сразу двух буферов магистрали *недопустима:* результат будет такой же, как и при объединении выходов двух обычных логических элементов.

Благодаря наличию в схеме на рис. 2.9, *а* транзистора *VT1*, быстро заряжающего паразитную емкость шины до высокого уровня, частотные свойства шины, управляемой буферами с тремя состояниями, выше, чем шины, управляемой буферами с открытым коллектором, или, при тех же частотных свойствах, меньше потребляемая мощность. Зато шина с открытым коллектором оказывается *fool proof:* она не боится ошибочного открытия сразу каких-либо и двух, и более выходных буферов. Подробнее прочесть о буферах и магистралях с тремя состояниями можно в [3] и другой литературе, посвященной микропроцессорам. Примеры буферов с тремя состояниями выхода или, как их еще называют, *шинных драйверов* — микросхемы К589АП16, К589АП26, К561ЛН1.

Способ обмена с помощью магистралей помимо небольших затрат оборудования очень удобен для расширения системы, когда в процессе ее эксплуатации требуется подключение дополнительных устройств. Для этого достаточно подсоединить двунаправленные выводы этих устройств к шине данных, а выбирающие выводы — к адресной шине. Функцию элемента ИЛИ выполняет сам способ подключения, монтаж.

2.7. Расширение логических возможностей элементов

При проектировании схем иногда возникает необходимость в логических элементах с числом входов или коэффициентом разветвления по выходу, превышающими паспортные значения реальных элементов. В этих случаях требуемые большие элементы строят из обычных.

Увеличение нагрузочной способности

Упражнение. Обеспечить разводку сигнала на число элементов, превышающее коэффициент разветвления по выходу $K_{раз}$ элемента, служащего источником сигнала. Дать ре-

шение как для небольшой кратности превышения, так и для превышений в десятки раз.

Р е ш е н и е. На рис. 2.10 показаны возможные варианты схем. Простое запараллеливание элементов (рис. 2.10, *а*) в ТТЛ-сериях допускается в виде исключения не

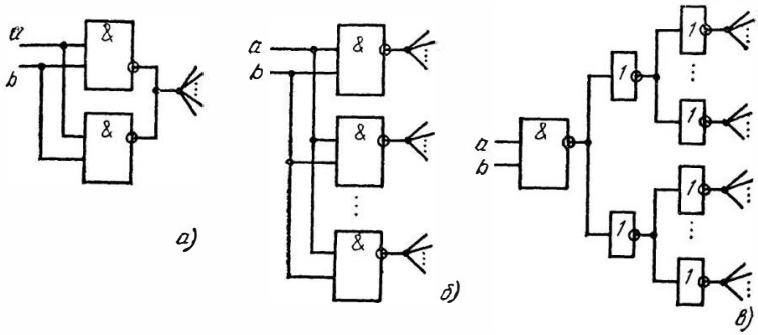


Рис. 2.10. Способы размножения сигнала

более чем для двух элементов, при этом считается, что результирующий коэффициент разветвления по выходу увеличивается лишь в 1,9 раза. Оба элемента должны быть размещены в одном корпусе, в противном случае возникающие уравнительные токи могут вызвать помехи в других цепях. В МДП-сериях ограничений на число запараллеливаемых элементов, как правило, не вводится.

В схеме по рис. 2.10, *б* число буферных элементов ограничено допустимой нагрузкой источников сигналов *a* и *b*. Схема каскадного размножения по рис. 2.10, *в* дает очень большой коэффициент разветвления, исчисляемый сотнями, что требуется, например, при разводке по цифровому устройству цепей синхронизации. Сигнальные токи в обоих случаях остаются номинальными.

Построение многовходовых И

Упражнение. На элементах серии К155 построить И на 16 входов.

П о д с к а з к а. Типовые ошибки начинающих схемотехников: а) включение двухвходовых И *ЛИ1* в четыре каскада; б) построение шестнадцативходовой И из восьмивходовых И, которые, в свою очередь, строятся из восьмивходовых И-НЕ *ЛА7* с подключенным на выходе инвертором *ЛН1*. Эти решения расточительны и по оборудованию, и по задержке.

Решение. Лучшее решение дает разбиение функции с помощью формулы де-Моргана на такие группы, размер которых соответствует числу входов имеющихся элементов:

$$Y = x_0 x_1 x_2 \ldots x_{14} x_{15} =$$

$$= \overline{\overline{x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7} \vee \overline{x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}}} =$$

$$= \overline{\overline{x_0 x_1 x_2 x_3} \vee \overline{x_4 x_5 x_6 x_7} \vee \overline{x_8 x_9 x_{10} x_{11}} \vee \overline{x_{12} x_{13} x_{14} x_{15}}}. \quad (2.6)$$

Схема, реализующая среднюю строку (2.6), показана на рис. 2.11, *а*. Если используется серия, в которой отсутствуют элементы ИЛИ-НЕ, их можно заменить элементами И-ИЛИ-НЕ, как показано на рис. 2.11, *б*. Для серий, в ко-

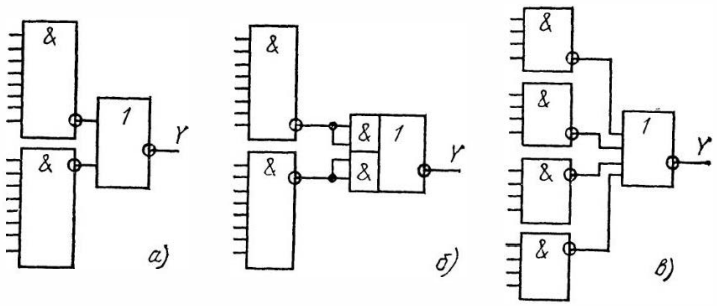


Рис. 2.11. Варианты построения многовходового И

торых отсутствуют восьмивходовые И-НЕ, массив из 16 входов нужно разбить на четырехвходовые части [нижняя строка (2.6)]. Схема для этого случая показана на рис. 2.11, *в*. Ее параметры: $T = 2\tau$, $W \approx 2{,}5$ корпуса. Варианты по рис. 2.11, *б* и *в* требуют несколько бо́льших аппаратурных затрат, чем схема по рис. 2.11, *а*. Материал данного примера иллюстрирует еще одно эвристическое правило: по-возможности выбирать разбиение, использующее больше многовходовых элементов. Это уменьшает общую сумму выводов корпусов, а следовательно, и оборудование.

Многовходовые И используются, например, для обнаружения нулевого значения содержимого регистра. При этом на схему И подаются инверсные выходы всех триггеров регистра.

Построение многовходовых ИЛИ

Упражнение. Построить схему ИЛИ на 16 входов на элементах серии К155.

Решение. После рассмотрения предыдущего примера задача становится типовой:

$$Y = x_0 \vee x_1 \vee x_2 \vee \ldots \vee x_{14} \vee x_{15} =$$

$$= x_0 \vee x_1 \vee x_2 \vee x_3 \cdot x_4 \vee x_5 \vee x_6 \vee x_7 \cdot x_8 \vee x_9 \vee x_{10} \vee x_{11} \cdot$$

$$\cdot x_{12} \vee x_{13} \vee x_{14} \vee x_{15}. \qquad (2.7)$$

Схема, реализующая (2.7), по начертанию аналогична схеме, показанной на рис. 2.11, *в*, но вместо элементов И-НЕ стоят ИЛИ-НЕ, и наоборот. Как и в варианте по рис. 2.11, *б*, роль элементов ИЛИ-НЕ могут играть элементы И-ИЛИ-НЕ.

Многовходовое ИЛИ используется, например, когда нужно обнаружить присутствие хотя бы одной единицы в регистре.

Построение многовходовых сумматоров по модулю 2.

Упражнение. Построить схему сложения по модулю 2 на восемь входов, используя двухвходовые сумматоры по модулю 2.

Подсказка. В аналитической форме требуемая функция представлена следующим образом:

$$Y = a \oplus b \oplus c \oplus d \oplus e \oplus f \oplus g \oplus h. \qquad (2.8)$$

Операция сложения по модулю 2, так же как и операции И и ИЛИ, подчиняется сочетательному закону

$$(x_1 \oplus x_2) \oplus x_3 = x_1 \oplus (x_2 \oplus x_3). \qquad (2.9)$$

Решение. Выражение (2.9) позволяет использовать двухвходовые элементы для представления (2.8) двумя способами:

$$Y1 = ((\ldots ((a \oplus b) \oplus c) \oplus d) \ldots) \oplus h; \qquad (2.10)$$

$$Y2 = [(a \oplus b) \oplus (c + d)] \oplus [(e \oplus f) \oplus (g \oplus h)]. \qquad (2.11)$$

Схемы реализации (2.10) и (2.11) приведены на рис. 2.12, *а* и *б* соответственно. Обе схемы эквивалентны по аппаратурным затратам, которые в общем случае для K входов составляют $K-1$ элементов. В то же время по задержке схемы отличаются заметно, и тем сильнее, чем больше K:

$$T1 = \tau(K-1); \quad T2 = \tau \log_2 K.$$

Если используются элементы не двухвходовые, а n-входовые, то логарифм берется по основанию n.

На материале этого упражнения можно сформулировать

еще одно полезное правило: если для некоторой операции справедлив сочетательный закон, то наилучшей конфигурацией для объединения маловходовых элементов в многовходовую схему будет пирамидальная. В пирамидальной структуре обработке подвергаются одновременно все операнды, т. е. операции в такой схеме распараллелены в наибольшей степени. Поэтому пирамидальная структура имеет минимально возможную задержку.

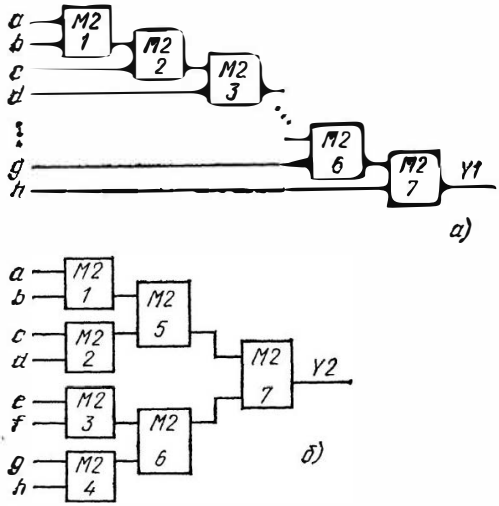


Рис. 2.12. Многовходовый сумматор по модулю 2, построенный по линейной *(а)* и пирамидальной *(б)* схемам

Способы построения многовходовых узлов М2 из элементов И-ИЛИ-НЕ и И-НЕ, имеющих минимально возможную задержку и притом достаточно экономичных по затратам оборудования, приведены в [18].

Знание типовых способов увеличения коэффициента разветвления по выходу и реализации многовходовых логических функций позволяет на первом этапе разработки функциональных схем игнорировать ограничения используемой серии на логические возможности элементов, сосредоточив все внимание на более сложных вопросах. При втором проходе, когда принципиальные проблемы проектируемой схемы уже решены, многовходовые блоки реализуются на элементах заданной серии стандартными приемами.

2.8. Узлы мажоритарного контроля

Одним из приемов повышения достоверности данных на выходе цифровых устройств является *троирование аппаратуры.* При этом устанавливаются три одинаковых цифровых блока, на каждый из которых заводятся все входные величины. Выходные сигналы всех трех блоков поступают на специальный узел *мажорирования (мажоритарный элемент, кворум-элемент),* который формирует из них выходной сигнал по принципу голосования «два из трех». Очевидно, что в случае отказа какого-либо одного из трех блоков сигнал на выходе мажоритарного элемента все равно останется верным. Если блок имеет несколько выходов, то мажорируется каждый из них.

Упражнение. Построить схему узла мажорирования «2 из 3», восстанавливающего информацию при отказе одного из трех каналов. Построить узел *AER*, обнаруживающий этот отказавший канал (от *A* — *address* — адрес, *ER* — *error* — ошибка).

Подсказка. Пример иллюстрирует типичный случай неформализованной постановки задачи, т. е. не в виде таблицы или уравнения, а на обычном русском языке. Суть задачи интуитивно ясна, но как решать ее, не всегда сразу очевидно. Проблема заключается в переходе от интуитивного понимания задачи к формальному ее описанию. В качестве приема можно рекомендовать поэтапную переформулировку задачи с повышением на каждом этапе степени ее формализованности. Полезно начертить схему будущего узла в виде черного ящика — пустого прямоугольника, поскольку о его содержании пока ничего не известно, но со всеми входами и выходами — так, как они понимаются на данный момент. Такая схема хорошо помогает на самом первом этапе ее решения, способствуя осознанию назначения каждой связи, что заметно проясняет задачу. Если построить схему с первой попытки не удалось, попытайтесь сделать это сейчас, используя полученные указания.

Решение. Чертеж первого этапа работы представлен на рис. 2.13. Узел мажорирования обозначается «$\geqslant 2$». *M* — выход восстановленной информации. Схема *AER* должна указывать номер неисправного канала, для чего она должна иметь как минимум два выхода. В соответствии с ЕСКД их можно обозначить *LSB* (*least significant bit*) и *MSB* (*most significant bit*) — соответственно младший и старший разряды номера отказавшего канала.

Из текста задания и рисунка следует, что выход M схемы «$\geqslant 2$» должен совпадать со значениями входов, если они все одинаковы (все каналы исправны), и с двумя одинаковыми, если третий вход отличается от двух других. По такому, теперь уже значительно более формализованному

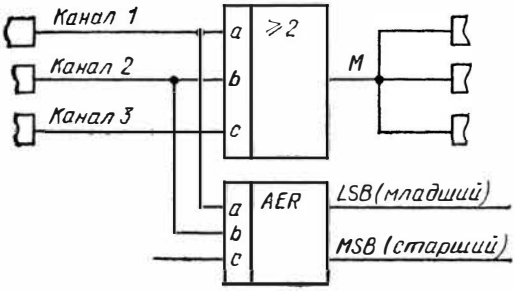


Рис. 2.13. Блоки системы мажоритарного контроля

заданию можно начать строить таблицу истинности искомой схемы, поочередно рассматривая каждую комбинацию аргументов. Значения выхода M схемы приведены в столбце M табл. 2.3.

Таблица 2.3

Входы			Выходы			Входы			Выходы		
a	*b*	*c*	*M*	*MSB*	*LSB*	*a*	*b*	*c*	*M*	*MSB*	*LSB*
0	0	0	0	0	0	1	0	0	0	0	1
0	0	1	0	1	1	1	0	1	1	1	0
0	1	0	0	1	0	1	1	0	1	1	1
0	1	1	1	0	1	1	1	1	1	0	0

Задание на построение схемы AER разработчик может формализовать для себя, например, таким образом: на двух выходах схемы AER должен формироваться двоичный номер того канала, значение информации в котором отличается от двух других. Если во всех каналах информация одинакова, то выходам AER можно назначить код 00, который можно рассматривать как признак исправности всех каналов. Функция номера отказавшего канала приведена в столбцах MSB и LSB (табл. 2.3).

Таблица 2.3 характерна тем, что она описывает состо-

яния не одного, а нескольких выходов, т. е. в отличие от рассматривавшихся ранее таблиц это таблица *системы логических функций*. Подробнее о системах функций будет говориться в § 3.4, а пока можно считать, что таблицей заданы три независимые функции одних и тех же аргументов.

Запись СНДФ и ее минимизация (например, используя геометрическое представление функции) выполняются в соответствии с § 1.3, 1.4. Окончательные аналитические выражения для искомых функций легко реализуются в базисе любой логической серии:

$$M = ab \vee ac \vee bc; \tag{2.12}$$

$$MSB = b \oplus c; \quad LSB = a \oplus c. \tag{2.13}$$

В ряде серий функциональные узлы мажорирования выпускаются в виде микросхем, например, К561ЛП13, 564ЛП13. В реальных системах мажоритарного контроля в отличие от схемы на рис. 2.13 обычно применяют три узла мажорирования. Своими входами они подключены к трем входным каналам параллельно, а их выходы образуют три выходных канала. Такое включение защищает систему от одиночной ошибки не только во входном канале, но и в самом узле мажорирования. Подробнее о мажоритарном контроле см. [29].

2.9. Компаратор

Компаратором называют функциональный узел сравнения чисел. Простейшие компараторы формируют на выходе однобитовый сигнал равенства — 1 или неравенства — 0 двух чисел. Более сложные компараторы, которые будут рассмотрены в § 4.5, в случае неравенства определяют, которое из чисел больше.

Упражнение. Построить на россыпи серии К155 схему сравнения на равенство двух 8-разрядных чисел.

Подсказка. Особенность задачи в том, что для ее решения практически невозможно построить таблицу: число строк в ней должно быть $2^{(8+8)}=65536$. Подобные задачи нужно решать по-иному — методом *блочного конструирования*, т. е. методом *декомпозиции задачи* — разбиения ее на более мелкие *подзадачи* (см. § 1.5). Все схемы с числом входов более 5—7 имеют уже необозримые таблицы, поэтому почти всегда строятся таким способом. Разработчик ищет возможность представить заданную функцию (и бу-

дущую схему) как совокупность более простых функций, схем, т. е. более простых фрагментов. Число фрагментов должно быть невелико, так чтобы схема связей между ними — *структурная схема* — была обозримой. Каждый фрагмент, будучи существенно меньше исходной задачи, также оказывается обозримым. Если какой-то фрагмент оказался слишком сложным, он снова представляется в виде уже своей структуры, и т. д. Только таким путем удается строить цифровые схемы устройств, имеющих сотни и тысячи элементов и десятки входов и выходов.

Простейшими и наиболее распространенными алгоритмами обработки многоразрядных чисел являются такие алгоритмы, которые над каждым разрядом числа или пары чисел производят одну и ту же операцию. Поэтому, полу-

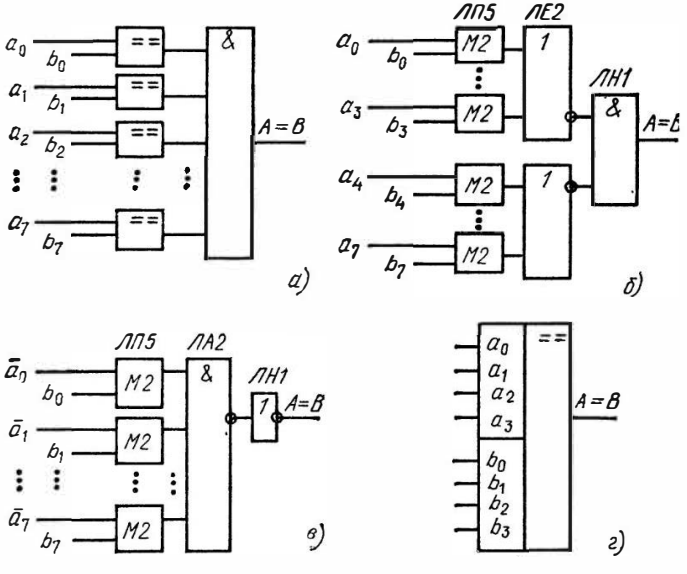


Рис. 2.14. Узел сравнения на равенство

чив задание на построение многоразрядной схемы, в первую очередь нужно проверить: не обладает ли заданный для схемной реализации алгоритм этим удобным свойством, или нельзя ли его свести к таковому.

Р е ш е н и е. В предложенной задаче алгоритм как раз обладает указанным свойством: два числа равны, если попарно равны все их одноименные разряды. Выявленная воз-

можность разбить алгоритм на части позволяет построить структурную схему будущего узла, которая показана на рис. 2.14, *а*.

При построении схем сравнения часто встречается ошибка: предпринимается попытка использовать для выявления равенства одноименных разрядов a_i и b_i элемент И. Это неверно, поскольку нужно выявить не только равенство единиц, но и равенство нулей. Как отмечалось в § 1.7, функция равенства двух аргументов — это инверсия их суммы по модулю 2, и за основу нужно брать элемент К155ЛП5 (см. табл. 2.1).

Поскольку ЛП5 вырабатывает инверсию требуемой функции равенства, то, чтобы исключить из проектируемой схемы восемь инверторов, рационально использовать двойственную форму выходной схемы И

$$\&_{i=0}^{7}(a_i \equiv b_i) = \overline{\bigvee_{i=0}^{7}\overline{(a_i \equiv b_i)}} = \bigvee_{i=0}^{7}(a_i \oplus b_i).$$

Это решение показано на рис. 2.14, *б*. Оно учитывает, что в серии К155 нет элемента ИЛИ-НЕ на восемь входов, а есть только на четыре.

В § 1.8 отмечалось, что проинвертировать функцию М2, т. е. превратить функцию М2 в функцию тождества, можно, проинвертировав один из аргументов. Если одно из чисел уже существует в инверсной форме, то можно воспользоваться схемой, показанной на рис. 2.14, *в*.

Упражнение. Проиллюстрировать сказанное выше цепочкой соответствующих преобразований алгебры логики. Это будет отражать естественный ход проектирования сложной схемы: сначала выработка некоторых гипотез о требуемых преобразованиях и конфигурации будущей схемы. Гипотезы выражаются в словесной форме. Затем следует строгое их доказательство с помощью формальных преобразований алгебры логики.

Условное изображение узла сравнения на равенство показано на рис. 2.14, *г*. Схемы сравнения на равенство используются при поиске устройств по их номеру, опознании заданных слов и т. п.

2.10. Преобразователи кода Грея

Код Грея, отраженный двоичный код, рефлексный (от *reflect*—отражать) *код* для первых восьми чисел представлен в табл. 2.4. В последовательности чисел кода Грея все разряды, кроме самого левого, подчиняются следующему

правилу: любая сплошная группа разрядов, считая справа, по некоторому закону перебирает все свои возможные комбинации, а затем начинает перебирать их в обратном порядке. Так, разряды кода Грея *b* и *c* сначала перебирают комбинации 00, 01, 11, 10, затем — те же комбинации в обратном порядке: 10, 11, 01, 00. Закон справедлив вплоть до группы в один разряд: разряд *c* изменяется сначала в одну сторону — 0,1, затем обратно — 1,0 и снова в том же цикле. Этот закон объясняет одно из названий кода «отраженный» и позволяет строить последовательность чисел кода любой разрядности. Код Грея — *не позиционный*, т. е. веса его разрядов не определяются занимаемыми ими местами, как в обычном двоичном коде, который относится к классу *позиционных*.

Упражнение. Построить таблицы кода Грея для четырех и пяти разрядов. Как проверить ответ, будет сказано в конце параграфа.

В коде Грея при переходе между любыми соседними числами изменяется значение всегда только одного разряда. Благодаря этому свойству код широко применяется в преобразователях углового положения вала в цифровой код, построенных на базе кодового диска или кодового барабана. В оптическом *кодовом диске* единицы и нули кодируются прозрачными и непрозрачными областями. На рис. 2.15, *а* показан трехразрядный диск, углы поворота которого закодированы обычным позиционным двоичным кодом. Диск просвечивается ориентирован-

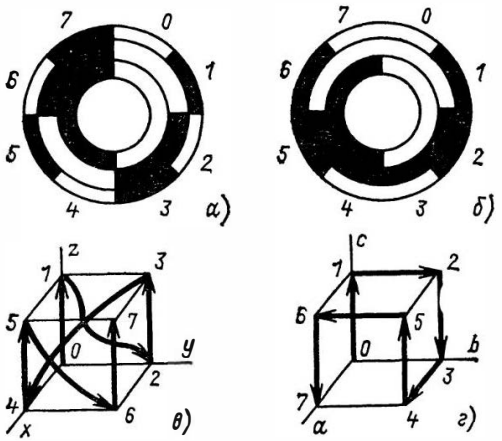


Рис. 2.15. Позиционный код и код Грея:
а — кодовый диск с позиционным кодом; *б* — диск с кодом Грея; *в* — обход куба в позиционном коде; *г* — обход куба в коде Грея

ной вдоль его радиуса световой щелью. По другую сторону диска, в полосе света, размещаются три фотодиода-приемника. В таком устройстве двоичный код, считанный с фотодиодов, будет отображать угол поворота диска. Подробнее о преобразователях угла в код см. [19].

Недостаток кодирования угла поворота позиционным двоичным кодом заключается в том, что при углах поворота, когда нечетный код сменяется четным, коды, считанные с фотодиодов, могут оказаться неверными. Поскольку характеристики фотодиодов не идентичны и юстировка их вдоль щели не идеальна, то при вызванной поворотом вала смене сразу нескольких разрядов кода выходные уровни фотодиодов будут изменяться не строго одновременно. Реальная очередность их переключения может быть любой, и в узком промежутке углов между, например, 011 и 100 можно получить любые значения выходных кодов от 000 до 111. Это так называемая ошибка неоднозначности. Код Грея, поскольку в нем соседние числа отличаются всегда значением только одного разряда, ни в одном переходе не дает ошибки большей, чем значение минимального шага дискретности, т. е. угла, оцифрованного какой-то одной кодовой комбинацией. Диск, угловые секторы которого закодированы 3-разрядным кодом Грея, показан на рис. 2.15, *б*. В реальных преобразователях используются кодовые диски с числом разрядов 8—12 и более.

На рис. 2.15, *г* показана последовательность обхода всех комбинаций значений трех двоичных переменных, использованная в коде Грея. Из сравнения рис. 2.15, *в* и *г* видно, что эта последовательность отличается от обхода тех же комбинаций в позиционном двоичном коде. Очевидно, что любой обход куба (или многомерного гиперкуба), выполненный строго по его ребрам, породит код, у которого разница между соседними значениями также будет всего в одном разряде, т. е. кодов с таким свойством существует много. Заслуга Грея в том, что среди этого множества он выбрал код с очень простым алгоритмом перевода его в двоичный позиционный: каждый i-й, считая с левого, старшего, разряд двоичного позиционного кода любого числа равен сумме по модулю два i-го и всех более левых разрядов того же числа, представленного кодом Грея. Самые левые разряды обоих кодов совпадают, что является частным случаем того же алгоритма, если считать, что еще левее стоят нули. В справедливости алгоритма легко убедиться по табл. 2.4.

Упражнение. Построить схему преобразователя кода Грея в позиционный, пользуясь приведенным словесным описанием алгоритма.

Таблица 2.4

Десятичный код	Двоичный позиционный код x y z	Код Грея a b c	Десятичный код	Двоичный позиционный код x y z	Код Грея a b c
0	0 0 0	0 0 0	4	1 0 0	1 1 0
1	0 0 1	0 0 1	5	1 0 1	1 1 1
2	0 1 0	0 1 1	6	1 1 0	1 0 1
3	0 1 1	0 1 0	7	1 1 1	1 0 0

Решение. Два варианта схем преобразования кода Грея в позиционный показаны на рис. 2.16. Первый (параллельная схема, рис. 2.16, *а*) имеет меньшую задержку, но дороже по затратам оборудования, второй (последовательная схема, рис. 2.16, *б*) — с большей задержкой, но экономичнее.

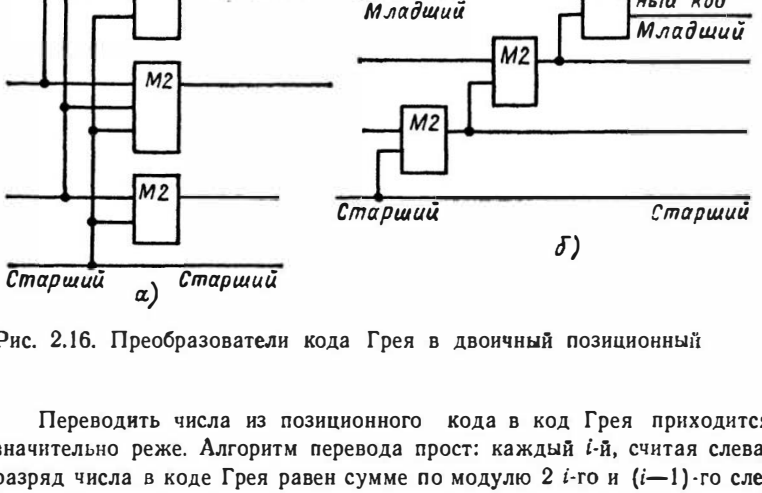


Рис. 2.16. Преобразователи кода Грея в двоичный позиционный

Переводить числа из позиционного кода в код Грея приходится значительно реже. Алгоритм перевода прост: каждый i-й, считая слева, разряд числа в коде Грея равен сумме по модулю 2 i-го и $(i-1)$-го слева разрядов того же числа, представленного позиционным кодом. Алгоритм можно использовать для проверки правильности самостоятельно построенной таблицы кода Грея длиной в 16 и 32 строки.

Упражнение. Построить схему преобразования двоичного позиционного кода в код Грея.

2.11. Узел свертки по четности

Операция сложения по модулю 2, кроме рассмотренных выше узлов, широко используется почти во всех методах контроля как передачи данных, так и работы схем, поэтому в составе многих серий выпускаются микросхемы, реализующие многовходовые функции М2. Нетрудно убедиться, что выход многовходовой функции М2 равен 1, если число единичных значений ее аргументов нечетное, и 0, если оно четное. Поэтому многовходовую операцию М2 над разрядами некоторого двоичного слова называют *сверткой* слова *по нечетности,* а инверсию этого результата — *сверткой по четности.* Функциональные узлы, выполняющие эти операции, называют узлами свертки по четности, а также в силу их широкого применения в схемах контроля — *узлами контроля по четности (parity generator/checker).*

В качестве примера на рис. 2.17, *а* показано условное изображение восьмивходовой микросхемы контроля по чет-

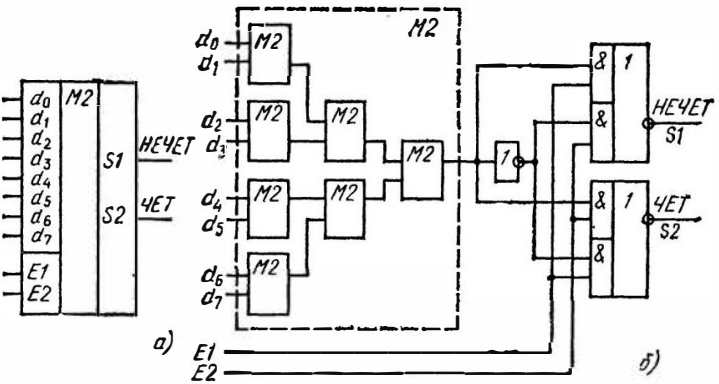


Рис. 2.17. Восьмивходовый узел контроля четности — нечетности К155ИП2

ности и нечетности К155ИП2, а на рис. 2.17, *б* — ее несколько упрощенная функциональная схема. Собственно схема М2 на восемь входов обведена пунктиром и, как это видно, построена по пирамидальному принципу (см. § 2.7).

Функционирование микросхемы задано табл. 2.5. Благодаря объединению в группы тех входных комбинаций, на которые микросхема реагирует одинаково, число строк в таблице всего шесть. Полная таблица истинности для десятивходовой схемы К155ИП2 имела бы 1024 строки.

Таблица 2.5

№	Входы			Выходы	
	данных: сумма единиц на входах d_0—d_7	управляющие			
		чет *E2*	нечет *E1*	чет *S2*	нечет *S1*
1	четная	1	0	1	0
2	нечетная	1	0	0	1
3	четная	0	1	0	1
4	нечетная	0	1	1	0
5	×	1	1	0	0
6	×	0	0	1	1

Примечание. × — безразличное состояние.

Упражнение. По схеме на рис. 2.17, *б* построить самостоятельно краткую форму ее таблицы и результат сравнить с табл. 2.5.

Элементы И-ИЛИ-НЕ, показанные на рис. 2.17, *б*, коммутируют выработанную пирамидой сумму по модулю 2 на два выхода — *S2* и *S1* в соответствии с управляющими сигналами *E1* и *E2*. Как видно из табл. 2.5, обе одноименные комбинации значений *E2* и *E1* запирают схему: при *E2E1*= =00 *S2S1*=11, а при *E2E1*=11 *S2S1*=00 независимо от значений входов данных d_0—d_7 (строки 6 и 5 табл. 2.5).

Две разноименные комбинации *E2* и *E1* устанавливают на выходах значения: одна — четности, а другая — нечетности суммы единиц, поданных на входы d_0—d_7. При этом если на одном выходе — *S1* или *S2* уровень равен 1, то на другом он равен 0 и наоборот. Такое представление информации сразу двумя разноименными уровнями сигналов называется *парафазным представлением* или *парафазным кодом*. Строки 1 и 2 таблицы говорят о том, что при *E2E1*= =10 значение четности числа входных единиц передается на выходы без изменения, а при *E2E1*=01 значение четности инвертируется.

Входы *E1* и *E2* могут использоваться различными способами. На рис. 2.18, *а* показан один из способов управления микросхемой. На вход *E2* задается константа 1 теми же средствами, что и на неиспользуемые входы элементов И-НЕ, и тогда вход *E1* играет роль запрещающего входа. При сигнале на входе *E1*, равном 0, схема свертки работает как обычно. При подаче на вход *E1* высокого уровня (сигнала *ЗАПРЕТ*) на обоих выходах микросхемы устанавливаются нули независимо от значений входов d_0—d_7.

На рис. 2.18, *б* показан способ построения девятого входа данных, который требуется при организации контроля однобайтных (8-разрядных) слов по четности.

На рис. 2.18, *в* показано использование входов *E1* и *E2* для наращивания схемы свертки. *Наращиванием* или *расширением* называют

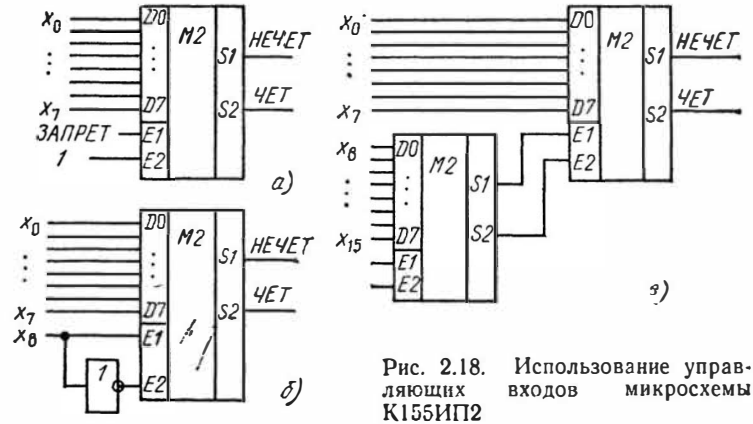


Рис. 2.18. Использование управляющих входов микросхемы К155ИП2

включение нескольких малоразрядных микросхем таким образом, чтобы получить функциональный узел большей разрядности. Большинство микросхем имеет специальные входы и выходы для наращивания. На рис. 2.18, *в* показан последовательный способ наращивания, при этом задержки всех микросхем суммируются. При большом числе входов целесообразнее микросхемы включать по пирамидальной схеме, аналогичной обведенной пунктиром на рис. 2.17, *б*.

Задержка микросхемы К155ИП2 характеризуется несколькими числами, поскольку она содержит несколько трактов распространения сигнала от входов к выходам.

Упражнение. Просмотрев еще раз § 2.2, выделить в схеме рис. 2.17, *б* тракты распространения сигнала, имеющие различные задержки, и составить список значений задержек, характеризующих рассматриваемую микросхему.

Решение. С точки зрения оценки задержки микросхема К155ИП2 имеет три характерные группы входов: группу идентичных входов данных d_0—d_7, вход *E1* и вход *E2*. Два последних входа нельзя объединять в общую группу, поскольку значение сигнала на каждом из них самостоятельно влияет на значение задержки тракта вход данных — выход, включая в этот тракт инвертор или шунтируя

его обходным проводом. Для исчерпывающего описания временных характеристик достаточно указать задержки по пяти трактам:

T1 — от входов данных до выхода *ЧЕТ* при *E1*=0;

T2 — от входов данных до выхода *НЕЧЕТ* при *E1*=0

T3 — от входов данных до выхода *ЧЕТ* при *E2*=0;

T4 — от входов данных до выхода *НЕЧЕТ* при *E2*=0;

T5 — от любого из входов *E1* и *E2* до любого из выходов.

Сигнал от входов данных до каждого из выходов — *ЧЕТ* и *НЕЧЕТ* — проходит по одному из двух трактов в зависимости от того, на который из управляющих входов — *E1* или *E2* подан 0. Поэтому задержка от входа данных до выходов имеет четыре значения — от *T1* до *T4*. Длины трактов от управляющих входов до выходов одинаковы — и для обоих выходов, и для обоих управляющих входов, и к тому же не зависят от кода на входах данных. Это и отражено в строке *T5*.

Для каждого тракта в справочнике может быть указано или среднее время задержки распространения, или задержка положительного $t^{01}_{\text{зд.р}}$ и отрицательного $t^{10}_{\text{зд.р}}$ фронтов по отдельности. Как уже говорилось, подробные справочники содержат весь список задержек, краткие — лишь значение задержки по самому длинному тракту, позволяющее оценить класс быстродействия данного узла.

Упражнение. Определить, какие из величин *T1*—*T5* характеризуют задержки схем рис. 2.18, *а* и *б* от входов данных до выхода *ЧЕТ*.

Решение. В схеме рис. 2.18, *а* значение *E2* никогда не бывает равным 0. Значение *E1* в режиме определения четности равно 0 (запрет отсутствует). Следовательно, задержка схемы в этом режиме $T_а = T3$.

В схеме на рис. 2.18, *б* значения *E1* и *E2* в процессе работы могут быть любыми, поскольку x_8 — это входные данные. Поэтому для оценки задержки от входов d_0—d_7 до выхода *ЧЕТ* нужно взять наибольшую из величин *T1* и *T3*. Задержка от x_8 до выхода *ЧЕТ* равна сумме задержки инвертора и величины *T5*. Следовательно, время, спустя которое на выходе *ЧЕТ* наверняка сформируется результат свертки по четности x_1—x_8, будет

$$T_б = \max [T1,\ T3,\ (T_{\text{инвертора}} + T5)].$$

Упражнение. Оценить максимальное время получения свертки 16-разрядного числа с помощью схемы, показанной на рис. 2.18, *в*, если результат снимать с выхода *ЧЕТ*, а входы первого каскада *E1*, *E2* включить по схеме рис. 2.18, *а*.

Ответ:

$$\max\{[\max(T1,\ T3)],\ \underbrace{[\max(T1,\ T2)+T5]}_{\text{тракт } x_8 - x_{15}}\}\ .\ .$$
$$\underbrace{\qquad\qquad}_{\text{тракт } x_0 - x_7}$$

Принципы выделения *критических путей,* т. е. трактов, определяющих задержку всего узла, оценки значений их задержки, выявление взаимного влияния различных трактов, использование этих данных для оценки задержки схемы при различных способах ее применения остаются такими же для всех узлов, которые будут рассматриваться в дальнейшем.

ГЛАВА 3

КОДИРУЮЩИЕ УСТРОЙСТВА

3.1. Дешифраторы

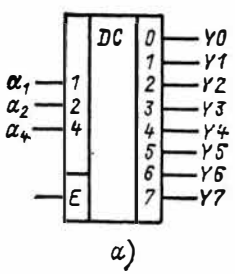
Кодирующим устройством называют логический узел, преобразующий многоразрядный входной код в выходной код, построенный по иному закону. Название в большой мере условно, поскольку любое цифровое устройство преобразует некоторый входной код в некоторый выходной, т. е. является кодовым преобразователем. Традиционно это название применяется к узлам, работа которых не описывается достаточно простым алгоритмом, как, например, работа сумматора, а задается таблицей соответствия входов и выходов. В таком смысле термин и будет применяться в дальнейшем.

Дешифратором или *декодером (decoder)* чаще всего называют кодирующее устройство, преобразующее двоичный код в унарный. Из всех m выходов дешифратора активный уровень имеется только на одном, а именно на том, номер которого равен поданному на вход двоичному числу. На всех остальных выходах дешифратора уровни напряжения неактивные. Условное изображение дешифратора на схемах показано на рис. 3.1, *а*. О разрешающем входе E будет сказано ниже.

Если декодер имеет n входов, m выходов и использует все возможные наборы входных переменных, то $m=2^n$. Такой декодер называют полным в отличие от неполного, использующего лишь часть возможных наборов и имеюще-

го соответственно меньшее число выходов и внутренних схемных элементов.

Декодер используют, когда нужно обращаться к различным цифровым устройствам, и при этом номер устройства — его *адрес* — представлен двоичным кодом. Входы декодера (их иногда называют *адресными входами)* часто-



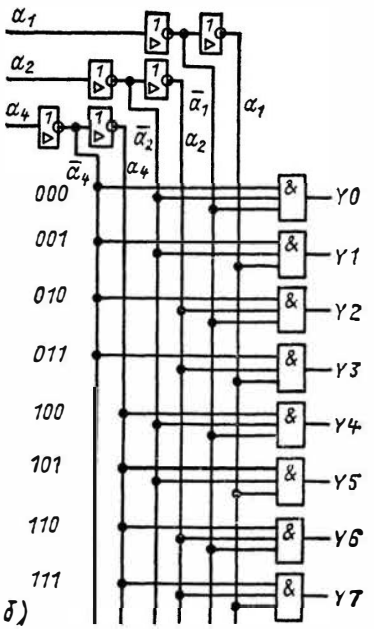


Рис. 3.1. Дешифратор 3—8

нумеруют не порядковыми номерами, а в соответствии с весами двоичных разрядов, т. е. не 1, 2, 3, 4, 5, ... , а 1, 2, 4, 8, 16 ... Число входов и выходов декодера указывают таким образом: декодер 3—8 (читается «три в восемь»); 4—16; 4—10 (это неполный декодер).

Формально описать работу дешифратора можно, задав список функций, отрабатываемых каждым из его выходов Y_i. Так, для дешифратора 3—8 $Y_0 = \bar{a}_4\bar{a}_2\bar{a}_1$; $Y_1 = \bar{a}_4\bar{a}_2 a_1$; $Y_2 = \bar{a}_4 a_2\bar{a}_1$; ... ; $Y_6 = a_4 a_2\bar{a}_1$; $Y_7 = a_4 a_2 a_1$. Реализация этих восьми выражений с помощью восьми трехвходовых элементов И дает наиболее простой по структуре дешифратор, называемый *линейным* (рис. 3.1, *б*). Основной объем его оборудования составляют в общем случае m n-входовых элемен-

тов И. Кроме того, к оборудованию дешифратора обычно относят n инверторов входных переменных и n буферных входных усилителей, обычно инвертирующих, что характерно для интегральной технологии. Назначение буферных усилителей — свести к единице кратность нагрузки, которую представляет дешифратор для источника сигнала. Иначе каждый источник сигнала, как следует из рис. 3.1, *б*, будет нагружен весьма существенно — на $m/2$ входов элементов И.

Если дешифратор состоит из элементов И-НЕ, то на его выходах будут отрабатываться не сами функции Y_i, а их инверсии, т. е. активным уровнем выхода будет низкий. Ликвидировать инверсии на выходах можно или подключив инверторы, или построив дешифратор по двойственной схеме — на элементах ИЛИ-НЕ. Число входных инверторов и буферных усилителей при этом не изменится.

Дешифраторы часто имеют *разрешающий* (управляющий, стробирующий) вход E (от *enable* — давать возможность). При $E=1$ дешифратор работает как обычно, при $E=0$ на всех выходах устанавливаются неактивные уровни независимо от поступившего кода адреса. Вход E часто выполняют инверсным. Дешифратор, имеющий разрешающий вход, иногда называют *декодер-демультиплексор* и на условном обозначении вместо символа DC используют символ DX. Откуда появилось это название, будет сказано в следующем параграфе.

На рис. 3.2, *а* показан вариант построения разрешающего входа, когда сигнал E воздействует непосредственно на

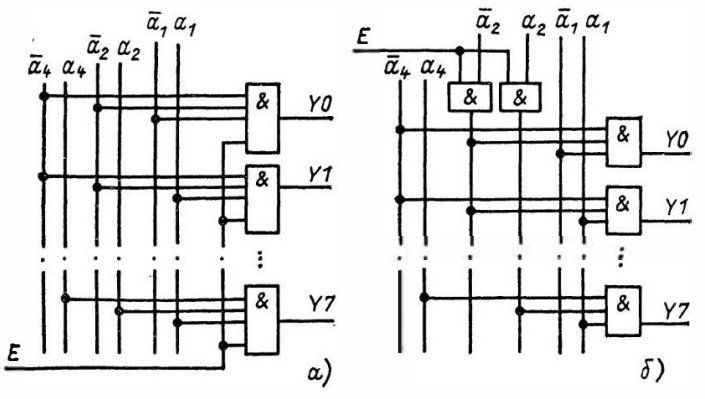


Рис. 3.2. Разрешающий вход дешифратора:
а — управление дешифрирующими элементами; *б* — управление по одной из переменных

все дешифрирующие элементы. Этот вариант требует увеличения на единицу числа входов дешифрирующих элементов, но не вносит дополнительной задержки. На рис. 3.2, *б* показан другой вариант, основанный на том, что, как видно из рис. 3.1, *б*, в дешифраторе не найдется ни одного дешифрирующего элемента, к которому любая переменная не была бы подключена или в своей прямой, или в инверсной форме. Поэтому если и в прямой, и в инверсный тракты л ю б о й входной переменной поставить элементы И и завести на них сигналы E, то при $E=0$ будут заперты абсолютно все конъюнкторы, подключенные к выходам. На рис. 3.2, *б* показано воздействие сигнала E на среднюю переменную a_2, чтобы не создавалось ложного впечатления, что запирать дешифратор надо обязательно или по старшей, или по младшей переменной. Способ управления по одной из входных переменных экономичен по оборудованию, но увеличивает задержку дешифратора.

На рис. 3.3 показана группа из пяти дешифраторов, соединенных в два *каскада*. Вся группа работает как де-

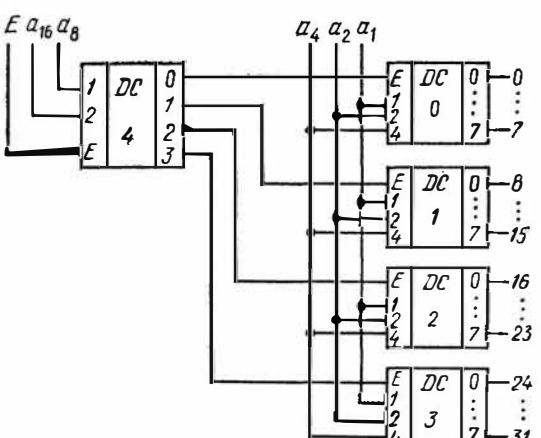


Рис. 3.3. Каскадное соединение дешифраторов

шифратор 5—32. Два старших разряда адреса a_{16} и a_8 расшифровываются дешифратором 2—4 *DC4*, который по входам E управляет четырьмя дешифраторами 3—8 второго каскада. Младшие разряды адреса a_4, a_2, a_1 поступают на все дешифраторы второго каскада, но открытым по вхо-

ду E оказывается лишь один из них. Ему и будет принадлежать единственный из всех 32 возбужденный выход. Так, при поступлении кода $a_{16}a_8a_4a_2a_1 = 01111$ у *DC4* сигнал появится на выходе *1*, и по входу E будет открыт *DC1*. Остальные дешифраторы второго каскада будут заперты. Разряды адреса $a_4a_2a_1 = 111$ вызовут появление 1 на выходе *7* дешифратора *DC1*, т. е. на выходе *15* всего составного дешифратора, что соответствует заданному адресу. Принцип используется при построении дешифраторов на много выходов из микросхем дешифраторов с меньшим числом выходов.

В рассмотренном примере 5-разрядный адрес был разбит на две группы в 2 и 3 разряда, и это определило структуру всей схемы. В общем случае многоразрядный адрес можно разбить на группы различными способами, и каждому способу будет соответствовать свой вариант схемы многокаскадного (не обязательно двухкаскадного) дешифратора. Варианты будут отличаться задержкой и аппаратурными затратами, и можно ставить задачу выбора оптимальной в заданной серии элементов структуры.

В предельном случае при числе каскадов, равном разрядности адреса, получается *пирамидальный дешифратор*, имеющий максимально возможную задержку. Достоинством его является использование только двухвходовых элементов И, что определило его широкое распространение на заре цифровой техники, и с тех пор он, как сорняк, неистребим в литературе. В современных многовходовых логических базисах пирамидальная схема дешифратора не выдерживает конкуренции с другими структурами даже по оборудованию, не говоря уже о задержке.

На рис. 3.4 показан двухкаскадный дешифратор 4—16, второй каскад которого собран по схеме *прямоугольного дешифратора*. Разряды адреса разбиты на две группы, каждая из которых независимо от другой расшифровывается своим *дешифратором первого каскада* *DC1* и *DC2*. При любой комбинации значений входных переменных оказываются выбранными одна строка и один столбец сетки, в узлах которой расположены элементы И второй ступени. В результате каждый входной набор возбуждает выход единственного соответствующего ему элемента И. Такую сетку из элементов И и называют *прямоугольным* или *матричным* дешифратором.

При использовании во второй ступени элементов И-НЕ выходы дешифратора будут инверсными. Их можно сде-

лать прямыми, построив сетку второго каскада по двойственному варианту, на элементах ИЛИ-НЕ; тогда инверсными должны быть выходы дешифраторов первого каскада.

Делить разряды адреса между *DC1* и *DC2* нужно по возможности поровну: чем ближе прямоугольник второго каскада к квадрату, тем при том же числе выходных эле-

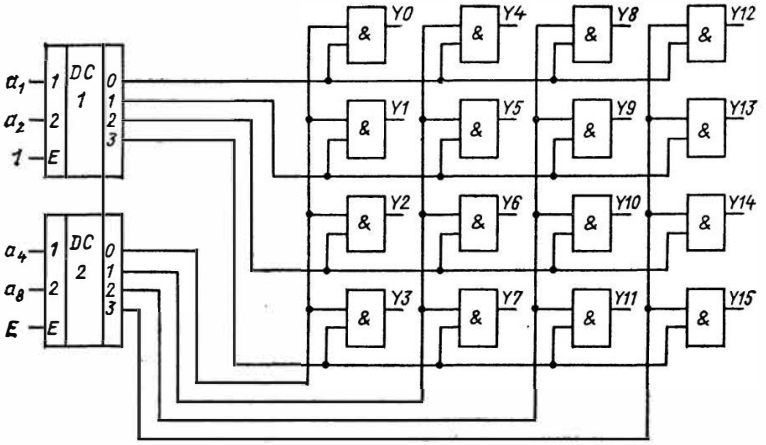


Рис. 3.4. Двухкаскадный дешифратор с прямоугольным (матричным) дешифратором во втором каскаде

ментов И меньше сумма его строк и столбцов, т. е. меньше число выходов дешифраторов первого каскада. В качестве входа E всего двухкаскадного дешифратора удобно использовать разрешающий вход одного из дешифраторов первого каскада. При этом запираются или все строки, или все столбцы.

Целесообразно сравнить три рассмотренных типа дешифраторов по величине задержки и аппаратурным затратам. Задержка минимальна для линейного дешифратора и в пределе может быть равна $(2 \div 3)\,\tau$ в зависимости от числа ступеней инверторов-усилителей. При включении линейных дешифраторов в несколько каскадов задержки всех каскадов складываются. Чем больше число выходов дешифратора, тем большая доля оборудования сосредоточена в элементах И самого последнего каскада. Число элементов предпоследнего каскада уже в несколько раз меньше, чем последнего, а предыдущих — тем более. Число элементов И последнего каскада дешифратора любого типа всегда равно числу его выходов, поэтому в первом приближении аппаратурные

затраты дешифраторов различных типов соотносятся как аппаратурные затраты их элементов И последнего каскада. Они и определяют основную разницу: у линейного дешифратора число входов каждого элемента И последнего (и единственного) каскада равно n (n — число адресных входов), у каскадного оно зависит от способа разбиения на группы, однако оно всегда меньше n, но больше двух, у прямоугольного оно равно двум — минимально возможному числу. Поэтому при большом числе выходов (сотни и более) прямоугольный дешифратор — самый экономичный по оборудованию, чем и объясняется его широкое применение в БИС памяти. При уменьшении числа выходов до нескольких десятков первенство по экономичности переходит к каскадным дешифраторам, а при малом числе выходов самым экономичным (а к тому же и самым быстрым) оказывается линейный дешифратор.

Приведенные оценки справедливы лишь для дешифраторов, построенных непосредственно из логических элементов, например при разработке схемы матричной БИС. При проектировании же блоков из готовых микросхем, когда затраты оборудования оцениваются не числом элементов, а числом корпусов, даже большие дешифраторы экономичнее строить по каскадному принципу, набирая их из микросхем небольших дешифраторов на 8 или 16 выходов.

Дешифраторы, выпускаемые в виде отдельных микросхем, имеют буквенное обозначение ИД. В сериях ТТЛ, в которых элементы И-НЕ наиболее технологичны, дешифраторы обычно имеют инверсные выходы, т. е. активный низкий уровень выходного сигнала. В КМДП-сериях, где элементы ИЛИ-НЕ не менее технологичны, чем И-НЕ, дешифраторы чаще имеют прямые выходы. Стремление возможно полнее использовать выводы типовых корпусов определяет размеры декодеров, выпускаемых в виде СИС. Обычно это 3—8, 4—10, сдвоенный 2—4, а также 4—16, но уже в корпусе с 24 выводами.

Часто в микросхемах дешифраторов делают несколько разрешающих входов, а разрешающей комбинацией является их конъюнкция. При этом удобно наращивать дешифраторы, используя каскадный принцип и строя первый каскад дешифрации не на отдельном специальном дешифраторе, а собирая его из конъюнкторов разрешающих входов. На рис. 3.5, *а* таким способом построен дешифратор 5—32 из четырех микросхем 3—8. Каждая микросхема имеет два инверсных разрешающих входа. Символ «&» над символом E обозначает, что разрешение существует лишь при совпадении всех сигналов группы входов, помеченной знаком «&». На рис. 3.5, *а* символы инверсии указывают на совпадение двух низких уровней на входах разрешения. Де-

шифратор первого каскада распределен по конъюнкторам разрешающих входов четырех микросхем. Такое решение— иметь несколько разрешающих входов, связанных опера-

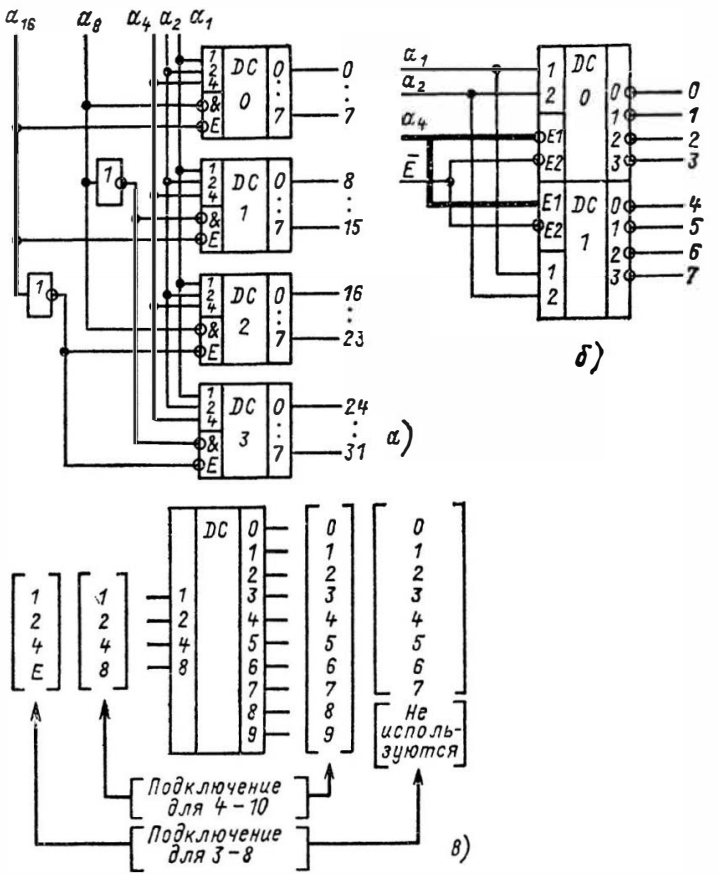


Рис. 3.5. Использование разрешающих входов микросхем декодеров: *а* — на разрешающих входах построена первая ступень каскадного дешифратора; *б*—универсальная микросхема декодеров-демультиплексоров: 2×(2—4) и 1×(3—8); *в* — универсальная микросхема дешифратора 4—10 и декодера-демультиплексора 3—8.

цией И, чтобы собирать на этих входах фрагменты дешифраторов, вообще типично для современных микросхем.

В микросхему по рис. 3.5, *б* входят два декодера-демультиплексора 2—4. Пример такой микросхемы — К155ИД4.

Каждый декодер имеет пару разрешающих входов. Один вход одной из секций инвертирован. Это позволяет, объединив его с неинвертированным разрешающим входом другой секции и подав на эту пару третью переменную a_4, использовать ту же самую микросхему как декодер-демультиплексор 3—8 с разрешающим входом $\overline{E}$.

Наиболее очевидное назначение микросхемы по рис. 3.5, *в* — дешифратор 4—10. Однако, поскольку в четвертом разряде двоично-десятичного кода единица появляется лишь в цифрах 8 и 9, а в цифрах от 0 до 7 четвертый разряд всегда равен нулю, микросхему можно использовать и как декодер-демультиплексор 3—8. При этом вход *8* играет роль разрешающего входа $\overline{E}$, а выходы *8* и *9* не используются. В микроэлектронике очень распространена тенденция выпускать микросхемы, способные выполнять несколько различных функций: это увеличивает спрос на микросхемы, а рост объема выпуска способствует снижению стоимости.

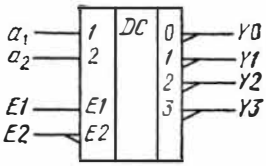


Рис. 3.6. Изображение одной секции микросхемы К155ИД4 с указателями полярности выводов

Микросхемы дешифраторов — характерный пример узлов, в которых входные и выходные сигналы представлены частично в положительной, а частично в отрицательной логике. Поэтому таблицы истинности дешифраторов часто даются не в терминах 1—0, а в терминах уровней *H*—*L*. В табл. 3.1 в этих терминах описано функционирование нижней секции дешифратора, показанного на рис. 3.5, *б*, а на рис. 3.6 приведено условное изображение этой же секции с указателями полярности выводов.

Таблица 3.1

Входы				Выходы			
1	*2*	*E1*	*E2*	*0*	*1*	*2*	*3*
×	×	*L*	×	*H*	*H*	*H*	*H*
×	×	×	*H*	*H*	*H*	*H*	*H*
L	*L*	*H*	*L*	*L*	*H*	*H*	*H*
L	*H*	*H*	*L*	*H*	*L*	*H*	*H*
H	*L*	*H*	*L*	*H*	*H*	*L*	*H*
H	*H*	*H*	*L*	*H*	*H*	*H*	*L*

Примечание. × — безразличное состояние.

Временные характеристики микросхем дешифраторов определяются задержками двух трактов; адресные входы — выход и входы разрешения — выход. Обе эти характеристики приводятся в полных справочниках. Задержка по тракту разрешения почти всегда меньше задержки по тракту адреса. Типовая задержка микросхем дешифратора лежит в пределах $(1 \div 2)\tau$ данной серии элементов.

3.2. Мультиплексоры

Мультиплексор (*multiplexor*) — это функциональный узел, осуществляющий подключение (*коммутацию*) *одного* из нескольких *входов данных* к выходу. Номер выбранного входа соответствует коду, поданному на *адресные входы* мультиплексора. Условное изображение мультиплексора показано на рис. 3.7. Вход E — разрешающий: при $E=1$

Рис. 3.7. Условное изображение мультиплексора

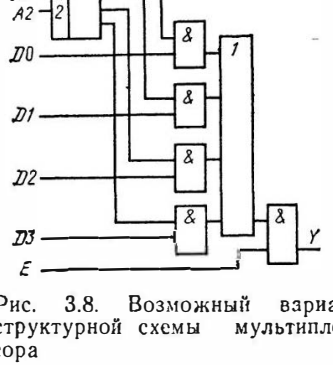


Рис. 3.8. Возможный вариант структурной схемы мультиплексора

мультиплексор работает как обычно, при $E=0$ выход узла находится в неактивном состоянии, мультиплексор заперт.

Упражнение. На элементах И-НЕ построить функциональную схему мультиплексора 4—1 («четыре в один», т. е. коммутирующего данные от четырех источников на одно направление).

Подсказка. Разработку, как всегда, полезно начать с изображения искомой схемы в виде прямоугольника с указанием всех входов и выходов. В качестве такого изображения можно воспользоваться рис. 3.7. Задача синтеза схемы поставлена в неформализованной, понятийной форме, на обычном языке. Попытка формализовать ее

с помощью таблицы истинности обречена на неудачу: число входов узла равно семи, поэтому таблица будет иметь 128 строк. Остается использовать метод декомпозиции задачи — разбиения ее на систему более простых подзадач. Нужно попытаться представить требуемый узел как некоторую структуру из более простых схем, которые в совокупности реализуют требуемую функцию.

Решение. На рис. 3.8 представлен вариант структурной схемы мультиплексора, состоящей из уже известных читателю функциональных узлов. В этой схеме отдельными узлами или элементами представлены все функции, которые должен выполнять мультиплексор: расшифровка адреса декодером, выбор входного канала с помощью одного из конъюнкторов, сборка всех каналов в один на ИЛИ, подключение этого канала к выходу при $E=1$ с помощью элемента И.

Схема, построенная методом разбиения на подзадачи, будет выполнять заданную функцию, однако она скорее всего окажется не оптимальной по задержке и аппаратурным затратам. Поэтому за первым этапом проектирования, позволившим получить решение «в принципе», следует второй этап — этап *оптимизации* полученного решения. В качестве рабочего приема на этом этапе можно рекомендовать постановку вопросов такого типа:

Что оказалось лишним и может быть исключено?

Что можно упростить, модифицировать?

Функции каких элементов можно объединить в одном элементе?

Какие специализированные узлы рационально заменить стандартными?

Модифицируя схему, разработчик еще раз ставит себе эти вопросы. Успех работы сильно зависит от знания функциональных схем узлов, использованных на первом этапе. Не понимания, а именно знания: только это позволяет строить новые блоки с совмещенными функциями. В случае схемы, показанной на рис. 3.8, знание того, что дешифратор — это группа конъюнкторов, наводит на мысль, что конъюнкторы дешифратора попутно могут выполнять и функцию конъюнкторов данных, и функцию выходного разрешающего конъюнктора. В результате этих усовершенствований получаем более компактную и быструю схему (рис. 3.9). С помощью формул де-Моргана ее легко перевести в базис И-НЕ. Это и будет окончательная схема, по которой часто строят реальные мультиплексоры.

Прием проектирования схем (и вообще решения инженерных задач) в два этапа: 1) принципиальное решение, 2) оптимальное решение — полезно освоить и применять возможно чаще, так как почти всегда оказывается, что получить сначала любое, пусть заведомо неоптимальное, но обязательно доведенное до конца решение и затем его оптимизировать намного результативнее, чем пытаться найти сразу оптимальное.

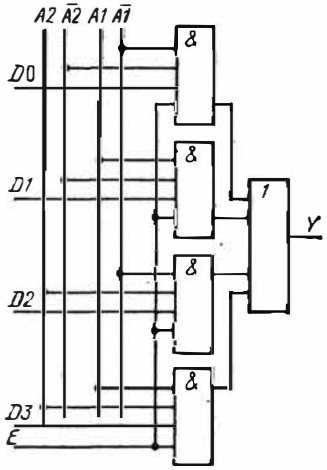


Рис. 3.9. Схема мультиплексора в базисе НЕ, И, ИЛИ

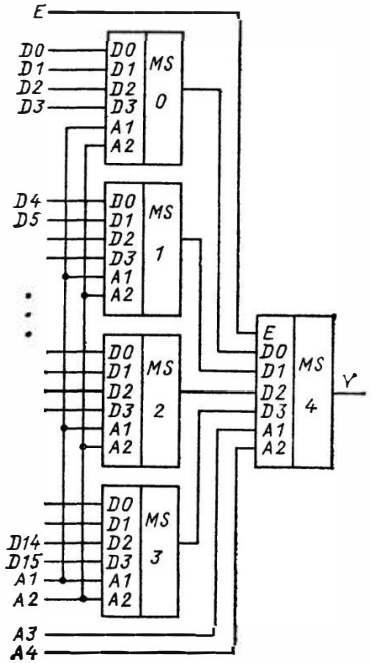


Рис. 3.10. Каскадное соединение мультиплексоров

Мультиплексоры 4—1, 8—1, 16—1 выпускаются в составе многих серий и имеют буквенный код КП. Их временные характеристики задаются задержками по трем трактам: вход адреса — выход, вход данных — выход, вход разрешения — выход. Для большинства серий эти величины лежат в пределах $(1 \div 2)\tau$ своей серии элементов. Универсальный способ наращивания мультиплексоров показан на рис. 3.10.

Все рассмотренные схемы мультиплексоров коммути-

руют только один разряд данных. При коммутации многоразрядных слов в каждом разряде используется свой мультиплексор. В составе ряда серий выпускаются микросхемы многоразрядных мультиплексоров. В одном корпусе размещаются или четыре мультиплексора 2—1 (например, К555КП11), или два мультиплексора 4—1 (например, К555КП12).

Термином «мультиплексирование» называют процесс передачи данных от нескольких источников по общему каналу, и любое устройство, осуществляющее на передающей стороне операцию сведения данных в один канал, принято называть *мультиплексором*. Это название исторически закрепилось за схемой по рис. 3.9, способной осуществлять временно́е мультиплексирование сигналов, передавая их в линию друг за другом в темпе смены кодов на своих адресных входах. Но эта же схема может выполнять и еще одну распространенную операцию — *выбор, селекцию* (от *select* — выбирать) данных из определенного, указанного адресным кодом источника. Любое устройство, выполняющее операцию селекции, называют селектором, и разработчики, у которых схема, показанная на рис. 3.9, выполняла эти функции, естественно стали называть ее *селектором*. Кроме того, поскольку схема выполняет коммутацию сигналов, ее еще называют *коммутатором*. Рассматриваемая схема побила среди цифровых узлов рекорд продолжительности жизни без определенного имени, ее до сих пор называют и мультиплексор, и селектор, и селектор-мультиплексор, и коммутатор. Например, одни и те же микросхемы К155КП5 и К155КП7 в [8] называются коммутаторами, а в [9] — селекторами-мультиплексорами. Еще интереснее: в одной и той же работе [10] микросхемы обсуждаемого типа, различающиеся лишь числом входов, называются то коммутатор (134КП9), то селектор-мультиплексор (К155КП7), то селектор (К155КП11), то мультиплексор (К555КП13). Терминологическая многозначность повлекла многозначность и в условных обозначениях на функциональных схемах, поскольку ЕСКД (см. [5]) требует обозначать мультиплексор символом *MUX*, селектор — *SL*, а мультиплексор-селектор — *MS*. Учитывая, что в случае рассматриваемой схемы все три термина, все их парные комбинации и все три обозначения абсолютно эквивалентны, автор в качестве критерия для выбора термина принял в основном соображение наибольшей распространенности, и отчасти приемлемости физической длины. Выбор пал на термин *мультиплексор* и обозначение *MS*, что и используется в данной книге.

На приемной стороне мультиплексированной магистрали требуется выполнить обратную операцию — демультиплексирование, т. е. распределение порций данных, поступающих в последовательные моменты времени, по своим приемникам. Эту операцию выполняет *демультиплек-*

сор. В роли демультиплексора успешно выступает декодер, если к его разрешающему входу E подключить мультиплексированную магистраль данных, а на адресные входы подавать друг за другом коды адресов приемников. Поэтому декодер, имеющий разрешающий вход E, иногда называют не просто декодер, а *декодер-демультиплексор*, о чем уже говорилось несколько раньше. Это двойное название, как и название селектор-мультиплексор, пришло к нам из-за рубежа, где фирмам-изготовителям по рекламным соображениям выгодно уже в самом названии отразить возможно больше выполняемых микросхемой функций.

Применительно к дешифратору еще иногда можно встретить термины декор-мультиплексор и дешифратор-мультиплексор (см., например, в [8] название микросхемы К155ИД4). Если исключить тривиальную версию о живучем гене опечатки, реплицирующимся из источника в источник, то это может быть терминологическим эхом событий 15—20-летней давности, когда мультиплексором в противоположность современному его значению действительно часто называли функциональный узел, распределяющий данные с одного направления на несколько.

Применение мультиплексоров не ограничивается операциями мультиплексирования и селекции. На рис. 3.11, *а* показан один, i-й разряд схемы *параллельного сдвигателя.*

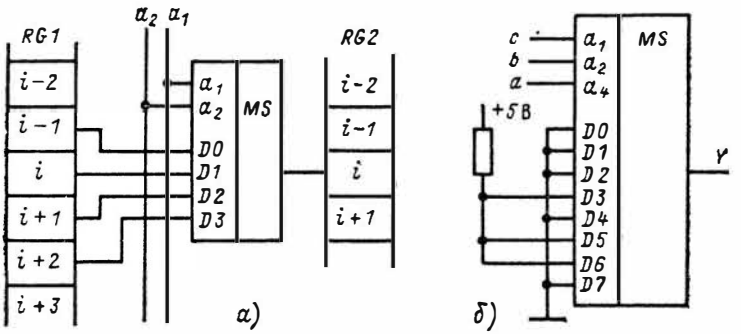


Рис. 3.11. Возможные применения мультиплексоров:
а — один разряд комбинационного сдвигателя; *б* — реализация произвольной функции, заданной таблицей истинности, в данном случае — табл. 3.2

В полной схеме сдвигателя ко входу каждого разряда регистра *RG2* подключено по такому же мультиплексору, входы данных которого в свою очередь подключены к выходам нескольких разрядов регистра *RG1*. На адресные входы мультиплексоров всех разрядов подается один и тот

же код. В результате в зависимости от значения адресного кода в i-й разряд $RG2$ будет переписываться содержимое различных разрядов $RG1$. При адресном коде $a_2a_1=01$, как видно из рисунка, данные будут передаваться в одноименный разряд регистра $RG2$ без сдвига. При коде $a_2a_1=00$ в i-й разряд регистра $RG2$ будет передаваться содержимое соседнего младшего, $(i-1)$-го разряда регистра $RG1$, т. е. передача произойдет со сдвигом на один разряд в сторону старших разрядов (*влево*). При кодах a_2a_1, равных 10 и 11, передаваемое число будет сдвинуто в сторону младших разрядов (*вправо*) на один или два разряда соответственно. Используя мультиплексоры с достаточным числом входов и подключая входы к соответствующим разрядам регистра-источника, можно строить сдвигатели, способные очень быстро, всего за время задержки мультиплексора и регистра-приемника, сдвигать число в любую сторону на любое заданное число разрядов (разумеется, в пределах возможностей мультиплексора).

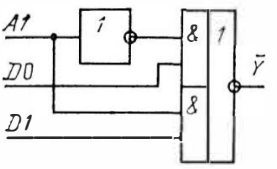


Рис. 3.12. Построение мультиплексора 2—1

Распространенным применением мультиплексора является передача слова прямым или обратным кодом в зависимости от управляющего уровня. Для этого в каждом разряде входы мультиплексора 2—1 подключаются к прямому и инверсному выходам триггера регистра-источника. Если мультиплексора 2—1 в используемой серии нет, то этот узел с активным низким уровнем выхода можно построить по рис. 3.12.

