СОВРЕМЕННАЯ СХЕМОТЕХНИКА

Ю. В. Новиков

ОСНОВЫ ЦИФРОВОЙ СХЕМОТЕХНИКИ

БАЗОВЫЕ ЭЛЕМЕНТЫ И СХЕМЫ МЕТОДЫ ПРОЕКТИРОВАНИЯ

Издательство «МИР»

СОВРЕМЕННАЯ СХЕМОТЕХНИКА

Ю. В. Новиков

ОСНОВЫ ЦИФРОВОЙ СХЕМОТЕХНИКИ

БАЗОВЫЕ ЭЛЕМЕНТЫ И СХЕМЫ МЕТОДЫ ПРОЕКТИРОВАНИЯ

Москва «МИР» 2001

УДК 621.374.1
ББК 32.85
Н73

Новиков Ю. В.

Н73 Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. — М.: Мир, 2001. — 379 с., ил. — (Современная схемотехника)

ISBN 5-03-003449-8

Книга представляет собой учебник по основам цифровой схемотехники. Рассматриваются основы схемотехники цифровых устройств, которыми должен свободно владеть и активно пользоваться каждый профессиональный разработчик цифровой аппаратуры. Обсуждается функционирование и взаимодействие всех основных типов цифровых микросхем — от самых простых до самых сложных. Описываются модели и уровни представления цифровых микросхем, используемых при проектировании цифровых электронных систем, способы оптимального построения высокоэффективных цифровых систем самой различной степени сложности. Книга позволяет освоить азбуку цифровой схемотехники даже читателям с начальным уровнем знаний по электронике. Усвоению материала помогает большое количество конкретных примеров построения самых различных цифровых устройств.

Для студентов, преподавателей и профессиональных разработчиков цифровых электронных систем.

ББК 32.85

Редакция литературы по информатике и новой технике

ISBN 5-03-003449-8

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Цель этой книги — помочь в изучении основ цифровой схемотехники любому желающему — от студента до профессионала. Причем речь в данном случае идет не о физических основах электроники, не о технологии производства электронных компонентов, не об электронике или схемотехнике вообще, а именно о цифровой схемотехнике с ее специфическими особенностями.

В книге рассмотрены многие важные вопросы цифровой схемотехники, которым в литературе обычно уделяется недостаточно внимания. Видимо, авторы книг считают их чем-то само собой разумеющимся или полагают, что читатели сами должны сформулировать эти вопросы и сами же найти на них ответы. В результате нередки случаи, когда даже выпускники институтов по специальностям, связанным с электроникой, не могут самостоятельно разработать более или менее сложное нестандартное цифровое устройство. После изучения данной книги читатель будет обладать необходимыми навыками для проектирования как простейших узлов, так и довольно сложных устройств.

Книга написана сжато, последовательно и доступно и требует минимального обращения к другим источникам. Часть материала, представленного в ней, написана на основе лекций, читаемых автором в МИФИ.

Конечно, в такой небольшой книге невозможно рассмотреть все компоненты цифровой электроники и все приемы проектирования цифровой схемотехники. Предполагается, что она станет первой в серии книг по цифровой схемотехнике. В последующих книгах серии будут рассмотрены, в частности, следующие темы:

- схемотехника устройств на основе программируемой логики;
- схемотехника микропроцессорных устройств;
- схемотехника устройств на основе микроконтроллеров;
- схемотехника персональных компьютеров;
- схемотехника компьютерных систем измерения, контроля и управления;
- аппаратно-программные средства отладки и контроля цифровых устройств;
- программные средства проектирования цифровых устройств.

Ю. В. Новиков
6 июня 2001 г.

ВВЕДЕНИЕ

Цифровая электроника в настоящее время все более и более вытесняет традиционную аналоговую. Ведущие фирмы, производящие самую разную электронную аппаратуру, все чаще заявляют о полном переходе на цифровую технологию. Причем это относится как к бытовой технике (аудио-, видеоаппаратура, средства связи), так и к профессиональной технике (измерительная, управляющая аппаратура). Ставшие уже привычными персональные компьютеры также полностью реализованы на основе цифровой технологии. Видимо, в ближайшем будущем полностью аналоговые устройства будут применяться только в тех редких случаях, когда требуется получить рекордные значения некоторых параметров электронных устройств (например, быстродействия).

Между тем литературы, позволяющей самостоятельно изучить основы и главные методы цифровой схемотехники, освоить основные практические приемы проектирования цифровых устройств, явно недостаточно. Книги, которые претендуют на освещение основ цифровой схемотехники, можно разделить на три большие группы.

К первой группе относятся книги, которые стремятся охватить всю электронику в целом (как цифровую, так и аналоговую). Такой глобальный подход неизбежно приводит к тому, что цифровая схемотехника рассматривается чересчур кратко, поверхностно. Даже самые лучшие из книг этой группы уделяют цифровой электронике не более четверти своего объема. Между тем цифровая электроника существенно отличается от аналоговой не только видом используемых сигналов, но, что самое главное, приемами проектирования, требуемым стилем мышления разработчика, принципами построения сложных систем. Данные книги хороши в основном для повышения общего уровня образования читателя, для создания общего базиса, на котором только и может вырасти настоящий специалист, разработчик электронной аппаратуры.

Ко второй группе относятся книги, посвященные цифровым микросхемам и их применению. Эти книги стремятся описать как можно более подробно все имеющиеся микросхемы, поэтому обязательно содержат большой справочный

материал. Однако в любом случае приводимые в них справочные данные далеко не полны и к тому же неизбежно содержат большое количество ошибок. Справочников по микросхемам должно быть как можно меньше, и они должны быть очень подробными, иначе они просто не имеют смысла. В идеале справочники должны выпускаться только фирмами-изготовителями микросхем. А попытка ввести в книгу даже краткие справочные данные по всем существующим микросхемам не оставляет достаточно места для описания самого главного — разнообразных применений этих микросхем, методам проектирования цифровых узлов и устройств на их основе. Поэтому книги этой группы годятся обычно только для первичного знакомства с темой.

Наконец, третья группа книг описывает готовые цифровые устройства различного назначения. В основном такие книги рассчитаны на «радиолюбителей», хотя обычно не совсем понятно, что подразумевается под этим понятием. Среди тех, кто называют себя радиолюбителями, немало прекрасных разработчиков, а среди дипломированных профессионалов встречаются те, кто не может разработать даже простейшей схемы. Книги этой третьей группы, как правило, не говорят о том, как разрабатывались те или иные устройства, какие приемы проектирования применялись и почему. Поэтому они могут сформировать у неподготовленного читателя привычку к бездумному повторению готовых решений, использованию шаблонов. Настоящую пользу данные книги могут принести только тому, у кого уже есть хороший базовый запас знаний и умений по проектированию цифровых устройств. Впрочем, для таких людей чужие схемы не слишком интересны, они могут проектировать свои.

К недостаткам многих существующих книг можно отнести также чрезмерное увлечение описание тонкостей физических процессов, лежащих в основе цифровой электроники, и подробное рассмотрение особенностей технологии микросхем. Конечно, все это тоже нужно, полезно, интересно, но с самой схемотехникой связано не жестко, не прямо и не слишком сильно. Разработчик цифровой аппаратуры в подавляющем большинстве случаев работает с микросхемами как с «черным ящиком», ему не слишком важно, что происходит внутри, как реализуется та или иная функция микросхемы. Видимо, целесообразно фи-

зику полупроводников и технологию изучать отдельно от схемотехники, параллельно с ней, до нее или после нее, чтобы не смешивать две различные области знаний.

Наконец, еще одним недостатком книг о цифровой схемотехнике можно назвать стремление охватить в одной книге все области цифровой электроники: от логических элементов до компьютеров и других сложных электронных систем. В результате все вопросы рассматриваются недостаточно глубоко, и до практики разработки дело не доходит. К тому же проектирование интеллектуальных устройств довольно сильно отличается от проектирования устройств на жесткой логике. Эти направления цифровой схемотехники требуют применения совершенно разных подходов, и поэтому их целесообразно рассматривать в различных книгах.

Данная книга посвящена самым основам цифровой схемотехники, ее азбуке, ее основным методам, подходам и приемам. Отличие ее состоит в том, что она может дать представление о цифровой схемотехнике даже тем читателям, которые имеют слабое представление об электронике вообще. Впрочем, это вовсе не означает, что книга посвящена только самым общим, неконкретным, отвлеченным вопросам. Наоборот, она призвана сформировать действительно хорошего проектировщика, способного строить высокоэффективные цифровые системы самой различной сложности и четко представляющего себе взаимосвязь всех процессов в этих системах сверху донизу. Конечно, чтобы стать таким классным специалистом, нужны определенные способности, даже талант, но строить простейшие цифровые устройства, пусть и не оптимальные, не уникальные, но полезные и вполне работоспособные, способен практически каждый человек. Хотелось бы надеяться, что данная книга окажет в этом деле действенную помощь.

Книга ни в коем случае не претендует на то, чтобы заменить собой справочники по микросхемам, хотя в ней и разъясняются функции многих микросхем. Главное, чему уделено внимание — это применение микросхем для различных задач (причем как стандартное применение, так и не стандартное), а также приемы объединения, комбинирования микросхем, позволяющие создавать разнообразные узлы, устройства и системы.

Материал данной книги представляет собой тот необходимый минимум знаний, который должен иметь и которым дол-

жен свободно и активно пользоваться каждый профессиональный разработчик цифровой аппаратуры. Любые другие, дополнительные знания, конечно же, не повредят, но заменить собой то, что изложено здесь, они не смогут.

Возможно, подход, предлагаемый в данной книге, несколько отличается от общепринятого. Возможно также, что используемый набор терминов не полностью совпадает со стандартным (отечественные стандарты слишком часто меняются). Но главное — это научить проектировать цифровые устройства и системы, а какие для этого используются подходы и термины, наверное, не слишком принципиально.

Книга написана на основе многолетнего личного опыта автора по разработке цифровых устройств, а также на базе материала учебных курсов, читаемых автором, доцентом кафедры электроники Московского инженерно-физического института (МИФИ).

Несколько слов о структуре книги.

Первая глава рассматривает основополагающие принципы цифровой электроники, знакомит с терминологией и основными правилами оформления схем.

Главы со второй по шестую посвящены основным базовым элементам цифровых устройств, типовым и нестандартным схемам их включения. Практически все приведенные схемы проверялись автором на практике. Микросхемы описаны начиная с самых простейших логических элементов в порядке усложнения через комбинационные микросхемы, триггеры, регистры, счетчики до микросхем памяти. Каждая глава включает в себя множество примеров включений микросхем, как рассматриваемых в данной главе, так и рассмотренных в предыдущих главах.

Седьмая глава содержит краткие сведения о цифро-аналоговых и аналого-цифровых преобразователях и основных методах их включения в аналого-цифровых устройствах. Без этих сведений книга была бы неполна.

В восьмой главе приводятся примеры нескольких сравнительно сложных цифровых устройств с подробным описанием всех этапов проектирования и принципов работы и взаимодействия всех узлов и микросхем. Цель этого не в том, чтобы читатель повторил данные устройства, а в том, чтобы на практике показать приемы проектирования, которые затем позволят строить любые другие цифровые устройства.

В приложении приведены таблицы параметров микросхем, таблицы основных обозначений микросхем и сигналов, таблицы соответствия отечественных и зарубежных микросхем. В конце книги имеется подробный словарь терминов и сокращений цифровой схемотехники.

В книгу не вошли материалы по основам микропроцессорной техники, по принципам устройства и применения микроконтроллеров и персональных компьютеров, по основам работы с программируемыми микросхемами (ПЛИС), по методам контроля и отладки цифровых устройств, а также по программным средствам, предназначенным для проектирования цифровых устройств и систем. Все эти вопросы планируется рассмотреть в других книгах данной серии.

Глава 1
ФИЛОСОФИЯ ЦИФРОВОЙ ЭЛЕКТРОНИКИ

Пусть не пугает читателя слово «философия» в названии главы. В данном случае имеются в виду всего лишь главные принципы цифровой электроники и обоснование ее преимуществ.

1.1. Аналог или цифра?

Для начала дадим несколько базовых определений.

Сигнал — это любая физическая величина (например, температура, давление воздуха, интенсивность света, сила тока и т. д.), изменяющаяся со временем. Именно благодаря этому изменению во времени сигнал может нести в себе какую-то информацию.

Электрический сигнал — это электрическая величина (например, напряжение, ток, мощность), изменяющаяся со временем. Вся электроника в основном работает с электрическими сигналами, хотя в последнее время все больше используются световые сигналы, которые представляют собой изменяющуюся во времени интенсивность света.

Аналоговый сигнал — это сигнал, который может принимать любые значения в определенных пределах (например, напряжение может плавно изменяться в пределах от нуля до десяти вольт). Устройства, работающие только с аналоговыми сигналами, называются аналоговыми устройствами.

Цифровой сигнал — это сигнал, который может принимать только два значения (иногда — три значения). Причем разрешены некоторые отклонения от этих значений (рис. 1.1). Например, напряжение может принимать два значения: от 0 до 0,5 В (уровень нуля) или от 2,5 до 5 В (уровень единицы). Устройства, работающие исключительно с цифровыми сигналами, называются цифровыми устройствами.

В природе практически все сигналы аналоговые, то есть они изменяются непрерывно в некоторых пределах. Именно поэтому первые электронные устройства были аналоговыми. Они преобразовывали физические величины в пропорциональные им напряжение или ток, выполняли над ними какие-то операции и затем выполняли обратные преобразования в физические величины. Например, голос человека (колебания воздуха) с помощью микрофона преобразуется в электрические колебания, затем эти электрические сигналы усиливаются электронным усилителем и с помощью акустической системы снова преобразуются в колебания воздуха, в более сильный звук.

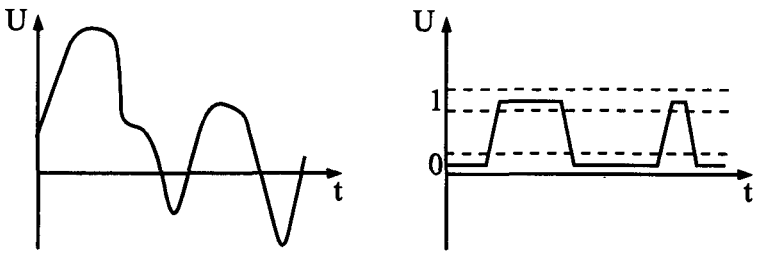


Рис. 1.1. Электрические сигналы: аналоговый (слева) и цифровой (справа).

Однако аналоговые сигналы и работающая с ними аналоговая электроника имеют большие недостатки, связанные именно с природой аналоговых сигналов. Дело в том, что аналоговые сигналы чувствительны к действию всевозможных паразитных сигналов — шумов, наводок, помех. Шум — это внутренние хаотические слабые сигналы любого электронного устройства (микрофона, транзистора, резистора и т. д.). Наводки и помехи — это сигналы, приходящие на электронную систему извне и искажающие полезный сигнал (например, электромагнитные излучения от радиопередатчиков или трансформаторов).

Все операции, производимые электронными устройствами над сигналами, можно условно разделить на три большие группы:

- обработка (или преобразование);
- передача;
- хранение.

Во всех этих случаях полезные сигналы искажаются паразитными сигналами — шумами, помехами, наводками. Кроме того, при обработке сигналов (например, при усилении, фильтрации) еще искажается и их форма из-за несовершенства, неидеальности электронных устройств. А при передаче на большие расстояния и при хранении сигналы к тому же ослабляются.

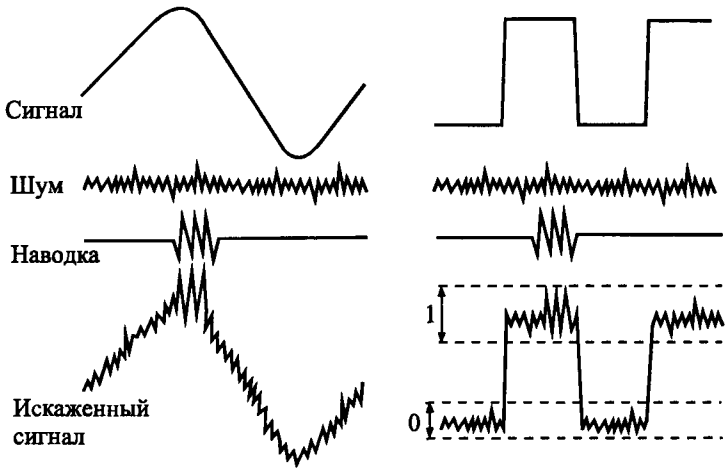


Рис. 1.2. Искажение шумами и наводками аналогового сигнала (слева) и цифрового сигнала (справа).

В случае аналоговых сигналов все это существенно ухудшает полезный сигнал, так как все его значения разрешены (рис. 1.2). Поэтому каждое преобразование, каждое промежуточное хранение, каждая передача по кабелю или эфиру ухудшает аналоговый сигнал, иногда вплоть до его полного уничтожения. Надо еще учесть, что все шумы, помехи и наводки принципиально не поддаются точному расчету, поэтому *точно* описать поведение любых аналоговых устройств абсолютно невозможно. К тому же со временем параметры всех аналоговых устройств изменяются из-за старения элементов, поэтому характеристики этих устройств не остаются постоянными.

В отличие от аналоговых, цифровые сигналы, имеющие всего два разрешенных значения, защищены от действия шумов, наводок и помех гораздо лучше. Небольшие отклонения от раз-

решенных значений никак не искажают цифровой сигнал, так как всегда существуют зоны допустимых отклонений (рис. 1.2). Именно поэтому цифровые сигналы допускают гораздо более сложную и многоступенчатую обработку, гораздо более длительное хранение без потерь и гораздо более качественную передачу, чем аналоговые. К тому же поведение цифровых устройств всегда можно абсолютно точно рассчитать и предсказать. Цифровые устройства гораздо меньше подвержены старению, так как небольшое изменение их параметров никак не отражается на их функционировании. Кроме того, цифровые устройства проще проектировать и отлаживать. Понятно, что все эти преимущества обеспечивают бурное развитие цифровой электроники.

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

Кроме того, цифровой сигнал передает информацию только двумя уровнями и изменением одного своего уровня на другой, а аналоговый передает информацию еще и каждым текущим значением своего уровня, то есть он более емкий с точки зрения передачи информации. Поэтому для передачи того объема полезной информации, который содержится в одном аналоговом сигнале, чаще всего приходится использовать несколько цифровых сигналов (обычно от 4 до 16).

К тому же, как уже отмечалось, в природе все сигналы аналоговые, то есть для преобразования их в цифровые сигналы и для обратного преобразования требуется применение специаль-

ной аппаратуры (аналого-цифровых и цифро-аналоговых преобразователей). Так что ничто не дается даром, и плата за преимущества цифровых устройств может порой оказаться неприемлемо большой.

1.2. Модели и уровни представления цифровых устройств

Все цифровые устройства строятся из логических микросхем, каждая из которых (рис. 1.3) обязательно имеет следующие выводы (или, как их еще называют в просторечии, ножки):

- выводы питания: общий (или «земля») и напряжения питания (в большинстве случаев +5 В или +3,3 В), которые на схемах обычно не показываются;
- выводы для входных сигналов (или «входы»), на которые поступают внешние цифровые сигналы;
- выводы для выходных сигналов (или «выходы»), на которые выдаются цифровые сигналы из самой микросхемы.

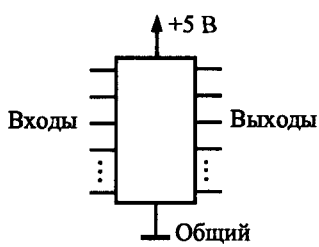


Рис. 1.3. Цифровая микросхема.

Каждая микросхема преобразует тем или иным способом последовательность входных сигналов в последовательность выходных сигналов. Способ преобразования чаще всего описывается или в виде таблицы (так называемой таблицы истинности) или в виде временных диаграмм, то есть графиков зависимости от времени всех сигналов.

Все цифровые микросхемы работают с логическими сигналами, имеющими два разрешенных уровня напряжения. Один из этих уровней называется уровнем логической единицы (или

единичным уровнем), а другой — уровнем логического нуля (или нулевым уровнем). Чаще всего логическому нулю соответствует низкий уровень напряжения, а логической единице — высокий уровень напряжения. В этом случае говорят, что принята «положительная логика». Однако при передаче сигналов на большие расстояния и в системных шинах микропроцессорных систем порой используют и обратное представление, когда логическому нулю соответствует низкий уровень напряжения, а логической единице — высокий уровень. В этом случае говорят об «отрицательной логике». Иногда логический нуль кодируется положительным уровнем напряжения (тока), а логическая единица — отрицательным уровнем напряжения (тока) или наоборот. Есть и более сложные методы кодирования логических нулей и единиц. Но мы в основном будем говорить о положительной логике.

Для описания работы цифровых устройств используют самые различные модели, отличающиеся друг от друга сложностью, точностью, большим или меньшим учетом тонких физических эффектов. В основном эти модели используются при компьютерных расчетах цифровых схем. В настоящее время существуют компьютерные программы, которые не только рассчитывают готовые схемы, но способны и проектировать новые схемы по формализованным описаниям функций, которые данное устройство должно выполнять. Это довольно удобно, но ни одна программа никогда не может сравниться с человеком. По-настоящему эффективные, оптимизированные по числу используемых аппаратурных модулей, наконец, красивые схемы может разрабатывать только человек, который всегда подходит к проектированию творчески и использует оригинальные идеи.