Мультиплексор можно использовать в качестве универсального логического элемента для реализации любой функции от числа аргументов, равного числу адресных входов мультиплексора. Мультиплексор, изображенный на рис. 3.11, *б*, реализует функцию, заданную табл. 3.2. Для

Таблица 3.2

№	a	b	c	Y	№	a	b	c	Y
0	0	0	0	0	4	1	0	0	0
1	0	0	1	0	5	1	0	1	1
2	0	1	0	0	6	1	1	0	1
3	0	1	1	1	7	1	1	1	0

этого входы данных мультиплексора подключены к источникам 1 и 0 в такой последовательности, которая полностью копирует последовательность единиц и нулей таблицы истинности. И при этом не требуется ни записи СДНФ, ни ее минимизации! Кстати, функция, заданная табл. 3.2, не минимизируется (в чем полезно убедиться лично), поэтому для своей реализации требует четырех элементов ЗИ-НЕ и трех инверторов, что в сумме даст почти два корпуса и 3τ задержки. Неудивительно, что способ реализации функций трех или четырех аргументов с помощью микросхемы мультиплексора весьма популярен у разработчиков. Следует помнить, что этот способ может дать экономию лишь при использовании микросхем. При разработке схем для кристаллов матричных и других БИС объем оборудования определяется числом базовых логических элементов, поэтому такой способ будет крайне расточительным.

3.3. Шифраторы

Шифратор, или *кодер* (*encoder*), выполняет функцию, обратную дешифратору. Условное изображение шифратора на схемах показано на рис. 3.13, *а*. Классический шифратор имеет m входов и n выходов, и при подаче сигнала на один из входов (обязательно на один, и не более) на выходе узла появляется двоичный код номера возбужденно-

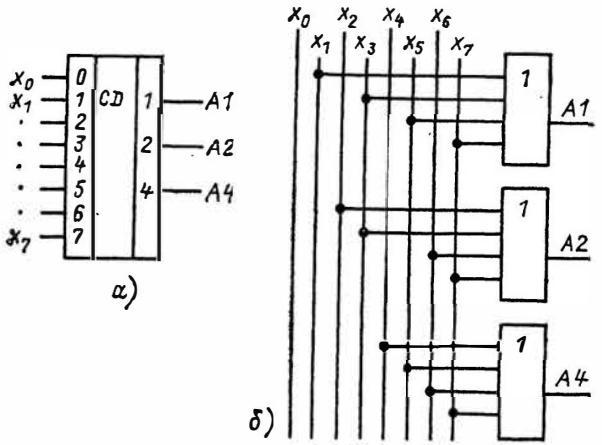


Рис. 3.13. Условное изображение (*а*) и функциональная схема (*б*) шифратора

го выхода. Число входов и выходов такого шифратора связано соотношением $m=2^n$. Шифратор можно использовать, например, для отображения в виде двоичного кода номера нажатой кнопки или положения многопозиционного переключателя.

Упражнение. На элементах серии К155 построить схему шифратора 8—3 (8 входов и 3 выхода).

Подсказка. Как и при построении компаратора и мультиплексора, табличный способ решения неприемлем из-за его громоздкости, и нужно попытаться найти какой-то более изящный подход. В данном случае существенно, что единица присутствует всегда только на одном из входов. Поэтому в создаваемой схеме не нужны конъюнкторы, выделяющие определенные комбинации нулей и единиц.

Решение. Схема может начинаться прямо с элементов ИЛИ — по одному на каждый выход. Это соображение сразу разбивает искомую схему на n простых фрагментов. Ко входу элементов ИЛИ каждого выходного разряда должны быть подключены те входы шифратора, в двоичном представлении номера которых есть единица в данном разряде. Так, к ИЛИ младшего разряда формируемого выходного кода должны быть подключены все нечетные входы, поскольку у всех нечетных номеров, и только у них, в младшем разряде содержится единица. Функциональная схема такого шифратора показана на рис. 3.13, *б*.

На схеме вход x_0 никуда не подключен, поскольку сигналу на этом входе соответствует выходной код «все нули». Недоумение может вызвать то, что схема совершенно не различает ситуаций «подан сигнал на вход x_0» и «не подано ни одного входного сигнала вообще». Причина в том, что при построении схемы учтено положение задания об обязательном присутствии сигнала на одном из входов, и оно использовано для минимизации аппаратурных затрат.

Схема, изображенная на рис. 3.13, *б*, годится для построения шифратора с числом входов, не превышающим удвоенного числа входов элементов ИЛИ. Для случаев, когда входов ИЛИ не хватает, можно воспользоваться экономичной двухкаскадной схемой шифратора, приведенной в [18]. Схему, показанную на рис. 3.13, *б*, легко преобразовать по формулам де-Моргана. Полученный двойственный вариант схемы будет иметь активные низкие уровни входов, а вместо элементов ИЛИ будут использованы более распространенные элементы И-НЕ.

Совместно с шифратором в состав кодирующих узлов может входить *схема выделения старшей единицы*. Эта схема преобразует m-разрядное слово следующим образом: все старшие нули и самая старшая единица входного кода

пропускаются на выход без изменения; все разряды, более младшие, чем старшая единица, заменяются нулями.

Один из возможных вариантов схемы выделения старшей единицы показан на рис. 3.14. На входы a_0, a_1, a_2 поступает преобразуемое слово (a_0 — младший разряд, a_2 — старший), на вход *EI* (от *enable in*) — *входной сигнал разрешения*. При $EI=1$ схема работает следующим образом.

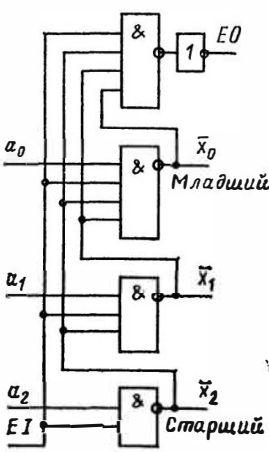


Рис. 3.14. Блок выделения старшей единицы

Любое число старших нулей порождает на выходах своих разрядов единицы и никак не влияет на работу элементов И-НЕ более младших разрядов. Любая самая старшая единица порождает на соответствующем выходе нуль (активный низкий уровень выхода) и запирает все более младшие элементы И-НЕ, устанавливая на их выходах неактивный высокий уровень. При этом низкий уровень появляется и на выходе *EO* (*от enable out*) — *выходе разрешения*.

Если разрядность обрабатываемого слова превышает разрядность схемы, то слово разбивается на группы и выход *EO* более старшей группы подается на вход *EI* более младшей. При таком включении единица, поступившая на любой вход любой группы, запрет не только все более младшие разряды своей группы, но по цепочке *EO—EI* — и все более младшие группы целиком. На выходах всей схемы останется только самая старшая единица входного слова, представленная активным низким уровнем.

Если к выходу схемы выделения старшей единицы подключить шифратор, то в сумме получится функциональный узел *приоритетного шифратора (priority encoder)*, формирующий в двоичном коде *номер самой старшей единицы* из всех, присутствующих во входном слове. С выходами схемы по рис. 3.14 хорошо стыкуются входы шифратора, если его выполнить по схеме, двойственной по отношению к показанной на рис. 3.13, *б*: инверсным выходам одной схемы будут соответствовать инверсные входы другой, и весь приоритетный шифратор будет построен на технологичных элементах без лишних инверторов. Если во входном слове присутствует

только одна единица, то приоритетный шифратор будет выполнять функцию обычного шифратора. Поэтому микросхемы обычных шифраторов не встречаются почти ни в одной серии, а приоритетные шифраторы выпускаются в составе многих серий.

На рис. 3.15, *а* показана схема 8-входового приоритетного шифратора К155ИВ1. Схема собственно приоритетного шифратора построена на элементах И-ИЛИ-НЕ, имеет вход-

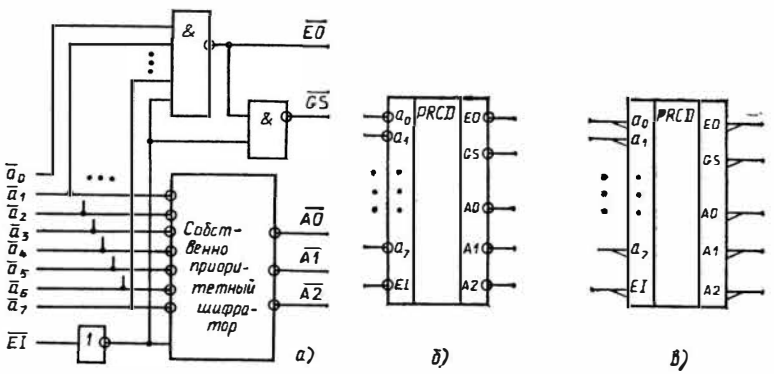


Рис. 3.15. Микросхема приоритетного шифратора:
а — функциональная схема группового канала; *б* — условное изображение на схемах, где действует соглашение положительной логики; *в* — изображение с указателями полярности выводов, не требующее соглашения о положительной логике

ные буферные инвертирующие усилители и ничего принципиально нового к уже изложенному не добавляет, поэтому на рисунке эта часть не раскрыта. Новым здесь является тракт *групповых сигналов*. Как видно из рисунка, сигнал *Выход разрешения EO* отражает ситуацию $EO = (EI)\&$. *(В данной группе нет ни одной единицы), а Групповой сигнал (group signal) GS* — ситуацию $GS = (EI)$ & *(В группе есть хотя бы одна единица).*

Сигнал $EI = 0$ запирает не только выходы адреса *AO, A1, A2,* но и выходы *EO* и *GS*.

Замечание. Все только что сказанное относится не к уровням сигналов, а к их смысловому содержанию. Фактические уровни сигналов в рассматриваемой схеме все инвертированы. Предыдущий абзац кроме объяснения работы схемы иллюстрирует также трудность объяснения поведения сколько-нибудь сложного устройства в терминах «единица — нуль» и рациональность разделения описания схемы на два

уровня: концептуальный, выполняемый в терминах смыслов сигналов и их взаимных отношений, и формальный, задающий четкое соответствие электрических уровней сигналов на входе и выходе. На примере К155ИВ1 (рис. 3.15) иллюстрируются взаимная увязка различных способов изображения узлов и обозначения входов и выходов при использовании инверсных сигналов.

На рис. 3.16 показаны примеры использования групповых сигналов при объединении микросхем для наращивания разрядности. Схема на рис. 3.16, *а* кроме адресных выходов *A'*0—*A'*3 имеет выходы $\overline{EO'}$ и *GS'*, поэтому ее можно нара-

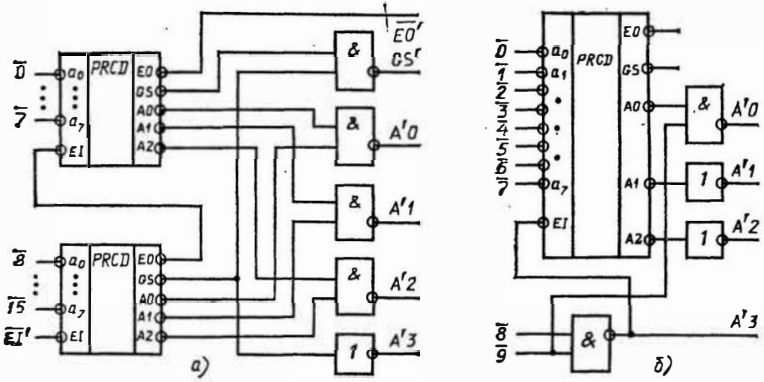


Рис. 3.16. Наращивание разрядности микросхем приоритетных шифраторов:

а — до шестнадцати входов; *б* — до десяти входов, в частности для десятичной клавиатуры или декадных переключателей

щивать и далее аналогичным способом. Схема на рис. 3.16, *б* ориентирована на кодирование номера нажатой десятичной клавиши или положения декадного переключателя. Если работа схем не понятна, рекомендуется сначала рассмотреть работу двойственных вариантов этих схем: в них почти не будет инверсий.

Перевод схемы для лучшего ее понимания, модификации или проверки правильности работы из инверсного базиса в привычный базис И, ИЛИ, НЕ — полезный и сильный прием при работе со сложными схемами.

Микросхема приоритетного шифратора К155ИВ1 (рис. 3.15, *а*) имеет две группы входных сигналов: a_i и *EI* и три группы выходов: *Aj*, *EO*, *GS*. Схема узла такова, что изменение сигналов одной входной группы не влияет на время

распространения сигналов по трактам другой входной группы. Поэтому микросхема полностью характеризуется лишь шестью значениями задержки распространения по трактам: a_i—Aj, a_i—EO, a_i—GS, EI—Aj, EI—EO, EI—GS. Если бы взаимное влияние трактов существовало, то схема характеризовалась бы двенадцатью значениями задержки.

Кроме кодирования состояний переключателей и номеров нажатых клавиш приоритетные шифраторы используются для определения номера устройства, подавшего сигнал запроса на обслуживание в микропроцессорных системах, входя в состав микросхем контроллеров прерываний, например КР580ВН59.

3.4. Преобразователи произвольных кодов

Если закон работы преобразователя не описывается каким-либо достаточно понятным правилом, как, например, работа декодера или шифратора, то единственной практически приемлемой формой задания преобразователя становится таблица. Поскольку таблица воплощает в себе идею полного перебора вариантов, она способна задавать абсолютно любой закон. В качестве примера пусть табл. 3.3 описывает закон работы некоторого трехсветного светофора, управляемого двухразрядным двоичным кодом. Таблица 3.4 дает сокращенное описание того же закона: в ней двоичные коды заменены их десятичными (или восьмеричными) эквивалентами. На рис. 3.17, *а* показано условное изображение кодового преобразователя, заданного табл. 3.3. Изображение по рис. 3.17, *б* допустимо использовать, когда коды имеют общепринятые названия.

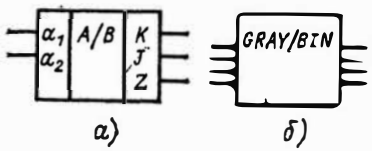


Рис. 3.17. Примеры условного изображения преобразователей кода:

а — преобразователь кода *A* в код *B* по произвольному закону, в данном случае — по закону, заданному табл. 3.3 или 3.4; *б* — преобразователь кодов, имеющих общепринятые названия, в данном случае — кода Грея в двоичный

К построению кодового преобразователя можно подойти с двух позиций. При первом подходе преобразователь реализуется как *система булевых функций* группы аргументов. Таблица 3.3 может рассматриваться как таблица истинности системы *B* функций *Z*, *J* и *K*. Простейшим способом построения схемы, отрабатывающей систему функ-

Таблица 3.3

A		B		
a_2	a_1	Z	J	K
0	0	1	0	0
0	1	0	0	1
1	0	0	1	1
1	1	0	0	0

Таблица 3.4

A	B
0	4
1	1
2	3
3	0

ций с m выходами, является синтез обычными методами m независимых одновыходных функций. Для трех выходов системы B записываются три выражения в дизъюнктивной форме:

$$Z = \bar{a}_2\bar{a}_1; \quad J = a_2\bar{a}_1; \quad K = \bar{a}_2 a_1 \vee a_2\bar{a}_1. \qquad (3.1)$$

Легко видеть, что независимая схемная реализация этих выражений скорее всего будет неоптимальной, поскольку элементы, реализующие конъюнкцию $a_2\bar{a}_1$, оказываются дублированными в схемах J и K. Более экономичное решение обычно получается при подходе к системе функций с учетом ее взаимосвязанности. Тогда часто удается выявить общие логические фрагменты, входящие в формулы нескольких выходов. Эти фрагменты достаточно реализовать схемно лишь один раз. На рис. 3.18, *а* показана возможная реализация кодового преобразователя, заданного (3.1), в базисе И-НЕ, ИЛИ-НЕ, учитывающая связность выражений (3.1). В более сложных многовыходных схемах экономия от учета связности обычно оказывается существенно большей, чем в рассмотренном очень простом примере, однако каких-

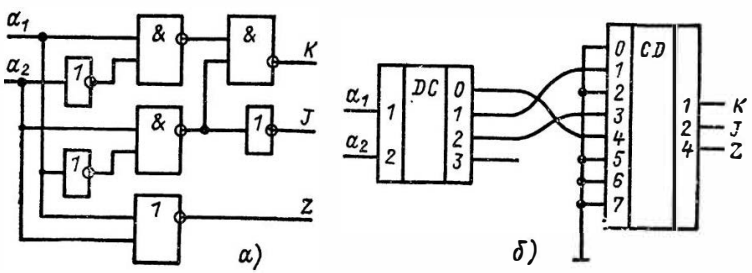


Рис. 3.18. Кодовый преобразователь, заданный табл. 3.3 или 3.4: *а* — синтезированный как система булевых функций; *б* — построенный как структура декодер—кодер

либо алгоритмов эффективного целенаправленного выявления общих частей, к сожалению, не существует. Как и синтез оптимальной одновыходной схемы, это поисковая задача, только обычно еще более сложная из-за большего числа возможных вариантов. В [1 и 2] описаны некоторые приемы, упорядочивающие и облегчающие поиск экономичных вариантов реализации систем булевых функций.

При втором подходе к построению кодового преобразователя он трактуется как пара *декодер — кодер*. Схема того же преобразователя, построенного по второму способу, показана на рис. 3.18, *б*. Число входов дешифратора равно числу входов преобразователя, число выходов шифратора — числу выходов преобразователя. Соединения дешифратора и шифратора выполняются в соответствии с таблицей, и в этом случае табл. 3.4 может оказаться удобнее, чем табл. 3.3. Часть выходов декодера и входов кодера может не использоваться. Если нескольким входным комбинациям соответствует одна и та же выходная, то соответствующие выходы декодера объединяют на элементе ИЛИ и выход последнего подают на нужный вход кодера. Эффективно стыкуются друг с другом декодер и кодер, построенные на элементах И-НЕ: первый имеет инверсный выход, а второй — инверсный вход. В качестве кодера можно использовать приоритетный шифратор.

При синтезе схемы на матричной БИС преобразователь, построенный по принципу реализации булевых функций, оказывается в среднем более экономичным по оборудованию, но менее быстродействующим, чем в варианте декодер—кодер, а при проектировании из готовых микросхем более выгодным и по числу корпусов, и по быстродействию обычно оказывается структура декодер — кодер. Однако потребляемая мощность в этом случае может быть больше, чем у схемы из отдельных логических элементов. Затраты времени инженера на логическое проектирование по схеме декодер — кодер неизмеримо меньше, чем затраты на проектирование преобразователя из россыпи.

3.5. Программируемые логические матрицы

Исключительная простота синтеза произвольных кодовых преобразователей по принципу декодер — кодер обусловила выпуск микросхем средней и даже большой интеграции, специально предназначенных для реализации кодовых преобразователей. Это микросхемы *программируемых*

логических матриц — ПЛМ (programmable logic array — PLA).

Условное изображение (*а*) ПЛМ и ее функциональная схема (*б*) показаны на рис. 3.19. Программируемая логическая матрица имеет n входов, k элементов *И*, выходы которых образуют k вертикальных шин, m элементов ИЛИ, выходы которых подключены к сумматорам по модулю 2, вы-

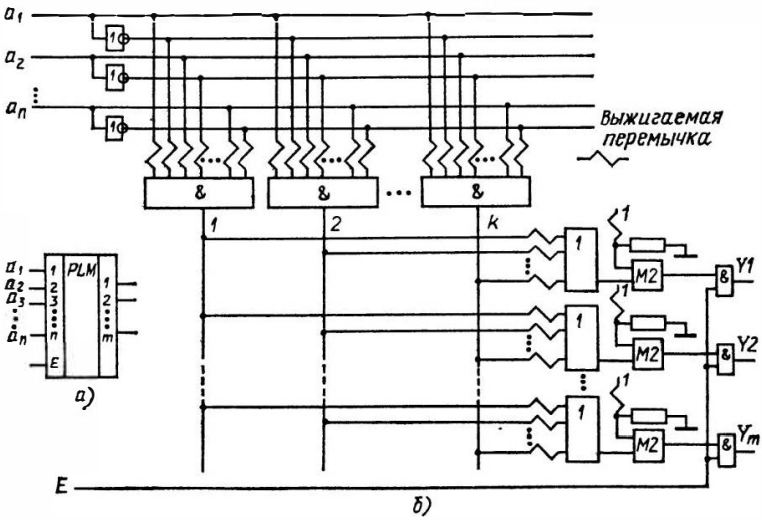


Рис. 3.19. Условное обозначение (*а*) и функциональная схема (*б*) программируемой логической матрицы (зигзагами обозначены разрушаемые перемычки)

полняющим роль управляемых инверторов. Выходы этих m инверторов являются выходами самой ПЛМ. Каждый элемент И имеет $2n$ входов, которыми он связан со всеми шинами входных сигналов и их инверсий. В линии связи включены специальные перемычки, обозначенные на рис. 3.19, *б* короткими зигзагами. Эти перемычки выполняются из определенного материала (например, нихром, кристаллический кремний) или в виде специальных p-n переходов так, чтобы их можно было выборочно разрушать (*«выжигать»*), оставляя лишь те связи, которые нужны потребителю ПЛМ. В ряде типов ПЛМ выжигать перемычки может сам потребитель, подавая на соответствующие выводы корпуса импульсы тока или напряжения определенной амплитуды и длительности.

Элементы ИЛИ в ПЛМ, так же как и элементы И, имеют на входах выжигаемые перемычки, с помощью которых они подключены ко всем вертикальным шинам. После выжигания на *программаторе* ненужных перемычек у элементов ИЛИ также остаются лишь те связи с вертикалями, которые необходимы потребителю. Техническая реализация элементов ИЛИ такова, что после выжигания перемычек на «ни к чему не подключенных» входах ИЛИ обеспечиваются уровни логического нуля.

Аналогичным образом программируют отсутствие или выполнение инвертирования выходов ИЛИ, соответственно пережигая или оставляя перемычки на верхних по рис. 3.19, *б* входах элементов М2.

Методы технологического исполнения элементов И, ИЛИ, М2 и разрушаемых перемычек могут быть различными. Они описаны в [20, 21, 59]. С точки зрения логического проектирования существенно лишь то, что схемотехник, использующий ПЛМ, может по своему усмотрению:

подать на любой элемент И любую комбинацию входов ПЛМ или их инверсий;

подключить к любому элементу ИЛИ любую комбинацию вертикальных шин (выходов И);

проинвертировать выходы любых ИЛИ.

Такие возможности позволяют очень просто реализовывать на ПЛМ преобразователи кодов или, что то же самое, системы логических функций. Пусть система из z функций от x аргументов содержит z выражений в виде дизъюнктивных форм. Для реализации этой системы на ПЛМ нужно каждый из u конъюнктивных членов ДНФ, описывающей первый выход системы, реализовать на u элементах И этой ПЛМ и затем эти u вертикальных шин подключить к первому элементу ИЛИ. Если по числу дизъюнктивных членов экономичнее оказывается выражение для инверсии функции первого выхода, то выход первого ИЛИ инвертируется. На других v конъюнкторах реализуются v конъюнктивных членов ДНФ второго выхода системы функций, и выходы этих конъюнкторов подключаются ко второму элементу ИЛИ. Если в ДНФ второго выхода есть конъюнктивные члены, совпадающие с членами ДНФ первого выхода, то дважды реализовывать их на элементах И не требуется. Нужно эти общие для первой и второй ДНФ вертикальные шины подключить и к первому, и ко второму элементам ИЛИ. И так далее, пока не будут реализованы все z ДНФ системы из z функций. В микросхеме ПЛМ будет занято x из n возмож-

ных входов, z из m возможных выходов, т. е. элементов ИЛИ, и y из k возможных вертикалей — столько, сколько неодинаковых конъюнктивных членов содержат в сумме ДНФ всех выходов.

Упражнение. Какие перемычки нужно выжечь в ПЛМ по рис. 3.19, чтобы реализовать на ней кодовый преобразователь, заданный табл. 3.3? Считать, что числа n, k и m, описывающие ПЛМ, достаточно велики.

Решение. При реализации на ПЛМ кодового преобразователя, заданного табл. 3.3, т. е. выражением (3.1), будут заняты:

два входа для a_2 и a_1 $(x=2)$;

три конъюнктора, реализующих конъюнкции $\bar{a}_2\bar{a}_1$, $\bar{a}_2 a_1$ (общая для выходов J и K), и $a_2\bar{a}_1$ $(y=3)$.

Инвертирования каких-либо выходов ИЛИ не требуется. Оставшиеся входы, вертикальные шины и выходы можно использовать для реализации других систем функций.

Запрограммированную ПЛМ удобно рассматривать как систему декодер - кодер в соответствии с рис. 3.18, *б*: группа задействованных элементов И образует неполный дешифратор, а группа задействованных элементов ИЛИ — неполный шифратор.

Сложность реализуемой функции может превышать возможности одной микросхемы ПЛМ. В этом случае используют несколько ПЛМ, при необходимости объединяя некоторые их выходы на дополнительных элементах ИЛИ. В некоторых микросхемах ПЛМ можно объединять выходы методом монтажного ИЛИ. Можно включать ПЛМ и последовательно, реализуя на микросхемах первого яруса логические фрагменты, общие сразу для нескольких выходов заданной функции. Более глубоко методы декомпозиции сложных функций для реализации их на ПЛМ освещены в [2, 22, 24].

Интересной модификацией ПЛМ является схема с *входными дешифраторами*. Входная часть такой ПЛМ показана на рис. 3.20, а ее выходная часть (элементы ИЛИ и инверторы) такая же, как у ПЛМ по рис. 3.19. В ПЛМ, изображенной на рис. 3.20, входные переменные объединены попарно с помощью двухвходовых декодеров с инверсными выходами. В обеих схемах каждая пара входных переменных представлена четырьмя горизонтальными линиями, т. е. на каждую пару переменных расходуется по четыре входа элементов И. Схема, показанная на рис. 3.19, позволяет для каждой пары входов реализовать восемь

функций:

$$a_1,\ a_2,\ \bar{a}_1,\ \bar{a}_2,\ \bar{a}_1\bar{a}_2,\ \bar{a}_1 a_2,\ a_1\bar{a}_2,\ a_1 a_2. \qquad (3.2)$$

Схема с двухвходовыми декодерами реализует те же функции за счет подачи на входы И сразу нескольких выходов декодера, например:

$$a_1 = (a_1 \vee a_2)(a_1 \vee \bar{a}_2); \quad a_1 a_2 = (\bar{a}_1 \vee a_2)(a_1 \vee \bar{a}_2)(a_1 \vee a_2).$$

Упражнение. Получите самостоятельно все остальные выражения (3.2).

Поскольку все входы И все равно существуют и даже предварительно подключены, никакой расточительности в таком способе нет. А достоинство есть: схема с двухвходовыми декодерами позволяет расширять по сравнению с набором (3.2) набор функций двух переменных, добавив еще шесть функций: четыре дизъюнкции, получаемые непосредственно с выходов декодера, и две функции

$$a_1 \oplus a_2 = (a_1 \vee a_2)(\bar{a}_1 \vee \bar{a}_2); \quad a_1 \equiv a_2 = (a_1 \vee \bar{a}_2)(\bar{a}_1 \vee a_2). \qquad (3.3)$$

Следовательно, ПЛМ с двухвходовыми декодерами при том же числе вертикальных шин и числе входов у элементов И и ИЛИ обладает заметно бо́льшими логическими возможностями, но для их реализации требует более высокой квалификации разработчика или соответствующих программ ЭВМ.

Многообещающим вариантом программируемых схем универсаль-

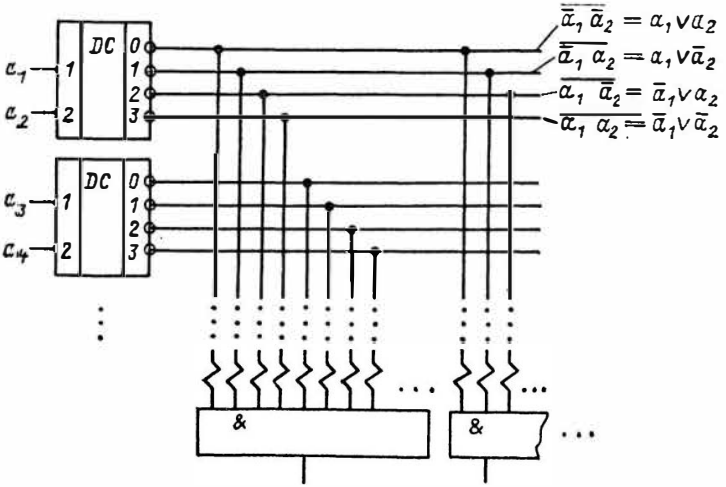


Рис. 3.20. Программируемая логическая матрица с входными дешифраторами. Показана лишь входная часть ПЛМ; выходная часть — элементы ИЛИ и М2, как и на рис. 3.19

ной логики является класс так называемых микросхем *программируемой матричной логики, ПМЛ (programmable array logic — PAL)*. Этим названием изготовители подчеркивают их отличие от микросхем ПЛМ *(PLA)*. В ПМЛ, как и в ПЛМ, входы и их инверсии с помощью разрушаемых перемычек подключены к многовходовым элементам И. Основное же отличие в том, что в ПМЛ выходные элементы ИЛИ не могут произвольно подключаться разработчиком к выходам любых элементов И. В ПМЛ каждый элемент ИЛИ постоянно подключен к определенной группе элементов И (группе вертикальных шин). Выпускаются несколько типов микросхем ПМЛ, отличающихся числом и составом этих групп. Необходимое разнообразие в коммутации выходов И со входами ИЛИ в ПМЛ обеспечивается тем, что разработчик в каждом случае выбирает определенный тип микросхемы точно так же, как он это делает при работе с россыпью.

Достоинство подхода — в существенном сокращении площади программируемой части кристалла (числа пережигаемых перемычек), что изготовители ПМЛ тут же превращают в выигрыш по скорости, цене, надежности и т. п. Серьезное изучение статистики проектируемых схем позволило изготовителям ограничиться совсем небольшим числом типов микросхем ПМЛ. Подробнее о них можно прочесть в [22].

В состав некоторых типов ПЛМ и ПМЛ изготовители вводят целый ряд остроумных решений, расширяющих возможности разработчика, например двунаправленные выводы, снабженные буфером с тремя состояниями, объединение некоторых входов в пары двухвходовыми декодерами, подключение к выходам ИЛИ четырех—восьми синхронных триггеров. Выходы триггеров выведены наружу, а, кроме того, разрушаемыми перемычками соединены со входами элементов И той же самой ПЛМ. Наличие триггерного регистра и возможности подавать выход ПЛМ на ее же собственный вход даже не внешним монтажом, а внутри корпуса, через выжигаемые перемычки, позволяет на одной ПЛМ реализовывать автоматы и весьма сложные фрагменты больших цифровых устройств. Значительная степень интеграции и доступность процедуры выжигания по сравнению с проектированием схемы матричной БИС делают ПЛМ и ПМЛ в некоторых применениях конкурентом даже базовых матричных кристаллов.

Типичный диапазон числа входов у микросхем ПЛМ и ПМЛ — 8—16, вертикальных шин (различных конъюнкций) — 24—96, выходов —4—12. Задержка составляет два — три значения задержки типового логического элемента соответствующей серии. Для подведения тока к выжигаемым перемычкам в состав микросхемы программируемой БИС кроме схемы, выполняющей требуемые функции, вводят специальные дешифраторы и мультиплексоры, управляемые группой *настроечных входов*. Иногда одни и те же

выводы микросхемы используют и как настроечные, и как рабочие, переключая их уровнем на соответствующем входе управления режимом. Примером ПЛМ является микросхема К556РТ1 с $n=16$, $k=48$, $m=8$ без входных двухвходовых декодеров, с задержкой 50 нс, изготовляемая по технологии ТТЛШ и совместимая по питанию и сигналам с ТТЛ- и ТТЛШ-сериями (см. [22]).

3.6. Постоянные запоминающие устройства

Программируемая логическая матрица, рассматриваемая как система декодер — кодер, имеет неполный дешифратор, т. е. число элементов И у ПЛМ много меньше числа всего возможного разнообразия входных кодов. Так, у 556РТ1 $n=16$, $2^n=65536$, а k всего 48.

Если при n входах число элементов И сделать максимально возможным, т. е. равным 2^n, и реализовать на них полный дешифратор, то ПЛМ превратится в *постоянное запоминающее устройство (ПЗУ)*. На рис. 3.21 показано условное изображение (*а*) на схемах и функциональная схема (*б*) ПЗУ. По принятой в среде специалистов по запоминающим устройствам терминологии входной код a_0—a_{n-1} называется адресом, 2^n вертикальных шин — числовыми

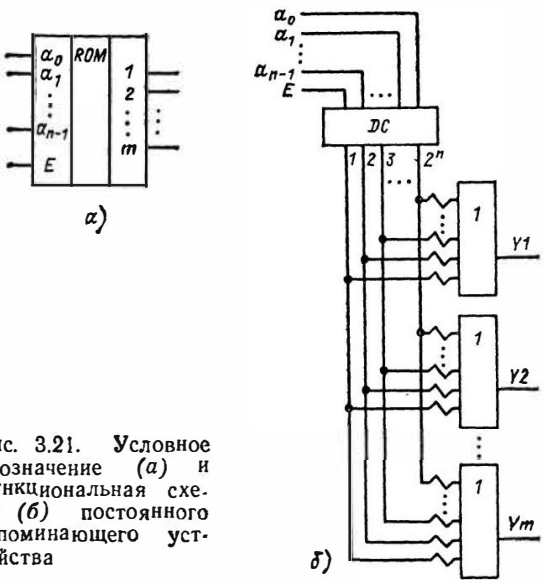


Рис. 3.21. Условное обозначение (*а*) и функциональная схема (*б*) постоянного запоминающего устройства

линейками, m выходов — разрядами хранимого слова. Выходные управляемые инверторы в ПЗУ не применяются, поскольку при наличии полного дешифратора необходимости в них нет. При поступлении на вход ПЗУ любого двоичного кода всегда выбирается одна из числовых линеек. При этом на выходе тех элементов ИЛИ, связь которых с данной числовой линейкой не разрушена, появляется 1. Это значит, что в данном разряде выбранного слова (или числовой линейки) записана 1. На выходах тех разрядов, связь которых с выбранной числовой линейкой выжжена, останутся нули. Закон программирования может быть и инверсным.

Таким образом, ПЗУ — это функциональный узел с n входами и m выходами, хранящий 2^n m-разрядных слов, которые при работе цифрового устройства не изменяются. При подаче на вход ПЗУ адреса на выходе появляется соответствующее ему слово. При логическом проектировании постоянное ЗУ рассматривают или как память с фиксированным набором слов, или как кодовый преобразователь, у которого в отличие от ПЛМ можно использовать все возможные кодовые комбинации на входах. Поэтому ПЗУ имеет значительно большую логическую мощность, чем ПЛМ, при том же числе выводов корпуса.

На схемах (рис. 3.21, *а*) ПЗУ обозначается *ROM* от *read only memory* — память, которая только считывает. Постоянные запоминающие устройства обычно имеют вход разрешения E. При активном уровне на входе E ПЗУ выполняет свои функции. При отсутствии разрешения выходы микросхемы неактивны. Разрешающих входов может быть несколько, тогда микросхема отпирается по совпадению сигналов на этих входах. В ПЗУ сигнал E часто называют *чтением ЧТ (read), выбором микросхемы ВМ, выбором кристалла ВК (chip select — CS)*.

Существует очень много различных технологических реализаций ПЗУ — на диодах, биполярных и МДП-транзисторах, других более сложных структурах. Некоторые ПЗУ программируются только изготовителем, еще до герметизации корпуса. Но кроме них существуют ПЗУ, которые может программировать заказчик, выжигая перемычки, аналогично программированию ПЛМ. Такие ПЗУ обозначаются *PROM* (от *programmable ROM*). Типовая емкость одного корпуса 256—8192 слов по 4—8 разрядов, время выборки (задержка распространения) — 50—100 нс.

На стадии отладки схем и алгоритмов удобны ПЗУ еще одного типа — *репрограммируемые* (перепрограммируемые) ПЗУ — *RPROM*, в которых информация хранится в виде электрических зарядов в МДП-

структурах. Запись (программирование) осуществляется импульсами напряжения от 48 В в более старых и до 12 В в более новых разработках. Время сохранения информации обычно от нескольких месяцев до нескольких лет. Стирание в различных сериях осуществляется или импульсами напряжения противоположной полярности, или, чаще, облучением кристалла ультрафиолетовым светом через специальное окошко в корпусе микросхемы. После стирания в *RPROM* можно с помощью программатора записать новую информацию. Недостаток этих ПЗУ по сравнению с прожигаемыми — большее время выборки: сотни наносекунд.

Технологические, схемные и эксплуатационные параметры ПЗУ освещены в [20, 21, 23, 59].

Микросхемы ПЗУ приспособлены для наращивания. Чтобы увеличить число разрядов хранимых слов, все входы микросхем включают параллельно (рис. 3.22, *а*), а с уве-

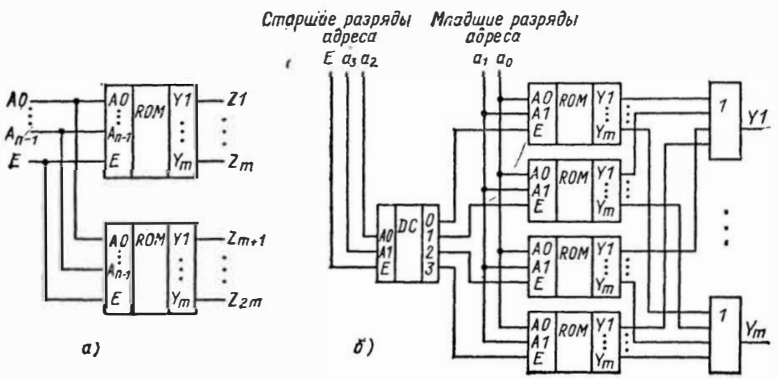


Рис. 3.22. Наращивание микросхем ПЗУ:
а — увеличение разрядности выходного слова; *б* — увеличение числа адресов

личившегося суммарного числа выходов снимается выходное слово соответственно увеличенной разрядности. Для увеличения числа самих хранимых слов (рис. 3.22, *б*) адресные входы микросхем включают параллельно и рассматривают как младшие разряды нового, расширенного адреса. Добавленные старшие разряды нового адреса поступают на декодер, который по входам E выбирает одну из микросхем. При малом числе микросхем дешифрацию старших разрядов можно делать на конъюнкции разрешающих входов самих ПЗУ. Выходы одноименных разрядов при увеличении числа хранимых слов должны объединяться с помощью функций ИЛИ. Специальных элементов ИЛИ не требуется,

если выходы микросхем ПЗУ выполнены или по схеме открытого коллектора для объединения методом монтажного ИЛИ, или по схеме буфера с тремя состояниями, допускающего непосредственное физическое объединение выходов.

Выходы микросхем ПЗУ обычно инверсные, инверсным часто бывает и вход E. Наращивание ПЗУ и ПЛМ может потребовать введения буферных усилителей для увеличения нагрузочной способности некоторых источников сигналов, учета вносимых этими усилителями дополнительных задержек, но в общем при сравнительно небольших объемах памяти, что типично для устройств автоматики, наращивание ПЗУ обычно не порождает принципиальных проблем. Ряд вопросов, связанных с наращиванием ПЗУ, освещен в [20, 21, 23].

3.7. О выборе способа реализации кодовых преобразователей

Использовать ПЗУ для построения преобразователя нужно тогда, когда преобразованию подлежат все или почти все комбинации входных переменных, а общее число переменных больше примерно шести—восьми. Если в заданной для реализации функции используется лишь сравнительно малая доля всех возможных входных комбинаций, то рациональнее применить ПЛМ или ПМЛ. В силу значительно меньшего числа элементов в корпусе ПЛМ или ПМЛ они обычно дешевле, имеют меньшую задержку и потребляют меньшую мощность, чем ПЗУ.

Что касается выбора между ПЛМ и набором отдельных логических микросхем для построения кодовых преобразователей, то, начиная уже с весьма простых преобразователей, ПЛМ оказываются выгоднее и по времени, и по затратам аппаратуры. Следует, однако, иметь в виду, что ПЛМ имеет явное преимущество перед россыпью при реализации именно систем функций, а в случае одновыходных функций — лишь когда источники аргументов и потребители функций расположены компактно. Если же одной логической матрицей заменяется множество разбросанных по плате отдельных логических элементов, обслуживающих БИС, то результирующий выигрыш по габаритным характеристикам становится уже сомнительным. Дело в том, что в большинстве технологий монтажа площадь, занятая связями, превышает площадь, занятую собственно логическими элементами. Поэтому тенденция концентрировать логические операции в одном крупном элементе, а не выполнять их на местах, в среднем увеличивает занятую площадь платы. Проигрышной по числу микросхем (правда, выигрышной с точки зрения унификации) является и попытка замены программируемыми

матрицами специализированных микросхем типа декодеров, мультиплексоров, схем свертки, сумматоров и т. п.

В специализированной микросхеме рационально использованы все выводы корпуса, а при вложении в ПЛМ различных схем конкретных узлов практически всегда часть выводов будет неиспользована. ПЛМ предназначена для упрощения реализации произвольных нестандартных функций, и, как всякий универсальный инструмент, она проигрывает специализированным микросхемам, каждая из которых специально спроектирована для экономной реализации именно ее функции. Тенденция использовать ПЛМ повсеместно, заменяя ими и логические микросхемы, и микросхемы средней интеграции, подогревается рекламой ПЛМ, а также модой, которая в технике, увы, тоже существует.

3.8. Применение преобразователей кода

Можно отметить ряд наиболее массовых, установившихся областей применения преобразователей кода.

Прежде всего это преобразователи двоичного кода в код управления различными световыми индикаторами. Преобразователи выпускаются на базе программируемых изготовителем ПЗУ и ПЛМ в составе ряда серий. Например, выпускаются преобразователи для следующих типов индикаторов: семисегментного цифрового (на управление которым ориентирована, в частности, серия К514); матрицы 7×5 точек; матрицы 7×4 точки; линейных индикаторов амплитуды со шкалами в виде светящегося столбика, подвижной точки, пары точек. По традиции преобразователи этой группы называют дешифраторами, и микросхемы обозначают символами ИД — как и обычные дешифраторы.

Кодовые преобразователи используют в устройствах цифровой автоматики для хранения табличных функций. Например, всего в одной распространенной микросхеме ПЗУ КР556РТ5 емкостью 512 8-разрядных слов можно записать таблицу синусов от 0 до 90° с шагом менее 0,2° и выходной погрешностью менее 0,5 %. Одним из видов табличной функции являются таблицы коррекции нелинейности датчиков измерительных устройств. Таблица может сниматься индивидуально для каждого прибора и записываться в ПЗУ коррекции.

Кодовые преобразователи являются основным, наиболее сложным звеном цифровых управляющих автоматов. Об этой области их применения будет сказано в гл. 11.

Кодовые преобразователи используют в шифровальных устройствах при аппаратном воплощении методов *криптографической* защиты информации. На рис. 3.23, *а* показан шифровальный узел *перестановки*, простой, но имеющий не очень высокую степень засекречивания: число возможных вариантов n-входового блока всего $n!$ Такой узел осуществляет преобразование кода, не изменяющее числа его единиц. Блок *под-*

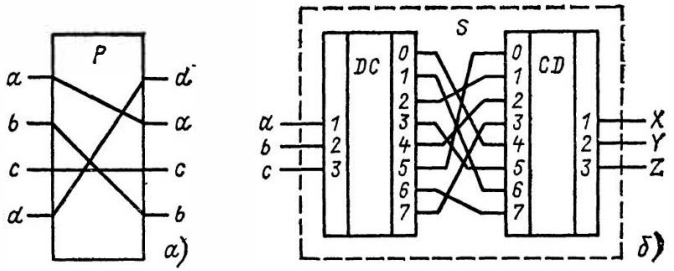


Рис. 3.23. Функциональные узлы криптографической защиты информации: перестановки *(а)* и подстановки *(б)*

становки (рис. 3.23, *б*), в котором перестановка производится после дешифрации входного кода, имеет значительно большую степень засекречивания: число возможных вариантов здесь равно $2^n!$. Для схемы, показанной на рисунке, это более 40 тыс. Дешифрирование информации осуществляется блоком с обратным законом подстановки. Для дальнейшего повышения степени засекречивания блоки перестановки и подстановки включают последовательно в несколько слоев. Меняя с помощью электронного коммутатора наборы блоков, можно быстро менять шифры. Информацию, пропущенную через подобные блоки, можно передавать по открытым каналам связи или хранить на общедоступных магнитных лентах при исключительно малом риске несанкционированного использования. Подробнее с методами защиты информации можно ознакомиться в [25].

ГЛАВА 4

СУММАТОРЫ И ПРОСТЫЕ СХЕМЫ КОНТРОЛЯ

4.1. Сумматоры

Сумматор выполняет арифметическое сложение чисел. Для сложения чисел, представленных позиционным параллельным двоичным кодом, применяются параллельные двоичные сумматоры. Условное обозначение сумматора показано на рис. 4.1. Сумматор имеет n входов разрядов слагаемого A, n входов разрядов слагаемого B и *вход переноса* cr (от *carry* — перенос). Выходами сумматора являются n выходов разрядов суммы S и *выход переноса (переполнения)* CR.

Сумматор характеризуется четырьмя значениями задержки распространения:

T_{crS} — от подачи входного переноса до установления всех выходов суммы при постоянном уровне на всех входах слагаемых;

T_{AS} — от одновременной подачи всех слагаемых до установления всех выходов суммы при постоянном уровне на входе переноса;

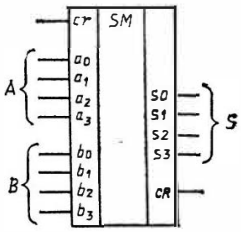


Рис. 4.1. Условное обозначение сумматора

T_{crCR} — от подачи входного переноса до установления выходного переноса при постоянном уровне на входах слагаемых;

T_{ACR} — от подачи всех слагаемых до установления выходного переноса при постоянном уровне на входах слагаемых.

4.2. Сумматоры с последовательным переносом

Одноразрядный сумматор

Простейший способ построения n-разрядного сумматора — это последовательное включение *n одноразрядных сумматоров,* складывающих одноименные разряды слагаемых, как показано на рис. 4.2. Выход переноса CR каждого разряда подключен ко входу переноса cr соседнего старшего разряда. Входной перенос всего n-разрядного сумматора подается на вход cr самого младшего разряда. Выходной перенос CR самого старшего разряда является выходом переноса всего n-разрядного сумматора. Таким образом, за-

Таблица 4.1

№ п/п	Входы			Выходы		№ п/п	Входы			Выходы	
	перенос	слагаемые		перенос	сумма		перенос	слагаемые		перенос	сумма
	cr	a	b	CR	S		cr	a	b	CR	S
0	0	0	0	0	0	4	1	0	0	0	1
1	0	0	1	0	1	5	1	0	1	1	0
2	0	1	0	0	1	6	1	1	0	1	0
3	0	1	1	1	0	7	1	1	1	1	1

дача построения сумматора с *последовательным переносом (ripple carry adder)* сводится к построению схемы одноразрядного сумматора. Основное требование к одноразрядному сумматору — минимизация задержки распространения T_{crCR} — тракта от входа переноса cr до выхода переноса CR. Как видно из рисунка, при оценке самого длинного тракта от входов слагаемых младшего разряда (или входного переноса $cr_{вх}$) до выхода суммы старшего разряда (или выхода переноса $CR_{вых}$) задержка T_{crCR} суммируется по всем разрядам и тем самым в основном определяет значения всех четырех задержек n-разрядного сумматора.

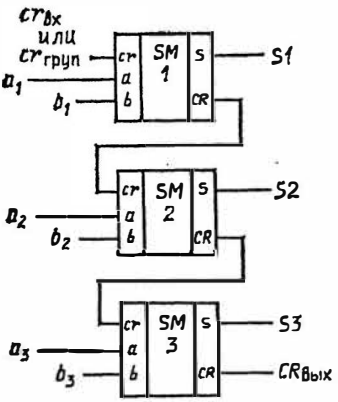


Рис. 4.2. Трехразрядный сумматор с последовательным переносом

Таблица истинности одноразрядного сумматора представлена в табл. 4.1. Совершенные дизъюнктивные нормальные формы функций суммы и переноса имеют вид

$$CR = \overline{cr} \cdot a \cdot b \vee cr \cdot \bar{a} \cdot b \vee cr \cdot a \cdot \bar{b} \vee cr \cdot a \cdot b;$$

$$S = \overline{cr} \cdot \bar{a} \cdot b \vee \overline{cr} \cdot a \cdot \bar{b} \vee cr \cdot \overline{a} \cdot \overline{b} \vee cr \cdot a \cdot b. \qquad (4.1)$$

Функции эти уже встречались: CR совпадает с функцией мажорирования «2 из 3» (см. табл. 2.3). По этой причине микросхема мажорирования выпускается в составе очень немногих серий. В сериях, где она отсутствует, операцию мажорирования может выполнять выход переноса CR микросхемы одноразрядного сумматора, а сумматоры входят в состав почти всех развитых серий. Функция S совпадает с функцией М2 (см. табл. 1.7). Это сумма по модулю 2 трех аргументов:

$$S = cr \oplus a \oplus b. \qquad (4.2)$$

Задача разработчика одноразрядного сумматора — так преобразовать СДНФ (4.1), чтобы задержка тракта вход переноса cr — выход переноса CR была минимальной и при этом чтобы аппаратурные затраты W были также возможно меньше. Для представления функции CR можно воспользо-

ваться выражением (2.12)

$$CR = cr \cdot a \vee cr \cdot b \vee a \cdot b. \qquad (4.3)$$

Как следует из § 1.8, СДНФ для S в базисе НЕ, И, ИЛИ не минимизируется. Тогда для реализации CR по (4.3) потребуется три двухвходовых И-НЕ плюс один трехвходовой И-НЕ, для реализации S по СДНФ — четыре трехвходовых И-НЕ плюс один четырехвходовой и плюс три инвертора для каждого из аргументов. Это дает оценку аппаратурных затрат одноразрядного сумматора по общему числу входов и выходов всех его элементов $W = 40$ выводов. Задержка тракта переноса $cr—CR$ равна 2τ (цепочка И-НЕ—И-НЕ). Это оценки схемы сумматора, построенной «в лоб».

Улучшить качество схемы можно за счет ликвидации ее избыточности, т. е. нахождения такой структуры, в которой общие части функций CR и S реализовывались бы только один раз. Ниже будут рассмотрены три схемы сумматоров, наиболее удачные с точки зрения реализации их на элементах с инверсными выходами.

Сумматор на элементах И-ИЛИ-НЕ

В этой схеме непосредственно из аргументов cr, a и b строится функция CR, которая затем используется в качестве четвертого аргумента для построения функции S:

$$\overline{CR} = \overline{cr \cdot a \vee cr \cdot b \vee a \cdot b};$$

$$\overline{S} = \overline{cr \cdot \overline{CR} \vee a \cdot \overline{CR} \vee b \cdot \overline{CR} \vee cr \cdot a \cdot b}. \qquad (4.4)$$

Схема, реализующая (4.4), показана на рис. 4.3. Она хорошо приспособлена к реализации по ТТЛ-технологии (элементы И-ИЛИ-НЕ), очень экономична по аппаратурным затратам (сумма выводов элементов $W = 17$, что в два с лишним раза лучше, чем затраты тривиальной схемы), задержка ее тракта переноса — всего 1τ.

Использование одного инвертирующего элемента в тракте переноса, казалось бы, должно затруднить стыковку сумматоров друг с другом. Действительно, с выхода соседнего младшего разряда снимается инверсия переноса, а на вход данного разряда требуется перенос без инверсии. Введение дополнительного инвертора сведет на нет получен-

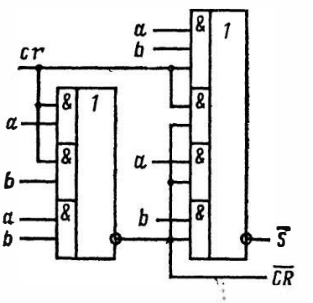


Рис. 4.3. Одноразрядный сумматор на элементах И-ИЛИ-НЕ

ное в данной схеме преимущество в скорости по сравнению со схемой, построенной непосредственно по (4.1). Однако при построении именно сумматора это затруднение легко обходится, поскольку как функция переноса, так и функция суммы относятся к классу самодвойственных функций. *Самодвойственными* называют такие функции, значения которых инвертируются при инвертировании всех входящих в них аргументов. В этом легко убедиться, сравнивая попарно строки 3 и 4, 2 и 5, 1 и 6, 0 и 7 табл. 4.1. В каждой паре взаимно обратны значения как всех аргументов, так и выходных величин CR и S. Учитывая это, можно поступить следующим образом. Для построения многоразрядного сумматора тракты переноса одноразрядных сумматоров соединяются цепочкой без инверторов. На те сумматоры, на которые поступают инвертированные значения переноса, слагаемые подаются также инверсным кодом (вместо a подается $\overline{a}$, вместо b—$\overline{b}$). Тогда в силу самодвойственности функций CR и S на выходах этих разрядов получатся неинвертированные значения CR и S. На те разряды, на которые поступают неинвертированные значения переносов, слагаемые a и b подаются неинвертированными. На выходах этих разрядов значения переноса и суммы получаются инвертированными: $\overline{CR}$ и $\overline{S}$. Использовать схему с черезразрядной инверсией несложно. Иногда удается функцию инвертирования возложить на триггеры регистров, снимая в соответствующих разрядах слагаемые и сумму не с прямых, а с инверсных выходов триггеров. Если это неудобно, то в нужных разрядах на входе и выходе сумматора ставят инверторы. Задержку всего многоразрядного сумматора это увеличивает всего на 2τ, что существенно меньше по сравнению с введением инверторов в тракт переноса.

Схема, показанная на рис. 4.3, использована в микросхемах ИМ1, ИМ2, ИМ3 серии К155, представляющих собой 1-, 2- и 4-разрядные сумматоры. В состав двух последних схем в половину разрядов введены необходимые инверторы, поэтому все входы и выходы микросхем ИМ2 и ИМ3 — прямые, без инверсий.

Сумматор на элементах И-НЕ

Для этой схемы функцию переноса удобно представить следующим образом:

$$CR = cr\,(a \oplus b) \vee ab = \overline{\overline{cr\,(a \oplus b)}\cdot\overline{ab}} = \overline{cr\cdot\overline{ab}\cdot\overline{a}\,\overline{b}\cdot\overline{ab}}\,,$$

или

$$\overline{CR} = \overline{cr \cdot \overline{ab} \cdot \overline{\overline{a}\overline{b}} \cdot \overline{a}\overline{b}}. \qquad (4.5)$$

Поскольку функция переноса самодвойственна, то при инвертировании всех аргументов она проинвертируется:

$$CR = \overline{\overline{cr} \cdot \overline{ab} \cdot \overline{\overline{a}\overline{b}} \cdot \overline{a}\overline{b}}. \qquad (4.6)$$

Таким образом, и перенос, и его инверсию можно представить в виде конъюнкции двух членов. Главная особенность описываемого сумматора заключается в том, что эта конъюнкция выполняется не в данном разряде, а в соседнем старшем. Схема такого сумматора дана на рис. 4.4, *а*. Пе-

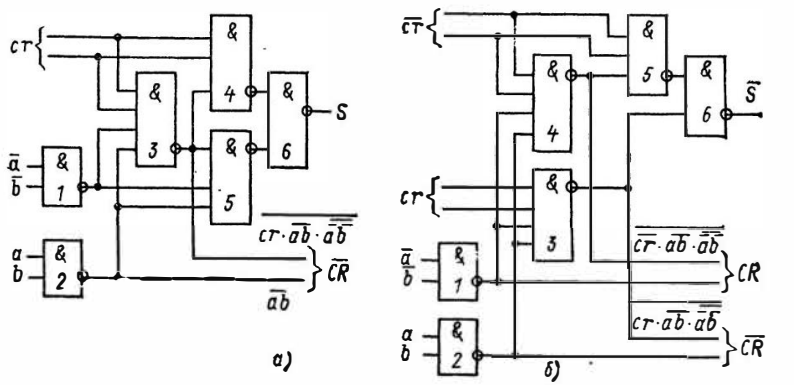


Рис. 4.4. Одноразрядные сумматоры на элементах И-НЕ
а — схема с черезразрядной инверсией суммы; *б* — схема без черезразрядной инверсии

ренос *cr* в виде конъюнкции сигналов, присутствующих на паре входных проводов, поступает на четырехвходовой элемент *3*. С выхода этого элемента снимается сигнал $cr \cdot \overline{ab} \cdot \overline{\overline{a}\overline{b}}$ — первая составляющая конъюнкции (4.5). Вторая составляющая снимается с элемента *2*, в результате на двух выходных проводах появляются обе составляющие инверсии выходного переноса $\overline{CR}$ в соответствии с (4.5).

Поскольку на соседний старший разряд сумматора поступает инверсия переноса $\overline{cr}$, то слагаемые на него также подаются инвертированными: вместо *a* и *b* на элемент *2* подаются $\overline{a}$ и $\overline{b}$ и наоборот, т. е. задача стыковки одноразр-

рядных сумматоров при инвертирующем тракте переноса решается так же, как и в сумматоре на И-ИЛИ-НЕ.

В первом разряде многоразрядного сумматора или оба входа переноса объединяются, или на один из них подается логическая 1. В последнем разряде пара проводов выхода переноса объединяется на И.

Функция суммы сумматора по рис. 4.4, *а* имеет сложный вид

$$S = \overline{cr \cdot \overline{ab} \cdot \overline{\overline{ab}} \cdot cr \cdot cr \cdot \overline{\overline{ab}} \cdot \overline{\overline{ab}} \cdot \overline{\overline{ab}} \cdot \overline{\overline{\overline{ab}}}}. \qquad (4.7)$$

Используя соотношение

$$\overline{ab} \cdot \overline{\overline{ab}} = a \oplus b \qquad (4.8)$$

и формулу (4.5), выражение для суммы (4.7) можно привести к привычному виду (4.2).

Сумматор, изображенный на рис. 4.4, *а* построен только из элементов И-НЕ, что удобно для реализации на матричных БИС, и имеет минимально возможную задержку тракта переноса — всего один элемент И-НЕ. По аппаратурным затратам схема также весьма экономична: 22 вывода самой схемы, показанной на рис. 4.4, *а*, плюс четыре вывода двух инверторов в трактах слагаемых, если нет возможности использовать инверсные выходы регистров.

Сумматор с двухколейным переносом

На рис. 4.4,*б* показана модификация схемы по рис. 4.4,*а*, вырабатывающая во всех разрядах только функцию $\overline{S}$ без черезразрядной инверсии. В этой схеме перенос между разрядами передается парафазным кодом по двум трактам. Один тракт образован выходами элементов *2* и *3*. Он идентичен тракту переноса сумматора по рис. 4.4, *а* и реализует инверсию переноса $\overline{CR}$ по выражению (4.5). Другой тракт переноса сумматора по рис. 4.4, *б* образован элементами *1* и *4* и реализует прямую функцию переноса CR согласно выражению (4.6). Наличие и прямого, и инверсного переноса позволяет в каждом разряде выбрать именно ту его фазу, которая требуется для выработки функции суммы всегда одной и той же фазности:

$$\overline{S} = \overline{\overline{cr} \cdot \overline{ab} \cdot \overline{\overline{ab}} \cdot \overline{cr} \cdot cr \cdot \overline{ab} \cdot \overline{\overline{ab}}}. \qquad (4.9)$$

Теми же средствами, что и в случае (4.7), выражение (4.9) можно привести к виду (4.2).

Черезразрядная инверсия суммы схемы по рис. 4.4, *а*

ликвидирована в схеме по рис. 4.4, *б* ценой увеличения оборудования, исчисляемого суммой выводов элементов, всего на единицу. Остальные характеристики схем идентичны.

4.3. Сумматор с параллельным переносом

В сумматоре с последовательным переносом тракты переносов всех одноразрядных сумматоров включены последовательно. Поэтому, даже при минимальной задержке тракта переноса одноразрядного сумматора в 1τ задержка n-разрядного сумматора не может быть менее $n\tau$. Для уменьшения задержки используется принцип *параллельного переноса,* когда входной перенос каждого разряда вырабатывается независимо от переноса соседнего младшего разряда. Он формируется как функция только слагаемых и входного переноса $cr_{\text{груп}}$ всего n-разрядного сумматора. Для всех разрядов сигналы переноса cr формируются параллельно.

Для описания работы сумматора с параллельным переносом удобно ввести две вспомогательные функции γ и π, называемые иногда *подготовительными:*

γ — функция *генерации переноса, CRG (от carry generation).* Функция $\gamma=1$, когда слагаемые данного разряда таковы, что перенос в соседний старший разряд равен 1 независимо от значения входного переноса cr данного разряда, т. е. $\gamma=1$, если в данном разряде сумматора генерируется перенос в соседний старший разряд:

$$\gamma = ab; \tag{4.10}$$

π — функция *прозрачности,* или *распространения, CRP* (от *carry propagation*). Функция $\pi=1$, когда слагаемые данного разряда таковы, что при переносе в данный разряд cr, равном 1, перенос в соседний старший разряд CR также равен 1, т. е. $\pi=1$, если тракт переноса данного разряда сумматора *прозрачен* для сигнала переноса cr.

С помощью γ и π можно представить работу тракта переноса одного разряда сумматора:

$$CR_i = cr_{i+1} = \gamma_i \vee cr_i \cdot \pi_i. \tag{4.11}$$

Прозрачность разряда может быть представлена двумя различными функциями:

$$\pi' = a \oplus b; \quad \pi'' = a \vee b = \overline{\bar{a}\bar{b}}. \tag{4.12}$$

Значение CR в (4.11) будет одним и тем же независимо

от того, используется функция π' или π''. Эти функции имеют различные значения лишь при $ab=1$, а поскольку при таком сочетании слагаемых $\gamma=1$, то перенос CR равен 1 независимо от вида второго члена дизъюнкции (4.11) — $cr\cdot\pi'$ или $cr\cdot\pi''$. Конкретная форма функции прозрачности выбирается разработчиком исходя из удобства ее реализации в используемом базисе.

Входным переносом первого разряда будет входной перенос $cr_{\text{груп}}$ всего n-разрядного сумматора

$$cr_1 = cr_{\text{груп}}.$$

Входной перенос второго разряда cr_2 будет равен 1, если слагаемые первого разряда a_1 и b_1 таковы, что в этом разряде генерируется перенос, или если на вход всего сумматора поступил перенос $cr_{\text{груп}}$, а слагаемые a_1 и b_1 таковы, что первый разряд прозрачен по тракту переноса:

$$CR_1 = cr_2 = \gamma_1 \vee cr_{\text{груп}}\,\pi_1.$$

Аналогично рассуждая, можно прийти к выводу, что перенос $CR_i = cr_{i+1}$ на входе $(i+1)$-го разряда сумматора должен быть равен 1, если перенос генерируется в i-м разряде; или он генерируется в $(i-1)$-м разряде, и при этом i-й разряд прозрачен; или он генерируется в $(i-2)$-м разряде, и при этом прозрачны разряды $(i-1)$-й и i-й; или...; или если на сумматор поступил входной перенос $cr_{\text{груп}}$, и при этом прозрачны все разряды от 1-го по i-й включительно;

$$CR_i = cr_{i+1} = \gamma_i \vee \gamma_{i-1}\,\pi_i \vee \gamma_{i-2}\,\pi_{i-1}\,\pi_i \vee \ldots$$
$$\ldots \vee \gamma_1\,\pi_2\,\pi_3 \ldots \pi_{i-1}\,\pi_i \vee cr_{\text{груп}}\,\pi_1\,\pi_2 \ldots \pi_{i-1}\,\pi_i. \quad (4.13)$$

Полученные выражения можно представить в любом базисе: И-НЕ, И-ИЛИ-НЕ, ИЛИ-НЕ. В базисе И-НЕ, например, (4.13) представляется следующим образом:

$$CR_i = cr_{i+1} = \overline{\overline{a_i b_i}\cdot\overline{a_{i-1} b_{i-1}\,\pi_i}\cdot\overline{a_{i-2} b_{i-2}\,\pi_{i-1}\,\pi_i}}\cdot\ldots$$
$$\ldots\cdot\overline{a_1 b_1\,\pi_2\,\pi_3 \ldots \pi_{i-1}\,\pi_i}\cdot\overline{cr_{\text{груп}}\,\pi_1\,\pi_2 \ldots \pi_i}. \quad (4.14)$$

Трехразрядный сумматор, тракт параллельного переноса которого построен в соответствии с (4.14), показан на рис. 4.5. Три блока в правой половине рисунка — это узлы сложения по модулю 2 трех аргументов: cr_i, a_i и b_i, вырабатывающие значение суммы данного разряда S_i в соответствии с (4.2). Они могут быть построены и по любой другой подходящей схеме. Выходной перенос $CR_{\text{груп}}$ n-раз-

рядного сумматора вырабатывается в последней секции блока параллельного переноса как перенос в очередной, $(n+1)$-й разряд.

Задержка T получения суммы сумматора с параллельным переносом слагается из одинаковых для всех (кроме

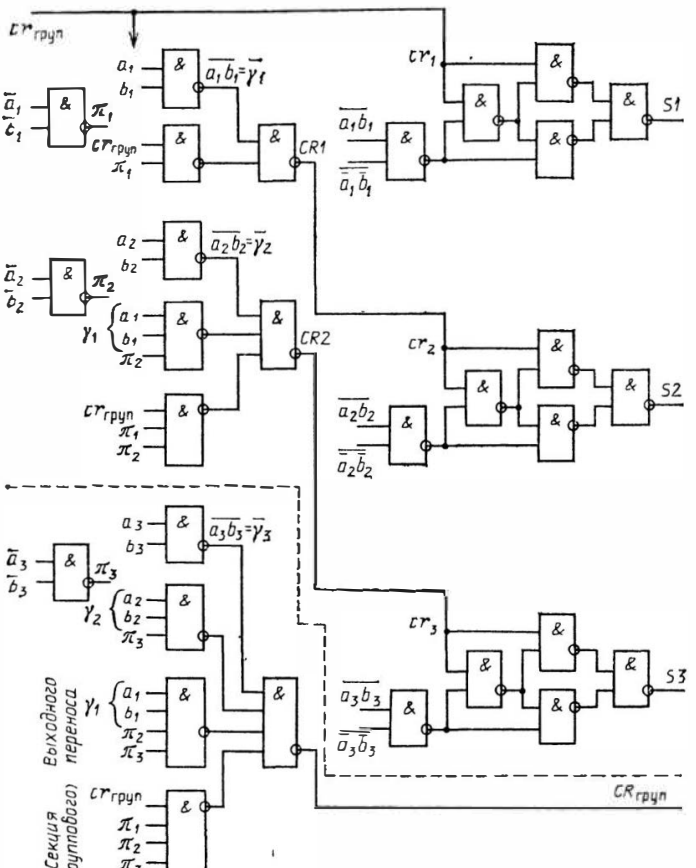


Рис. 4.5. Трехразрядный сумматор с параллельным переносом

первого) разрядов задержки блока переноса — $(2 \div 3)$ τ в зависимости от логического базиса и задержки трехвходовой схемы сложения по модулю 2 — $(2 \div 4)\tau$. От числа разрядов ни задержка получения суммы, ни задержка по-

лучения выходного переноса $CR_{\text{груп}}$ не зависят. Аппаратурные затраты W сумматора с параллельным переносом заметно превышают W сумматора с последовательным переносом и быстро растут с ростом разрядности.

Диапазон разрядности, в пределах которого сумматор с параллельным переносом эффективен, невелик. При малой разрядности — 2, 3, даже 4 — он хуже сумматора с последовательным переносом и по W, и по T (сравните рис. 4.4 и особенно рис. 4.3 с 4.5). Однако при каждом шаге увеличения разрядности на единицу также на единицу растет и требуемое число входов элементов И блока параллельного переноса. Поэтому, начиная с разрядности, примерно равной максимальному числу логических входов элементов используемой элементной базы, многовходовые И пришлось бы набирать, соединяя каскадом несколько маловходовых элементов. В принципе это возможно, но для схем сумматоров есть более эффективное решение (см. § 4.4). Поэтому разрядность реально используемых сумматоров с параллельным переносом сверху ограничивается максимальным числом входов элементов и редко превышает восемь. Параллельный перенос имеет 4-разрядный сумматор К155ИМ6, задержка которого примерно вдвое меньше тоже 4-разрядного К155ИМ3, но с последовательным переносом.

4.4. Краткий обзор сложных сумматоров

Для ускорения переноса в сумматорах с большим числом разрядов применяют принцип *группового переноса.* Сумматор разбивают на группы, представляющие собой небольшие сумматоры с разрядностью обычно от 2 до 8. Каждый такой мини-сумматор имеет свой штатный вход переноса *cr.* Суть группового переноса заключается в том, что в дополнение к тракту переноса внутри группы, который в общем может быть как параллельным, так и последовательным, строят *тракт переноса II яруса* между группами, который вырабатывает сигналы *групповых переносов,* подаваемые на входы *cr* всех мини-сумматоров. Тракт группового переноса удается построить так, что время распространения переноса в нем между группами оказывается меньше, чем если бы этот перенос распространялся по цепям внутригрупповых трактов (*трактов I яруса*).

Как и в обычном сумматоре, который можно рассматривать как частный случай сумматора с групповым переносом, когда разрядность каждой группы равна 1, тракт меж-

группового переноса может быть построен как параллельным, когда все групповые переносы *cr* вырабатываются параллельно как функции только слагаемых, так и последовательным, когда исходным материалом для переноса в каждую следующую группу служит перенос, поступающий на вход данной группы.

Параллельный перенос между группами в сочетании с параллельным переносом внутри группы дает самые быстрые сумматоры в диапазоне разрядности, приблизительно от 24 до 64. Задержка таких *параллельно-параллельных* сумматоров не зависит от разрядности и составляет $(9 \div 10)\,\tau$ в зависимости от используемого логического базиса. За скорость приходится платить, и аппаратурные затраты таких сумматоров заметно превышают затраты сумматоров с другими типами переносов. В диапазоне разрядности примерно от 8 до 24 первенство по скорости переходит к сумматорам с параллельным переносом между группами и с последовательным внутри групп. Разрядность групп при этом выбирают небольшой — от 2 до 4.

С принципами построения трактов группового переноса можно ознакомиться по [59]. Подробно схемы различных вариантов межгрупповых трактов переноса приведены в [26], а в [16], кроме того, даны оценки значений задержки и аппаратурных затрат для большого числа унифицированных структур сложных сумматоров во всем диапазоне практически используемой разрядности.

При необходимости выполнять над числами не только суммирование, но и другие операции целесообразно применять микросхемы универсальных *арифметико-логических устройств (АЛУ)*. Они выпускаются в составе многих серий, содержат обычно по 4 разряда и хорошо приспособлены для наращивания разрядности. Подавая на управляющие входы микросхемы 5-разрядный код, можно задать одну из 32 арифметических или логических операций. С правилами выполнения операций над двоичными числами можно ознакомиться по [3, 26]. В комплекте с АЛУ обычно выпускают микросхему, содержащую тракт группового переноса, что позволяет ускорить операцию суммирования при большом числе разрядов. Сами микросхемы АЛУ при этом играют роль 4-разрядных групп. Примерами микросхем АЛУ являются К155ИП3 (и к ней — схема ускорения переноса К155ИП4) (см. [8, 28]), а также 564ИП3 (к ней ускорение переноса — 564ИП4) (см. [11]). При необходимости выполнять над числами цепочки последовательных операций нуж-

но переходить на использование микропроцессорных БИС (см. [3, 54, 59]).

В блоках, обменивающихся информацией с человеком, наряду с двоичной используется и *двоично-десятичная* система счисления. При этом каждая десятичная цифра — от 0 до 9 — кодируется четверкой двоичных разрядов — *тетрадой* со значениями соответственно от 0000 до 1001. Сложение десятичных чисел выполняется на *десятичном сумматоре,* который по объему оборудования лишь примерно вдвое сложнее двоичного. Способ построения десятичного сумматора изложен в [59]. Зная принцип работы десятичного сумматора, можно построить сумматор и по любому другому основанию, например по основанию 60 для работы с секундами и минутами.

Перевод данных из двоично-десятичной системы в двоичную в пределах двух-трех десятков можно выполнить достаточно экономично, если десятки с помощью кодового преобразователя преобразовать в двоичный код и затем сложить их с единицами на двоичном сумматоре. Обратный перевод — из двоичной системы в двоично-десятичную—также может быть выполнен с помощью кодового преобразователя из восьмеричного кода в двоично-десятичный и десятичного сумматора, но с увеличением чисел аппаратурные затраты быстро растут. Экономичный по аппаратурным затратам, хотя и медленный способ преобразования в обе стороны на основе счетчиков описан в § 9.5. Но в общем для чисел большой разрядности аппаратное преобразование становится слишком громоздким, и перевод стараются выполнять с помощью микропроцессоров. Алгоритмы перевода приведены в [3 и 27].

4.5. Инкрементор

Кроме схемы одноразрядного сумматора, складывающего три числа: cr, a, b, применяется схема *полусумматора (half-adder),* складывающего два числа — cr и a в соответствии с табл. 4.2. Функции выхода полусумматора

$$CR = cr \cdot a, \; S = cr \oplus a \tag{4.15}$$

легко реализуются в любом техническом базисе.

Полусумматоры, соединенные по тракту переноса цепочкой, как показано на рис. 4.6, *б*, образуют схему *инкрементора,* условное обозначение которого показано на рис. 4.6, *а*. При уровне на входе «+1», равном 0, инкрементор пропус-

Таблица 4.2

cr	a	CR	S	cr	a	CR	S
0	0	0	0	1	0	0	1
0	1	0	1	1	1	1	0

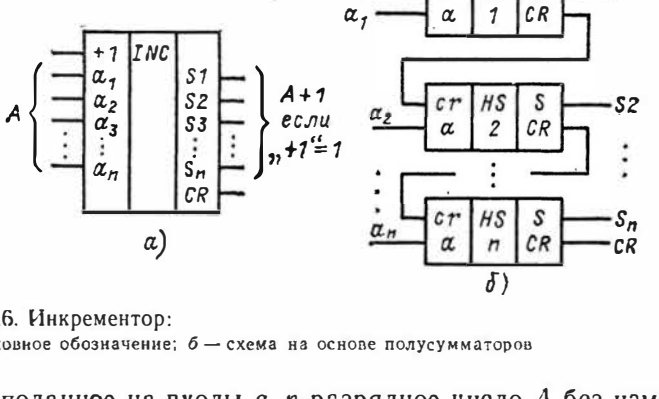


Рис. 4.6. Инкрементор:
а — условное обозначение; *б* — схема на основе полусумматоров

кает поданное на входы a_i n-разрядное число A без изменения, при уровне на «+ 1», равном 1, прибавляет к A единицу, т. е. формирует на выходе S число $A+1$.

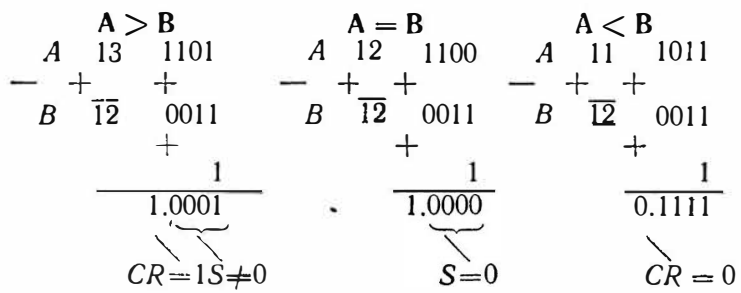
На рис. 4.6, *б* показан инкрементор с последовательным переносом. По аналогии с сумматорами можно построить инкрементор с параллельным или групповым переносом. Схемы при этом будут намного проще, чем схемы переноса сумматоров, поскольку функций генерации в инкрементере не существует. Существуют только функции прозрачности $\pi_i = a_i$.

По аналогии с инкрементором легко построить схему *декрементора* — узла, уменьшающего на 1 поданное на вход число A, если на вход «*—1*» *займа* подан единичный потенциал. Инкрементеры и декрементеры используются, например, при организации серий обращений к последовательным адресам памяти и для этого вводятся в состав схем микропроцессоров. Они выполняют функции счетчиков, но без запоминания результата и соответственно без потери исходного числа. Оба числа — и A, и $A+1$ — существуют одновременно, одно на входе, а другое на выходе схемы. Счетчик, состоящий из инкрементора и двух регистров — для A и $A+1$ — легко тестировать, поскольку в нем нет традиционных для счетчиков цепочек последовательно включенных триггеров, существенно замедляющих проверку.

4.6. Компаратор величин

Этот узел (*magnitude comparator*) выявляет не только факт равенства двух поданных на вход n-разрядных чисел A и B, как это делает простейший компаратор, рассмотренный в § 2.9, но и сравнивает числа по значению. Он имеет три выхода: «$A>B$», «$A=B$» и «$A<B$», и в зависимости от соотношения величин A и B активный уровень появляется на одном из этих выходов.

Построить компаратор величин можно на основе сумматора, выполнив на нем операцию вычитания $A-B$ и проанализировав результат. Для этого на сумматор нужно число B (в отличие от числа A) подать в инверсной форме (т. е. заменив в нем все нули единицами, а все единицы — нулями), а на вход переноса $cr_{вх}$ сумматора подать единицу. Тогда выходной перенос CR будет равен 0 лишь в том случае, когда A строго меньше B. Равенство суммы 0 будет признаком того, что $A=B$. Единица переноса при ненулевой сумме говорит о том, что A строго больше B. Сказанное иллюстрируют примеры:



Примечание. Вычитание из числа A числа $B=12_{10}=1100_2$ заменено прибавлением к A обратного кода числа B (его инверсии), равного 0011_2, и еще единицы младшего разряда.

Правила справедливы, если числа A и B рассматриваются как положительные величины, без знака. Если же их старшие разряды трактуются как знаки, то правила будут несколько иные. Их легко вывести самостоятельно, если уметь обращаться с *обратными и дополнительными кодами* (см. [3 или 26]). Схема, реализующая описанный алгоритм, показана на рис. 4.7, *а*. Ее можно перевести в любой базис и дополнить инверторами на входах числа B.

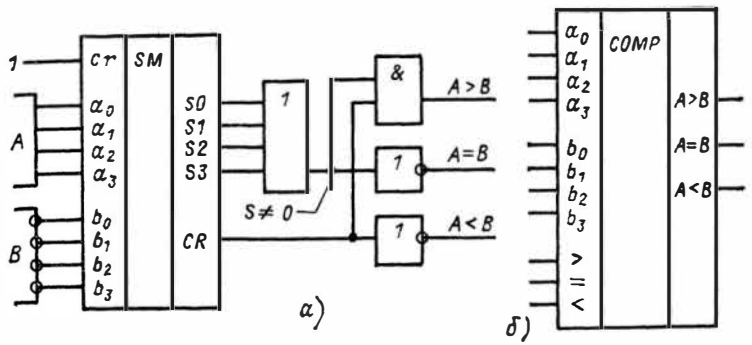


Рис. 4.7. Компаратор величин:
а — простейший вариант схемы; *б* — выводы корпуса микросхемы компаратора величин

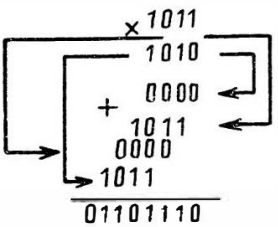
При изготовлении схемы из отдельных элементов ее объем можно уменьшить по сравнению со схемой на рис. 4.7, *а*. Применение сумматора избыточно, поскольку вырабатываемое им фактическое значение суммы никак не используется, кроме нулевого его значения. От сумматора нужен лишь тракт переноса, а его разрядные схемы выработки суммы можно заменить схемами $a_i \oplus b_i$, фиксирующими равенство a_i и b_i, как в компараторе, изображенном на рис. 2.14. Функция $a_i \oplus b_i$ одновременно используется и как функция прозрачности π_i при построении тракта переноса.

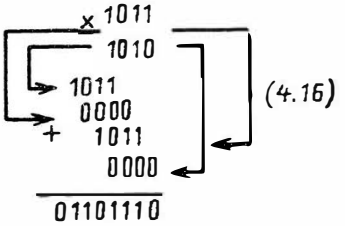
Примером компараторов величин могут служить 4-разрядные микросхемы К561ИП2 и К555СП1. Они имеют встроенные инверторы числа B и еще три входа — «>», «=», «<» (рис. 4.7, *б*) для наращивания разрядности. Наращивать разрядность можно, включая микросхемы или последовательно, или в виде пирамиды. Схемы наращивания для К561ИП2 приведены в [28]. Схемы наращивания К555СП1 отличаются некоторыми деталями в подключении выводов, а главное тем, что при пирамидальном наращивании, подавая разряды сравниваемых чисел не только на входы a_i и b_i, но и на входы «>» и «<», можно разрядность сравниваемых чисел увеличить с 20 до 24.

В устройствах автоматики компараторы используются для сигнализации о выходе величин за пределы допуска, в приводах следящих систем для определения направления воздействия, ликвидирующего рассогласование, при построении счетчиков и сумматоров по произвольному основанию.

4.7. Умножители

Умножение в двоичной системе выполняется аналогично умножению в десятичной согласно приведенной схеме:





(4.16)

n-разрядное множимое умножается на каждую из n цифр множителя, в результате получается n n-разрядных частичных произведений. Поскольку цифрой множителя может быть только или 0, или 1, умножение на данную цифру множителя сводится к пропусканию или непропусканию множимого сквозь n-разрядную цепочку конъюнкторов, управляемую данной цифрой множителя. Каждое частичное произведение сдвигается на столько разрядов, каков номер породившей его цифры множителя (считая с нуля). Все взаимно сдвинутые частичные произведения складываются, образуя $2n$-разрядное произведение. Принято говорить, что умножение выполняется начиная с младших разрядов множителя, если последовательность сложения частичных произведений такая, как показано в левом столбце примера (4.16), и со старших разрядов множителя, если это делается, как показано в правом столбце примера.

Схема, выполняющая перемножения двух 4-разрядных чисел A и B, показана на рис. 4.8, *а*. На входы k_i и z_i внимания пока обращать не следует, о них будет сказано позднее. Частичные произведения формируются из множимого на управляемых разрядами множителя B многоразрядных конъюнкторах *И0—И3* (см. рис. 4.8, *а* и *б*). Складываются частичные произведения на четырех горизонтальных линейках одноразрядных сумматоров (рис. 4.8, *а* и *в*), сдвинутых одна по отношению к другой на один разряд. Умножение ведется начиная от старших разрядов множителя. Выходной перенос каждой линейки сумматоров (кроме верхней) заводится на неиспользуемый вход сумматора соседнего старшего разряда, расположенного в предыдущей линейке. Восьмиразрядное произведение M снимается с выходов сумматоров.

Кроме входов a_3—a_0 и b_3—b_0 умножитель может иметь входы

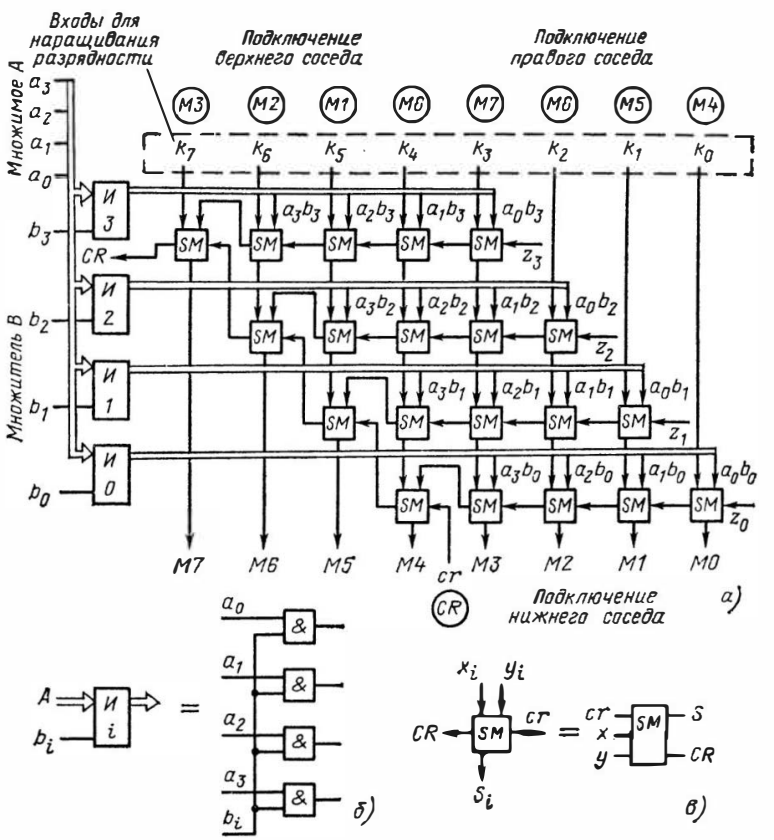


Рис. 4.8. Умножитель:

а — структурная схема; *б*, *в* — упрощенные обозначения; *б* — многоразрядного вентиля; *в* — сумматора

k_7—k_0 и z_3—z_0. Тогда он реализует более сложную функцию, чем просто произведение AB:

$$M = AB + K + Z, \qquad (4.17)$$

что расширяет возможности разработчика.

Если умножитель выполнен в виде микросхемы, то входы k используются при наращивании разрядности умножителя. Схема включения четырех умножителей для удвоения разрядности сомножителей показана на рис. 4.9. На рис. 4.8, *а* в кружках показаны обозначения выводов соседних микросхем такого же типа, подключаемых к соответствующим входам k данной микросхемы. Умножитель увеличенной разрядности также способен реализовывать выражение (4.17).

Если от умножителя не требуется ни возможности наращивания разрядности, ни реализации выражения (4.17), то его схему можно упростить. Верхнюю линейку сумматоров можно исключить, а на входы слагаемых второй линейки подать оба первых частичных произведения: $b_3(a_3a_2a_1a_0)$ и сдвинутое $b_2(a_3a_2a_1a_0)$. Правые, самые младшие сумматоры каждой линейки можно также исключить. Произведение a_0b_0 поступает прямо на выход *M0*.

Показанная на рис. 4.8, *а* структура умножителя не единственно возможная. Можно сложение частичных произведений вести, начиная

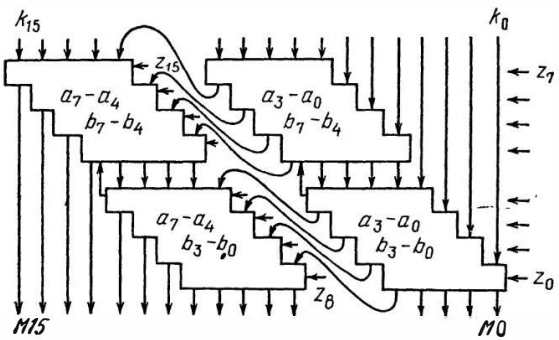


Рис. 4.9. Включение умножителей по рис. 4.8 при наращивании разрядности

с младших разрядов множителя по аналогии с левым столбцом примера (4.16). Тогда каждая следующая линейка сумматоров будет сдвинута не вправо, а влево по отношению к своей верхней соседке. Но при одинаковых задержках одноразрядных сумматоров время получения произведения в такой схеме будет больше, чем в схеме, показанной на рис. 4.8, *а*. Перенос с выхода каждого одноразрядного сумматора может передаваться не на соседний старший разряд своей линейки, а на соседний старший разряд следующей нижней линейки. Существует множество других вариантов, имеющих свои особенности, достоинства и недостатки, которые полно и квалифицированно изложены в [26]. Там же приведены схемы для быстрого деления чисел.

Схемы перемножения чисел входят в состав микросхем расширителей арифметических устройств микропроцессоров, в состав самих микропроцессоров более поздней разработки, выпускаются в виде самостоятельных микросхем. Примером может служить 2-разрядная К561ИП5 (см. [11]), реализующая выражение (4.17) и приспособленная для наращивания разрядности. Структура связей между ее сумматорами отличается от показанной на рис. 4.8 (см. [11]). В устройствах автоматики умножители используются для введения в систему изменяемых коэффи-

циентов. Примером могут служить торговые цифровые весы, высвечивающие платежную сумму при наборе на клавиатуре цены килограмма товара.

4.8. Контроль по четности

На передаваемые по линии связи или хранимые в памяти данные воздействуют различные помехи, которые могут исказить эти данные. Простейшим способом удостовериться, что полученные с линии или извлеченные из памяти данные искажены ошибкой и использовать их нельзя, служит введение *контроля по четности (контроль по нечетности, parity check — контроль по паритету)*. В его основе лежит операция сложения по модулю 2 всех двоичных разрядов контролируемого слова. Если число единиц в слове четное, то сумма по модулю 2 его разрядов будет 0, если нечетное — то 1. *Признаком четности* называют инверсию этой суммы.

Общая схема организации контроля показана на рис. 4.10. На n-входовом элементе $\overline{M2}$ формируется признак

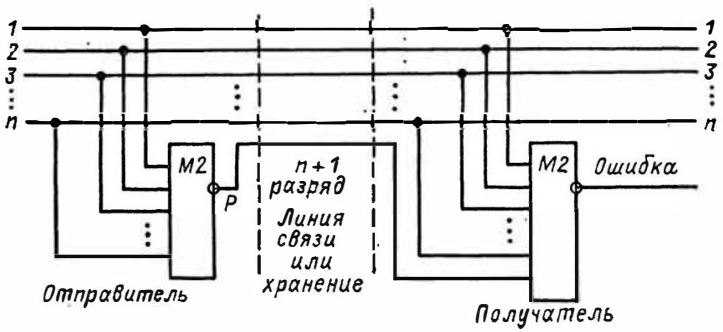


Рис. 4.10. Организация контроля по нечетности

четности P числа, который в качестве дополнительного, $(n+1)$-го *контрольного разряда (parity bit)* отправляется вместе с передаваемым словом в линию связи или запоминающее устройство (ЗУ). Передаваемое $(n+1)$-разрядное слово имеет всегда нечетное число единиц. Если в исходном слове оно было нечетным, то функция $\overline{M2}$ от такого слова равна 0, и нулевое значение контрольного разряда не меняет числа единиц при передаче слова. Если же число единиц в исходном слове было четным, то контрольный

разряд P для такого числа будет равен 1, и результирующее число единиц в передаваемом $(n+1)$-разрядном слове станет нечетным. Вид контроля, когда по линии передается нечетное число единиц, по строгой терминологии называют контролем по нечетности.

На приемном конце линии или после чтения из памяти от полученного $(n+1)$-разрядного слова снова берется свертка по четности. Если значение этой свертки равно 1, то или в передаваемом слове, или в контрольном разряде при передаче или хранении произошла ошибка. Столь простой контроль не позволяет исправить ошибку, но он по крайней мере дает возможность при обнаружении ошибки исключить неверные данные, затребовать повторную передачу и т. д.

Систему контроля можно построить на основе не только инверсии функции М2, но и прямой функции М2 (строго контроль по четности). Однако в этом случае исходный код «все нули» будет иметь контрольный разряд, равный 0. В линию отправится посылка из сплошных нулей, и на приемном конце она будет неотличима от весьма опасной неисправности — полного пропадания связи. Поэтому контроль по четности в своем чистом виде почти никогда не применяют, контрольный разряд формируют как функцию четности, и в нестрогой терминологии «контролем по четности» называют то, что, строго говоря, на самом деле является контролем по нечетности.

Контроль по четности основан на том, что одиночная ошибка (безразлично — пропадание единицы или появление лишней) инвертирует признак четности. Однако две ошибки проинвертируют его дважды, т. е. оставят без изменения, поэтому двойную ошибку контроль по четности не обнаруживает. Рассуждая аналогично, легко прийти к выводу, что контроль по четности обнаруживает все нечетные ошибки и не реагирует на любые четные. Пропуск четных ошибок — это не какой-либо дефект системы контроля. Это следствие предельно малой *избыточности* при контроле по четности, равной всего одному разряду. Для более глубокого контроля требуется соответственно и бóльшая избыточность. Если ошибки друг от друга не зависят, то из необнаруживаемых чаще всего будет встречаться двойная ошибка, и при вероятности одиночной ошибки, равной q, вероятность двойной будет q^2. Поскольку в нормальных цифровых устройствах $q \ll 1$, необнаруженные двойные ошибки встречаются значительно реже, чем обнаруженные одиночные. Поэтому даже при таком простом

контроле качество работы устройства существенно возрастет. Еще раз напомним: это верно лишь для взаимно независимых ошибок.

Признак четности можно использовать для контроля только неизменяемых данных. При выполнении над данными каких-либо логических операций признаки четности слов в общем изменяются, и попытки компенсировать эти изменения оказываются неэффективными. Счастливое исключение — операция арифметического сложения: сумма по модулю 2 признаков четности двоичных слагаемых и всех возникших в процессе сложения переносов равна признаку четности кода арифметической суммы этих слагаемых.

Контроль по четности — самый дешевый по аппаратурным затратам вид контроля, и применяется он очень широко. Практически любой канал передачи цифровых данных или запоминающее устройство, если они не имеют какого-либо более сильного метода контроля, защищены контролем по четности. Подробнее об этом виде контроля см. [29].

4.9. Контроль по Хэммингу

Развитие принципа контроля по четности приводит к *корректирующему коду Хэмминга,* который позволяет не только обнаруживать, но и исправлять одиночную ошибку. Возможность исправления ошибки основывается на повторенной k раз системе контроля по четности, но не всего слова сразу, а k определенных групп его разрядов. Слово разбивается на группы так, чтобы номер каждого разряда однозначно определялся по его принадлежности или непринадлежности к этим группам.

Рисунок 4.11, *а* иллюстрирует принцип построения 15-разрядного слова, передаваемого по линии связи, — *кодового слова.* Оно состоит из одиннадцати разрядов *информационного слова,* первоначально предназначавшегося для передачи по линии (или в память), разряды которого обозначены малыми латинскими буквами, и четырех *контрольных разрядов.* Контрольные разряды обозначены греческими буквами и показаны размещенными в кодовом слове не компактным массивом, а вперемежку с информационными. На рис. 4.11, *а* контрольные разряды зачернены.

Группы контроля по четности компонуются из разрядов кодового слова по следующим законам:

1) каждый разряд кодового слова входит в состав стольких групп, сколько единиц содержится в двоичном

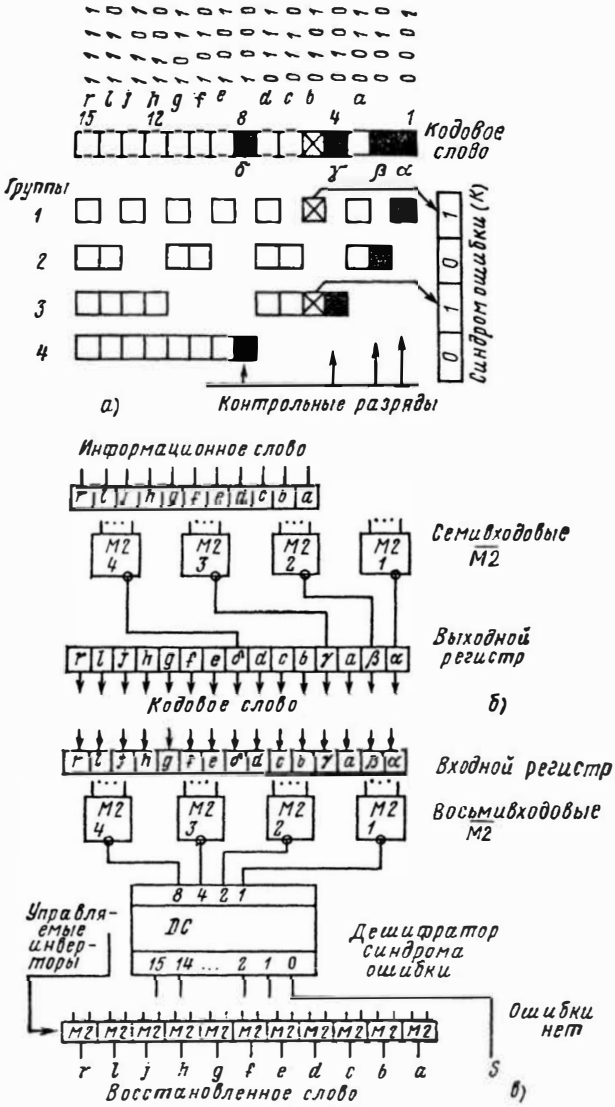


Рис. 4.11. Контроль по Хэммингу:

а — структура кодового слова и контрольных групп; *б* — узлы кодирован датчика; *в* — узлы контроля и коррекции приемника

коде его номера. Так, разряд 4 входит в состав всего одной, 3-й, контрольной группы; разряд 7 входит в состав трех групп: 1, 2 и 3-й;

2) в каждую i-ю контрольную группу входят те разряды кодового слова, в двоичном номере которых в i-й позиции стоит единица. Например, 3-я контрольная группа включает разряды 4, 5, 6, 7; 12, 13, 14 ... Самый младший разряд каждой группы — контрольный. Очевидно, что это разряды, в двоичных номерах которых содержится только одна единица: 1, 2, 4, 8-й ... Такое расположение контрольных разрядов облегчает понимание принципа построения схем формирования этих разрядов.

На рис. 4.11, *б* показан набор функциональных узлов, формирующих кодовое слово на передающей стороне. Семиразрядные контрольные группы информационного слова, скомпонованные в соответствии с рис. 4.11, *а*, поступают на семивходовые свертки по четности *М2.1—М2.4*. Результаты свертки каждой группы записываются в ее контрольный разряд.

Функциональные узлы приемной стороны представлены на рис. 4.11, *в*. Разряды принятого кодового слова также поступают на четыре свертки по четности, но здесь на каждой свертке обрабатывается вся группа вместе с ее контрольным разрядом. Выходы сверток образуют 4-разрядный *синдром ошибки*, или *корректирующий код К*. Если при передаче кодового слова в одном из его разрядов произошла ошибка (на рис. 4.11, *а* в качестве примера крестиком отмечен поврежденный пятый разряд), то будет зафиксировано нарушение четности именно в тех контрольных группах, в состав которых входит неверный разряд. В результате код синдрома ошибки укажет номер неисправного разряда принятого кодового слова. Так, код синдрома 0101 есть двоичный номер поврежденного разряда № 5.

Для восстановления слова неверный разряд нужно проинвертировать. Синдромные разряды поступают на дешифратор, активный выход которого возбуждает управляемый инвертор (двухвходовой элемент М2) в том разряде, в котором произошла ошибка. На рис. 4.11, *в* показаны 11 инверторов, обеспечивающих исправление разрядов только информационного слова. Предполагается, что оно дальше передаваться не будет, поэтому контрольные разряды больше не нужны, и в них ошибки корректировать не требуется. Разумеется, можно включить в состав схемы все 15 управляемых инверторов и исправлять не только информацион-

ные, но и контрольные разряды. Если ошибка не обнаружена, то все синдромные разряды будут нулями, о чем сигнализирует возбужденный нулевой выход S дешифратора. Инверторы при этом пропускают принятое слово без изменения.

Следует отметить, что цель рис. 4.11 — возможно проще проиллюстрировать закон, по которому определенные разряды информационного или кодового слова должны взаимодействовать с определенными элементами схемы при кодировании и декодировании. Физическое расположение разрядов в слове при его передаче или хранении может и отличаться от показанного на рис. 4.11, если это технически оправдано. Например, информационные разряды могут образовывать один компактный массив, контрольные — другой.

Для быстрого исправления одиночной ошибки код Хэмминга имеет минимально возможную избыточность: меньшего числа контрольных разрядов не хватит для непосредственного указания номера разряда, подлежащего исправлению. Пусть в общем случае длина информационного слова равна n разрядам, кодового — m разрядам, и синдрома ошибки — k разрядам. Чтобы закодировать k разрядами номер любого из m возможных неисправного разряда кодового слова, оставив при этом один из кодов синдрома — код «все нули» для идентификации отсутствия ошибки, должны выполняться соотношения

$$m = n + k; \quad m \leqslant 2^k - 1. \tag{4.18}$$

Код Хэмминга действительно реализует эти идеальные соотношения. Например, передаваемое кодовое слово длиной в 31 разряд ($m=31$) будет содержать 5 контрольных разрядов ($k=5$) и 26 информационных ($n=26$).

Если требуемая длина информационного слова n' меньше того максимального значения, которое может обеспечить некоторое k, т. е.

$$m' = n' + k < 2^k - 1,$$

то число входов схем свертки по четности, естественно, уменьшается. При этом неиспользуемыми не обязательно должны быть именно старшие разряды наибольшего допускаемого числом k кодового слова. Могут быть использованы любые n' его разрядов. При этом изменяется закон дешифрации синдрома ошибки, но, кроме того, может

улучшиться способность кода обнаруживать какой-то класс кратных ошибок.

Избыточность рассмотренного варианта кода Хэмминга достаточна лишь для исправления одиночной ошибки. Если ошибок, например, две, то схема проведет коррекцию по тем же формальным правилам, но результат этой коррекции будет уже неверный: одна ошибка просто превратится в другую. Существует модифицированный код Хэмминга, способный исправлять одиночную и обнаруживать двойную ошибки. Код строится из уже рассмотренного варианта кода Хэмминга добавлением к нему еще одного, $(m+1)$-го разряда P контроля по нечетности всего m-разрядного слова. Ситуацию анализирует логическая схема по следующим правилам:

$P=0$, синдром $K=0$ — ошибки нет;

$P\neq0$, синдром $K\neq0$ — одиночная ошибка, подлежащая исправлению;

$P=0$, синдром $K\neq0$ — двойная ошибка (или четная групповая). Исправить ее невозможно. Слово использовать нельзя;

$P\neq0$, синдром $K=0$ — групповая ошибка нечетной кратности. Слово использовать нельзя.

Как и в случае простейшего контроля по четности, при использовании кодов Хэмминга с целью обнаружения ошибки «все нули» результаты сверток при получении контрольных разрядов инвертируют. Именно такое решение показано на схемах, изображенных на рис. 4.11, *б* и *в*. Еще раз отметим, что хотя модифицированный код Хэмминга и обнаруживает некоторые кратные ошибки, но среди ошибок, кратность которых больше двух, найдутся такие, которые этот код не обнаруживает, т. е. или просто пропускает, или выполняет неверную коррекцию. Поэтому коды Хэмминга применяются там, где требования к достоверности данных высоки, а с учетом схемы устройства в нем наиболее вероятна именно одиночная и уже совсем редко — двойная ошибка. Типичный пример — полупроводниковая оперативная память, построенная из одноразрядных микросхем, вероятность сбоев в которых на сегодня довольно высока. Если же используются 4-разрядные микросхемы памяти, где вероятен сразу 4-разрядный *пакет ошибок*, то применение кода Хэмминга становится неэффективным. Для уверенного обнаружения пакетов ошибок, а тем более для их исправления нужны коды с еще большей корректирующей способностью, а следовательно, и с большей избыточностью. Об основных принципах построения схем, используемых при работе с такими кодами, говорится в § 10.3 и 10.4.

Подробнее о кодах Хэмминга можно прочесть в [29, 30, 60]. При-

мером изделия, использующего одну из модификаций кода Хэмминга, может служить микросхема К555ВЖ1, предназначенная для формирования контрольных разрядов и синдрома ошибки, для исправления одиночной и обнаружения двойных, тройных и ряда групповых ошибок в 16-разрядных словах, хранимых в памяти. Микросхема подключается к магистралям памяти, обрабатывает числа при их записи и считывании.

4.10. Схемы контроля логических преобразований

Помехоустойчивые коды контролируют правильность лишь неизменяемых данных. Если данные подвергать логической обработке, то коды, контролирующие правильность результата, становятся очень сложными. Петерсоном в 1959 г. была доказана теорема о том, что число контрольных разрядов кода при произвольном кодовом преобразовании не может быть меньше числа информационных разрядов этого кода. Это значит, что при произвольных кодовых преобразованиях может и не существовать более дешевого метода контроля выполняемой операции, чем дублирование аппаратуры и сравнение результатов. Для экономно спроектированных одновыходных логических схем это всегда так, и простейший способ контроля их дублированием показан на рис. 4.12, *а*.

Сбои в работе схем могут вызываться не только дефектами, присущими самим схемам, но и вредными внешними воздействиями, например помехами по питанию. В этом случае если дублирующие друг друга схемы строго одинаковы, то велика вероятность одинаковой их ошибочной реакции, в результате сбой обнаружен не будет. С этой точки зрения дублирующие друг друга узлы нужно выполнять по отличающимся схемам. Простейший метод построения логических узлов, различных по схеме и применяемым элементам, но реализующих одну и ту же функцию, это создание взаимнодвойственных схем. Такое решение показано на рис. 4.12, *б*, и оно встречается на практике.

Если информация об ошибках в отдельных схемах не используется непосредственно на месте, а собирается и обрабатывается в каком-то едином контролирующем центре, для чего передается на существенные расстояния и коммутируется логическими схемами, то в этом случае надежность всей системы будет определяться не только надежностью контролируемых схем, но и безошибочной работой контролирующих схем. Способом защиты контролирующих

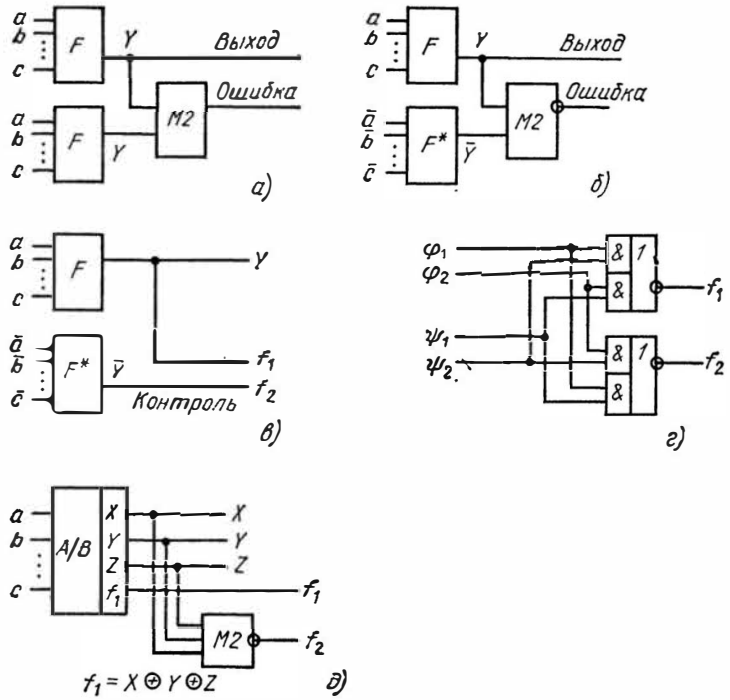


Рис. 4.12. Способы контроля комбинационных схем:

а — простое дублирование; *б* — дублирование схемы F двойственной ей схемой F^*; *в* — передача результатов контроля в форме контрольных функций; *г* — сжатие контрольных функций, полученных от двух источников; *д* — контроль кодового преобразователя

схем от одиночной ошибки является представление и передача сигнала о правильном функционировании проверяемого узла парафазным кодом, как показано на рис. 4.12, *в*.

Парафазным кодом называют способ передачи информации сразу по двум проводам так, что когда по одному проводу передается некоторый сигнал, то по другому — всегда его инверсия. При сложении по модулю 2 в контролирующем центре контрольных функций f_1 и f_2 нулевой результат говорит об ошибке или в контролируемой схеме, или в цепях передачи контрольной информации, поскольку и в том, и в другом случае будет нарушено свойство нечетности, присущее парафазному коду.

Контрольную информацию о работе нескольких узлов

можно подвергать сжатию с помощью схемы, показанной на рис. 4.12, *г*. Получаемые на выходе этой схемы функции f_1 и f_2 при правильной работе всегда взаимно инверсны. Совпадение их значений говорит или об одиночной ошибке в одном из проверяемых узлов, о которой сигнализирует совпадение значений обеих фаз аналогичных входных функций φ и ψ, или об ошибке в самой схеме сжатия. Функции f_1 и f_2 могут подвергаться дальнейшему сжатию вместе с другими подобными функциями. Вычисленное на элементе $\overline{M2}$ последней инстанции системы контроля значение свертки пары функций f_1 и f_2 является сигналом об ошибке или в блоке, контролируемом данным деревом, системы контроля, или в самом этом дереве. Такой сигнал можно выводить на контрольную панель блока, пульт оператора, вход устройства, принимающего решения в аварийных ситуациях, и т. д.

Комбинационные схемы, реализующие систему булевых функций (кодовые преобразователи), в отличие от одновыходных схем можно контролировать, как показано на рис. 4.12, *е*, т. е. не прибегая к их дублированию. При этом кодовый преобразователь усложняется всего на одну добавочную функцию f_1. Она задается как функция аргументов, значение которой совпадает со значением свертки по модулю 2 всех остальных функций. В качестве примера в табл. 4.3 повторена табл. 3.3 преобразователя кода светофора с добавлением столбца контрольной функции $f_1 = Z \oplus J \oplus K$.

Таблица 4.3

A		B			
a_2	a_1	Z	J	K	f_1
0	0	1	0	0	1
0	1	0	0	1	1
1	0	0	1	1	0
1	1	0	0	0	0

Второй, противофазной, контрольной функцией f_2 является выполненная на элементе $\overline{M2}$ свертка по четности всех фактических выходов преобразователя, кроме f_1 (см. рис. 4.12, *д*). Одиночная ошибка в работе как схемы преобразователя, так и схемы свертки изменит значение четности одной из функций f_1 или f_2 и будет обнаружена

схемой контроля. Легко видеть, что такой метод контроля кодовых преобразователей по своему принципу очень похож на контроль по четности передаваемых данных.

Функция f_1 при реализации кодового преобразователя должна синтезироваться непосредственно из входных переменных a, b ... Ее ни в коем случае нельзя строить «экономичным» способом, складывая по модую 2 уже построенные основные функции Z, I, K ... В противном случае ошибка в отработке схемой преобразователя любой из этих функций изменит значения сразу и f_1, и f_2 и, следовательно, не будет обнаружена.

Высказанное предостережение носит более глубокий характер. Если в схеме преобразователя есть участки логической обработки, общие для каких-нибудь хотя бы двух выходов, то для произвольной схемы в принципе может найтись неисправность в этой общей части, изменяющая значения сразу двух (или четного числа) выходов преобразователя, и, следовательно, не обнаруживаемая узлом свертки по четности. Поэтому при данном способе контроля многовыходных логических схем нужно или использовать многовыходные логические схемы обязательно без общих частей, или для получения парафазных функций f_1 и f_2 придумать вместо свертки по модулю 2 какие-то другие логические функции. Анализ этих возможностей и подробное изложение методов синтеза самопроверяемых логических схем содержатся в [31, 60]. Еще один путь, по которому часто идут разработчики, это вероятностный подход к работе системы контроля, когда снимается требование обнаружить абсолютно все одиночные сбои. Это оправдано, поскольку в цифровой аппаратуре наряду с одиночными распространены и групповые, взаимозависимые сбои и отказы, поэтому обнаружение всех одиночных сбоев все равно еще не дает абсолютной гарантии правильности работы аппаратуры.

ГЛАВА 5

ПЕРЕХОДНЫЕ ПРОЦЕССЫ. ГОНКИ

5.1. Переходные процессы в логических схемах

Задержка логической схемы слагается из задержек срабатывания логических элементов и задержек распространения сигналов по цепям связи между ними. Трудоемкость

учета задержек зависит от соотношения значений задержек самих логических элементов и задержек в цепях связи. Если эти значения близки, то задержки различных трактов схемы можно определить лишь после размещения элементов на поверхности платы или кристалла БИС, когда станут известны фактические длины связей. Если при этом задержки некоторых цепей не соответствуют требуемым, то нужно или переставлять элементы, или даже вносить изменения в функциональную схему, снова трассировать связи и снова определять задержки в них. Процесс становится итерационным, длительным. Именно в таком положении оказываются разработчики аппаратуры на быстрых элементах ЭСЛ, устанавливаемых на платах в виде микросхем или изготавливаемых прямо на поверхности кристаллов БИС. Подходы к оценке задержек в схемах такого класса изложены в [13, 32]. Сложность учета задержек — одна из причин, препятствующих широкому распространению элементов ЭСЛ в схемах цифровой автоматики.

В цифровой автоматике в основном используются элементы с временем переключения не менее 20 нс, что примерно на порядок превышает задержку распространения сигнала в любом проводе монтажной платы типового размера. Паразитная емкость монтажа при использовании типовых плат также не настолько велика, чтобы существенно изменить задержку элемента. В этих случаях задержку внутриплатного и близкого межплатного монтажа рационально не учитывать отдельно, а, оценив худший случай, включить ее в состав задержки логического элемента. Небольшая потеря потенциально достижимого быстродействия с лихвой окупается упрощением разработки схем, поскольку задержки могут быть учтены без каких-либо итераций, сразу, и притом уже на этапе логического проектирования. *Технические этапы проектирования — размещение* элементов и *трассировка* связей — выполняются тоже только один раз и не вызывают необходимости корректировать функциональные схемы. Учитывая сказанное, в дальнейшем будем предполагать, что задержки в цепях связи включены в состав задержек логических элементов.

Ситуации, когда задержки в связях превышают задержки в элементах, возникают и при использовании не очень быстродействующих элементов — когда сигналы передаются между блоками на достаточно большое расстояние. Однако доля подобных связей невелика, поэтому их можно выделить особо и учесть задержку в кабеле.

Задержки различных экземпляров элементов какого-то определенного типа имеют технологический разброс, который обычно описывают некоторым статистическим законом. Кроме того, задержка каждого конкретного элемента зависит от его температуры, длительности фронта входного сигнала, от того, на сколько элементов и притом каких он нагружен, от паразитной емкости монтажа, числа лет с момента выпуска и еще ряда других факторов. В паспортах элементов некоторых серий влияние части этих факторов учитывается дифференцированно в виде графиков, таблиц, линеаризованных зависимостей, но чаще это влияние просто оценивается по максимуму. При этом паспортные значения задержек и фронтов приводятся для худшего случая, который может встретиться при соблюдении указанных в паспорте ограничений. В первом случае удается полнее использовать возможности элемента, во втором — упрощается проектирование.

На рис. 5.1 показан возможный вид кривой технологического разброса задержки элементов при испытаниях на

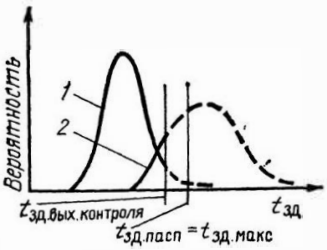
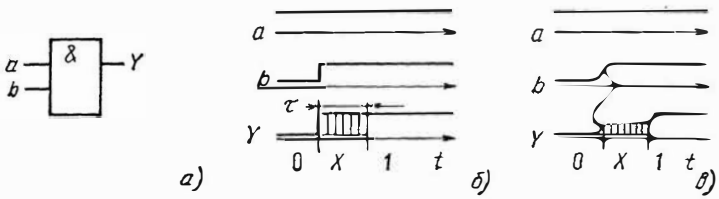


Рис. 5.1. Плотность вероятности распределения задержки элемента в условиях налаженного производства *1* и в период освоения *2*

Рис. 5.2. Способы изображения состояния неопределенности логического элемента



предприятии-изготовителе. Выходной контроль отсекает хвост кривой в соответствии с ТУ на элемент с учетом необходимого запаса на старение, допуски и т. п. Если правильно налажены производство и контроль, то потребитель

всегда имеет дело с элементами, задержка которых не превышает паспортную.

Однако, как будет видно из дальнейшего, разработчику иногда весьма полезно знать кроме максимальной еще и минимально возможную задержку. К сожалению, для большинства серийно выпускаемых микросхем значение минимальной задержки в ТУ не указано и, следовательно, изготовителем не гарантируется. Опыт работы схемотехника с данными элементами здесь также бесполезен, поскольку кривая технологического разброса у разных изготовителей различна и к тому же чувствительна к перестройкам производства, что и иллюстрируют две кривые на рис. 5.1. Поэтому если разработчик аппаратуры, предназначенной для серийного выпуска, использует микросхемы, в паспорте которых не оговорено минимальное значение задержки, то он вынужден полагать минимальное время задержки равным нулю. Никаких юридических оснований считать, что это значение больше нуля, у него нет.

Уровень выхода элемента в течение отрезка времени от минимально возможного до максимально возможного значения задержки, когда фактическое состояние выхода элемента разработчику не известно, называют *состоянием неопределенности* и обозначают символом x. Состояние x, поступая на входы других логических элементов, может в зависимости от типа элемента порождать на их выходе или определенные состояния 1 или 0, или также неопределенное, обозначаемое X. Поведение логических элементов задается при этом законами уже не двоичной, а одного из видов троичной логики. Соотношения, расширяющие основные функции на третью переменную x, достаточно очевидны:

$$\begin{aligned} &\bar{x} = X;\ x \cdot 0 = 0; \quad x \cdot 1 = X; \quad x_1 \cdot x_2 = X; \\ &x \vee 0 = X; \quad x \vee 1 = 1; \quad x_1 \vee x_2 = X; \\ &x \oplus 0 = X; \quad x \oplus 1 = X; \quad x_1 \oplus x_2 = X, \end{aligned} \tag{5.1}$$

где x, x_1, x_2 — неопределенные значения сигналов на входах элементов.

Эффективным средством анализа переходных процессов в схемах являются временные диаграммы. При их построении состояние неопределенности изображают одним из двух способов, которые показаны для элемента И (рис. 5.2, *а*). Изображение на рис. 5.2, *б* строже, но менее наглядно; изображение на рис. 5.2, *в* нагляднее, но может быть спутано с состоянием высокого импеданса элемента, имею-

щего три состояния выхода. Линии со стрелками обозначают причинно-следственные отношения в цепочке переключений. Линия начинается на фронте, который непосредственно вызывает переключение рассматриваемого элемента и оканчивается стрелкой на фронте выходного сигнала этого элемента. Наличие таких указателей заметно облегчает понимание работы сложных схем.

На рис. 5.3 показан фрагмент схемы (*а*) и варианты начертания временны́х диаграмм переходных процессов. Здесь и в дальнейшем для обозначения выходного сигнала элемента используется номер самого элемента. Диаграмма на рис.

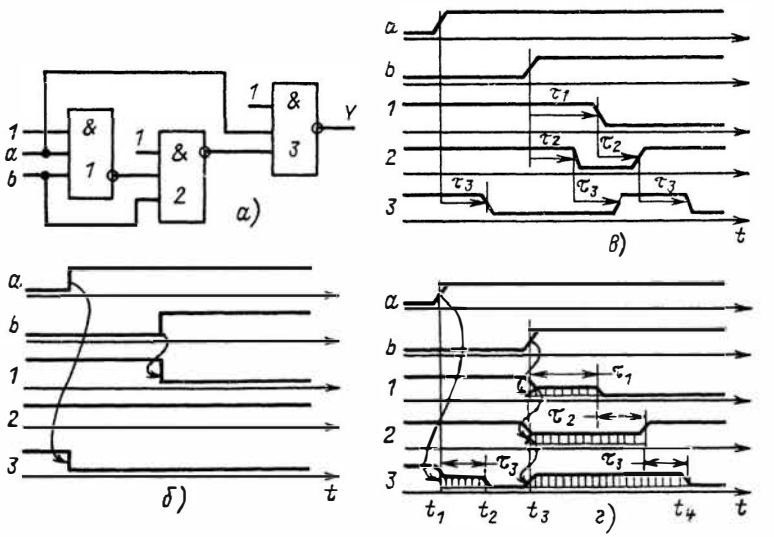


Рис. 5.3. Временны́е диаграммы переходных процессов:
а — фрагмент схемы; *б*, *в*, *г* — изображение переходных процессов; *б* — без учета задержек элементов, *в* — в предположении, что задержки максимальны; *г* — с использованием состояния неопределенности

5.3, *б* игнорирует переходные процессы в элементах и схеме. Такие диаграммы применяют, когда основной целью является иллюстрация логических и причинно-следственных отношений, а длительностью переходных процессов по сравнению с интервалами между поступлением сигналов можно пренебречь.

Диаграмма на рис. 5.3, *в* построена в предположении, что все элементы имеют максимально возможные значения задержки. Эта диаграмма наглядна, поэтому удобна для

первого знакомства с поведением сложной схемы. Но она годится лишь для оценки максимальной длительности переходного процесса. Делать по такой диаграмме выводы о состояниях элементов во время переходного процесса непозволительно: это лишь один частный случай из множества возможных процессов.

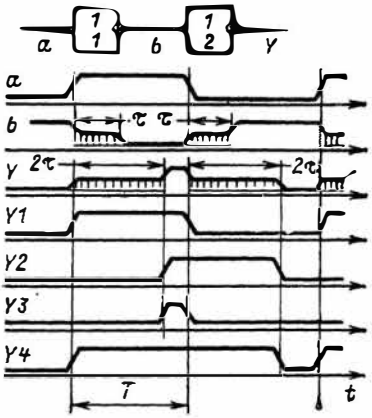
Диаграмма рис. 5.3, *г* учитывает состояния неопределенности элементов в соответствии с (5.1). Она достаточно строго моделирует поведение схемы при любых комбинациях задержек, допускаемых паспортами элементов. Полезно сравнить диаграммы на рис. 5.3, *в* и *г*, обращая внимание на их расхождения, причиной которых является общность диаграммы *г* и частность диаграммы *в*. До момента t_1 и после момента t_4 обе диаграммы совпадают.

Быстрое чтение и особенно построение временны́х диаграмм требуют некоторой тренировки. Полезно самостоятельно построить несколько вариантов диаграмм, хотя бы изменяя моменты поступления входных сигналов и соотношения задержек элементов схемы на рис. 5.3, *а*. Построение диаграммы нужно начинать с тех элементов, для которых известны все входные сигналы, в данном примере — с элемента *1*. После определения выхода элемента *1* известными становятся все входы элемента *2* и т. д. Если построение диаграммы с учетом состояния неопределенности вызывает затруднение, можно рекомендовать сначала построить диаграмму с нулевыми задержками типа показанной на рис. 5.3, *б*, затем на том же чертеже наложить на нее диаграмму с максимальными задержками, после чего интервалы состояний неопределенности выделяются намного легче.

В организациях, специализирующихся на разработке логических схем, построение и анализ временны́х диаграмм выполняются на ЭВМ с помощью специальных моделирующих программ. При этом для выявления некоторых тонких случаев неопределенности используют не только троичное, но и пятеричное, и более сложные формы представления ситуаций во время переходных процессов. Эти подходы и принципы построения моделирующих программ изложены в [32 и 33].

Введение состояния неопределенности позволяет выявить важный, хотя и не очевидный с первого взгляда эффект, который всегда нужно учитывать. На рис. 5.4 показана цепочка из двух элементов, на вход которой поступает сигнал в виде единичного импульса длительностью T. У выходного сигнала в его начале и конце будут зоны неопределенности

Рис. 5.4. Изменение задержки и длительности импульса при прохождении его по цепочке элементов



длительностью по 2τ каждая. В частном случае при $\tau^{10} = \tau^{01} \approx 0$ на выходе будет сигнал *Y1*, почти повторяющий входной. Однако если задержки включения и выключения равны и максимальны, то полученный сигнал *Y2*, будет сдвинут относительно выходного на 2τ. В результате может оказаться, что один и тот же сигнал, переданный по двум цепочкам на два блока устройства, запустит их не одновременно. Понятие одновременности расплывается и становится относительным. Если задержки включения существенно отличаются от задержек выключения, получится укороченный на 2τ (*Y3*) или удлиненный на 2τ (*Y4*) сигнал. В случае *Y4* укороченной окажется пауза между последовательными импульсами. Могут получиться и любые промежуточные формы рассмотренных частных случаев, причем предугадать характер эффекта заранее невозможно. Если цепочка содержит k элементов, то во всех рассмотренных случаях вместо двойки в качестве множителя при τ войдет k. У разработчика нет никаких официальных документов, позволяющих проигнорировать любой из возможных эффектов, и он вынужден проектировать схему так, чтобы ни один из них не привел к сбою в работе. Если на выходе цепочки требуется получить импульс с минимальной длительностью T, то длительность импульса на входе цепочки должна быть на $k\tau$ больше. Аналогично нужно обеспечивать на выходе цепочки и минимальную длительность паузы, и максимальную длительность импульса, если это требуется. Двустороннего допуска на длительность импульса, более строгого, чем $\pm k\tau$, требовать нельзя. О том, как обеспе-

чить нечувствительность схемы к некоторому расхождению одновременных событий, говорится в гл. 7.

5.2. Гонки

В логических схемах встречаются участки, где сигнал разветвляется, получившиеся два сигнала распространяются по двум независимым цепочкам элементов, а затем оба сигнала снова встречаются на входах одного элемента. Подобная ситуация показана на рис. 5.5, *а*, где в рассматриваемый момент времени в представленном фрагменте схемы два тракта оказались прозрачными для входного сигнала благодаря тому, что все конъюнкторы фрагмента в этот момент открыты сигналами единичного уровня. Пусть в тракте *чет* четное число инверторов, а в тракте *нечет* — нечетное. Анализ подобной схемы методами алгебры Буля без учета задержек даст на ее выходе 0 при любом значении входного сигнала (рис. 5.5, *б*). Но реальные элементы имеют конечную задержку срабатывания, и если обозначить задержки в трактах *чет* и *нечет* через $T_{чет}$ и $T_{нечет}$, то в зависимости от соотношения этих величин получится один из процессов, изображенных на рис. 5.5, *в* и *г*. В обоих случаях в выходном сигнале появится помеха, не предусмотренная булевыми выражениями. Легко проверить, что замена последнего элемента И на элемент ИЛИ не ликвидирует помеху, а лишь проинвертирует ее и изменит момент появления.

Существенно, что полученная помеха — это не пренебрежимо короткий всплеск напряжения малой амплитуды. При достаточно большой разности $T_{чет}$ и $T_{нечет}$ помеха будет иметь длительность, во много раз превышающую время переключения элемента, и амплитуду, равную номинальному сигналу. Это уже полноценный логический сигнал, на который могут реагировать последующие элементы. Если выход схемы подключен к запоминающему элементу (триггеру), то помеха может запомниться и будет влиять на последующие процессы в устройстве. Если выход схемы подан в качестве обратной связи на вход, там появится непредвиденный сигнал, который может вызвать непредвиденное повторное срабатывание этой же схемы.

Описанное явление называют *гонками* или *состязаниями* (*races*). Два сигнала идут разными путями, и схема может реагировать на них по-разному (верно или неверно) в зависимости от того, какой сигнал выиграет гонку.

Основная проблема в том, что разработчик, как прави-

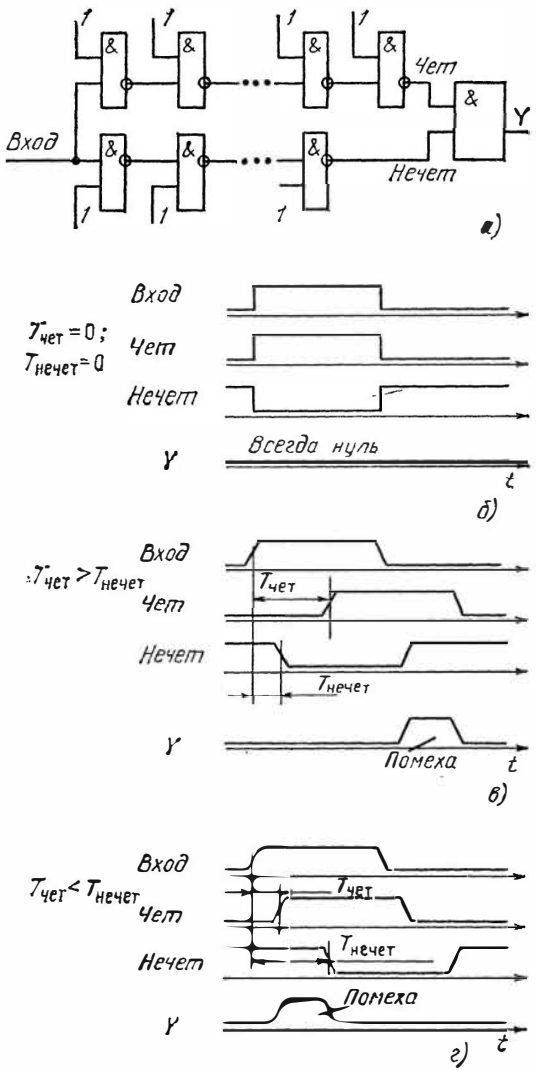


Рис. 5.5. Варианты временны́х диаграмм (*б, в, г*), иллюстрирующие гонки в логической схеме (*а*)

ло, не знает, в каком тракте задержка окажется меньше. Изготовитель элементов гарантирует лишь максимальное время задержки элемента данного типа и ничего не говорит

ни о конкретной задержке конкретного элемента, ни о минимально возможном времени задержки. Поэтому разработчик логических схем не может воспользоваться тем фактом, что число элементов в цепочке *чет*, скажем, больше, чем в цепочке *нечет*: при массовом производстве схем из произвольно взятых элементов найдется достаточно большое число таких узлов, в которых в цепочку *чет* попадут более быстрые элементы, а в цепочку *нечет* — более медленные и вопреки ожидаемому будет выполняться неравенство $T_{чет} < T_{нечет}$. Даже если в цепочке *нечет* один элемент, а в цепочке *чет* — два, то в последнюю вполне могут попасть элементы, имеющие время задержки втрое меньшее, чем элемент цепочки *нечет*.

Специальный подбор элементов по задержке в условиях современного автоматизированного массового производства недопустим, проверка реально получившегося соотношения задержек обычно неприемлема, так как сильно удорожает наладку аппаратуры. Кроме того, при изменении температуры и старении задержки различных элементов изменяются с разной скоростью, и по этому поводу изготовитель, как правило, никаких гарантий не дает. Единственное, что гарантирует изготовитель элементов и на что может опереться разработчик схем (фактически или хотя бы юридически), это то, что задержка не выйдет за пределы, указанные в ТУ на элемент, и если в борьбе с гонками разработчик хочет одержать победу, то он должен основывать свои расчеты лишь на этих данных.

Распространены три метода борьбы с гонками: введение тактирования, построение противогоночных схем и учет минимального времени задержки. Наиболее универсальным, эффективным и поэтому широко используемым методом борьбы с гонками является *тактирование*. Основная суть его заключается в следующем. По всему цифровому устройству разводится единая система тактирующих (синхронизирующих) сигналов. В широко распространенной *двухтактной* или *двухфазной* системе синхронизации используются две периодические последовательности *синхросигналов — синхросигнал С1* и *синхросигнал С2*. Взаимное расположение этих сигналов во времени показано на рис. 5.6, *б*.

Схема (рис. 5.6, *а*) разделена штриховой линией на две части. Левая принимает и обрабатывает сигнал *ВХОД*: ее выходной сигнал *У1* является входным для схемы правой части, которая запоминает результат в триггере *Тг*.

Если сигнал *ВХОД* каким-либо образом «привязан» к одной из синхросерий, например к *С1* (способы такой привязки рассматриваются в гл. 7), то этот сигнал будет изменяться только в момент поступления синхроимпульсов *С1*, а в промежутках между ними будет оставаться посто-

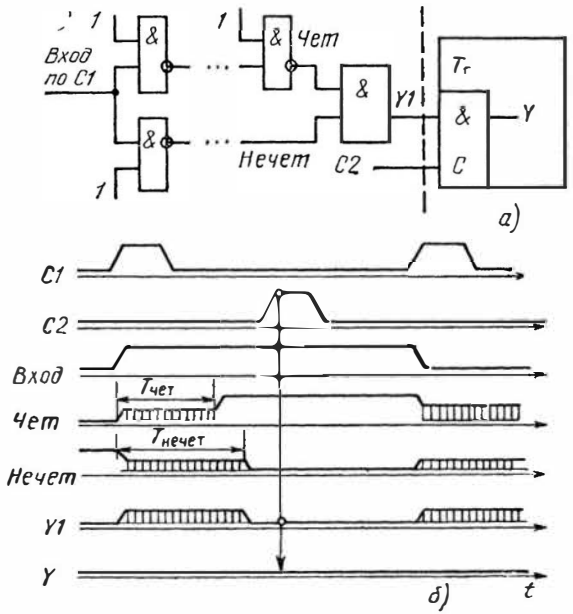


Рис. 5.6. Исключение помех, порожденных гонками, за счет тактирования

янным. Схема, показанная в левой части рис. 5.6, *а*, имеет параллельные пути, в ней существуют гонки и возможно появление на выходе *Y1* ложных сигналов. На рис. 5.6, *б* эту ситуацию в общем виде отражают интервалы неопределенности трактов *чет* и *нечет*. На входной конъюнктор *С* правой части схемы подается сигнал синхросерии *С2*. Обязательным условием является такой временной сдвиг *С2* по отношению к *С1*, который превышает самый длинный интервал неопределенности, т. е. самую большую задержку из всех параллельных трактов схемы. Это значит, что сигнал *С2* откроет конъюнктор *С* заведомо п о с л е окончания всех переходных процессов в схеме и пропустит логически правильное, не искаженное гонками установившееся значе-

ние функции *Y1*. Как правило, конъюнктор *С* вводят в состав триггера, что делает триггер *синхронным*. Такой триггер переключается только по команде синхросигнала и не воспринимает информацию при его отсутствии. Конъюнктор *С* применяют и без триггера — в тех случаях, когда запоминать выходной сигнал схемы не требуется, а нужно лишь очистить его от помех, порожденных гонками. Тогда сигнал, открывающий конъюнктор *С*, обычно называют не синхросигналом, а *стробом,* а сам процесс отсечки помех — *стробированием.*

Система синхронизации едина для всего цифрового устройства, и интервал между синхросигналами задается в начале разработки. Схемотехник, разрабатывая каждый фрагмент логической схемы, так подбирает число последовательно включаемых в нее элементов и их типы, чтобы все переходные процессы в этом фрагменте с гарантией закончились к моменту поступления очередного синхросигнала. Достоинство синхронизации как средства борьбы с гонками в том, что разработчику не требуется вникать в специфику протекания переходных процессов, в характер возникающих гоночных ситуаций, не нужно знать минимального значения задержки и т. д. Все, что должен знать разработчик, — это максимально возможную задержку самого длинного тракта логической схемы, а это легко вычисляется по паспортным данным используемых элементов. Подробнее системы тактирования будут рассматриваться в гл. 7.

Противогоночные схемы — это схемы, построенные так, что в них если и возникают, то только неопасные гонки, т. е. такие, при которых отсутствует *риск* появления на выходе сигналов, не предусмотренных логическим выражением. Примером неопасной гонки может служить гонка фронта по двум трактам, в каждом из которых содержится четное (или в каждом нечетное) число инверторов и которые объединяются на выходах элементом ИЛИ: кто бы ни выиграл гонку, результат все равно будет верный, изменится лишь задержка его получения. Для исключения опасных гонок можно вводить в схему дополнительные связи и элементы так, чтобы нежелательные параллельные пути запирались самим входным сигналом еще до достижения им опасной развилки тракта. Есть целый ряд других приемов, используемых при построении противогоночных схем. О них можно прочитать в [33], где дается подробная классификация типов гонок и обширная библиография.

Полезным свойством противогоночных схем оказывается их способность обрабатывать данные по мере поступления, *асинхронно,* т. е. без привязки к тактирующим сигналам и связанным с этим потерям времени. Однако процедура построения таких схем очень сложна, она требует скрупулезного изучения характера протекания переходных процессов, выявления всех возможных гоночных путей, отделения опасных состязаний от неопасных и т. д. С ростом числа элементов схемы разработчик очень скоро оказывается в плену невыполнимого по объему перебора вариантов. Специалистами разработан ряд удачных, остроумных асинхронных противогоночных схем, в основном — сложных типов триггеров с числом элементов в пределах десятка. Они широко применяются, в том числе и в тактируемых устройствах, для быстрого выполнения операций внутри самого такта, и некоторые из них в дальнейшем будут рассматриваться. Однако рекомендовать начинающему разработчику использовать этот метод борьбы с гонками для схем, создаваемых самостоятельно, автор не решится.

Учет минимального времени задержки. Если известно минимально возможное время задержки элемента, то во многих практически важных случаях можно постулировать отсутствие гонок. Пусть в схеме на рис. 5.5, *а* глубина цепочки *Чет* настолько больше глубины цепочки *Нечет*, что задержка в длинной цепочке, даже если последняя состоит только из самых быстрых элементов, будет все равно больше задержки сигнала в короткой цепочке, даже если в нее попадут только самые медленные элементы. Схема со столь большой разницей в длине путей всегда будет вести себя так, как показано на рис. 5.5, *в*, т. е. пока входной сигнал равен единице, помеха на выходе не появится. Помеху после выключения входного сигнала можно ликвидировать введением дополнительной блокирующей связи. Можно, например, взять в качестве выходного элемента трехвходовой элемент и на его третий вход подать сам входной сигнал. В этом случае выход будет заперт сразу после перехода входного сигнала в нуль. Несложно предложить и другие формы использования для борьбы с гонками сведений о минимально возможной задержке или о наибольшей возможной кратности максимального и минимального значений задержки.

По возможностям применения этого метода борьбы с гонками разработчики, использующие различную элементную базу, находятся в не-

одинаковых условиях. В лучшем положении обычно находятся разработчики схем, предназначенных для реализации на поверхности кристалла. Результирующая задержка элементов на кристалле определяется длиной и шириной проводников межэлементных связей и временем переключения транзисторов, которое в свою очередь зависит от геометрических размеров элементов его маски. На все эти размеры разработчик схемы кристалла в принципе может влиять, что позволяет ему в случаях, когда это важно, делать задержку одной группы элементов гарантированно больше задержки другой группы. Правда, по мере роста числа элементов, размещаемых на одном кристалле, разработчик логических схем все реже допускается к воздействию на геометрические размеры элементов маски. Технология автоматизированного проектирования схем на перспективных матричных кристаллах разрешает схемотехнику лишь соединять между собой проводниками стандартной ширины уже размещенные на кристалле стандартные транзисторы или логические элементы. Однако и здесь положение разработчика все-таки лучше, чем того, который использует отдельные микросхемы, поскольку задержки однотипных элементов, расположенных на одном кристалле, существенно между собой коррелируют, чего нельзя сказать о микросхемах даже одной закупочной партии, одного изготовителя и т. д. Эта корреляция позволяет изготовителю кристаллов после проведения соответствующих исследований постулировать для схемотехника максимально возможную кратность задержек элементов, расположенных на одном кристалле данного типа. С точки зрения борьбы с гонками это во многих случаях не хуже, чем знание минимально возможной задержки. Поэтому внутри интегральных схем метод борьбы с гонками за счет назначения параллельными путями таких соотношений задержек, при которых опасные гонки невозможны, используется, особенно при построении небольших узлов — типа триггеров, счетчиков и т. п.

В худшем положении находится разработчик, использующий готовые микросхемы, поскольку юридического документа о минимальном значении задержки он чаще всего, к своему сожалению, не имеет. Правда, и тут опытный инженер может утверждать, что при использовании любой современной серии элементов и при любом их сочетании пробег сигнала по цепочке из, скажем, 64 элементов длится «наверняка» дольше, чем пробег сигнала по параллельной ветви из одного элемента. На сегодня нет серий, задержка элементов внутри которых отличалась бы в 64 раза. И в 32 раза тоже нет. И в 16, пожалуй, не найдется. Относительно восьми можно задуматься, в защиту четырех большинство специалистов серьезно спорить уже не станет, а отклонение времени задержки вдвое встретится в большинстве серий. Таким образом, если отсутствие гонок обосновывается большой кратностью числа элементов параллельных путей, то нужно отдавать себе отчет в том, что есть

зоны явно допустимых решений (например, кратность 64) и зоны явно недопустимых (кратность 2), а граница между ними не определена. Каждый разработчик определяет ее для себя индивидуально в зависимости от опыта, соотношения поощрения за экономичную схему и наказания за сбои в ней из-за гонок, от своего темперамента и других факторов. Проблема из технической становится психологической, организационной... В инженерной практике так или иначе пользуются этим приемом и строят схемы, в которых в принципе, юридически, гонки возможны, но по утверждению разработчика их «наверняка» не будет. Рекомендовать этот прием молодым схемотехникам не следует: у них, как правило, темперамент превалирует над опытом.

В литературе при описании схем, к сожалению, не принято называть примененных способов борьбы с гонками. Если в публикуемой схеме эта проблема решалась путем соответствующего подбора задержек, то при изменении технологического процесса схема может стать совершенно неработоспособной. Об этом нужно помнить при слепом копировании «хорошо зарекомендовавшей себя на практике» схемы.

В последние годы растет интерес к еще одному методу борьбы с гонками — построению *самосинхронизирующихся схем*. Рабочие узлы в этом случае строятся непротивогоночными, но они дополняются специальными схемами, которые обнаруживают факт окончания переходных процессов и вырабатывают разрешающий сигнал для следующих схем, играющий в каком-то смысле роль «асинхронного синхросигнала». Это направление рассматривается как весьма перспективное для построения БИС и особенно сверхБИС, где применение обычной синхронизации встречает ряд трудностей. Однако в схемах и микросхемах обычного размера и технологии это направление пока не находит применения ввиду как сложности построения такого рода схем, так и, приблизительно, удвоения их аппаратурных затрат. Поэтому данная книга не касается вопросов построения самосинхронизирующихся схем, а интересующемуся читателю можно рекомендовать работу [34], написанную на высоком научном уровне коллективом, который уже более 15 лет успешно работает в этом направлении.