Разработчик цифровой аппаратуры тоже использует своеобразные модели или, как еще можно сказать, различные уровни представления цифровых схем. Но в отличие от компьютера человек может гибко выбирать нужную модель, ему надо только взглянуть на схему, чтобы понять, где достаточно простейшей модели, а где требуется более сложная. То есть человек никогда не будет делать лишней, избыточной работы и, следовательно, не будет вносить дополнительных ошибок, свойственных любой, даже самой сложной, модели. Правда, простота цифровых устройств по сравнению с аналоговыми устройствами обычно не дает возможности сделать чересчур серьезные ошибки.

В подавляющем большинстве случаев разработчики цифровых схем используют три модели, три уровня представления о работе цифровых устройств.

1. Логическая модель.
2. Модель с временными задержками.
3. Модель с учетом электрических эффектов (или электрическая модель).

Опыт показывает, что первая, простейшая модель оказывается достаточной примерно в 20% всех случаев. Она применима для всех цифровых схем, работающих с низкой скоростью, в которых быстродействие не принципиально. Привлечение второй модели, учитывающей задержки срабатывания логических элементов, позволяет охватить около 80% всех возможных схем. Ее применение необходимо для всех быстродействующих устройств и в случае одновременного изменения нескольких входных сигналов. Наконец, добавление третьей модели, учитывающей входные и выходные токи, входные и выходные сопротивления и емкости элементов, позволяет проектировать практически 100% цифровых схем. В первую очередь эту третью модель надо применять при объединении нескольких входов и выходов, при передаче сигналов на большие расстояния и при нетрадиционном включении логических элементов (с переводом их в аналоговый, в линейный режим).

Таблица 1.1. Таблица истинности инвертора

Вход	Выход
0	1
1	0

Для иллюстрации работы перечисленных моделей рассмотрим работу самого простейшего логического элемента — инвертора. Инвертор изменяет (инвертирует) логический уровень входного сигнала на противоположный уровень выходного сигнала или, как еще говорят, изменяет полярность логического сигнала. Таблица истинности инвертора (табл. 1.1) элементарно проста, так как возможны только две ситуации: нуль на входе или единица на входе. На рис. 1.4 показано, как будет выглядеть выходной сигнал инвертора при использовании трех его моде-

лей (трех уровней его представления). Такие графики логических сигналов называются временными диаграммами, они позволяют лучше понять работу цифровых схем.

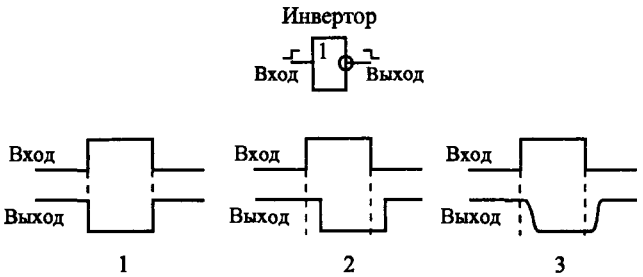


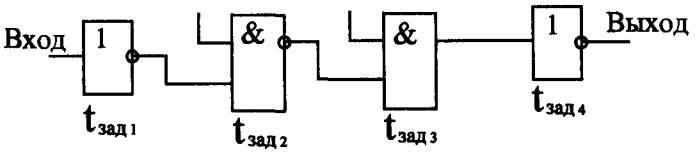
Рис. 1.4. Три уровня представления цифровых устройств.

Из рисунка видно, что в первой, логической, модели считается, что элемент срабатывает мгновенно, любое изменение уровня входного сигнала сразу же, без всякой задержки приводит к изменению уровня выходного сигнала. Во второй модели выходной сигнал изменяется с некоторой задержкой относительно входного. Наконец, в третьей модели выходной сигнал не только задерживается по сравнению с входным, но и его изменение происходит не мгновенно, процесс смены уровней сигнала (или, как говорят, *фронт сигнала*) имеет конечную длительность. Кроме того, третья модель учитывает изменение уровней логических сигналов.

На практике разработчик, как правило, в начале проектирования пользуется исключительно первой моделью, а затем для некоторых узлов применяет вторую модель или (реже) еще и третью модель. При этом первая модель не требует вообще никаких цифровых расчетов, для нее достаточно только знание таблиц истинности или алгоритмов функционирования микросхем. Вторая модель предполагает расчет (по сути, суммирование) временных задержек элементов на пути прохождения сигналов (рис. 1.5). В результате этого расчета может выясниться, что в схему нужно внести изменения.

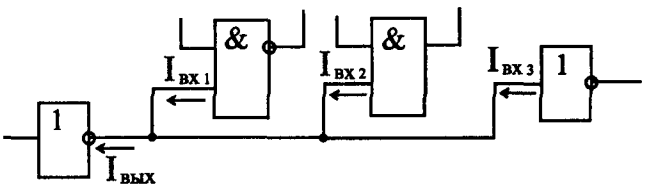
Расчеты по третьей модели могут быть различными, в том числе и довольно сложными, но в большинстве случаев они все-таки сводятся всего лишь к суммированию входных и выходных токов логических элементов (рис. 1.6). В результате этих

расчетов может выясниться, что требуется применение микросхем с более мощными выходами или включение дополнительных элементов.



$$t_{зад} = t_{зад\,1} + t_{зад\,2} + t_{зад\,3} + t_{зад\,4}$$

Рис. 1.5. Суммирование задержек элементов.



$$I_{вых} = I_{вх\,1} + I_{вх\,2} + I_{вх\,3}$$

Рис. 1.6. Суммирование входных токов элементов.

Таким образом, проектирование цифровых устройств принципиально отличается от проектирования аналоговых устройств, при котором сложные расчеты абсолютно неизбежны. Разработчик цифровых устройств имеет дело только с логикой, с логическими сигналами и с алгоритмами работы цифровых микросхем. А что происходит внутри этих микросхем, для него практически не имеет значения.

Справочные данные на цифровые микросхемы обычно содержат большой набор параметров, каждый из которых можно отнести к одному из трех перечисленных уровней представления, к одной из трех моделей.

Например, таблица истинности микросхемы (для простых микросхем) или описание алгоритма ее работы (для более сложных микросхем) относится к первому, логическому уровню. Поэтому знать их наизусть каждому разработчику необходимо в любом случае.

Величины задержек логических сигналов между входами и выходами относятся ко второму уровню представления. Типичные величины задержек составляют от единиц наносекунд (1 нс = 10^{-9} с) до десятков наносекунд. Величины задержек для разных микросхем могут быть различными, поэтому в справочниках всегда указывается максимальное значение задержки. Необходимо также помнить, что задержка при переходе выходного сигнала из единицы в нуль (t_{PHL}), как правило, отличается от задержки при переходе выходного сигнала из нуля в единицу (t_{PLH}). Например, для одной и той же микросхемы $t_{PLH} < 11$ нс, а $t_{PHL} < 8$ нс. Здесь английская буква P означает Propagation (распространение), L означает Low (низкий уровень сигнала, нуль), а H — High (высокий уровень сигнала, единица). Количество величин задержек, определяемых справочником для микросхемы, может изменяться от двух до нескольких десятков.

Уровни входных и выходных токов, а также уровни входных и выходных напряжений относятся к третьему уровню представления.

Входной ток микросхемы при приходе на вход логического нуля (I_{IL}), как правило, отличается от входного тока при приходе на вход логической единицы (I_{IH}). Например, $I_{IL} = -0{,}1$ мА, а $I_{IH} = 20$ мкА (считается, что положительный ток втекает во вход микросхемы, а отрицательный — вытекает из него). Точно так же выходной ток микросхемы при выдаче логического нуля (I_{OL}) может отличаться (и обычно отличается) от выходного тока при выдаче логической единицы (I_{OH}). Например, для одной и той же микросхемы $I_{OH} < -0{,}4$ мА, а $I_{OL} < 8$ мА (считается, что положительный ток втекает в выход микросхемы, а отрицательный — вытекает из него). Надо также учитывать, что разные входы и выходы одной и той же микросхемы могут иметь различные входные и выходные токи.

Для выходных напряжений логического нуля (U_{OL}) и единицы (U_{OH}) в справочниках обычно задаются предельно допустимые значения при заданной величине выходного тока. При этом чем больше выходной ток, тем меньше напряжение логической единицы и тем больше напряжение логического нуля. Например, $U_{OH} > 2{,}5$ В (при $I_{OH} < -0{,}4$ мА), а $U_{OL} < 0{,}5$ В (при $I_{OL} < 8$ мА).

Задаются в справочниках также и допустимые уровни входных напряжений, которые микросхема еще воспринимает как

правильные логические уровни нуля и единицы. Например, $U_{IH} > 2{,}0$ В, $U_{IL} < 0{,}8$ В. Как правило, входные напряжения логических сигналов не должны выходить за пределы напряжения питания.

В обозначениях напряжений и токов буква I означает Input (вход), буква O означает Output (выход), L — Low (нуль), а H — High (единица).

К третьему уровню представления относятся также величины внутренней емкости входов микросхемы (обычно от единиц до десятков пикофарад) и допустимая величина емкости, к которой может подключаться выход микросхемы, то есть емкость нагрузки C_L (порядка 100 пФ). Отметим, что 1 пФ = 10^{-12} Ф. На этом же уровне представления задаются максимально допустимые величины длительности положительного фронта (t_{LH}) и отрицательного фронта (t_{HL}) входного сигнала, например $t_{HL} < 1{,}0$ мкс, $t_{LH} < 1{,}0$ мкс. То есть при большей длительности перехода входного сигнала из единицы в нуль и из нуля в единицу микросхема может работать неустойчиво, неправильно, нестандартно.

К третьему уровню представления можно отнести также такие параметры, как допустимое напряжение питания микросхемы (U_{CC}) и максимальный ток, потребляемый микросхемой (I_{CC}). Например, может быть задано:

$$4{,}5 \text{ В} < U_{CC} < 5{,}5 \text{ В}; \quad I_{CC} < 100 \text{ мА}.$$

При этом потребляемый ток I_{CC} зависит от уровней выходных токов микросхемы I_{OH} и I_{OL}. Эти параметры надо учитывать при выборе источника питания для проектируемого устройства, а также в процессе изготовления печатных плат при выборе ширины токоведущих дорожек.

Наконец, к третьему уровню относится ряд параметров, которые часто упоминаются в литературе, но не всегда приводятся в справочных таблицах:

- *Порог срабатывания* — уровень входного напряжения, выше которого сигнал воспринимается как единица, а ниже — как нуль. Для наиболее распространенных ТТЛ микросхем он примерно равен 1,3...1,4 В.

- *Помехозащищенность* — параметр, характеризующий величину входного сигнала помехи, накладывающегося на входной сигнал, который еще не может изменить состояние выходных сигналов. Помехозащищенность определяется разницей между напряжением U_{IH} и порогом срабатывания (это помехозащищенность единичного уровня), а также разницей между порогом срабатывания и U_{IL} (это помехозащищенность нулевого уровня).

- *Коэффициент разветвления* — число входов, которое может быть подключено к данному выходу без нарушения работы. Этот параметр определяется отношением выходного тока к входному. Стандартная величина коэффициента разветвления при использовании микросхем одного типа (одной серии) равна 10.

- *Нагрузочная способность* — параметр выхода, характеризующий величину выходного тока, которую может выдать в нагрузку данный выход без нарушения работы. Чаще всего нагрузочная способность прямо связана с коэффициентом разветвления.

Таким образом, большинство справочных параметров микросхемы относятся к третьему уровню представления (к модели с учетом электрических эффектов), поэтому в большинстве случаев (до 80%) знать их точные значения наизусть не обязательно. Достаточно знать примерные типовые значения параметров для данной серии микросхем.

1.3. Входы и выходы цифровых микросхем

Характеристики и параметры входов и выходов цифровых микросхем определяются прежде всего технологией и схемотехникой внутреннего строения микросхем. Но для разработчика цифровых устройств любая микросхема представляет собой всего лишь «черный ящик», внутренности которого знать не обязательно. Ему важно только четко представлять себе, как поведет себя та или иная микросхема в данном конкретном включении, будет ли она правильно выполнять требуемую от нее функцию.

Наибольшее распространение получили две технологии цифровых микросхем:

- ТТЛ (TTL) и ТТЛШ (TTLS) — биполярная транзисторно-транзисторная логика и ТТЛ с диодами Шоттки;
- КМОП (CMOS) — комплементарные транзисторы со структурой «металл-окисел-полупроводник».

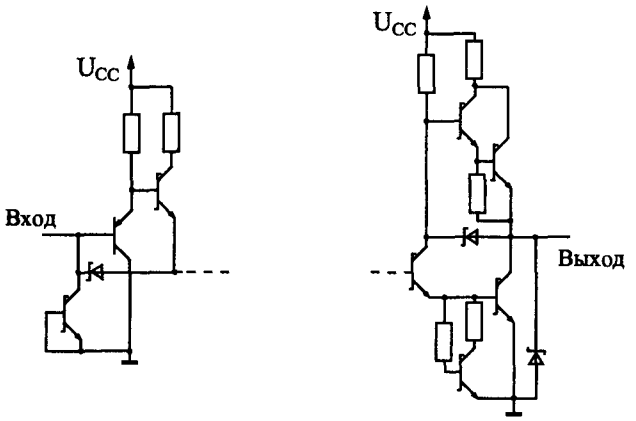


Рис. 1.7. Входной и выходной каскады микросхем ТТЛШ.

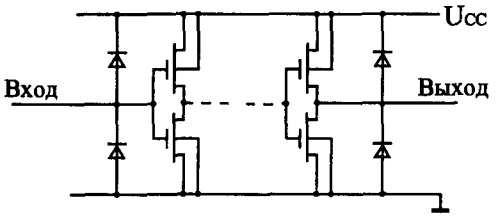


Рис. 1.8. Входной и выходной каскады микросхем КМОП.

Различаются они типами используемых транзисторов и схемотехническими решениями внутренних каскадов микросхем. Отметим также, что микросхемы КМОП потребляют значительно меньший ток от источника питания, чем такие же микросхемы ТТЛ (или ТТЛШ), правда, только в статическом режиме или на небольших рабочих частотах. На рис. 1.7 и 1.8 показаны

примеры схем входных и выходных каскадов микросхем, выполненных по этим технологиям. Понятно, что точный учет всех эффектов в этих схемах, включающих в себя множество транзисторов, диодов и резисторов, крайне сложен, но обычно он просто не нужен разработчику цифровых схем.

Рассмотрим сначала **входы** микросхем.

На первом уровне представления (логическая модель) и на втором уровне представления (модель с временными задержками) о входах микросхем вообще ничего знать не надо. Вход рассматривается как бесконечно большое сопротивление, никак не влияющее на подключенные к нему выходы. Правда, количество входов, подключаемых к одному выходу, влияет на задержку распространения сигнала, но, как правило, незначительно, поэтому это влияние учитывается редко.

Даже на третьем уровне представления (электрическая модель) в большинстве случаев не надо знать о внутреннем строении микросхемы, о схемотехнике входов. Достаточно считать, что при подаче на вход сигнала логического нуля из этого входа вытекает ток, не превышающий I_{IL}, а при подаче сигнала логической единицы в этот вход втекает ток, не превышающий I_{IH}. А для правильной логики работы микросхемы достаточно, чтобы уровень напряжения входного сигнала логического нуля был меньше U_{IL}, а уровень напряжения входного сигнала логической единицы был больше U_{IH}.

Особым случаем является ситуация, когда какой-нибудь вход не подключен ни к одному из выходов, ни к общему проводу, ни к шине питания (так называемый *висящий вход*). Иногда возможности микросхемы используются не полностью, и на некоторые входы не подаются сигналы. Однако при этом микросхема может не работать или работать нестабильно, так как ее правильное включение подразумевает наличие на всех входах логических уровней, пусть даже и неизменных. Поэтому рекомендуется подавать на неиспользуемые входы напряжение питания микросхемы U_{CC} или подключать их к общему проводу (земле) в зависимости от того, какой логический уровень необходим на этом входе. Но для некоторых серий микросхем, выполненных по технологии ТТЛ (например, К155 или КР531), неиспользуемые входы надо подключать к шине питания не прямо, а только через резистор величиной около 1 кОм (достаточно одного резистора на 20 входов).

На неподключенных входах микросхем ТТЛ формируется напряжение около 1,5—1,6 В, которое иногда называют висячим потенциалом. Обычно этот уровень воспринимается микросхемой как сигнал логической единицы, но надеяться на это не стоит. Потенциал, образующийся на неподключенных входах микросхем КМОП, может восприниматься микросхемой и как логический нуль, и как логическая единица. В любом случае все входы надо куда-то подключать. Неподключенными допускается оставлять только те входы (ТТЛ, а не КМОП), состояние которых в данном включении микросхемы не имеет значения.

Выходы микросхем принципиально отличаются от входов тем, что учет их особенностей необходим даже на первом и втором уровнях представления.

Существуют три разновидности выходных каскадов, существенно различающихся как по своим характеристикам, так и по областям применения:

- стандартный выход или выход с двумя состояниями (обозначается 2С, 2S или, реже, ТТЛ, TTL);
- выход с открытым коллектором (обозначается ОК, ОС);
- выход с тремя состояниями или (что то же самое) с возможностью отключения (обозначается 3С, 3S).

Стандартный выход 2С имеет всего два состояния: логический нуль и логическую единицу, причем оба этих состояния активны, то есть выходные токи в обоих этих состояниях (I_{OL} и I_{OH}) могут достигать заметных величин. На первом и втором уровнях представления такой выход можно считать состоящим из двух выключателей, которые замыкаются по очереди (рис. 1.9), причем замкнутому верхнему выключателю соответствует логическая единица на выходе, а замкнутому нижнему — логический нуль.

Выход с открытым коллектором ОК тоже имеет два возможных состояния, но только одно из них (состояние логического нуля) активно, то есть обеспечивает большой втекающий ток I_{OL}. Второе состояние сводится, по сути, к тому, что выход полностью отключается от присоединенных к нему входов. Это состояние может использоваться в качестве логической единицы, но для этого между выходом ОК и напряжением питания необходимо подключить нагрузочный резистор R (так называе-

мый pull-up) величиной порядка сотен Ом. На первом и втором уровнях представления такой выход можно считать состоящим из одного выключателя (рис. 1.9), замкнутому состоянию которого соответствует сигнал логического нуля, а разомкнутому — отключенное, пассивное состояние. Правда, от величины резистора R зависит время переключения выхода из нуля в единицу, что влияет на задержку t_{LH}, но при обычно используемых номиналах резисторов это не слишком важно.

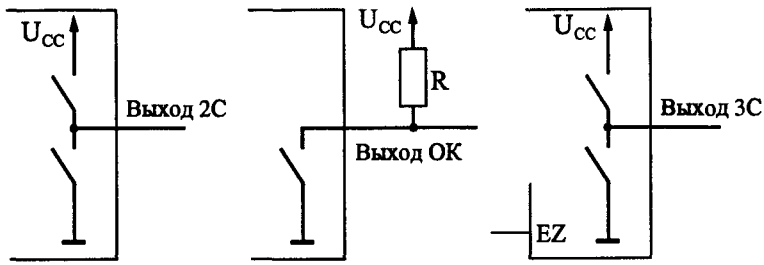


Рис. 1.9. Три типа выходов цифровых микросхем.

Наконец, выход с тремя состояниями 3С очень похож на стандартный выход, но к двум состояниям добавляется еще и третье — пассивное, в котором выход можно считать отключенным от последующей схемы. На первом и втором уровнях представления такой выход можно считать состоящим из двух переключателей (рис. 1.9), которые могут замыкаться по очереди, давая логический нуль и логическую единицу, но могут и размыкаться одновременно. Это третье состояние называется также высокоимпедансным или Z-состоянием. Для перевода выхода в третье Z-состояние используется специальный управляющий вход, обозначаемый OE (Output Enable — разрешение выхода) или EZ (Enable Z-state — разрешение Z-состояния, или третьего состояния).

Почему же помимо стандартного выхода (2С) были предложены еще два типа выходов (ОК и 3С)? Дело в том, что выходы, имеющие помимо активных состояний еще и пассивное состояние, очень удобны для объединения их между собой. Например, если на один и тот же вход надо по очереди подавать сигналы с двух выходов (рис. 1.10), то выходы 2С для этого не подходят, а вот выходы ОК и 3С подходят.

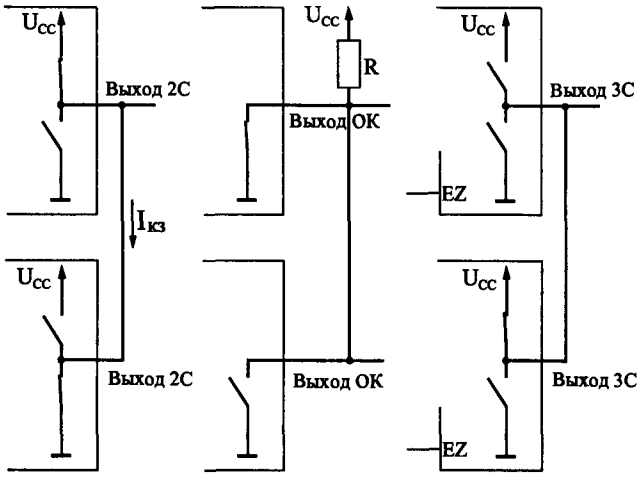


Рис. 1.10. Объединение выходов цифровых микросхем.

При объединении двух или более выходов 2С вполне возможна ситуация, при которой один выход стремится выдать сигнал логической единицы, а другой — сигнал логического нуля. Легко заметить, что в этом случае через верхний замкнутый ключ выхода, выдающего единицу, и через нижний замкнутый ключ выхода, выдающего нуль, пойдет недопустимо большой ток короткого замыкания $I_{кз}$. Это аварийная ситуация, при которой уровень получаемого выходного логического сигнала точно не определен, он может восприниматься последующим входом и как нуль, и как единица. Конфликтующие выходы могут даже выйти из строя, нарушив работу микросхем и схемы в целом.

Зато в случае объединения двух выходов ОК такого конфликта в принципе произойти не может. Даже если ключ одного выхода замкнут, а другого разомкнут, аварийной ситуации не произойдет, так как недопустимо большого тока не будет, а на объединенном выходе будет сигнал логического нуля. А при объединении двух выходов 3С аварийная ситуация хотя и возможна (если оба выхода одновременно находятся в активном состоянии), но ее легко можно предотвратить, если организовать схему так, что в активном состоянии всегда будет находиться только один из объединенных выходов 3С.

Объединение выходов цифровых микросхем совершенно необходимо также при шинной (или, как еще говорят, магистральной) организации связей между цифровыми устройствами. Шинная организация связей применяется, например, в компьютерах, других микропроцессорных системах. Суть ее сводится к следующему.

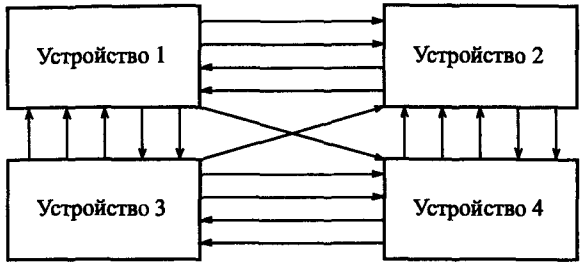


Рис. 1.11. Классическая организация связей.

При классической организации связей (рис. 1.11) все сигналы между устройствами передаются по своим отдельным линиям (проводам). Каждое устройство передает свои сигналы всем другим устройствам независимо от других устройств. В этом случае обычно получается очень много линий связи, к тому же правила обмена сигналами по этим линия (или протоколы обмена) чрезвычайно разнообразны.

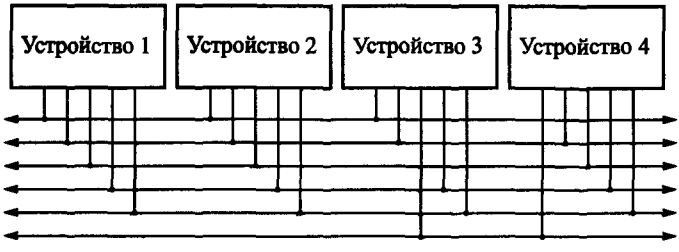


Рис. 1.12. Шинная организация связей.

При шинной организации связей (рис. 1.12) все сигналы между устройствами передаются по одним и тем же линиям (проводам), но в разные моменты времени (это называется временным мультиплексированием). В результате количество линий

связи резко сокращается, а правила обмена сигналами существенно упрощаются. Группа линий (сигналов), используемая несколькими устройствами, как раз и называется шиной. Понятно, что объединение выходов в этом случае совершенно необходимо, ведь каждое устройство должно иметь возможность выдавать свой сигнал на общую линию. К недостаткам шинной организации относится прежде всего невысокая (по сравнению с классической структурой связей) скорость обмена сигналами. При простых структурах связи шинная организация может быть избыточна.

Но вернемся к типам выходов цифровых микросхем.

На третьем уровне представления (электрическая модель) необходимо уже учитывать, что выходные ключи (рис. 1.9) представляют собой не простые тумблеры (как на первых двух уровнях представления), а транзисторные ключи со своими специфическими параметрами. Однако в большинстве случаев достаточно знать, какой ток может выдать данный выход в состояниях логического нуля (I_{OL}) и логической единицы (I_{OH}). Величины этих токов не должны превышать суммы токов всех входов, подключенных к данному выходу (соответственно I_{IL} и I_{IH}). Количество входов, которое можно подключить к одному выходу, определяет коэффициент разветвления или нагрузочную способность микросхемы. Существуют микросхемы с обычной нагрузочной способностью и с повышенной нагрузочной способностью (больше обычной в два раза и более). Выходы 3С, как правило, имеют повышенную нагрузочную способность (то есть обеспечивают большие выходные токи). Выходы 2С и ОК могут быть как с обычной, так и с повышенной нагрузочной способностью.

На третьем уровне представления (электрическая модель) необходимо также учитывать выдаваемые выходом микросхемы величины выходных напряжений U_{OL} и U_{OH}. Выходы ОК могут быть рассчитаны как на обычное выходное напряжение логической единицы ($U_{OH} = U_{CC} = 5$ В), так и на повышенное напряжение логической единицы (до 30 В). В последнем случае внешний резистор этого выхода (см. рис. 1.9) подключается к источнику повышенного напряжения.

Только в сложных случаях, например при переводе логического элемента в линейный режим за счет обратных связей, нужен учет других параметров входных и выходных каскадов. Но

в этих редких случаях гораздо проще и надежнее не считать ничего самому, а воспользоваться стандартными схемами включения микросхем или подобрать режимы работы и номиналы внешних элементов (резисторов, конденсаторов) непосредственно на макете проектируемого устройства. В отличие от расчетов такой подход даст полную гарантию работоспособности выбранного решения.

1.4. Основные обозначения на схемах

Для изображения электронных устройств и их узлов применяется три основных типа схем:

- принципиальная схема;
- структурная схема;
- функциональная схема.

Различаются эти три вида схем своим назначением и, самое главное, степенью детализации изображения устройств.

Принципиальная схема — это наиболее подробная схема. Она обязательно показывает все использованные в устройстве элементы и все связи между ними. Если схема строится на основе микросхем, то должны быть показаны номера выводов всех входов и выходов этих микросхем. Принципиальная схема должна позволять полностью воспроизвести устройство. Обозначения принципиальной схемы наиболее жестко стандартизованы, отклонения от стандартов не рекомендуются.

Структурная схема — это наименее подробная схема. Она предназначена для отображения общей структуры устройства, то есть его основных блоков, узлов, частей и главных связей между ними. Из структурной схемы должно быть понятно, зачем нужно данное устройство, и что оно делает в основных режимах работы, как взаимодействуют его части. Обозначения структурной схемы могут быть довольно произвольными, хотя некоторые общепринятые правила все-таки лучше выполнять.

Функциональная схема представляет собой гибрид структурной и принципиальной схем. Некоторые наиболее простые блоки, узлы, части устройства отображаются на ней, как на структурной схеме, а остальные — как на принципиальной схеме. Функциональная схема позволяет понять всю логику работы устройства, все его отличия от других подобных устройств, но

не позволяет без дополнительной самостоятельной работы воспроизвести это устройство. Что касается обозначений, используемых на функциональных схемах, то в части, показанной как структура, они не стандартизованы, а в части, показанной, как принципиальная схема, они стандартизованы.

В технической документации обязательно приводится структурная или функциональная схема, а также обязательно принципиальная схема. В научных статьях и книгах чаще всего ограничиваются структурной или функциональной схемой, приводя принципиальные схемы только некоторых узлов.

А теперь рассмотрим основные обозначения, используемые на схемах.

Все узлы, блоки, части, элементы, микросхемы показываются в виде прямоугольников с соответствующими надписями. Все связи между ними, все передаваемые сигналы показываются в виде линий, соединяющих эти прямоугольники. Входы и входы/выходы должны быть расположены на левой стороне прямоугольника, выходы — на правой стороне, хотя это правило часто нарушают, когда необходимо упростить рисунок схемы. Выводы и связи питания, как правило, не показывают, если, конечно, не используются нестандартные включения элементов схемы. Это самые общие правила, касающиеся любых схем.

Прежде чем перейти к более частным правилам, надо дать несколько определений.

Положительный сигнал (сигнал положительной полярности) — это сигнал, активный уровень которого — логическая единица, то есть: нуль — это отсутствие сигнала, единица — сигнал пришел (рис. 1.13).

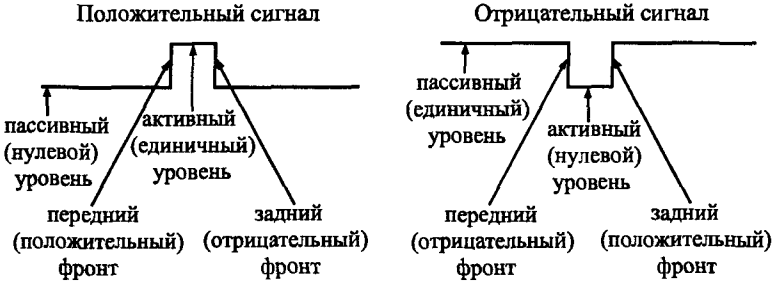


Рис. 1.13. Элементы цифрового сигнала.

Отрицательный сигнал (сигнал отрицательной полярности) — это сигнал, активный уровень которого — логический нуль, то есть: единица — это отсутствие сигнала, нуль — сигнал пришел (рис. 1.13).

Активный уровень сигнала — это уровень, соответствующий приходу сигнала, то есть выполнению этим сигналом соответствующей ему функции.

Пассивный уровень сигнала — это уровень, в котором сигнал не выполняет никакой функции.

Инвертирование или инверсия сигнала — это изменение его полярности.

Инверсный выход — это выход, выдающий сигнал инверсной полярности по сравнению с входным сигналом.

Прямой выход — это выход, выдающий сигнал такой же полярности, какую имеет входной сигнал.

Положительный фронт сигнала — это переход сигнала из нуля в единицу.

Отрицательный фронт сигнала (спад) — это переход сигнала из единицы в нуль.

Передний фронт сигнала — это переход сигнала из пассивного уровня в активный уровень.

Задний фронт сигнала — это переход сигнала из активного уровня в пассивный уровень.

Тактовый сигнал (или строб) — управляющий сигнал, который определяет момент выполнения элементом или узлом его функции.

Шина — группа сигналов (и соответствующих физических линий передачи этих сигналов), объединенных по какому-то принципу. Например, шиной называют сигналы, соответствующие всем разрядам какого-то двоичного кода.

Для обозначения полярности сигнала на схемах используется простое правило: если сигнал отрицательный, то перед его названием ставится знак минус, например, –WR или –OE, или же (реже) над названием сигнала ставится черта. Если таких знаков нет, то сигнал считается положительным. Для названий сигналов обычно используются латинские буквы, представляющие собой сокращения английских слов. Например, WR — сигнал записи (от Write — писать).

Инверсия сигнала обозначается кружочком на месте входа или выхода. Существуют инверсные входы и инверсные выходы (рис. 1.14).

Если какая-то микросхема выполняет функцию по фронту входного сигнала, то на месте входа ставится косая черта (под углом 45°), причем наклон вправо или влево определяется тем, какой фронт — положительный или отрицательный — используется в данном случае (рис. 1.14).

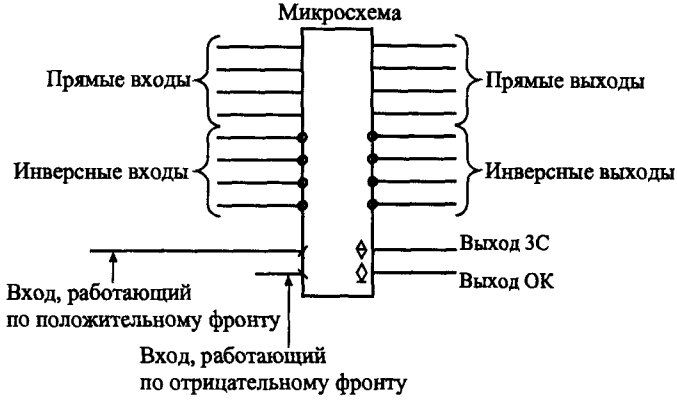


Рис. 1.14. Обозначение входов и выходов.

Тип выхода микросхемы помечается специальным значком: выход 3С — перечеркнутым ромбом, а выход ОК — подчеркнутым ромбом (рис. 1.14). Стандартный выход (2С) никак не помечается.

Наконец, если у микросхемы необходимо показать неинформационные выводы, то есть выводы, не являющиеся ни логическими входами, ни логическими выходами, то такой вывод помечается косым крестом (две перпендикулярные линии под углом 45°). Это могут быть, например, выводы для подключения внешних элементов (резисторов, конденсаторов) или выводы питания (рис. 1.15).

В схемах также предусматриваются специальные обозначения для шин (рис. 1.16). На структурных и функциональных схемах шины обозначаются толстыми линиями или двойными стрелками, причем количество сигналов, входящих в шину, указывается рядом с косой чертой, пересекающей шину. На прин-

ципиальных схемах шина тоже обозначается толстой линией, а входящие в шину и выходящие из шины сигналы показываются в виде перпендикулярных к шине тонких линий с указанием их номера или названия (рис. 1.16). При передаче по шине двоичного кода нумерация начинается с младшего разряда кода.

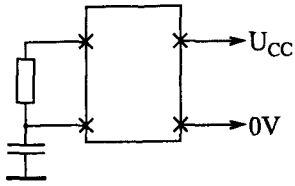


Рис. 1.15. Обозначение неинформационных выводов.

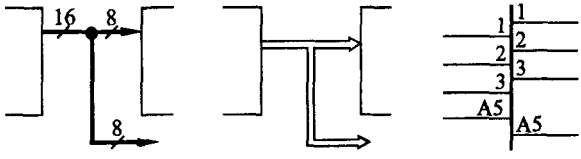


Рис. 1.16. Обозначение шин.

При изображении микросхем используются сокращенные названия входных и выходных сигналов, отражающие их функцию. Эти названия располагаются на рисунке рядом с соответствующим выводом. Также на изображении микросхем указывается выполняемая ими функция (обычно в центре вверху). Изображение микросхемы иногда делят на три вертикальных поля. Левое поле относится к входным сигналам, правое — к выходным сигналам. В центральном поле помещаются название микросхемы и символы ее особенностей. Неинформационные выводы могут указываться как на левом, так и на правом поле, иногда их показывают на верхней или нижней стороне прямоугольника, изображающего микросхему.

В табл. 1.2 приведены некоторые наиболее часто встречающиеся обозначения сигналов и функций микросхем. Микросхема в целом обозначается на схемах буквами DD (от английского Digital — цифровой) с соответствующим номером, например DD1, DD20.1, DD38.2 (после точки указывается номер элемента или узла внутри микросхемы).

Таблица 1.2. Некоторые обозначения сигналов и микросхем

Обозначение	Название	Назначение
&	And	Элемент И
=1	Exclusive Or	Элемент Исключающее ИЛИ
1	Or	Элемент ИЛИ
A	Address	Адресные разряды
BF	Buffer	Буфер
C	Clock	Тактовый сигнал (строб)
CE	Clock Enable	Разрешение тактового сигнала
CT	Counter	Счетчик
CS	Chip Select	Выбор микросхемы
D	Data	Разряды данных, данные
DC	Decoder	Дешифратор
EZ	Enable Z-state	Разрешение третьего состояния
G	Generator	Генератор
I	Input	Вход
I/O	Input/Output	Вход/Выход
OE	Output Enable	Разрешение выхода
MS	Multiplexer	Мультиплексор
Q	Quit	Выход
R	Reset	Сброс (установка в нуль)
RG	Register	Регистр
S	Set	Установка в единицу
SUM	Summator	Сумматор
T	Trigger	Триггер
TC	Terminal Count	Окончание счета
Z	Z-state	Третье состояние выхода

Более полная таблица обозначений сигналов и микросхем, используемых в принципиальных схемах, приведена в Приложении.

1.5. Серии цифровых микросхем

В настоящее время выпускается огромное количество разнообразных цифровых микросхем от простейших логических элементов до сложнейших процессоров, микроконтроллеров и специализированных БИС (больших интегральных микросхем). Выпуском цифровых микросхем занимается множество фирм как у нас

в стране, так и за рубежом. Поэтому даже классификация этих микросхем представляет собой довольно трудную задачу.

Однако в качестве базиса в цифровой схемотехнике принято рассматривать классический набор микросхем малой и средней степени интеграции, в основе которого лежат ТТЛ серии семейства 74, выпускаемые уже несколько десятилетий рядом фирм, например американской фирмой Texas Instruments (TI). Эти серии включают в себя функционально полный комплект микросхем, используя который можно создавать самые разные цифровые устройства. Даже при компьютерном проектировании современных сложных микросхем с программируемой логикой (ПЛИС) применяются модели простейших микросхем этих серий семейства 74. При этом разработчик рисует на экране компьютера схему в привычном для него элементном базисе, а затем программа создает прошивку ПЛИС, выполняющую требуемую функцию.

Каждая микросхема серий семейства 74 имеет свое обозначение, и система обозначений отечественных серий существенно отличается от принятой за рубежом.

Рис. 1.17. Система обозначений фирмы Texas Instruments.

В качестве примера рассмотрим систему обозначений фирмы Texas Instruments (рис. 1.17). Полное обозначение состоит из шести элементов:

1. Идентификатор фирмы SN (для серий AC и ACT отсутствует).
2. Температурный диапазон (тип семейства):
   - 74 — коммерческие микросхемы (температура окружающей среды для биполярных микросхем — 0...70°C, для КМОП микросхем — −40...+85°C),
   - 54 — микросхемы военного назначения (температура окружающей среды — −55...+125°C).

3. Код серии (до трех символов):
   - Отсутствует — стандартная ТТЛ серия.
   - LS (Low Power Schottky) — маломощная серия ТТЛШ.
   - S (Schottky) — серия ТТЛШ.
   - ALS (Advanced Schottky) — улучшенная серия ТТЛШ.
   - F (FAST) — быстрая серия.
   - HC (High Speed CMOS) — высокоскоростная КМОП серия.
   - HCT (High Speed CMOS with TTL inputs) — серия HC, совместимая по входу с ТТЛ.
   - AC (Advanced CMOS) — улучшенная серия КМОП.
   - ACT (Advanced CMOS with TTL inputs) — серия AC, совместимая по входу с ТТЛ.
   - BCT (BiCMOS Technology) — серия с БиКМОП технологией.
   - ABT (Advanced BiCMOS Technology) — улучшенная серия с БиКМОП технологией.
   - LVT (Low Voltage Technology) — серия с низким напряжением питания.
4. Идентификатор специального типа (2 символа) — может отсутствовать.
5. Тип микросхемы (от двух до шести цифр). Перечень некоторых типов микросхем приведен в Приложении.
6. Код типа корпуса (от одного до двух символов) — может отсутствовать. Например, N — пластмассовый корпус DIL (DIP), J — керамический корпус DIL (DIC), T — плоский металлический корпус.

Примеры обозначений: SN74ALS373, SN74ACT7801, SN7400.

Отечественная система обозначений микросхем отличается от рассмотренной довольно существенно (рис. 1.18). Основные элементы обозначения следующие:

1. Буква К обозначает микросхемы широкого применения, для микросхем военного назначения буква отсутствует.
2. Тип корпуса микросхемы (один символ) — может отсутствовать. Например, Р — пластмассовый корпус, М — керамический корпус, Б — бескорпусная микросхема.
3. Номер серии микросхем (от трех до четырех цифр).
4. Функция микросхемы (две буквы).

5. Номер микросхемы (от одной до трех цифр). Таблица функций и номеров микросхем, а также таблица их соответствия зарубежным аналогам приведены в Приложении.

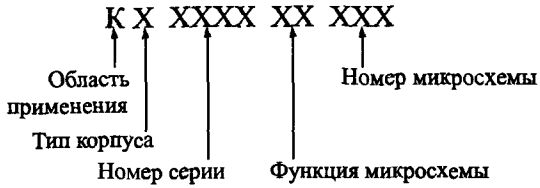


Рис. 1.18. Обозначения отечественных микросхем.

Примеры обозначений: КР1533ЛА3, КМ531ИЕ17, КР1554ИР47.

Главное достоинство отечественной системы обозначений состоит в том, что по обозначению микросхемы можно легко понять ее функцию. Зато в системе обозначений Texas Instruments виден тип серии с его особенностями.

Чем отличается одна серия от другой?

На первом уровне представления (логическая модель) серии не различаются ничем. То есть одинаковые микросхемы разных серий работают по одним и тем же таблицам истинности, по одним и тем же алгоритмам. Правда, надо учитывать, что некоторые микросхемы имеются только в одной из серий, а некоторых нет в нескольких сериях.

На втором уровне представления (модель с учетом задержек) серии отличаются величиной задержки распространения сигнала. Это различие может быть довольно существенным. Поэтому в тех схемах, где величина задержки принципиальна, надо использовать микросхемы более быстрых серий (табл. 1.3).

На третьем уровне представления (электрическая модель) серии различаются величинами входных и выходных токов и напряжений, а также, что не менее важно, токами потребления (табл. 1.3). Поэтому в тех устройствах, где эти параметры принципиальны, надо применять микросхемы, обеспечивающие, например, низкие входные токи, высокие выходные токи и малое потребление.

Серия К155 (SN74) — это наиболее старая серия, которая постепенно снимется с производства. Она отличается не слишком хорошими параметрами по сравнению с другими сериями. С этой классической серией принято сравнивать все остальные.