Проблема гонок в цифровой схемотехнике является очень серьезной. Без натяжки можно сказать, что большинство труднообнаруживаемых и удивительно разнообразно проявляющихся ошибок в функциональных схемах связано с гонками, возможности появления которых разработчик не заметил. Основная причина здесь — ограниченность поля внимания человека. При разработке сложной схемы все внимание поглощается конструктированием главного пути распространения сигнала, непосредственно

решающего поставленную задачу. При этом побочные, не нужные для дела пути выпадают из поля зрения, а в них-то и «таится погибель».

Гонки во вновь разработанной схеме нужно искать специально. Если есть такая возможность, то наиболее надежным и простым методом является моделирование работы схемы с помощью специальных программ (см., например, [32]). При поиске гонок «вручную» сначала нужно выявить все подозрительные места и затем методически их исследовать. Обнаружению таких мест способствуют специально предпринимаемые просмотры схемы, нацеленные на выявление всех возможных параллельных путей распространения сигнала. Полезен анализ временных диаграмм, в которых зоны неопределенности указывают на возможность появления ложных сигналов.

В борьбе с гонками наряду с излагаемым подходом, ориентированным на гарантированную работу даже при наиболее неблагоприятном сочетании задержек, существуют менее строгие походы, в основе которых лежит утверждение, что худший случай встречается редко, поэтому для оценки задержки схемы можно брать некоторое вероятностное значение, меньшее, чем максимально возможное время задержки. Подобные подходы, против которых в принципе возражать нельзя, для цифровой техники оказываются плодотворными, только если есть возможность легко провести абсолютно полное тестирование готового устройства во всех возможных режимах и условиях окружающей среды.

Если же надежность работы устройства основывается лишь на статистической правильности работы отдельных его цепей, то, когда цепей много, что типично для цифровой техники, даже небольшое уменьшение надежности срабатывания элементов приводит к резкому снижению надежности всего устройства. Легко подсчитать, что если в каждой цепочке допустить вероятность помехи из-за гонок всего в 1 %, то вероятность работоспособности устройства, содержащего 100 таких цепочек, будет около 37 %. В среднем из каждых трех устройств два будут неработоспособны. Поэтому приемами, приносящими в жертву надежность срабатывания отдельных цепочек, в цифровой технике пользоваться не следует.

5.3. Гонки по входу

Гонки по входу возникают, когда ветвящийся сигнал поступает на элементы, имеющие разброс по уровню срабатывания (рис. 5.7, *а*), а фронт этого сигнала излишне пологий (рис. 5.7, *б*). Если длительность фронта входного сигнала заметно больше времени срабатывания элементов, то

где-то в середине фронта будет существовать отрезок времени, когда с точки зрения одного элемента входной сигнал уже равен 1, а с точки зрения другого — еще равен 0. Элементы будут реагировать на один и тот же сигнал как на два различных, а такая ситуация при проектировании схемы ее алгоритмом не предусматривается. В результате

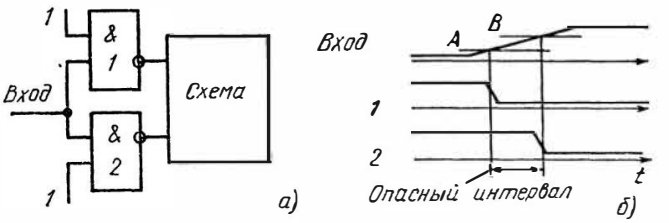


Рис. 5.7. Гонки по входу: иллюстрация условий их возникновения

схема в течение этого времени может выработать ложные сигналы. Это явление и называют «гонки по входу». Гонки по входу не наблюдаются, если логическая схема собрана на элементах одной серии микросхем. Потенциально опасны с этой точки зрения схемы, собранные из элементов различных серий, имеющих одинаковый уровень сигналов, но существенно различные времена задержек и фронтов. Гонки по входу возникают в схемах некоторых БИС, если их межэлементные связи сильно заваливают фронты. Обычно появление таких гонок и при приеме внешних сигналов, источниками которых могут быть более медленные элементы, вплоть до электромеханических, поэтому внешние сигналы должны проходить специальную обработку, описанную в гл. 8. Опасностью возникновения гонок по входу объясняются ограничения на максимальную длительность фронтов входных сигналов, приводимые в паспортах многих микросхем.

Если нет возможности увеличить крутизну фронта, то единственным средством борьбы с гонками по входу остается тактирование, поскольку в тактированном устройстве выходной сигнал схемы не используется до тех пор, пока в этой схеме не окончатся абсолютно все переходные процессы независимо от их физической природы. Однако тактирование не спасает от гонок по самому тактирующему входу. Поэтому, если крутизна фронтов синхросигналов мала, то нужно применять такие синхронные триггеры, в которых гонки по входу не возникают.

ГЛАВА 6

ТРИГГЕРЫ

6.1. *RS*-триггер

Триггером называют логическую схему с положительной обратной связью, имеющую два устойчивых состояния, которые называются *единичным* и *нулевым* и обозначаются 1 и 0. Перевод триггера в единичное состояние путем воздействия на его входы называют *установкой* (*set*) триггера, а устанавливающий сигнал и вход, на который он воздействует, обозначают S (от *set*). Перевод триггера в нулевое состояние называют *сбросом* или *гашением* (*reset*), а соответствующий сигнал и вход обозначают R.

Схема простейшего триггера (рис. 6.1, *а*) получается, если включить кольцом два элемента ИЛИ—НЕ. Такой триггер имеет два входа R и S, два выхода Q и $\bar{Q}$ и называется *RS-триггером*. Его обозначение на функциональных схемах показано на рис. 6.1, *б*.

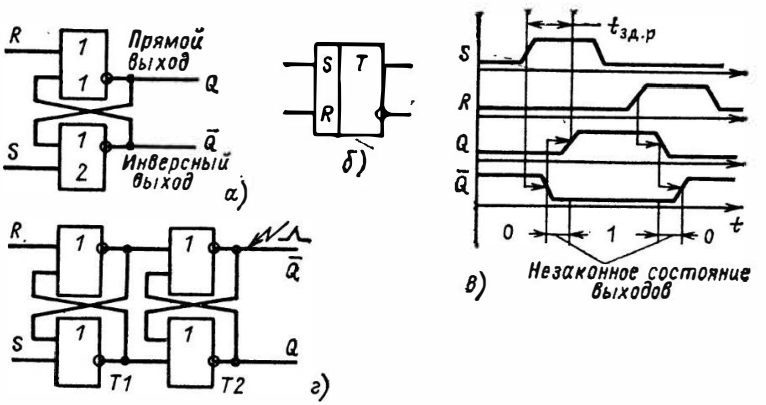


Рис. 6.1. *RS*-триггер на элементах ИЛИ-НЕ

Пока на обоих управляющих входах R и S уровни сигналов не активны, в данном случае равны 0, триггер находится в каком-либо одном из двух устойчивых состояний. Если значение сигнала на выходе Q равно 1, то, как видно из схемы, этот единичный сигнал, поступая по цепи обратной связи на вход элемента 2, вызывает появление на выходе $\bar{Q}$ сигнала с нулевым уровнем. В свою очередь нулевой уро-

вень выхода $\overline{Q}$, поступая на вход элемента *1*, поддерживает Q в состоянии 1. Иначе говоря, при входных сигналах R и S, равных 0, появившаяся по любой причине на выходе Q единица по цепи обратной связи будет сама себя поддерживать сколь угодно долго. Когда на прямом выходе Q сигнал равен 1, говорят, что триггер *находится в состоянии 1* или что он *установлен*.

В силу симметрии схемы она будет столь же устойчива в своем противоположном — нулевом состоянии, когда уровень на выходе Q равен 0, а уровень на инверсном выходе $\overline{Q}$ равен 1. В этом случае говорят, что триггер *сброшен* или *погашен*. Режим RS-триггера, когда оба управляющих сигнала R и S неактивны, называют *режимом хранения*.

На рис. 6.1, *в* показана временна́я диаграмма переходных процессов в схеме при подаче на нее управляющих сигналов. Исходное состояние триггера — нулевое, на его входы поступают по очереди сначала сигнал S, затем, после его окончания — сигнал R. Попытайтесь сначала самостоятельно построить временны́е диаграммы на выходах триггера.

Из диаграммы видно, что после окончания входного сигнала триггер способен сохранять свое новое состояние также сколь угодно долго. Говорят, что триггер *запоминает* входной сигнал. Это специфическое и очень важное свойство триггера, отличающее его от всех рассмотренных ранее схем, не имевших обратных связей: после исчезновения входного сигнала выходной сигнал в тех схемах также исчезал.

Характерно, что оба элемента триггера переключаются не одновременно, а последовательно, друг за другом. Как видно из временно́й диаграммы, существуют моменты времени, когда и на прямом Q, и на инверсном $\overline{Q}$ выходах триггера уровни одинаковы. В то же время алгоритмы работы управляемых триггерами схем и соответственно сами эти схемы строят исходя из установившихся значений сигналов на выходах триггера, когда оба они взаимно инверсны. Поэтому управляемая триггером схема, получив на вход непредусмотренную комбинацию сигналов, сформирует на своем выходе также нечто совершенно не предусмотренное алгоритмом ее работы. В дальнейшем будут рассмотрены меры, которые разработчик должен принять, чтобы возникающая при переключении триггера неверная комбинация его выходов не приводила к сбою.

По временно́й диаграмме можно оценить время задержки распространения $t_{зд.р}$ триггера как отрезок времени, по прошествии которого на обоих выходах триггера устанавливаются правильные уровни: $t_{зд.р}=2\tau$. Можно оценить и минимально допустимую длительность R- и S-сигналов, ниже которой обратная связь триггера еще не успеет замкнуться и в результате выходы триггера вернутся в исходное состояние. Это значение лежит в пределах $(2\div3)\tau$. Для более точной оценки нужно знать допуски на пороги срабатывания и длительности фронтов элементов. Для триггеров, выпускаемых в виде схем средней интеграции, значения $t_{зд.р}$ и минимальной длительности входных сигналов указывают в паспорте.

Если на RS-триггер подать одновременно оба входных сигнала, то на обоих выходах Q и $\bar{Q}$ появятся нули. Если теперь одновременно снять единицы со входов R и S, то оба элемента начнут переключаться в единичное состояние, каждый стремясь при этом оставить своего партнера в нуле. Какой элемент одержит в этом поединке победу, будет зависеть от их коэффициентов усиления, скоростей переходных процессов и ряда других неизвестных заранее факторов. Для разработчика схемы результирующее состояние триггера оказывается неопределенным, неуправляемым. Поэтому комбинация $R=S=1$ считается *запрещенной*, и в обычных условиях ее не используют. В некоторых справочниках эту комбинацию даже называют *неустойчивой*, хотя пока она держится на входах, схема вполне устойчива. Комбинацию входов $R=S=1$ допустимо применять, лишь когда обеспечено не одновременное, а строго поочередное снятие R- и S-сигналов.

От рассматривавшихся ранее схем без обратных связей RS-триггер отличается еще и тем, что его выходы одновременно являются и его входами. Действительно, если на линию связи, подключенную к выходу Q триггера, находящегося в нулевом состоянии, подействует короткая единичная помеха, она одновременно подействует и на вход второго элемента триггера, что может вызвать его переключение, а это приведет к переключению всего триггера, как от обычного входного сигнала. Свойство триггера запоминать помехи, превращая их из мимолетных в постоянно действующие, в большинстве применений крайне нежелательно. Поэтому если триггер работает на линию, в которой возможны помехи, то ее подключают через буферные элементы. Для повышения быстродействия эти элементы

часто тоже соединяют по схеме триггера, как показано на рис. 6.1, *г*. Поскольку на входах буферного триггера *T2* постоянно присутствует или R-, или S-сигнал, этот триггер уже не сможет запомнить помеху и после ее окончания сразу вернется в правильное состояние.

На рис. 6.2 показан триггер, построенный на элементах И-НЕ, т. е. двойственный по отношению к триггеру рис. 6.1.

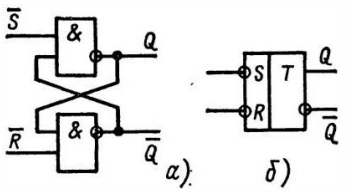


Рис. 6.2. Функциональная схема *(а)* и условное обозначение *(б)* RS-триггера на элементах И-НЕ

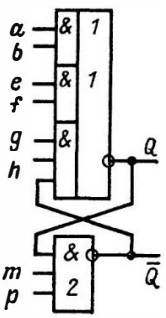


Рис. 6.3. RS-триггер, имеющий по несколько R- и S-входов

Как и следовало ожидать, в нем все «наоборот». В режиме хранения на обоих входах должны быть не нули, а единицы; сигналы управления $\overline{R}$ и $\overline{S}$ имеют активный низкий уровень (отсюда символы инверсий на входах); одновременная подача двух нулей на входы запрещена. В ТТЛ-сериях практически все триггеры строят по схеме, изображенной на рис. 6.2, *а*. Полезно самостоятельно построить временну́ю диаграмму этого триггера.

Триггеры можно строить на инвертирующих элементах различных типов. В качестве примера на рис. 6.3 показан триггер, у которого R- и S-воздействиями являются логические комбинации нескольких входов. Подобные схемы используют тогда, когда нужно устанавливать и гасить триггер сигналами от нескольких источников.

Упражнение. Найти комбинацию входов, обеспечивающую режим хранения этого триггера, и комбинации входов, выполняющие функции R- и S-воздействий.

Р е ш е н и е. Для режима хранения комбинация входов должна быть такова, чтобы выход каждого элемента однозначно определялся только сигналом, поступающим по петле обратной связи от второго элемента триггера. Это выполняется при комбинациях входов, удовлетворяющих еди-

ничному значению функции хранения G:

$$G = \overline{ab} \cdot \overline{ef} \cdot gh \cdot mp.$$

Для режима переключения комбинация входов, воздействующая на один из элементов, должна, наоборот, обеспечивать на выходе этого элемента требуемое состояние независимо от значения сигнала, поступающего по цепи обратной связи от другого элемента. Этому условию удовлетворяют функции

$$S = \overline{gh} = \bar{g} \vee \bar{h}; \quad R = ab \vee ef \vee \bar{m} \vee \bar{p}.$$

Предполагается, что на выходах, не являющихся аргументами функций S и R, поддерживаются уровни, соответствующие режиму хранения.

Если соединить в кольцо не два, а *любое четное* число *инвертирующих* элементов, то полученная схема также будет иметь *два устойчивых состояния*. Пример такой схемы показан на рис. 6.4, *а*.

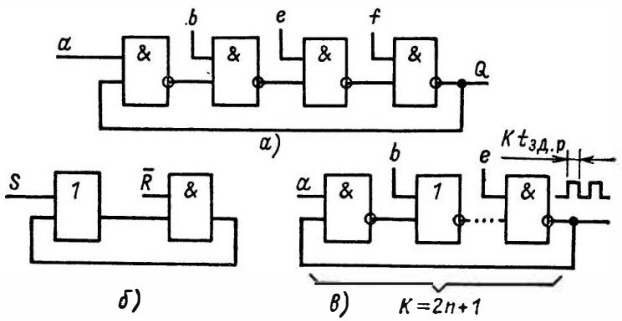


Рис. 6.4. Соединение в кольцо различного числа инвертирующих элементов

Упражнение. Определить функции от входных сигналов a, b, e, f, обеспечивающие режим хранения и переключения получившегося триггера.

Преднамеренно подобные схемы не строят ввиду их неоправданно большой задержки переключения, но они иногда возникают самопроизвольно в результате ошибок или отказов, и схемотехник должен уметь по характеру поведения схемы распознавать подобные новообразования. Поскольку *нуль* — число четное, кольцо, имеющее нуль инверторов, также обладает триггерными свойствами (см. рис. 6.4, *б*).

Если в кольцо соединить *любое нечетное* число *инверторов*, как, например, показано на рис. 6.4, *в*, то сигнал обратной связи будет уже

не подтверждать состояние элемента, на который он поступает, так это происходит в триггере, а переключать элемент в противоположное состояние. Фронт переключения элемента, задержанный в кольце обратной связи, вновь проинвертирует состояние рассматриваемого элемента и т. д. Кольцо с нечетным числом инверторов будет работать как генератор *меандра* — последовательности импульсов прямоугольной формы, длительность которых равна длительности пауз между ними. В данном случае длительность импульсов (и пауз) равна суммарной задержке элементов кольца. В качестве задающих генераторов цифровых устройств подобные кольца применять нерационально ввиду низкой стабильности генерируемой частоты: частота определяется суммой задержек, значение которых ограничено лишь по максимуму. Такие кольца используют в качестве вспомогательных наладочных генераторов, а также для экспериментального определения среднего времени задержки логических элементов, особенно быстрых. Кроме того, такие цепочки, как и цепочки триггерного типа, могут возникать в схемах и помимо желания разработчика.

Основное назначение триггеров в цифровых схемах — хранить выработанные логическими схемами результаты. Для отсечения еще не установившихся, искаженных переходными процессами результатов между выходом логической схемы и входом триггера можно включить конъюнктор типа элемента *С* на рис. 5.6. Это решение оказалось очень эффективным, быстро стало типовым и побудило изготовителей триггеров ввести конъюнктор, управляемый синхросигналом, в состав триггера. Так появились *синхронные триггеры,* которые переключаются в состояние, предписываемое управляющими входами, лишь по сигналу синхронизации, поступающему на синхровход *С* триггера. Синхросигнал называют также *синхроимпульсом, С-сигналом, С-импульсом,* а синхровход — *С-входом.* При неактивном уровне *С*-сигнала синхронный триггер находится в режиме хранения и не реагирует ни на какие управляющие сигналы. Развитие идеи синхронного триггера привело к появлению разнообразных и довольно сложных триггерных устройств, в которых кроме собственно *RS*-триггера присутствует логическая схема обработки входных сигналов, а часто еще один-два вспомогательных триггера. Такие устройства по традиции продолжают именовать триггерами, добавляя перед словом триггер различные буквы, обозначающие принцип функционирования всего устройства. Всевозможных видов, а тем более конкретных схем сложных синхронных триггеров очень много. В данной книге приведены лишь те типы и схемы, которые в настоящее время

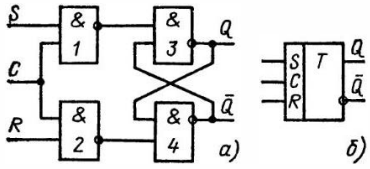


Рис. 6.5. Синхронный *RS*-триггер

наиболее широко применяют в цифровых устройствах и к тому же четко различаются по выполняемым функциям. Для расширения эрудиции в этом вопросе можно рекомендовать [13, 33, 35—37, 59].

Схема простейшего синхронного *RS*-триггера показана на рис. 6.5, *а*, где конъюнкторы *1* и *2* аналогичны конъюнктору *C* на схеме рис. 5.6: при $C=0$ триггер *3-4* отключен от управляющих *S*- и *R*-входов и находится в режиме хранения ранее полученной информации. При $C=1$ схема функционирует как обычный *RS*-триггер. Условное изображение синхронного *RS*-триггера показано на рис. 6.5, *б*. Синхровход *C* может в принципе иметь и активный низкий уровень; в этом случае он, как обычно, помечается кружочком или указателем уровня. Характерной особенностью схемы по рис. 6.5, *а* является то, что в течение всего отрезка времени, когда синхросигнал равен 1, как сами потенциалы на управляющих *S*- и *R*-входах, так и любые их изменения тут же передаются на выход. О такой схеме можно сказать, что она *прозрачна по S- и R-входам* при $C=1$. Не все схемы синхронных триггеров обладают этим свойством.

6.2. *D*-триггер типа «защелка»

Этот тип синхронного триггера исключительно широко используется в цифровых устройствах. Другие его названия: *прозрачная защелка* (*transparent latch*), *прозрачный фиксатор, синхронный фиксатор, D*-триггер, управляемый уровнем синхросигнала. Часто используемое название просто *D*-триггер не отражает свойства прозрачности схемы по *D*-входу и не выделяет ее среди других типов *D*-триггеров. В литературе укоренился термин *защелка (latch)*, иногда используется термин *фиксатор*. Эти термины хотя и не подчеркивают специально свойства прозрачности, но по крайней мере именуют схемы только этого типа. В дальнейшем будет в основном использоваться термин *защелка*.

Триггер защелка относится к классу *D*-триггеров,

D-триггером называют синхронный триггер, имеющий два входа: вход данных *D* и вход синхронизации *C*. *D*-триггер переключается только по сигналу на *C*-входе и притом в состояние, предписываемое *D*-входом. В некотором смысле он задерживает прохождение поступившего по *D*-входу уровня до появления *C*-сигнала, откуда и произошло название *D*-триггера (*delay* — задержка). Другое назначение *D*-триггера — сохранить данные (*data*), поступившие однажды по *D*-входу. *C*-сигналы в этом случае играют роль команды ЗАПИСАТЬ В ТРИГГЕР. *RS*-триггеры в своём чистом виде для хранения данных неудобны и в этой роли не используются, поскольку для записи они требуют двух последовательных сигналов: гашения по *R*-входу и затем собственно записи по *S*-входу. Условное обозначение *D*-триггера показано на рис. 6.6, *а*.

На рис. 6.6, *б* показан универсальный способ построения *D*-триггера из синхронного *RS*-триггера: с помощью

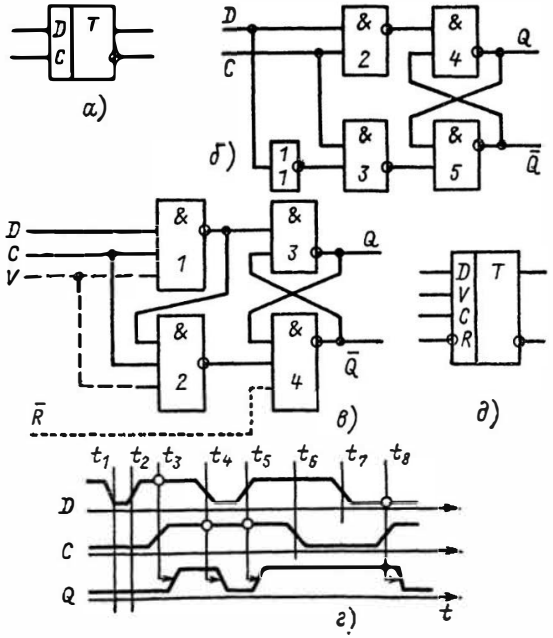


Рис. 6.6. *D*-триггер типа «прозрачная защелка»:

а — условное обозначение; *б* — универсальный способ построения *D*-триггера из синхронного *RS*-триггера; *в* — вариант схемы прозрачной защелки; *г* — пример временной диаграммы работы *D*-триггера-защелки; *д* — условное обозначение *DV*-триггера с дополнительным входом сброса

инвертора *1* однофазный вход данных D превращается в парафазный и подается на S- и R-входы. На рис. 6.6, *в* показана несколько более экономичная схема прозрачной защелки. На входы V и $\bar{R}$ обращать внимание пока не нужно.

Особенности поведения прозрачной защелки иллюстрирует диаграмма на рис. 6.6, *г*. Изменения D-входа при $C=0$ (моменты t_1, t_2, t_7) никак не влияют на состояние выхода Q: триггер заперт по C-входу и находится в *режиме хранения*. Фронт C-сигнала (момент t_3) вызывает переключение триггера в то состояние, которое было к этому моменту на входе D. При $C=1$ защелка прозрачна: любое изменение D-входа (моменты t_4 и t_5) вызывает изменение выхода Q. По срезу синхросигнала (момент t_6) D-триггер-защелка *фиксирует* на выходе то состояние, которое было на D-входе непосредственно перед этим моментом. Следующее изменение Q будет возможно только по фронту следующего синхроимпульса (момент t_8). Если на C-вход подать постоянный единичный уровень, то свойство запоминания защелки проявляться никак не будет и она будет выполнять функции обычного буферного усилителя мощности в тракте передачи данных.

Способ табличного описания функционирования триггера-защелки представлен в табл. 6.1.

Таблица 6.1

Режим	Входы		Выходы	
	C	D	Q	$\bar{Q}$
Управление	*H* *H*	*L* *H*	*L* *H*	*H* *L*
Хранение	*L*	*X*	Q	$\bar{Q}$

Иногда в триггер-защелку вводят дополнительный вход $\bar{R}$ сброса в нуль, показанный на рис. 6.6, *в* пунктиром. Такой простейший вариант входа гашения имеет активный низкий уровень, и пользоваться им можно лишь при $C=0$. Если его подавать при $C=1$, то в случае, когда и $D=1$, RS-триггер, образованный элементами *3* и *4*, окажется под воздействием сразу и S-, и R-сигнала, после одновременного снятия которых состояние его станет неопределенным.

Существуют D-триггеры, в которых параллельно C-сиг-

налу на входные вентили заведен еще один разрешающий сигнал — *V*-сигнал (от *valve* — клапан), как показано штриховой линией на рис. 6.6, *в*. Такие триггеры называют *DV*-триггерами. Разрешением на прием *D*-уровня является конъюнкция сигналов на *C*- и *V*-входах. Условное обозначение *DV*-триггера показано на рис. 6.6, *д*. На рис. 6.7, *а* показана схема защелки, для которой активным является низкий уровень *C*-сигнала, т. е. триггер прозрачен при низком уровне на входе *C*.

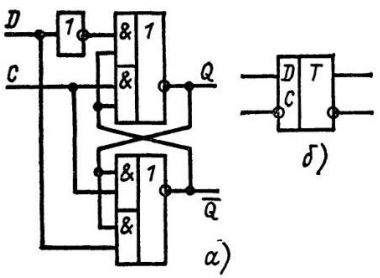
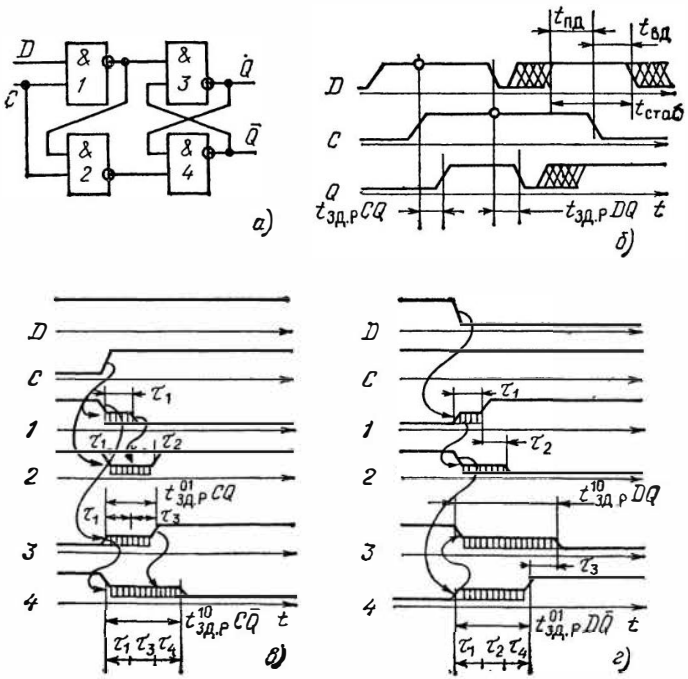


Рис. 6.7. *D*-триггер-защелка с активным низким уровнем *C*-сигнала

Рис. 6.8. Временные характеристики *D*-триггера-защелки:

а — схема триггера; *б* — основные временные характеристики; *в*, *г* — определение задержки; *в* — при изменении *C*-сигнала; *г* — при изменении *D*-сигнала



Временные диаграммы (рис. 6.8, *б*) триггера-защелки (рис. 6.8, *а*) построены с учетом состояний неопределенности. Переходной процесс характеризуется значениями задержек распространения по двум трактам: $t_{зд.р}CQ$ — от входа C до выхода при $D=\text{const}$ и $t_{зд.р}DQ$ — от входа D до выхода при $C=1$. Иногда каждый тракт расщепляют на два подтракта: до выхода Q и до выхода $\bar{Q}$. Каждый тракт (или подтракт) в свою очередь характеризуется задержками распространения при переключении выходов в 1 — $t^{01}_{зд.р}$ и в нуль $t^{10}_{зд.р}$.

Чтобы процесс фиксации состояния D-входа прошел без сбоев, т. е. был бы однозначно предсказуемым, переходной процесс в схеме защелки, вызванный срезом C-сигнала, не должен накладываться на переходной процесс, вызванный переключением D-входа. Это значит, что всякие изменения состояния D-входа должны прекратиться за некоторое время до среза C-сигнала, называемое *временем подготовки (setup time)* $t_{пд}$, и могут снова начинаться после среза C-сигнала не ранее чем через *время выдержки (удержания) (hold time)* $t_{вд}$. Эти временные отрезки также показаны на рис. 6.8, *б*. В зависимости от конкретных значений порогов переключения и длительностей фронтов их можно оценить как $t_{пд}=(1\div 2)\tau$, $t_{вд}=(0\div 1)\tau$.

Необходимость введения и нормирования интервалов подготовки и выдержки характерна не только для защелки, но и для всех функциональных узлов, имеющих вход синхронизации. Кроме того, для защелки, как и для любого синхронного узла, существует *минимально допустимая длительность синхроимпульса,* обеспечивающая отсутствие сбоев из-за наложения переходных процессов от фронта и среза этого импульса.

В схеме защелки, показанной на рис. 6.6, *в*, могут возникать опасные гонки по входу. Если длительность среза C-сигнала существенно превышает времена задержки элементов, из которых собран D-триггер, а порог переключения элемента *1* заметно выше порога элемента *2*, и если на D-входе перед фиксирующим срезом была 1, то выход элемента *1* переключится в 1 где-то вскоре после начала пологого среза C-сигнала. Элемент *2*, имея низкий порог переключения, в это время воспринимает C-сигнал еще как 1. Поэтому, получив на свой вход еще одну 1, он переключится в 0, переключив тем самым в 0 и выходной триггер на элементах *3-4*. В результате вопреки 1 на D-входе D-триггер

после среза C-сигнала окажется в состоянии 0. Поэтому при использовании защелки по рис. 6.6, *в* длительность среза C-сигнала не должна быть заметно больше задержки элементов, из которых собран триггер. Существенно, что в схеме, показанной на рис. 6.6, *б*, опасных гонок по C-входу не возникает, что в большинстве применений оправдывает немного большие аппаратурные затраты этой схемы. Если все сказанное не совсем ясно, постройте самостоятельно соответствующие временны́е диаграммы.

Таким образом, временны́ми параметрами D-триггеров-защелок являются: времена задержки распространения по трактам вход C — выходы и вход D — выходы; время подготовки по D-входу; время выдержки по D-входу; минимальная длительность C-импульса; для схем, в которых возможны гонки по входу, — еще максимальная длительность фиксирующего среза C-сигнала. Для защелки, построенной из отдельных элементов, например на поверхности кристалла БИС, все перечисленные временны́е параметры могут быть определены по временны́м диаграммам переходных процессов, построенным по правилам, показанным на рис. 5.3, *в* или *г*. В качестве примера на рис. 6.8 приведены временны́е диаграммы переходных процессов, вызванных переключением C-сигнала при $D=1$ (рис. 6.8, *в*) и D-сигнала при $C=1$ (рис. 6.8, *г*). С их помощью определяют значения задержек по трактам CQ и DQ. Для триггеров-защелок, выпускаемых в виде микросхем, временны́е характеристики приводятся в справочниках. Примерами выпускаемых промышленностью D-триггеров-защелок могут служить К155ТМ5, К155ТМ7, К561ТМ3, которые содержат по четыре триггера с объединенными C-входами.

6.3. Двухступенчатые триггеры

На рис. 6.9, *а* показана схема, состоящая из двух последовательно включенных синхронных RS-триггеров, первый из которых называется *ведущим* или *М-триггером* (от *master* — хозяин), а второй — *ведомым* или *S-триггером* (от *slave* — раб). Благодаря общему синхросигналу C вся схема функционирует как единое целое и называется *двухступенчатым* или *MS-триггером* (*master-slave flip-flop*). Из временно́й диаграммы (рис. 6.9, *б*) видно, что информация, задаваемая уровнями на входах S и R, по фронту C-сигнала принимается в M-триггер, но в течение всего времени, пока C-сигнал равен 1, не проходит в S-триггер,

поскольку его входные конъюнкторы *5* и *6* в это время перекрыты инверсией *C*-сигнала — сигналом $\bar{C}$. Они откроются лишь при $\bar{C}=1$, т. е. на срезе *C*-сигнала, и только тогда *S*-триггер примет состояние *M*-триггера. Сказанное иллюстрирует очень важное отличие *MS*-триггера от триггера-защелки: *MS*-триггер, собранный по схеме рис. 6.9, *а*, *непрозрачен* по управляющим *R*- и *S*-входам ни при $C=0$,

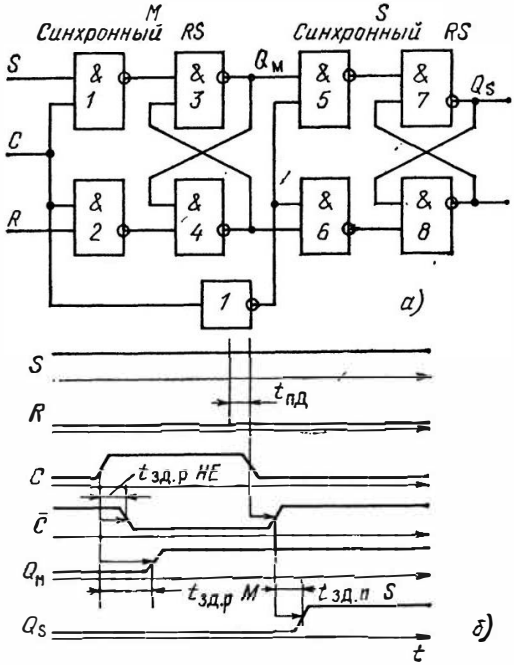


Рис. 6.9. Двухступенчатый *RS*-триггер

ни при $C=1$. Каждая ступень его сама по себе прозрачна, но включены ступени последовательно, и какая-нибудь одна из них всегда оказывается запертой — или синхросигналом, или его отсутствием. Таким образом, в этом *MS*-триггере при $C=1$ (и тем более при $C=0$) никакое изменение на управляющем входе не может само по себе, без переключения *C*-сигнала, проникнуть на выход. Триггер может изменить состояние выхода только по срезу *C*-сигнала. В зарубежной литературе непрозрачные триггеры называ-

ют *flip-flop* в отличие от прозрачных *D*-триггеров, за которыми укрепился термин *latch*.

Срезу *C*-сигнала предшествует показанный на рис. 6.9, *б* интервал подготовки $t_{пд}$, в течение которого сигналы на *S*- и *R*-входах не должны изменяться. Иначе, если срез *C*-сигнала наложится на процесс переключения *M*-триггера, особенно в момент, когда на обоих его выходах присутствуют единицы, правильную работу всего *MS*-триггера гарантировать нельзя. Интервал выдержки $t_{вд}$ для двухступенчатого триггера обычно считают равным 0, поскольку входные конъюнкторы *1* и *2* закрываются срезом самого синхроимпульса и не пропускают никаких более поздних изменений. Это значит, что управляющие *S*- и *R*-сигналы могут обновляться по срезу того же синхроимпульса, который управляет триггером, и триггер при этом всегда будет воспринимать лишь предыдущее, еще не обновленное состояние *S*- и *R*-сигналов. Как будет видно дальше, на этом свойстве держится вся идеология однофазной синхронизации.

Свойство непрозрачности *MS*-триггера использовано для построения интересного и широко применяемого *JK-триггера,* схема которого показана на рис. 6.10, *а*. *JK*-триггер — это непрозрачный триггер, выходы которого петлями инвертирующих обратных связей (накрест) заведены на входные конъюнкторы *1* и *2*. Внешние входы самого триггера при этом принято называть уже не *S* и *R*, а *J* и *K*.

При $J=K=0$ *C*-сигнал не может открыть входные элементы *1* и *2*, и триггер находится в режиме хранения. При $J=1$, $K=0$ синхросигналом может быть открыт лишь элемент *1* и только при условии, что перед поступлением *C*-сигнала на выходе триггера был 0 ($Q=0$, $\bar{Q}=1$). Тогда по срезу синхросигнала триггер переключится в 1. Если же триггер до синхросигнала был в 1, то он так и останется в 1. Таким образом, *J*-вход выполняет функции синхронизированного *S*-входа. В силу симметрии схемы легко показать, что *K*-вход выполняет функции синхронизированного *R*-входа, переводя триггер в 0. Таким образом, при разноименных уровнях на *J*- и *K*-входах *JK*-триггер ведет себя как синхронный непрозрачный *RS*-триггер.

Существенно отличным от *RS*-триггера является поведение *JK*-триггера при $J=K=1$. Для *RS*-триггера такое состояние входов запрещено. Диаграмма работы *JK*-триггера в этом режиме показана на рис. 6.10, *б*. При любом состоянии триггера сигналы обратной связи открывают для

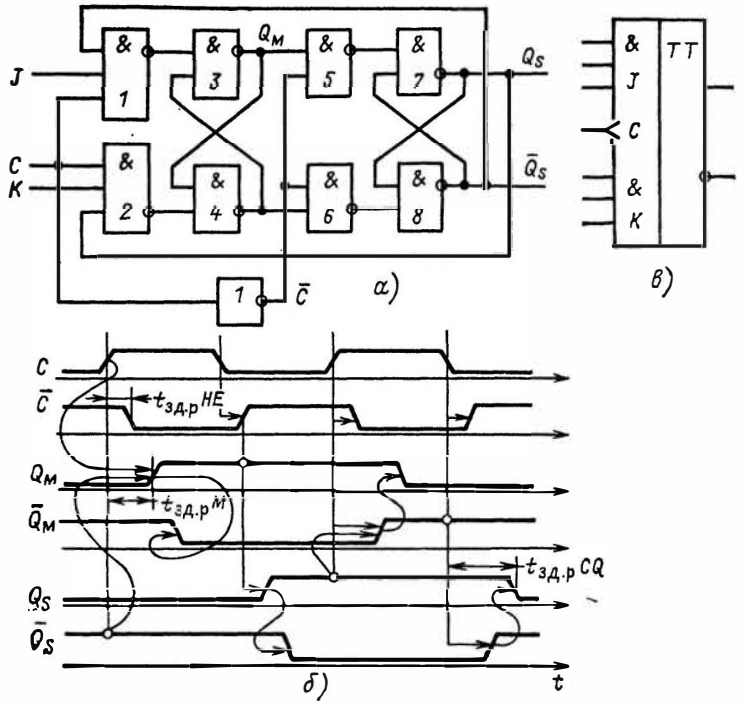


Рис. 6.10. Двухступенчатый JK-триггер с инвертором в цепи синхронизации:

а — схема; *б* — временна́я диаграмма при $J=K=1$; *в* — условное обозначение двухступенчатого JK-триггера с тройными конъюнктивными J- и K-входами, переключаемого срезом C-сигнала

C-сигнала именно тот входной конъюнктор, пройдя через который, C-сигнал переведет триггер в противоположное состояние. Таким образом, при $J=K=1$ по срезу каждого C-сигнала JK-триггер меняет состояние своих выходов на противоположное. Это так называемый *счетный* режим, или T-режим работы триггера (от *toggle* — кувыркаться).

Кратко функционирование JK-триггера описывается табл. 6.2. Новым символом в таблице является символ среза синхроимпульса, который также иногда изображается направленной вниз стрелкой. Таблица отражает тот факт, что для JK-триггера переключающей сущностью синхроимпульса является не уровень его, а перепад уровня. Строчная буква q обозначает состояние триггера перед поступлением среза C-сигнала.

Таблица 6.2

Режим	Входы			Выходы	
	C	J	K	Q	$\bar{Q}$
Хранение	×	0	0	q	$\bar{q}$
Сброс	⊓	0	1	0	1
Установка	⊓	1	0	1	0
Счетный	⊓	1	1	$\bar{q}$	q

Важно отметить, что для возможности построения JK-триггера путем охвата синхронного триггера петлей отрицательной обратной связи существенным является свойство непрозрачности триггера. Если петлей обратной связи охватить прозрачный одноступенчатый синхронный RS-триггер, то при $C=1$ выход триггера, поступив с некоторой задержкой на его вход, вызовет его переключение в обратное состояние и т. д., т. е. схема будет вести себя как генератор меандра.

Схема, близкая к показанной на рис. 6.10, *а*, лежит в основе триггера К155ТВ1. Эта микросхема имеет тройные конъюнктивные входы J и K, т. е. сам двухступенчатый триггер получает J- или K-сигнал лишь при совпадении единиц на всех трех J- или K-входах микросхемы. Условное обозначение двухступенчатого JK-триггера, имеющего тройные конъюнктивные J- и K-входы, показано на рис. 6.10, *в*. Две буквы T указывают на наличие двух ступеней. Вход C, реагирующий не на уровень потенциала C, а на его отрицательный перепад, выделен специальным значком. Вход, реагирующий именно на перепад, иногда называют *динамическим*.

В триггерах, показанных на рис. 6.9 и 6.10, есть параллельные пути распространения сигнала: C-сигнал проходит и через M-триггер, и через инвертор, а затем оба сигнала встречаются на входах элементов *5* или *6*. Следовательно, в схеме есть опасность возникновения гонок, и ее нужно проанализировать на возможность сбоев по этой причине. В данном случае опасения подтверждаются. Если задержка инвертора превысит задержку M-ступени, то при поступлении C-сигнала новое состояние M-ступени успеет проскочить в S-ступень, прежде чем инвертор закроет элементы *5* и *6*. Выход триггера при этом изменится по фронту C-сигнала, а не по его срезу, что вызовет сбой в узле, ко-

торым триггер управляет. Схемы по рис. 6.9 и 6.10 используют в интегральной технологии, но там опасность гонок ликвидируется за счет или нормирования минимально возможной задержки *M*-ступени, или специально введенной гарантированной разницы в порогах срабатывания инвертора и входных конъюнкторов *M*-ступени. Делают так, чтобы инвертор переключался при более низком уровне *C*-сигнала, чем входные конъюнкторы *1* и *2*, и тогда нужная последовательность срабатывания инвертора и *M*-ступени обеспечивается за счет конечной длительности фронтов *C*-сигнала. Инвертор реагирует на фронт сигнала раньше, а на его срез — позже, чем это делает *M*-ступень.

Для того чтобы принципиально снять проблему гонок в двухступенчатом триггере, нужно каким-то способом вообще ликвидировать параллельный логический тракт, представленный инвертором. Одним из возможных решений является выполнение обеих ступеней *MS*-триггера взаимно двойственными, использовав, например, в качестве элементов *1—4* схем по рис. 6.9 и 6.10 элементы И-НЕ, а в качестве *5—8* — элементы ИЛИ-НЕ. *C*-сигнал заводится непосредственно и на пару элементов *1, 2,* и на пару *5, 6*, но в силу взаимной их двойственности обе пары реагируют на взаимно противоположные уровни *C*-сигнала: пара *1, 2* — на высокий, а пара *5, 6* — на низкий. Задержка же поступления перепада *C*-сигнала на вторую ступень оказывается практически нулевой, т. е. гарантированно меньшей задержки срабатывания первой ступени *MS*-триггера. Если приведенное словесное объяснение порождает сомнения или вопросы, начертите полную схему этого разношерстного триггера и постройте его временную диаграмму. Описанная схема хорошо согласуется с КМДП-технологией, где цены элементов И-НЕ и ИЛИ-НЕ одинаковы, но существенно хуже — с ТТЛ-технологией. Схему можно использовать в тех случаях, когда не нормируются ни минимальные времена задержки элементов, ни различия в порогах их срабатывания.

Другое решение, исключающее в двухступенчатом триггере параллельный логический тракт, показано на рис. 6.11. Это тоже противогоночная схема, в которой возможность ложного проникновения сигнала *M*-ступени в *S*-ступень на фронте *C*-импульса блокируется нулевым уровнем выхода того из входных конъюнкторов *1* или *2*, который срабатывает в данный момент. Такую схему триггера иногда называют схемой с *запрещающими связями*.

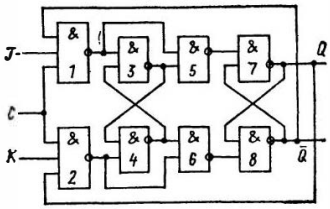


Рис. 6.11. Двухступенчатый *JK*-триггер с запрещающими связями

Очевидно, что для всех рассмотренных двухступенчатых триггеров можно построить двойственные варианты, у которых прием в *M*-ступень будет осуществляться по отрицательному перепаду *C*-сигнала, а выход изменяться по положительному. Тогда, чтобы однозначно описывать поведение любого варианта триггера при подаче на *C*-вход как синхроимпульсов прямой полярности, так и инвертированных, нужно в описании иметь возможность отличать перепад, связанный с появлением *C*-импульса, от перепада, связанного с его исчезновением. Поэтому наряду с терминами *фронт* и *срез*, обозначающими полярность перепада, в дальнейшем будут использоваться термины *передний фронт* и *задний фронт*, обозначающие соответственно начало и конец импульса независимо от того, какова при этом полярность перепада.

Фронты в цепях синхронизации могут затягиваться, поскольку эти цепи часто имеют значительную длину. В *JK*-триггере, построенном по схеме рис. 6.11, *C*-сигнал поступает сразу на два элемента — *1* и *2*, и на эти же элементы приходят сигналы обратной связи с выходов триггера второй ступени. Такое решение должно тут же насторожить схемотехника, побудив его исследовать схему на возможность гонок по входу. И действительно, такая опасность существует — при работе в *T*-режиме на пологом срезе *C*-сигнала. Пусть весь *JK*-триггер до поступления некоторого *C*-сигнала находился в 0. Тогда по фронту этого *C*-сигнала триггер первой ступени, как это и должно быть, переключится в 1. Пусть элемент *1* имеет высокий порог переключения, а элемент *2* — низкий. Тогда где-то в начале пологого среза *C*-сигнала элемент *1* уже начнет воспринимать *C*-сигнал как 0 и, переключившись, откроет элемент *5* второй ступени, пропустив единичное состояние триггера первой ступени в триггер второй ступени. Эта единица с выхода элемента *7* по петле обратной связи попадет на вход элемента *2* первой ступени, который в силу своего

более низкого порога переключения воспринимает C-сигнал еще как 1. Конъюнкция единиц C-сигнала и обратной связи порождает на выходе элемента *2* нулевой уровень, что вызывает обратный, уже ложный переброс триггера первой ступени в 0. После того как C-сигнал уже с точки зрения элемента *2* станет равным 0, это ложное состояние первой ступени будет передано во вторую ступень. В результате выход триггера за время среза дважды изменит свое состояние, посчитав тем самым один C-сигнал за два. В силу симметрии схемы то же самое произойдет и при противоположной разнице порогов срабатывания элементов *1* и *2*, только уже при единичном исходном состоянии JK-триггера. Если словесное описание не вызывает в сознании достаточно четкой картины, постройте самостоятельную временну́ю диаграмму.

В схеме, показанной на рис. 6.10, *а*, гонки по входу на пологих фронтах C-сигнала могут возникать, если порог переключения инвертора окажется выше порогов входных конъюнкторов *1* и *2*. Если же порог инвертора гарантированно ниже порогов этих конъюнкторов, то схема не боится пологих фронтов C-сигнала.

Рассмотренные схемы JK-триггеров, у которых сигналы J- и K-входов поступают непосредственно на первую ступень, выполненную по схеме синхронного RS-триггера, обладают еще одним очень коварным свойством. Пусть триггер, показанный на рис. 6.10, *а*, находится в состоянии 0, и при этом $J=0$, а состояние K-входа безразлично. В этой ситуации очередной C-сигнал не должен изменить состояния триггера. Если, однако, в то время, когда очередной C-сигнал равен 1 на J-вход триггера подействует короткая единичная помеха, вызванная неокончившимися переходными процессами в комбинационной схеме, вырабатывающей J-уровень, то она пройдет через открытый конъюнктор *1*, в результате чего триггер M-ступени перебросится в 1, т. е. запомнит эту помеху. Затем по срезу C-сигнала он передаст свое новое, ложное состояние во вторую ступень, т. е. на выход. Это так называемое свойство *захвата 1* JK-триггером. В силу симметрии схемы в ней точно так же будет проявляться и свойство захвата нуля.

Более откровенно вредит противогоночная схема, показанная на рис. 6.11. Пусть такой триггер находится в режиме хранения ($J=K=0$). Тогда, если при $C=1$ на тот из входов J или K, конъюнктор которого открыт по цепи обратной связи (соответствующим состоянием триггера

второй ступени), поступит короткая единичная помеха, фронт этой помехи, как и в схеме, изображенной на рис. 6.10, *а*, вызовет переключение триггера первой ступени. Однако по срезу помехи, в отличие от поведения схемы на рис. 6.10, *а*, в схеме по рис. 6.11 новое, ложное состояние триггера первой ступени будет сразу передано во вторую ступень, не ожидая окончания *С*-сигнала. Если же при $C=1$ на *J*- или *K*-вход, находящийся в 1 и воздействующий на тот из конъюнкторов — *1* или *2*, который открыт по цепи обратной связи, подействует нулевая помеха, то она пройдет во вторую ступень и переключит ее непосредственно по своему переднему, отрицательному фронту. Получается, что схема по рис. 6.11 при $C=1$ непрозрачна для статических уровней на *J*- и *K*-входах, т. е. для уровней, установившихся еще при $C=0$ и сохраняющихся неизменными при $C=1$, и в то же время прозрачна для некоторых изменений этих же уровней, происходящих при $C=1$. Читателю полезно, строя соответствующие временные диаграммы, найти и другие комбинации уровней на управляющих входах и состояний триггера, при которых наблюдается это явление. Оказывается, что понятия *прозрачность* и *непрозрачность* еще не описывают всех возможных взаимоотношений между изменениями сигналов на *JK*-входах и на выходе. Можно предложить термин *проскок фронта*, обозначающий факт передачи на выход не всех, но некоторых переключений входных уровней. Триггер, показанный на рис. 6.11, хоть и непрозрачен, но допускает проскок фронта. Триггер на рис. 6.10, *а* и непрозрачен, и не допускает проскока, но, как уже говорилось, обладает свойством захвата.

Вывод из сказанного весьма категоричен: используя триггеры по рис. 6.10, *а*, 6.11 и вообще любые триггеры, обладающие свойствами захвата или проскока, разработчик должен обеспечить окончание всех переходных процессов в логических схемах, формирующих *J*- и *K*-уровни, еще до начала *С*-сигнала. *В течение всего времени действия С-сигнала уровни на J- и K-входах не должны изменяться.* Для характеристики триггеров, обладающих любым из этих неприятных свойств, можно предложить термин *проницаемый для помех* при $C=1$. Не широко, к сожалению, известное явление проницаемости объясняет кажущееся иногда странным поведение в схемах триггеров К155ТВ1, 133ТВ1 и им подобных.

Двухступенчатый *JK*-триггер можно построить на осно-

ве не только *RS*-триггеров, но и *D*-триггеров-защелок, как, например, показано на рис. 6.12. Важным свойством такого триггера является отсутствие явлений как захвата, так и проскока. Этот триггер *непроницаем для помех,* поэтому состояния его *J*- и *K*-входов можно безболезненно изменять как при $C=0$, так и при $C=1$. В этом легко убедиться, построив соответствующие временны́е диаграммы. Запрет-

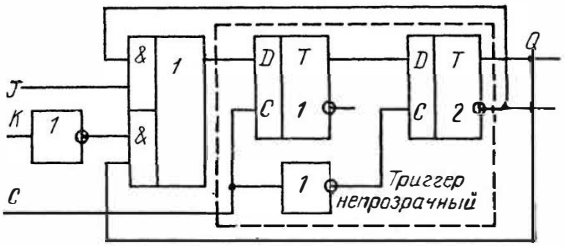


Рис. 6.12. Схема двухступенчатого *JK*-триггера без захвата 1 и 0 (триггера, переключаемого фронтом)

ными являются лишь короткие интервалы подготовки $t_{пд}$ перед самым срезом синхроимпульса и выдержки $t_{вд}$ сразу после среза. В зарубежной терминологии непроницаемые триггеры называют *edge-triggered flip-flops — триггеры, переключаемые перепадом.*

При применении этих триггеров вместо триггеров с захватом или проскоком период синхронизации используется гораздо лучше, поскольку переходные процессы в комбинационных схемах, управляющих *J*- и *K*-входами, не обязательно должны оканчиваться к моменту появления *C*-сигнала. Они могут продолжаться и когда *C*-сигнал уже стал равным 1. Кроме того, эти триггеры рационально применять при приеме информации с засоренной случайными помехами линии, поскольку помеха оказывается опасной лишь в течение короткого интервала подготовки плюс выдержки. Все остальное время этот триггер ни на какие помехи реагировать не будет. Для сравнения полезно вспомнить, что защелка прозрачна (в том числе и для помех) все время, пока $C=1$, а *JK*-триггер со свойствами захвата даже услужливо запомнит промелькнувшую при $C=1$ короткую помеху. Примером триггера, переключаемого положительным перепадом *C*-сигнала, является собранный на двух последовательно включенных *D*-защелках *JK*-триггер К561ТВ1.

Временны́ми параметрами JK-триггера, да и вообще любого непрозрачного триггера являются: задержка распространения от синхровхода до выхода (или до каждого из выходов); времена подготовки и выдержки по управляющим, в данном случае по J- и K-входам; минимально допустимая длительность C-сигнала; минимально допустимый период следования C-сигналов; для схем, в которых возможны гонки по C-входу, еще и максимально допустимая длительность фронтов C-сигнала.

На основе двухступенчатого JK-триггера по рис. 6.10, *а* можно построить D-триггер, как это показано на рис. 6.13, *а*. В отличие от защелки такой D-триггер будет *непро-*

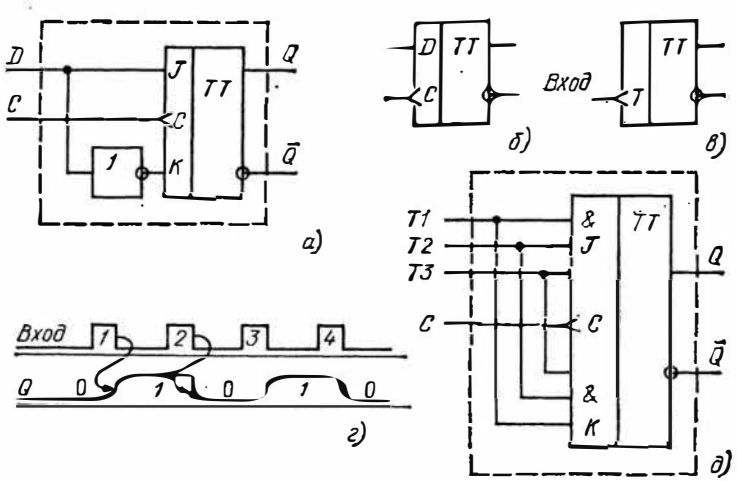


Рис. 6.13. Использование JK-триггера в роли D- и T-триггеров:
а — D-триггер на основе JK-триггера; *б* — условное обозначение D-триггера, показанного на рис. 6.13, *а*; *в* — условное обозначение T-триггера; *г* — временна́я диаграмма работы T-триггера; *д* — синхронный T-триггер

зрачен по D-входу. Явления захвата и проскока у него также отсутствуют, т. е. это триггер, переключаемый перепадом. Обозначение такого триггера показано на рис. 6.13, *б*. Однако если по этому же принципу построить D-триггер, используя схему по рис. 6.11, то он уже будет при $C=1$ допускать проскок фронта по D-входу.

Счетным, или *T-триггером* (от *toggle*) называют JK-триггер, у которого на J- и K-входы поданы не управляемые, а постоянные единичные уровни, т. е. который исполь-

зуется только в счетном режиме. Вход, вызывающий переключение триггера, называют *счетным входом*. Временна́я диаграмма работы T-триггера, условное обозначение которого представлено на рис. 6.13, *в*, показана на рис. 6.13, *г*. Изменяя по срезу каждого входного сигнала состояние своего выхода (или по фронту для двойственной схемы), T-триггер *пересчитывает входные сигналы на два* или, как говорят, *делит частоту входных сигналов пополам*. Полезной оказывается и такая интерпретация его работы: T-триггер *суммирует по модулю 2* поступивший T-сигнал со своим собственным состоянием, в котором находился к моменту поступления активного фронта T-сигнала. Результатом этого сложения является новое состояние триггера.

На рис. 6.13, *д* показан *синхронный T-триггер,* меняющий свое состояние по срезу C-сигнала при условии, что конъюнкция потенциалов на T-входах есть 1. Счетные входы T_i являются в данном случае потенциальными, управляющими, которые разрешают пересчет синхросигналов.

6.4. Шестиэлементный триггер

На рис. 6.14, *а* показана схема еще одного синхронного триггера, который широко распространен в микросхемотехнике. Другие его названия: *триггер Вебба, триггер с самоблокировкой, схема трех триггеров*. Существуют модификации этого триггера с большей и даже полной симметрией межэлементных связей (см. [36]). Рассматриваемая модификация наиболее экономична по суммарному числу входов элементов. На базе элементов ИЛИ-НЕ можно построить двойственную схему.

На рис. 6.14, *б* приведена временна́я диаграмма переходных процессов в триггере. Рекомендуется в виде упражнения самостоятельно построить эту диаграмму для показанных и любых других комбинаций входных сигналов и начальных состояний триггера. Первую попытку желательно сделать, еще не читая дальнейших объяснений. Это даст полезный навык построения диаграмм нетривиальных схем и поможет лучше усвоить особенности работы и использования этого триггера.

Подсказка. Начинать построение диаграммы нужно с тех элементов, состояния которых известны с самого начала, например с элемента *1*. Диаграмму доводить до любого нового входного воздействия, после чего учитывать его влияние на схему и т. д.

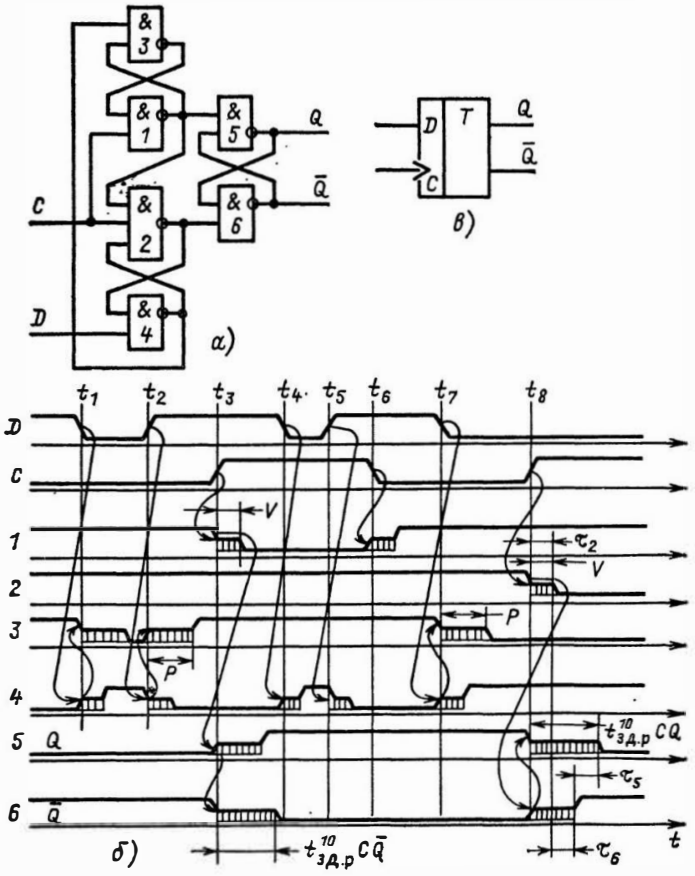


Рис. 6.14. Шестиэлементный непрозрачный *D*-триггер

На рис. 6.14, *б* временна́я диаграмма построена не на языке максимальных задержек переключающихся элементов, а на языке их зон неопределенности (см. рис. 5.3), что хорошо выявляет интервалы времени, захваченные переходными процессами. Однако следует иметь в виду, что *использование зон неопределенности при анализе схем с обратными связями порождает специфические трудности,* в которых схемотехник должен разбираться. Как уже говорилось, минимально возможное значение задержки элемента удобно полагать равным 0. Но на самом деле это все-

таки не так. И если для комбинационных схем без обратных связей это предположение не вступает в противоречие с выполняемой схемой функцией, упрощая к тому же расчеты и способствуя повышению надежности, то при рассмотрении схем триггеров, тем более непрозрачных, слишком буквальное следование этому положению может дать уже совершенно ошибочные результаты.

Дело в том, что борьба с гонками в непрозрачных триггерах и других нетактируемых схемах с обратными связями широко эксплуатирует один достаточно очевидный момент: сигнал, идущий в такой схеме по короткому проводу, всегда обгоняет сигнал, идущий через логический элемент. Другими словами, в основе работы триггерных схем лежит реальный факт обязательно ненулевой задержки логического элемента. А положение о нулевой задержке выравнивает предполагаемые времена распространения сигнала и по проводу, и через элемент, порождая подозрение в гонках там, где их в действительности нет.

Ситуацию можно проиллюстрировать на примере *JK*-триггера по рис. 6.11. Его непрозрачность на фронте *C*-сигнала базируется, в частности, на том, что если элемент *1* сработал, то его выходной нулевой уровень прямо по проводу попадает на вход элемента *5* и запирает его раньше, чем туда же попадет инициированный тем же элементом *1* открывающий единичный уровень с выхода элемента *3*. Поэтому элемент *5* в результате поступления фронта *C*-сигнала никогда не переключается, оставаясь в это время стабильно в 1. Но, с другой стороны, если для элементов *1*, *3* и *5* построить зоны неопределенности, начинающиеся с нулевой задержки, то формальный (а тем более программный) анализ таких зон допустит возможность ложного срабатывания элемента *5*. Действительно, поскольку значения сигналов на выходах элементов *1* и *3* в каком-то интервале не определены, то, говоря формально, они могут оба быть равными и 1, т. е. элемент *5* вполне может из-за этого переключиться в 0, переключив в 1 триггер *7-8* второй ступени непосредственно по фронту *C*-сигнала. А это квалифицируется как сбой. С помощью зон неопределенности, имеющих минимальные значения, равные 0, подобные ошибочные обвинения в неработоспособности можно вынести в адрес многих реально используемых схем непрозрачных триггеров.

Обобщая, можно сказать, что формальный анализ временных диаграмм, основанный на гипотезе о возможности

нулевой задержки элементов, не порождает никаких противоречий в схемах без обратных связей, однако в схемах с обратными связями эта гипотеза при выполнении формального анализа приводит к постулированию возможности гонок не только там, где они действительно могут быть, но и там, где их никогда не бывает. При разработке программ, моделирующих работу схем, эту трудность обходят различными путями (см. [32]), в частности и отказываясь от гипотезы о возможности нулевой задержки. Если смотреть строже, то, когда в триггерных схемах анализируются пути, разность хода сигнала по которым составляет всего один элемент, становится сомнительной сама правомерность отображения (моделирования) всех аспектов переходных процессов с помощью единственного понятия — задержка элемента по уровню 0,5. Требуются более полные модели, учитывающие длительности фронтов, вариации порогов срабатывания, характер развития переходного процесса в зависимости от энергии входного воздействия и т. д. Этим же объясняется и неоднозначность результатов при попытках точного определения значений интервалов подготовки и выдержки, если принять во внимание лишь задержку элемента по уровню 0,5.

Не следует относиться ко всему сказанному как к клубку досадных или загадочных противоречий. Вполне естественно, что простые схемы описываются более простыми моделями, допускающими нулевые значения задержки, а более сложные схемы, имеющие обратные связи, требуют более тонкого учета реальности, и предположение о нулевой задержке становится неадекватным.

При анализе схем без программных моделей («вручную») можно в качестве первого приближения построить временну́ю диаграмму, исходя из предположения о равных и притом максимально возможных значениях задержки всех элементов. Затем ввести в диаграмму линии причинно-следственных связей. И уже после этого, поняв поведение схемы и имея представление о последовательности переключения элементов, дополнить диаграмму зонами неопределенности.

Вернемся к шестиэлементному триггеру (рис. 6.14). Выходным триггером в этой схеме является RS-триггер на элементах *5-6*. Элементы *1*, *2* служат его входными конъюнкторами. При $C=0$ оба они заперты и триггер *5-6* не реагирует на изменения D-входа (моменты t_1, t_2, t_7), как это

и положено любому D-триггеру. D-триггер находится в режиме хранения.

По фронту C-сигнала (моменты t_3 и t_8) в зависимости от уровня на D-входе открывается один из конъюнкторов (*1* или *2*) и переключает триггер *5-6* (или подтверждает его состояние, если оно до этого совпадало с требуемым). Элементы *4* и *3*, управляемые D-входом, своими уровнями подготавливают один из конъюнкторов (*2* или *1*) для того, чтобы он открылся очередным C-сигналом (проследите по схеме воздействие элементов *3* и *4* на входы элементов *1* и *2* в моменты t_1, t_2, t_7).

Система связей элементов *1—4* обладает одним нетривиальным свойством. Переключившись на фронте C-сигнала в 0, элемент *1* по связи выход элемента *1* — вход элемента *2* блокирует для элемента *2* возможность открыться, даже если несколько позже, уже при $C=1$, вход D переключится в 0 (момент t_4). Элемент *2* не остается в долгу и, получив однажды шанс сработать на фронте C-сигнала (момент t_8), по цепочке *2—4—3—1* делает невозможным последующее срабатывание элемента *1*, даже в случае переключения входа D в 1. Продолжите временну́ю диаграмму за момент t_9, чтобы убедиться в этом.

Рассмотренное поведение триггера при $C=1$ показывает, что данный триггер *непроницаем по D-входу* ни для каких изменений его уровня. *Шестиэлементный D-триггер — это триггер, переключаемый перепадом.* Полезно сравнить временну́ю диаграмму по рис. 6.14, *б* с диаграммой прозрачного D-триггера, показанной на рис. 6.6, *г*, которая построена для той же серии входных воздействий. Изображение триггера, реагирующего не на уровень C-сигнала, а на его фронт, показано на рис. 6.14, *в*.

Схема шестиэлементного D-триггера чисто противогоночная, и она допускает для своих элементов любые сочетания задержек. Это видно, в частности, из временно́й диаграммы: все элементы переключаются строго один за другим. Тонкие линии причинно-следственных связей нигде не раздваиваются, что говорит об отсутствии параллельных путей, на которых возможны гонки. Поэтому схема оказывается очень удобной для применения в матричных БИС и в других ситуациях, где отсутствует нормирование минимальных значений задержек элементов.

Временна́я диаграмма на рис. 6.14, *б* позволяет оценить задержки всей схемы, если известны задержки элементов. В качестве примера на рисунке показаны компоненты

$t^{10}_{зд.р}$ CQ: эта величина слагается из суммы задержек элементов *2*, *6* и *5*.

Время подготовки $t_{пд}$, в течение которого вход D не должен изменяться, определяется временем исчезновения неопределенного состояния на всех входах конъюнкторов *1* и *2*. Самой длинной при этом является цепочка срабатывания элементов *4-3* (на рисунке — отрезок времени P). Время выдержки, в течение которого после фронта C-сигнала уровень D не должен изменяться, определяется временем, требуемым для того, чтобы элемент *1* заблокировал вход элемента *2* или чтобы элемент *2* заблокировал вход элемента *4* (интервал V после моментов t_3 и t_8). Это наиболее пессимистическая оценка при условии, что у элементов нормированы лишь максимальные значения задержек. Если же обеспечено положение, при котором задержка элемента *2* никогда не превышает задержки элемента *4*, то время выдержки обратится в 0, как и в двухступенчатом триггере.

Нарушение свойства проницаемости D-триггера возможно на срезе C-сигнала, если срез сильно затянут, а порог элемента *2* существенно выше порога элемента *1* (гонки по C-входу). Так, если триггер находился в 0, и если во время изменения C-сигнала на срезе от уровня порога элемента *2* до порога элемента *1* состояние D-входа переключится из 0 в 1, то элемент *1* выдаст ложный нулевой импульс, который, вопреки статусу непроницаемости, переключит пару элементов *5-6* в 1. Поэтому если срез C-сигнала пологий, то в течение этого среза уровень D-входа изменять нельзя.

Шестиэлементный D-триггер по числу элементов экономичнее большинства других непроницаемых триггеров и, что очень важно для триггеров, реализуемых на матричных БИС, обеспечивает непроницаемость для помехи по D-входу при $C=1$. При этом в отличие от двухступенчатого триггера он не предъявляет требований к нормированию ни разницы времен задержки определенных элементов, ни разницы их порогов переключения. Опасность гонок по входу уменьшают, включая в тракт C-сигнала перед его разветвлением на входы элементов *1* и *2* дополнительный инвертор. Благодаря крутой передаточной характеристике он сокращает длительности фронтов C-сигнала. Примером серийно выпускаемого шестиэлементного D-триггера может служить микросхема К155ТМ2.

Поскольку шестиэлементный D-триггер непрозрачен по

D-входу, на его основе можно построить T-триггер, введя инвертирующую обратную связь с выхода $\overline{Q}$ на вход D, как показано на рис. 6.15, *а*. На показанные штриховыми линиями входы J и K пока не нужно обращать внимания. По фронтам C-сигналов полученная схема будет переключаться, каждый раз инвертируя свое собственное состояние, т. е. будет вести себя как T-триггер, срабатывающий по фронту

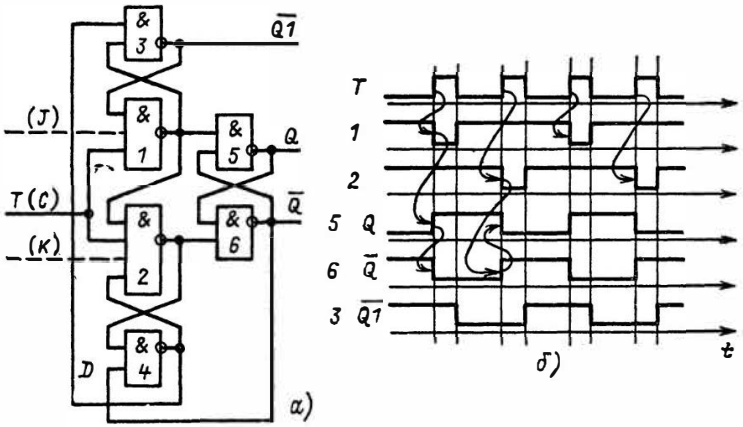


Рис. 6.15. T-триггер (и JK-триггер) на базе шестиэлементного триггера

(рис. 6.15, *б*). Необходимым условием грамотного превращения шестиэлементного D-триггера в T-триггер является следующее: задержка D-триггера по тракту вход C — выход $\overline{Q}$ должна быть больше времени выдержки $t_{вд}$ по D-входу. В противном случае T-триггер может давать сбои. Это требование накладывает ограничения на минимальные значения задержек некоторых элементов схемы, что легко выявить самостоятельно.

При работе T-триггера входные T-сигналы, каждый раз изменяющие состояние выходного триггера *5-6*, проходят через элементы *1* и *2* поочередно: четные через один, нечетные — через другой. На выходе каждого элемента входные импульсы оказываются пересчитанными на два (рис. 6.15, *б*), что можно использовать для построения счетчиков.