Таблица 1.3. Сравнение параметров одинаковых микросхем в разных стандартных сериях

	К155ЛА3 (SN7400N)	К555ЛА3 (SN74LS00N)	КР1533ЛА3 (SN74ALS00N)	КР1554ЛА3 (SN74AC00N)
T_{PLH}, нс не более	22	15	11	8,5
T_{PHL}, нс не более	15	15	8	7,0
I_{IL}, мА не более	−1,6	−0,4	−0,1	−0,001
I_{IH}, мА не более	0,04	0,02	0,02	0,001
I_{OL}, мА не менее	16	8	15	86
I_{OH}, мА не менее	−0,4	−0,4	−0,4	−75
U_{OL}, В не более	0,4	0,5	0,5	0,3
U_{OH}, В не менее	2,4	2,7	2,5	4,4
I_{CC}, мА не более	12	4,4	3	0,04

Серия К555 (SN74LS) отличается от серии К155 малыми входными токами и меньшей потребляемой мощностью (ток потребления почти втрое меньше, чем у К155). По быстродействию (по временам задержек) она близка к серии К155.

Серия КР531 (SN74S) отличается высоким быстродействием (задержки примерно в 3—4 раза меньше, чем у серии К155), но большими входными токами (на 25% больше, чем у К155) и большой потребляемой мощностью (ток потребления больше в полтора раза по сравнению с серией К155).

Серия КР1533 (SN74ALS) отличается повышенным примерно вдвое по сравнению с К155 быстродействием и малой потребляемой мощностью (в четыре раза меньше, чем у К155). Входные токи еще меньше, чем у серии К555.

Серия КР1531 (SN74F) отличается высоким быстродействием (на уровне КР531), но малой потребляемой мощностью. Входные токи и ток потребления примерно вдвое меньше, чем у серии К155.

Серия КР1554 (SN74AC) отличается от всех предыдущих тем, что она выполнена по КМОП-технологии. Поэтому она характеризуется сверхмалыми входными токами и сверхмалым потреблением при малых рабочих частотах. Задержки примерно вдвое меньше, чем у серии К155.

Наибольшим разнообразием имеющихся микросхем отличаются серии К155 и КР1533, наименьшим — серии КР1531 и КР1554.

Следует отметить, что приведенные здесь соотношения по быстродействию стандартных серий довольно приблизительны и выполняются не для всех разновидностей микросхем, имеющихся в разных сериях. Точные значения задержек необходимо находить в справочниках, причем желательно использовать фирменные справочные материалы.

Микросхемы разных серий обычно легко сопрягаются между собой, то есть сигналы с выходов микросхем одной серии можно смело подавать на входы микросхем другой серии. Одно из исключений — соединение выходов ТТЛ микросхем со входами КМОП микросхем серии КР1554 (74AC). При таком соединении необходимо применение резистора номиналом 560 Ом между линиями сигнала и напряжения питания (рис. 1.19).

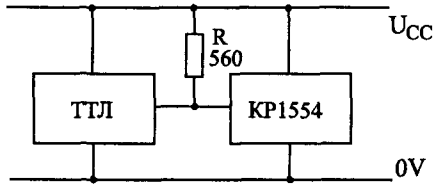


Рис. 1.19. Сопряжение микросхем ТТЛ и КР1554 (КМОП).

При выборе той или иной серии микросхем следует также учитывать, что микросхемы мощной и быстрой серии КР531 создают высокий уровень помех по шинам питания, а микросхемы маломощной серии К555 очень чувствительны к таким помехам. Поэтому серию КР531 рекомендуется использовать только в крайних случаях, когда необходимо получить очень высокое быстродействие. Не рекомендуется также применять в одном и том же устройстве мощные быстродействующие и маломощные микросхемы.

1.6. Корпуса цифровых микросхем

Большинство микросхем имеют корпус, то есть прямоугольный контейнер (пластмассовый, керамический, металлокерамический) с металлическими выводами (ножками). Предложено множество различных типов корпусов, но наибольшее распространение получили два основных типа:

- Корпус с двухрядным вертикальным расположением выводов, например: DIP (Dual In Line Package, Plastic) — пластмассовый корпус, DIC (Dual In Line Package, Ceramic) — керамический корпус. Общее название для таких корпусов — DIL (рис. 1.20). Расстояние между выводами составляет 0,1 дюйма (2,54 мм). Расстояние между рядами выводов зависит от количества выводов.
- Корпус с двухрядным плоскостным расположением выводов, например: FP (Flat-Package, Plastic) — пластмассовый плоский корпус, FPC (Flat-Package, Ceramic) — керамический плоский корпус. Общее название для таких корпусов — Flat (рис. 1.20). Расстояние между выводами составляет 0,05 дюйма (1,27 мм) или 0,025 дюйма (0,0628 мм).

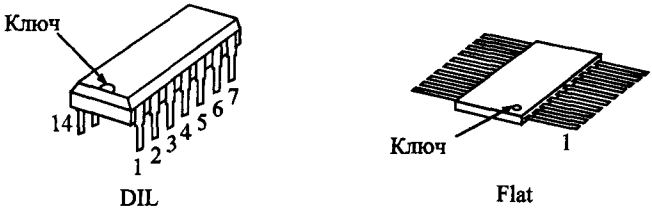


Рис. 1.20. Примеры корпусов DIL и Flat.

Номера выводов всех корпусов считаются начиная с вывода, помеченного ключом, по направлению против часовой стрелки (если смотреть на микросхему сверху). Ключом может служить вырез на одной из сторон корпуса микросхемы, точка около первого вывода или утолщение первого вывода (рис. 1.20). Первый вывод может находиться в левом нижнем углу или в правом верхнем углу (в зависимости от того, как повернут корпус). Микросхемы обычно имеют стандартное число выводов из ряда: 4, 8, 14, 16, 20, 24, 28,... Для микросхем стандартных цифровых серий используются корпуса с количеством выводов начиная с 14.

Назначение каждого из выводов микросхемы приводится в справочниках по микросхемам, которых сейчас имеется множество. Правда, лучше ориентироваться на справочники, издаваемые непосредственно фирмами-изготовителями. В данной книге назначение выводов микросхем не приводится.

Отечественные микросхемы выпускаются в корпусах, очень похожих на DIL и Flat, но расстояния между их выводами вычисляются по метрической шкале и поэтому чуть-чуть отличаются от принятых за рубежом. Например, 2,5 мм вместо 2,54 мм, 1,25 мм вместо 1,27 мм и т. д. Для корпусов с малым числом выводов (до 20) это не слишком существенно, но для больших корпусов расхождение в расстоянии может стать существенным. В результате на плату, рассчитанную на зарубежные микросхемы, нельзя поставить отечественные микросхемы и наоборот.

1.7. Двоичное кодирование

Одиночный цифровой сигнал не слишком информативен, ведь он может принимать только два значения: нуль и единица. Поэтому в тех случаях, когда необходимо передавать, обрабатывать или хранить большие объемы информации, обычно применяют несколько параллельных цифровых сигналов. При этом все эти сигналы должны рассматриваться только одновременно, каждый из них по отдельности не имеет смысла. В таких случаях говорят о двоичных кодах, то есть о кодах, образованных цифровыми (логическими, двоичными) сигналами. Каждый из логических сигналов, входящих в код, называется разрядом. Чем больше разрядов входит в код, тем больше значений может принимать данный код.

Рис. 1.21. Десятичное и двоичное кодирование.

В отличие от привычного для нас десятичного кодирования чисел, то есть кода с основанием десять, при двоичном кодировании в основании кода лежит число два (рис. 1.21). То есть каждая цифра кода (каждый разряд) двоичного кода может принимать не десять значений (как в десятичном коде: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), а всего лишь два значения — 0 и 1. А система позиционной записи остается такой же, то есть справа пишется самый младший разряд, а слева— самый старший. Но если в десятичной системе вес каждого следующего разряда в десять раз больше веса предыдущего разряда, то в двоичной системе (при двоичном кодировании) — в два раза. Каждый разряд двоичного кода называется *битом* (от английского Binary Digit — двоичное число).

Таблица 1.4. Соответствие чисел в десятичной и двоичной системах

Десятичная система	Двоичная система	Десятичная система	Двоичная система
0	0	10	1010
1	1	11	1011
2	10	12	1100
3	11	13	1101
4	100	14	1110
5	101	15	1111
6	110	16	10000
7	111	17	10001
8	1000	18	10010
9	1001	19	10011

В табл. 1.4 показано соответствие первых двадцати чисел в десятичной и двоичной системах.

Из таблицы видно, что количество разрядов двоичного кода, требуемое для представления каждого числа (кроме 0 и 1), значительно больше, чем требуемое количество разрядов десятичного кода. Наибольшее число, которое можно представить 3-разрядным кодом, в десятичной системе составляет 999, а в двоичной системе — всего лишь 7 (111 в двоичном коде). В общем случае n-разрядное двоичное число может принимать 2^n различных значений, а n-разрядное десятичное число — 10^n значений. Поэтому запись больших двоичных чисел (с количеством разрядов больше десяти) становится не слишком удобной.

Для того чтобы упростить запись двоичных чисел, была предложена так называемая шестнадцатеричная система (16-ричное кодирование). В этом случае все двоичные разряды разбиваются на группы по четыре разряда (начиная с младшего), а затем уже каждая группа кодируется одним символом. Каждая такая группа называется *полубайтом* (или *нибблом*, *тетрадой*), а две группы (8 разрядов) — *байтом*. Из табл. 1.4 видно, что 4-разрядное двоичное число может принимать 16 разных значений (от 0 до 15). Поэтому требуемое число символов для шестнадцатеричного кода тоже равно 16, откуда и происходит название кода. В качестве первых 10 символов берутся цифры от 0 до 9, а затем используются 6 начальных заглавных букв латинского алфавита: A, B, C, D, E, F.

Таблица 1.5. 16-ричная система кодирования

Десятичная система	16-ричная система	Десятичная система	16-ричная система
0	0 (0)	10	A (1010)
1	1 (1)	11	B (1011)
2	2 (10)	12	C (1100)
3	3 (11)	13	D (1101)
4	4 (100)	14	E (1110)
5	5 (101)	15	F (1111)
6	6 (110)	16	10 (1 0000)
7	7 (111)	17	11 (1 0001)
8	8 (1000)	18	12 (1 0010)
9	9 (1001)	19	13 (1 0011)

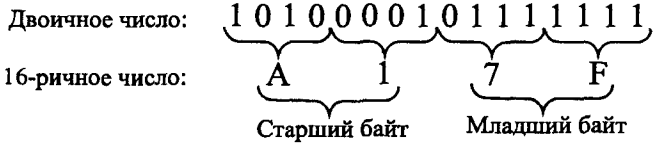


Рис. 1.22. Двоичная и шестнадцатеричная запись числа.

В табл. 1.5 приведены примеры 16-ричного кодирования первых 20 чисел (в скобках приведены двоичные числа), а на рис. 1.22 показан пример записи двоичного числа в 16-ричном виде. Для обозначения 16-ричного кодирования иногда приме-

няют букву «h» или «H» (от английского Hexadecimal) в конце числа. Например, запись A17F h обозначает 16-ричное число A17F. Здесь A1 представляет собой старший байт числа, а 7F — младший байт числа. Все число (в нашем случае — двухбайтовое) называется *словом*.

Для перевода 16-ричного числа в десятичное необходимо умножить значение младшего (нулевого) разряда на единицу, значение следующего (первого) разряда на 16, второго разряда на 256 (16^2) и т. д., а затем сложить все произведения. Например, возьмем число A17F:

$$A17F = F \cdot 16^0 + 7 \cdot 16^1 + 1 \cdot 16^2 + A \cdot 16^3 =$$
$$= 15 \cdot 1 + 7 \cdot 16 + 1 \cdot 256 + 10 \cdot 4096 = 41343.$$

Но каждому специалисту по цифровой аппаратуре (разработчику, оператору, ремонтнику, программисту и т. д.) необходимо научиться так же свободно обращаться с 16-ричной и двоичной системами, как и с обычной десятичной, чтобы никаких переводов из системы в систему не требовалось.

Таблица 1.6. 8-ричная система кодирования

Десятичная система	8-ричная система	Десятичная система	8-ричная система
0	0 (0)	10	12 (1 010)
1	1 (1)	11	13 (1 011)
2	2 (10)	12	14 (1 100)
3	3 (11)	13	15 (1 101)
4	4 (100)	14	16 (1 110)
5	5 (101)	15	17 (1 111)
6	6 (110)	16	20 (10 000)
7	7 (111)	17	21 (10 001)
8	10 (1 000)	18	22 (10 010)
9	11 (1 001)	19	23 (10 011)

Значительно реже, чем 16-ричное, используется восьмеричное кодирование, которое строится по такому же принципу, что и 16-ричное кодирование, но двоичные разряды разбиваются на группы по три разряда. Каждая группа (разряд кода) затем обозначается одним символом. Каждый разряд 8-ричного кода может принимать восемь значений: 0, 1, 2, 3, 4, 5, 6, 7 (табл. 1.6).

Помимо рассмотренных кодов существует также и так называемое двоично-десятичное представление чисел. Как и в 16-ричном коде, в двоично-десятичном коде каждому разряду кода соответствует четыре двоичных разряда, однако каждая группа из четырех двоичных разрядов может принимать не шестнадцать, а только десять значений, кодируемых символами 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. То есть одному десятичному разряду соответствует четыре двоичных разряда. В результате получается, что написание чисел в двоично-десятичном коде ничем не отличается от написания в обычном десятичном коде (табл. 1.7), но в реальности это всего лишь специальный двоичный код, каждый разряд которого может принимать только два значения: 0 и 1. Двоично-десятичный код иногда очень удобен для организации десятичных цифровых индикаторов и табло.

Таблица 1.7. Двоично-десятичная система кодирования

Десятичная система	Двоично-десятичная система	Десятичная система	Двоично-десятичная система
0	0 (0)	10	10 (1 0000)
1	1 (1)	11	11 (1 0001)
2	2 (10)	12	12 (1 0010)
3	3 (11)	13	13 (1 0011)
4	4 (100)	14	14 (1 0100)
5	5 (101)	15	15 (1 0101)
6	6 (110)	16	16 (1 0110)
7	7 (111)	17	17 (1 0111)
8	8 (1000)	18	18 (1 1000)
9	9 (1001)	19	19 (1 1001)

В двоичном коде над числами можно проделывать любые арифметические операции: сложение, вычитание, умножение, деление.

Рассмотрим, например, сложение двух 4-разрядных двоичных чисел. Пусть надо сложить число 0111 (десятичное 7) и 1011 (десятичное 11). Сложение этих чисел не сложнее, чем в десятичном представлении:

$$\begin{array}{r} 0111 \\ +\ 1011 \\ \hline 10010 \end{array}$$

При сложении 0 и 0 получаем 0, при сложении 1 и 0 получаем 1, при сложении 1 и 1 получаем 0 и перенос в следующий разряд 1. Результат — 10010 (десятичное 18). При сложении любых двух n-разрядных двоичных чисел может получиться n-разрядное или (n+1)-разрядное число.

Точно так же производится вычитание. Пусть из числа 10010 (18) надо вычесть число 0111 (7). Записываем числа с выравниванием по младшему разряду и вычитаем точно так же, как в случае десятичной системы:

```
  10010
- 0111
-------
  1011
```

При вычитании 0 из 0 получаем 0, при вычитании 0 из 1 получаем 1, при вычитании 1 из 1 получаем 0, при вычитании 1 из 0 получаем 1 и заем 1 в следующем разряде. Результат — 1011 (десятичное 11).

При вычитании возможно получение отрицательных чисел, поэтому необходимо использовать двоичное представление отрицательных чисел.

Для одновременного представления как двоичных положительных, так и двоичных отрицательных чисел чаще всего используется так называемый дополнительный код. Отрицательные числа в этом коде выражаются таким числом, которое, будучи сложено с положительным числом такой же величины, даст в результате нуль. Для того чтобы получить отрицательное число, нужно дополнить до 1 каждый разряд такого же положительного числа, то есть заменить в его двоичном коде все нули на единицы и единицы на нули, и затем прибавить к результату 1. Например, запишем двоичное представление числа –5 с использованием дополнительного кода. Двоичный код числа 5 есть 0101. Заменяем во всех разрядах 1 на 0 и 0 на 1: 1010. Прибавляем единицу: 1011. Суммируем результат с исходным числом: 1011 + 0101 = 0000 (перенос в пятый разряд игнорируем).

Отрицательные числа в дополнительном коде отличаются от положительных значением старшего разряда: единица в старшем разряде определяет отрицательное число, а нуль — положительное.

Помимо стандартных арифметических операций в двоичной системе счисления используются и некоторые специфические

операции, например сложение по модулю 2. Это операция (обозначается $\oplus$) является побитовой, то есть никаких переносов из разряда в разряд и заемов в старших разрядах здесь не существует. Правила сложения по модулю 2 следующие: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. Эта же операция называется функцией Исключающее ИЛИ. Например, просуммируем по модулю 2 два двоичных числа 0111 и 1011:

$$\begin{array}{r} 0111 \\ \oplus\ 1011 \\ \hline 1100 \end{array}$$

Среди других побитовых операций над двоичными числами можно отметить функцию И и функцию ИЛИ. Функция И дает в результате единицу только тогда, когда соответствующие биты двух исходных чисел оба имеют единичное значение, в противном случае результат — 0. Функция ИЛИ дает в результате единицу тогда, когда значение хотя бы одного из соответствующих битов исходных чисел равно 1, в противном случае результат — 0.

1.8. Функции цифровых устройств

Любое цифровое устройство от самого простейшего до самого сложного всегда действует по одному и тому же принципу (рис. 1.23). Оно принимает входные сигналы, выполняет их обработку, передачу, хранение и выдает выходные сигналы. При этом совсем не обязательно любое изменение входных сигналов приводит к немедленному и однозначному изменению выходных сигналов. Реакция устройства может быть очень сложной, отложенной по времени, неочевидной, но суть от этого не меняется.

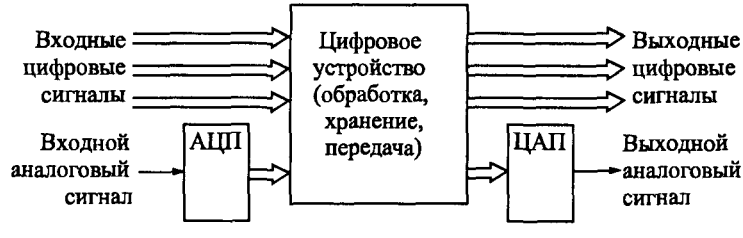


Рис. 1.23. Включение цифрового устройства.

В качестве входных сигналов нашего устройства могут выступать сигналы с выходов других цифровых устройств, с тумблеров и клавиш или с датчиков физических величин. Причем в последнем случае, как правило, необходимо осуществлять преобразование выходных аналоговых сигналов датчиков в потоки цифровых кодов (рис. 1.24) с помощью аналого-цифровых преобразователей (АЦП). Например, в случае персонального компьютера входными сигналами являются сигналы с клавиатуры, с датчиков перемещения мыши, с микрофона (давление воздуха, то есть звук, преобразуется в аналоговый электрический сигнал, а затем — в цифровые коды), из кабеля локальной сети и т. д.

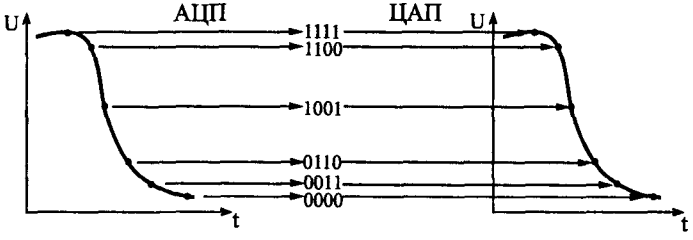


Рис. 1.24. Аналого-цифровое и цифро-аналоговое преобразование.

Выходные сигналы цифрового устройства могут предназначаться для подачи на другие цифровые устройства, для индикации (на экране монитора, на цифровом индикаторе и т. д.), а также для формирования физических величин. Причем в последнем случае необходимо преобразовывать потоки кодов с цифрового устройства в непрерывные (аналоговые) сигналы (рис. 1.24) с помощью цифро-аналоговых преобразователей (ЦАП) и в физические величины. Например, в случае персонального компьютера выходными сигналами будут сигналы, подаваемые компьютером на принтер, сигналы, идущие на видеомонитор (аналоговые или цифровые), звук, воспроизводимый динамиками компьютера (потоки кодов с компьютера преобразуются в аналоговый электрический сигнал, который затем преобразуется в давление воздуха — звук).

Одно цифровое устройство может состоять из нескольких более простых цифровых устройств. Часто эти составные элементы называют блоками, модулями, узлами, частями. Если

объединяется несколько сложных цифровых устройств, то говорят уже о цифровых системах, комплексах, установках. Мы в основном будем использовать термин «устройство» как занимающий промежуточное положение.

Связь между входными и выходными сигналами может быть жесткой, неизменной или гибко изменяемой (то есть программируемой). То есть цифровое устройство может работать по жесткому, раз и навсегда установленному алгоритму или по программируемому алгоритму. Как правило, при этом выполняется один очень простой принцип: чем больше возможностей для изменения связи входных и выходных сигналов, чем больше возможностей изменения алгоритма работы, тем медленнее будет цифровое устройство. Речь в данном случае, конечно же, идет о предельно достижимом быстродействии.

Иначе говоря, простые устройства с жесткой логикой работы всегда могут быть сделаны быстрее программируемых, гибких устройств со сложным алгоритмом работы. Жесткая логика также обеспечивает малый объем аппаратуры (малые аппаратурные затраты) для реализации простых функций. Зато программируемые, интеллектуальные устройства обеспечивают более высокую гибкость и меньшую стоимость при необходимости сложной обработки информации. А для реализации простых функций они часто оказываются избыточно сложными. Так что выбор между двумя этими типами цифровых устройств зависит от конкретной решаемой задачи.

Значительное число задач может быть решено как чисто аппаратным путем (с помощью устройств на жесткой логике), так и программно-аппаратным путем (с помощью программируемых устройств). В таких случаях надо смотреть, какие характеристики устройства являются самыми важными: скорость работы, стоимость, гибкость, простота проектирования и т. д., и в зависимости от этого выбирать то или иное решение, так или иначе перераспределять функции между программным обеспечением и аппаратурой.

В данной книге основное внимание будет уделено устройствам и узлам с жесткой логикой работы. Однако уяснение принципов их работы и их проектирования может оказать большую помощь и при создании программируемых, интеллектуальных устройств.

Глава 2
ПРИМЕНЕНИЕ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Изучение базовых элементов цифровой электроники мы начнем с наиболее простых элементов, а затем будем рассматривать все более сложные. Примеры применения каждого следующего элемента будут опираться на все элементы, рассмотренные ранее. Таким образом будут постепенно даны главные принципы построения довольно сложных цифровых устройств.

Логические элементы (или, как их еще называют, вентили, gates) — это наиболее простые цифровые микросхемы. Именно в этой простоте и состоит их отличие от других микросхем. Как правило, в одном корпусе микросхемы может располагаться от одного до шести одинаковых логических элементов. Иногда в одном корпусе могут располагаться и разные логические элементы.

Обычно каждый логический элемент имеет несколько входов (от одного до двенадцати) и один выход. При этом связь между выходным сигналом и входными сигналами (таблица истинности) предельно проста. Каждой комбинации входных сигналов элемента соответствует уровень нуля или единицы на его выходе. Никакой внутренней памяти у логических элементов нет, поэтому они относятся к группе так называемых комбинационных микросхем. Но в отличие от более сложных комбинационных микросхем, рассматриваемых в следующей главе, логические элементы имеют входы, которые не могут быть разделены на группы, различающиеся по выполняемым ими функциям.

Главные достоинства логических элементов по сравнению с другими цифровыми микросхемами — это их высокое быстродействие (малые времена задержек), а также малая потребляемая мощность (малый ток потребления). Поэтому в тех случаях, когда требуемую функцию можно реализовать исключительно на логических элементах, всегда имеет смысл проанализировать этот вариант. Недостаток логических элементов состоит в том, что на их основе довольно трудно реализовать сколько-нибудь сложные функции. Поэтому чаще всего логические элементы

используются только в качестве дополнения к более сложным, к более «умным» микросхемам. И любой разработчик обычно стремится использовать их как можно меньше и как можно реже. Существует даже мнение, что мастерство разработчика обратно пропорционально количеству используемых им логических элементов. Однако это мнение верно далеко не всегда.

2.1. Инверторы

Самый простой логический элемент — это инвертор (логический элемент НЕ, inverter), уже упоминавшийся в первой главе. Инвертор выполняет простейшую логическую функцию — инвертирование, то есть изменение уровня входного сигнала на противоположный. Инвертор имеет всего один вход и один выход. Выход инвертора может быть типа 2С или типа ОК. На рис. 2.1. показаны условные обозначения инвертора, принятые у нас и за рубежом, а в табл. 2.1 представлена таблица истинности инвертора.

Таблица 2.1. Таблица истинности инвертора

Вход	Выход
0	1
1	0

Рис. 2.1. Условные обозначения инверторов: зарубежные (слева) и отечественные (справа).

В одном корпусе микросхемы обычно бывает шесть инверторов. Отечественное обозначение микросхем инверторов — «ЛН». Примеры: КР1533ЛН1 (SN74ALS04) — шесть инверторов с выходом 2С, КР1533ЛН2 (SN74ALS05) — шесть инверторов с выходом ОК. Существуют также инверторы с выходом ОК и с повышенным выходным током (ЛН4) и с повышенным выходным напряжением (ЛН3, ЛН5). Для инверторов с выходом ОК необходимо включение выходного нагрузочного резистора

pull-up. Его минимальную величину можно рассчитать очень просто: $R \cong U/I_{OL}$, где U — напряжение источника питания, к которому подключается резистор. Обычно величина резистора выбирается порядка сотен Ом — единиц кОм.

Две основные области применения инверторов — это изменение полярности сигнала и изменение полярности фронта сигнала (рис. 2.2). То есть из положительного входного сигнала инвертор делает отрицательный выходной сигнал и наоборот, а из положительного фронта входного сигнала — отрицательный фронт выходного сигнала и наоборот. Еще одно важное применение инвертора — буферизация сигнала (с инверсией), то есть увеличение нагрузочной способности сигнала. Это бывает нужно в том случае, когда какой-то сигнал надо подать на много входов, а выходной ток источника сигнала недостаточен.

Рис. 2.2. Инверсия полярности сигнала и инверсия полярности фронта сигнала.

Именно инвертор как наиболее простой элемент чаще других элементов используется в нестандартных включениях. Например, инверторы обычно применяются в схемах генераторов прямоугольных импульсов (рис. 2.3), выходной сигнал которых периодически изменяется с нулевого уровня на единичный и обратно. Все приведенные схемы, кроме схемы *д*, выполнены на элементах К155ЛН1, но могут быть реализованы и на инверторах других серий при соответствующем изменении номиналов резисторов. Например, для серии К555 номиналы резисторов увеличиваются примерно втрое. Схема *д* выполнена на элементах КР531ЛН1, так как она требует высокого быстродействия инверторов.

Схемы *а, б* и *в* представляют собой обычные RC-генераторы, характеристики которых (выходную частоту, длительность импульса) можно рассчитать только приблизительно. Для схем *а* и *б* при указанных номиналах резистора и конденсатора частота генерации составит порядка 100 кГц, для схемы *в* — около 1 МГц. Эти схемы рекомендуется использовать только в тех случаях, когда частота не слишком важна, а важен сам факт ге-

нерации. Если же точное значение частоты принципиально, то рекомендуется использовать схемы *г* и *д*, в которых частота выходного сигнала определяется только характеристиками кварцевого резонатора. Схема *г* используется для кварцевого резонатора, работающего на первой (основной) гармонике. Величину емкости можно оценить по формуле:

$$C > 1/(2RF),$$

где F — частота генерации. Схема *д* применяется для гармониковых кварцевых резонаторов, работающих на частоте, большей основной в 3, 5, 7 раз (это бывает нужно для частот генерации выше 20 МГц).

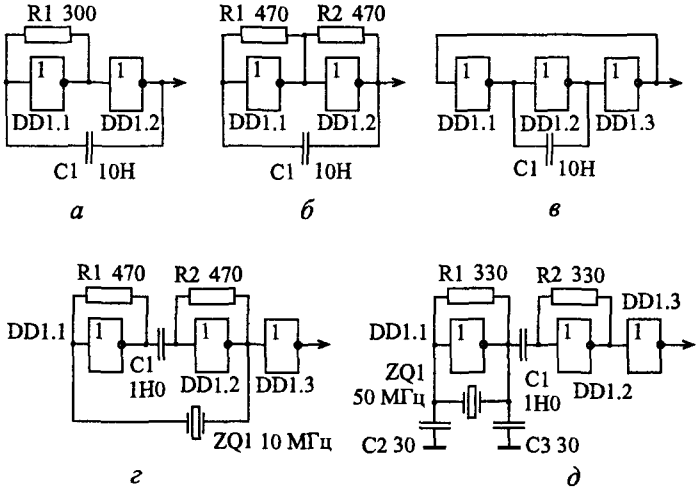


Рис. 2.3. Схемы генераторов импульсов на инверторах.

Инверторы также применяются в тех случаях, когда необходимо получить задержку сигнала, правда, незначительную (от 5 до 100 нс). Для получения такой задержки последовательно включается нужное количество инверторов (рис. 2.4, верхняя схема). Суммарное время задержки, например, для четырех инверторов можно оценить по формуле:

$$t_З = 2t_{PHL} + 2t_{PLH}.$$

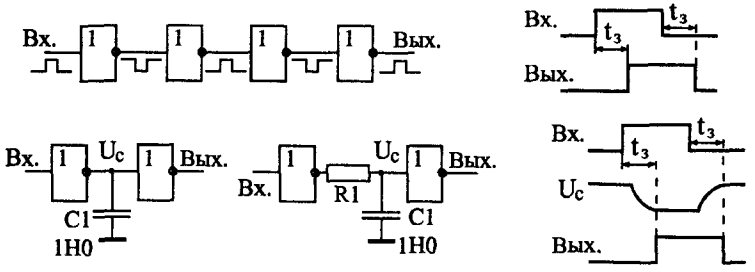


Рис. 2.4. Использование инверторов для задержки сигнала.

Правда, нужно учитывать, что обычно реальные задержки элементов оказываются существенно меньше (иногда даже вдвое), чем табличные значения параметров t_{PHL} и t_{PLH}. То есть о точном значении получаемой задержки говорить не приходится, ее можно оценить только примерно.

Для задержки сигнала используются также конденсаторы (рис. 2.4, две нижние схемы). При этом задержка возникает из-за медленного заряда и разряда конденсатора (напряжение на конденсаторе — U_C). Схема без резистора (внизу слева на рисунке) дает задержку около 100 нс. В схеме с резистором (внизу справа на рисунке) номинал резистора должен быть порядка сотен Ом. Но при выборе таких схем с конденсаторами надо учитывать, что некоторые серии микросхем (например, КР1533) плохо работают с затянутыми фронтами входных сигналов. Кроме того, надо учитывать, что количество времязадающих конденсаторов в схеме обратно пропорционально уровню мастерства разработчика схемы.

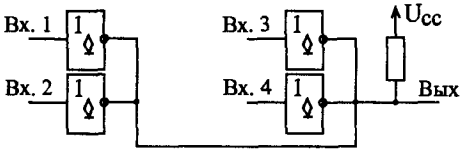


Рис. 2.5. Объединение выходов инверторов с ОК для реализации функции ИЛИ-НЕ.

Наконец, еще одно применение инверторов, но только с выходом ОК, состоит в построении на их основе так называемых элементов «Проводного ИЛИ». Для этого выходы нескольких инвер-

торов с выходами ОК объединяются и через резистор присоединяются к источнику питания (рис. 2.5). Выходом схемы является объединенный выход всех элементов. Такая конструкция выполняет логическую функцию ИЛИ-НЕ, то есть на выходе будет сигнал логической единицы только при нулях на всех входах. Но о логических функциях подробнее будет рассказано в разделе 2.3.

В заключение раздела надо отметить, что инверсия сигнала применяется и внутри более сложных логических элементов, а также внутри цифровых микросхем, выполняющих сложные функции.

2.2. Повторители и буферы

Повторители и буферы отличаются от инверторов прежде всего тем, что они не инвертируют сигнал (правда, существуют и инвертирующие буферы). Зачем же тогда они нужны? Во-первых, они выполняют функцию увеличения нагрузочной способности сигнала, то есть позволяют подавать один сигнал на много входов. Для этого имеются буферы с повышенным выходным током и выходом 2С, например ЛП16 (шесть буферных повторителей). Во-вторых, большинство буферов имеют выход ОК или 3С, что позволяет использовать их для получения двунаправленных линий или для мультиплексирования сигналов. Поясним подробнее эти термины.

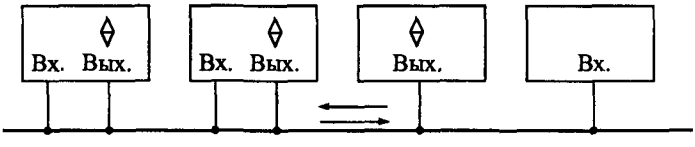


Рис. 2.6. Двунаправленная линия.

Под *двунаправленными линиями* понимаются такие линии (провода), сигналы по которым могут распространяться в двух противоположных направлениях. В отличие от однонаправленных линий, которые идут от одного выхода к одному или нескольким входам, к двунаправленной линии могут одновременно подключаться несколько выходов и несколько входов (рис. 2.6). Понятно, что двунаправленные линии могут органи-

зовываться только на основе выходов ОК или 3С. Поэтому почти все буферы имеют именно такие выходы.

Мультиплексированием называется передача разных сигналов по одним и тем же линиям в разные моменты времени. Основная цель мультиплексирования состоит в сокращении общего количества соединительных линий. Двунаправленная линия обязательно является мультиплексированной, а мультиплексированная линия может быть как однонаправленной, так и двунаправленной. Но в любом случае к ней присоединяется несколько выходов, только один из которых в каждый момент времени находится в активном состоянии. Остальные выходы в это время отключаются (переводятся в пассивное состояние). В отличие от двунаправленной линии к мультиплексированной линии, построенной на основе буферов, может быть подключен всего лишь один вход, но обязательно несколько выходов с ОК или 3С (рис. 2.7). Мультиплексированные линии могут строиться не только на буферах, но и на микросхемах мультиплексоров, которые будут рассмотрены в главе 3.

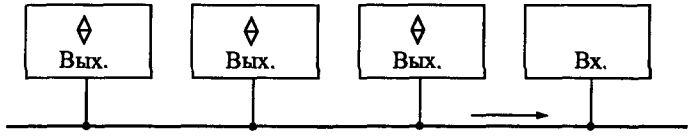


Рис. 2.7. Однонаправленная мультиплексированная линия на основе буферов.

Примером буферов с выходом ОК является микросхема ЛП17 (шесть буферов о ОК). Точно так же, как и в случае инверторов с ОК (см. рис. 2.5), выходы нескольких буферов с ОК могут объединяться для получения функции «Монтажное И», то есть на выходе будет сигнал логической единицы только при единицах на всех входах (рис. 2.8). То есть реализуется многовходовой элемент И (см. раздел 2.3).

Буферы с выходом 3С представлены гораздо большим количеством микросхем, например: ЛП8, ЛП11, АП5, АП6, АП14. Эти буферы обязательно имеют управляющий вход EZ (или OE), переводящий выходы в третье, пассивное состояние. Как правило, третьему состоянию соответствует единица на этом входе, а активному состоянию выходов — нуль, то есть сигнал EZ имеет отрицательную полярность.

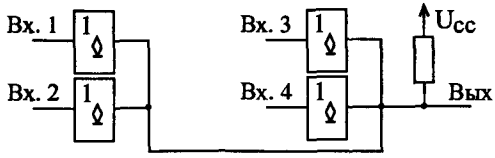


Рис. 2.8. Объединение выходов буферов с ОК.

Буферы бывают однонаправленные или двунаправленные, с инверсией сигналов или без инверсии сигналов, с управлением всеми выходами одновременно или с управлением группами выходов. Все это и определяет большое разнообразие микросхем буферов.

Таблица 2.2. Таблица истинности буфера без инверсии

Вход	-EZ	Выход
0	0	0
1	0	1
0	1	3С
1	1	3С

Простейшим однонаправленным буфером без инверсии является микросхема ЛП8 (четыре буфера с выходами типа 3С и раздельным управлением). Каждый из четырех буферов имеет свой вход разрешения EZ. Таблица истинности буфера очень проста (табл. 2.2.): при нулевом сигнале на входе управления выход повторяет вход, а при единичном — выход отключен. Эту микросхему удобно применять для обработки одиночных сигналов, то есть для повторения входного сигнала с возможностью отключения выхода.

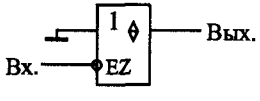


Рис. 2.9. Применение буфера с выходом 3С в качестве буфера с ОК.

Эти же буферы иногда удобно использовать для замещения буферов с выходом ОК (рис. 2.9). В этом случае вход управления служит информационным входом. При нуле на входе мы

получаем нуль на выходе, а при единице на входе — третье состояние на выходе.

Очень часто надо обрабатывать не одиночные сигналы, а группы сигналов, например сигналы, передающие многоразрядные коды. В этом случае удобно применять буферы с групповым управлением, то есть имеющие один вход разрешения EZ для нескольких выходов. Примерами могут служить микросхемы ЛП11 (шесть буферов, разделенные на две группы: четыре и два буфера, для каждой из которых имеется свой вход управления) и АП5 (восемь буферов, разделенные на две группы по четыре буфера, каждая из которых имеет свой вход управления).

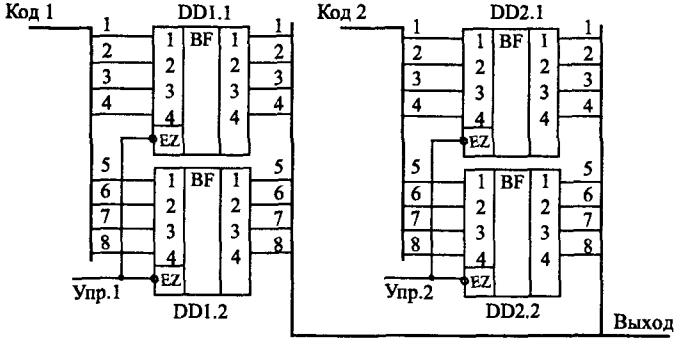


Рис. 2.10. Мультиплексирование двух входных кодов с помощью буферов с выходом 3С.

На рис. 2.10 показан пример мультиплексирования двух восьмиразрядных кодов с помощью двух микросхем АП5. Одноименные выходы обеих микросхем объединены между собой. Пропускание на выход каждого из двух входных кодов разрешается своим управляющим сигналом (Упр.1 и Упр.2), причем должен быть исключен одновременный приход этих двух сигналов, чтобы не было конфликтов на выходах.

Двунаправленные буферы в отличие от однонаправленных позволяют передавать сигналы в обоих направлениях. В зависимости от специального управляющего сигнала T (другое обозначение — BD) входы могут становиться выходами и наоборот — выходы входами. Обязательно имеется и вход управления третьим состоянием EZ, который может отключить как входы, так и выходы.

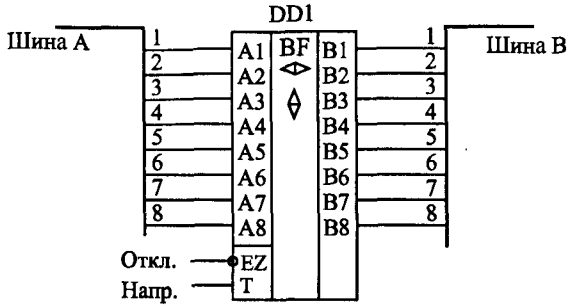


Рис. 2.11. Включение двунаправленного буфера.

На рис. 2.11 для примера показан двунаправленный буфер АП6, который может передавать данные между двумя двунаправленными шинами А и В в обоих направлениях. При единичном уровне на управляющем входе Т (сигнал Напр.) данные передаются из шины А в шину В, а при нулевом уровне — из шины В в шину А (табл. 2.3). Единичный уровень на управляющем входе -EZ (сигнал Откл.) отключает микросхему от обеих шин.

Таблица 2.3. Таблица истинности двунаправленного буфера

Вход Т	Вход -EZ	Операция
0	0	B —> A
1	0	A —>B
0	1	ЗС
1	1	ЗС

Двунаправленную передачу можно организовать и на основе однонаправленных буферов. На рис. 2.12 показано, как это можно сделать на двух микросхемах АП5. Здесь при нулевом сигнале Упр.1 информация будет передаваться с шины А на шину В, а при нулевом сигнале на входе Упр.2 — с шины В на шину А. Если оба входа Упр.1 и Упр.2 находятся в единичном состоянии, то шины А и В отключены друг от друга, а подача нулей на оба входа Упр.1 и Упр.2 должна быть исключена, иначе состояние обеих шин А и В будет не определено.

Микросхемы буферов в отечественной системе обозначений имеют разнообразные обозначения: ЛН, ЛП, АП, ИП (например ЛН6, ЛП8, ЛП11, АП5, АП6, ИП5, ИП6), что порой затрудняет

их выбор. Буферы с буквами ЛН имеют инверсию, буферы АП и ИП могут быть с инверсией, а могут быть и без инверсии. Все параметры у буферов довольно близки, отличие — в инверсии, в количестве разрядов и в управляющих сигналах.

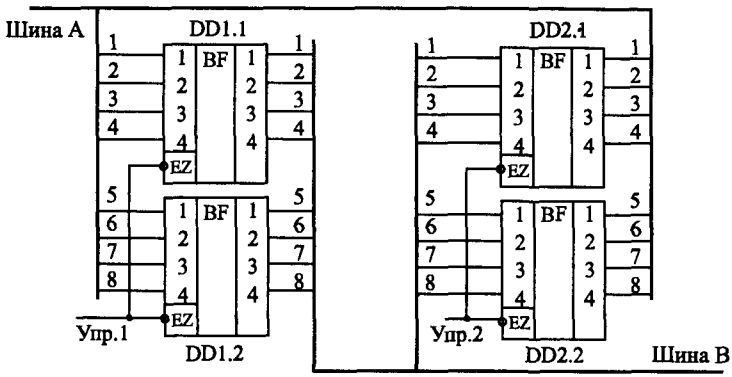


Рис. 2.12. Организация двунаправленной передачи с помощью однонаправленных буферов.