Как уже отмечалось, выходы элементов *5* и *6* изменяются по фронту T-сигнала. А вот состояние выхода элемента *3* в этом режиме изменяется по срезу синхросигнала (рис. 6.15, *б*), что дает основание для проведения аналогии

между шестиэлементным и двухступенчатым счетными триггерами: если выходы элементов *5* и *6* отождествить с выходами первой ступени *MS*-триггера, то выход элемента *3* будет тождествен выходу второй ступени (точнее, ее инверсии). Возможность одновременно фиксировать четность как фронтов, так и срезов входных импульсов также может найти применение в пересчетных схемах. Постройте более подробную, чем на рис. 6.15, *б*, временну́ю диаграмму работы шестиэлементного *T*-триггера (аналогичную диаграмме на рис. 6.14, *б*, используя сокращенную диаграмму рис. 6.15, *б* для контроля), тогда отмеченные особенности поведения схемы будут отчетливо видны. В силу асимметрии схемы детали процесса при четном и нечетном входных сигналах несколько различаются, поэтому строить диаграмму нужно как минимум для пары последовательных входных сигналов. Более подробное описание работы шестиэлементного *T*-триггера приведено в [18].

Если применить в качестве входных конъюнкторов *1* и *2* шестиэлементного триггера элементы с бо́льшим числом входов, то можно построить *JK*-триггер (рис. 6.15, *а*, с учетом входов, показанных штриховой линией, и их обозначений в скобках). Этот триггер при $C=1$ допускает проскок фронта по *J*- и *K*-входам, в чем легко убедиться, построив соответствующие временны́е диаграммы. Поэтому уровни на *J*- и *K*-входах не должны изменяться в течение всего интервала, пока $C=1$. Это требование не позволяет при передаче информации с одного триггера на другой использовать в качестве выходов триггера элементы *5* и *6*, поскольку их состояния изменяются как раз во время интервала $C=1$. Поэтому, применяя *JK*-триггеры по схеме рис. 6.15, *а*, приходится в качестве их выхода использовать выход элемента *3*. Выход $\overline{Q1}$ и есть инверсный выход *JK*-триггера, состояние которого изменяется не по фронту, как у элемента *6*, а по срезу *C*-сигнала. В таком виде этот *JK*-триггер по основным потребительским свойствам (непрозрачность, но проницаемость) эквивалентен схеме, показанной на рис. 6.11. При этом выход элемента *3* ($\overline{Q1}$) схемы по рис. 6.15, *а* соответствует выходу элемента *8* ($\overline{Q}$) схемы по рис. 6.11, а выходы элементов *5* и *6* на рис. 6.15, *а* соответствуют выходам *M*-ступени (элементы *3*, *4*) на рис. 6.11. В [33] приведены конкретные схемы счетчиков и сдвигающих регистров на шестиэлементных *JK*-триггерах.

На базе шестиэлементного триггера можно построить

и такой JK-триггер, который не имеет свойства проницаемости, т. е. переключается строго перепадом. Для этого используют принцип, частный случай которого проиллюстрирован на рис. 6.12. Этот принцип позволяет получить JK-триггер из любого непрозрачного D-триггера и заключается в том, что на место прямоугольника, обведенного на рис. 6.12 штриховой линией, нужно подставить любой непрозрачный D-триггер. В частности, туда можно включить и шестиэлементный D-триггер. Именно так построен переключаемый положительным перепадом $J\bar{K}$-триггер К155ТВ15, в нем лишь отсутствует показанный на рис. 6.12 инвертор в тракте K-входа, а элемент И-ИЛИ схемы на рис. 6.12 функционально объединен с элементом *4* схемы на рис. 6.14, *а* за счет использования одного элемента И-ИЛИ-НЕ с бо́льшим числом входов И.

6.5. Асинхронные входы триггеров

Непрозрачные триггеры кроме штатных входов — синхровхода C и управляющих входов D, J, K часто дополняют независимыми от них R- и S-входами. При этом схема строится так, что R- и S-входы имеют приоритет в своем воздействии на триггер по отношению к штатным входам, т. е. R- или S-входы устанавливают диктуемое ими состояние триггера независимо от сигналов, поступающих в это время на штатные входы, в том числе и на вход C. Поэтому такие R- и S-входы называют *асинхронными*. По окончании асинхронного сигнала установленное им состояние сохраняется вплоть до очередного активного фронта C-сигнала. По этому фронту триггер сработает уже в соответствии с этим установленным состоянием и с действующими в данный момент уровнями на штатных управляющих входах. Существенно, что асинхронными входами, т. е. входами, результат действия которых не зависит от уровня C-сигнала, можно снабдить лишь непрозрачные триггеры. Прозрачная защелка, построенная, например, по схеме рис. 6.6, *в*, не сможет сохранить установленное R-входом состояние, если R-сигнал окончился во время действия C-сигнала, поскольку из-за прозрачности на ее выходе тут же установится уровень D-входа.

Типовые схемы организации асинхронных R- и S-входов показаны на рис. 6.16. Как правило, эти входы имеют активный низкий уровень. В двухступенчатом триггере каждый асинхронный вход воздействует сразу на тригге-

ры обеих ступеней и на один из входных конъюнкторов (*1* или *2*), пресекая их возможное противодействие при активном уровне *C*-сигнала. В шестиэлементном триггере асинхронные сигналы перекрывают управляющие пути от *D*-входа, если уровень последнего при $C=1$ может им препятствовать, и устанавливают требуемое состояние в выходном триггере *5-6*. Подача сигналов на оба асин-

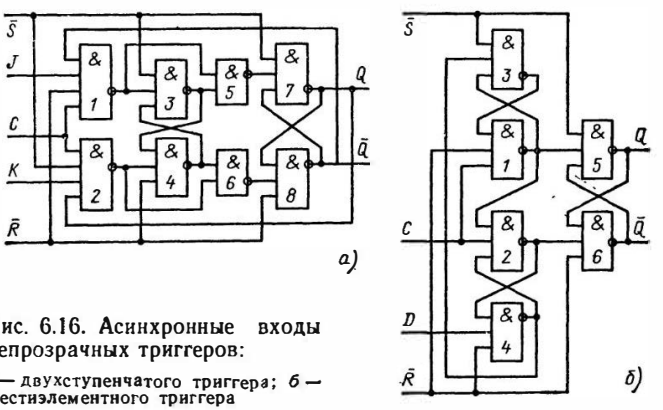


Рис. 6.16. Асинхронные входы непрозрачных триггеров:

а — двухступенчатого триггера; *б* — шестиэлементного триггера

хронных входа (и на *R*, и на *S*) и последующее одновременное их снятие приводят, как всегда, к неопределенному состоянию триггера. Наличие асинхронных входов существенно расширяет возможности разработчика схем, и этими входами снабжаются почти все триггеры, выпускаемые в виде микросхем. В качестве примера в табл. 6.3 отраже-

Таблица 6.3

Режим	Входы					Выходы	
	$\bar{S}$	$\bar{R}$	C	$\&J$	$\&K$	Q	$\bar{Q}$
Асинхронная установка	L	H	X	X	X	H	L
Асинхронное гашение	H	L	X	X	X	L	H
Не определен	L	L	X	X	X	H	H
Загрузка 1	H	H	↓	H	L	H	L
Загрузка 0	H	H	↓	L	H	L	H
Счетный	H	H	↓	H	H	q	$\bar{q}$
Хранение	H	H	X	L	L	Q	$\bar{Q}$

Примечание. X — безразличные состояния; стрелки — полярности переключающего перепада синхросигнала; q — состояние триггера непосредственно перед переключающим перепадом.

Таблица 6.4

Режим	Входы				Выходы	
	$\bar{S}$	$\bar{R}$	C	D	Q	$\bar{Q}$
Асинхронная установка	L	H	X	X	H	L
Асинхронное гашение	H	L	X	X	L	H
Не определен	L	L	X	X	H	H
Загрузка 1	H	H	↑	H	H	L
Загрузка 0	H	H	↑	L	L	H
Хранение	H	H	L	X	Q	$\bar{Q}$
Хранение	H	H	H	X	Q	$\bar{Q}$
Хранение	H	H	↓	X	Q	$\bar{Q}$

ны взаимоотношения между штатными и асинхронными входами JK-триггера К155ТВ1, а в табл. 6.4 — шестиэлементного D-триггера К155ТМ2.

6.6. JK-триггер, использующий задержку

На рис. 6.17, *а* показана еще одна схема JK-триггера, которая использована, например, в микросхемах К531ТВ9, К555ТВ9 и др. Из временно́й диаграммы (рис. 6.17, *б*) видно, что триггер не реагирует на фронт синхросигнала (момент t_1), а изменяет состояние своего выхода по срезу синхросигнала (момент t_4), что и отражено в условном обозначении этого триггера на рис. 6.17, *в*. Отсутствие изменений выхода при изменении J-входа в моменты t_2 и t_3 говорит об отсутствии проскока фронта при $C=1$. То, что изменения J-входа при $C=1$ не запоминаются никакой группой элементов, говорит о том, что триггер не обладает и свойством захвата. Таким образом, это непроницаемый триггер, т. е. триггер, переключаемый перепадом, в данном случае отрицательным.

Из временно́й диаграммы следует, что для правильной работы схемы задержка входных элементов *1* и *2* должна быть больше суммы задержек обоих элементов И-ИЛИ-НЕ, т. е. больше задержки переключения выходного RS-триггера. Если по срезу C-сигнала элемент *1* переключится в 1 не в момент времени t_6, а существенно раньше, например в момент t_5, когда на выходе элемента *8* еще держится 1, то на выходе конъюнктора *3* появится 1, что вызовет переключение выхода элемента *5* обратно в нулевое состояние. Начало развития этого процесса, который с большой вероятностью закончится сбоем, показано на времен-

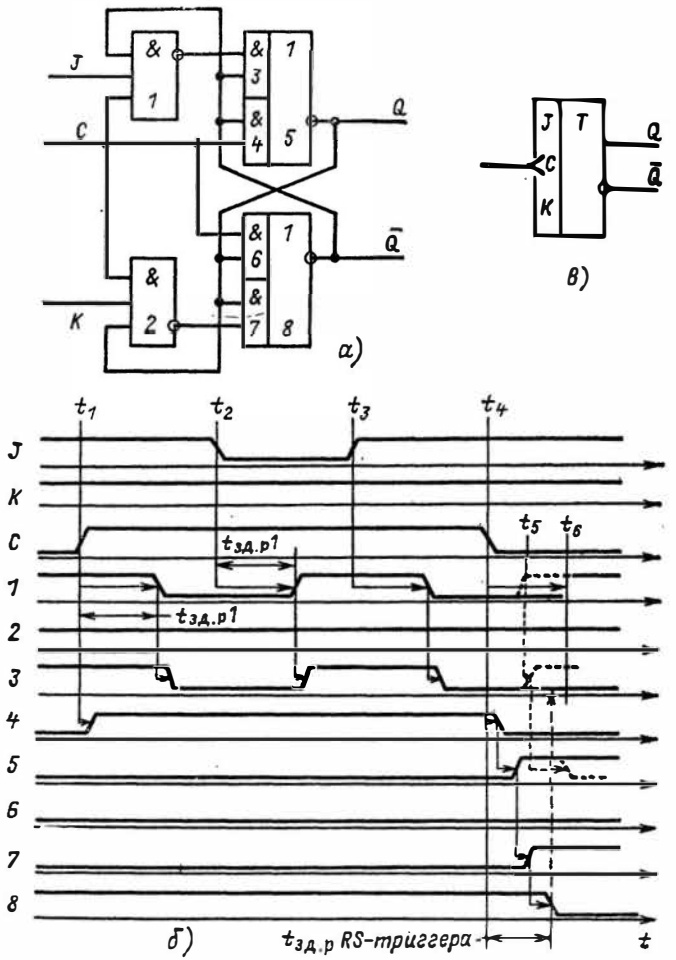


Рис. 6.17. *JK*-триггер, использующий большую задержку входных вентилей

нóй диаграмме пунктирными линиями. Таким образом, схема, показанная на рис. 6.17, не является противогоночной, и такие триггеры можно строить, лишь когда есть возможность выдерживать двусторонние допуски на значения задержек элементов, т. е. контролировать как максимальные, так и минимальные их значения.

Достоинством рассматриваемой схемы является также

нулевое значение времени выдержки $t_{вд}$. Правда, это достигается ценой увеличения времени подготовки $t_{пд}$, поскольку в состав этого интервала входит задержка одного из входных вентилей — *1* или *2*. Однако, как будет видно из гл. 7, именно нулевое значение времени выдержки очень важно для бессбойной работы однотактных схем.

Асинхронный R-вход с активным низким уровнем в JK-триггерах рассмотренного типа заводят на дополнительные конъюнктивные входы сразу трех элементов: *1*, *6* и *7*.

6.7. Классификация синхронных триггеров

Выбор для проектируемой схемы того или иного типа синхронного триггера тесно связан с параметрами системы синхронизации, о которых речь будет идти в гл. 7. Распространенные серии микросхем содержат по нескольку типов триггеров, к тому же многие серии взаимно совместимы, поэтому разнообразие триггеров, которыми схемотехник может располагать, обычно довольно велико. Работая на матричных БИС, схемотехник часто волен использовать вообще любую известную ему схему триггера. В то же время неграмотная организация взаимодействия триггеров различных типов может привести к очень неожиданным результатам. К сожалению, распространенная система обозначений и параметров триггеров, по которым они классифицируются в литературе, в том числе и в справочниках, не отражает в нужной мере свойств, существенных для схемотехника-пользователя. Например, К155ТМ7 и К155ТМ2 во многих справочниках называются одинаково — D-триггеры, хотя это — очень различные триггеры: первый — прозрачный, второй — нет. Распространенная классификация триггеров по числу ступеней (одноступенчатые и двухступенчатые) также не отражает важных для потребителя триггера свойств. По этой классификации в одну группу попадают такие по-разному используемые триггеры, как прозрачная защелка и шестиэлементный триггер, JK-триггер на двух RS-триггерах, проницаемый для помех при $C=1$, и двухступенчатая схема из двух D-защелок, работающая строго по перепаду. Неудачна традиция именовать (и соответственно изображать на схемах) C-вход шестиэлементных триггеров статическим, в то время как C-вход JK-триггеров именуется динамическим, поскольку оба триггера реагируют не на уровень, а на переключение C-сигнала.

Можно предложить систему параметров, которая непро-

тиворечива и значительно лучше характеризует триггер с точки зрения применения его в синхронных устройствах. Параметры этой системы описывают взаимные отношения сигналов на трех группах выводов триггера: на *синхровходе C;* на *управляющих входах D, J, K,* синхронизированных *R* и *S*; на *выходах Q* и $\bar{Q}$. В предыдущих параграфах эти параметры постепенно вводились на неформальном, понятийном уровне. Теперь будут даны их более строгие определения, показаны их взаимные связи и способ описания триггерных схем в системе этих параметров.

Синхронные триггеры взаимодействуют с двумя элементами синхросигнала: уровнем и фронтом. Уровень *C*-сигнала может быть пассивным и активным. *Пассивный уровень* закрывает конъюнкторы триггера, через которые проходят управляющие сигналы, в результате никакие изменения на управляющих входах принципиально не могут воздействовать на выход триггера. При пассивном уровне *C*-сигнала триггер всегда находится в режиме хранения. *Активный уровень* обеспечивает на управляющих конъюнкторах отпирающий потенциал, и прохождение через эти конъюнкторы управляющих воздействий регламентируется лишь внутренними сигналами обратных связей самого триггера. В различных схемах триггеров эти сигналы вырабатываются различными способами, что порождает широкий спектр реакций триггеров различных типов на уровни и переключения управляющих сигналов при активном уровне синхросигнала. Для схем триггеров, построенных на элементах И-НЕ, И-ИЛИ-НЕ, активным обычно является высокий уровень *C*-сигнала.

В соответствии с разделением уровней *C*-сигнала на пассивный и активный фронты этого сигнала с точки зрения вызываемых ими переключений в схеме триггера удобно разделить на: *передний фронт*, при котором уровень *C*-сигнала из пассивного становится активным; *задний фронт,* при котором уровень *C*-сигнала становится пассивным. Если активным уровнем *C*-сигнала является высокий, то передним фронтом является положительный перепад синхросигнала, задним — отрицательный, и наоборот.

Переключающий фронт триггера — фронт, вызывающий изменение состояния выходов триггера в соответствии с предварительно установленными уровнями на его управляющих входах. У различных триггеров переключающим может быть как передний, так и задний фронт активного уровня *C*-сигнала.

Кроме того, для разработчика, использующего триггер, существенным является и фактический знак (полярность) перепада, который вызывает смену уровней на выходе триггера, т. е. *знак переключающего перепада: положительный* — от *L* к *H* или *отрицательный* — от *H* к *L* не-

зависимо от того, является этот перепад передним или задним фронтом активного уровня синхросигнала.

Прозрачность — свойство триггера при активном уровне *C*-сигнала адекватно в соответствии с его типом (например, *RS*) отслеживать на выходе все переключения управляющих входов; если это *D*-триггер, то просто повторять состояния *D*-входа. Можно сказать, что прозрачный триггер при активном *C*-уровне ведет себя как асинхронный. Встречающиеся в некоторых источниках термины «триггер, переключаемый уровнем» или «триггер со статическим входом» при их правильном употреблении эквивалентны термину «прозрачный триггер».

Прозрачность в таком понимании может быть лишь свойством триггера, переключаемого передним фронтом *C*-сигнала, ибо если триггер переключается только на заднем фронте, то для проявления свойства прозрачности у него в силу определения просто не остается времени.

Непрозрачность — свойство триггера даже при активном уровне *C*-сигнала не передавать на выход изменений управляющих входов, происшедших вскоре после переключающего фронта *C*-сигнала, вплоть до поступления очередного переключающего фронта. Если переключающий фронт — задний, то сразу после него уровень *C*-сигнала становится пассивным. Этого уже достаточно, чтобы никакие изменения управляющих уровней, происшедшие вскоре после переключающего фронта, не проникли на выход триггера. Поэтому, чтобы триггер имел статус непрозрачного, от него требуется, чтобы после окончания изменений на входе установившиеся состояния управляющих уровней не попали на выход даже тогда, когда синхросигнал снова станет активным, вплоть до поступления его заднего, т. е. переключающего, фронта.

Если переключающий фронт передний, то у непрозрачного триггера даже в течение активного уровня *C*-сигнала, непосредственно следующего за переключающим фронтом, на выход не передается то новое состояние, в которое могли переключиться его управляющие уровни. Тем более не передадутся на выход изменения во время последующего неактивного периода *C*-сигнала.

Свойство непрозрачности — очень важное свойство триггера, поскольку оно позволяет строить счетные триггеры и организовывать такие сети из триггеров (цифровые автоматы), в которых информация может передаваться с одного триггера на другой — непосредственно или через комбинационную схему, причем переключения выходов всех триггеров этой сети синхронизируются единой последовательностью *C*-сигналов. Читатель, уже знакомый с цифровой техникой, узнает здесь принцип однофазной синхронизации, о котором подробнее будет рассказано в гл. 7.

Проскок фронта—свойство триггера, переключаемого задним фронтом, при активном *C*-уровне передавать на выход не все (в отличие от

проявления прозрачности), но некоторые (в частности, лишь повторные или кратные) переключения управляющих входов при некоторых состояниях триггера. О проскоке фронта имеет смысл говорить лишь применительно к непрозрачным триггерам, поскольку прозрачность в силу ее определения требует *адекватной* передачи уровня со входа на выход. Проскок фронта — вредное явление. Если триггер допускает проскок, то на схему налагается дополнительное ограничение: управляющие уровни триггера не должны изменяться при активном уровне *С*-сигнала.

Свойство захвата 1 и (или) 0 также характеризует триггеры, переключаемые задним фронтом. Это — свойство триггера, не пропуская сразу на выход, запоминать короткие помехи, появляющиеся на управляющих входах при активном *С*-уровне при некоторых состояниях триггера и затем по заднему фронту *С*-сигнала передавать какую-то логическую функцию этих помех на выход. Хотя в отличие от эффекта проскока эффект от захвата проявляется не сразу, окончательный результат оказывается тем же: ошибка на выходе. И меры, предпринимаемые схемотехником при использовании триггеров с захватом, те же: запрещение изменения входных сигналов триггера при активном уровне *С*-сигнала.

Таким образом, для разработчика цифровых устройств, который использует триггеры в виде законченных функциональных узлов, различие между явлениями проскока и захвата несущественно. Ему важна лишь дизъюнкция этих свойств: при наличии любого из них или обоих вместе схемотехник вынужден использовать одни схемные решения, а при полном их отсутствии может использовать и другие. Различие становится существенным при поиске неисправностей в схемах, проектировании схем новых триггерных устройств и в других случаях, когда разработчику приходится рассматривать значительно более широкое поле вариантов поведения схемы, далеко выходящих за рамки набора типовых реакций, предусмотренных таблицей истинности справочника.

Проницаемость триггера или, полнее, проницаемость для помех при активном уровне *С*-сигнала — наличие у триггера свойств проскока фронта или захвата. *Если триггер проницаем,* то схема цифрового устройства должна быть спроектирована так, чтобы *управляющие входы изменялись лишь при неактивном уровне С-сигнала, а при активном его уровне оставались постоянными.* В триггерных сетях, синхронизируемых единым *С*-сигналом (в однофазных), это достаточно легко осуществить, когда используются триггеры, переключаемые задним фронтом *С*-сигнала, поскольку при этом смена управляющих уровней на входах триггеров (вызванная сменой выходных уровней других триггеров — источников информации) происходит, когда *С*-сигнал стал уже неактивным и поэтому вредное свойство проницаемости проявиться не может. Если же сеть построена на основе триггеров, переключаемых передним фронтом, то смена состояний управляющих входов будет про-

исходить как раз при активном состоянии *C*-сигнала. Если при этом триггеры проницаемы, то новый входной сигнал триггера может тут же проникнуть на его выход и схема даст сбой. Поэтому переключаемые передним фронтом проницаемые триггеры реально не используются. Напротив, триггеры проницаемые, но переключаемые задним фронтом, используются довольно широко.

Триггеры, *переключаемые перепадом*, — это триггеры, не обладающие свойством проницаемости, *непроницаемые* триггеры. Уровни управляющих входов у них могут изменяться как угодно, в том числе и многократно, без опасности повредить состояние выхода. Переключения входов могут происходить при любом уровне синхросигнала, кроме короткого интервала подготовки $t_{пд}$ непосредственно перед переключающим перепадом и интервала выдержки $t_{вд}$ сразу за ним. В принципе для этих триггеров также можно выявить активный уровень *C*-сигнала, открывающий вентили для прохождения управляющих сигналов, и пассивный, закрывающий их. Но в связи с тем, что при активном уровне никаких изменений на выходе триггера этого типа никогда не происходит и вообще активный уровень никак внешне себя не проявляет, деление уровней *C*-сигнала на активный и пассивный для разработчика цифровых устройств теряет актуальность. Как следствие ненужной становится и идентификация переключающего фронта как переднего или заднего. Единственно, что для разработчика остается актуальным, это фактическая полярность переключающего перепада — положительная или отрицательная.

Если разработчик использует проницаемый триггер, то в зависимости от ситуации ему должны быть известны значения одного или двух из следующих трех параметров: на каком уровне *C*-сигнала свойство проницаемости может проявиться, т. е. какой уровень *C*-сигнала для этого триггера является активным; вид переключающего фронта данного триггера, поскольку возможности использования проницаемых триггеров, переключаемых передним или задним фронтом, существенно различны; фактическая полярность переключающего перепада *C*-сигнала — положительная (фронт) или отрицательная (срез).

В табл. 6.5 приведены достаточно очевидные взаимные соотношения между упомянутыми тремя параметрами. Как видно из таблицы, соот-

Таблица 6.5

Активный *C*-уровень	Переключающий фронт	Фактическая полярность переключающего перепада
L	Передний	↓
L	Задний	↑
H	Передний	↑
H	Задний	↓

ношения таковы, что, зная для некоторого триггера значения любых двух параметров, можно (по таблице, или путем несложных рассуждений) восстановить значение и третьего. Чаще требуются сведения, содержащиеся в первой и третьей колонках, и их хотелось бы видеть в справочниках. По известному же значению лишь одного из этих трех параметров восстановить значения двух других уже невозможно. Тем не менее традиционно почти во всех справочниках приводится лишь значение фактической полярности переключающего перепада. Правда, квалифицированный схемотехник может определить все три параметра, изучив функциональную схему, но во-первых, она приводится далеко не во всех справочниках. Во-вторых, различные справочники иногда предлагают и различные схемы. Сравните, например, схемы триггера К155ТВ1 в [8 и 10]: в первом случае структура схемы близка к схеме по рис. 6.11, во втором — к схеме по рис. 6.10, *а* с безынерционным инвертором ***C***-сигнала. В-третьих, нерационально требовать от всех разработчиков столь высокой квалификации.

Существенную помощь схемотехнику могло бы оказать уже высказывавшееся соображение о том, что микросхемы триггеров, переключаемых передним фронтом, выпускаются только непроницаемыми, если это соображение дополнить сведениями о том, обладает или нет данный триггер свойством проницаемости. Однако, если некоторые триггеры более поздних разработок уже иногда начали сопровождаться в справочниках пометкой «переключаемый перепадом, то информация о степени проницаемости таких распространенных триггеров, как К155ТМ2, К155ТВ1 и многих других, отсутствует. Более того, сами понятия о присущих некоторым схемам триггеров характерных помехах из-за проскока фронта или захвата почти не встречаются в литературе и очень мало известны в среде разработчиков. Этими сведениями обладает в основном узкий круг специалистов по имитационному моделированию схем, и то лишь в форме конкретных результатов тестирования конкретных схем. Такое положение порождает неграмотное использование триггеров в схемах и как следствие приводит к трудно диагностируемым сбоям в аппаратуре, заложенным уже на стадии логического проектирования.

Необходимость отражения в справочниках отсутствия или наличия проницаемости, а также уровня *C*-сигнала, при котором она может проявиться, представляется весьма назревшей.

Как вывод из всего сказанного можно предложить набор из пяти параметров, который описывает самые главные, структурные аспекты поведения триггера в синхронном устройстве:

1. Логический тип триггера: *D*, *JK* и т. п.
2. Знак переключающего перепада: положительный или

отрицательный. Для прозрачных триггеров фиксирующим является перепад, противоположный переключающему.

3. Степень прозрачности триггера: прозрачный или непрозрачный.

4. Для непрозрачных: переключаемый перепадом или обладающий свойствами либо проскока фронта, либо захвата. Последние два свойства часто можно не разделять, назвав их дизъюнкцию общим термином «проницаемость».

5. Для триггеров, обладающих свойством проницаемости: уровень синхросигнала, при котором это свойство проявляется (активный уровень) — высокий или низкий.

Значения этих параметров содержат тот минимум информации, который необходим для грамотного выбора триггера, совершенно не обращаясь при этом к его функциональной схеме и даже не зная принципов ее работы. Эта система параметров представляется полезной для введения ее в справочники по микросхемам и библиотеки функциональных узлов базовых матричных кристаллов.

В качестве примера в табл. 6.6 приведены значения этих параметров для рассмотренных схем синхронных триггеров. Таблица поможет усвоению введенных понятий, а также грамотному выбору схемы триггера применительно к параметрам системы синхронизации.

6.8. Регистры и регистровая память

Для запоминания многоразрядных слов необходимое число триггеров объединяют вместе (рис. 6.18, *а*) и рассматривают как единый функциональный узел — *регистр*. Если регистр построен на триггерах-защелках, то его называют *регистр-защелка*. Типовыми внешними связями регистра являются информационные входы D_i, вход сигнала записи (или загрузки) C, вход гашения R, выходы триггеров Q_i. В упрощенном варианте регистр может не иметь входа гашения инверсных выходов.

Условным изображением регистра по рис. 6.18, *б* пользуются тогда, когда на схеме необходимо показать каждый вход и выход данных. Если же тракт данных рассматривается как единое, укрупненное понятие — *шина данных*, то пользуются обозначением, показанным на рис. 6.18, *в*. Микросхемы К155ТМ5, К155ТМ7, К561ТМ3 можно рассматривать не только как четверки D-триггеров-защелок, но и как 4-разрядные регистры-защелки. Микросхема

№ п/п	Вид триггера	Логический тип	Знак переключающего перепада	Степень прозрачности	Степень проницаемости для помех	Активный уровень *C*-сигнала
1	Синхронный *RS* по рис. 6.5 *а*	*RS*	↑	Прозрачный	—	*H*
2	*D*-защелка по рис. 6.6, *б, в*; К155ТМ7	*D*	↑	То же	—	*H*
3	*D*-защелка по рис. 6.7, *а*	*D*	↓	»	—	*L*
4	Синхронный *RS* по рис. 6.9, *а*	*RS*	↓	Непрозрачный	Захват	*H*
5	*JK* по рис. 6.10; К155ТВ1	*JK*	↓	То же	То же	*H*
6	*JK* по рис. 6.11	*JK*	↓	»	Проскок	*H*
7	*JK* по рис. 6.12	*JK*	↓	»	Переключаемый перепадом	—
8	К561ТВ1	*JK*	↑	»	То же	—
9	*D*-триггер по рис. 6.13, *а*, в качестве триггера *TT* — триггер по рис. 6.10, *а*	*D*	↓	»	»	—
10	*D*-триггер по рис. 6.13, *а*, в качестве триггера *TT* — триггер по рис. 6.11, *а* или *б*	*D*	↓	»	Проскок	*H*
11	*D*-триггер шестиэлементный по рис. 6.14; К155ТМ2	*D*	↑	»	Переключаемый перепадом	—
12	*JK*-триггер на базе шестиэлементного по рис. 6.15, *а*, выход с элементов *5* и *6*	*JK*	↑	»	Проскок	*H*
13	*JK*-триггер на базе шестиэлементного по рис. 6.15, *а*, инверсный выход с элемента *3*	*JK*	↓	»	То же	*H*
14	*JK* по принципу рис. 6.12, в качестве непрозрачного триггера — шестиэлементный триггер по рис. 6.14; К155ТВ15	*JK*	↑	»	Переключаемый перепадом	—
15	*JK* по рис. 6.17; К555ТВ9	*JK*	↓	»	То же	—

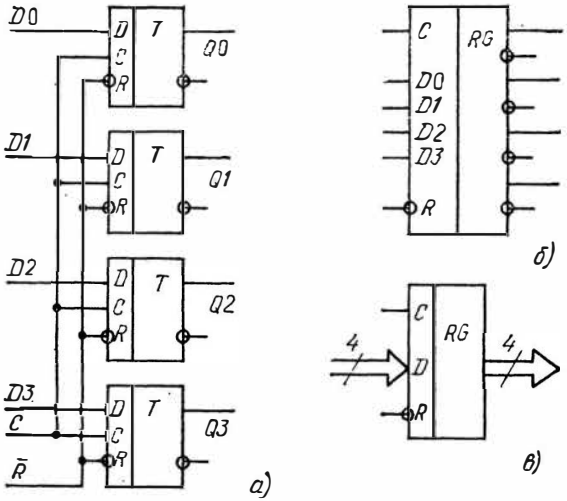


Рис. 6.18. Регистр:
а — схема; *б*, *в* — условное обозначение

К155ТМ8—4-разрядный регистр на базе непрозрачных триггеров, переключаемых положительным фронтом.

Выпускаемые промышленностью регистры иногда объединяют на кристалле микросхемы с другими узлами, в паре с которыми регистры часто используются в схемах цифровой аппаратуры. Пример такого комплексного узла — микросхема *многорежимного буферного регистра (МБР)* К589ИР12, основу которой составляет 8-разрядный регистр-защелка с входами *D0—D7*, *C*, $\overline{R}$ и восьмью выходами *Q0—Q7*, снабженными усилителями мощности *(буферами)* с тремя состояниями выхода. Кроме того, в состав микросхемы входят несколько элементов управления. Усилители с тремя состояниями выхода имеет и 4-разрядный регистр К155ИР15, построенный на непрозрачных триггерах без свойств захвата или проницаемости, т. е. управляемых строго фронтом.

Существуют микросхемы, в которых регистр объединен с входным мультиплексором, позволяющим принимать входные данные с двух и более направлений, выбираемых сигналами на адресных входах микросхемы. Объединяют регистр и с выходным демультиплексором, позволяющим передавать содержимое регистра на различные направления.

Сразу несколько регистров содержат микросхемы *регистровой памяти (register memory, register file, сверхоперативная память)*. Схема комбинированного узла такого типа

показана на рис. 6.19. Входы D_i четырех или восьми регистров, как правило 4-разрядных, подключены к общей входной шине данных *DIN (data in)*. Вход загрузки требуемого регистра выбирается дешифратором записи *DCW* на основании поступающего на вход *DCW* адреса записи *WA (write address)*, т. е. кода номера загружаемого регистра. За-

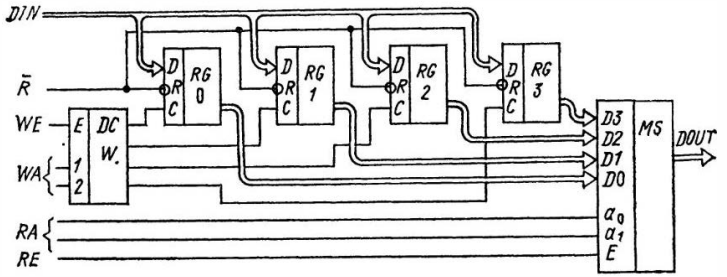


Рис. 6.19. Схема регистровой памяти

пись данных, присутствующих на шине *DIN*, происходит в момент поступления сигнала разрешения записи *WE (write enable)*.

Выходы регистров мультиплексором *MS* подключаются к выходной шине *DOUT (data out)*. Номер регистра, с которого происходит чтение, определяет код адреса чтения *RA (read address)*. Выдачу данных в шину *DOUT* разрешает сигнал *RE (read enable)*.

Поскольку дешифрация адреса записи и адреса чтения производится двумя независимыми узлами, имеющими автономные адресные входы *WA* и *RA*, регистровая память может одновременно записывать число в один из регистров и читать число из другого. Описанная структура использована в кристалле микросхемы К155РП1. Аналогичные, но более развитые структуры имеют регистровые памяти ИР11 и ИР12 серий К561 и 564 (см. [11]).

Микросхемы регистровой памяти легко наращиваются по разрядности и допускают наращивание по числу регистров. Они разработаны для построения блоков *регистров общего назначения (РОН)* и других специализированных блоков памяти небольшого объема, предназначенных для временного хранения исходных данных и промежуточных результатов в цифровом устройстве.

По мере увеличения числа регистров памяти разработ-

чики отказываются от независимой адресации регистров при записи и чтении. Остается лишь один комплекс адресных входов и один дешифратор адреса, которые используются и при записи, и при считывании. Такую схему регистровой памятью уже не называют. По ЕСКД она обозначается *RAM (random access memory,* т. е. *память с произвольным доступом)*. Используются также термины: *запоминающее устройство с произвольной выборкой* (ЗУПВ), *оперативное запоминающее устройство* (ОЗУ), *оперативная память,* а иногда — просто память. В микросхемах ЗУПВ ввод и вывод данных при записи и чтении могут осуществляться через одни и те же выводы корпуса за счет использования в тракте считывания элементов с тремя состояниями выхода или с открытым коллектором. Режимы работы микросхемы *запись, чтение* и *хранение* задаются комбинациями сигналов на ее входах управления. Если для ввода данных при записи и вывода их при чтении используются различные выводы корпуса (входы D_i и выходы Q_i), то режим хранения может быть совмещен с режимом чтения.

Микросхемы ОЗУ малой емкости часто выпускаются в составе распространенных серий. Они имеют входы адреса A_j; входы данных D_i; вход режима W/R: запись или чтение (он же и хранение); выходы данных Q_i; вход (или несколько конъюнктивных входов) разрешения E, по принятой среди специалистов по памяти терминологии чаще называемый *выбор кристалла ВК, выбор микросхемы ВМ* или *CS (chip select)*. Такую микросхему можно рассматривать как группу регистров, дешифратор для их выборки, цепи записи в регистры и считывания с них. Каких-либо специальных знаний для работы с микросхемами малой емкости, примерно до 1024 бит, не требуется. Примерами подобных микросхем могут служить К155РУ2 емкостью 16×4 (16 слов по 4 разряда), К155РУ5 — 256×1, 564РУ2—256×1. Наращивание разрядности и числа хранимых слов производится, как и в случае постоянных ЗУ (см. рис. 3.24).

Микросхемы ЗУПВ большей емкости выпускают уже в составе определенных серий БИС памяти. Часто такие микросхемы имеют временную диаграмму с большим числом регламентированных интервалов, адрес может подаваться по частям, есть микросхемы, требующие регенерации (подновления) хранимых данных, и т. п. Для построения схемы управления такими запоминающими устройствами нужны некоторые специализированные знания. С подходами к построению ОЗУ на микросхемах большой емкости можно ознакомиться по [20, 23].

6.9. Буферы типа «очередь» и «магазин»

Кроме адресуемых регистров общего назначения в цифровой аппаратуре используются небольшие вспомогательные запоминающие устройства с *неявно* выраженной адресацией, служащие для хранения очередей и называемые иногда *буферами данных*. Часто их строят на основе регистровой памяти.

Для организации обычной очередности служит буфер типа *очередь*, или буфер *FIFO* (*first in — first out* — первым вошедший первым выходит, читается «фифо»). Необходимость в таком буфере возникает, когда источник данных поставляет приемнику слова, распределенные во времени нерегулярно, причем интервалы времени между некоторыми словами могут быть меньше, чем время, необходимое приемнику для обработки одного слова. Если потери информации недопустимы, то между источником и приемником включается буфер *FIFO*, в котором хранится очередь слов, ожидающих обработки.

Схема буфера типа «очередь», построенного на основе регистровой памяти, показана на рис. 6.20, *а*. На рис. 6.20, *б* набор регистров памяти, т. е. адреса памяти показаны в виде кольца. Часть регистров занята очередью, остальные—свободный резерв на случай ее увеличения. Адрес записи при постановке в очередь задается счетчиком хвоста очереди *СТХВ* (о счетчиках см. гл. 9). Сигнал *Поставить в очередь*, поступая на вход *WE* разрешения записи, записывает поступившие по входной шине *DI* данные в тот регистр памяти, номер которого хранится в *СТХВ*. По срезу сигнала *Поставить в очередь* выходной код счетчика хвоста увеличивается на 1, подготавливая адрес записи для очередного сигнала *Поставить в очередь*.

При поступлении сигнала *Извлечь из очереди* на выходной шине *DO* появляется слово, хранящееся в том регистре памяти, номер которого задан кодом счетчика головы очереди *СТГОЛ*. По срезу сигнала выходной код счетчика увеличится на 1, подготовив для выдачи следующее слово, ставшее теперь первым в очереди. Переполнение счетчика хвоста очереди осложнений не вызовет, поскольку после максимально возможного кода счетчика ВСЕ ЕДИНИЦЫ в нем автоматически появится код ВСЕ НУЛИ. Очередь в своем кольце просто переползет хвостом через нулевую отметку счетчика. Так же со временем переползет и голова. В процессе нормальной работы очередь двигается в кольце значений адресов по часовой стрелке, хвостом вперед, удлиняясь или укорачиваясь в соответствии с флюктуациями активности передатчика. Перед началом работы оба счетчика сбрасываются в нуль.

Схема буфера *FIFO* должна сигнализировать о двух особых ситуациях. Первая — *буфер полон*, тогда в него нельзя больше записывать, и нужно приостановить передатчик. Вторая — *буфер пуст*, тогда из него нельзя брать данные, и нужно приостановить приемник. Обе ситуа-

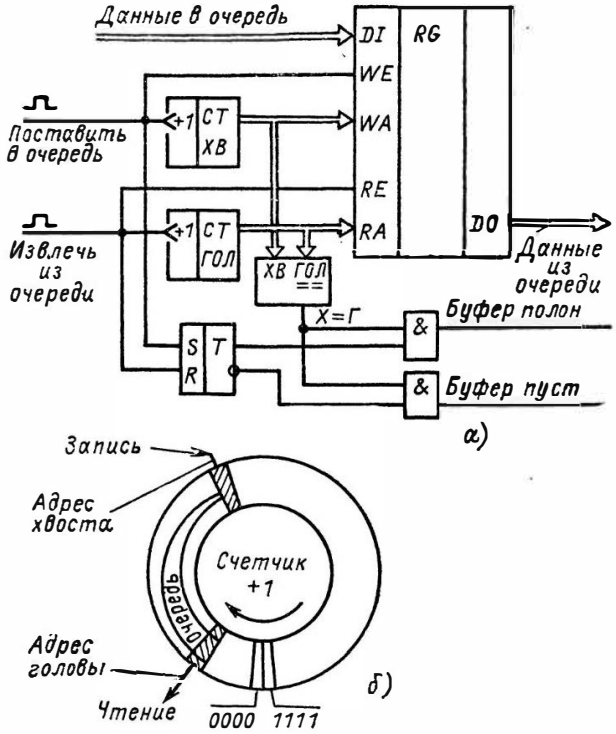


Рис. 6.20. Буфер типа «очередь» (буфер *FIFO*):
а — функциональная схема; *б* — диаграмма использования адресов

ции имеют общий признак: равенство показаний обоих счетчиков после исчезновения входного сигнала. Этот признак выявляет компаратор. Если счетчики стали равны после очередного извлечения из очереди, то это значит, что очередь иссякла, буфер пуст. Если они стали равны после очередной постановки в очередь, то буфер полон. Характер последнего обращения к буферу запоминается в *RS*-триггере. Сигналы, информирующие устройство управления об особых состояниях буфера, получаются как конъюнкции того или иного состояния триггера и признака равенства показаний счетчиков головы и хвоста.

Другим часто используемым в цифровой технике буфером является буфер типа *магазин*, или *стек* (*stack*), или буфер *LIFO* (*last in — first out*), последним вошедший первым выходит, читается «лифо»). В отличие от нормальной очереди здесь в качестве первого кандидата на обслуживание выбирается то слово, которое встало в очередь последним. По такому закону заряжается патронами и освобождается

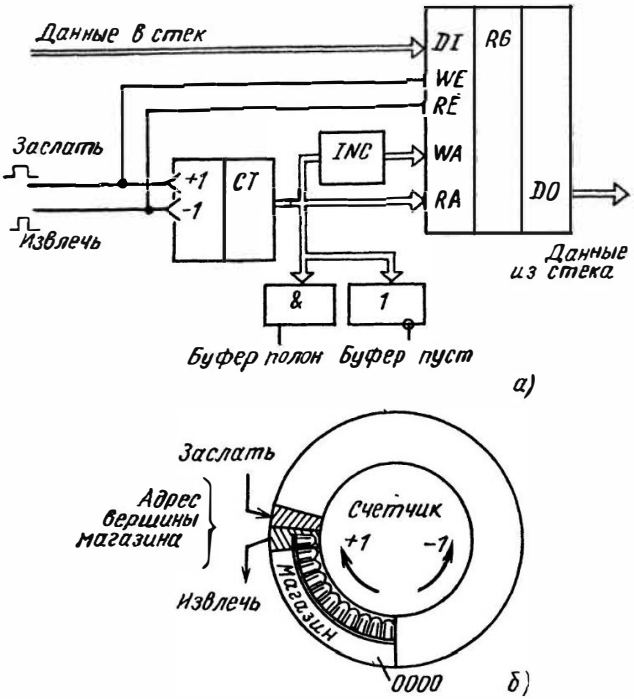


Рис. 6.21. Буфер типа «магазин» (буфер *LIFO*):
а — функциональная схема; *б* — диаграмма использования адресов

в процессе работы магазин автоматического оружия, откуда и произошло одно из названий. *Stack* в переводе значит скирда сена, поленница дров, т. е. вообще такое хранилище, которое, загружаясь сверху, разгружается также сверху, в обратном порядке. В нашем языке нет эквивалента слову *stack*, но кто хоть раз пытался разобрать поленницу снизу, согласится, что такой обобщающий термин полезен. Стековые структуры данных возникают в цифровых устройствах, когда процесс выполнения менее срочного задания прерывается более срочным и все данные, связанные с прерванной работой, засылаются на временное хранение в буфер типа «магазин». Выполнение срочного задания может быть в свою очередь прервано поступлением сверхсрочного, и т. д. — и в стеке по мере погружения туда (сверху) новых и новых данных формируется очередь слов, стековый порядок извлечения которых (тоже сверху) соответствует правильной по рангу срочности очередности их обработки.

Аппаратная реализация буфера типа «магазин» показана на рис.

6.21, *а*. Основу его также составляют регистровая память и счетчик адреса, в котором хранится номер регистра вершины стека (рис. 6.21, *б*). В отличие от буфера ранее рассмотренного типа здесь счетчик должен быть реверсивным, т. е. уметь прибавлять 1, когда поступает команда *Заслать в стек (push)*, и вычитать 1 при команде *Извлечь из стека (pop)*. Как видно из диаграммы на рис. 6.21, *б*, адрес, по которому производится засылка в стек, всегда на единицу больше адреса, по которому выполняется чтение из стека. Постоянный сдвиг на единицу адреса записи относительно адреса чтения выполняет инкрементор *INC*. Стек, как и буфер типа «очередь», также имеет два особых состояния: *буфер пуст* и *буфер полон*. Обнаруживаются они непосредственно по нулевому и по максимально возможному (все единицы) состояниям счетчика адреса.

Возможен и другой вариант аппаратной реализации стека — на основе реверсивных сдвигающих регистров (см. § 10.1). Число регистров равно разрядности засылаемых в стек слов, а число разрядов регистров определяет *глубину* стека, т. е. его емкость. Кроме того, и в стеке, и в буфере типа «очередь» можно вместо регистровой памяти использовать память с произвольным доступом. Это несколько усложнит схему из-за необходимости коммутации двух различных адресов на единый адресный вход, однако при требовании большой емкости буфера усложнение может оказаться оправданным. Как всегда, окончательный выбор можно сделать, лишь оценив в каждом конкретном случае временные параметры и аппаратурные затраты.

ГЛАВА 7

СИСТЕМА СИНХРОНИЗАЦИИ

7.1. Система двухфазной синхронизации

Подавляющее большинство цифровых устройств использует синхронный принцип работы. Рассмотрение систем синхронизации проще всего начать с *двухфазной* (или *двухтактной*) синхронизации, когда все схемы устройства тактируются двумя взаимно перемежающимися во времени последовательностями *синхроимпульсов (clock pulses) C1* и *C2* (рис. 7.1). Их получают от единого задающего генератора. Какой-либо информации эти импульсы не несут, они служат только для привязки ко времени всех процессов цифрового устройства. Система синхронизации характеризуется *длительностью тактового периода* $T_т$, *длительностью фазового*

периода $T_ф$ *и длительностью синхроимпульса* $T_и$. Для симметричной двухфазной синхронизации $T_т = 2T_ф$. Схема на рис. 7.1, *а* иллюстрирует основные черты структуры цифрового устройства, существенные для идеи синхронизации: все логические схемы устройства разделены на два класса— *триггеры и комбинационные схемы*.

Для двухфазной синхронизации характерно применение триггеров-защелок, например показанных на рис. 6.6, *б* или

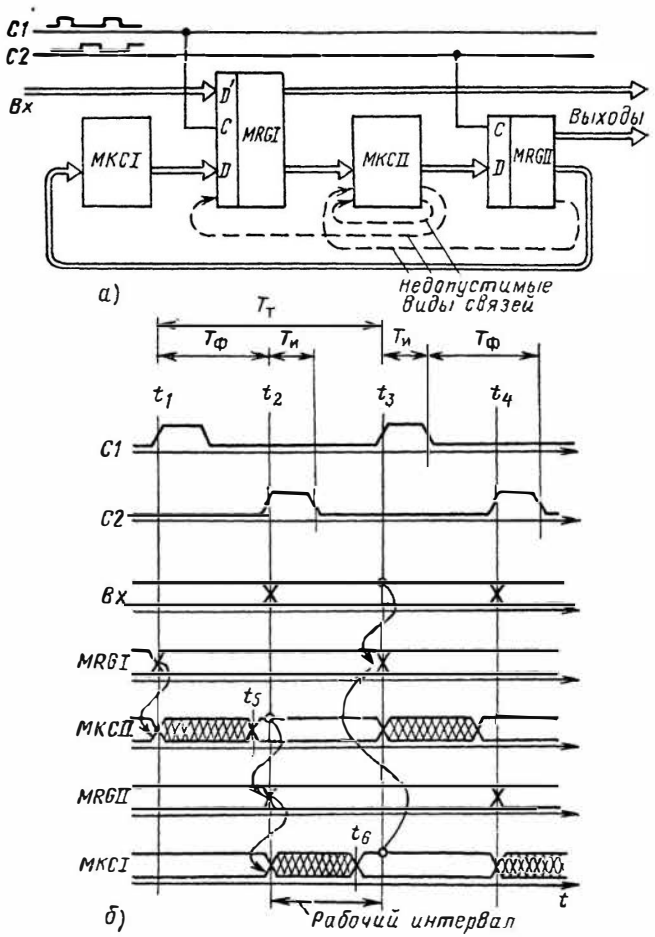


Рис. 7.1. Схема (*а*) и временна́я диаграмма (*б*) системы двухфазной синхронизации

в. Всю совокупность триггеров, синхронизируемых импульсами *C1*, можно рассматривать как один большой *макрорегистр*-защелку, обозначенный на рис. 7.1, *а* как *MRGI*. То, что *MRGI* включает в себя множество самостоятельных регистров, имеющих свои собственные имена и не имеющих между собой смысловой связи, значения не имеет. Важно лишь, что все они срабатывают по *C1*. Аналогично все триггеры, синхронизируемые *C2*, рассматриваются как макрорегистр *MRGII*.

Комбинационные схемы (КС) — это логические схемы, в составе которых нет триггеров (они все учтены в регистрах), а также любых цепей обратной связи. Информационные процессы в КС распространяются только в одном направлении — от входов к выходам. Примерами КС могут служить узлы свертки по четности, дешифраторы, мультиплексоры, сумматоры, а также любые их комбинации, не образующие петель обратной связи. Каждая КС получает информацию с выходов триггеров, синхронизируемых одной фазой синхронизации, а выходы этой КС подключены к *D*-входам триггеров, синхронизируемых другой фазой. Именно это, как было показано в § 5.2, позволяет исключить влияние гонок.

Все комбинационные схемы, выходы которых подключены к входам триггеров *MRGI*, можно рассматривать вместе как одну большую многовыходную *комбинационную макросхему*, обозначенную на рис. 7.1, *а MKCI*. Аналогично вся совокупность схем, выходы которых подключены к входам триггеров *MRGII*, рассматривается как макросхема *MKCII*. Реализуемые макросхемами *MKCI* и *MKCII* функции в данном случае значения не имеют, важен лишь характер подключения их к регистрам. В частности, некоторые тракты КС могут не содержать ни одного логического элемента и передавать без изменения уровни выходов триггеров одного регистра на входы триггеров другого.

Для определенности предположим, что сигналы, поступающие на вход устройства извне, принимаются на некоторые триггеры *MRGI*, причем изменяться эти сигналы могут лишь по фронту *C2*. Как это сделать, будет сказано в гл. 8.

Суть процессов, происходящих в системе двухфазной синхронизации, в общих чертах такова (рис. 7.1, *б*). По фронту некоторого синхроимпульса *C1* в момент t_1 входные конъюнкторы защелок регистра *MRGI* открываются, и триггеры изменяют состояния своих выходов. Выходные уровни *MRGI* начинают обрабатываться комбинационной макро-

схемой *МКСII*. В схеме существуют параллельные пути, и выходы ее сначала искажены гоночными процессами, показанными на рис. 7.1, *б* в виде сетки наложенных друг на друга фронтов. Для содержимого *MRGII* ложные состояния выходов *МКСII* не опасны, поскольку его входы заперты нулевым уровнем синхросигнала *С2*. По прошествии времени, равного максимально возможной задержке *МКСII* (задержке самого длинного ее тракта), к моменту t_5 все переходные процессы в *МКСII* наверняка затухнут и на ее выходах установятся стабильные уровни. Пусть задержка *МКСII* меньше, чем длительность фазового периода синхронизации $T_ф$, т. е. вся *МКСII* успокоится к моменту t_5, еще до поступления фронта очередного синхросигнала *С2*.

На фронте *С2* в момент t_2 установившиеся значения выходов *МКСII* принимаются в триггеры *MRGII*, и с этого момента можно начинать отсчет времени переходных процессов уже в *МКСI*. Пусть задержка схем *МКСI* такова, что они успокаиваются к моменту t_6, до поступления фронта очередного сигнала *С1*. Тогда по фронту *С1* в момент t_3 установившиеся, не искаженные гонками уровни *МКСI* будут приняты в *MRGI*. Одновременно в *MRGI* принимаются установившиеся уровни входных сигналов *Вх*, поскольку по условию изменяться они могут лишь по фронту *С2*.

Содержимое *MRGI* снова обрабатывается на *МКСII*, принимается в *MRGII* и т. д. В устройстве идет циклическая многоступенчатая обработка входных данных, каждый момент времени часть комбинационных схем работает, в них идут переходные процессы, а другая часть схем находится в покое, ждет своей очереди. Затем они меняются ролями. Основной результат такой организации: несмотря на любые гоночные процессы, протекающие в любых комбинационных схемах, информация в регистры будет приниматься всегда верная. Для этого требуется лишь, чтобы задержка всех КС, входящих в состав МКС, была как-то ограничена сверху, а это разработчик схем вполне может обеспечить, опираясь на паспортные значения максимальных задержек элементов.

Важный момент принципа двухфазной синхронизации при использовании прозрачных защелок: *синхросигналы С1 и С2 не должны взаимно перекрываться во времени*, т. е. конъюнкция их всегда должна быть равна 0. Если где-то произойдет перекрытие синхросигналов, то информация пройдет последовательно сразу сквозь несколько защелок разных фаз и синхронность устройства будет нарушена.

Типовая ошибка разработчика синхронных устройств — неправильное заведение обратных связей при схемной реализации алгоритмов, имеющих циклы. Во всех схемах с двухфазной синхронизацией петля обратной связи, как содержащая логические элементы, так и в виде просто проводника, начавшись на выходе триггера-защелки, синхронизируемого одной фазой, должна окончиться на входе другой защелки, синхронизируемой обязательно другой фазой. Связи, передающие сигнал с выхода одной защелки на вход другой, синхронизируемой той же фазой, недопустимы: они не обеспечивают поочередной работы триггеров и потактного продвижения информации, т. е. противоречат самому принципу двухфазной синхронизации. Фактически такие связи при $C=1$ просто включают последовательно две комбинационные схемы через соединяющую их защелку. Недопустимы и связи выход КС — вход той же КС, порождающие неуправляемые кольца из логических элементов, поведение которых рассматривалось в § 6.1. Если выход КС соединить с ее входом через одну защелку, то кольцо возникнет при $C=1$. Связи, недопустимые в системе двухфазной синхронизации при использовании триггеров-защелок, показаны на рис. 7.1, *а* штриховыми линиями.

7.2. Временные соотношения двухфазной синхронизации

На выбор временных характеристик синхросигналов — длительности фазового периода $T_ф$ и длительности синхроимпульса $T_и$—влияют три группы факторов, связанных с логическим проектированием: задержки комбинационных схем, тип используемых синхронных триггеров, схемные решения распределения синхросигналов по блокам устройства.

Разнообразие используемых комбинационных схем зависит от размера устройства и решаемых им задач. В качестве части КС могут использоваться готовые микросхемы с заданными значениями максимальных задержек, какую-то часть КС разработчик строит сам и, выбирая различные варианты схем, может влиять на задержку этих КС. Если задержка некоторых комбинационных узлов существенно меньше $T_ф$, то отрицательного влияния на правильность работы устройства это оказывать не будет, однако потенциальное быстродействие аппаратуры окажется недоиспользованным: схемы, выполнив свою работу, будут еще некоторое время бесполезно простаивать, ожидая очередного синхросигнала. Если же задержка некоторых комбинационных узлов будет больше какого-то порогового значения

которое назовем *рабочим интервалом* (пока будем полагать, что рабочий интервал равен $T_ф$, как на рис. 7.1, *б*), то устройство будет неработоспособным, поскольку триггеры, синхронизируемые следующей фазой, будут запоминать и передавать дальше еще неустановившиеся, ложные сигналы КС. Если такой слишком неповоротливый комбинационный узел набран из более простых логических фрагментов, включенных последовательно, то его можно рассечь пополам, как это показано на рис. 7.2, *а* и *б*. Промежуточный полуфабрикат функции, снятый с первой половины узла, запоминается на специально введенном вспомогательном регистре-защелке *RGY*, и обработка его продолжается в следующей

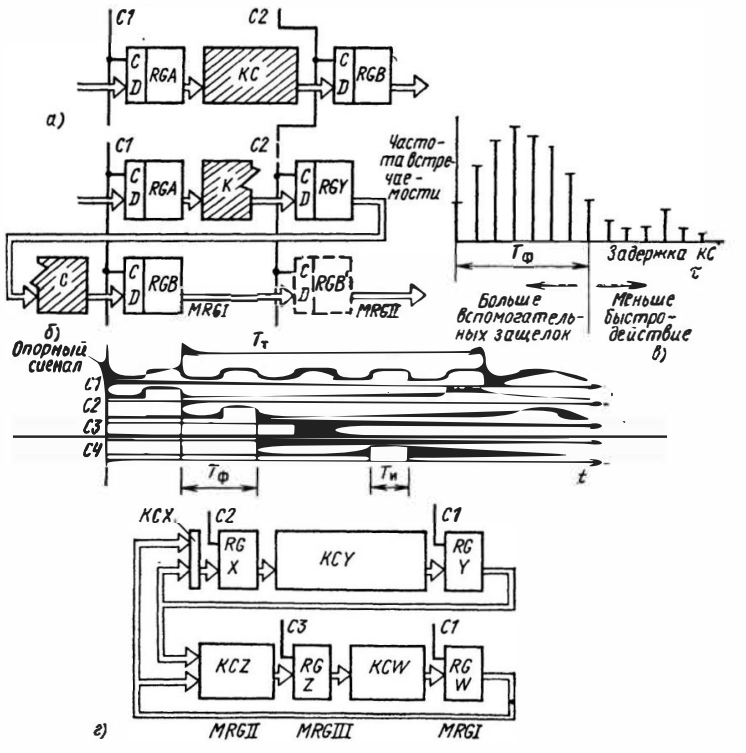


Рис. 7.2. Взаимосвязь задержки комбинационных схем и длительности фазового пернода синхронизации:

а — КС, имеющая слишком большую задержку; *б* — та же КС, разрезанная регистром *Y*; *в* — гистограмма задержек различных КС; *г* — диаграмма синхросигналов и примеры информационных связей при четырехфазной системе синхронизации

фазе периода синхронизации. Нужно только помнить, что это потребует перефазировки регистра-приемника *RGB* (на рис. 7.2, *а* и *б* — перевод его из *MRGII* в *MRGI*), а значит, и перефазировки некоторых последующих регистров. Если данные с выхода рассеченной КС желательно оставить в той же фазе, то можно ввести в схему еще один фазирующий регистр — *RGB'*, показанный на рис. 7.2, *б* штриховыми линиями. Если медленный функциональный узел неделим (например, это микросхема ПЗУ, задержки которых бывают довольно большими), то нужно или увеличить фазовый период $T_ф$ всей системы синхронизации, или сформировать специально для этого узла вспомогательную синхросерию с более длинным периодом, например пересчитав *С*-сигналы на счетчике.

На рис. 7.2, *в* показан типичный вид гистограммы встречаемости величин задержки различных КС цифрового устройства. Реальную гистограмму можно построить по результатам анализа схем устройств, аналогичных проектируемому. Если фактор быстродействия для проектируемого устройства достаточно важен, то рабочий интервал, в течение которого в схемах могут протекать переходные процессы, разумно выбрать таким, чтобы он оканчивался где-то в зоне крутого спада гистограммы. Если увеличивать длительность рабочего интервала, а с ним и $T_ф$, правее этой зоны, то за счет уменьшения тактовой частоты будет падать быстродействие устройства, а если длительность уменьшить, то будет резко увеличиваться число рассекаемых КС и соответственно число вспомогательных регистров.

Схемы небольших устройств часто удается спроектировать так, что все КС с большой задержкой размещаются в одной фазе периода двухфазной синхронизации, а все КС другой фазы имеют малую задержку. Тогда можно применить *асимметричную* систему синхронизации, в которой синхроимпульс *С2* размещен не в середине между соседними *С1*, а смещен ближе к одному из них. Это позволяет при тех же задержках КС уменьшить длительность тактового периода.

Еще лучше время тактового периода используется в *многофазных системах синхронизации*, получивших широкое распространение в быстродействующих устройствах. На рис. 7.2, *г* показаны временна́я диаграмма синхросигналов четырехфазной системы и фрагмент схемы устройства. Длина изображения КС на рисунке символизирует значение ее задержки. В зависимости от конкретной задержки каждой

КС на *С*-вход регистра, принимающего ее результат, можно заводить различные фазы синхронизации и открывать триггеры-приемники со сдвигом $1T_ф$, $2T_ф$, $3T_ф$ относительно той фазы, которая синхронизирует регистр-передатчик. Нельзя только синхронизировать принимающий регистр той же фазой, которой синхронизируется регистр на входе данной КС. Реально используемые четырех- и тем более шестифазные системы позволяют привести в соответствие задержку конкретной КС и время, отводимое системой синхронизации на ее работу, уменьшив, таким образом, непроизводительные простои КС в ожидании синхросигнала.

Тип триггеров, используемых в регистрах, существенно влияет на длину рабочего интервала, выделенного для протекания переходных процессов при заданной длительности $T_ф$. Система синхронизации, в которой в *MRGI* и *MRGII* используются *прозрачные защелки,* обладает весьма полезным свойством: *длительность переходных процессов в некоторых комбинационных узлах имеет право превышать фазовый период синхронизации* $T_ф$ даже в двухфазной системе. На рис. 7.3, *а* показано, как поступающее на вход *n*-разрядное слово, несущее некоторое смысловое содержание, последовательно, шаг за шагом обрабатывается на цепочке КС *M*, *N*, *P*, ..., передаваясь из одной КС в другую через соответствующие регистры. Максимальные значения задержек различных КС, как и реализуемые ими функции, в общем случае различны, но, разумеется, они известны разработчику. Синхросигналы заведены так, что *RGL* и *RGN* относятся к *MRGI*, *RGM* и *RGP* — к *MRGII*, *KCN* — к *MKCI*, *KCM* и *KCP*—к *MKCII*.

На фронте *C1* (рис. 7.3, *б*) в момент t_1 защелки *RGL* начинают переключаться, и спустя время их задержки по тракту *CQ* (см. рис. 6.8) в *KCM* начинаются переходные процессы. Пусть задержка *KCM* превышает длительность фазового периода $T_ф$ и переходный процесс в ней оканчивается лишь в момент t_8, на Δt позже поступления фронта синхросигнала *C2* (момента t_3). Вопреки распространенному убеждению к сбою это не приведет. В силу прозрачности защелок *RGM* они при активном уровне *C2* (t_3—t_4) будут передавать на вход *KCN* все изменения выходов *KCM*. После успокоения *KCM* (*момент* t_8), спустя еще время задержки защелок *RGM* по тракту $DQ(t_{зд.р}DQ)$, окончательные значения выходов *KCM* установятся на входах *KCN*. С этого момента можно начать отсчет времени переходных процессов в *KCN*. Поскольку *KCN* начала успокаиваться на Δt позже поступления фронта *C2,* то, чтобы к фронту очеред-

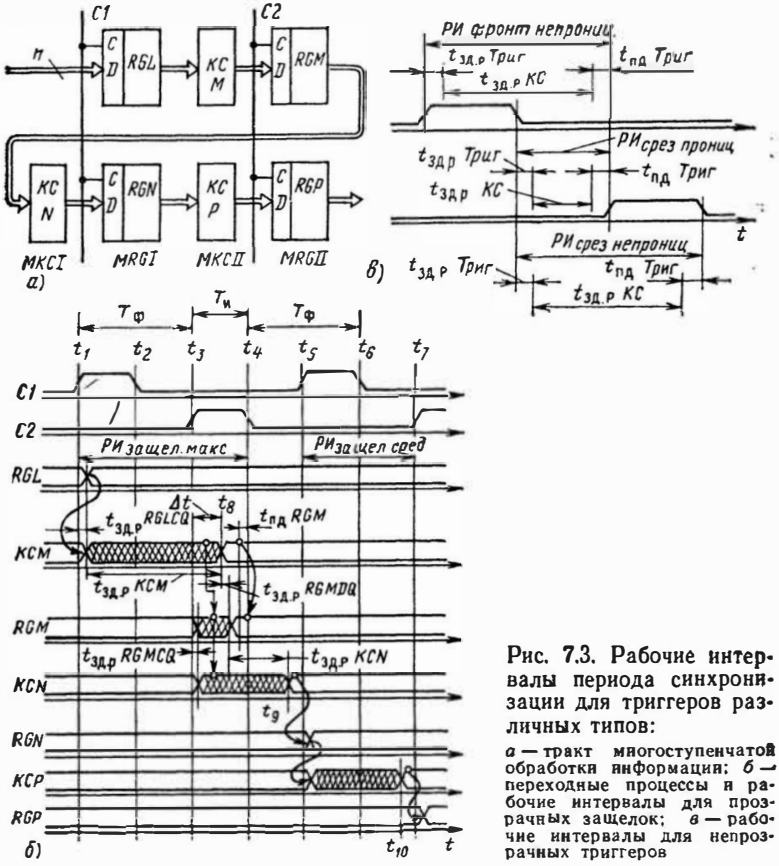


Рис. 7.3. Рабочие интервалы периода синхронизации для триггеров различных типов:

а — тракт многоступенчатой обработки информации; *б* — переходные процессы и рабочие интервалы для прозрачных защелок; *в* — рабочие интервалы для непрозрачных триггеров

ного *C1* (к моменту t_5) процесс вошел в типовое русло (рис. 7.1, *а*), задержка *KCN* должна быть соответственно на Δt меньше фазового периода $T_Ф$. Однако если *KCN* также настолько сложна, что ее не удается построить так, чтобы она окончила работу к моменту t_5, то ситуация наложения переходного процесса на *C*-сигнал перейдет в следующую фазу синхронизации, и теперь уже придется сделать короче следующую КС — *KCP*, работающую на интервале t_5—t_{10}.

В описываемой ситуации сбой произойдет в том случае, если переходный процесс *KCM* окажется еще длиннее и не успеет закончиться к моменту t_4 фиксации защелок *RGM* (точнее — чуть раньше: на интервал подготовки $t_{пд}$ тригге-

ров *RGM*). Для этого задержка *КСМ* должна превысить фазовый период $T_ф$ почти на значение длительности импульса $T_и$ серии *С2*. Сбой произойдет и тогда, когда переходный процесс, перелившийся из *КСМ* в *КСN*, не успеет закончиться к моменту t_6 фиксации защелок *RGN*, и т. д.

Таким образом, если переходный процесс в некоторых функциональных узлах в одной из фаз оказался длиннее $T_ф$ на Δt, то последующие функциональные узлы, продолжающие перерабатывать эту информацию, должны иметь задержку меньше $T_ф$, так чтобы за несколько следующих фаз (или всего за одну, как на рис. 7.3, *б*) выбег задержки Δt был погашен и переходные процессы в КС снова стали оканчиваться к моменту поступления фронта очередного *С*-сигнала. Таким образом, рабочий интервал *РИ*, в течение которого в схемах, относимых к данной фазе, допустимы переходные процессы, в случае двухфазной синхронизации и прозрачных защелок характеризуется двумя значениями: максимальным, $РИ_{защел.макс}$, равным сумме фазового периода $T_ф$ и длительности импульса $T_и$ (рис. 7.3, *б*), и средним $РИ_{защел.сред}$, равным $T_ф$. Если задержку какой-то КС пришлось сделать большей, чем $T_ф$, то в течение нескольких фазовых периодов после этого средняя задержка в данном информационном тракте должна быть меньше $T_ф$.

Чтобы при необходимости разработчик мог сделать выбег Δt возможно бóльшим, длительность импульса $T_и$ нужно выбирать возможно ближе к ее верхнему допустимому пределу, который ограничен опасностью наложения *С1* и *С2*.

Возможность иметь задержку КС, заметно превышающую фазовый период, — удобное для разработчика свойство системы синхронизации. Свойство это известно не широко, и в большинстве разработок оно лишь деликатно подправляет ошибки схемотехников и технологов, когда реальная задержка какой-то КС оказывается больше расчетной. Однако квалифицированные разработчики могут использовать это свойство вполне осознанно. В многофазных системах синхронизации при той же длительности такта $T_т$ длительность импульса $T_и$ существенно меньше, чем в двухфазных, соответственно меньше и эффект удлинения максимально возможного рабочего интервала.

Вместо защелок в системе двухфазной синхронизации можно использовать и непрозрачные триггеры. Если триггер непроницаем (триггер, переключаемый перепадом) и переключается по положительному фронту *С*-сигнала (например, К155ТМ2), то КС будет начинать работать по фронту

C-сигнала одной фазы (точнее — спустя время задержки $t_{зд.р}$ триггера) и должна будет обязательно успокоиться к моменту поступления фронта C-сигнала другой фазы (точнее — чуть раньше: на время подготовки $t_{пд}$ триггера). Положение рабочего интервала такого триггера показано в верхней части рис. 7.3, *в* ($РИ_{фронт\ непроницаем}$). Рабочий интервал непроницаемого триггера строго равен $T_ф$ в отличие от рабочего интервала защелки, который в некоторых случаях можно сделать заметно длиннее.

Если непроницаемый триггер переключается срезом, то его рабочий интервал, оставаясь также равным $T_ф$, смещается по циклу синхронизации, что также показано на рис. 7.3, *в* ($РИ_{срез\ непроницаем}$). Если же используется проницаемый триггер (со свойствами проскока или захвата, например К155ТВ1), то интервал, в течение которого могут идти переходные процессы в КС, сокращается со значения $T_ф$ до размера промежутка между C-сигналами соседних фаз, т. е. до значения разности $T_ф - T_и$ ($РИ_{срез\ проницаем}$ на рис. 7.3, *в*).

Таким образом, в системе двухфазной синхронизации (и многофазной — тоже) при одинаковой длительности фазового периода $T_ф$ непрозрачные триггеры по сравнению с прозрачными защелками оказываются не только неоправданно громоздкими, но и отводят в общем меньше времени на переходные процессы в КС.

Кроме того, многие типы непрозрачных триггеров из-за возможности гонок по входу чувствительны к завалам фронтов C-сигнала (см. § 6.3 и 6.4), что может потребовать ревизии электрической схемы разводки синхросигналов. В распространенной схеме защелки по рис. 6.6, *б* никаких опасных гонок по входу не возникает, поэтому она не предъявляет особых требований к длительности фронтов C-сигналов. Это еще одно достоинство системы двухфазной синхронизации с использованием триггеров-защелок.