Временные параметры буферов включают помимо задержки сигнала от информационного входа до информационного выхода также задержки перехода выхода в третье состояние и из третьего состояния в активное состояние (t_{PHZ}, t_{PLZ} и t_{PZH}, t_{PZL}). Величины этих задержек обычно примерно вдвое больше, чем величины задержек между информационным входом и выходом.

Отключаемый выход буферов (как ОК, так и 3С) требует применения нагрузочных резисторов. В противном случае вход, подключенный к отключенному выходу, оказывается подвешенным, в результате чего схема может работать неустойчиво, давать сбои. Подключение резистора в случае выхода ОК (pull up) производится стандартным способом (см. рис. 2.8). Точно так же может быть включен резистор между выходом 3С и шиной питания (рис. 2.13), тогда при отключенном выходе на вход будет поступать уровень логической единицы. Однако можно включить и резистор между выходом и землей, тогда при отключенном выходе на вход будет поступать сигнал логического нуля. Применяется также и включение двух резисторов (резистивного делителя), при этом номинал (сопротивление) верхне-

го резистора (присоединенного к шине питания) обычно выбирается в 2–3 раза меньше, чем нижнего резистора (присоединенного к «земле»), а величина сопротивления двух параллельно соединенных резисторов выбирается равной примерно 100 Ом. Например, резисторы могут иметь номиналы 240 Ом и 120 Ом, 360 Ом и 130 Ом. В данном случае отключенный выход воспринимается присоединенным к нему входом как единица.

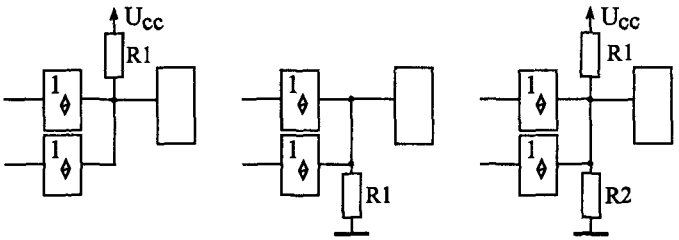


Рис. 2.13. Включение резисторов на выходе буферов 3С.

Иногда к выходам 3С резисторы не присоединяют вообще, но в этом случае надо обеспечить, чтобы последующий вход воспринимал сигнал с выхода 3С (то есть реагировал на него) только тогда, когда выход находится в активном состоянии. Иначе возможны сбои и отказы в работе устройства.

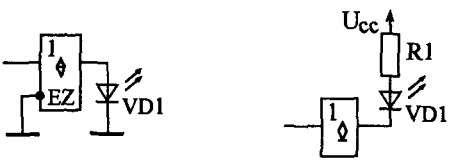


Рис. 2.14. Применение буферов для индикации.

Еще одно типичное применение буферов, связанное с их большими выходными токами — это светодиодная индикация. Светодиоды могут подключаться к выходу буферов двумя основными способами (рис. 2.14). При первом из них (слева на рисунке) светодиод горит, когда на выходе 3С или 2С сигнал логической единицы, а при втором (справа на рисунке) — когда на выходе ОК сигнал логического нуля. Сопротивление резистора выбирается исходя из характеристик светодиода, но обычно составляет порядка 1 кОм.

2.3. Логические элементы И, И-НЕ, ИЛИ, ИЛИ-НЕ

Следующая группа микросхем на пути усложнения компонентов цифровой электроники — это элементы, выполняющие простейшие логические функции. Объединяет все эти элементы то, что у них есть несколько *равноправных* входов (от 2 до 12) и один выход, сигнал на котором определяется комбинацией входных сигналов.

Самые распространенные логические функции, выполняемые такими элементами, — это И (в отечественной системе обозначений микросхем — ЛИ), И-НЕ (обозначается ЛА), ИЛИ (обозначается ЛЛ) и ИЛИ-НЕ (обозначается ЛЛ). Присутствие слова НЕ в названии элемента обозначает только одно — встроенную инверсию сигнала. В международной системе обозначений используются следующие сокращения: AND — функция И, NAND — функция И-НЕ, OR — функция ИЛИ, NOR — функция ИЛИ-НЕ.

Название самих функций И и ИЛИ говорит о том, при каком условии на входах появляется сигнал на выходе. При этом важно помнить, что речь в данном случае идет о положительной логике, о положительных, единичных сигналах на входах и на выходе.

Таблица 2.4. Таблица истинности двухвходовых элементов И, И-НЕ, ИЛИ, ИЛИ-НЕ

Вход 1	Вход 2	Выход И	Выход И-НЕ	Выход ИЛИ	Выход ИЛИ-НЕ
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	0

Элемент И формирует на выходе единицу тогда и только тогда, когда на всех его входах (и на первом, и на втором, и на третьем и т. д.) присутствуют единицы. Если речь идет об элементе И-НЕ, то на выходе формируется нуль, когда на всех входах единицы (табл. 2.4). Цифра перед названием функции говорит о количестве входов элемента. Например, 8И-НЕ — это восьмивходовой элемент И с инверсией на выходе.

Элемент ИЛИ формирует на выходе нуль тогда и только тогда, когда хотя бы на одном из входов присутствует единица

(или на первом, или на втором, или на третьем и т. д.). Элемент ИЛИ-НЕ дает на выходе нуль при наличии хотя бы на одном из входов единицы (табл. 2.4). Пример обозначения: 4ИЛИ-НЕ — четырехвходовой элемент ИЛИ с инверсией на выходе.

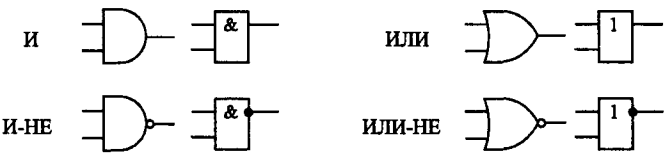


Рис. 2.15. Обозначения элементов И, И-НЕ, ИЛИ, ИЛИ-НЕ: зарубежные (слева) и отечественные (справа).

Отечественные и зарубежные обозначения на схемах двухвходовых элементов И, И-НЕ, ИЛИ, ИЛИ-НЕ показаны на рис. 2.15. Все эти элементы бывают с выходами типа 2С, ОК и 3С. В последнем случае обязательно имеется вход разрешения -EZ.

Нетрудно заметить (см. табл. 2.4), что в случае отрицательной логики, при нулевых входных и выходных сигналах, элемент И выполняет функцию ИЛИ, то есть на выходе будет нуль в случае, когда хотя бы на одном из входов нуль. А элемент ИЛИ при отрицательной логике выполняет функцию И, то есть на выходе будет нуль только тогда, когда на всех входах присутствуют нули. А так как в реальных электронных устройствах сигналы могут быть любой полярности (как положительные, так и отрицательные), то надо всегда очень аккуратно выбирать требуемый в каждом конкретном случае элемент. Особенно важно помнить об этом тогда, когда последовательно соединяются несколько разноименных логических элементов с инверсией и без нее для получения сложной функции.

Поэтому разработчику далеко не всегда удобно рассматривать элементы И, И-НЕ, ИЛИ, ИЛИ-НЕ именно как выполняющие указанные в их названии логические функции. Иногда их удобнее рассматривать как элементы разрешения/запрещения или смешивания/совпадения. Но сначала мы рассмотрим случаи реализации на этих элементах именно логических функций.

На рис. 2.16 приведены примеры формирования элементами выходных сигналов на основании требуемых временных диаграмм входных и выходных сигналов. В случае *а* выходной сигнал должен быть равен единице при двух единичных входных

сигналах, следовательно, нужен элемент 2И. В случае *б* выходной сигнал должен быть равен нулю, когда хотя бы один из входных сигналов равен единице, следовательно, требуется элемент 2ИЛИ-НЕ. Наконец, в случае *в* выходной сигнал должен быть равен нулю при одновременном приходе единичного сигнала Вх.1, нулевого сигнала Вх.2 и единичного сигнала Вх.3. Следовательно, требуется элемент 3И-НЕ, причем сигнал Вх.2 надо предварительно проинвертировать.

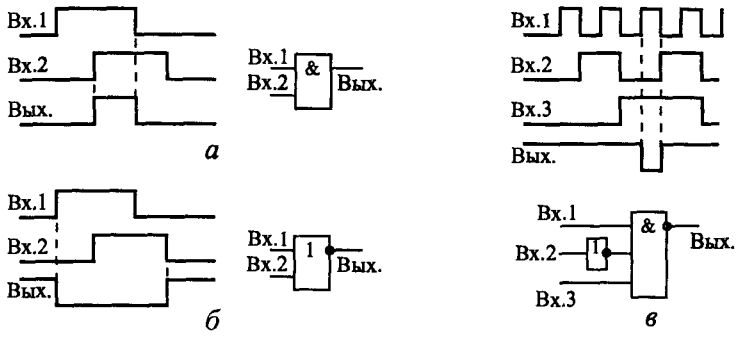


Рис. 2.16. Примеры применения элементов И и ИЛИ.

Любой из логических элементов рассматриваемой группы можно рассматривать как управляемый пропускатель входного сигнала (с инверсией или без нее).

Например, в случае элемента 2И-НЕ один из входов можно считать информационным, а другой — управляющим. В этом случае при единице на управляющем входе выходной сигнал будет равен проинвертированному входному сигналу, а при нуле на управляющем входе выходной сигнал будет постоянно равен единице, то есть прохождение входного сигнала будет запрещено. Элементы 2И-НЕ с выходом ОК часто используют именно в качестве управляемых буферов для работы на мультиплексированную или двунаправленную линию.

Точно так же в качестве элемента разрешения/запрещения могут применяться элементы И, ИЛИ, ИЛИ-НЕ (рис. 2.17). Разница между элементами состоит только в полярности управляющего сигнала, в инверсии (или ее отсутствии) входного сигнала, а также в уровне выходного сигнала (нуль или единица) при запрещении прохождения входного сигнала.

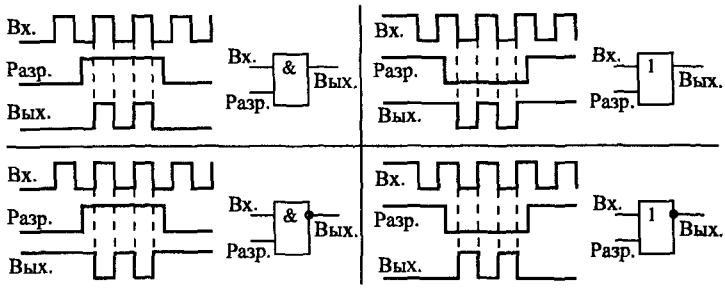


Рис. 2.17. Разрешение/запрещение прохождения сигналов на элементах И, И-НЕ, ИЛИ, ИЛИ-НЕ.

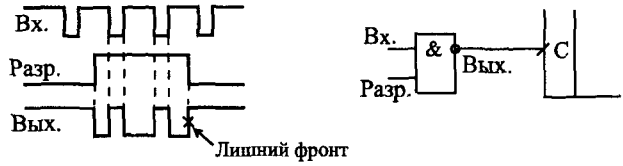


Рис. 2.18. Появление лишнего фронта при запрещении входного сигнала.

При использовании элементов разрешения/запрещения могут возникнуть дополнительные проблемы в случае, когда сигнал с выхода элемента идет на вход, реагирующий на фронт сигнала. В момент перехода из состояния разрешения в состояние запрещения и из состояния запрещения в состояние разрешения в выходном сигнале может появиться дополнительный фронт, никак не связанный с входным сигналом (рис. 2.18). Чтобы этого не произошло, надо придерживаться следующего простого правила: если вход реагирует на положительный фронт, то в состоянии запрещения на выходе элемента должен быть нуль и наоборот.

Иногда необходимо реализовать функцию смешивания двух сигналов той или иной полярности. То есть выходной сигнал должен вырабатываться как при приходе одного входного сигнала, так и приходе другого входного сигнала. Если оба входных сигнала положительные и выходной сигнал положительный, то мы имеем в чистом виде функцию ИЛИ, и требуется элемент 2ИЛИ. Однако при отрицательных входных сигналах и отрицательном выходном сигнале для такого же смешивания

понадобится уже элемент 2И. А если полярность входных сигналов не совпадает с нужной полярностью выходного сигнала, то нужны уже элементы с инверсией (И-НЕ при положительных выходных сигналах и ИЛИ-НЕ при отрицательных выходных сигналах). На рис. 2.19 показаны варианты смешивания на разных элементах.

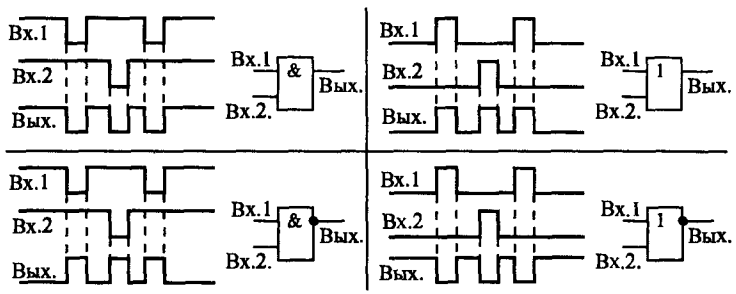


Рис. 2.19. Реализация смешивания двух сигналов.

Наконец, элементы И, И-НЕ, ИЛИ, ИЛИ-НЕ иногда бывает удобно рассматривать в качестве схем совпадения различных сигналов. То есть выходной сигнал должен вырабатываться тогда, когда сигналы на входах совпадают (приходят одновременно). Если же совпадения нет, то выходной сигнал должен отсутствовать. На рис. 2.20 показаны варианты таких схем совпадения на четырех разных элементах. Различаются они полярностями входных сигналов, а также наличием или отсутствием инверсии выходного сигнала.

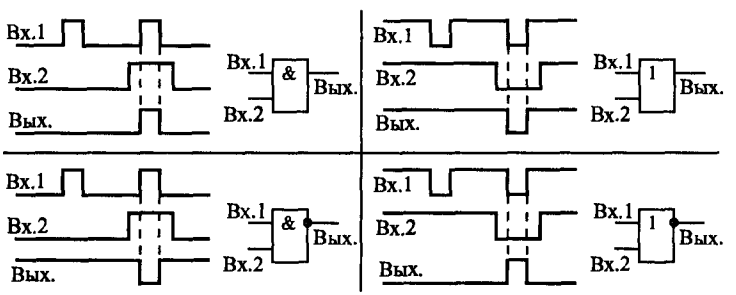


Рис. 2.20. Схемы совпадения двух сигналов.

Рассмотрим два примера совместного использования элементов И, И-НЕ, ИЛИ, ИЛИ-НЕ (рис. 2.21).

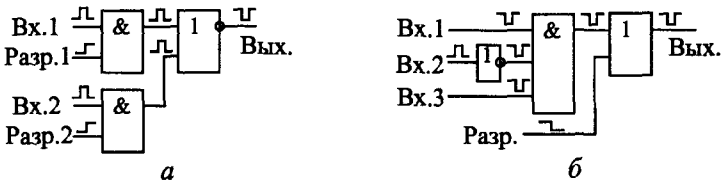


Рис. 2.21. Примеры совместного использования элементов.

Пусть необходимо смешать два сигнала, каждый из которых может быть разрешен или запрещен. Пусть полярность входных сигналов и сигналов разрешения положительная, а выходной сигнал должен быть отрицательным. В этом случае надо взять два двухвходовых элемента И и смешать их выходные сигналы с помощью двухвходового элемента ИЛИ-НЕ (*а*).

Пусть необходимо смешать два отрицательных сигнала и один положительный сигнал, причем результирующий сигнал может быть разрешен или запрещен. Полярность сигнала разрешения — отрицательная, полярность выходного сигнала — отрицательная. Для этого нужно взять трехвходовой элемент И, инвертор для отрицательного входного сигнала и двухвходовой элемент ИЛИ (*б*).

Элементы И, И-НЕ, ИЛИ, ИЛИ-НЕ могут использоваться также в качестве инверторов или повторителей (рис. 2.22), для чего необходимо объединить их входы или на неиспользуемые входы подать сигнал нужного уровня. Второе предпочтительнее, так как объединение входов не только увеличивает входной ток, но и несколько снижает быстродействие элементов.

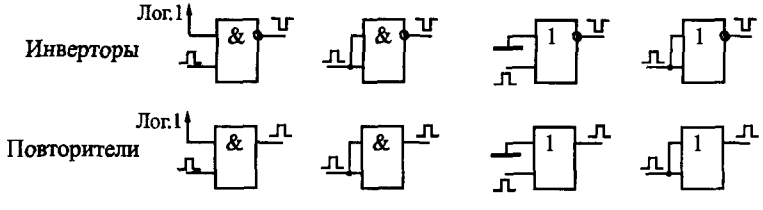


Рис. 2.22. Инверторы и повторители.

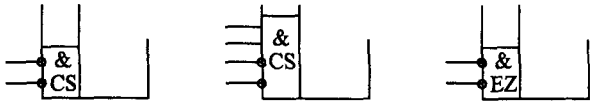


Рис. 2.23. Объединение по И входов микросхем.

По функции И часто объединяются входы более сложных микросхем. То есть какая-то функция выполняется только тогда, когда на все объединенные по И входы поступают сигналы нужной полярности. Чаще всего по И объединяются входы выбора микросхемы CS и входы управления третьим состоянием выхода микросхемы EZ. На рис. 2.23 показано три примера такого объединения по И. При этом надо учитывать, что на инверсные входы для выполнения функции должны поступать нулевые сигналы, а на прямые входы — единичные сигналы. Примерами могут служить микросхемы КР556РТ4, КР556РТ5, КР1533АП14, КР1533АП15.

До сих пор, рассматривая элементы И, И-НЕ, ИЛИ, ИЛИ-НЕ, мы не выходили за рамки первого уровня представления (логической модели). Это вполне допустимо в том случае, когда входные сигналы элементов не меняются одновременно или почти одновременно, когда их фронты разнесены во времени существенно (больше, чем на время задержки элемента). При одновременном изменении входных сигналов все будет гораздо сложнее, необходимо привлекать второй, а иногда и третий уровень представления. В момент изменения входных сигналов выходной сигнал становится неопределенным, нестабильным, непредсказуемым. В результате этого при неправильном проектировании может не работать вся сложная схема, целый прибор или даже большая система.

Например, возьмем логический элемент 2И-НЕ. Пусть на его входы приходят сигналы, изменяющиеся одновременно, причем в противофазе, то есть один переключается из нуля в единицу, а другой — из единицы в нуль. Пусть по тем или иным причинам (вследствие передачи по проводам, вследствие разных задержек элементов и т. д.) один из сигналов чуть-чуть сдвинулся во времени относительно другого (рис. 2.24). При этом на двух входах в течение кратковременного периода будет присутствовать два единичных сигнала. В результате выход начнет переключаться из единицы в нуль. Он может успеть переключиться, и тогда

сформируется короткий импульс. Он может не успеть переключиться, и тогда импульса не будет. Он может иногда успевать переключиться, а иногда не успевать, и тогда выходной импульс то будет появляться, то не будет. Здесь все зависит от быстродействия элемента и величины задержки. Последняя ситуация наиболее неприятна, так как может вызвать нестабильную неисправность, выявить которую крайне сложно.

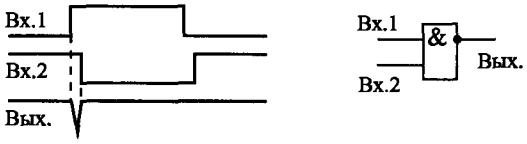


Рис. 2.24. Короткий импульс на выходе элемента 2И-НЕ.

На этапе проектирования схемы бороться с такими паразитными импульсами довольно просто: достаточно выбрать такое схемотехническое решение, при котором вся дальнейшая схема просто не реагировала бы на эти импульсы, например отключалась на некоторое время после изменения входных сигналов элементов. То есть необходимо временное согласование, синхронизация различных элементов схемы.

В качестве примера рассмотрим одно из самых распространенных применений рассматриваемых элементов И, И-НЕ, ИЛИ, ИЛИ-НЕ — селектирование кодов. Суть селектирования сводится к следующему. Пусть имеется некоторая шина, по которой передаются коды. Необходимо выявить появление на этой шине какого-то определенного кода, то есть сформировать выходной сигнал, соответствующий требуемому входному коду.

Схема, выполняющая такую функцию, довольно проста (рис. 2.25). В ее основе — многовходовые элементы И-НЕ. При этом сигналы, соответствующие разрядам кода, в которых должны быть единицы, подаются на входы элементов И-НЕ непосредственно. А сигналы, соответствующие разрядам кода, в которых должны быть нули, подаются на входы элементов И-НЕ через инверторы. Выходные сигналы элементов И-НЕ объединяются с помощью элемента ИЛИ-НЕ. В результате на выходе элемента ИЛИ-НЕ формируется сигнал Вых.1 в тот момент, когда на входе присутствует нужный код.

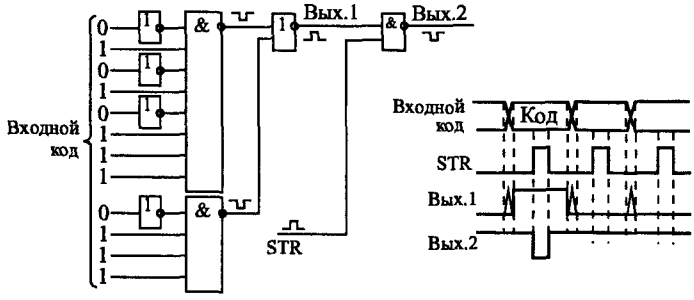


Рис. 2.25. Селектирование кодов со стробированием.