Применение непрозрачных триггеров при двухфазной синхронизации оправдано, если они уже входят в состав микросхем, которые рационально использовать благодаря их каким-то другим полезным свойствам. В этих случаях нужно учитывать, что положения рабочих интервалов защелки и непрозрачных триггеров в общем случае не совпадают, что накладывает дополнительные ограничения на длительности переходных процессов тех КС, которые включены на стыке защелок и непрозрачных триггеров, а также на стыке непрозрачных триггеров с различными положениями рабочих интервалов. Положение рабочего интервала однозначно

определяется параметрами синхронизации триггера, которые рассмотрены в § 6.7 и для некоторых триггеров приведены в табл. 6.6. Когда известны типы соседних триггеров, допустимые значения задержек пограничных КС легко получить из диаграммы рис. 7.3, *б* и *в*. Как видно из диаграмм, с защелками К155ТМ7 наиболее естественно стыкуются непрозрачные триггеры, переключаемые положительным фронтом, например К155ТМ2.

Временны́е параметры триггеров, существенные при их использовании в двухфазных или многофазных системах синхронизации, следующие:

1. $t_{зд.р}$ по тракту вход C — выходы.
2. $t_{зд.р}$ по тракту вход D — выходы при $C=1$ — только для защелок.
3. $t_{пд}$ по управляющим входам.
4. Минимально допустимая длительность C-сигнала.
5. Максимально допустимая длительность фронтов C-сигнала — только для тех триггеров, в которых возможны гонки по C-входу. Если схема триггера такова, что эти гонки невозможны, то данный параметр неактуален.

Величина $t_{вд}$ триггера при использовании его в двухфазной или многофазной системе синхронизации значения не имеет, поскольку входные сигналы триггера непосредственно после фиксирующего среза не изменяются. Их изменения могут начаться лишь при поступлении переднего фронта C-сигнала другой фазы.

Схемы распределения синхросигналов приходится строить в связи с очень большим числом узлов-потребителей этих сигналов в цифровом устройстве. Разводить синхросигналы от единого мощного генератора не следует, ибо мощные цепи порождают столь же мощные помехи. Поэтому систему синхронизации строят в виде многоярусного дерева из обычных элементов, которое размножает сигналы маломощного задающего генератора. При этом ярусы дерева часто совмещают с ярусами конструктивного деления устройства на платы (ТЭЗы, карты), рамы, шкафы и т. п.

Задержки в цепях каждого яруса имеют разброс и нестабильность, в результате временны́е соотношения синхросигналов на выходе задающего генератора и на C-входе узла-потребителя выглядят, как показано на рис. 7.4. Фронты синхросигнала генератора задерживаются на некоторую постоянную $\tau_к$, в основном определяемую достаточно стабильными задержками в кабелях, и нестабильную $\tau_{ус}$, определяемую задержками в усилителях дерева размножения.

В общем случае синхросигналы у различных потребителей не будут совпадать во времени, а будут *расфазированы*. Стабильный компонент задержки $\tau_к$ поддается компенсации. Для этого задержки синхросигналов всех ветвей дерева увеличивают до наибольшей из них, добавляя по необходимости мерные отрезки кабеля или другие элементы задержки. Нестабильный компонент контролю не поддается и мо-

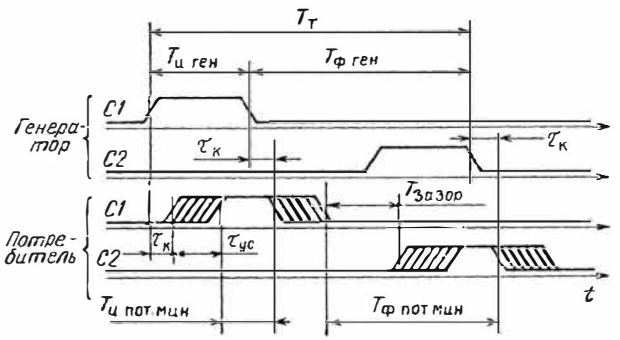


Рис. 7.4. Временные соотношения в системе распределения синхроимпульсов

жет вызывать в различных узлах-потребителях сужение, расширение и взаимное смещение импульсов. Для обеспечения у потребителя гарантированного минимума ширины импульса $T_{и.пот}$ ширина импульса задающего генератора должна быть на $\tau_{ус}$ больше. Максимально допустимая ширина импульсов генератора определяется тем, что после возможного их расширения на $\tau_{ус}$ синхроимпульсы различных фаз у потребителей не должны взаимно перекрываться для любой пары потребителей, обменивающихся информацией. При неудачном сочетании задержек срезов синхросигналов фазовый период у потребителя $T_{ф.пот}$ уменьшится на $\tau_{ус}$. Для восстановления его потребуется увеличить фазовый период генератора, что приведет к потере быстродействия устройства. Если это нежелательно, то $\tau_{ус}$ уменьшают, выбирая для дерева разводки синхросигналов более быстродействующие элементы с меньшим временем неопределенности. Ряд вопросов построения схем распределения синхросигналов освещен в [13, 38, 39].

Кроме влияния рассмотренных трех факторов, связанных с логической схемой устройства (задержек КС, типа триггеров и схемы разводки C-сигналов), на выбор тактовой

частоты влияет также ряд чисто электрических параметров, например частотные характеристики цепей питания и общего провода, определяемые конструктивным выполнением устройства. Влияют на выбор тактовой частоты и организационные факторы: требования совместимости с другой аппаратурой, соображения унификации и т. д. Для аппаратуры, построенной на ТТЛ-микросхемах, в конечном итоге тактовая частота лежит обычно в пределах 0,5—5 МГц, для устройств промышленной автоматики она обычно ниже: 0,5—1 МГц. Примерно с такой же тактовой частотой могут работать микросхемы быстродействующих МДП-серий, а для распространенных КМДП-серий К561 и 564 тактовую частоту выбирают примерно на порядок меньше. Традиции на частоты синхронизации матричных БИС пока еще не установились.

7.3. Однофазная синхронизация

При двухфазной синхронизации отсутствие в схемах замкнутых колец из логических элементов обеспечивается тем, что в любой момент времени хотя бы один из макрорегистров-защелок — *MRGI* или *MRGII* (см. рис. 7.1) не пропускает сигналы со входа *D* на выход. Отсюда следует, что применение непрозрачных триггеров позволит обойтись одним регистром в кольце циклической схемы, т. е. синхронизация в этом случае может быть *однофазной*. На рис. 7.5 показаны схема информационных связей цифрового устройства с однофазной синхронизацией и временна́я диаграмма его работы. Как и на рис. 7.1, все комбинационные схемы устройства, получающие данные с выходов триггеров и передающие результаты на *D*-входы триггеров, рассматриваются совместно как большая комбинационная макросхема *МКС*, в данном случае — всего одна. Все триггеры синхронизируются одним и тем же синхросигналом и рассматриваются как один большой макрорегистр *MRG*. На некоторые триггеры этого регистра поступают внешние сигналы, с некоторых триггеров снимаются выходные сигналы. Пусть для определенности все триггеры непроницаемы для помех при $C=1$ и меняют состояние выхода по срезу *C*-сигнала, например это триггеры, построенные по схеме рис. 6.12.

Как следует из временно́й диаграммы (рис. 7.5, *б*), по срезу *C*-сигнала в момент t_1 выходы всех триггеров переключаются в те состояния, которые были на их входах непосредственно перед поступлением среза *C*-сигнала, т. е.

в состояния, выработанные к этому моменту логическими схемами *МКС*. После среза синхроимпульса информация с выхода *МКС* оказывается переданной на ее вход, и в *МКС* начинаются переходные процессы очередного цикла. В грамотно спроектированном устройстве все переходные процессы должны закончиться к срезу следующего синхроимпульса, а точнее, к моменту t_5 (началу интервала подготовки

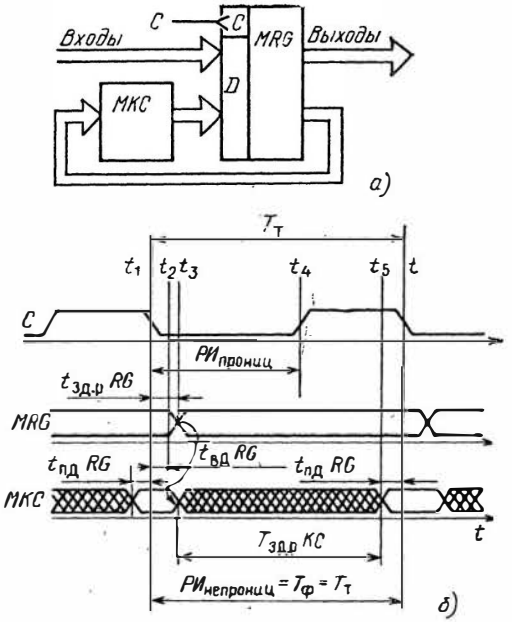


Рис. 7.5. Схема *(а)* и временна́я диаграмма *(б)* системы однофазной синхронизации

$t_{пд}$). Установившиеся, не искаженные гонками состояния выходов *МКС* снова будут переданы на вход той же *МКС* и т. д.

Как видно из рис. 7.5, *б*, рабочий интервал при однофазной синхронизации равен полному периоду синхросерии $T_т$, но только если триггеры не имеют свойства проницаемости. В противном случае переходные процессы в *МКС* должны окончиться до поступления фронта *С*-сигнала, т. е. к моменту t_4. Интервал t_4—t_5 при этом для дела пропадает. Если схема построена на базе триггеров, переключаемых положительным фронтом, то рабочий интервал будет занимать

положение между фронтами C-сигналов. При использовании в одном устройстве триггеров с различными положениями рабочих интервалов максимально допустимая задержка согласующих их КС будет отличаться от номинальной — так же, как и в случае двухфазной синхронизации.

Тонким моментом, заметно влияющим на надежность работы однофазных схем, является соотношение задержки распространения $t_{зд.р}$ и времени выдержки $t_{вд}$ триггеров регистра. Если задержка комбинационной схемы мала, особенно если КС представляет собой просто провод, к тому же короткий, то новое состояние выхода быстро переключившегося триггера может поступить на вход другого триггера слишком рано для последнего, еще в течение его интервала выдержки, когда уровни его управляющих входов изменять еще нельзя. Для гарантии отсутствия сбоев такого типа требуется, чтобы максимальное значение $t_{вд}$ любого триггера не превышало минимального значения $t_{зд.р}$ также любого триггера, т. е. момент времени t_2 на рис. 7.5, *б* должен обязательно предшествовать моменту t_3. В этом смысле вне конкуренции оказываются схемы триггеров, у которых $t_{вд}=0$. В двухступенчатом триггере и триггере, использующем задержку, это свойство присуще самой схеме, а в шестиэлементном оно может быть достигнуто за счет определенного соотношения задержек элементов триггера (см. § 6.4). При сомнительном соотношении $t_{зд.р}$ и $t_{вд}$ приходится ограничивать минимальное значение задержки КС, например исключая прямые связи триггер — триггер за счет введения между ними одного-двух холостых логических элементов. Такие ограничения встречаются, например, в инструкциях по проектированию схем на некоторых матричных БИС. Требования к остальным временным параметрам триггеров при использовании их в системе однофазной синхронизации такие же, что и в двухфазной.

Однофазная система синхронизации в отличие от двухфазной болезненно чувствительна к расхождению активных фронтов синхроимпульсов, возникающему в системе распределения синхросигналов. На рис. 7.6 показаны две последовательности синхроимпульсов, C_A и C_B, имеющие взаимный сдвиг (расфазировку) за счет различных задержек в ветвях дерева разводки, и выходы Q_A и Q_B двух триггеров, синхронизируемых соответствующими последовательностями. Если при показанной на рисунке расфазировке передача информации идет с выхода триггера B на D-вход триггера A (связь не показана, она подразумевается), то оши-

бок не возникает: та информация, которая поступила на вход T_B в $(i-1)$-й такт, поступает с выхода T_B на D-вход T_A уже после исчезновения синхросигнала C_A в момент t_2, т. е. в i-й такт, как это и должно быть. После среза очередного C_A в $(i+1)$-й такт эта информация появится на выходе T_A. Если же передача идет с выхода T_A на вход T_B, то

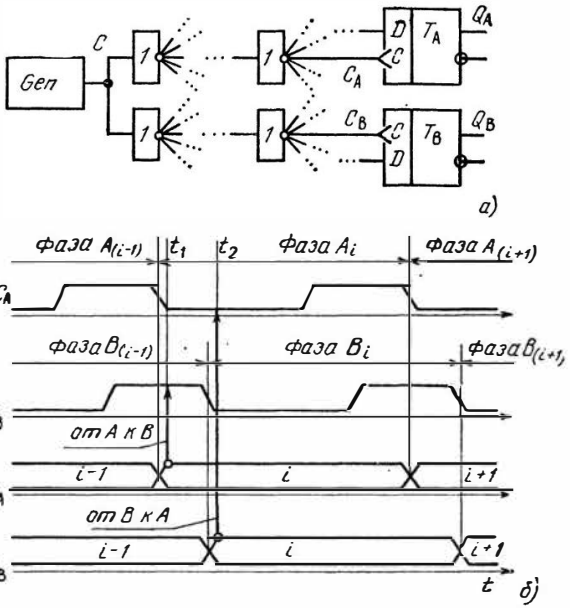


Рис. 7.6. Схема (*а*) и временна́я диаграмма (*б*), иллюстрирующие механизм появления сбоев при расфазировке синхросигналов однотактной синхронизации

информация с выхода T_A, предназначенная, как и в прошлом случае, для передачи на вход T_B в i-м такте, появится на входе T_B в момент t_1, еще до переключающего среза синхросигнала C_B. Тогда открытая при $C=1$ первая ступень триггера T_B взамен информации, выработанной схемой в $(i-1)$-м такте и уже загруженной в эту ступень, воспримет новую, по замыслу относящуюся к i-му такту. По срезу синхросигнала C_B эта информация появится на выходе T_B как входная для КС в i-м такте. Это уже ошибка: информация, выработанная в i-м такте и предназначенная для обработки в $(i+1)$-м, обрабатывается в i-м такте, т. е., по

сути, из своего такта проникает в чужой, более ранний. Ситуация полностью аналогична расхождению в датировке астрономического явления, которое может наблюдаться на большой территории и произошло около полуночи по уральскому времени: московский и новосибирский наблюдатели датируют это событие различными числами.

Таким образом, *однофазная синхронизация не терпит, чтобы информация обгоняла синхросигнал.* В основном в этом кроется причина ограниченного применения однофазной синхронизации. В противоположность ей двухфазная синхронизация легко переносит любые задержки в трактах разводки *С*-сигналов, для этого достаточно лишь в нужной мере увеличить длительность тактового периода задающего генератора. Регистры при двухфазной синхронизации строят на основе самого простого из синхронных триггеров, но при необходимости могут использоваться триггеры почти всех типов. Никаких особых требований к значениям или соотношениям их времен задержки, выдержки, подготовки не предъявляется. При использовании прозрачных защелок по схеме рис. 6.6, *б* не предъявляется особых требований и к крутизне фронтов синхросигналов. Этими же свойствами обладают и многофазные системы, которые позволяют лучше использовать комбинационные схемы по скорости и тем повысить общее быстродействие устройства. Однако это усложняет схему разводки синхросигналов и процесс проектирования временно́й диаграммы устройства из-за усложнения задач фазировки процессов в различных КС.

Поэтому, несмотря на то что в однофазных системах разводить нужно всего одну синхросерию, и с разработчика, составляющего временну́ю диаграмму, сняты все заботы о фазировании различных узлов, однофазную синхронизацию широко применяют при построении лишь небольших отдельных узлов — счетчиков, сдвиговых регистров и т. п. Такие однофазные вкрапления в большое устройство, в целом имеющие двухфазную или многофазную синхронизацию, удобны тем, что позволяют за один фазовый период, а иногда лишь за время активного уровня *С*-сигнала выполнить какое-то действие, связанное с заменой содержимого некоторого регистра (сдвиг в регистре, прибавление единицы к счетчику и т. п.), на которое при классической двухфазной синхронизации требуются два фазовых периода. При введении таких однофазных вкраплений нужно помнить о возможности несовпадения их рабочих интервалов с рабочими интервалами защелок и о связанной с этим необхо-

димости уменьшать (или возможности увеличивать) задержки пограничных КС (см. рис. 7.3). Чисто однофазную синхронизацию используют в некоторых микропроцессорных наборах, например в К589, и в других небольших устройствах, компактно сконструированных и не требующих многоступенчатого размножения синхросигналов. В большинстве устройств и даже микроЭВМ используется двухфазная и многофазная синхронизация.

Расфазировку синхросигналов порождают не только физические задержки в связях, но и некоторые схемные решения, связанные с управлением *D*-триггерами. Последние заново принимают состояние своего *D*-входа в каждом такте синхронизации. Поэтому, если в *D*-триггере необходимо сохранить поступившую в некотором такте информацию в течение нескольких последующих тактов, синхросигналы от *C*-входа триггера нужно на это время отключить с помощью конъюнктора, заперев его неактивным уровнем разрешающего сигнала *V*. В *DV*-триггерах (рис. 7.7, *г* и *д*) этот конъюнктор входит в состав самого триггера, но если использовать такой триггер почему-либо нет возможности, то конъюнктор в тракт *C*-сигнала приходится включать в виде самостоятельного элемента. При этом иногда для экономии оборудования КС бывает удобно ввести этот прерывающий *C*-сигналы конъюнктор куда-то в начало КС, как условно показано на рис. 7.7, *а*. Такое решение рискованно, поскольку за счет задержки в КС синхросигнал этого триггера сдвигается относительно синхросигналов остальных триггеров (рис. 7.7, *б*), что приводит к обычным последствиям расфазировки. Поэтому, даже в ущерб экономичности схемы,

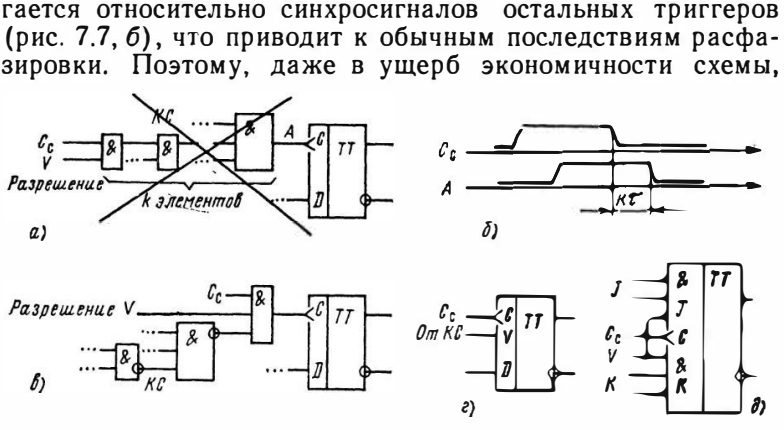


Рис. 7.7. Выполнение логических операций над синхросигналом C_c:
а — плохое решение; *б* — временна́я диаграмма плохого решения; *в* — удовлетворительное решение; *г* — *DV*-триггер; *д* — *JK*-триггер в роли *JKV*-триггера

конъюнктор, управляющий *C*-сигналом *D*-триггера, должен подключаться непосредственно к *C*-входу триггера (рис. 7.7, *в*).

Двухфазная синхронизация устойчива к расфазировке синхросигналов и в принципе допускает решения по рис. 7.7, *а*. Однако задержка в срабатывании триггера выбивает его из четкой шеренги переключения всех остальных триггеров *MRG* и требует соответствующего укорочения трактов КС, которыми данный триггер управляет, что порождает дополнительные ограничения и необходимость увязок между разработчиками различных КС. Поэтому при двухфазной синхронизации также предпочтительнее использовать решения по рис. 7.7, *в* или *DV*-триггеры, если они есть.

При некотором усложнении и однофазную систему синхронизации можно сделать устойчивой к расфазировке. Для этого используется вспомогательный *выравнивающий регистр,* включаемый перед входом приемника, как показано на рис. 7.8, *а*.

Если передаваемая информация обновляется по срезу *C*-сигнала, то в качестве выравнивающего регистра *RG1* используется регистр, переключаемый положительным перепадом, который выставляет на выходе информацию по фронту *C*-сигнала линии связи, т. е. информацию, которая с гарантией относится именно к данному такту. Если в линии новая информация обгонит срез синхросигнала, т. е. новые данные появятся, когда еще $C=1$ (как на рис. 7.6 при передаче от T_A к T_B), то благодаря непрозрачности и непроницаемости выравнивающего регистра на его выход эти спешащие данные не попадут. *RG1* можно построить и на прозрачных защелках, имеющих низкий активный уровень, но при этом нужно быть уверенным, что регистр приемника *RG2* обладает нулевым временем выдержки.

Чтобы расфазировку можно было допустить в любых цепях однофазного устройства, описанным способом нужно модифицировать все его триггеры. Возможные схемы таких усложненных триггеров показаны на рис. 7.8, *б*, *в* и *г* (обведены штриховой линией). Триггеры имеют структуру шлюзовых камер с затворами на входе и выходе. Первый затвор *триггера-шлюза* срабатывает по фронту синхросигнала, запирая вход *D* и фиксируя в первом триггере *T1*, т. е. в первой ступени всего составного триггера-шлюза, уровень *D*, предшествовавший фронту синхросигнала. Второй затвор шлюза (триггер *T2*) срабатывает по срезу синхросигнала, передавая на выход содержимое первой ступени. Триггер *T1* должен быть непроницаем для помех.

В схеме по рис. 7.8, *б* (временная диаграмма — на рис. 7.8, *е*) время выдержки $t_{вд}$ защелки *T2* на фронте *C*-сигнала должно быть меньше задержки $t_{зд.р}$ непрозрачного триггера *T1* по такту *CQ*. Можно

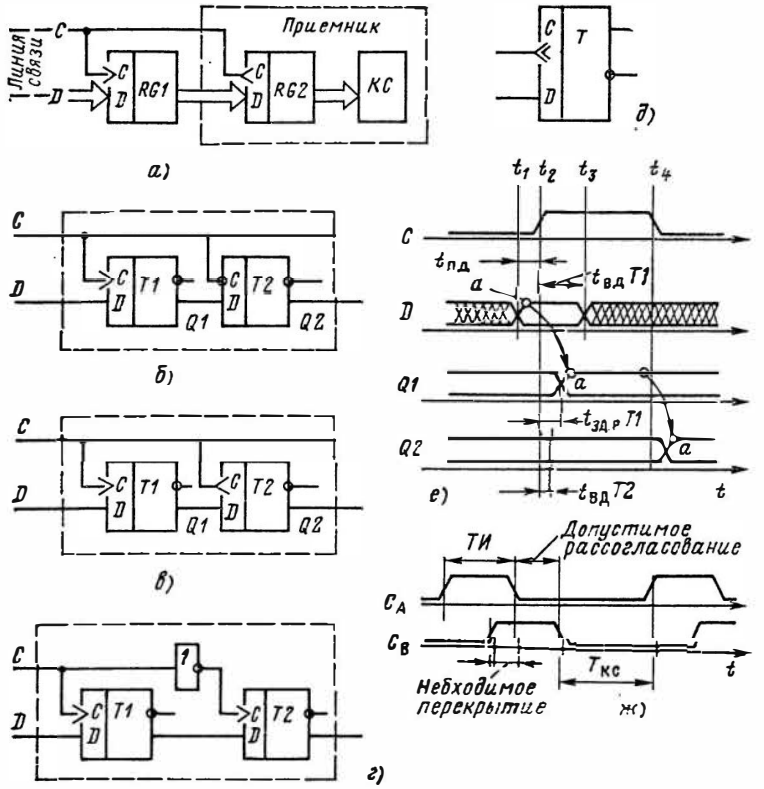


Рис. 7.8. К допущению расфазировки в однофазных системах синхронизации:

а — применение фазирующего регистра «по месту»; *б, в, г* — схемы триггеров, допускающих расфазировку; *д* — условное обозначение такого триггера; *е* — временна́я диаграмма работы триггеров; *ж* — допустимое значение расфазировки и время, остающееся на работу КС

модифицировать схему, поменяв местами непрозрачный триггер и защелку. Тогда у непрозрачного триггера активным должен быть отрицательный перепад синхросигнала, а его время выдержки должно быть меньше задержки распространения защелки по такту *CQ*. Для иллюстрации сказанного достаточно построить временну́ю диаграмму модифицированного триггера. Схемы триггеров-шлюзов по рис. 7.8, *в* и *г* никаких специальных требований к временны́м параметрам триггеров, из которых они собраны, не предъявляют, но зато оба триггера должны быть непрозрачными, а первый из них к тому же должен быть непроницаемым.

На рис. 7.8, *д* показано используемое иногда обозначение такого триггера-шлюза. Рисунок 7.8, *ж* иллюстрирует допустимую расфазировку синхросигналов между любыми ветвями дерева разводки, при этом обозначения C_A и C_B имеют тот же смысл, что и на рис. 7.6. Минимальное значение зоны перекрытия синхросигналов определяется временными характеристиками ступеней шлюзового триггера. Платить за допустимость расфазировки приходится не только усложнением триггеров, но и потерей скорости, поскольку, как видно из рис. 7.8, *ж*, время на работу КС по сравнению со строго фазированной системой сокращается почти на двойное значение максимальной длительности синхроимпульса $T_и$: в худшем случае КС сможет начать работу на срезе самого позднего C_B, а окончить работу должна к фронту самого раннего C_A. Очевидно, что $T_и$ в такой схеме не следует увеличивать сверх необходимого минимума.

7.4. О проектировании схем с обратными связями

Эта книга посвящена вопросам разработки схем цифровых устройств умеренной сложности и не затрагивает проблем схемотехники сверхБИС. Поэтому, как уже отмечалось в § 5.2, в ней не рассматриваются вопросы построения самосинхронизирующихся схем, о чем см. [34]. Разработчиков же схем не сверхбольшого размера автор ориентирует на грамотное применение *синхронного* принципа работы, который долго еще будет вне конкуренции для устройств цифровой автоматики объемом по крайней мере до тысяч или десятков тысяч логических элементов.

Как следует из рассмотрения переходных процессов в синхронных схемах — и однофазных, и многофазных, для бессбойной работы цифрового устройства целый ряд временных параметров триггеров и синхросигналов должен находиться во вполне определенных, иногда довольно тонких отношениях. При использовании триггеров, построенных по хорошо изученным разработчиком схемам, для которых известны степень их прозрачности и проницаемости, а также величины $t_{зд.р}$, $t_{пд}$, $t_{вд}$, $t_{фронта\ макс}$, разработчик в состоянии уверенно выдержать все соотношения, требующиеся для того, чтобы гоночные процессы никак себя не проявляли.

Совершенно иная ситуация возникает, если разработчик отступает от канонической структуры синхронного устройства RG — КС — RG — КС... и начинает на полуинтуитивном уровне вводить различные обратные связи, запрещаемые правилами построения синхронных схем. Эти связи иногда делают несколько понятнее логику работы схемы, иногда они

позволяют получить небольшую экономию оборудования или времени, что и служит побудительным мотивом к нарушению канонов. Однако при этом фактически создаются какие-то схемы с асинхронными обратными связями наподобие триггеров и еще более сложных автоматов с неисследованными, не известными разработчику временами подготовки, выдержки, задержки, с возможностью проявления захвата 1 или 0, проскока фронтов, гонок по входу и других, более сложных (в соответствии со сложностью схем) неприятностей, которые будут вступать между собой в какие-то неконтролируемые отношения. Полезно вспомнить, сколь неочевидны с первого взгляда и тем не менее сколь опасны могут быть проявления этих отношений, которые к тому же склонны вредить лишь эпизодически, только при некоторых сочетаниях отклонений от номиналов, при некоторой передаваемой информации... Многие аспекты поведения триггерных схем на самом деле существенно сложнее, чем это представляется при первом с ними знакомстве. Поэтому, и особенно для не очень опытного разработчика, нельзя считать безопасным введение в состав устройства нового узла с нетактируемыми обратными связями, узла, свойства которого не исследованы внимательно и всесторонне. Надеяться на автоматическое обнаружение в любой схеме любых проявлений гоночных процессов с помощью имитационного моделирования сегодня неправомерно. Модели недостаточно хорошо учитывают тонкости переходных процессов, возникающих в схемах с обратными связями, и не дают абсолютно надежных результатов. Небольшие узлы с нетактируемыми обратными связями, соответствующие по сложности распространенным непрозрачным триггерам, т. е. еще обозримые для человека, на сегодня надежнее исследовать «вручную», поскольку при этом удается учесть больше факторов, чем заложено в моделирующей программе. Если же схема узла с нетактируемыми обратными связями настолько сложна, что понять и исследовать все аспекты ее поведения при возможных вариациях задержек и порогов элементов не удается, то применять схему в таком виде не следует. Обратные связи ее нужно разорвать и пропустить через синхронные триггеры, управляемые *C*-сигналами, смириться с возникшей в связи с этим потерей быстродействия и компенсировать потерю другими методами, если это требуется.

Для минимизации порождаемых гонками неприятностей можно рекомендовать в процессе логического проектирования придерживаться следующих легко выполнимых правил:

1. *Триггеры для проектируемой схемы брать лишь из ограниченного предварительно составленного списка.* Перед занесением в список триггер внимательно проверяют на то, чтобы он по своим временны́м параметрам (соотношение времен подготовки, задержки, выдержки, отсутствие гонок по входу) и параметрам синхронизации (знак переключающего перепада, степень прозрачности и проницаемости, активный уровень *С*-сигнала) был совместим как с другими триггерами списка, так и с заданной системой синхронизации.

2. *Заводить в схеме только разрешенные* (показаны на рис. 7.1 и 7.5) *обратные связи* и никогда не заводить неразрешенных, т. е. не создавать новых, не исследованных триггеров или других схем с нетактируемыми обратными связями (асинхронных автоматов).

Для обеспечения легкой читаемости, обозримости всей структуры синхронизации нужно четко отобразить на схеме, какие триггеры какой фазой синхронизируются, т. е. к какому макрорегистру относятся. Если есть возможность, то полезно начертить схему (можно по частям) в форме, близкой к показанной на рис. 7.1, *а*, собрав все триггеры каждой фазы в вертикально расположенные группы. Или можно по форме рис. 7.1, *а* построить упрощенную вспомогательную карту схемы, на которой изображены и промаркированы только триггеры и регистры. Если в схеме используются триггеры с несовмещенными рабочими интервалами, например если применяются готовые счетчики или регистры, то каждый триггер (регистр) на карте снабжается сведениями о расположении и длительности его рабочего интервала. Полезны значения таких свойств, как прозрачность, проницаемость, непроницаемость. Все эти данные позволяют оперативно определять допустимые задержки КС, включенных между триггерами или регистрами.

Выполнение двух сформулированных выше правил не порождает каких-либо принципиальных проблем, и их можно рассматривать просто как требования грамотного логического проектирования. Если эти требования соблюдены, то все нерассекаемые синхросигналами обратные связи оказываются локализованными только внутри типовых триггеров с хорошо изученными свойствами. Обеспечить работоспособность такой схемы для любых реально встречающихся вариаций задержек и порогов логических элементов уже несложно: нужно лишь обеспечить, чтобы задержки всех КС не превышали некоторых легко определимых значений. По-

этому при программном моделировании тактируемой схемы на предмет выявления возможных сбоев из-за гонок достаточно проверить выполнение тех же самых неравенств. Поскольку наиболее критичными оказываются максимальные значения задержек, моделирование можно вести, не используя зон неопределенности, а непосредственно на языке максимальных задержек, что намного проще. Если положения рабочих интервалов всех применяемых типов триггеров совпадают и если запрещено перетекание переходных процессов через фронты C-сигналов защелок, то моделирующие программы становятся совсем простыми.

Концепция, согласно которой нетактируемые обратные связи не следует выпускать за пределы небольших хорошо изученных узлов (типовых триггеров), на сегодня еще не является общепризнанной. В противовес ей продолжает жить зародившаяся еще на заре теории автоматов концепция допустимости любых обратных связей, лишь бы они не порождали опасных состязаний. Такой подход совершенно однозначно влечет за собой сразу целый букет проблем: выявление самого факта состязаний, их классификацию (в [33] сформулированы строгие определения для 14 (!) видов состязаний), необходимость отличать опасные состязания от неопасных и т. п. Проблемы в силу своей сложности оказываются исключительно привлекательными для пытливых умов, и число работ здесь уже достигло астрономических цифр, продолжая стабильно расти. Для выявления состязаний в схемах, построенных согласно этой «автоматной» концепции, создаются все более сложные моделирующие программы, в которых состояния логического элемента описываются в троичном и пятеричном алфавите и перебирается множество возможных сочетаний задержек у различных элементов (см. [32]). Эти программы работают очень медленно, а достоверность их результатов пока еще недопустимо далека от единицы.

Причина скромных успехов кроется в том, что задача выявления опасных состязаний носит переборный характер и алгоритмов эффективного ее решения не существует. Единственное кардинальное решение всей проблемы — это не выпускать объем перебора за некоторые рамки, внутри которых задача еще обозрима и с перебором справляться не сложно. Именно это и достигается тактированием и использованием типовых не слишком сложных триггеров. Однако, несмотря на, вроде бы, очевидность ситуации, единого общепризнанного взгляда на эту проблему еще не сформиро-

валось. Требование замыкать все неконтактируемые обратные связи только в рамках типовых триггеров находится пока еще на уровне одной из возможных парадигм, которую каждый разработчик волен либо принять для себя, либо отвергнуть.

Контролировать работоспособность схем с тактируемыми обратными связями намного проще, чем схем, переплетенных несинхронизируемыми обратными связями. Используя метод сквозного сдвигового регистра (см. § 10.1), в схемах, построенных на унифицированных триггерах и не имеющих каких-либо еще асинхронных связей, работу комбинационной части и триггерных регистров можно проверять по отдельности, что сильно сокращает длину тестовых последовательностей.

Как и всякая разумная стандартизация, ограничение произвола в способах заведения обратных связей дает большую экономию на всех последующих этапах разработки и эксплуатации логических схем, обрезая наиболее пышную часть кроны дерева вариантов. Программисты уже давно и успешно проводят в жизнь идеи структурного программирования. Видимо, и схемотехникам, особенно работающим на матричных БИС, пора провозгласить начало эры *структурной схемотехники.*

7.5. Генераторы синхросигналов

Генератор синхросигналов обычно состоит из задающего генератора, формирователя длительности синхроимпульсов и логической схемы, формирующей фазовые импульсы. В генераторах, от которых не требуется высокой стабильности параметров, некоторые из этих трех функций могут быть совмещены в одном элементе.

Наиболее простыми и малогабаритными задающими генераторами являются мультивибраторы, выпускаемые в виде микросхем, имеющих выводы для подключения резистора и конденсатора, значения которых определяют частоту генерации.

Генератор, показанный на рис. 7.9, *а*, можно построить на любых инвертирующих элементах. Резистор R выполняет две функции: смещает рабочую точку элемента *1* на крутой участок передаточной характеристики, обеспечивая мягкое самовозбуждение, и вместе с конденсатором C служит времязадающим элементом. Значение R для ТТЛ-схем берут обычно около 300 Ом, для МДП — около 30 кОм. Эле-

мент *3* служит развязывающим буфером. Регенеративный процесс переключения схемы начинается, когда напряжение на входе элемента *1* достигает порога его срабатывания, показанного на рис. 7.9, *б* штриховой линией.

Процесс заканчивается переключением элементов *1* и *2* в противоположные состояния и передачей фронта напря-

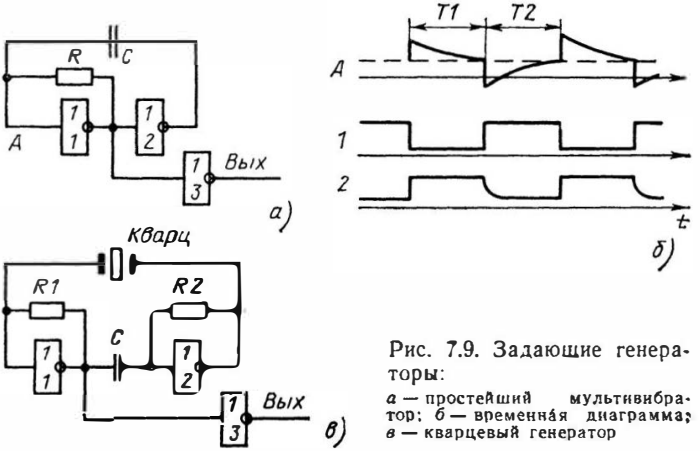


Рис. 7.9. Задающие генераторы:
а — простейший мультивибратор; *б* — временна́я диаграмма; *в* — кварцевый генератор

жения с выхода элемента *2* через конденсатор *C* на вход элемента *1*, после чего потенциал точки *A* снова будет стремиться к пороговому уровню с постоянной времени *RC*. Для ТТЛ-схем процесс несколько асимметричен: выше порогового уровня входной ток элемента *1* не превышает десятков микроампер, а ниже порога может быть более 1 мА, т. е. дает заметную добавку к зарядному току резистора. Поэтому обычно полупериод *T2* процентов на 10 короче *T1*. Ориентировочная длительность каждого полупериода — около $2RC$. При необходимости получения существенно асимметричной (или, наоборот, более симметричной) временно́й диаграммы резистор *R* можно шунтировать цепочкой из включенных последовательно резистора и кремниевого диода. Нестабильность частоты подобных генераторов имеет значение 10—20 %.

Если нужна высокая стабильность частоты, применяют кварцевые генераторы, простой вариант которого показан на рис. 7.9, *в*. Резисторы *R1* и *R2* — той же величины, что и в предыдущей схеме, выводят рабочие точки на крутые участки характеристик. Конденсатор *C* позволяет

логическим элементам иметь различные по постоянному смещению уровни крутых участков. Его импеданс на рабочей частоте должен быть существенно меньше сопротивления *R2*. В генераторах на МДП-элементах этот конденсатор не нужен. Элемент *3* — буфер. Частота выходного меандра жестко задается частотой кварцевого резонатора, и приведенная схема хорошо работает в диапазоне от 100 кГц до 10 МГц. В ТТЛ-генераторах на частотах ниже примерно 1 МГц резистор *R1* приходится шунтировать конденсатором емкостью 50—200 пФ, чтобы задавить генерацию элемента *1*, возникающую из-за слишком пологих для ТТЛ элементов фронтов. Существуют микросхемы, имеющие в своем составе схемы задающих генераторов или основные фрагменты таких схем. К выводам этих микросхем необходимо подключить лишь кварцевый резонатор на требуемую частоту и иногда — еще несколько резисторов и конденсаторов. Примерами могут служить счетчики-делители для секундомеров и часов К176ИЕ5 и К176ИЕ12, а также генератор двухфазной синхросерии для микропроцессора К580, КР580ГФ24. Более подробные сведения о различных схемах задающих генераторов приведены в [11, 28, 40].

Формирование синхроимпульсов определенной длительности часто выполняется с помощью одновибратора (см. [40]), как это показано на рис. 7.10, *а*. Если допустима нестабильность частоты более 10 %, то кольцо из одновибраторов может совмещать функции и формирователя длительности импульса, и задающего генератора, как на рис. 7.10, *б*.

В состав ряда серий входят микросхемы одновибраторов, например К155АГ1 (см. [28]). К выводам *10* и *11* микросхемы внешним монтажом подключается времязадающая *RC*-цепочка, как показано на рис. 7.10, *в*. Вывод *9* при этом остается свободным. Длительность формируемого на выходе Q импульса высокого ТТЛ-уровня со стандартными фронтами (и его инверсии на выходе $\overline{Q}$) равна приблизительно 0,7 *RC*. Диапазон подключаемых сопротивлений — от 2 до 30 кОм, емкостей — от 10 пФ до 10 мкФ. Импульс минимальной длительности около 40 нс получается, если к выводам *10* и *11* ни конденсатора, ни резистора вообще не подключать, оставив их свободными, а вывод *9* внутрикорпусного резистора подключить к +5 В. Одновибратор запускается при подаче среза на один из входов — *3* или *4* или фронта на вход *5*. Вход *5* — это вход триггера Шмитта, поэтому фронт запуска может быть очень пологим (подробнее о триггере Шмитта см. § 8.2). Вход *5* можно использовать и как вход разрешения *E*, поскольку при нулевом уровне на этом входе элемент по входам *3* и *4* не запускается. Длительность запускающего импульса, если только она не меньше 50 нс, значения не имеет, управляющим фактором является только отрицательный перепад.

Если к стабильности длительности синхроимпульса предъявляются

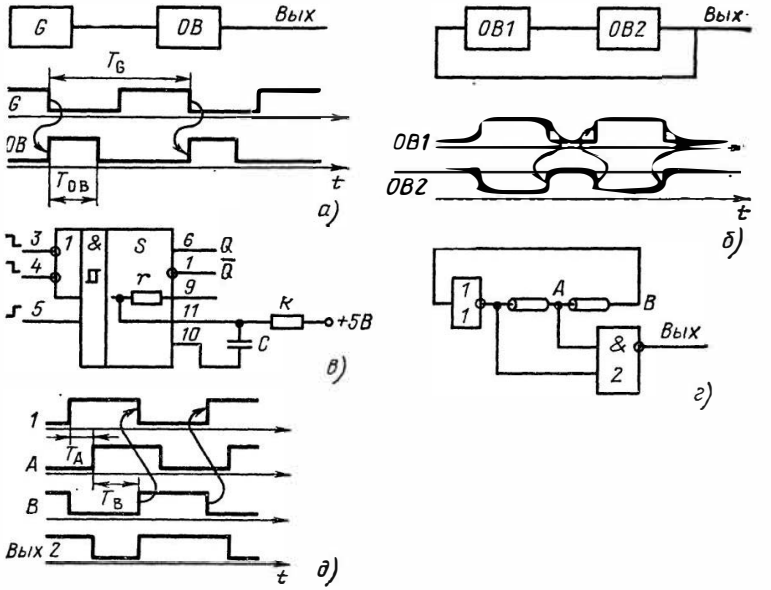


Рис. 7.10. Формирование длительности синхроимпульса:
а — цепочка генератор—одновибратор; *б* — кольцо одновибраторов; *в* — схема включения одновибратора К155АГ1; *г* — генератор на отрезках коаксиального кабеля; *д* — его временна́я диаграмма

высокие требования, то необходимую задержку формируют на отрезках коаксиального кабеля или печатных полосковых линиях. При этом нестабильность длительности не превышает единиц процента. Такой генератор показан на рис. 7.10, *г*, а его временна́я диаграмма — на рис. 7.10, *д*. Здесь T_A и T_B — задержки соответствующих отрезков кабеля (линии).

Иногда используют задающий генератор на более высокую частоту, чем тактовая, а затем эту частоту делят на счетчике. В этом случае длительность синхроимпульса и паузы можно формировать, задавая коэффициенты пересчета. Так, в частности, построена микросхема генерации двухфазной синхросерии КР580ГФ24.

Несмотря на встречающиеся в литературе рекомендации, цепочки и кольца из последовательно включенных логических элементов в качестве формирователей длительности и задающих генераторов следует использовать очень осмотрительно, поскольку, как уже упоминалось, никакой стабильности и технологической воспроизводимости частот и значений задержек в этом случае получить нельзя.

Распределение сформированных синхросигналов по фазам синхро-

низации выполняется пересчетом на логических схемах. На рис. 7.11, *а* и *б* показано, как это можно сделать с помощью шестиэлементного триггера, на рис. 7.11, *в* — с помощью *JK*-триггера в счетном режиме.

При многофазной синхронизации непосредственная разводка всех фаз становится слишком громоздкой, поэтому синхросерию формируют в два приема. На рис. 7.11, *г* показан первичный полуфабрикат шестифазной синхросерии, состоящий из опорных импульсов задающего генератора и импульсов первой фазы, полученных путем пересчета им-

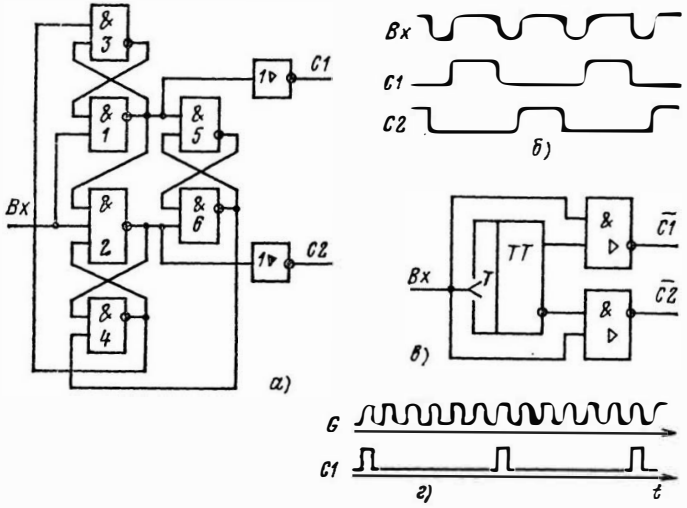


Рис. 7.11. Формирование фазовых импульсов:

а — пересчет на шестиэлементном триггере; *б* — временна́я диаграмма схемы по рис. *а*; *в* — пересчет на двухступенчатом *T*-триггере; *г* — заготовка для получения шестифазной синхросерии

пульсов генератора. Эта пара синхросигналов разводится кабелями одинаковой электрической длины по крупным блокам устройства, где из них формируются все шесть фаз синхроимпульсов с помощью, например, кольцевого распределителя (см. § 10.2).

Генераторы синхроимпульсов часто дополняют служебными схемами, изменяющими по сигналам оператора частоту синхроимпульсов. Это удобно при наладке устройства и поиске неисправностей. Применяют дискретное, по переключателю, или плавное изменение частоты синхроимпульсов на ± 10 % и более, дискретное уменьшение частоты синхросигналов в несколько раз с помощью счетчика, включаемого перед схемой распределения, пошаговый режим, т. е. выдачу каждого синхроимпульса по отдельному нажатию кнопки.

В момент смены режимов могут возникнуть нестандартные, зауженные импульсы. Чтобы этого не произошло, переключающие сигналы нужно привязать к импульсам синхронизации (см. § 8.4). Такая привязка совершенно обязательна для сигналов, управляющих вентилем пошаговой выдачи одиночных синхроимпульсов. Ряд практических вопросов проектирования систем синхронизации в целом изложен в [13, 38, 39].

ГЛАВА 8

СХЕМЫ ПРИЕМА ВНЕШНИХ СИГНАЛОВ

8.1. Функции схем приема внешних сигналов

Источниками сигналов, поступающих на цифровое устройство извне, могут быть контакты тумблеров, кнопок, реле, импульсные сигналы различных электронных схем, сигналы других цифровых устройств. Характеристики этих сигналов, как правило, отличаются от характеристик сигналов, применяемых в данном цифровом устройстве. Входной сигнал должен иметь полярность и амплитуду, которые соответствуют уровням сигналов элементов цифрового устройства. Это требование очевидно как в смысле его постановки, так и в смысле выполнения: нужно лишь ввести делители напряжения или соответствующие усилители. Для взаимных переходов между типовыми уровнями сигналов МДП, ТТЛ, ЭСЛ выпускаются специальные микросхемы. Менее очевидными оказываются требования к временным параметрам внешних сигналов. Существенными вопросами здесь являются длительность фронтов входных сигналов, дребезг контактов и привязка входных сигналов к синхросерии.

8.2. Формирование длительности фронтов

Длительность фронтов сигналов ряда источников во много раз превышает не только длительность фронтов логических элементов, но даже и длительность тактового периода. Длинные фронты могут вызвать гонки по входу, причем если связанная с этим неодновременность срабатывания настолько велика, что разные элементы срабатывают в различные такты, то любая система синхронизации здесь будет бессильна. Если на длинный фронт наложен высокочас-

тотный шум, то результирующая кривая может за время действия фронта несколько раз пересечь порог срабатывания элемента, вызвав серию срабатываний от существенно однократного сигнала. Сами элементы также плохо реагируют на длинные фронты: когда входной сигнал находится в зоне порога переключения, могут оказаться открытыми сразу оба последовательно включенных выходных транзистора, что вызывает их перегрев; в зоне переключения коэффициент усиления элемента велик (кривая вход-выход идет круто), и нахождение элемента в этом режиме при наличии даже небольшой паразитной обратной связи, например по питанию, приводит к вспышке генерации элемента, перегревающей его самого и создающей помехи. В общем, по целому ряду соображений крутизну фронтов входных сигналов желательно иметь соизмеримой с крутизной фронтов элементов.

Для превращения пологих фронтов в крутые существует специальная схема — *триггер Шмитта* (рис. 8.1), примером

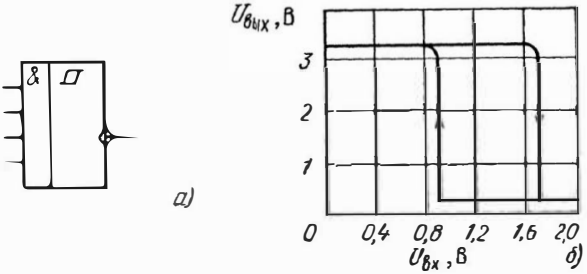


Рис. 8.1. Условное обозначение (*а*) и типовая характеристика вход-выход (*б*) триггера Шмитта

которой может служить К155ТЛ1 (см. [28]). По сравнению с характеристикой обычного ТТЛ-элемента триггер Шмитта имеет повышенный порог срабатывания, а также обладает гистерезисом шириной примерно 0,8 В, защищающим его от повторного срабатывания, если на фронт входного сигнала наложены небольшие (в пределах гистерезиса) высокочастотные шумы. Благодаря присущей триггеру Шмитта положительной обратной связи медленно меняющееся входное напряжение при достижении порогового уровня вызывает быстрое переключение элемента с длительностью фронта выходного сигнала, типичной для ТТЛ-элементов. Свойство формировать фронт проявляется на любом из четырех

входов К155ТЛ1, если на остальные три входа подан высокий ТТЛ-уровень. При низком уровне на любом из входов уровень на выходе будет однозначно высоким. Кроме формирования фронтов входных сигналов триггеры Шмитта используются и как приемники сигналов с линий связи благодаря их повышенному порогу и гистерезису.

8.3. Дребезг контактов

Дребезгом контактов называют процесс вибрации контактов, вызванный ударом при включении. После первого касания контакт размыкается, затем снова замыкается, и так несколько раз. Дребезг наблюдается у любых контактов — реле, тумблеров, кнопок. Частота коммутации при дребезге обычно лежит в пределах сотен герц — единиц килогерц, т. е. последовательные касания разделены сотнями или тысячами периодов синхронизации. При этих условиях устройство может реагировать на последовательные касания при дребезге как на отдельные входные сигналы, и действие, которое должно выполняться однократно по одному нажатию кнопки, например добавление 1 в счетчик, может повториться несколько раз.

Ликвидировать влияние дребезга контактов можно с помощью переключающего контакта и *RS*-триггера (рис. 8.2).

Рис. 8.2. Схема *(а)* и ее временна́я диаграмма *(б)* для исключения влияния дребезга контактов

При срабатывании реле, кнопки и т. д. подвижный контакт, подсоединенный к нулевому уровню сигнала, отрывается от точки *1* и движется к точке *2*. Во время пролета (отрезок t_1—t_2 на временно́й диаграмме) на обоих входах триггера резисторы *R* поддерживают высокие уровни и триггер хранит состояние, в котором он был до начала движения

контакта. При первом же касании подвижным контактом точки *2* триггер, инерционность которого на порядки меньше, чем у контакта, переключается в новое состояние (момент времени t_2). Отрывы контакта при дребезге (интервал t_2—t_3 диаграммы) не переключают триггер обратно, поскольку потенциалы на его входах при отрыве соответствуют режиму хранения. Если используемое электромеханическое устройство не имеет переключающего контакта, то для защиты от дребезга можно первым касанием замыкающего контакта запустить одновибратор с временем выдержки, бóльшим времени дребезга (время дребезга контактов нормируется ТУ). Выходной сигнал при этом формируется как дизъюнкция импульса одновибратора и включенного состояния контакта. Можно, наконец, сигнал с замыкающего контакта проинтегрировать на *RC*-цепочке с достаточно большой постоянной времени, а фронт затем сформировать на триггере Шмитта. В микропроцессорных системах временну́ю задержку для защиты от дребезга обычно формируют программным способом, подсчитывая синхроимпульсы, поэтому включать на входе специальный *RS*-триггер не требуется.

8.4. Привязка входных сигналов к синхроимпульсам

Такая привязка необходима потому, что тактированное устройство воспринимает сигналы правильно лишь в определенные интервалы времени. Если входной сигнал поступит на комбинационную схему незадолго до активного фронта синхроимпульса, то переходные процессы в КС еще не успеют затухнуть и в регистр будет принята ложная информация. Кроме того, внешний сигнал часто имеет смысл одиночного события, например нажатие кнопки, и в то же время по длительности он может занимать большое число тактов. Однако для многих узлов одиночным является событие, длящееся один такт или один фазовый период, например входной сигнал синхронного счетчика. Поэтому кроме сдвига фронта входного сигнала к началу тактового периода иногда нужно еще и длительность сигнала сделать равной длительности или такта, или фазового периода, или синхроимпульса. Узлы, решающие поставленные задачи, называют *синхронизаторами*.

Синхронизатор для двухфазной системы синхронизации на базе прозрачных защелок показан на рис. 8.3, *а*. Положительный фронт асинхронного входного сигнала может прийти в любое время, что для интервала в один такт по-

казано на рис. 8.3, *б*. Если фронт сигнала поступит не позже некоторого граничного момента *p*, например в моменты *a*, *b* или *p*, то он вызовет последовательность (показана штриховой линией) срабатываний прозрачной защелки *T1* и последующих элементов, что завершится появлением на выходе синхронизатора одиночного импульса *A*, совпадающего по времени с синхроимпульсом *2* последовательности *C2*. Если фронт поступит позже некоторого граничного момента *d* (моменты *d*, *e*), то триггер *T1* не успеет переключиться по синхроимпульсу *1* и переключится по фронту им-

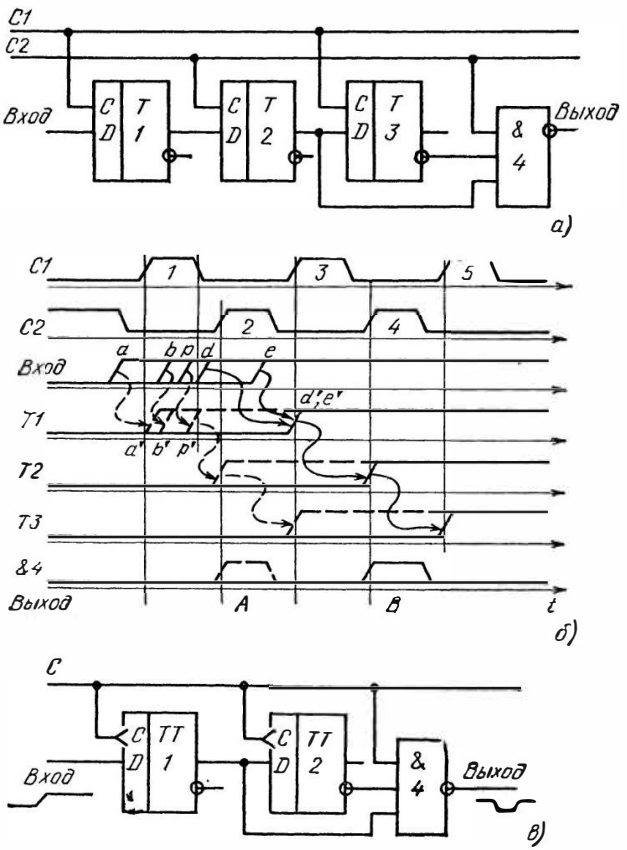


Рис. 8.3. Синхронизаторы:

а — схема на базе прозрачных защелок; *б* — временна́я диаграмма для схемы *а*; *в* — схема на базе непрозрачных триггеров

пульса *3* в момент с двойной маркировкой *d'*, *e'*. Это вызовет сдвиг выходного импульса синхронизатора в положение *B*, но все равно выходной импульс будет сформирован, и опять только один. Появившийся сдвиг на интервал, равный такту синхронизации, значения не имеет, поскольку входной сигнал по самому условию задачи асинхронный, т. е. ни к чему не привязанный. Триггер *T1* выполняет роль, как принято говорить, *арбитра,* решающего, к какому из двух соседних синхроимпульсов отнести момент появления асинхронного входного сигнала.

Между моментами *p* и *d* существует момент наибольшей неопределенности, когда энергия той части входного сигнала, которую еще пропустил синхроимпульс *1*, достаточна для перевода триггера *T1* в некоторое промежуточное состояние, но недостаточна для того, чтобы он сразу уверенно переключился. Это наиболее сложный случай *арбитража.* Поскольку в роли арбитра выступает узел с положительной обратной связью — триггер, промежуточное состояние для него неустойчиво. Из этого состояния он самостоятельно перейдет или в 0 или в 1, но чем ближе промежуточное состояние к точке равновесия, тем дольше будет развиваться экспонента перехода. Такое аномальное поведение арбитра принято называть *метастабильной аномалией.* Фронты триггера в процессе выхода его из метастабильного состояния будут нестандартными, поэтому выходной уровень первого триггера не может выполнять роль стандартного выходного сигнала синхронизатора. Эту роль выполняет выходной уровень триггера *T2,* фиксирующего уже строго по срезу *C2* то состояние, в котором к этому моменту оказался триггер *T1.* С вероятностью, весьма близкой к 1, это будет одно из его устойчивых состояний — 0 или 1, в которые триггер успеет перейти за время от среза импульса *1* до среза импульса *2.* Однако в принципе не исключена возможность «зависания» триггера *T1* в состоянии неустойчивого равновесия и на большее время, чем интервал между упомянутыми срезами. По данным некоторых экспериментов, доля встречи случаев зависания триггера на время, более чем в 10 раз превышающее задержку переключения его элементов, равна 10^{-5}—10^{-6}. Если разработчик опасается сбоев, которые могут порождаться этим явлением, то можно рекомендовать искусственно раскачивать триггер, лишая его возможности неподвижно находиться в самой точке неустойчивого равновесия. Для этого достаточно подмешать к входному сигналу, поступающему на *D*-вход триггера *T1*, меандр или сину-

соиду с амплитудой ниже порога переключения триггера и с периодом, в несколько раз превышающим номинальное время его переключения.

Таким образом, когда бы ни поступил входной сигнал, схема по рис. 8.3, *а* отреагирует на него всегда одним импульсом, притом имеющим стандартную длительность и совпадающим с синхросигналом. Такой импульс можно использовать в любой точке устройства.

По минимуму длительность входного сигнала ограничена тем, что при любом моменте поступления должно быть обеспечено его взаимодействие со срезом хотя бы одного синхроимпульса *С1*. Самым неблагоприятным моментом поступления является момент непосредственно перед срезом импульса *1* последовательности *С1* (рис. 8.3, *б*), когда *Т1* может или переключиться в 1, или — нет. В последнем случае должно быть обеспечено переключение триггера сигналом *3*, для чего входной сигнал должен перекрыть срез импульса *3*. Отсюда можно считать, что минимальная длительность входного сигнала должна быть равна длительности тактового периода плюс сумма интервалов подготовки и выдержки триггера *Т1*. По максимуму длительность одиночного входного сигнала ограничений не имеет. На срезе входного сигнала триггеры синхронизатора будут переключаться в 0 в том же порядке *Т1—Т2—Т3*, но элемент *4* выходного импульса уже не сформирует. Если это не очевидно, то можно самостоятельно продолжить временну́ю диаграмму.

Если входной сигнал по смыслу воздействия должен быть превращен не в одиночный синхроимпульс, а в постоянный уровень, например сигнал переключения режима, то выходной сигнал синхронизатора нужно снимать с выхода триггера *Т2*. Как уже отмечалось, использовать для этого выход *Т1* не следует: изредка, попадая в полузависшее состояние, триггер *Т1* будет тратить на выход из этого состояния время такта, предназначенное для управляемой им КС. К сожалению, подобные упрощенные решения в практических схемах встречаются. Встречается и совсем простое решение: без какого бы то ни было триггера входной сигнал управляет конъюнктором, на который поданы синхроимпульсы. Если в этой схеме фронт входного сигнала поступит уже во время действия синхроимпульса, то сигнал на выходе конъюнктора получится укороченным и часть элементов схемы на него среагирует, а часть — нет. Беда подобных решений в том, что они в основном работают, поскольку вероятность сбоя невелика, что развращает схемотехника. Однако не-

сколько подобных «экономичных» решений порождают в аппаратуре поток исключительно странных отдельных сбоев, которые очень трудно вылавливать именно потому, что они — отдельные.

При использовании непрозрачных триггеров схема синхронизатора несколько упрощается (рис. 8.3, *в*). Временну́ю диаграмму легко построить самостоятельно в качестве упражнения. Такую схему можно использовать и при однофазной, и при двухфазной синхронизации. Как и в схеме по рис. 8.3,*а*, если входной сигнал интерпретируется как длительно действующий уровень, то выход берется с триггера *T2*, а не *T1*, чтобы не вредило возможное зависание *T1*. Предпочтительнее использовать непроницаемые триггеры.

В цифровых системах иногда приходится обмениваться информацией между устройствами, каждое из которых имеет собственный тактовый генератор. В этом случае сигналы, поступающие из другого устройства, являются для приемника асинхронными, и их нужно пропустить через синхронизатор. Если поток сообщений большой, то возникает вопрос о максимально возможной частоте передачи при асинхронной связи. Пусть для определенности синхронизация однофазная и для привязки используется схема, показанная на рис. 8.3, *в*. Временна́я диаграмма для этого случая показана на рис. 8.4. Здесь входной сигнал

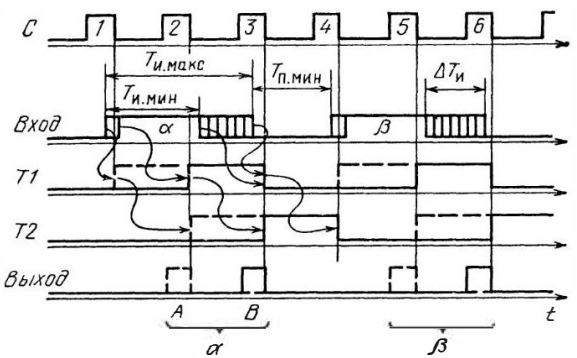


Рис. 8.4. Асинхронный обмен между устройствами

начинается в самый сложный для схемы момент времени, и в зависимости от небольших флюктуаций его фронта импульс на выходе появится или в момент *A*, или в момент *B*. Минимальная длительность паузы $T_{п.мин}$, как и минимальная длительность входного импульса $T_{и.мин}$, должна несколько превышать период следования синхросигналов. Вычитая сумму этих минимальных длительностей из периода сле-

дования входных сигналов, можно получить допустимое отклонение $\Delta T_и$ длительности импульса и паузы.

Из рис. 8.4 видно, что частота передачи информации при асинхронном обмене падает в 2,5—3 раза по сравнению с синхронным обменом, позволяющим передавать по линии бит за такт. Кроме того, усложняется обработка сигналов, которые могут поступать в любой из соседних тактов — *A* или *B*. По этой причине ту часть устройства, где интенсивность информационных обменов велика, выполняют с единой системой синхронизации, тактируемой единым генератором.

ГЛАВА 9

ДВОИЧНЫЕ СЧЕТЧИКИ

9.1. Основные характеристики счетчиков

Счетчиком называют функциональный узел, предназначенный для счета сигналов. По мере поступления входных сигналов счетчик последовательно перебирает свои состояния в определенном для данной схемы порядке. Длину списка используемых состояний (параметр *K*) называют *модулем пересчета, основанием пересчета* или *емкостью* счетчика. Одно из возможных состояний счетчика принимается за *начальное* (*нулевое*). Если счетчик начал считать с начального состояния, то через каждые *K* сигналов в нем снова устанавливается начальное состояние, а на выходе счетчика при этом появляется сигнал *K-ичного переноса CR* (от *carry* — нести).

Различные схемы счетчиков могут перебирать свои состояния в самом различном порядке. Чаще всего применяют *двоичные* счетчики, у которых порядок смены состояний триггеров соответствует последовательности двоичных чисел. Кроме того, применяют *одинарное* кодирование, когда состояние счетчика представлено местом расположения единственной единицы (например, сдвигающий регистр с одной двигающейся единицей), *унитарное* кодирование, когда состояние представлено числом единиц и более сложные виды кодирования.

Обычно счетчик перебирает свои состояния в возрастающем порядке. Если состояния перебираются в убывающем порядке, то такой счетчик называют *вычитающим*, а если направление перебора может изменяться, то счетчик назы-

вают *реверсивным*. Счетчики, которые в процессе работы для переключения требуют подачи синхросигналов, называют *синхронными*, а счетчики, у которых для переключения достаточно подавать лишь входные сигналы, — *асинхронными*. Часто счетчик снабжен входом общего сброса (*master reset*) R и входами данных D_i для *параллельной загрузки* произвольного кода. Загрузка осуществляется при подаче сигнала на еще один вход — вход параллельной загрузки PL (*parallel load*).

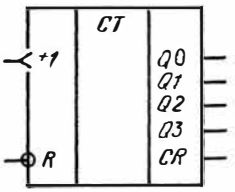


Рис. 9.1. Условное обозначение счетчика

Условное изображение счетчика показано на рис. 9.1. Здесь $Q0$—$Q3$ — выходы счетчика, комбинация значений которых определяет номер его состояния; CR — выход переноса. Суммирующий вход счетчика обозначается «$+1$», вычитающий — «-1». Это *счетные входы*. У асинхронных счетчиков они помечаются таким же символом, как и счетный вход T-триггера, указывающим полярность перепада входного сигнала, по которому счетчик меняет состояние своего выхода.

Счетчики в силу, с одной стороны, очевидности требуемого окончательного результата, а с другой — не абсолютной тривиальности схем всегда представляли плодотворнейшее поле для пробы сил изобретателей-схемотехников. Число различных типов и схем счетчиков огромно. В настоящей книге рассмотрены лишь схемы, которые широко распространены и которые лучше всего иллюстрируют наиболее результативные принципы. Специально счетчикам посвящена книга [41]. Также можно рекомендовать [33 и 36].

Двоичные счетчики строят чаще всего на основе T-триггеров, поскольку эти триггеры могут и хранить свое состояние, и суммировать с ним по модулю 2 входной сигнал. Двоичный n-разрядный счетчик содержит n T-триггеров, и его емкость $K=2^n$. Последовательность состояний 3-разрядного

Таблица 9.1

Номер	Выходы				Номер	Выходы			
	Q2	*Q1*	*Q0*	*CR*		*Q2*	*Q1*	*Q0*	*CR*
0	0	0	0		5	1	0	1	
1	0	0	1		6	1	1	0	
2	0	1	0		7	1	1	1	
3	0	1	1						1
4	1	0	0		0	0	0	0	

двоичного счетчика приведена в табл. 9.1. Сигнал восьмеричного переноса условно показан между строк, как бы возникающим после появления в счетчике его последней комбинации 111 и исчезающим при установлении в счетчике состояния 000.

9.2. Организация переносов в счетчике

Связи между триггерами, обеспечивающие их переключение в соответствии с табл. 9.1, могут быть различных типов. От вида связи существенно зависят время переключения счетчика в новое состояние, его аппаратурные затраты и ряд других свойств. Чаще всего используют три типа связей: непосредственную, тракт последовательного переноса, тракт параллельного переноса.

Схема счетчика с непосредственной связью показана на рис. 9.2, *а*. Триггер *ТТ0* пересчитывает входные сигналы по модулю 2, а состояния его выхода следующим *ТТ1* воспринимаются как входные сигналы и снова пересчитываются на 2 и т. д. Полезно самостоятельно построить временну́ю диаграмму счетчика, использующего триггеры, переключающиеся не по отрицательному, а по положительному фронту, и обратить внимание на получившийся несколько иной закон счета, а затем модифицировать схему так, чтобы она считала правильно, в соответствии с табл. 9.1.

В счетчике с непосредственной связью переключение триггеров, вызванное срезом входного сигнала, происходит

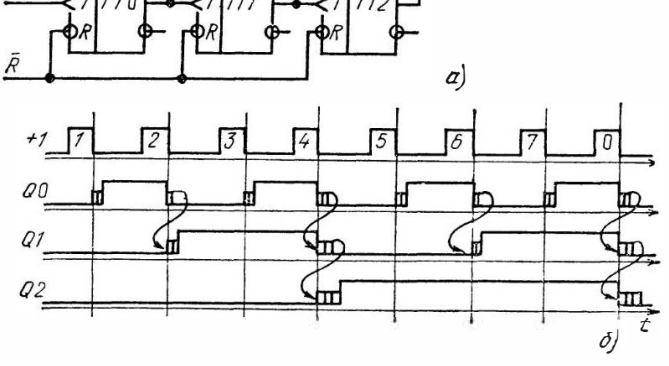


Рис. 9.2. Счетчик с непосредственной связью

триггер за триггером, *последовательно,* и задержка распространения n-разрядного счетчика, оцениваемая задержкой самого худшего случая — сменой всех единиц на все нули, в n раз больше задержки распространения одного T-триггера. Если разрядов много, то большая задержка может оказаться серьезным недостатком такого счетчика. В силу неодновременного переключения триггеров, прежде чем верное состояние установится окончательно, на выходах счетчика будут появляться различные мимолетные неверные коды. Поэтому в тактированной системе такой счетчик нельзя рассматривать как часть обобщенного синхронного регистра, несмотря на то что формально он состоит из триггеров. С таким счетчиком приходится обращаться как с комбинационной схемой, правильный выход которой устанавливается лишь спустя интервал максимальной задержки распространения.

Из-за невозможности выполнить смену состояния всего счетчика в единый момент времени, счетчики с непосредственной связью бывают только асинхронные, т. е. сигналом, переключающим их, является сам входной сигнал. Специального выхода переноса счетчики с непосредственной связью не имеют; роль выходного переноса играет срез состояния старшего разряда. Для наращивания счетчика вход еще одного T-триггера или такого же счетчика подключается непосредственно к выходу старшего разряда. Для гашения и параллельной загрузки используют приоритетные R- и S-входы триггеров, срабатывающие независимо от состояния их T-входов, что абсолютно необходимо, поскольку на T-входе триггера может быть любой уровень выхода предыдущего каскада.

Достоинствами счетчика с непосредственной связью являются предельная простота схемы и легкость ее наращивания для увеличения разрядности. В некоторых применениях автоматики достоинством оказывается и то, что такой счетчик не сбивается, если на его вход поступит некачественный, например зауженный, импульс, что бывает при работе счетчика от нецифровых приборов: фотоумножителей, радиоприемников и т. п. Не сработать от такого сигнала может лишь первый триггер, вызвав при этом ошибку не более единицы младшего разряда. Примером счетчика с непосредственной связью может служить микросхема К155ИЕ5.

Схема счетчика с трактом последовательного переноса показана на рис. 9.3, *а*. Если рассматри-

вать временну́ю диаграмму рис. 9.3, *б* по вертикалям, начиная с входных импульсов, то видно, что входной импульс проходит сквозь все каскады счетчика, в которых содержатся единицы, попутно сбрасывая все их в нуль, и переводит в 1 первый встреченный на пути погашенный триггер, причем сквозь этот каскад импульс уже не проходит. Как видно из табл. 9.1, этот принцип построения счетчика соответствует

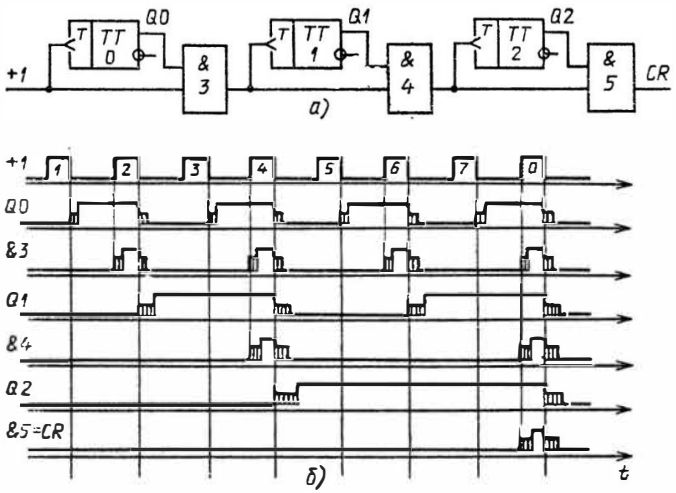


Рис. 9.3. Счетчик с трактом последовательного переноса

алгоритму двоичного счета. На выходе последнего элемента И вырабатывается сигнал переноса *CR*, который используется при наращивании разрядности счетчика. Диаграмма на рис. 9.3, *б* построена с указанием уровней неопределенности, чтобы подчеркнуть, что длительность входного сигнала такого счетчика должна выбираться с учетом возможного его заужения при распространении по цепочке элементов тракта переноса. Эта схема больше других боится зауженных входных импульсов.

Счетчик с трактом последовательного переноса по значению задержки распространения не имеет существенных преимуществ перед счетчиком с непосредственной связью. Однако он оказывается значительно удобнее, если в процессе работы счетчика потребуется с помощью логических элементов выполнить какие-либо переключения в связях между

триггерами, например переключить счетчик на вычитание или изменить содержимое отдельных триггеров. Поскольку в диаграмме работы такого счетчика существуют отрезки времени, когда T-входы всех триггеров отключены от схемы запертыми конъюнкторами тракта переноса, вносимые изменения не вызовут нежелательных переключений триггеров.

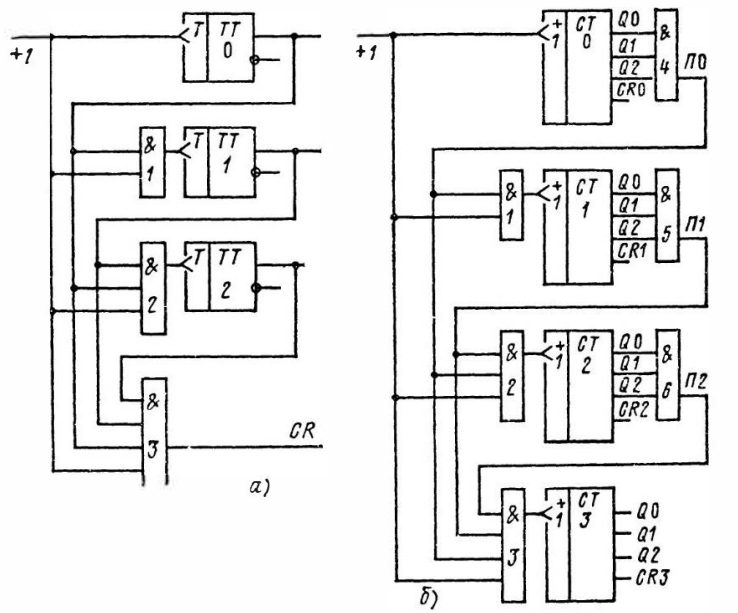


Рис. 9.4. Счетчик с параллельным переносом асинхронный:
а — функциональная схема малоразрядного счетчика; *б* — способ построения параллельного межгруппового переноса

Оба счетчика — и с непосредственными связями, и со связями по тракту переноса — являются счетчиками с последовательным переносом (*ripple carry counter*). Задержка распространения такого счетчика растет пропорционально числу его разрядов. Для уменьшения времени задержки используют счетчик с параллельным переносом (*parallel carry counter*), распространенный вариант схемы которого показан на рис. 9.4, *а*. Принцип параллельного переноса заключается в следующем. На входе каждого триггера (кроме самого первого) включен конъюнктор. Входной сигнал $+1$ поступает параллельно на все конъюнк-

торы и там, где они открыты, вызывает одновременное переключение соответствующих триггеров. На вход каждого конъюнктора кроме входного сигнала поданы выходы всех триггеров младше данного разряда. Поэтому при подаче сигнала *+1* изменят свое состояние все те триггеры, перед которыми все более младшие триггеры находились в состоянии 1. Это полностью соответствует табл. 9.1. По такому принципу построен, например, 4-разрядный счетчик К561ИЕ10.

Здесь и далее объяснение работы сложных схем для облегчения понимания ведется в привычном базисе НЕ, И, ИЛИ. Уяснив принцип работы, можно без труда перевести схему на язык любого инвертирующего базиса. Конъюнкторы могут входить в состав триггера, и читателю полезно построить схему по рис. 9.4, *а*, выполненную на *JK*-триггерах, показанных на рис. 6.10, *в*.

Импульс выходного переноса формируется конъюнктором переноса *3*, который открывается, когда счетчик находится в состоянии *все единицы*, и пропускает на выход *CR* тот самый входный импульс, который сбрасывает весь счетчик в 0.

В счетчике с параллельным переносом, построенном по рис. 9.4, *а*, все триггеры начинают переключаться почти одновременно в пределах лишь разброса задержки входных конъюнкторов триггеров, в результате время задержки у счетчиков с параллельным переносом заметно меньше, чем у счетчиков с последовательным, и притом не зависит от числа разрядов. Для схемы по рис. 9.4, *а* время задержки равно сумме задержек конъюнктора (если он выполнен отдельно) и *T*-триггера.

Следует заметить, что этот счетчик, в отличие от счетчика с непосредственной связью не безразличен к фронтам и длительности входных сигналов. От зауженного импульса или слишком короткой паузы между ними часть триггеров может сработать, а часть — нет, и код в счетчике будет искажен самым неожиданным образом.

Малая задержка счетчиков с параллельным переносом оплачивается дополнительной аппаратурой входных конъюнкторов триггеров, причем, как видно из рис. 9.4, *а*, конъюнктор каждого следующего разряда должен иметь на один вход больше конъюнктора предыдущего. Поэтому на максимальную разрядность таких счетчиков накладывает ограничение максимально возможное число входов И логического элемента. Схемы большей разрядности приходится наби-

рать из нескольких малоразрядных счетчиков, которые при этом называют *группами.* Малоразрядные группы, построенные по схеме рис. 9.4, *а*, проще всего наращивать, подключая выход переноса более младшей группы к входу *+1* более старшей. Получается *последовательный межгрупповой перенос,* который на уровне межгрупповых связей эксплуатирует тот же принцип, который в счетчике с трактом последовательного переноса использован на уровне межразрядных связей. Как и следует ожидать, группы работают не одновременно: каждая следующая группа начинает (и, видимо, оканчивает) работу со сдвигом, равным значению задержки в элементе, формирующем межгрупповой перенос (обычно даже в двух элементах, поскольку они инвертирующие). Счетчик описанного типа имеет параллельный перенос внутри группы и последовательный между группами.

Перенос между группами можно сделать и параллельным точно по тому же принципу, что и параллельный перенос между разрядами, как показано, например, на рис. 9.4, *б*. Для определенности каждая из групп подразумевается построенной по рис. 9.4, *а*. Конъюнкторы *4—6*, подключенные к выходам *Q0*, *Q1*, *Q2* каждой группы, выявляют в своей группе ситуацию *все единицы,* которую, используя сумматорную терминологию, можно назвать ситуацией *прозрачности* (*П*) тракта переноса группы. Если группа прозрачна по переносу, то сигнал *+1* должен, как бы пройдя ее насквозь, стать сигналом *+1* для следующей группы. Конъюнкторы *1—3* на входе каждой 3-разрядной группы на рис. 9.4, *б* включены так же, как и конъюнкторы на входе каждого триггера, т. е. одноразрядной группы» на рис. 9.4, *а*. При совпадении сигналов прозрачности групп на всех входах каких-либо конъюнкторов эти конъюнкторы одновременно пропускают входные сигналы *+1* на входы тех групп, где должны произойти переключения. Выходной перенос каждой группы при таком соединении не используется. В схемах средней интеграции элемент И, вырабатывающий сигнал прозрачности группы, иногда вводят в состав микросхемы счетчика. Однако, существуют микросхемы счетчиков с параллельным переносом, в которых не формируется никаких функций межгрупповых переносов, например К561ИЕ10, и при необходимости весь межгрупповой тракт переноса создается внешним монтажом.

Счетчик с параллельным переносом, собранный по схеме рис. 9.4, *а*, является *асинхронным:* все процессы в нем инициируются входным сигналом и в принципе могут быть со-

вершенно не привязаны к синхросерии. Триггеры счетчика начинают переключаться с некоторой задержкой относительно среза входного сигнала, определяемой сложностью логической схемы тракта переноса. Поэтому при использовании таких счетчиков в синхронных устройствах смена их состояний будет отставать от смены состояний других синхронных регистров устройства, и возможные последствия такого отставания приходится учитывать. От этого недостатка свободен *синхронный* счетчик, собранный по схеме, показанной на рис. 9.5, *а*, на основе синхронных триггеров

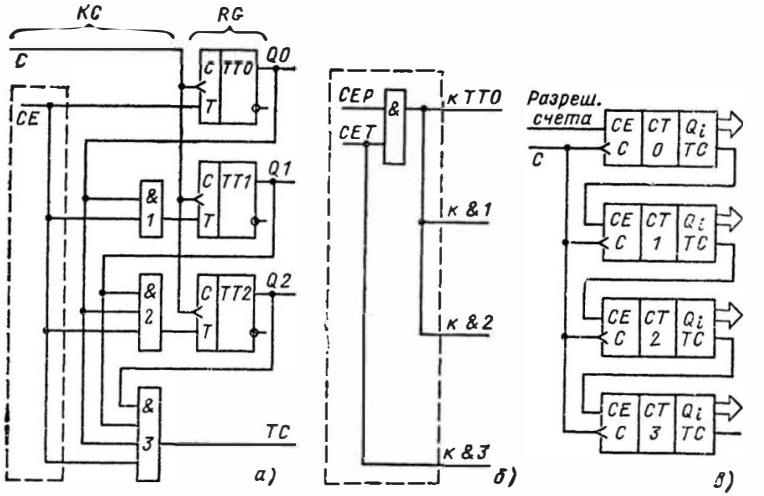


Рис. 9.5. Счетчик с параллельным переносом синхронный:

а — функциональная схема малоразрядного счетчика; *б* — двухвходовой вариант тракта разрешения счета; *в* — способ построения синхронного счетчика из нескольких групп

типа, например, таких, как на рис. 6.13, *д*. Входным сигналом, играющим для счетчика роль сигнала $+1$, служит не фронт, а потенциал *CE* (*count enable* — разрешение счета), который может вырабатываться комбинационными схемами с любой задержкой в пределах такта синхронизации. Сигнал *CE* не переключает триггеры, а лишь подготавливает их к этому по потенциальным *T*-входам (или по объединенным *J*- и *K*-входам). Смена состояний триггеров происходит строго по активному фронту синхроимпульса одновременно со всеми другими регистрами цифрового устройства, поскольку синхроимпульс в данном случае заведен непосредственно на

C-входы триггеров. Синхронный счетчик уже без каких-либо натяжек вписывается в обобщенную схему однотактного цифрового устройства комбинационная схема—регистр, показанную на рис. 7.5. Выходной перенос синхронного счетчика формируется на выходе *TC* (*terminal count*). Уровень *TC* становится активным при переключении счетчика в состояние *Все единицы* и существует до момента переключения его в состояние *Все нули*. При наращивании разрядности синхронного счетчика выход *TC* потенциала переноса одной группы подключается к потенциальному входу разрешения *CE* следующей, более старшей группы, на которую поступают те же синхроимпульсы, что и на первую, как показано на рис. 9.5, *в*. Триггеры всех групп при этом переключаются одновременно, по активному фронту *C*-сигнала, без какой-либо задержки относительно друг друга. Все переходные процессы в тракте переноса начинают протекать после переключения триггеров счетчика, и к моменту поступления активного фронта следующего синхросигнала оказываются уже оконченными.

Развитием рассмотренной схемы является синхронный счетчик с двумя разрешающими входами: *CET* (*count enable trickle*, в вольном переводе — разрешение переноса) и *CEP* (*count enable parallel* — параллельное разрешение счета). Такой счетчик получится, если обведенный штриховой линией фрагмент схемы рис. 9.5, *а* заменить фрагментом, показанным на рис. 9.5, *б*. Примером может служить микросхема К155ИЕ9. Вход *CET* разрешает формировать потенциал переноса *TC*. При наращивании разрядности выход *TC* более младшей группы счетчика подключается ко входу *CET* соседней старшей группы, так же как на рис. 9.5, *в* соединены выходы *TC* со входами *CE*. В результате образуется потенциальный тракт переноса всего счетчика. Синхроимпульсы на *C*-входы всех групп подаются параллельно, так же как и на рис. 9.5, *в*.

Вход *CEP* при наличии сигнала *CET* разрешает переключение триггеров при поступлении синхросигнала, т. е. служит общим сигналом разрешения счета. Входы *CEP* нескольких групп счетчика могут быть подключены параллельно к источнику внешнего разрешающего сигнала. Если этого не требуется, то обычно входы *CEP* второй и всех последующих групп счетчика соединяют параллельно и подключают к выходу *TC* первой группы.

Очевидно, что для любого из рассмотренных трактов переноса можно построить двойственную схему, в которой

вместо элементов И будут стоять элементы ИЛИ-НЕ, сигналы для которых будут сниматься с инверсных выходов триггеров, а входы $+1$, *CE*, *CET* и т. д. будут иметь активный низкий уровень. Аналогично, введя необходимые коррекции, можно в любой схеме использовать *T*-триггеры, у которых активным фронтом является положительный.

Любые из рассмотренных типов счетчиков можно применять в устройствах и с однофазной, и с многофазной синхронизацией. Необходимо лишь четко представлять, по какому фронту синхросигнала счетчик переключается, какая относительно этого фронта возникает задержка на выходах триггеров, за какое время до активного фронта нужно установить все подготавливающие потенциалы, и соответствующим образом спроектировать схемы, включенные до и после счетчика.

9.3. Реверсивные счетчики

Принцип работы вычитающего счетчика прослеживается в табл. 9.1, если просматривать ее в обратном порядке, снизу вверх. Если в счетчике с непосредственной связью в *режиме сложения* более старший разряд переключается при переходе соседнего младшего разряда от 1 к 0, то для *режима вычитания* он должен переключаться при переходе соседнего младшего разряда от 0 к 1. В счетчике с трактом последовательного переноса при сложении входной сигнал проходил последовательно сквозь те конъюнкторы, триггеры которых были в 1, а при вычитании сигнал последовательного *займа* должен проходить через те конъюнкторы, триггеры которых находились в 0. В счетчике с параллельным переносом на входном конъюнкторе каждого триггера собирались единицы всех более младших разрядов, а теперь для формирования параллельного займа на каждом входном конъюнкторе должны собираться нули всех более младших разрядов. Если функцией прозрачности для формирования группового переноса была конъюнкция всех единиц группы, то для формирования группового займа функцией прозрачности должна быть конъюнкция нулей всех разрядов группы. Высказанные частные соображения можно подытожить в виде общего правила: *для превращения суммирующего счетчика в вычитающий нужно сигналы управления трактом переноса снимать с противоположных выходов триггеров счетчика,* т. е. везде вместо выхода Q использовать выход $\overline{Q}$, и наоборот, если, например, тракт переноса построен по двойственной схеме.

Реверсивный счетчик можно построить, если на выходы каждого триггера подключить мультиплексор «2—1», который по управляющему сигналу $UP/\overline{DN}$ (*up/down* — вверх/вниз) подключает к тракту переноса один из выходов триггера, например прямой для сложения, инверсный для вычитания. Такой счетчик для случая параллельного переноса показан на рис. 9.6. На выходе последнего конъюнктора *5* в режиме сложения вырабатывается перенос, а в режиме вычитания — заем. Импульс займа формируется из входного счетного импульса при состоянии счетчика *Все нули*. Переключать уровень направления счета можно лишь при нулевом уровне счетного сигнала, когда входные конъюнкторы триггеров (элементы *3* и *4*) закрыты, т. е. триггеры находятся в режиме хранения и изменение уровней при переключении мультиплексоров не воздействует на их счетные входы.

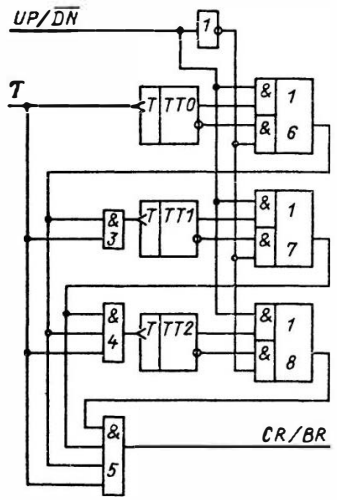


Рис. 9.6. Реверсивный счетчик с общим входом сложения-вычитания

Добавив мультиплексоры «2—1» к схеме, показанной на рис. 9.3, можно построить реверсивный счетчик с трактом последовательного переноса. Переключение его режима также должно выполняться при нулевом уровне на счетном входе. А на основе схемы с непосредственной связью (см. рис. 9.2) хороший реверсивный счетчик построить не удастся. Входы *T*-триггеров этого счетчика никогда не отключаются от выходов их младших соседей, поэтому при переключении мультиплексоров, управляющих режимом счета, на входах в среднем у половины триггеров сигналы сменятся с 1 на 0; эти триггеры перебросятся, вызвав опять в среднем в половине случаев переключение своих старших соседей и т. д., т. е. при каждой смене режима счета код в счетчике будет существенно изменяться. В большинстве применений это недопустимо, и разработчики добавляют к счетчику громоздкие схемы блокировки. В таких случаях рациональнее отказаться от принципа непосредственных

связей и строить реверсивный счетчик с другим типом переноса.

Счетчик, показанный на рис. 9.6, имеет один счетный вход и для сложения, и для вычитания и управляющий вход переключения режима. Этот принцип использован, например, в микросхемах К561ИЕ11 и К561ИЕ14. Находит применение и другой способ управления реверсивными счетчиками. Его иллюстрирует рис. 9.7, *а* на примере 2-разрядного

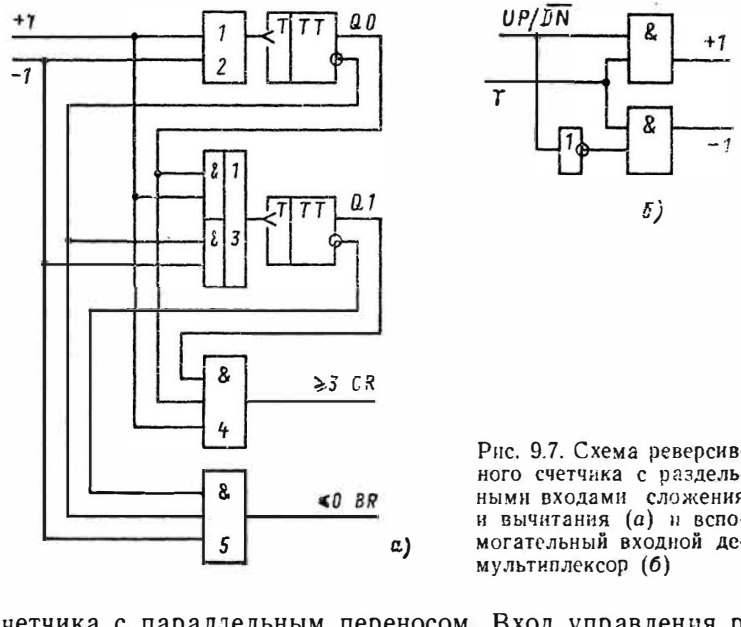


Рис. 9.7. Схема реверсивного счетчика с раздельными входами сложения и вычитания (*а*) и вспомогательный входной демультиплексор (*б*)

счетчика с параллельным переносом. Вход управления режимом здесь отсутствует, а импульсы для сложения и вычитания подаются по двум различным входам. Счетчик имеет, по сути, два независимых тракта переноса — один для сложения, а другой для вычитания, подключенных ко входу триггера через элемент ИЛИ. Сразу оба счетных сигнала подавать нельзя. Счетчик имеет раздельные выходы переноса *CR* и займа *BR* (от *borrow* — занимать). При наращивании разрядности выход *CR* подключается к входу *+1* соседней старшей группы, а выход *BR* — к ее входу *—1*. Принцип двух счетных входов использован, в частности, в микросхемах К155ИЕ6 и К155ИЕ7.

Упражнение. На базе счетчиков, показанных на рис. 9.7, самостоятельно построить многоразрядный реверсивный

счетчик с раздельными счетными входами и с параллельным межгрупповым переносом.

Счетчик, имеющий раздельные входы для сложения и вычитания, можно превратить в счетчик с общим счетным входом, включив на входе демультиплексор «1—2», изображенный на рис. 9.7, *б*.

9.4. Счетчики по произвольному основанию

Различные области применения требуют счетчиков с модулями пересчета (основаниями), не только кратными целой степени двойки, но и другими, например для работы в десятичной системе — 10, для схем часов и календарей — 60, 24, 7... В общем случае требуется строить счетчики по любому заданному основанию K. Иногда пересчет выгоднее реализовать на единственном счетчике, иногда — разложить основание на два сомножителя: целую степень двойки, реализуемую на обычном двоичном счетчике, и оставшееся нечетное число, являющееся основанием счетчика уже меньшего размера, чем он получится, если его строить непосредственно для заданного основания. На базе готовых счетчиков счетчик по произвольному основанию можно построить тремя основными способами.

1. Двоичный счетчик разрядности n, такой, чтобы 2^n было больше K, дополняется элементом И, который по состо-

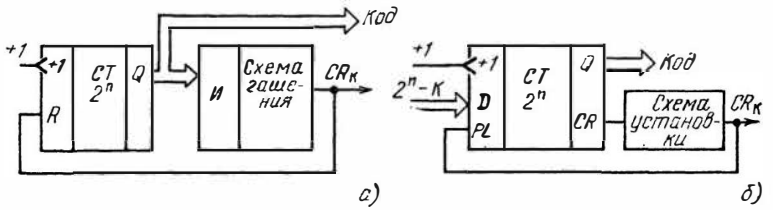


Рис. 9.8. Структурные схемы счетчиков по произвольному основанию: *а* — со сбросом счетчика в 0; *б* — с загрузкой дополнения

яниям выходов Q_i обнаруживает код конца счета (обычно $K-1$), после чего по цепи R сбрасывает счетчик в 0. Структурная схема, реализующая этот способ, показана на рис. 9.8, *а*. Сигнал, сбрасывающий счетчик, одновременно является и сигналом *K-ичного* переноса CR_K. Достоинства способа: естественная двоичная последовательность кодов от 0 до $K-1$; использование обычно имеющегося в счетчиках

входа R. В случае суммирующего счетчика достаточно собрать на элементе И лишь прямые выходы тех триггеров, которые при коде конца счета равны 1. Число входов элемента И, таким образом, зависит от кода конца счета.

2. Второй способ иллюстрирует рис. 9.8, *б*. Двоичный счетчик перед началом счета по тракту параллельной загрузки загружается кодом дополнения числа K до 2^n. Кодом конца счета в этом случае является естественное переполнение счетчика, т. е. код *Все единицы,* обнаруживаемый штатным трактом переноса, в результате чего вырабатывается сигнал CR. Сигнал CR, воздействуя на вход PL, управляющий параллельной загрузкой, снова устанавливает в счетчике дополнение K до 2^n. Достоинство способа — использование штатного тракта переноса и имеющихся во многих счетчиках входов параллельной загрузки, а также легкая смена основания пересчета. Для этого входы D_i нужно подключить не к константам 1 и 0 (питание и общий провод), а к выходу специального регистра начальных состояний. Недостатком способа является неестественная последовательность получаемых кодов, требующая в случае их использования перекодировки. Поэтому данный способ применяется, когда показания счетчика не важны, а используется лишь сигнал его выходного переноса. Это типично для задачи деления частоты входных сигналов на некоторое число K. Счетчики, выполняющие эту функцию, называют *делителями.*

3. При третьем способе в качестве кода начала и конца счета выбирают некоторую произвольную пару кодов, разность между которыми равна $K-1$. Этому способу присущи сложности как первого, так и второго, и используется он лишь в специальных случаях.

Попытка реализовать счетчик по произвольному основанию простейшими, наиболее очевидными способами, например путем подачи выхода конъюнктора, обнаруживающего код K, на вход гашения R или выхода переноса CR непосредственно на вход параллельной загрузки PL, почти всегда нарушает правила заведения обратных связей в тактируемых устройствах и, естественно, порождает сбои из-за гонок. Ошибки этого типа при построении счетчиков по произвольному основанию очень распространены среди малоопытных разработчиков, поэтому целесообразно подробнее рассмотреть некоторые не совсем очевидные моменты.

Во-первых, если счетчик имеет тракт межразрядного или межгруппового переноса последовательного типа, то в про-

цессе счета триггеры переключаются не строго одновременно. Порядок их переключения может быть любым, и при этом сразу после активного фронта счетного сигнала на некоторые мгновения могут возникнуть коды конца счета. Это приводит к появлению на выходе элемента И помех, способных досрочно сбросить счетчик в 0. Поэтому элемент И или схема, использующая его сигнал, должны открываться лишь спустя время задержки счетчика, например, синхросигналом иной фазы или другим задержанным импульсом. Счетчик с непосредственной связью, имеющий жесткий порядок переключения триггеров, опасных помех такого типа не порождает. Счетчики, кодом конца счета которых являются все единицы, также не имеют этого недостатка, поскольку схемы штатных трактов переноса всегда построены так, что ложных сигналов на выходе CR не появляется.

Во-вторых, для гарантированного гашения счетчика (или для его параллельной загрузки) длительность импульса гашения (или загрузки) должна быть не менее некоторого определенного значения. В то же время, при гашении счетчика сигналом элемента И, реагирующего на код конца счета, любой триггер счетчика, первым переключившийся в 0, тут же вызовет исчезновение кода конца счета и как следствие — сигнала гашения R. Чтобы длительность сигнала гашения была не менее требуемой паспортом, сумма двух величин — задержки счетчика по тракту вход гашения R — выход Q и задержки элемента И конца счета — должна быть гарантированно больше минимально допустимой длительности импульса гашения по R-входу. Это требует гарантии минимальных значений задержек, входящих в рассматриваемую сумму, а как раз этих гарантий изготовитель практически никогда и не дает. Типовым примером могут быть паспортные данные одной из распространенных микросхем счетчиков: максимально возможная задержка тракта вход R — выход Q — 40 нс, минимально допустимая длительность импульса гашения по R-входу — 50 нс. При таких значениях допусков обсуждаемая сумма задержек с очень большой вероятностью окажется меньше требуемых 50 нс. Поэтому у заманчиво простой схемы счетчика по произвольному основанию, в которой сигнал элемента И конца счета заведен непосредственно на вход R гашения счетчика, малая вероятность сбоев не гарантируется паспортными данными элементов. Несмотря на то что подобные схемы в реальной практике применяются и при этом даже работают, для серийной аппаратуры такие ре-

шения нельзя считать грамотными. Как и во всех других случаях, связанных с гонками, эксперимент не может быть критерием правильности решения: при другой температуре или других образцах комплектующих элементов схема может давать сбои.

Чтобы гарантировать правильность работы, сигнал элемента И (или переноса) необходимо или удлинить до требуемого значения, запустив им одновибратор, или запомнить на триггере. Эти узлы и обслуживающие их элементы (гашения триггера и т. п.) на рис. 9.8, *а* и *б* показаны условно в виде схем гашения и установки.

В-третьих, как известно, счетчик по основанию K должен при поступлении каждого K-го импульса переходить из состояния $K-1$ в 0. Например, счетчик по основанию 10 по каждому десятому импульсу переключается из 9 в 0. Однако схема, изображенная на рис. 9.8, *а*, решает эту задачу не корректно: из $(K-1)$-го состояния она сначала переходит в K-е состояние, что для счетчика по основанию K лишено смысла, и только потом, спустя один или два фазовых интервала многофазной синхронизации или некоторое время задержки одновибратора, переходит в 0. Не все потребители могут пережить столь длительное неверное состояние.

Схема счетчика по произвольному основанию, годящаяся для любого типа переноса, не боящаяся гонок и выдающая на выходе только допустимые K-ичные коды, т. е. схема, универсальная и правильная во всех отношениях, показана на рис. 9.9, *а*. В тракте входных сигналов $+1$ установлен демультиплексор, «1—2», представленный конъюнкторами *1* и *2*, который направляет входные сигналы или на счетный вход двоичного счетчика (канал *СЧЕТ*), или на вход сброса R (канал *СБРОС*). Демультиплексор управляется триггером T, который на один такт запоминает сигнал элемента И, обнаружившего код конца счета $K-1$ (например, код *9* для десятичного счетчика). Для исключения реакции на неустановившиеся состояния двоичного счетчика триггер синхронизирован фазой, отличной от фазы поступления входных сигналов $+1$ (рис. 9.9, *б*). Сигнал *СБРОС* в принципе может воздействовать и не на вход гашения R, а на вход параллельной загрузки PL. Несколько развив схему рис. 9.9, *а*, можно построить и реверсивный счетчик по произвольному основанию.

Схема, показанная на рис. 9.9, *а*, предназначена для схемотехника, которому доступны лишь входы и выходы счетчика и недоступны его внутренние цепи, что типично

при разработке устройств на микросхемах. Если же разработчику доступна вся схема счетчика, например при проектировании схем на элементах матричных БИС, то для счета по произвольному основанию можно модифицировать тракт переноса счетчика. Лучше всего это получается при параллельном переносе, когда входной сигнал $+1$ заведен параллельно на все каскады счетчика. Чтобы двоичный

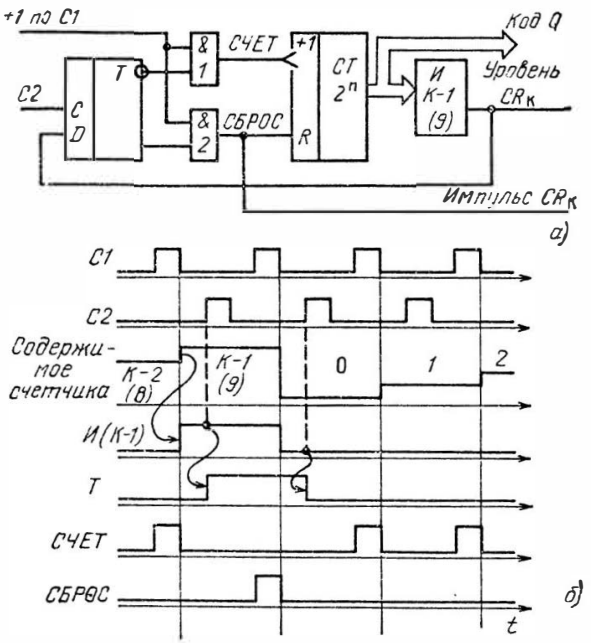


Рис. 9.9. Универсальный способ построения счетчика по пронзвольному основанию на базе двоичного счетчика

счетчик перешел из состояния $K-1$ не в состояние K, а в 0, необходимо:

закрыть входные конъюнкторы тех триггеров, которые при смене кода с $K-1$ на K переключаются из 0 в 1, т. е. запретить им это переключение;

у тех триггеров, которые при смене кода с $K-1$ на K остаются в 1, провести на T-вход, в обход их входных конъюнкторов, сигнал $+1$, обеспечив таким образом требуемое переключение этих триггеров в 0;

в каскады, которые при смене кода с $K-1$ на K оста-

ются в 0 или переключаются из 1 в 0, не вмешиваться, дав возможность штатным цепям переноса сделать свое полезное дело.

На рис. 9.10 в качестве примера показана схема счетчика с параллельным переносом по основанию $K=6$. Код $K-1=5$ обнаруживается конъюнктором И. Схема нормального двоичного переноса при переходе счетчика от $K-1=101$ к $K=110$ переводит средний разряд из 0 в 1, поэтому конъюнктор *3* в этом такте запирается сигналом с элемента И. Старший разряд при смене кода с 101 на 110 остается в 1, поэтому параллельно входному конъюнктору *a* триггера *TT2* включен конъюнктор сброса *b*, который при обнаружении кода конца счета пропускает на *T*-вход триггера входной сигнал. Младший разряд при переходе от 101 к 110 переводится в 0 схемой двоичного переноса, поэтому вмешательства не требует. Конъюнктор переноса *6*, открываемый элементом И при коде $K-1=5$, пропускает шестой входной сигнал, формируя тем самым шестеричный перенос CR_6. Модификация тракта параллельного переноса используется в микросхемах счетчиков по недвоичным основаниям, в частности в десятичном счетчике К155ИЕ6.

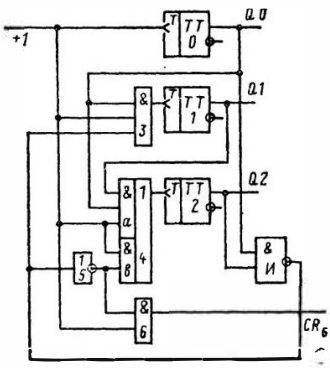


Рис. 9.10. Счетчик по основанию $K=6$, построенный путем модификации тракта переноса

В счетчиках с трактом последовательного переноса принцип модификации тракта переноса применим ограниченно, поскольку только при коде *Все единицы* перенос попадает на все разряды, а при других кодах цепочка переноса обрывается раньше, при четных кодах уже на самом младшем разряде. Поэтому решения часто оказываются находками «по месту» для конкретного основания. Используются приоритетные *R*- и *S*-входы триггеров, запрещение переключений по *J*- или *K*-входу, сочетание последовательного и параллельного переносов, разложение основания пересчета на сомножители (см. [36, 42, 59]). Подобным образом построены десятичный счетчик К155ИЕ2 и двенадцатиричный К155ИЕ4 (см. [9]), который, кстати, перебирает коды не строго в соответствии с двоичной системой. Ряд практических схем счетчиков приведен в [8, начиная со стр. 689].

9.5. Особенности микросхем счетчиков. Области применения

Микросхемы счетчиков выпускают в составе целого ряда серий (см. [9, 11, 28]), как правило, в корпусах с 14 или 16 выводами, по четыре триггера в корпусе. Все они хорошо приспособлены к наращиванию разрядности. Поскольку ограниченное число выводов корпуса не позволяет иметь полный набор желаемых управляющих воздействий на счетчик, различные комбинации воздействий реализуются в микросхемах различных типов. Отсюда большое разнообразие типов и модификаций выпускаемых счетчиков. Варьируя связи между триггерами, предприятия изготавливают микросхемы счетчиков с модулем пересчета 16, 12 и 10. В некоторых ТТЛ-сериях счетчик разбивается на две части: триггер (т. е. счетчик на 2), с выведенными входом и выходом и счетчик на 8, 6 или 5, также с автономными входом и выходом. В результате одну и ту же микросхему можно использовать например, как счетчик и на 6, и на 12. Таковы счетчики ИЕ2, ИЕ4, ИЕ5 серии К155. Счетчик 564ИЕ14 путем задания 1 или 0 на входе переключения режима превращается в двоичный или десятичный. Счетчики с параллельным переносом часто выполняют реверсивными.

Счетные микросхемы выпускают в виде как *счетчиков-делителей*, имеющих лишь счетный вход и выход переноса, без выводов состояний триггеров, так и собственно счетчиков, у которых выведены выходы всех триггеров, а часто и их входы D_i для параллельной загрузки начального состояния. Большинство счетчиков имеет вход гашения R. Загрузку и гашение обычно делают по приоритетным S- и R-входам триггеров, а J- и K-входы используют только для счета. Как правило, вход PL, управляющий загрузкой счетчика параллельным кодом, имеет приоритет по отношению к счетному входу, т. е. при одновременной подаче активного счетного фронта и активного уровня загрузки операция счета игнорируется, а выполняется операция параллельной загрузки кода со входов D_i. Вход гашения R практически всегда имеет наивысший приоритет по отношению как к счету, так и к загрузке. Однако обычно не допускается, чтобы переходные процессы этих трех операций налагались друг на друга. Их активные фронты должны быть разделены интервалами *восстановления*, значения которых указываются в справочниках, а типовое значение для ТТЛ-счетчиков составляет 20—40 нс. Поскольку ряд входов микросхемы счетчика воздействует сразу на много элементов, в микросхемы часто вводят *буферные усилители*. В результате нагрузка входов счетчика для источников сигналов равна единичной нагрузке данной серии или близка к ней. Эти буферы часто инвертируют сигнал, в различных микросхемах используются триггеры, срабатывающие как по фронту, так и по срезу, поэтому у счетчиков входы, а также выход переноса встречаются как с активным высоким,

так и с активным низким уровнями; активные уровни управляющих сигналов также могут быть и высокими, и низкими. Это разнообразие часто порождает ошибки и требует внимательного отношения к документации, а иногда и построения вспомогательных временных диаграмм, особенно при совместной работе счетчиков различных типов.

Микросхемы счетчиков имеют несколько различных по смыслу входов и несколько типов выходов, поэтому их динамические качества характеризуются многими параметрами. Типовой набор временных параметров следующий: 1) максимальная частота счета; 2) минимальные длительности импульсов: счетного, сброса, параллельной загрузки; 3) задержки распространения трактов: счетный вход — выход переноса; счетный вход — выход Q; вход параллельной загрузки — выход Q; вход сброса — выход Q; 4) времена подготовки и выдержки входов данных D_i по отношению к активному фронту сигнала параллельной загрузки; 5) времена восстановления (т. е. затухания переходных процессов) цепи параллельной загрузки и цепи сброса перед поступлением активного фронта счетного импульса.

Области применения счетчиков и счетчиков-делителей исключительно разнообразны. Счетчики-делители с фиксированным коэффициентом пересчета частоты стабильных кварцевых генераторов используют в различных датчиках времени, часах, календарях. Примерами являются 15-разрядный двоичный делитель К176ИЕ5, разработанные специально для часов микросхемы К176ИЕ12, К176ИЕ17 и др. Ряд практических схем делителей для секундомеров и часов приведен в [42]. В составе серии К145 есть специализированные делители для применения в электромузыкальных инструментах, в том числе генератор тонов музыкальной шкалы.

Если счетчик-делитель по сигналу каждого переполнения не сбрасывать в 0, а загружать по входам D_i некоторым начальным кодом из специального загрузочного регистра, то коэффициент пересчета такого счетчика можно менять, заменяя содержимое регистра. Для суммирующего счетчика в регистр загрузки нужно вводить дополнение требуемого коэффициента пересчета до полной емкости счетчика, а для вычитающего счетчика — сам коэффициент пересчета. Такие делители с изменяемым модулем пересчета используют при цифровом управлении частотой тиристорных преобразователей, скоростью шаговых двигателей, в качестве генераторов тонов в электромузыкальных инструментах.

Иногда необходим сигнал не в виде импульсов, а в виде меандра регулируемой частоты. Его легко получить пересчетом на T-триггере импульсов с выхода счетчика-делителя, а если основание пересчета есть целая степень двойки, то прямо с выхода последнего триггера.

Примерами микросхем делителей частоты с коэффициентом пересчета, управляемым цифровым кодом, являются программируемый счетчик 564ИЕ15 [11] и программируемый таймер КР580ВИ53 (см. [43]). Последняя микросхема содержит три счетчика и предназначена в основном для работы в составе микропроцессорных систем, поэтому управлять ею без микропроцессора довольно сложно. Максимальный коэффициент пересчета 564ИЕ15 — более 21 тыс., каждого счетчика КР580ВИ53 — более 65 тыс.

На рис. 9.11, *а* показана схема *дозатора* импульсов, выдающего по сигналу *ПУСК* на выходе *ПАЧКА* одиночную пачку, содержащую заданное число импульсов, вырезанных из непрерывной последовательности *С1*. Дозаторы используют, например, при обмене последовательным кодом, чтобы отмерять посылки определенного числа сигналов. В качестве счетчика, на базе которого построен дозатор, можно включить де-

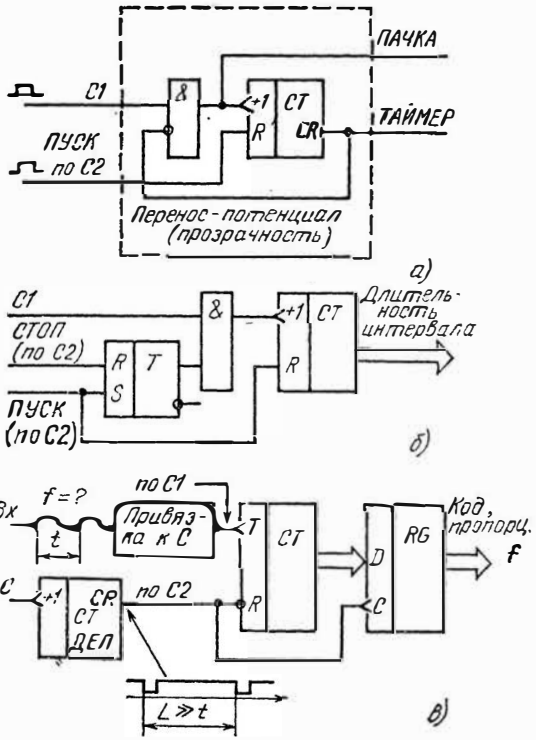


Рис. 9.11. Узлы цифровой автоматики на базе счетчиков:
а — дозатор импульсов и таймер; *б* — измеритель интервалов времени; *в* — цифровой частотомер

литель с изменяемым модулем пересчета, значение которого задается двоичным кодом. Если в таком дозаторе применен двоичный счетчик и если отсчитанную им пачку выходных импульсов подать на вход другого счетчика, но уже десятичного, то после окончания цикла пересчета число, введенное в дозатор в качестве модуля пересчета, окажется преобразованным в двоично-десятичный код, который можно снять параллельно с выходов триггеров десятичного счетчика. Если двоичный и десятичный счетчики поменять местами, то получится обратный преобразователь — двоично-десятичного кода в двоичный.

Выход *ТАЙМЕР* имеет низкий уровень все время, пока выдается пачка. Таймерный выход предназначен для получения выдержек времени при управлении различными объектами. Точность цифровых таймеров намного выше, чем аналоговых. Время выдержки, как и число импульсов в пачке дозатора, можно задавать цифровым кодом. Отмеченные выше микросхемы 564ИЕ15 и КР580ВИ53 могут использоваться и в режиме таймера.

На рис. 9.11, *б* показана схема измерения интервала времени между двумя сигналами, поступающими на входы *ПУСК* и *СТОП*. Содержимое счетчика, считываемое после сигнала *СТОП*, пропорционально длительности измеряемого интервала. Если исследуемый процесс по отношению к *С1*, *С2* асинхронный, то входы *ПУСК* и *СТОП* должны сначала пройти через схему синхронизатора. Заменив *RS*-триггер на триггер Шмитта, можно измерять длительность процесса, отображаемого двумя уровнями одного сигнала, например напряжение на фотодиоде от вспышки света. Измерение целого ряда величин сводится к измерению интервалов времени: период гармонического сигнала, время оборота вала, сдвиг фаз гармонических сигналов, дальность в схеме радиолокатора и т. п.

На рис. 9.11, *в* показан принцип построения цифрового частотомера. В течение измерительного интервала *L* счетчик подсчитывает входные сигналы измеряемой частоты f (период $t=1/f$). В конце каждого измерительного интервала получаемое в счетчике число переписывается в регистр, содержимое которого в некотором масштабе отображает частоту входных сигналов. Длительность интервала *L*, отмеряемого делителем *СТ ДЕЛ*, во много раз больше t, чтобы обеспечить необходимую точность измерения. Входные сигналы пропускаются через схему синхронизатора.

На рис. 9.12, *а* показана схема, которую в зависимости от области применения называют *интерполятором*, цифро-частотным умножителем, цифро-частотным интегратором. Основу схемы составляет счетчик. На рисунке показан синхронный счетчик с параллельным переносом на синхронных триггерах. Если, имея двоичный счетчик, образовать несколько конъюнкций, в каждую из которых входят инверсное значение данного разряда $\overline{Q}_i$ и прямые значения всех более младших разрядов

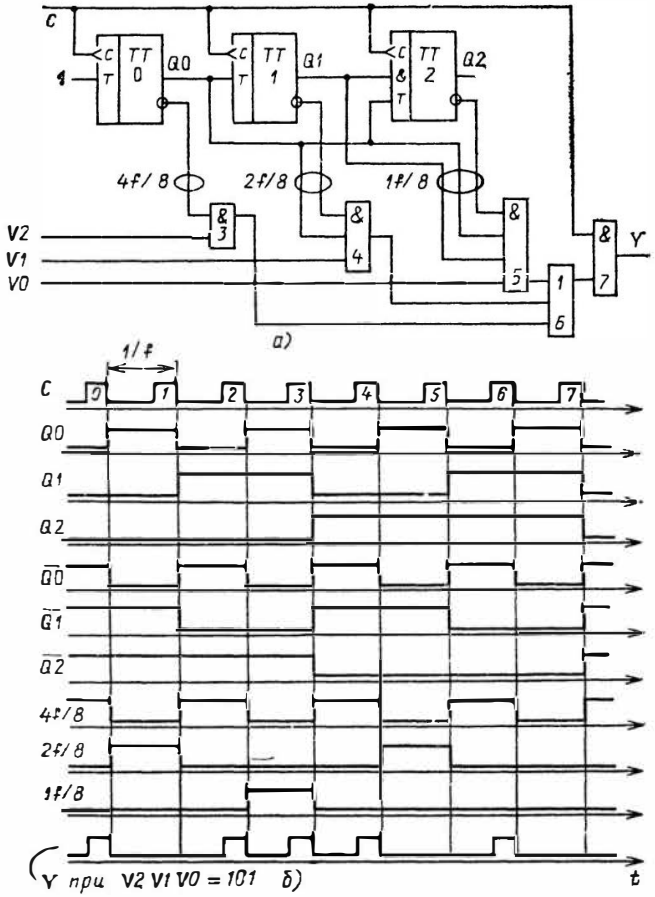


Рис. 9.12. Интерполятор, или частотный умножитель

(эти конъюнкции на рис. 9.12 помечены как *1f/8, 2f/8, 4f/8*), то, как видно на рис. 9.12, *б*, при пересчете счетчиком синхроимпульсов *C*, имеющих частоту *f*, частоты этих конъюнкций будут по мере продвижения от младших разрядов к старшим ступенчато уменьшаться вдвое. При этом единичные значения полученных сигналов *1f/8, 2f/8, 4f/8* никогда не будут совпадать друг с другом во времени, что и видно на временно́й диаграмме рис. 9.12, *б*. Поэтому в выходном напряжении дизъюнктора *6* каждый частотный компонент будет представлен самостоятельно независимо от наличия или отсутствия других. Это позволяет, пропуская или не пропуская через конъюнкторы *3*, *4* и *5* каждый компо-

нент *4f/8*, *2f/8*, *1f/8* в соответствии со значениями входных сигналов *V2*, *V1*, *V0*, пропускать через элемент *7* на выход *Y* за каждый цикл счетчика число синхроимпульсов, равное двоичному коду *V2V1V0*. На рис. 9.12, *б* показана выходная последовательность *Y* для кода *V2V1V0*=101.

Число импульсов, соответствующее управляющему двоичному коду, обеспечивает на своем выходе и обычный дозатор. Особенностью именно интерполятора является то, что он выдает выходные импульсы не компактной пачкой, а как-то распределяя их по временно́му интервалу своего цикла. Интерполятор применяют для управления шаговым двигателем, когда его скорость задана двоичным кодом *V2V1V0*, эта же схема используется для умножения величины, представленной частотой импульсов *C*, на другую величину, представленную двоичным кодом *V*, при этом результат в некотором масштабе отображается частотой выходных импульсов. Далее, если на вход *V* поступает в двоичном коде некоторая функция $F(x)$, а на вход *C* — приращения Δx аргумента этой функции (пусть, например, единичный шаг Δx для схемы на рис. 9.12, *а* отображается пачкой из 8 импульсов), то число импульсов на выходе будет пропорционально интегралу $\int F(x)dx$.

Следует обратить внимание на то, что в общем случае для произвольных кодов *V2V1V0* выходные импульсы интерполятора распределены во времени неравномерно, как это, в частности, видно и на рис. 9.12, *б*, поэтому далеко не для всех применений выходную последовательность допустимо трактовать как «последовательность импульсов, частота которой пропорциональна двоичному коду», что иногда встречается в литературе. Термин «частота» здесь используется в весьма нестрогом значении. Влияние неравномерности следования импульсов интерполятора на управляемый объект должно быть обязательно оценено. Примером микросхемы интерполятора может служить схема К155ИЕ8, имеющая 6-разрядный счетчик, 6-разрядный код *V*, несколько вспомогательных разрешающих и управляющих входов и выход для наращивания разрядности.

ГЛАВА 10

УЗЛЫ НА БАЗЕ СДВИГАЮЩИХ РЕГИСТРОВ

10.1. Сдвигающие регистры

Сдвигающий, или *сдвиговый*, *регистр* (*shift register*) это регистр, содержимое которого при подаче управляющего сигнала *СДВИГ* может сдвигаться в сторону старших или

младших разрядов. Схема сдвигающего регистра показана на рис. 10.1, *а*, а условное обозначение — на рис. 10.1, *б*. Регистр состоит из цепочки непрозрачных триггеров. Пусть на рисунке триггер *ТТ0* — младший, *ТТ3* — старший; *D*-вход каждого триггера (кроме *ТТ0*) подключен к выходу соседнего младшего триггера. Когда на все объединенные *С*-вхо-

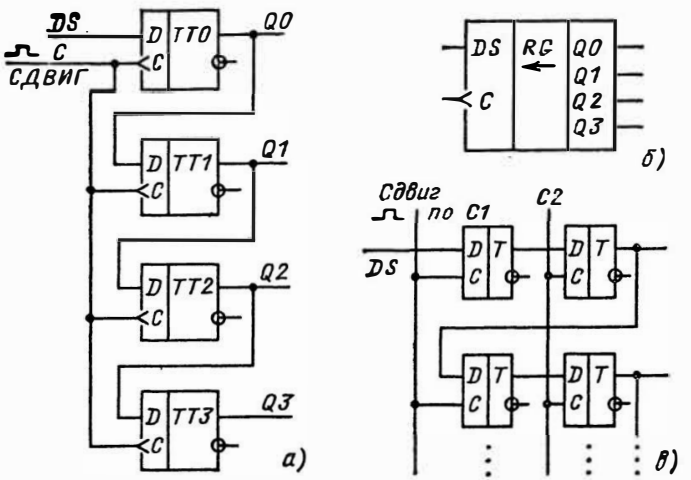


Рис. 10.1. Сдвигающий регистр:

а — схема на непрозрачных триггерах; *б* — условное обозначение; *в* — схема на прозрачных защелках

ды триггеров поступает активный отрицательный фронт сигнала *СДВИГ*, выход каждого триггера принимает состояние своего младшего соседа и, таким образом, информация, содержащаяся в регистре, сдвигается на один разряд в сторону старших разрядов, *влево*. Триггер *ТТ0* принимает при этом состояние *последовательного входа DS* (*data serial*). Регистр загружается данными, последовательно поступающими по этому входу. Считывать данные, хранимые в регистре, можно как в последовательном коде, с выхода последнего разряда, так и в параллельном, сразу со всех разрядов. По схеме, близкой к показанной на рис. 10.1, *а*, построен регистр 564ИР2.

Сдвигающий регистр — это простейший узел с однофазной синхронизацией. Роль синхроимпульса играет импульс *СДВИГ*, КС представлена просто связями с выходов одних триггеров на входы других. В однофазной схеме, изобра-

женной на рис. 10.1, *а*, существенно, что использованы именно непрозрачные триггеры. Если на их место поставить прозрачные защелки, то при активном уровне сигнала *СДВИГ* все триггеры становятся прозрачными, а уровень, например, входа *DS* успеет пройти сквозь столько триггеров, сколько позволит длительность сигнала *СДВИГ*. Процесс передачи информации окажется во власти соотношений задержек различных элементов. Сдвигающий регистр, построенный на защелках, должен иметь двухфазную систему синхронизации, как на рис. 10.1, *в*.

На рис. 10.2, *а* показано условное обозначение более сложного, *двунаправленного (bidirectional)* сдвигающего

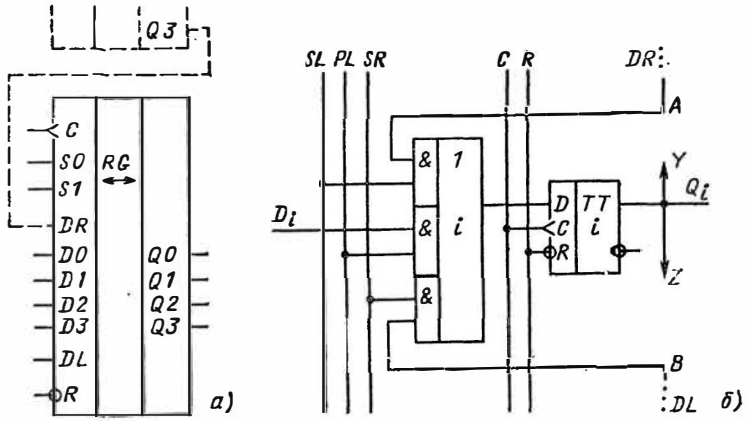


Рис. 10.2. Условное обозначение *(а)* двунаправленного сдвигающего регистра и схема одного разряда *(б)*

регистра, способного сдвигать содержимое и влево, и вправо. При двунаправленном сдвиге появляются два последовательных входа: вход, принимающий данные, вдвигаемые в регистр со стороны младшего разряда справа, — *DR* (*data right*) и вход со стороны старшего разряда слева — *DL* (*data left*). Эти входы используются и при наращивании регистра: *DR* подключается к выходу старшего разряда соседней младшей секции общего регистра (на рис. 10.2, *а* показано штриховой линией); *DL* подключается к выходу *Q0* младшего разряда соседней старшей секции общего регистра. Кроме последовательных входов сдвигающие регистры часто имеют параллельные входы (*D0—D3* на рис.

10.2, *а*), с помощью которых регистр можно загрузить параллельным кодом сразу за один такт.

Схема одного, *i*-го разряда двунаправленного сдвигающего регистра с параллельной загрузкой показана на рис. 10.2, *б*. К *D*-входу каждого триггера регистра подключен мультиплексор, который при подаче 1 на один из управляющих входов — *SL (shift left)*, *SR (shift right)* или *PL (parallel load)* подключает вход *i*-го триггера соответственно к выходу триггера младшего соседа (направление *A*), старшего соседа (*B*) или ко входу параллельной загрузки D_i. Точка *A* самого младшего разряда является входом *DR* всего регистра, точка *B* самого старшего разряда — входом *DL*. Выход *i*-го триггера подключен к соответствующим входам мультиплексоров соседних разрядов (направления *Y* и *Z*). По *C*-сигналам триггеры регистра принимают информацию с направлений, диктуемых мультиплексорами.

Часто для более экономного использования выводов микросхемы управляющие входы *SL*, *SR*, *PL* не выводят из корпуса непосредственно, а управляют ими через небольшой дешифратор режимов. Такое решение подразумевается на рис. 10.2, *а*, где *S1*, *S0* — входы этого дешифратора управления режимом. В разных микросхемах кодировка режимов различна. В частности, 8-разрядный регистр К155ИР13 при коде *S1S0*=00 находится в режиме хранения, поскольку нулевой выход дешифратора режимов перекрывает тракт *C*-сигналов; при *S1S0*=01 по *C*-сигналам выполняется сдвиг вправо, при *S1S0*=10 — влево, и при *S1S0*=11 — прием параллельного кода с *D*-входов. В однонаправленных сдвиговых регистрах код режима может быть одноразрядным: 0 — сдвиг, 1 — параллельная загрузка. Так, в частности, сделано в 4-разрядных регистрах К561ИР9 и К155ИР1. Последняя микросхема имеет два импульсных входа, один — для сдвига, другой — для параллельной загрузки, но каждый раз реагирует лишь на один из них в зависимости от значения сигнала на входе управления режимом. Режим хранения в регистрах с одноразрядным кодом режима можно обеспечить, прерывая с помощью внешнего конъюнктора поток *C*-импульсов.

Сдвигающие регистры могут иметь вход *общего сброса (master reset)* $\bar{R}$, который часто выполняют по схемам, показанным на рис. 6.16, т. е. приоритетным по отношению к входам *C*, *D* и входам режима. Иногда приоритетные асинхронные *RS*-входы триггеров регистра используются не для общего сброса, а для параллельной загрузки его, что

делает операцию загрузки приоритетной по отношению к операции сдвига. Для однонаправленных регистров в этом случае не нужны ни вход переключения режима, ни управляемые им мультиплексоры на входах триггеров. С рядом схем сдвиговых регистров различных серий можно ознакомиться по [11, 28].

Применения сдвигающих регистров очень разнообразны. В арифметике сдвиг числа на один разряд влево соответствует умножению его на 2, сдвиг вправо — делению пополам. Сдвигающий регистр, содержащий всего одну единицу, может выполнять роль счетчика, отображающего число поступивших на вход сигналов положением единицы на линейной шкале (например, горящая лампочка номера этажа в лифте). Если на вход сдвига такого счетчика подать импульсы тактового генератора, то временна́я диаграмма сигналов на его выходах будет, по сути, близка к диаграмме сигналов многофазовой синхронизации, показанной на рис. 7.2, *г*, т.е. сдвигающий регистр может быть использован для распределения опорных тактовых импульсов по нескольким фазам синхронизации. В системах радиосвязи и радиолокации сдвигающие регистры применяют для построения радиозамков, двоичных корреляторов и других устройств более сложной обработки радиосигналов.

Сдвигающие регистры преобразуют параллельный код в последовательный и обратно, как это показано на рис. 10.3. Выходной регистр *RG1* некоторого блока передает

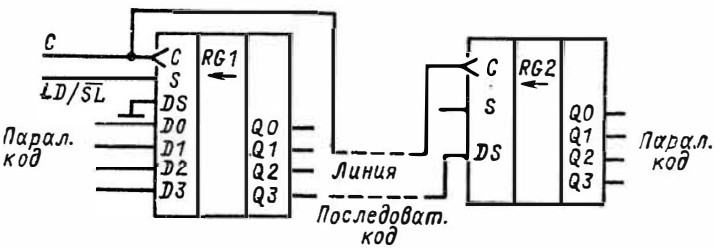


Рис. 10.3. Преобразование параллельного кода в последовательный

данные в линию. Входной регистр *RG2* другого блока принимает их. При соответствующем значении сигнала управления режимом данные параллельным кодом загружаются через *D*-входы в регистр *RG1*. Затем и *RG1*, и *RG2* переводятся в режим сдвига и на их *C*-входы подается серия из четырех (в данном случае) *C*-импульсов. При этом со-

держимое регистра-передатчика разряд за разрядом появляется на выходе *Q3*, последовательным кодом передается по линии и через вход *DS* вдвигается в регистр-приемник. После этого переданные данные могут быть считаны параллельным кодом с выходов *Q0—Q3* регистра *2*. Следует обратить внимание на то, что в приемник нужно передавать не только последовательный код данных, но и синхроимпульсы, необходимые для управления сдвигом на приемной стороне. О методах борьбы с возможной расфазировкой синхронизирующего и информационного сигналов говорилось в § 7.3.

Существуют и *асинхронные* способы последовательной передачи данных, при которых синхронизирующий сигнал не идет по отдельному каналу, а за счет специального кодирования передается по тому же каналу, что и данные, перемежаясь с ними во времени. Канал связи экономится за счет снижения скорости информационного обмена и усложнения приемо-передающей аппаратуры. Чаще всего используются два способа такой однопроводной передачи. Первый — когда синхронизирующий перепад сопровождает каждый бит информации, это *фазовое* и *частотное* кодирование (см. [30], раздел «Методы записи на магнитный носитель»). Второй — когда синхронизирующий перепад, называемый стартовым битом, сопровождает цепочку из 5—8 бит (чаще всего байт). Это *стандартный последовательный интерфейс*, о котором можно прочесть в [3]. Для выполнения функций как приемника, так и передатчика при обмене в стандартном последовательном формате выпускаются специальные микросхемы, например КР580ВВ51 (см. [43]), или КР581ВА1. Если для управления обменом не используется микропроцессор, то последняя микросхема удобнее.

Передача информации последовательным кодом по сравнению с передачей параллельным существенно экономит число линий связи. Это покупается ценой увеличения времени обмена, однако в случае механических и тем более тепловых процессов возникающие при этом задержки, как правило, вполне допустимы. В то же время существующее соотношение стоимости и массогабаритных показателей кабелей, с одной стороны, и аппаратуры преобразователей — с другой таково, что при разрядности в 1—2 байт передача последовательным кодом уже при расстояниях в несколько метров становится выгоднее передачи параллельным кодом. Этот факт иногда выпадает из поля зрения разработчиков цифровой автоматики.

Много обещает применение сдвиговых регистров в системах контроля цифровой аппаратуры, особенно БИС. Проблема здесь в том, что в силу технических ограничений

на число внешних выводов кристалла из корпуса БИС невозможно вывести все желаемые контрольные точки схемы, т. е. те точки, на которые можно задавать поверочные (тестовые) сигналы, и точки, в которых можно наблюдать результаты этих воздействий. А при использовании только входных и выходных выводов БИС тесты оказываются чрезмерно длинными. То же, правда, в меньшей степени, относится и к ТЭЗам, где часто не хватает контактов разъема. В связи с этим появилось решение: в качестве контрольных точек применить триггеры — как штатные, выполняющие в схеме рабочие функции, так и специально вводимые для облегчения контроля интересующих схемотехника точек. Соединив все триггеры в один общий сдвиговый регистр, можно с помощью всего двух выводов последовательным кодом вводить и выводить информацию, связанную с контрольными точками.

Триггеры при такой системе контроля требуются несколько более сложные, чем обычные синхронные. Фрагмент схемы, включающей два таких триггера, показан на рис. 10.4. Каждый триггер, обведенный на рисунке штри-

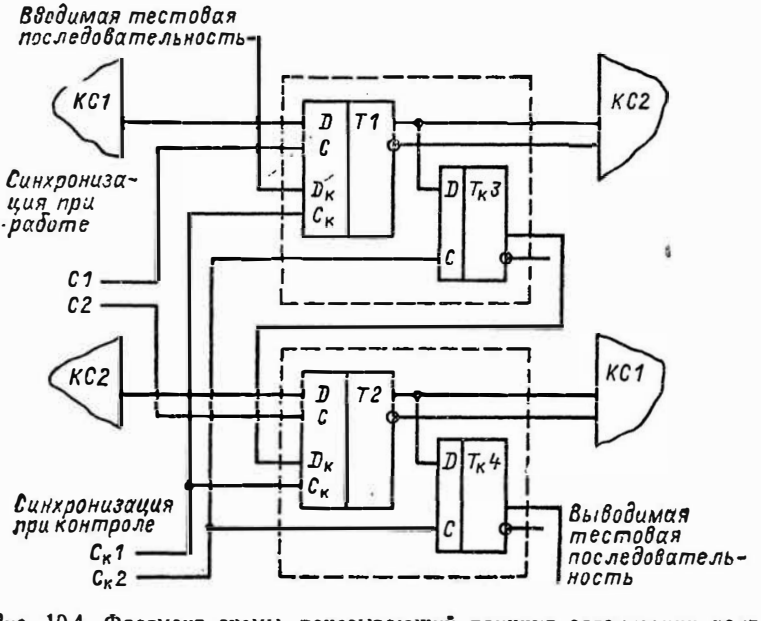


Рис. 10.4. Фрагмент схемы, показывающий принцип организации контроля методом сквозного сдвигового регистра

ховой линией, — двухступенчатый, состоит из двух прозрачных защелок. Первую ступень можно условно назвать рабочей, вторую — контрольной. Вторая ступень — это обычная защелка, а первая — защелка более сложная. Она имеет две пары независимо работающих входов: D, C и $D_к$, $C_к$. Первая пара входов — D и C — используется стандартным образом во время выполнения триггером своих рабочих функций в схеме: при поступлении тактирующих сигналов на C-вход триггер по своему D-входу принимает данные с одной КС и передает их на другую КС, как и подобает триггеру макрорегистра *MRG*.

Вторая пара входов — $D_к$ и $C_к$ — предназначена для режима контроля. Вход $D_к$ каждого двухступенчатого триггера (например, защелки *T2*) подключен к выходу другого триггера (на рисунке — к выходу $T_к3$) так, что все двухступенчатые триггеры образуют один большой сдвиговый регистр, информация в котором сдвигается при поочередной подаче сигналов на входы $C_к1$ и $C_к2$. Сигналы рабочей синхросерии по входам C при этом не подаются. Таким образом, вторые ступени триггеров (на рисунке — защелки $T_к3$ и $T_к4$) функционируют только в режиме *СДВИГ* при контроле.

Начало и конец единого сдвигового регистра подключаются к внешним выводам микросхемы (или к контактам разъема, если речь идет о ТЭЗе). Наружу выводятся и общие цепи $C_к1$ и $C_к2$ сигналов сдвига, а также, если этого требует конкретная схема сложного триггера, управляющий вывод, открывающий одну или другую пару входов триггера — D и C или $D_к$ и $C_к$, т. е. вход, задающий его режим: *РАБОТА* или *СДВИГ* (для схемы по рис. 10.4 этого входа не требуется).

Процедуру контроля исправности схемы в упрощенном виде можно представить так. Отключив рабочие синхросигналы *C1* и *C2* и подавая столько пар сдвигающих сигналов $C_к1$ и $C_к2$, сколько всего в сдвигающем регистре триггеров, вводят в этот регистр последовательным кодом тестовый набор, предназначенный для проверки правильности выполнения каких-то операций комбинационной схемой. Методы синтеза таких наборов изложены в [32, 60]. После заполнения сдвигового регистра подается одиночный импульс одной из рабочих синхросерий, например *C1*. По нему результат обработки комбинационной схемой тестового набора принимается в макрорегистр *MRG1*, т. е. в первые ступени тех триггеров, которые синхронизируются фазой *C1* (на

рис. 10.4 — в защелку *T1*). После этого дается еще одна серия сдвиговых сигналов $C_к1$—$C_к2$, которая последовательным кодом выводит наружу содержимое всего сдвигового регистра (для этого серия начинается с $C_к2$). Если после ввода теста подать одиночный сигнал *C2* (вместо *C1*), то будет проверена другая половина комбинационной схемы — схема *KC2*, а результат будет принят в триггеры *MRGII* (на рисунке — *T2*). Пропустив таким образом сквозь испытуемую схему требуемое число необходимых тестовых наборов и сравнивая каждый раз фактический код, выведенный из сдвигового регистра, с тем, который должен быть при исправной работе КС и триггеров, можно судить о правильности работы схемы и даже о характере повреждения. Разумеется, ввод, вывод и сверка кодов выполняются автоматически специальным контролирующим блоком.

Изложенный метод построения блоков с *высокой контролепригодностью* сочетает в себе два важных достоинства: возможность добраться до любой точки схемы и малое число требуемых для этого выводов. Объем дополнительной аппаратуры обычно оценивается значением 15 %. Метод был назван *Level-sensitive scan design,* сокращенно *LSSD*. В отечественной терминологии используют названия *Метод сквозного сдвигового регистра* и *Метод сканирования* [44, 45, 60].

Сдвиговые регистры с большим числом разрядов используются для построения *памяти на последовательных регистрах*. В такой памяти сигналы сдвига подаются на регистр непрерывно от синхрогенератора, а выход сдвигающего регистра замыкают кольцом на его последовательный вход *DS*. В результате однажды записанный в регистр код будет в этом кольце циркулировать бесконечно долго.

Поставив параллельно m одинаковых сдвиговых регистров и подавая сигналы сдвига от общего генератора, можно записывать и считывать m-разрядные слова параллельным кодом.

Поскольку появления требуемого слова приходится ждать, память значительной емкости на последовательных регистрах характеризуется довольно большим временем обращения. Однако если это не является помехой, то реализуются серьезные ее достоинства — малые аппаратурные затраты и низкая стоимость, поскольку методами интегральной технологии ряд типов сдвигающих регистров получить легче, чем эквивалентные по емкости наборы триг-

геров статической памяти. Примером микросхем сдвигающих регистров небольшой емкости, называемых также (особенно у радистов) *цифровыми линиями задержки,* является 144ИР3 емкостью 64 бит. Память на приборах с зарядовой связью (ПЗС) и на цилиндрических магнитных доменах (ЦМД) также организована по принципу сдвигающих регистров, имеющих емкость в десятки и сотни тысяч бит на кристалл. Подробнее с памятью на последовательных регистрах можно ознакомиться по [20].

10.2. Кольцевые распределители

Распределителями называют функциональные узлы, распределяющие поток импульсов последовательно, импульс за импульсом, по нескольким выходам так, что результирующая временна́я диаграмма имеет вид, как на рис. 10.5, *б*. Распределитель можно рассматривать как одну из форм счетчика, некоторые авторы их так и называют. Распределители могут строиться по различным схемам. Например, функции распределителя выполняет двоичный счетчик с дешифратором, подключенным к его выходам Q_i. Счетчик последовательно перебирает двоичные коды, дешифратор в соответствии с ними последовательно возбуждает свои выходы. При этом нужно помнить, что дешифратор будет расшифровывать не только верные, установившиеся состояния счетчика, но, настолько, насколько он успеет, и неустановившиеся, ложные. В результате на выходах такого распределителя кроме нормальных сигналов сразу после активных фронтов в произвольных местах будут появляться короткие всплески напряжения — *иголки,* ширина которых определяется запаздыванием срабатывания триггеров относительно друг друга. Иголок нет вовсе у распределителей на синхронных счетчиках, и они пренебрежимо малы (или их практически нет) при использовании счетчиков с параллельным переносом. Если межразрядный или межгрупповой перенос последовательный, то нужно или иметь уверенность, что управляемое распределителем устройство реагировать на иголки не будет (например, механический шаговый двигатель), либо исключить их с помощью синхронизируемого регистра, как исключаются помехи переходных процессов любых комбинационных схем. Роль распределителя, как уже отмечалось, может играть и сдвиговый регистр с единственной двигающейся единицей.