Однако в момент установления нужного кода и в момент его снятия возникает период неопределенности, когда в выходном сигнале могут быть короткие паразитные импульсы. Это связано как с неодновременным приходом различных разрядов, так и с внутренними задержками нашей схемы. Более того, короткие паразитные импульсы могут возникать на выходе и в том случае, когда любой входной код меняется на любой другой входной код, даже если оба этих кода не селектируются нашей схемой. То есть любое изменение кода всегда сопровождается периодом неопределенности в сигнале Вых.1.

Как же добиться, чтобы выходной сигнал не имел паразитных импульсов, не имел периодов неопределенности? Для этого обычно используется стробирование или тактирование передаваемого кода. То есть помимо кода параллельно с ним передается стробирующий или тактирующий сигнал STR, задержанный во времени относительно кода. Активным этот сигнал становится тогда, когда все предыдущие переходные процессы уже завершены, все разряды кода установились в нужные уровни и схема, обрабатывающая код, тоже закончила свою работу. А пассивным этот сигнал становится до начала новых переходных процессов. Это называется *вложенным циклом* (то есть в нашем случае сигнал STR вложен в сигналы кода). В результате, если мы будем разрешать выходной сигнал нашей схемы Вых.1 таким сигналом STR с помощью элемента 2И-НЕ, то мы получим сигнал Вых.2, свободный от паразитных импульсов и периодов неопределенности.

Подробнее о синхронизации будет рассказано в следующих главах книги.

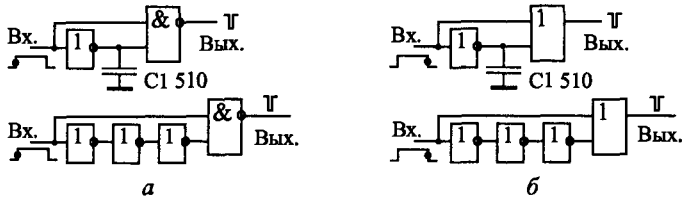


Рис. 2.26. Формирователи коротких импульсов по фронту входного сигнала.

Однако бывают случаи, когда указанная особенность элементов И, И-НЕ, ИЛИ, ИЛИ-НЕ формировать короткие импульсы при изменении входных сигналов оказывается очень полезной. Например, нам необходимо сформировать короткий импульс по положительному или отрицательному фронту имеющегося сигнала. Тогда этот сигнал инвертируют, специально задерживают с помощью цепочки элементов или емкости и подают исходный сигнал и задержанный сигнал на входы элемента (рис. 2.26).

Импульс по положительному фронту входного сигнала формируется на элементе 2И или 2И-НЕ (*а*), а импульс по отрицательному фронту входного сигнала — на элементе 2ИЛИ или 2ИЛИ-НЕ (*б*). Если элемент с инверсией, то выходной импульс будет отрицательным, если без инверсии, то положительным. При указанной на схемах величине емкости длительность импульса получается около 50 нс. Для увеличения длительности импульса надо увеличивать величину емкости или же количество инверторов в цепи задержки (при этом количество инверторов обязательно должно быть нечетным).

2.4. Логические элементы Исключающее ИЛИ

Элементы Исключающее ИЛИ (по-английски — Exclusive-OR) также можно было бы отнести к простейшим элементам, но функция, выполняемая ими несколько сложнее, чем в случае элемента И или элемента ИЛИ. Все входы элементов Исключающее ИЛИ равноправны, однако ни один из входов не может заблокировать другие входы, установив выходной сигнал к уровню единицы или нуля.

Под функцией Исключающим ИЛИ понимается следующее: единица на выходе появляется тогда, когда только на одном

входе присутствует единица. Если единиц на входах две или больше, или если на всех входах нули, то на выходе будет нуль. Таблица истинности двухвходового элемента Исключающее ИЛИ приведена ниже (табл. 2.5). Обозначения, принятые в отечественных и зарубежных схемах, показаны на рис. 2.27. Надпись на отечественном обозначении элемента Исключающее ИЛИ «=1» как раз и обозначает, что выделяется ситуация, когда на входах одна и только одна единица.

Таблица 2.5. Таблица истинности элемента Исключающее ИЛИ

Вход 1	Вход 2	Выход
0	0	0
0	1	1
1	0	1
1	1	0

Элементов Исключающее ИЛИ в стандартных сериях немного. Отечественные серии предлагают микросхемы ЛП5 (четыре двухвходовых элемента с выходом 2С), ЛЛ3 и ЛП12, отличающиеся от ЛП5 выходом ОК. Слишком уж специфическая функция реализуется этими элементами.

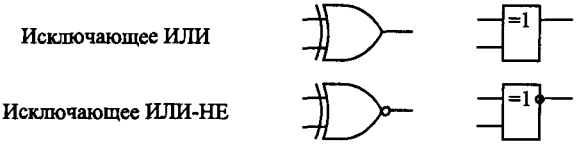


Рис. 2.27. Обозначения элементов Исключающее ИЛИ: зарубежные (слева) и отечественные (справа).

С точки зрения математики, элемент Исключающее ИЛИ выполняет операцию так называемого суммирования по модулю 2. Поэтому эти элементы также называются сумматорами по модулю два. Как уже отмечалось в предыдущей главе, обозначается суммирование по модулю 2 знаком плюса, заключенного в кружок.

Основное применение элементов Исключающее ИЛИ, прямо следующее из таблицы истинности, состоит в сравнении двух входных сигналов. В случае когда на входы приходят две единицы или два нуля (сигналы совпадают), на выходе формирует-

ся нуль (см. табл. 2.5). Обычно при таком применении на один вход элемента подается постоянный уровень, с которым сравнивается изменяющийся во времени сигнал, приходящий на другой вход. Но значительно чаще для сравнения сигналов и кодов применяются специальные микросхемы компараторов кодов, которые будут рассмотрены в следующей главе.

В качестве сумматора по модулю 2 элемент Исключающее ИЛИ используется также в параллельных и последовательных делителях по модулю 2, предназначенных для вычисления циклических контрольных сумм. Но подробно эти схемы будут рассмотрены в гл. 8.

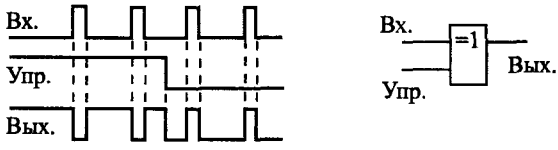


Рис. 2.28. Элемент Исключающее ИЛИ как управляемый инвертор.

Важное применение элементов Исключающее ИЛИ — управляемый инвертор (рис. 2.28). В этом случае один из входов элемента используется в качестве управляющего, а на другой вход элемента поступает информационный сигнал. Если на управляющем входе единица, то входной сигнал инвертируется, если же нуль — не инвертируется. Чаще всего управляющий сигнал задается постоянным уровнем, определяя режим работы элемента, а информационный сигнал является импульсным. То есть элемент Исключающее ИЛИ может изменять полярность входного сигнала или фронта, а может и не изменять в зависимости от управляющего сигнала.

В случае когда имеется два сигнала одинаковой полярности (положительные или отрицательные), и при этом их одновременный приход исключается, элемент Исключающее ИЛИ может быть использован для смешивания этих сигналов (рис. 2.29). При любой полярности входных сигналов выходные сигналы элемента будут положительными. При положительных входных сигналах элемент Исключающее ИЛИ будет работать как элемент 2ИЛИ, а при отрицательных входных сигналах он будет заменять элемент 2И-НЕ. Такие замены могут быть по-

лезны в тех случаях, когда в схеме остаются неиспользованными некоторые элементы Исключающее ИЛИ. Правда, при этом надо учитывать, что задержка распространения сигнала в элементе Исключающее ИЛИ обычно несколько больше (примерно в 1,5 раза), чем задержка распространения в простейших элементах И, И-НЕ, ИЛИ, ИЛИ-НЕ.

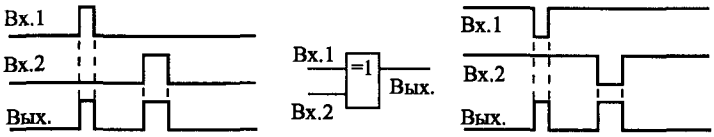


Рис. 2.29. Применение элемента Исключающее ИЛИ для смешивания двух неодновременных сигналов.

Еще одно важнейшее применение элемента Исключающее ИЛИ — формирование коротких импульсов по любому фронту входного сигнала (рис. 2.30). В данном случае не важно, положительный фронт входного сигнала или отрицательный, на выходе все равно формируется положительный импульс. Входной сигнал задерживается с помощью конденсатора или цепочки элементов, а затем исходный сигнал и его задержанная копия поступают на входы элемента Исключающее ИЛИ. В обеих схемах в качестве элементов задержки используются также двухвходовые элементы Исключающее ИЛИ в неинвертирующем включении (на неиспользуемый вход подается нуль). В результате такого преобразования можно говорить об удвоении частоты входного сигнала, так как выходные импульсы следуют вдвое чаще, чем входные.

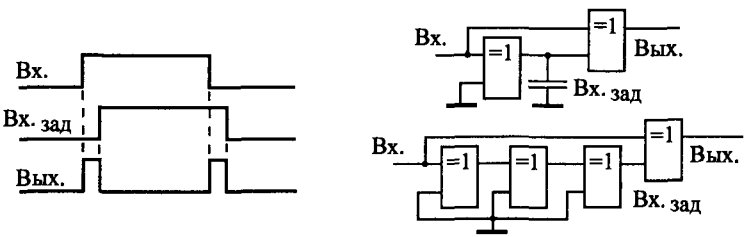


Рис. 2.30. Выделение фронтов входного сигнала с помощью элемента Исключающее ИЛИ.

Данную особенность элементов Исключающее ИЛИ надо учитывать в том случае, когда на оба входа элемента поступают одновременно изменяющиеся сигналы. При этом на выходе элемента возможно появление коротких паразитных импульсов по любому из фронтов входных сигналов. Исключить их влияние на дальнейшую схему можно, например, с помощью синхронизации, подобной рассмотренной в предыдущем разделе.

2.5. Сложные логические элементы

Помимо простейших логических элементов, рассмотренных в предыдущих разделах, в состав стандартных серий входит и несколько более сложных логических элементов. Они представляют собой несложную комбинацию из простейших логических элементов. От более сложных комбинационных микросхем, которым будет посвящена следующая глава, эти элементы отличаются именно очевидной сводимостью к простейшим элементам. Поэтому в справочниках обычно даже не приводятся таблицы истинности этих элементов.

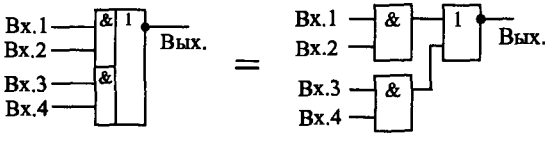


Рис. 2.31. Логический элемент ЛР1 и его эквивалентная схема.

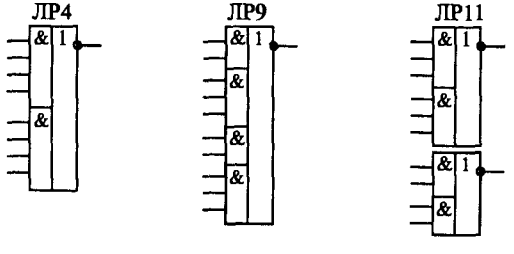


Рис. 2.32. Примеры логических элементов ЛР.

Типичный пример сложного логического элемента — ЛР1. В корпусе микросхемы содержится два элемента, каждый из которых представляет собой комбинацию из двух элементов 2И и одно-

го элемента 2ИЛИ-НЕ (рис. 2.31). По такому же принципу строятся и другие микросхемы ЛР. Разница между ними только в количестве элементов И и в количестве входов этих элементов (рис. 2.32). Некоторые из микросхем ЛР (ЛР1, ЛР3) допускают подключение к специальным входам микросхем расширителей ЛД, хотя такое расширение применяется на практике довольно редко. Микросхема ЛР10 отличается от ЛР9 наличием выхода ОК.

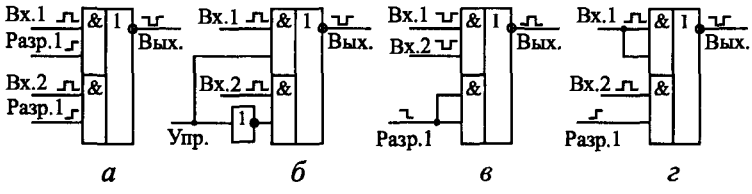


Рис. 2.33. Примеры использования элементов ЛР1.

На рис. 2.33 приведено несколько примеров наиболее типичных применений микросхемы ЛР1. Самое распространенное ее использование (*а*) состоит в организации двухканального мультиплексирования, то есть в переключении сигналов с двух входов на один выход. При этом один из входов каждого из элементов 2И используется в качестве информационного, а другой вход — в качестве разрешающего. Вариант этого включения (*б*) — использование одного управляющего входа переключения каналов и дополнительного инвертора. При единице на управляющем входе работает верхний канал, при нуле — нижний. Еще один вариант использования элемента ЛР1 (*в*) — смешивание двух отрицательных входных сигналов с возможностью разрешения/запрета выходного сигнала. Наконец, последний показанный на рисунке вариант (*г*) — смешивание двух положительных сигналов, один из которых может быть разрешен или запрещен. То есть такое объединение в одном элементе функций И и ИЛИ довольно удобно.

На других элементах ЛР можно строить более сложные схемы. Например, элемент ЛР9 позволяет построить четырехканальный мультиплексор, так как в его структуру входят четыре элемента И и элемент 4ИЛИ-НЕ. Однако в большинстве случаев применение элементов ЛР для мультиплексирования оказывается не слишком удобным, так как в стандартных сериях имеются специальные микросхемы мультиплексоров с более удобным управлением.

При необходимости элементы ЛР1 могут использоваться в качестве более простых элементов 2И-НЕ и 2ИЛИ-НЕ (рис. 2.34). Элемент 2ИЛИ-НЕ получается при попарном объединении входов. Элемент 2И-НЕ получается при отключении половины схемы путем подачи нулей на два входа. При желании можно, конечно, свести элемент ЛР даже к простому инвертору, но это, наверное, уже недопустимая роскошь.

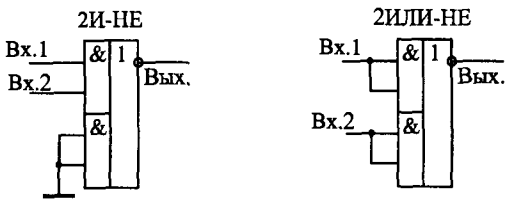


Рис. 2.34. Использование элементов ЛР в качестве элементов 2И-НЕ и 2ИЛИ-НЕ.

К сложным логическим элементам помимо ЛР можно отнести также и элементы И-НЕ с выходом 3С (например, ЛА17 — 4И-НЕ, ЛА19 — 12И-НЕ). Их можно рассматривать как комбинацию обычного элемента И-НЕ и выходного буфера с выходом 3С. Наличие дополнительного управляющего входа -EZ и выход 3С создают принципиально новые возможности применения этих элементов. Например, их можно использовать для работы на мультиплексированную или двунаправленную линию, при этом они еще и выполняют функцию И-НЕ над входными сигналами. Но на практике значительно чаще элемент ЛА19 используют как самый обычный элемент 12И-НЕ с выходом 2С, для чего на управляющий вход -EZ постоянно подается сигнал логического нуля.

Среди элементов И, ИЛИ, ИЛИ-НЕ элементы с выходом 3С отсутствуют.

2.6. Триггеры Шмитта

Триггеры Шмитта представляют собой специфические логические элементы, специально рассчитанные на работу с входными аналоговыми сигналами. Они предназначены для преобразования входных аналоговых сигналов в выходные цифровые сигна-

лы. Появление таких микросхем связано в первую очередь с необходимостью восстановления формы цифровых сигналов, искаженных в результате прохождения по линиям связи. Фронты таких сигналов оказываются пологими, в результате чего форма сигналов вместо прямоугольной может стать близкой к треугольной или синусоидальной. К тому же сигналы, передаваемые на большие расстояния, сильно искажаются шумами и помехами. Для восстановления их формы в исходном виде и устранения влияния помех и шумов как раз и предназначены триггеры Шмитта.

На первом и втором уровнях представления (логическая модель и модель с временными задержками) триггеры Шмитта представляют собой обычные логические элементы, которые с определенной задержкой распространения выполняют логическую функцию над входными цифровыми сигналами. Но на третьем уровне представления их отличие от обычных логических элементов очень существенно.

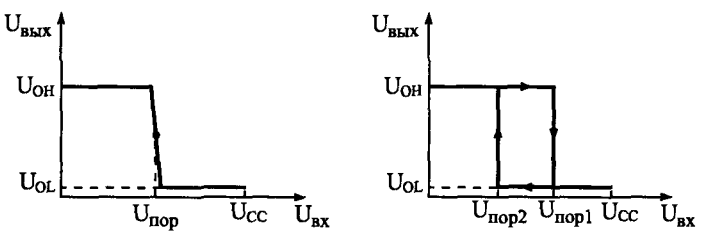


Рис. 2.35. Передаточные характеристики обычного инвертора и триггера Шмитта с инверсией.

Если построить график зависимости выходного напряжения элемента от входного (передаточную характеристику), то для триггера Шмитта он будет гораздо сложнее, чем для обычного элемента (рис. 2.35).

В случае обычного элемента с инверсией (*а*) при входных напряжениях ниже определенного порога срабатывания $U_{пор}$ выходной сигнал имеет высокий уровень, а при входных напряжениях выше этого порога $U_{пор}$ — низкий уровень. При этом не имеет значения, возрастает входное напряжение или убывает.

А в случае триггера Шмитта с инверсией (*б*) принципиально как раз направление изменения сигнала. При возрастании вход-

ного сигнала от нуля до напряжения питания порог срабатывания будет одним ($U_{пор1}$), а при уменьшении сигнала от напряжения питания до нуля порог срабатывания будет другим ($U_{пор2}$), причем $U_{пор1} > U_{пор2}$. В результате на графике образуется своеобразная петля. Выходной сигнал как бы запаздывает переключаться при возврате входного сигнала к исходному уровню. Это называется эффектом гистерезиса (запаздывания).

Наличие гистерезиса приводит к тому, что любой шум, любые помехи с амплитудой, меньшей величины ($U_{пор1} - U_{пор2}$), отсекаются. А любые фронты входного сигнала, даже самые пологие, преобразуются в крутые фронты выходного сигнала. Главное — чтобы амплитуда входного сигнала была большей, чем ($U_{пор1} - U_{пор2}$). На рис. 2.36 показано, как будет реагировать на сигнал с пологими фронтами и с шумами обычный инвертор и триггер Шмитта с инверсией.

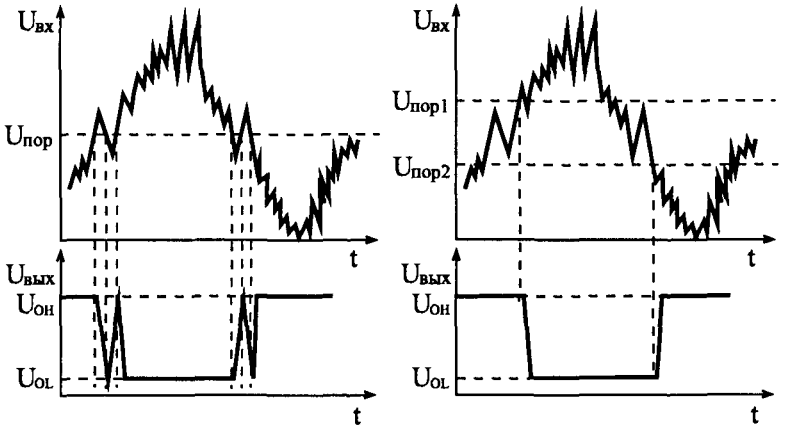


Рис. 2.36. Реакция на искаженный входной сигнал инвертора (слева) и триггера Шмитта с инверсией (справа).

В стандартные серии цифровых микросхем входят триггеры Шмитта, представляющие собой инверторы (ТЛ2 — 6 инверторов), элементы 2И-НЕ (ТЛ3 — 4 элемента) и элементы 4И-НЕ (ТЛ1 — 2 элемента). Пороговые напряжения составляют для всех этих микросхем около 1,7 В ($U_{пор1}$) и около 0,9 В ($U_{пор2}$). Графическое обозначение триггера Шмитта представляет собой

упрощенное изображение его передаточной характеристики с гистерезисом (рис. 2.37).

Наиболее распространенное применение триггеров Шмитта — это формирователь сигнала начального сброса по включению питания схемы. Необходимость такого сигнала сброса вызвана тем, что при включении питания выходные сигналы сложных микросхем, имеющих внутреннюю память (например, регистров, счетчиков) могут принимать произвольные значения, что не всегда удобно. Для приведения их в необходимое состояние (чаще всего — для установки их в нуль) как раз и призван сигнал начального сброса.