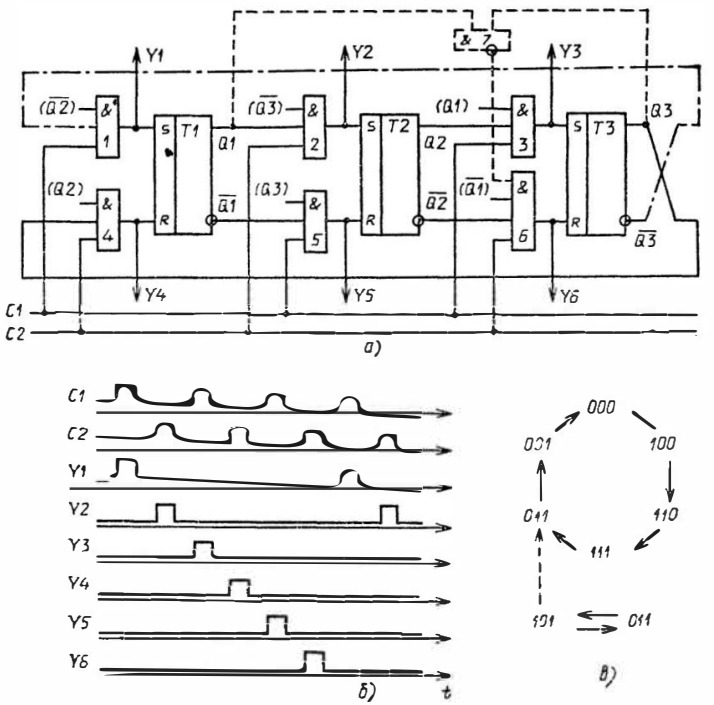


Рис. 10.5. Кольцевой распределитель с нечетным числом триггеров: *а* — схема; *б* — временная диаграмма; *в* — диаграмма переходов

При разработке интегральных схем и при работе на россыпи более экономичными, чем регистры, оказываются специальные схемы *кольцевых распределителей,* которые еще называют *счетчиками Джонсона,* кольцевыми счетчиками, *счетчиками Мёбиуса, счетчиками в коде Либау — Крейга, кольцами Реженера.*

Схема кольцевого распределителя с нечетным числом триггеров показана на рис. 10.5, *а*. Кольцо может содержать любое нечетное число *RS*-триггеров. Реально вместо конъюнкторов, помеченных на рисунке номерами от *1* до *6*, и триггеров *RS* используются элементы И-НЕ и триггеры $\overline{R}\overline{S}$, но это несколько затрудняет понимание схемы, хотя принципа работы не меняет. На элемент *7* и штриховые связи обращать внимания не нужно, полагая пока, что их нет вовсе. Кроме того, сначала нужно учитывать лишь те входные сигналы элементов *1—6*, которые поступают по

Таблица 10.1

Состояние	$Q1$	$Q2$	$Q3$	Состояние	$Q1$	$Q2$	$Q3$
0	0	0	0	4	0	1	1
1	1	0	0	5	0	0	1
2	1	1	0	0	0	0	0
3	1	1	1				

начерченным на рисунке линиям связи. О входных сигналах, обозначенных $(Q1) — (Q3)$, $(\overline{Q1}) — (\overline{Q3})$, речь будет идти позже. Последовательность кодов Либау — Крейга, реализуемая счетчиком, приведена в табл. 10.1. Регистр сначала последовательно заполняется единицами, а затем благодаря перекрестной обратной связи (аналогичной склейке ленты Мёбиуса) начинает заполняться нулями. Число состояний равно удвоенному числу триггеров.

Синхросигналы двух фаз заведены так, что обеспечивают последовательное переключение триггеров без проскоков информации через каскад. Если работа схемы не очевидна, полезно самостоятельно построить временные диаграммы сигналов на выходах конъюнкторов и триггеров. Для того чтобы схема стала распределителем, нужно расшифровать последовательные состояния счетчика так, чтобы каждому его состоянию соответствовал только один возбужденный выход. При этом дешифратор кода Либау — Крейга в отличие от дешифратора двоичного кода оказывается очень простым. При смене состояний счетчика по цепочке триггеров движется характерная точка — граница сплошных единиц и сплошных нулей, с которой легко связать состояние возбужденного выхода дешифратора. Для этого достаточно на каждый j-й элемент И дешифратора подать левую по отношению к нему 1 и правый 0, т. е. прямой выход соседнего слева триггера и инверсный выход соседнего справа. На верхнюю по рисунку линейку входных конъюнкторов с номерами j от *1* до *3* прямые выходы предыдущих триггеров уже поданы, и для дешифрации состояний остается подать лишь инверсные выходы правых соседей. Предназначенные для этого входы конъюнкторов обозначены символами: $(\overline{Q2})$, $(\overline{Q3})$, $(Q1)$. Аналогично на второй половине кодового набора при заполнении счетчика нулями дешифрирующими входами нижнего ряда конъюнкторов являются $(Q2)$, $(Q3)$, $(\overline{Q1})$.

Если разорвать обратную связь с инверсного выхода *Т3* на входной конъюнктор *1* (штрихпунктирная линия), то кольцо, пройдя полный цикл, остановится. При подаче на конъюнктор *1* запускающего сигнала кольцо снова пройдет один цикл. Это *разомкнутый режим* работы. В *замкнутом режиме* кольцо непрерывно генерирует последовательность, показанную на рис. 10.5, *б*.

На рис. 10.5, *в* последовательность генерируемых кодов представлена не в виде таблицы, а в виде *диаграммы переходов*. Из восьми возможных комбинаций состояний трех триггеров в схеме распределителя используются только шесть, остальные два не используются и при нормальной работе никогда не возникают. Однако при включении питания или в результате воздействия помехи в счетчике может возникнуть один из неиспользуемых, запрещенных кодов, показанных в нижней части диаграммы переходов. Работа счетчика из трех триггеров на этих кодах сопровождается гоночными процессами, и счетчик может самопроизвольно выйти на коды рабочего цикла, однако при некоторых сочетаниях задержек он так и останется в ложном цикле *2—5—2—5—...*

С увеличением длины счетчика число неиспользуемых кодов быстро растет, многие из них гонок не порождают и образуют устойчивые циклы. Если в разомкнутом кольце ложное состояние существует всего один цикл, то в замкнутом оно останется навсегда. Поэтому в кольца, предназначенные для работы в цикле, вводят дополнительные логические элементы, обнаруживающие одну из комбинаций запрещенного цикла. Выходные сигналы этих элементов принудительно устанавливают в кольце одну из комбинаций рабочего цикла. На рис. 10.5, *а* эту функцию выполняют конъюнктор *7* и его штриховые связи. Он обнаруживает запрещенный код 101 и, не давая триггеру *Т3* переключиться в 0, переводит стремящуюся возникнуть ложную комбинацию 010 в рабочую 011. Этот переход показан штриховой линией на диаграмме рис. 10.5, *в*. В счетчиках с бо́льшим числом триггеров задача блокировки имеет множество решений, а из части ложных состояний счетчик выходит самостоятельно. Для каждого конкретного числа триггеров эти вопросы требуют специального рассмотрения.

На рис. 10.6 показан другой вариант кольцевого распределителя, отличающийся от предыдущего тем, что содержит четное число триггеров. Поэтому разводка цепей

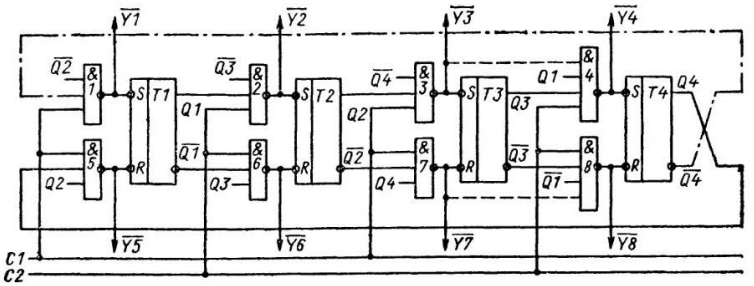


Рис. 10.6. Кольцевой распределитель с четным числом триггеров

синхронизации у него несколько иная. Схема показана в исполнении на элементах И-НЕ, а по всем остальным свойствам она идентична схеме, данной на рис. 10.5, *а*.

Для распределения импульсов однофазного источника эти импульсы перед подачей на кольцевой счетчик пересчитывают на триггере, чтобы сформировать серию *C1*, *C2*. При этом может появиться перекрытие во времени единичных значений полученных сигналов *C1* и *C2*, поскольку плечи триггера переключаются последовательно. Перекрытие *C1* и *C2* приводит к проскокам единиц сразу сквозь несколько каскадов в зависимости от величины перекрытия. Для защиты можно ввести в кольцо дополнительные противогоночные связи. Примеры таких связей показаны на рис. 10.6 штрихпунктирными линиями. Даже при перекрытии конца *C1* с началом *C2* конъюнктор *4* или *8* не сможет открыться ранее, чем исчезнут активные низкие уровни выходных сигналов $\overline{Y3}$ или $\overline{Y7}$, что произойдет лишь после исчезновения синхросигнала *C1*. Аналогичные связи нужно ввести между всеми каскадами. С выходов конъюнкторов последнего триггера *T4* на конъюнкторы первого *T1* эти связи заводятся перекрестно. Аналогичные связи можно вводить и в схему по рис. 10.5.

Кольцевой распределитель в коде Либау — Крейга можно построить и с однофазной синхронизацией. В этом случае его роль выполняет однотактный сдвигающий регистр на непрозрачных *D*-триггерах с перекрещенной циклической связью: с инверсного выхода $\overline{Q}$ последнего триггера на последовательный вход *DS* первого. Дешифраторы выходов строятся по описанному принципу объединением на двухвходовых элементах И прямого выхода данного триггера и инверсного выхода соседнего. По числу элементов такой распределитель почти вдвое экономичнее однотактного сдвигающего регистра с одной движущейся единицей, но

почти вдвое дороже рассмотренных выше колец с двухтактной синхронизацией, поскольку число элементов у непрозрачного триггера примерно вдвое больше, чем у синхронного *RS*-триггера. На однотактных непрозрачных *D*-триггерах построены, например, десятичный распределитель К561ИЕ8 и восьмеричный К561ИЕ9.

Кольцевые распределители в отличие от распределителей по схеме двоичный счетчик — дешифратор, имеют очень малую задержку распространения по тракту вход *C* — выход *Y*, а при малом числе разрядов могут конкурировать со счетчиками других типов также и по затратам оборудования. К тому же структура их регулярна и поэтому удобна для реализации средствами интегральной технологии. Кольцевые распределители применяют для формирования различных многофазных серий сигналов — для управления, например, шаговыми двигателями, сдвигающими регистрами на ПЗС и т. п. С помощью разомкнутых распределителей формируются *K*-фазные синхросерии типа показанной на рис. 7.2, *г*. При этом из опорного сигнала задающего генератора получают для кольцевого распределителя синхросигналы, а сигнал начала первой фазы в качестве фазирующего сигнала подают на конъюнктор *S*-входа первого триггера разомкнутого распределителя (т. е. распределителя без штрихпунктирных связей по рис. 10.5, *а* или 10.6). Модификации, а также практические схемы распределителей рассмотрены в [11 и 36].

10.3. Полиномиальные счетчики

Полиномиальными счетчиками или (реже) *линейными последовательностными машинами* (не последовательными!) называют пересчетные схемы, которые получаются из сдвигающих регистров при введении в них обратных связей через сумматоры по модулю 2. Регистры при этом строят на непрозрачных *D*-триггерах. На рис. 10.7, *а* показан один из вариантов такого счетчика, а в табл. 10.2 приведена последовательность смены его состояний. При нулевых состояниях всех триггеров и нуле на входе схема не реагирует на поступающие синхросигналы *C*, оставаясь в нулевом состоянии. При подаче на вход одной единицы по очередному *C*-сигналу счетчик переходит в состояние 100, и далее, уже при нулевом входном сигнале, в каждый такт происходит смена состояний по некоторому закону. Схема по рис. 10.7, *а* входит в цикл и генерирует последовательность

состояний в соответствии с табл. 10.2, пока все триггеры не будут сброшены по входу общего гашения (на схеме не показан). В общем случае длина и вид генерируемой счетчиком последовательности зависят от числа триггеров сдвигающего регистра и от того, между какими разрядами заведены обратные связи.

На рис. 10.7, *б* приведена схема счетчика другой конфигурации, в которой использован всего один, но многовходовой сумматор по модулю 2. Последовательность состояний счетчика представлена в табл. 10.3. На рис. 10.7, *в* показана обобщенная схема полиномиального счетчика этой конфигурации. Цепи синхронизации и обозначения типов триггеров при этом опускаются, поскольку для всех подобных счетчиков они одинаковы. Внимание обращается на число триггеров регистра и закон заведения обратной связи. Оба эти параметра характеризуются *вектором*, или *кодом обратной связи* $q_1, q_2, q_3 \ldots q_n$, где q_i равно единице, если выход i-го триггера заведен на вход *M2*. Вектор обратной связи представляет собой n-разрядное число, однозначно определяющее вид генерируемой последовательности. Счетчик по рис. 10.7, *б* описывается вектором $q_1q_2q_3 = 101$.

Особый интерес представляют полиномиальные счетчики, генерирующие *последовательности максимальной длины*, в состав которых входят (2^n-1) комбинаций значений n разрядов, т. е. все комбинации, кроме одной — начальной и устойчивой (не изменяющейся при подаче на регистр *С*-сигналов): *Все нули.* Схемы на рис. 10.7, *а* и *б*

Таблица 10.2

Такт	Вход	Q_1	Q_2	Q_3
	0	0	0	0
	:	:	:	:
	0	0	0	0
0	1	0	0	0
→1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	1	1	0
5	0	0	1	1
6	0	1	1	1
—7	0	1	0	1
	0	1	0	0

случаях, особенно когда наряду с необходимостью иметь случайные числа требуется еще иметь и повторяемые результаты, т. е. последовательность «случайных» чисел не должна быть совершенно случайной. Цифровой шум, порождаемый генератором, можно превратить в аналоговый, сгладив однобитовую цифровую последовательность на фильтре нижних частот, частота среза которого мала по сравнению с тактовой частотой регистра. Более высокочастотный шумовой генератор получится, если с датчика псевдослучайных чисел снять параллельный код, и, перемешав разряды, пропустить его через цифро-аналоговый преобразователь.

10.4. Кодеры и фильтры циклических кодов

Термин «полиномиальные» для обозначения узлов, построенных на базе сдвигающих регистров и сумматоров по модулю 2, отражает способность этих узлов аппаратно реализовывать ряд операций над полиномами (многочленами), имеющими двоичные (бинарные) коэффициенты при степенях переменной x. Эти операции позволяют построить целый класс *циклических кодов*, которые широко используются для обнаружения или исправления ошибок при передаче и хранении данных.

Некоторый полином с бинарными коэффициентами (т. е. с коэффициентами, которые могут принимать значения лишь 0 или 1), например

$$A = 1 + x^3,$$

можно записать в более развернутой форме

$$A = 1 \cdot x^0 + 0 \cdot x^1 + 0 \cdot x^2 + 1 \cdot x^3 + 0 \cdot x^4.$$

В отличие от обычной алгебры здесь старшие степени x удобнее писать справа. Такая запись позволяет поставить в соответствие полиному *код* значений коэффициентов при степенях x: 10010. Тогда умножению полинома на x будет соответствовать сдвиг его *кодового эквивалента* вправо

$$A \cdot x = x + x^4 = 0 \cdot x^0 + 1 \cdot x^1 + 0 \cdot x^2 + 0 \cdot x^3 + 1 \cdot x^4.$$

Кодовый эквивалент $A \cdot x$ равен 01001, т. е. коду исходного полинома A, равного 10010, но сдвинутому вправо. Далее над бинарными полиномами вводится операция, которую из-за сходства с операцией алгебраического перемножения также называют *перемножением полиномов*. При этом показатели степеней x перемножаемых членов складывают, как обычно, по арифметическим правилам, но коэффициенты

при получившихся одинаковых степенях *x суммируются по модулю 2.* Например:

$$K = A \cdot G = (1 + x^3) \cdot (1 + x + x^3) = 1 + x + x^4 + x^6. \qquad (10.1)$$

Или более детально, выделив для каждой степени x отдельный столбец,

$$\begin{array}{lllllll}
A & & 1 & + & x^3 & \text{(полином—множимое)} \\
G & & 1 + x & + & x^3 & \text{(полином—множитель)} \\
\hline
A \cdot 1 = & 1 & & + & x^3 & \\
A \cdot x = & & x & & + x^4 & \\
A \cdot x^3 = & & & x^3 & & + x^6 \\
\hline
A * G = & 1 + & x + & 0 \cdot x^3 + & x^4 & + x^6
\end{array} \quad \oplus \qquad (10.2)$$

Если полиномы представить их кодовыми эквивалентами, то эту же операцию можно записать следующим образом:

$$\begin{array}{ll}
A & 1001 \\
G & 1101 \\
\hline
 & 1001 \\
 & 1001 \\
 & 0000 \\
 & 1001 \\
\hline
K = A \cdot G & 1100101
\end{array} \quad \oplus \qquad (10.3)$$

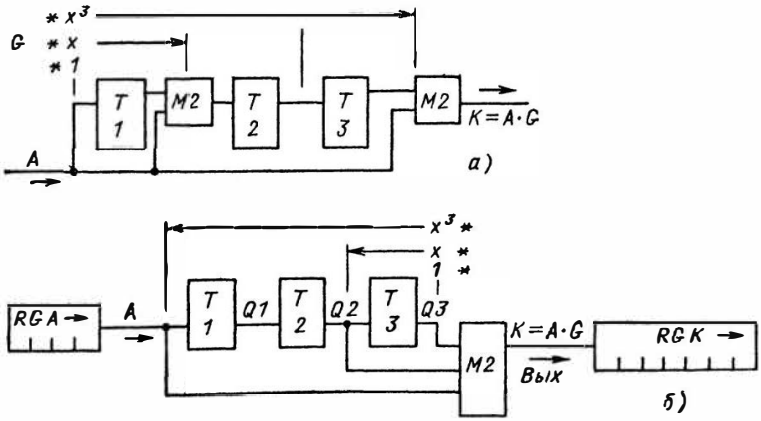


Рис. 10.8. Узел перемножения полиномов

Две конфигурации схем, реализующих эту операцию, т. е. выполняющих перемножение двух полиномов, показаны на рис. 10.8, *а* и *б*. От полиномиальных счетчиков схемы отличаются лишь отсутствием петли обратной связи. Один полином — полином A в кодовом представлении поступает на вход схемы последовательным кодом, начиная с коэффициентов при старших степенях x. Это может быть любой код, отображающий любой полином. Второй полином — G задан структурой схемы: в схеме по рис. 10.8, *а* — расположением мест ввода в регистр входного кода, а в схеме по рис. 10.8, *б* — расположением выводов на общий сумматор. Именно такая конфигурация обеспечивает сложение трех копий входного кода A, взаимно сдвинутых в соответствии с кодом множителя G и примера (10.3). Ниже показан процесс продвижения кода A из сдвигающего регистра источника через схему, показанную на рис. 10.8, *б*, в сдвигающий регистр приемника:

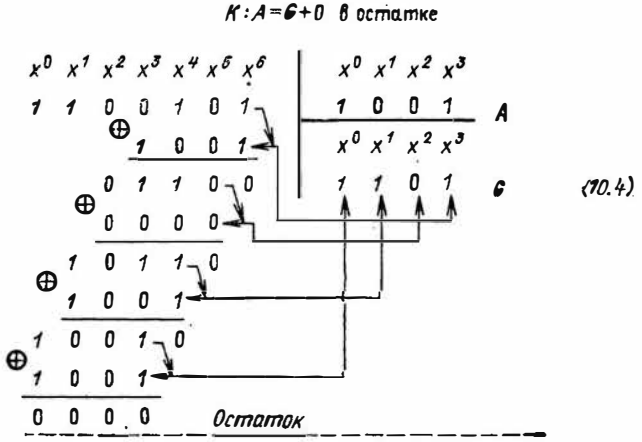
Такт	A	$Q2$ $Q1$ $Q3$	K
0	1 0 0 1		
1	1 0 0	1	1
2	1 0	0 1	0 1
3	1	0 0 1	1 0 1
4		1 0 0	0 1 0 1
5		1 0	0 0 1 0 1
6		1	1 0 0 1 0 1
7			1 1 0 0 1 0 1
			$K = A*G$

Чтобы лучше выявить сам процесс прохождения кода и не затемнять данную диаграмму, нули начальной установки регистров и нули, остающиеся в них после прохождения кода, представлены пустыми местами.

Упражнение. Проведите через узел умножения на G коды $A_2=1011$ и $A_3=1101$. Ответы: $K_2=A_2G=1111111$; $K_3=A_3G=1010001$.

Постройте узел умножения на другой полином G_1. Полином G_1 должен обязательно начинаться с единицы, а не с x, не с x^2 и т. д.

По аналогии с перемножением двоичных полиномов вводится и операция *деления полинома на полином*. При делении необходима операция вычитания, но результат вычитания по модулю 2 совпадает с результатом сложения по модулю 2. Как и при перемножении, старшие степени x располагаются справа. Проиллюстрировать деление можно на примере уже рассмотренных кодов:



В результате, как и следовало ожидать, получился код полинома G и нуль в остатке.

Комплекс сложений взаимно сдвинутых версий кода A, совпадающий с записью (10.4), реализуют схемы деления многочленов, показанные на рис. 10.9, *а* и *б*. По принципу построения они целиком совпадают с полиномиальными счетчиками. Как и узлы умножения, изображенные на рис. 10.8, обе схемы, показанные на рис. 10.9, специализированы

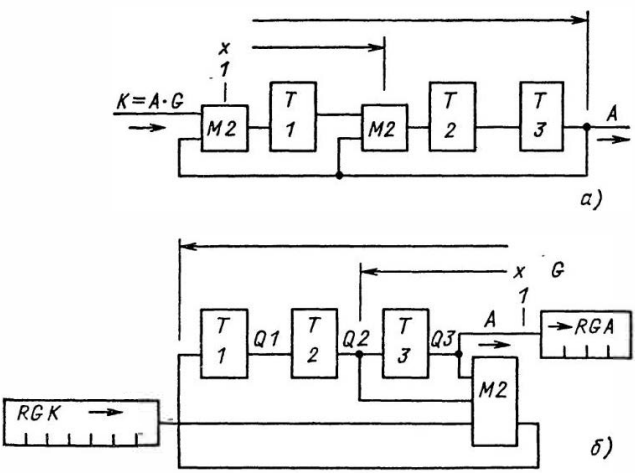


Рис. 10.9. Узел деления полиномов

для деления на полином $G=1+x+x^3$, что отражено соответствующим включением сумматоров. Очевидно, что аналогично можно построить схему деления на любой заданный полином, начинающийся с единицы. Ниже представлен процесс прохождения кода $K=A\cdot G$ через схему, показанную на рис. 10.9,*б*.

Такт	$K=AG$ →	$Q2$ $Q1$ $Q3$	
0	1 1 0 0 1 0 1		
1	1 1 0 0 1 0	1	
2	1 1 0 0 1	0 1	
3	1 1 0 0	0 0 1	
4	1 1 0	1 0 0	1
5	1 1	0 1 0	0 1
6	1	0 0 1	0 0 1
7		0 0 0	1 0 0 1
		ост. $=0$	A

В результате на выходе получается код A и нуль в остатке.

Узлы умножения и деления на полиномы широко используются в схемах контроля. Наиболее очевидный принцип контроля следующий. Перед подачей исходный код A на узле *кодера,* аналогичном показанному на рис. 10.8, умножается на некоторый *порождающий полином* G, и в линию (или в память) передается *кодовое слово* $K=A\cdot G$. На приемной стороне кодовое слово на *фильтре,* аналогичном показанному на рис. 10.9, делится на тот же полином G. Если оказалось, что остаток не равен 0, то значит, что в процессе передачи (хранения) произошла ошибка. Если остаток равен 0, то считается, что ошибки не было, хотя, в принципе, можно подобрать такую ошибку, которая при делении на G даст нулевой остаток. Очевидно, что она должна быть кратной G. Отсюда следует, что, в общем, чем сложнее полином G, тем меньше вероятность необнаруженной ошибки.

Распространены и другие способы передачи кодов. Так, вместо передачи по линии связи кодового слова K может передаваться сам код A и следом за ним остаток от деления его на порождающий полином G. На приемном конце еще раз выполняется деление и оба остатка сравниваются.

Хорошо разработанная теория полиномов позволяет целенаправленно подбирать вид порождающего полинома G, обеспечивающего заданные *корректирующие свойства* полученного с его помощью кода. Коды, основанные на операциях с порождающими полиномами, относятся к классу *циклических кодов.* Разработаны коды этого класса, способные исправлять одиночные ошибки в слове (ко-

ды Хэмминга), компактные группы (*пакеты*) ошибок длиной не более l (коды Файра), группы числом не более t произвольно разбросанных по слову ошибок (например, коды БЧХ). Параметры l и t также определяются видом порождающего полинома G.

Кодеры и фильтры циклических кодов применяют в запоминающих устройствах на магнитных дисках и лентах, в технике кабельной и радиосвязи и в других случаях, когда нужно контролировать информацию, передаваемую последовательным кодом. Для начального знакомства с принципами построения циклических кодов можно рекомендовать [29, 47, 48], для более глубокого изучения вопроса — [49, 50]. Примером микросхемы циклического регистра диагностики ошибок может служить КР1818ВЖ1.

Узлы деления полиномов используют также при контроле правильности работы логических схем методом *сигнатурного анализа*. На входы проверяемой схемы с генератора псевдослучайных чисел подается определенной длины и всегда одинаковая последовательность псевдослучайных кодов. Выход проверяемой схемы подключен к узлу деления полиномов, который делит получающуюся выходную последовательность на достаточно сложный полином G. При таком применении узел деления обычно называют *сигнатурным анализатором*. Остаток, оставшийся в регистре узла деления после окончания входной тестовой последовательности, называют *сигнатурой* (*signature* — подпись). Сигнатура на выходе исправной логической схемы заранее известна и может быть опознана оператором по цифровому индикатору или автоматически, с помощью компаратора. Сигнатура неисправной схемы иная, причем благодаря тому, что при делении на полином разряды кода сильно перемешиваются, любые небольшие отличия получающихся кодов резко усиливаются. Сигналы с проверяемых точек должны попадать в сигнатурный анализатор только после окончания в схеме переходных процессов, что достигается грамотным тактированием. Распространен 16-разрядный сигнатурный анализатор с обратными связями, заведенными в соответствии с предпоследней строкой табл. 10.4. Долю ошибок, не обнаруживаемых таким анализатором, можно оценить значением 2^{-16}.

Сигнатурный анализ используют как при стендовом контроле схем, так и при создании встроенной системы *самоконтроля* цифровых блоков. В системах самоконтроля хорошие результаты дает совместное применение сигнатур-

ного анализатора и сквозного сдвигового регистра, например, в таком варианте: тестовая последовательность снимается с генератора псевдослучайных чисел и непосредственно в процессе ее генерации заводится в сквозной сдвиговый регистр. Результат тестирования КС выводится из сквозного регистра и по мере своего выхода пропускается через сигнатурный анализатор. Полученная сигнатура сравнивается на компараторе с эталонной. В качестве генератора и анализатора есть возможность использовать один и тот же регистр. Подробнее эти вопросы изложены в [45, 46, 51, 52, 60].

Узлы умножения и деления на порождающий полином находят широкое применение также в автоматическом шифровании сообщений: если полином достаточно сложен, то выходной код имеет исключительно отдаленное сходство с кодом на входе узла (см. [25]).

ГЛАВА 11

АВТОМАТЫ

11.1. Обобщенная схема автомата

Автоматами, или *последовательностными* (не последовательными!) *схемами*, называют схемы, выходы которых зависят не только от значений входов в данный момент, но и от комбинаций значений входов в определенные прошлые моменты времени. Автомат в некотором смысле помнит прошлые воздействия в отличие от комбинационной схемы (КС), выход которой определяется значениями входов только в настоящее время (разумеется, после окончания переходных процессов). *Автомат имеет память*, КС памяти не имеет. Иногда КС трактуют как частный случай автомата — автомата без памяти, но это, пожалуй, лишь неоправданное усложнение терминологии.

Тем или иным содержимым памяти автомата определяется его *внутреннее состояние*, или просто *состояние*. Внешнее проявление различных состояний — это различные реакции автомата на одни и те же воздействия. Как и мы: находясь в различных состояниях, или улыбаемся, или обижаемся на одну и ту же шутку.

Обобщенная структурная схема цифрового автомата показана на рис. 11.1. *Выходной код Y*, вырабатываемый КС, есть функция не только *входного кода X*, но и *кода состояния Z*, который хранится в *регистре состояний RGZ*. Этот регистр и есть *память* автомата. Чем больше его емкость, тем богаче спектр поведения автомата, шире разнообразие его реакций на одни и те же входные воздействия, полнее учет прошлого опыта. Новое состояние Z_{i+1}, в которое переходит автомат после очередного входного воздействия, т. е. новое содержимое *RGZ*, задается *кодом перехода F*. Код *F*, так же как и выходной код *Y*, есть функция и входных сигналов, и состояния автомата непосредственно перед его переходом в новое.

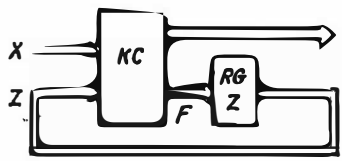


Рис. 11.1. Обобщенная схема автомата

Формально под определение автомата подходит любое цифровое устройство, имеющее хотя бы один триггер. Отдельный триггер, регистр, счетчик любого типа, целая ЭВМ — это все автоматы. Однако рассмотрение триггера или регистра с позиций теории автоматов сегодня имеет скорее академический интерес. Практически схемотехники пользуются типовыми отработанными схемами, основные из которых были описаны в гл. 6, 9 и 10. Законы функционирования этих схем хорошо известны, и при работе с ними никто не применяет каких-либо специфических приемов теории автоматов. В среде разработчиков понятие «автомат» трактуется уже. О *синтезе автоматов* начинают говорить, когда нужно разработать схему цифрового блока с памятью, закон работы которого, с одной стороны, не очевидно прост, т. е. он не воспроизводится каким-либо стандартным узлом-автоматом типа триггера, счетчика, распределителя или очевидными их комбинациями, например счетчика с мультиплексором на выходе. Но при этом, с другой стороны, закон поведения проектируемого блока еще не слишком сложен. Под этим понимается ситуация, когда все рабочие комбинации входов, состояний, выходных сигналов как-то еще можно отобразить в виде обозримых списков и таблиц, с которыми реально можно работать. Несмотря на расплывчатость этого определения, именно фактор *обозримости* имеет здесь решающее значение. Если сложность устройства выходит за рамки обозримости, то его уже не проектируют как единый автомат. Его или разбивают на систему отдельных автоматов, или, чаще, используют общепринятую структуру *процессора*, а если есть возможность, то готовый микропроцессор. Рассмотрение с позиций тео-

рии синтеза автоматов устройств, сложность которых соизмерима со сложностью хотя бы простейшего микропроцессора, также носит или академический, или методологический характер. Построить целый микропроцессор по методике синтеза автоматов практически невозможно. Микропроцессор для этого слишком сложен.

Задачу синтеза автомата средней сложности удобно разбить на три части: *формализация задания, кодирование состояний, синтез комбинационной схемы.* Эти части не полностью автономны, и работа над одной из них обычно требует коррекции результатов двух других.

11.2. Формализация задания автомата

Автоматы обычно строят для управления работой различных технических объектов: исполнительных механизмов, приводов, электронных устройств вплоть до целых ЭВМ. В этой роли автоматы называют программными датчиками, генераторами управляющих сигналов, блоками (устройствами) управления. Управляющим воздействием на объект являются выходы автомата Y. На входы автомата X поступают команды оператора (или вышестоящего автомата), сигналы, описывающие окружающую обстановку, сигналы о ситуациях, возникающих в управляемом объекте. Автомат проектируется под конкретный объект с его конкретным алгоритмом управления, и сложность проектирования автомата — именно в его нестандартности. Задание на автомат сначала дается в виде словесного описания закона управления, и, как правило, первый вариант задания оказывается очень неполным, допускающим неоднозначное толкование. Поэтому сначала задание нужно *формализовать.* В процессе этой работы задание уточняется, корректируется. Чтобы с этим справиться, разработчик или сам должен хорошо знать условия работы объекта, или иметь на этапе формализации тесный контакт с заказчиком. Проблемы согласования задания полно и образно изложены в [53]. Чтобы проиллюстрировать характер возникающих при синтезе автомата вопросов, процедура синтеза будет излагаться на примере простого объекта с хорошо понятными алгоритмом и условиями работы.

Задание. Спроектировать схему кодового замка двери комнаты. Соленоид должен оттягивать ригель замка после поочередного нажатия двух (а не трех, как обычно: это для упрощения) из десяти кнопок, а именно *1* и *3*. В общем случае обе цифры должны быть различными. В исходное

состояние схема возвращается единичным уровнем сигнала *Д*, снимаемого с контакта при открывании двери.

Это и есть первичная, словесная, интуитивно ясная и воспринимающаяся как достаточно полная формулировка задания. Требуемый автомат исключительно прост, и схему его можно построить на интуитивном уровне понимания без какой-либо методики. Полезно попытаться сделать это, а затем сравнить затраты времени, схему, а особенно — число ошибок и недоделок с результатом работы по книжке: будет хорошо видно, на каких этапах методика помогает интуиции даже в случае очень простой задачи.

Проектируемая схема не может быть просто комбинационной. Это автомат, поскольку на один и тот же входной сигнал, например на нажатие кнопки *3*, схема должна реагировать по-разному: как на неверный сигнал, если это первое нажатие, как на верный, если перед этим была нажата кнопка *1*, и никак не реагировать после того, как соленоид уже сработал, а дверь еще не открыта. Процесс формализации задания удобно разбить на этапы.

1-й этап формализации — формирование списка входных сигналов *X* и выходных *Y*, которые будут использоваться при разработке схемы. В данном случае формально на вход воздействует сигнал открытой двери *Д* и 10 кнопок. Однако можно ожидать, что избирательно автомат будет реагировать лишь на пять входных ситуаций: сигнал правильной первой цифры *П1* (цифра *1*), сигнал правильной второй цифры *П2* (цифра *3*), сигнал неправильной первой цифры *Н1* (любая цифра, кроме *1*, в том числе и *3*), сигнал неправильной второй цифры *Н2* (любая цифра, кроме *3*, в том числе и *1*) и сигнал *Д*. Для упрощения дальнейшей работы рационально весь набор ситуаций *X*, фактически воздействующих на автомат, сформировать заранее на вспомогательных логических схемах, как показано на рис. 11.2. На цепочку элементов, начинающуюся с шины питания $U_{\text{ип}}$, внимания пока обращать не нужно. Аналогично формируется массив выходных сигналов *Y*, который в примере представлен единственным выходным уровнем *Р*.

2-й этап формализации — определение требуемого задачей числа состояний автомата. В данном случае решение очевидно: три состояния. *Z0* — *начальное* состояние, состояние ожидания первого правильного сигнала; *Z1* — первый правильный сигнал получен, ожидание второго; *ZP* — оба сигнала получены, вырабатывается единичный уровень выходного сигнала *Р* на усилитель соленоида зам-

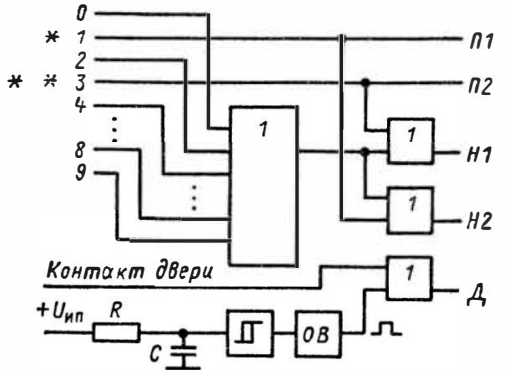


Рис. 11.2. Формирование входного набора сигналов

ка. Если бы отпирающая последовательность состояла из трех нажатий кнопок, то добавилось бы еще одно состояние. Можно, если этого требуют условия работы, ввести еще состояние — тревога, в которое автомат переходит в случае набора неверной цифры и при этом выдает сигнал куда следует. Кстати, как автомат должен поступать в случае неправильного набора — это первый вопрос, ответа на который не было в первичной формулировке задания, который возникает при проектировании и требует выяснения у заказчика. Допустим, ответ таков: при неправильном наборе схема сбрасывается в начальное состояние.

При проектировании сложных автоматов число состояний становится очевидным далеко не сразу, а лишь в процессе изучения задания, по мере более глубокого понимания алгоритма автомата и уяснения разнообразия его реакций на одинаковые входные сигналы.

3-й этап формализации — построение предварительного, упрощенного *графа* или *таблицы* автомата (синтез абстрактного автомата). В примере с замком лучше начинать с графа, поскольку при числе состояний в пределах 10—15 граф обычно нагляднее таблицы. В *графовом представлении* функционирования автомата (рис. 11.3) состояния автомата отображаются вершинами графа, а возможные переходы автомата из одного состояния в другое — дугами графа. Направления переходов обозначены стрелками. Дуги помечены входными сигналами-условиями, при которых эти переходы выполняются.

Выходные сигналы, если они связаны только с определенным состоянием, изображаются внутри кружка состояния, порождающего этот сигнал. Если кроме состояния выходной сигнал зависит еще и от входных сигналов, то он изображается выходящей из соответствующего состояния стрелкой, помеченной порождающим входным сигналом. В рассматриваемом примере разумно полагать, что выходной сигнал *P* однозначно связан с состоянием автомата *ZP*, т. е. $P=1$, тогда, когда автомат находится в состоянии *ZP* независимо от действующих при этом входных сигналов. Граф на рис. 11.3 есть символическое отражение словесного описания автомата. Пока на вход не поступит сигнал *П1*, автомат находится в начальном состоянии *Z0*. Сигнал *П1* переводит его в состояние *Z1*, откуда он при поступлении неверного сигнала *Н2* вернется в *Z0*, а при поступлении второго правильного сигнала *П2* перейдет в состояние *ZP* разрешения открыть замок. Сигнал *Д*, возникающий при открывании двери, сбрасывает автомат в *Z0*, подготавливая его к следующему циклу работы. Никакие другие сигналы не выводят автомат из состояния *ZP*: пусть такова воля заказчика. В принципе можно задать и иные условия.

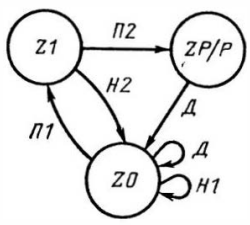


Рис. 11.3. Граф переходов автомата: первая версия

Для строгого указания на тот факт, что, несмотря на поступление некоторых сигналов, автомат тем не менее не изменяет своего состояния, на графе переходов около этих состояний чертят петли, помеченные соответствующими сигналами. Примеры двух таких петель показаны у вершины *Z0* (рис. 11.3). Строгое начертание графа потребовало бы еще одной петли *П1* у вершины *Z1* и трех петель — *П1*, *П2* и *Н2* — у вершины *ZP* (или хотя бы двух — *П2* и *Н2*, поскольку сигнал *П1* входит в состав *Н2*). Петли сильно загромождают чертеж, и их, как правило, не изображают, а лишь подразумевают, полагая, что при поступлении любого сигнала, кроме тех, которыми помечены стрелки перехода из данного состояния, автомат своего состояния не изменит. Несколько позже будет показано, что замалчивание существования петель оправдано и с позиций построения логической схемы.

Наряду с графовым используют *табличное описание* ав-

Таблица 11.1

Входы	Состояния		
	Z0	*Z1*	*ZP*
П1	*Z1*	*Z1*	*ZP*
П2	*Z0*	*ZP*	*ZP*
Н2	*Z0*	*Z0*	*ZP*
Д	*Z0*	*Z0*	*Z0*
Выход	—	—	*P*

томата, общепринятая форма которого приведена в табл. 11.1. Строки таблицы отображают входные сигналы, столбцы — состояния. На пересечении строки и столбца записывается новое состояние, в которое автомат перейдет из состояния данного столбца под воздействием входного сигнала данной строки. В последней строке приведены те выходные сигналы, которые однозначно связаны с состоянием. Если выходной сигнал связан еще и со входным, то он изображается на соответствующем пересечении и отделяется от нового состояния косой чертой, как знаменатель дроби. Если в таблице заполнены все клетки, то говорят, что автомат *определен полностью*. В задании на *недоопределенный* автомат в некоторых клетках могут стоять кресты — символы безразличия. В процессе построения логической схемы разработчик доопределяет таблицу исходя из соображений качества схемы — аналогично доопределению таблицы логической функции (см. § 1.6).

Граф и таблица — два взаимно эквивалентных способа формального описания поведения автомата. Каждый из них задает реакцию проектируемого автомата на каждый входной сигнал (же однозначно), без опасности разночтения. Чтобы граф был строго эквивалентен таблице, на нем нужно изобразить в явном виде все петли.

Описанный процесс построения графа или таблицы называют этапом *абстрактного синтеза* автомата или этапом *синтеза абстрактного автомата*. *Абстрактный автомат* — это еще не устройство и даже не схема. Это лишь математическая модель, это алгоритм функционирования некоторого преобразователя кодовых последовательностей. Если входы в табл. 11.1 (или на графе) отождествить с буквами некоторого входного алфавита, а выходы — с буквами выходного алфавита (в случае замка выходной алфавит состоит лишь из двух букв: P_1

и *ПРОБЕЛ*), то абстрактный автомат — это закон преобразования цепочек букв (т. е. слов) входного алфавита в цепочки букв (слова) выходного алфавита. Примерно такая постановка задачи и послужила в свое время исходным толчком к созданию теории автоматов. Создание абстрактного автомата — это первый ощутимый результат на пути построения схемы автомата. Дальнейшие шаги разработки автомата, приблизительно до уровня получения его функциональной схемы, принято называть этапом *структурного синтеза*.

4-й этап формализации — построение полного графа (полной таблицы) автомата. Формальное описание абстрактного автомата, полученное на предыдущем этапе, как правило, оказывается недостаточно полным для того, чтобы непосредственно по нему можно было вести синтез логической схемы. Обычно требуется еще рассмотреть и формализовать следующие не учтенные абстрактной моделью моменты: вопрос о *неиспользуемых состояниях* автомата, *начальная установка* его после включения питания, возможность *совпадения входных сигналов* во времени.

Неиспользуемые (запрещенные) состояния. О выборе кодов регистра Z для представления состояний автомата речь будет впереди, а пока пусть известно, что три рабочих состояния автомата хранятся в двухзарядном триггерном регистре. При этом четвертая из возможных комбинаций состояний триггеров не используется, в цепочке переходов автомата не участвует, и о ней, вроде бы, можно и не говорить. Однако если из-за воздействия помехи в регистре ошибочно возникнет это четвертое состояние, автомат в нем «зависнет», и никакие внешние сигналы уже не смогут вернуть его в рабочий цикл. Говорят, что в графе переходов существует *изолированная вершина*. В грамотно спроектированной схеме изолированных вершин или циклов из нескольких таких вершин быть не должно. От неиспользуемых вершин нужно сделать «стоки» в рабочий цикл графа, и это должно быть отражено на самом графе. Задача выхода из изолированного цикла уже однажды решалась в ее конкретном применении к кольцевым распределителям (см. § 10.2 и рис. 10.5).

В рассматриваемом примере на новом чертеже полного графа (рис. 11.4) лишняя вершина $ZЛ$ уже показана явно и подключена к остальной части графа дугами $Д$ и $П1 \cdot \overline{Н1} \times \times \overline{Д}$. Это решение допустимое и неплохое: если в автомате вдруг возникнет состояние $ZЛ$, то или первое же открывание двери вернет автомат в начальное состояние, или нажа-

тие первой верной кнопки переведет его в *Z1*. Почему вм сто простого условия перехода *П1* использовано более слож. ное — $П1 \cdot \overline{Н1} \cdot \overline{Д}$, будет ясно немного позже.

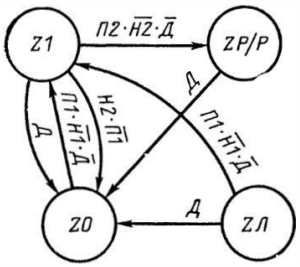


Рис. 11.4. Полный граф переходов автомата

Начальная установка. При включении питания триггеры регистра, а вместе с ними и весь автомат оказываются в случайном состоянии. При некоторых применениях допустимо устанавливать автоматы в начальное состояние *Z0*, выполнив первый холостой установочный прогон до конца цикла. Если эффект, вызываемый таким неполным циклом, начавшимся с произвольной промежуточной точки, недопустим, то автомат снабжают цепями *начальной установки (начального сброса)*. На графе переходов для этого вводят дуги переходов от каждой вершины к нулевой, которые помечают еще одним специально введенным входным сигналом — сигналом начального сброса.

В примере с кодовым замком начальная установка путем холостого прогона чревата неприятностями. Если автомат при включении питания с равной вероятностью устанавливается в любое состояние, в том числе и в *ZP*, то злоумышленник, выворачивая в обеденный перерыв на несколько секунд пробки и пробуя затем дверь, будет проникать в любую защищенную комнату в среднем за четыре попытки. Для начальной установки замка специальной цепи сброса можно не вводить. Роль сигнала сброса может выполнить сигнал *Д*: дуги *Д* от вершин *ZP* и *ZЛ* к *Z0* уже есть и если добавить еще одну дугу *Д* от *Z1* к *Z0*, то цель, которой служит автомат, не пострадает: хотя открываемая неожиданно выходящим из комнаты дверь и прервет интеллектуальный процесс набора кода другим лицом, желающим попасть в комнату, но в результате дверь все равно окажется открытой и желание набиравшего — выполненным. Отметим, что решение использовать тракт *Д* еще и таким образом требовало знания реальных условий работы автомата, а не вытекало непосредственно из текста задания. Дуга от *Z1* к *Z0* показана на рис. 11.4. Вообще в цифровой аппаратуре цепи начального сброса автоматов делают как с ручным запуском, вводя специальную кнопку *СБРОС*, так

и с автоматическим, от включения питания. Последний вариант показан на рис. 11.2. После включения питания *RC*-цепочка с постоянной времени порядка секунды через триггер Шмитта запускает одновибратор *ОВ*, импульс которого через элемент ИЛИ поступает в тракт *Д* и устанавливает автомат в начальное состояние. И еще один метод: в электрических схемах БИС триггеры иногда выполняют немного асимметричными, так, чтобы в процессе появления напряжения питания они сами устанавливались в 0.

Совпадение входных сигналов во времени. Если автомат создается не для обработки текстов, где буквы поступают строго по одной, а как управляющее устройство, которое должно реагировать на изменения окружающей обстановки, то возможности совпадения во времени входных сигналов, по крайней мере от независимых источников, исключить нельзя. Однако в таблице переходов и выходов абстрактного автомата реакция на возможные совпадения сигналов никак не отражена, поэтому схема, построенная по такой таблице, может реагировать на совпадения входных сигналов любым, самым неожиданным образом. Чтобы этого не произошло, разработчик должен предусмотреть адекватные реакции автомата на эти случаи и отразить их в таблице или на графе переходов.

Формальным приемом учета возможности совпадения сигналов, который часто рекомендуют к применению на этапе структурного синтеза автомата, является переход к новому набору абстрактных входных сигналов, число которых равно числу всех комбинаций реальных сигналов. В случае замка пять входных сигналов с помощью дешифратора можно превратить в 2^5, т. е. в 32 абстрактных сигнала, которые уже будут появляться строго по одному. Для описания автомата в этом случае потребуется таблица в 32 строки, столько же дуг (включая петли) будет и у графа. Соответственно усложнится процедура синтеза схемы.

Этот прием оправдан и даже необходим, когда разработчик приписывает определенный смысл именно *комбинациям* входных сигналов, как, например, при кодировании k-разрядным кодом команды 2^k запускаемых этой командой различных операций. Однако в устройствах автоматики намного чаще определенный смысл приписывается не комбинации сигналов, а именно отдельному сигналу (перегрев, ошибка, дверь открыта и т. п.) вне связи с другими. Поэтому наиболее адекватной реакцией при совпадении сигналов часто оказывается реакция на наиболее важный из них как

на единственный в данный момент, а все остальные сигнал игнорируются. Если приоритет не определен четко самим содержанием задачи, он может быть назначен произвольно Ранжирование входных сигналов по приоритетам снимает с разработчика большой труд по осознанию требуемой реакции автомата на каждую комбинацию различных входных сигналов и существенно упрощает как таблицу, так и схему автомата. При этом ранжировать требуется не все сигналы, а лишь те, которые порождают *конфликтные ситуации,* т. е. при своем совпадении вызывают на графе переходов ситуации неопределенности или генерации. Для каждого состояния автомата конфликтующие группы сигналов в общем различны, и начинать работу нужно с выявления этих групп.

В рассматриваемом примере в состоянии *Z0* конфликтуют две пары сигналов: *П1* с *Д* и *П1* с *Н1*. При одновременном появлении любой пары сигнал *П1* будет стремиться перевести автомат в состояние *Z1*, а *Д* или *Н1* — оставить в *Z0*. По смыслу работы замка сигналам *Д* и *Н1* нужно назначить более высокий приоритет, чем сигналу *П1*, т. е. разрешить переход в *Z1* по сигналу *П1* лишь при условии, что дверь закрыта ($\overline{Д}$) и кроме правильной кнопки не нажата больше никакая другая, а тем более сразу все — ладонью ($\overline{Н1}$). Для этого переход *Z0—Z1*, который на рис. 11.3 был помечен безусловным сигналом *П1*, на уточненном графе (см. рис. 10.4), помечается уже конъюнкцией $П1 \cdot \overline{Н1} \cdot \overline{Д}$. Снова для формализации задания потребовалось знание условий работы объекта.

При рассмотрении вершин *Z0* и *Z1* конфликтующей парой оказываются сигналы *П1* и *Н2*. Оба они порождаются нажатием кнопки *1* (см. рис. 11.2), но один из них вызывает переход *Z0—Z1*, а второй — обратно, *Z1—Z0*. Все время, пока нажата кнопка *1*, схема будет находиться в режиме генерации. Чтобы состояние *Z1* сделать устойчивым, необходимо сигналу *П1* в этом состоянии дать более высокий приоритет, чем сигналу *Н2*. Для этого дуга *Z1—Z0* помечается функцией $Н2 \cdot \overline{П1}$, и тогда вернуться в *Z0* от неверно нажатой кнопки автомат сможет лишь после отпускания кнопки *П1*.

В состоянии *Z1* сигналы *Д* и $Н2 \cdot \overline{П1}$ не конфликтуют, поскольку заняты одним общим делом, и совпадение их во времени не вызовет осложнений. Поэтому здесь введения приоритетов не требуется.

В неполном графе, показанном на рис. 11.3, переход *Z1—ZP* выполняется по условию *П2*. Однако в этом случае нажимать кнопку *П2* и переходить в *ZP* можно, не отпустив предварительно кнопки *П1*. Нажатая кнопка *П1* не дает автомату сброситься из *Z1* в *Z0* даже при одновременном нажатии любой неверной кнопки. Значит, фактически существенным оказывается знание лишь первой цифры. После ее нажатия злоумышленник, не отпуская первой кнопки, нажимает ладонью все остальные. Неверная кнопка замка не сбросит, а верная его откроет. Чтобы вторая цифра была столь же значимой, что и первая, в графе на рис. 11.4 переход *Z1—ZP* делается по условию $П2 \cdot \overline{Н2}$, т. е. переход в *ZP* по нажатию *П2* осуществится, только если *П1* отпущена и вообще не нажата никакая другая кнопка, кроме *П2*.

В *Z1* конфликт возникает между нажатием кнопки *П2* (сигнал $П2 \cdot \overline{Н2}$) и сигналом *Д*. Конфликт ликвидируется, если одному из сигналов, например *Д*, назначить более высокий приоритет. Это отражено в условии перехода *Z1—ZP*: $П2 \cdot \overline{Н2} \cdot \overline{Д}$.

Введение приоритетов позволило ликвидировать все возможные сбои при совпадении входных сигналов во времени, не увеличив при этом числа дуг графа. Граф, показанный на рис. 11.4, считается окончательным. Автор не утверждает, что построенный граф — самый правильный или самый удачный. Цель его — проиллюстрировать сам процесс формализации задания. Характерно, что для построения графа понадобился ряд сведений о реальных условиях работы автомата, которые не были изложены в первоначальном задании и, вообще говоря, требовали согла-

Таблица 11.2

№ п/п	Аргументы						Функции	
	Состояние в данный момент	Входы автомата					Следующее состояние	Выход *Р*
		П1	*Н1*	*П2*	*Н2*	*Д*		
1	*Z0*	1	0	×	×	0	*Z1*	0
2	*Z1*	×	×	×	×	1	*Z0*	0
3	*Z1*	0	×	×	1	×	*Z0*	0
4	*Z1*	×	×	1	0	0	*ZP*	0
5	*ZP*	×	×	×	×	1	*Z0*	1
6	*ZЛ*	×	×	×	×	1	*Z0*	0
7	*ZЛ*	1	0	×	×	0	*Z1*	0

сования с заказчиком. Построенный граф представле
в табличной форме в табл. 11.2. Поскольку входные сигнал
теперь уже не абстрактные, таблица не получается стол
изящной, как табл. 11.1. Построением полной таблицы а
томата, охватывающей все его возможные (в том числ
и нежелательные) состояния и все комбинации входных си
налов, способные повлиять на работу, и заканчивается эта
формализации задания.

В табл. 11.2 $\times$ обозначает, что как следующее состоя
ние автомата, так и его выходной сигнал, относящиеся
данной строке, не должны зависеть от значения входног
сигнала того столбца, в котором стоит этот символ.

11.3. Регистр состояний автомата

К вопросам разработки регистровой части автомат
принято относить выбор числа триггеров регистра и способ
кодирования состояния. Минимально возможное числ
триггеров равно ближайшему сверху целому от двоичног
логарифма числа состояний. Максимальное число тригге
ров равно числу состояний, при этом каждое состояние ко
дируется единицей в одном из триггеров. В среднем с ро
том числа триггеров в пределах указанного диапазон
уменьшается число логических элементов, требующихся дл
дешифрации состояний. При максимальном числе триггеро
дешифрации вообще не требуется. Поэтому высказываемс
иногда утверждение, что чем меньше триггеров, тем автом
экономичнее по аппаратурным затратам, нельзя считат
безусловно верным. Кроме соображений экономичности К
увеличению числа триггеров сверх необходимого минимум
способствует использование микросхем регистров с число
разрядов, кратным четырем.

При кодировании состояний, если нет других соображ
ний, вершины графа переходов нумеруются в произвольно
порядке и номера кодируются по двоичной системе. Одна
объем комбинационной схемы можно уменьшить, если дл
вершин, связанных бо́льшим числом дуг переходов, подб
рать коды, отличающиеся возможно меньшим числом ра
рядов. Тогда для обеспечения каждого перехода придет
заводить сигналы на входы меньшего числа триггеров. Ин
гда для уменьшения числа переключаемых триггеров ув
личивают общее их число. Алгоритма нахождения наиб
лее экономичного решения, кроме полного перебора вар

антов, не существует. Разработчики используют метод проб и ошибок и свой опыт.

Распространено мнение, что в автомате не будет гонок и он может быть сделан асинхронным, без системы синхронизации, если его состояния закодировать так, чтобы при любом переходе изменялось значение только одного разряда. Это так называемое *противогоночное кодирование* состояний автомата. В случае разветвленного графа такое кодирование требует введения дополнительных триггеров в регистр. В основе мнения о действенности противогоночного кодирования лежат добротные работы, но большой давности, основанные на характерном для того времени соотношении задержек, когда задержка переключения триггера существенно превышала задержку КС. В этих условиях основной причиной возникновения гонок действительно было неодинаковое время переключения триггеров. Поэтому если при любом переходе автомата в регистре состояний переключался всего только один триггер, гонок не возникало.

В современной элементной базе задержка триггера соизмерима с задержкой одного логического элемента, поэтому задержка комбинационной схемы средней сложности превышает задержку триггера. Еще хуже ситуация внутри БИС, где существенную роль играют еще и трудноконтролируемые задержки в связях. В этих условиях основное число помех гоночного типа зарождается в недрах КС автомата, и противогоночное кодирование его состояний здесь бессильно. Тем не менее поток рекомендаций применять этот метод до сих пор не иссякает. Построение КС, в которых гонки невозможны, весьма трудоемко и реально выполнимо лишь для очень простых схем. Кроме того, такие схемы требуют больше оборудования, и эта разница уже при достаточно простых схемах превышает затраты оборудования на генератор синхросигналов. Если же желание построить автомат по противогоночной идеологии все-таки сильно, то за помощью можно обратиться к [33].

Учитывая все сказанное, число триггеров регистра состояний кодового замка принимается равным двум. Кодирование его состояний: $Z0=00$; $Z1=01$; $ZP=11$; $ZЛ=10$. Коды для $Z0$ и $Z1$ и $Z1$ и ZP выбраны отличающимися лишь одним разрядом (*соседнее кодирование*), чтобы уменьшить, как уже говорилось, объем логических схем: именно эти состояния связаны наибольшим числом дуг. Поскольку устройство легко разместится на одной плате, т. е. расфазировки активных фронтов можно не опасаться, система синхронизации выбирается однофазной, а триггеры — соответственно синхронные JK-триггеры. Генератором синхросигналов может служить мультивибратор на логических микросхемах. Входные сигналы привязываются к синхросерии с помощью схем-синхронизаторов. Дребезга контактов схема не боится:

таблица переходов автомата построена так, что снятие, а затем повторная подача входного сигнала не нарушают работы автомата.

11.4. Комбинационная схема автомата

После задания способа кодирования состояний разработчик имеет уже всю необходимую информацию для построения КС автомата. Для кодового замка эта информация представлена в табл. 11.3, которая, по сути, повторяет

Таблица 11.3

№ п/п.	Входы КС								Выходы КС				
	Состояние Z			Входные сигналы X					Имя следующего состояния	Код перехода в следующее состояние, F				Выход Y
	имя	код												
		$Q2$	$Q1$	$П1$	$Н1$	$П2$	$Н2$	$Д$		$J2$	$K2$	$J1$	$K1$	P
1	Все	×	×	×	×	×	×	1	$Z0$	0	1	0	1	0
2	$Z0$	0	0	1	0	×	×	0	$Z1$	0	0	1	0	0
3	$Z1$	0	1	0	×	×	1	×	$Z0$	0	0	0	1	0
4	$Z1$	0	1	×	×	1	0	0	ZP	1	0	0	0	0
5	ZP	1	1	×	×	×	×	×	ZP	0	0	0	0	1
6	ZJ	1	0	1	0	×	×	0	$Z1$	0	1	1	0	0

табл. 11.2, но уже с расшифровкой текущего и следующего состояний автомата в виде значений $Q2$, $Q1$ — выходов триггеров регистра и $J2$, $K2$, $J1$, $K1$ — их входов. КС имеет семь входов, и полная таблица ее значений имела бы 128 строк. Сократить число строк до шести удалось благодаря двум достаточно универсальным приемам.

Во-первых, конфликтующим входным сигналам были присвоены приоритеты. Чем выше приоритетность некоторого сигнала, тем больше в таблице на строке, где этот сигнал проявляется, крестов (×), а каждый крест, относящийся к какой-то переменной, отображает сразу две строки — и для значения этой переменной 0, и для значения 1. Одна строка с четырьмя крестами заменяет сразу 16 строк. Первая строка табл. 11.3 вобрала в себя все дуги сброса автомата в $Z0$ сигналом $Д$ (строки 2, 5 и 6 табл. 11.2), в том числе и из состояния ZP, поэтому в строке ZP (строка 5 табл. 11.3) достаточно было отобразить лишь значение выходного сигнала P.

Во-вторых, в качестве триггеров использованы JK-триггеры, которые при $J=K=0$ хранят свое состояние. Следо-

вательно, все строки с такими комбинациями входов КС, которые не вызывают смены состояния автомата, т. е. переключения триггеров, можно в таблицу не вносить: все равно на выходе КС они дают нули и поэтому в состав СДНФ не войдут. Другими словами, строя таблицу, достаточно заносить в нее только метки на дугах смены состояний автомата, и не требуется заносить метки петель, указывающих на сохранение состояния автомата. Поэтому петли на графе можно с самого начала не чертить. Если бы вместо JK- использовать D-триггеры, то число строк таблицы увеличилось бы существенно, поскольку для простого поддержания выхода D-триггера в 1 на его D-вход должна обязательно каждый такт подаваться 1.

Таблица 11.3 позволяет построить логические функции для всех пяти выходов КС методами, изложенными в гл. 1—3.

$$J2 = \overline{Q2} \cdot Q1 \cdot П2 \cdot \overline{Н2} \cdot \overline{Д};$$

$$K2 = Д \vee Q2 \cdot \overline{Q1} \cdot П1 \cdot \overline{Н1};$$

$$J1 = \overline{Q2} \cdot \overline{Q1} \cdot П1 \cdot \overline{Н1} \cdot \overline{Д} \vee Q2 \cdot \overline{Q1} \cdot П1 \cdot \overline{Н1} \cdot \overline{Д} =$$

$$= \overline{Q1} \cdot П1 \cdot \overline{Н1} \cdot \overline{Д};$$

$$K1 = Д \vee \overline{Q2} \cdot \overline{Q1} \cdot \overline{П1} \cdot Н2;$$

$$P = Q2 \cdot Q1.$$

Комбинационная схема автомата — это заданный таблицей кодовый преобразователь некоторого входного кода в выходной, и при такой трактовке для его построения годятся все методы, изложенные в § 3.4—3.7. В частности, КС можно строить на ПЛМ или ПЗУ, если это даст экономию затрат. Существуют микросхемы ПМЛ, в состав которых входят программируемая логическая матрица и небольшой триггерный регистр. Запрограммировав такую микросхему, можно весь автомат реализовать на одном корпусе.

Подход к синтезу автоматов подробно изложен в [2, 22, 53, 55], ряд деталей этого процесса освещен в [56, 57, 58]. Методы построения апериодических (самосинхронизирующихся) автоматов изложены в [34].

СПИСОК ЛИТЕРАТУРЫ

1. **Поспелов Д. А.** Логические методы анализа и синтеза схем. М.: Энергия, 1974.

2. **Закревский А. Д.** Логический синтез каскадных схем. М.: Наука, 1981.

3. **Гивоне Д., Россер Р.** Микропроцессоры и микрокомпьютеры: Пер. с англ. М.: Мир, 1983.

4. **Голдсуорт Б.** Проектирование цифровых логических устройств: Пер. с англ./Под ред. Ю. И. Топчеева. М.: Машиностроение, 1985.

5. **ГОСТ 2.743—82.** Единая система конструкторской документации. Обозначения условные графические в схемах. Элементы цифровой техники.

6. **Шоломов Л. А.** Основы теории дискретных логических и вычислительных устройств. М.: Наука, 1980.

7. **Артюхов В. Л., Копейкин Г. А., Шалыто А. А.** Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат. Ленингр. отд-ние, 1981.

8. **Справочник** по интегральным микросхемам/Б. В. Тарабрин, С. В. Якубовский, Н. А. Барканов и др.; Под ред. Б. В. Тарабрина. М.: Энергия, 1980.

9. **Интегральные** микросхемы: Справочник/Б. В. Тарабрин, Л. Ф. Лунин, Ю. Н. Смирнов и др.: Под ред. Б. В. Тарабрина. М.: Энергоатомиздат, 1985.

10. **Аналоговые** и цифровые интегральные микросхемы: Справочное пособие/С. В. Якубовский, Н. А. Барканов, Л. И. Ниссельсон и др.; Под ред. С. В. Якубовского. М.: Радио и связь, 1984.

11. **Ланцов А. Л., Зворыкин Л. Н., Осипов И. Ф.** Цифровые устройства на комплементарных МДП интегральных микросхемах. М.: Радио и связь, 1983.

12. **Кармазинский А. Н.** Синтез принципиальных схем цифровых элементов на МДП-транзисторах. М.: Радио и связь, 1983.

13. **Основы** построения технических средств ЕС ЭВМ на интегральных микросхемах/В. В. Саморуков, В. М. Микитин, В. А. Павлычев и др.; Под ред. Б. Н. Файзулаева. М.: Радио и связь, 1981.

14. **Разумов Ю. И., Пупин А. А., Курочкин В. Г.** Выбор функционального состава ячейки базового кристалла//Микроэлектроника и полупроводниковые приборы/Под ред. А. А. Васенкова, Я. А. Федотова. М.: Радио и связь, 1981. Вып. 6. С. 82—94.

15. **Пономарев М. Ф., Коноплев Б. Г., Фомичев А. В.** Базовые матричные кристаллы: Проектирование специализированных БИС на их основе. М.: Радио и связь, 1985.

16. **Потемкин И. С.** Автоматизация синтеза функциональных схем. М.: Энергонздат, 1981.

17. **Брахман Т. Р.** Многокритериальность и выбор альтернативы в технике. М.: Радио и связь, 1984.

18. **Потемкин И. С.** Функциональные узлы на потенциальных элементах. М.: Энергия, 1976.

19. **Гитис Э. И., Пискулов Е. А.** Аналого-цифровые преобразователи. М.: Энергоиздат, 1981.

20. **Полупроводниковые** запоминающие устройства и их применение./В. П. Андреев, В. В. Баранов, Н. В. Бекин и др.; Под ред. А. Ю. Гордонова. М.: Радио и связь, 1981.

21. **Косарев Ю. А., Виноградов С. В.** Электрически изменяемые ПЗУ. Л.: Энергоатомиздат. Ленингр. отд-ние, 1985.

22. **Баранов С. И., Скляров В. А.** Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986.

23. **Полупроводниковые** БИС запоминающих устройств: Справочник/В. В. Баранов, Н. В. Бекин, А. Ю. Гордонов и др.; Под ред. А. Ю. Гордонова, Ю. Н. Дьякова. М.: Радио и связь, 1986.

24. **Новиков С. В.** Теория регулярных структур. Минск: Университетское изд-во, 1987.

25. **Хоффман Л.** Современные методы защиты информации: Пер. с англ./Под ред. В. А. Герасименко. М.: Советское радио, 1980.

26. **Карцев М. А., Брик В. А.** Вычислительные системы и синхронная арифметика. М.: Радио и связь, 1981.

27. **Уокерли Дж.** Архитектура и программирование микроЭВМ: Пер. с англ. М.: Мир, 1984. Кн. 1.

28. **Зельдин Е. А.** Цифровые интегральные микросхемы в информационно-измерительной аппаратуре. Л.: Энергоатомиздат. Ленингр. отд-ние, 1986.

29. **Надежность** и контроль ЭВМ/Ю. П. Журавлев, Л. А. Котелюк, Н. И. Цнклинский. М.: Советское радио, 1978.

30. **Каган Б. М.** Электронные вычислительные машины и системы. М.: Энергоатомиздат, 1985.

31. **Пархоменко П. П., Согомонян Е. С.** Основы технической диагностики: Оптимизация алгоритмов диагностирования, аппаратурные средства/Под ред. П. П. Пархоменко. М.: Энергия, 1981.

32. **Автоматизированное** проектирование цифровых устройств/ /С. С. Бадулин, Ю. М. Барнаулов, В. А. Бердышев и др.; Под ред. С. С. Бадулина. М.: Радио и связь, 1981.

33. **Лазер И. М., Шубарев В. А.** Устойчивость цифровых микроэлектронных устройств. М.: Радио и связь, 1983.

34. **Автоматное** управление асинхронными процессами в ЭВМ и дискретных системах/Под ред. В. И. Варшавского. М.: Наука, 1986.

35. **Цифровые** устройства на микросхемах/Под ред. В. Л. Волчека, Е. Г. Ойхмана. М.: Энергия, 1975.

36. **Букреев И. Н., Мансуров Б. М., Горячев В. И.** Микроэлектронные схемы цифровых устройств. М.: Советское радио, 1975.

37. **Зельдин Е. А.** Триггеры. М.: Энергоатомиздат, 1983.

38. **Электронная** вычислительная машина ЕС-1050/Под ред. А. М. Ларионова. М.: Статистика, 1978.

39. **Электронная** вычислительная машина ЕС-1033/Под ред. В. А. Кормаринского, Г. П. Сорокина. М.: Машиностроение, 1982.

40. **Горошков Б. И.** Радиоэлектронные устройства: Справочник. М.: Радио и связь, 1984.

41. **Оберман Р. М.** Счет и счетчики: Пер с англ. М.: Радио и связь, 1984.

42. **Микросхемы** и их применение: Справочное пособие/В. А. Батушев, В. Н. Вениаминов, В. Г. Ковалев и др. М.: Радио и связь, 1985.

43. **Алексенко А. Г., Галицын А. А., Иванников А. Д.** Проектирование радиоэлектронной аппаратуры на микропроцессорах: Программирование, типовые решения, методы отладки. М.: Радио и связь, 1984.

44. **Берглунд.** Проверка кристаллов, плат и системы в целом методом сквозного сдвигового регистра//Электроника. 1979. № 6. С. 35—39.
45. **Комоницки Д.** Полное самотестирование системы — результат синтеза существующих методов//Электроника. 1983. № 5. С. 26—35.
46. **Кирьянов К. Г.** К теории сигнатурного анализа//Техника средств связи. Сер. Радиоизмерительная техника. 1980, Вып. 2. С. 1—46.
47. **Селлерс Ф.** Методы обнаружения ошибок в работе ЭЦВМ: Пер. с англ. М.: Мир, 1972.
48. **Конопелько В. К., Лосев В. В.** Надежное хранение информации в полупроводниковых запоминающих устройствах. М.: Радио и связь, 1986.
49. **Питерсон У., Уэлдон Э.** Коды, исправляющие ошибки: Пер. с англ./Под ред. Р. Л. Добрушина, С. И. Самойленко. М.: Мир, 1976.
50. **Блейхут Р.** Теория и практика кодов, контролирующих ошибки: Пер. с англ. М.: Мир, 1986.
51. **Измерение** параметров цифровых интегральных микросхем//Д. Ю. Эйдукас, Б. В. Орлов, Л. М. Попель и др.; Под ред. Д. Ю. Эйдукаса, Б. В. Орлова. М.: Радио и связь, 1982.
52. **Смирнов Н. И., Стручков А. А., Судовцев В. А.** Диагностика неисправностей в цифровой аппаратуре на БИС//Зарубежная радиоэлектроника. 1979. № 1, С. 53—60
53. **Захаров В. Н., Поспелов Д. А., Хазацкий В. Е.** Системы управления. Задание. Проектирование. Реализация. М.: Энергия, 1972.
54. **Каган Б. М., Сташин В. В.** Микропроцессоры в цифровых системах. М.: Энергия, 1979.
55. **Майоров С. А., Новиков Г. И.** Принципы организации цифровых машин. Л.: Машиностроение. Ленингр. отд-ние, 1974.
56. **Хоуп Г.** Проектирование цифровых вычислительных устройств на интегральных схемах: Пер. с англ. М.: Мир, 1984.
57. **Титце У., Шенк К.** Полупроводниковая схемотехника: Пер. с нем./Под ред. А. Г. Алексенко. М.: Мир, 1983.
58. **Пупырев Е. И.** Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984.
59. **Угрюмов Е. П.** Проектирование элементов и узлов ЭВМ. М.: Высшая школа, 1987.
60. **Горяшко А. П.** Синтез диагностируемых схем вычислительных устройств. М.: Наука, 1987.