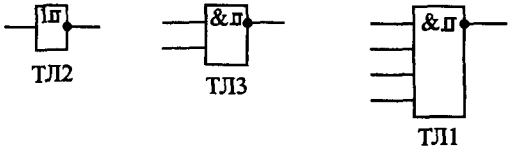


Рис. 2.37. Триггеры Шмитта.

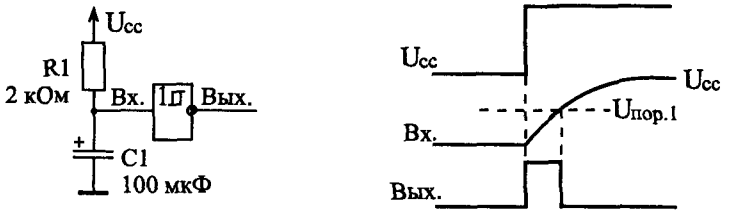


Рис. 2.38. Формирователь импульса начальной установки по включению питания.

Для формирования сигнала начального сброса используется простая RC-цепочка, причем конденсатор берется с большой емкостью. Напряжение на конденсаторе при включении питания нарастает медленно, в результате чего на выходе триггера Шмитта формируется положительный импульс (рис. 2.38). Использовать для этого обычный инвертор не рекомендуется.

Точно так же триггеры Шмитта рекомендуется применять во всех случаях, когда с помощью емкости формируется сигнал с пологими, затянутыми фронтами. В отличие от обычных логи-

ческих элементов триггеры Шмитта всегда обеспечивают надежную и стабильную работу. Правда, надо учитывать, что триггеры Шмитта имеют несколько большую задержку, чем обычные логические элементы.

Еще одно применение триггера Шмитта — построение генераторов импульсов. В отличие от генераторов на обычных инверторах (см. раздел 2.1) в данном случае схема получается гораздо проще: нужен всего лишь один инвертирующий триггер Шмитта, один резистор (с номиналом порядка сотен Ом) и один конденсатор (рис. 2.39). При этом очень удобно, что конденсатор одним выводом присоединен к общему проводу, к «земле». Это позволяет применять электролитические конденсаторы большой емкости, а также переменные конденсаторы. Использование двухвходовых триггеров Шмитта дает возможность легко разрешать или запрещать генерацию с помощью управляющего сигнала Разр. При уровне логической единицы на входе Разр. генерация идет, при уровне логического нуля генерации нет.

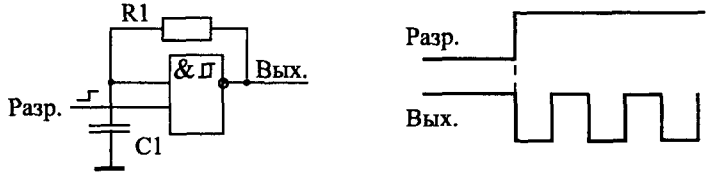


Рис. 2.39. Управляемый генератор на триггере Шмитта.

Нестандартные триггеры Шмитта можно строить также на основе самых обычных логических элементов с обратной связью через резисторы. При этом путем подбора номиналов этих резисторов можно выбирать значения пороговых напряжений триггера Шмитта.

Для примера на рис. 2.40 показана схема триггера Шмитта на инверторах, работающая с входными сигналами, симметричными относительно нулевого уровня. Такие сигналы могут быть, например, в передающем кабеле с трансформаторной развязкой. В данном случае триггер Шмитта не только позволяет восстановить искаженную форму сигнала, но еще и усиливает сигнал, а также сдвигает его уровни до значений стандартных нуля и единицы.

Но чаще всего вполне хватает возможностей стандартных триггеров Шмитта, которые не требуют включения внешних элементов и имеют гарантированные характеристики.

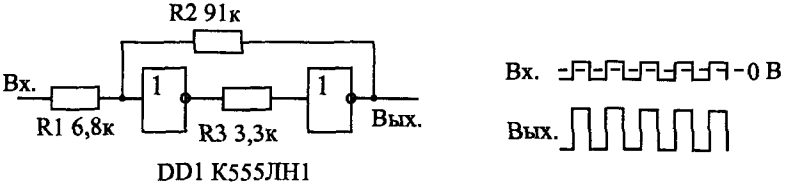


Рис. 2.40. Триггер Шмитта, построенный на обычных логических элементах.

Наконец, последнее применение триггеров Шмитта, которое мы здесь рассмотрим, состоит в подавлении так называемого дребезга контактов. Дело в том, что любой механический контакт (в кнопках, тумблерах, переключателях и т. д.) не замыкается и не размыкается сразу, мгновенно. Любое замыкание и размыкание сопровождается несколькими быстрыми замыканиями и размыканиями, приводящими к появлению паразитных коротких импульсов, которые могут нарушить работу дальнейшей цифровой схемы. Триггер Шмитта с RC-цепочкой на входе позволяет устранить этот эффект (рис. 2.41).

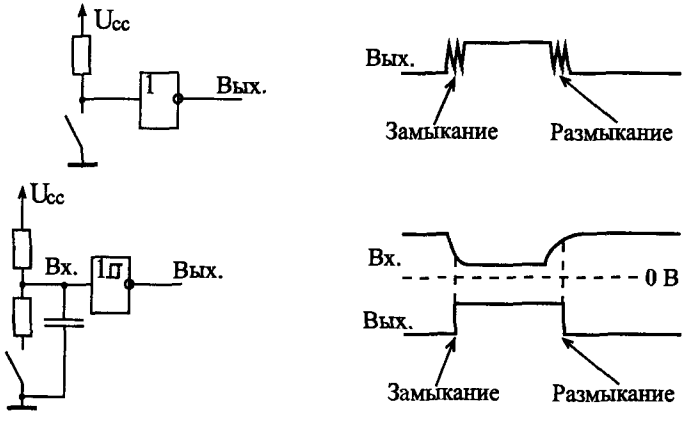


Рис. 2.41. Дребезг контактов (вверху) и его подавление с помощью триггера Шмитта (внизу).

Конденсатор заряжается и разряжается довольно медленно, в результате чего короткие импульсы подавляются и не проходят на выход триггера Шмитта. Номинал верхнего по схеме резистора должен в данном случае быть в 6–7 раз больше номинала нижнего, чтобы резистивный делитель при замкнутом тумблере давал на входе триггера Шмитта уровень логического нуля. Сопротивления резисторов должны быть порядка сотен ом — единиц килоом. Емкость конденсатора может выбираться в широком диапазоне и зависит от того, какова продолжительность дребезга контактов конкретного тумблера.

Глава 3

ПРИМЕНЕНИЕ КОМБИНАЦИОННЫХ МИКРОСХЕМ

Комбинационные микросхемы выполняют более сложные функции, чем простые логические элементы. Их входы объединены в функциональные группы и не являются полностью взаимозаменяемыми. Например, любые два входа логического элемента И-НЕ совершенно спокойно можно поменять местами, от этого выходной сигнал никак не изменится, а для комбинационных микросхем это невозможно, так как у каждого входа своя особая функция.

Объединяет комбинационные микросхемы с логическими элементами то, что и те и другие не имеют внутренней памяти. То есть уровни их выходных сигналов всегда однозначно определяются текущими уровнями входных сигналов и никак не связаны с предыдущими значениями входных сигналов. Любое изменение входных сигналов обязательно изменяет состояние выходных сигналов. Именно поэтому логические элементы иногда также называют комбинационными микросхемами в отличие от последовательных (или последовательностных) микросхем, которые имеют внутреннюю память и управляются не уровнями входных сигналов, а их последовательностями.

Строго говоря, все комбинационные микросхемы построены внутри из простейших логических элементов, и эта их внутренняя структура часто приводится в справочниках. Но для разработчика цифровой аппаратуры эта информация обычно лишняя, ему достаточно знать только таблицу истинности, только принцип преобразования входных сигналов в выходные, а также величины задержек между входами и выходами и уровни входных и выходных токов и напряжений. Внутренняя же структура важна для разработчиков микросхем, а также в тех редчайших случаях, когда надо построить новую комбинационную микросхему из микросхем простых логических элементов.

Состав набора комбинационных микросхем, входящих в стандартные серии, был определен исходя из наиболее часто

встречающихся задач. Требуемые для этого функции реализованы в комбинационных микросхемах наиболее оптимально, с минимальными задержками и минимальным потреблением мощности. Поэтому пытаться повторить эту уже проделанную однажды работу не стоит. Надо просто уметь грамотно применять то, что имеется.

3.1. Дешифраторы и шифраторы

Функции дешифраторов и шифраторов понятны из их названий. Дешифратор преобразует входной двоичный код в номер выходного сигнала (дешифрирует код), а шифратор преобразует номер входного сигнала в выходной двоичный код (шифрует номер входного сигнала). Количество выходных сигналов (и соответствующих им выходов) дешифратора и входных сигналов (и соответствующих им входов) шифратора равно количеству возможных состояний двоичного кода (входного кода у дешифратора и выходного кода у шифратора), то есть 2^n, где n — разрядность двоичного кода (рис. 3.1). Микросхемы дешифраторов обозначаются на схемах буквами DC (от английского Decoder), а микросхемы шифраторов — CD (от английского Coder).

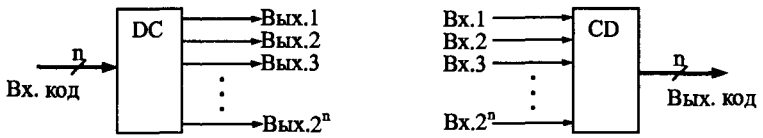


Рис. 3.1. Функции дешифратора (слева) и шифратора (справа).

Активным всегда является только один выход дешифратора, причем номер этого выхода (и соответствующего ему сигнала) однозначно определяется входным кодом. Выходной код шифратора однозначно определяется номером входного сигнала.

Рассмотрим подробнее функцию дешифратора.

В стандартные серии входят дешифраторы на 4 выхода (2 разряда входного кода), на 8 выходов (3 разряда входного кода) и на 16 выходов (4 разряда входного кода). Они обозначаются соответственно как 2–4, 3–8, 4–16. Различаются микросхемы дешифраторов входами управления (разрешения/запрета вы-

ходных сигналов), а также типом выхода: 2С или ОК. Выходные сигналы всех дешифраторов имеют отрицательную полярность. Входы, на которые поступает входной код, называют часто адресными входами. Обозначают эти входы 1, 2, 4, 8, где число соответствует весу двоичного кода (1 — младший разряд, 2 — следующий разряд и т. д.) или A0, A1, A2, A3. В отечественных сериях микросхемы дешифраторов обозначаются буквами ИД. На рис. 3.2 показаны три наиболее типичные микросхемы дешифраторов.

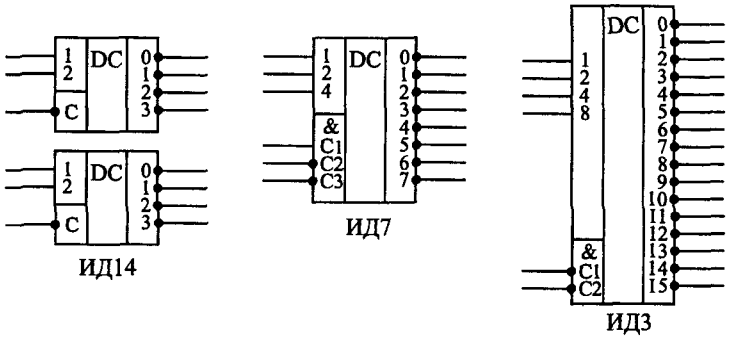


Рис. 3.2. Примеры микросхем дешифраторов.

Код на входах 1, 2, 4, 8 определяет номер активного выхода (вход 1 соответствует младшему разряду кода, вход 8 — старшему разряду кода). Входы разрешения С1, С2, С3 объединены по функции И и имеют указанную на рисунке полярность. В качестве примера ниже приведена таблица истинности дешифратора ИД7 (3–8) (табл. 3.1). Существуют и дешифраторы 4–10 (например, ИД6), которые обрабатывают не все возможные 16 состояний входного кода, а только первые 10 из них.

Первые три строки таблицы истинности соответствуют запрету выходных сигналов. Разрешением выхода будет единица на входе С1 и нули на входах С2 и С3. Символ «X» обозначает безразличное состояние данного входа (неважно, нуль или единица). Нижние восемь строк соответствуют разрешению выходных сигналов. Номер активного выхода (на котором формируется нулевой сигнал) определяется кодом на входах 1, 2, 4, причем вход 1 соответствует младшему разряду кода, а вход 4 — старшему разряду кода.

Таблица 3.1. Таблица истинности дешифратора 3–8 (ИД7)

Входы						Выходы							
C1	-C2	-C3	4	2	1	0	1	2	3	4	5	6	7
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

Наиболее типичное применение дешифраторов состоит именно в дешифрировании входных кодов, при этом входы С используются как стробирующие, управляющие сигналы. Номер активного (то есть нулевого) выходного сигнала показывает, какой входной код поступил. Если нужно дешифровать код с большим числом разрядов, то можно объединить несколько микросхем дешифраторов (пример показан на рис. 3.3).

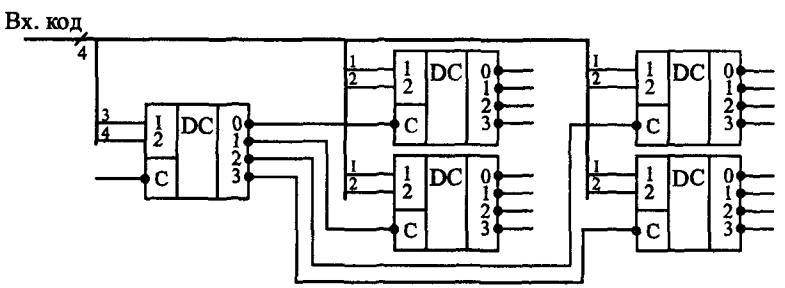


Рис. 3.3. Увеличение разрядов дешифратора.

При этом старшие разряды кода подаются на основной дешифратор, выходы которого разрешают работу нескольких дополнительных дешифраторов. На объединенные входы этих дополнительных дешифраторов подаются младшие разряды входного кода. Используя пять микросхем дешифраторов 2–4,

можно получить дешифратор 4–16, как показано на рисунке (хотя лучше, конечно, взять готовую микросхему). Точно так же на девяти микросхемах 3–8 можно реализовать дешифратор 6–64, а на семнадцати микросхемах 4–16 — дешифратор 8–256.

Еще одно распространенное применение дешифраторов — селекция (выбор) заданных входных кодов. Появление отрицательного сигнала на выбранном выходе дешифратора будет означать поступление на вход интересующего нас кода. В данном случае увеличивать число разрядов входного селектируемого кода гораздо проще, чем в предыдущем (см. рис. 3.3). Например, две микросхемы 4–16 позволяют селектировать 8-разрядный код (рис. 3.4). В примере на рисунке селектируется 16-ричный код 2А (двоичный код 0010 1010). При этом один дешифратор работает с четырьмя младшими разрядами кода, а другой — с четырьмя старшими разрядами. Объединяются дешифраторы так, что один из них разрешает работу другого по входам -С1 и -С2. Применяя механические переключатели выходов дешифраторов (тумблеры, перемычки), можно легко изменять код, селектируемый данной схемой.

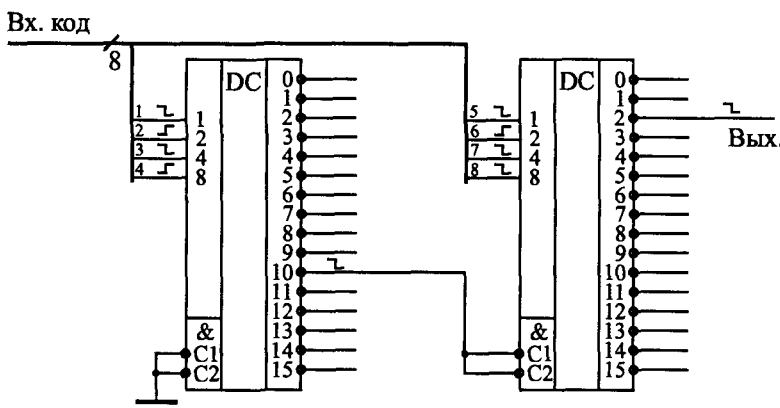


Рис. 3.4. Селектирование кода на дешифраторах.

Еще одно важное применение дешифраторов состоит в перекоммутации одного входного сигнала на несколько выходов. Или, как еще можно сказать, дешифратор в данном случае выступает в качестве демультиплексора входных сигналов, который позволяет разделить входные сигналы, приходящие в разные моменты

времени, на одну входную линию (мультиплексированные сигналы). При этом входы 1, 2, 4, 8 дешифратора используются в качестве управляющих, адресных, определяющих, на какой выход переслать пришедший в данный момент входной сигнал (рис. 3.5). А сам сигнал подается на один из входов С и пересылается на заданный выход. Если у микросхемы имеется несколько стробирующих входов С, то оставшиеся входы С можно использовать в качестве разрешающих работу дешифратора.

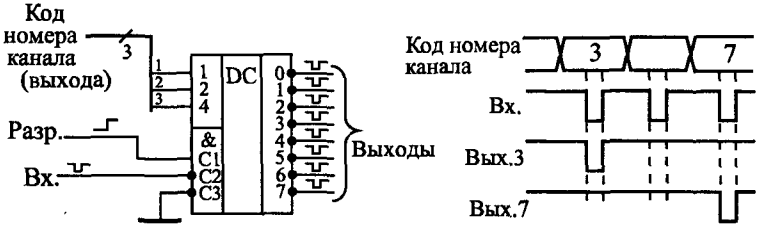


Рис. 3.5. Включение дешифратора как демультиплексора.

Как и для любых других цифровых микросхем, для дешифраторов наиболее критична ситуация одновременного или почти одновременного изменения входных сигналов. Например, если стробы С постоянно разрешают работу дешифратора, то в момент изменения входного кода на любом выходе дешифратора могут появиться паразитные отрицательные короткие импульсы. Это может быть связано как с неодновременным выставлением разрядов кода (из-за несовершенства микросхем источников кода или из-за разных задержек распространения по линиям связи), так и с внутренними задержками самих микросхем дешифраторов.

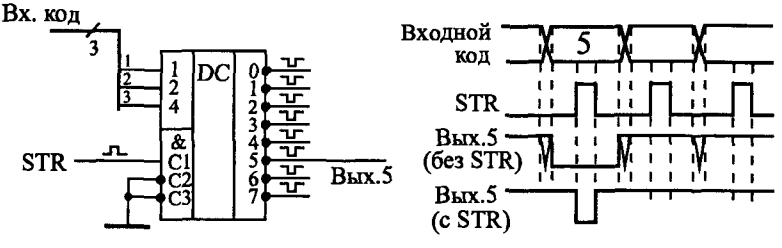


Рис. 3.6. Стробирование выходных сигналов дешифратора.

Для исключения таких паразитных импульсов можно применить синхронизацию с помощью стробирующих сигналов. Используемый для этого сигнал С должен начинаться *после* текущего изменения кода, а заканчиваться *до* следующего изменения кода. То есть должен быть реализован вложенный цикл. На рис. 3.6 показано, как будет выглядеть выходной сигнал дешифратора без стробирования и со стробированием.

На втором уровне представления (модель с временными задержками) нужно также учитывать то, что задержки дешифратора больше задержек простых логических элементов: примерно вдвое для входного кода и примерно в полтора раза для стробирующих входов. Если попытаться заменить дешифратор схемой на логических элементах, то такой дешифратор получится медленнее. Точные величины задержек можно найти в справочниках.

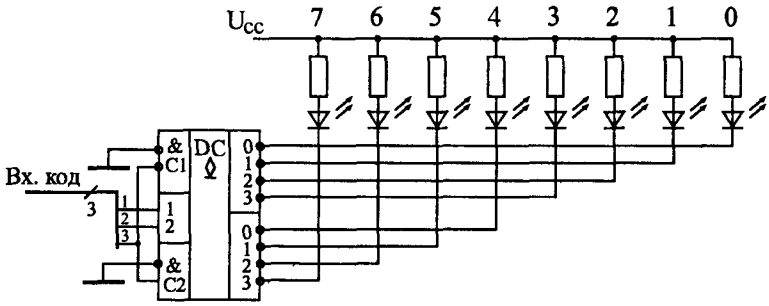


Рис. 3.7. Позиционная индикация на дешифраторе с выходами ОК.

Дешифраторы, имеющие выходы типа ОК (ИД5, ИД10), удобно применять в схемах позиционной индикации на светодиодах. На рис. 3.7 приведен пример реализации позиционного индикатора на микросхеме ИД5, которая представляет собой два дешифратора 2–4 с объединенными входами для подачи кода и стробами, позволяющими легко строить дешифратор 3–8. При этом старший разряд кода выбирает один из дешифраторов 2–4 (нуль соответствует верхнему по схеме дешифратору, а единица — нижнему). То есть в данном случае номер горящего светодиода равен входному коду дешифратора. Такая индикация называется позиционной.

Выходы микросхем дешифраторов с ОК можно объединять между собой для реализации проводного ИЛИ (рис. 3.8.). Нуль на

объединенном выходе будет тогда, когда хотя бы на одном из выходов вырабатывается нуль. При равномерном пошаговом наращивании входного кода (например, с помощью счетчика) такое решение позволяет формировать довольно сложные последовательности выходных сигналов. Правда, каждый выход дешифратора может использоваться для получения только одного выходного сигнала. Это ограничивает возможности таких схем.

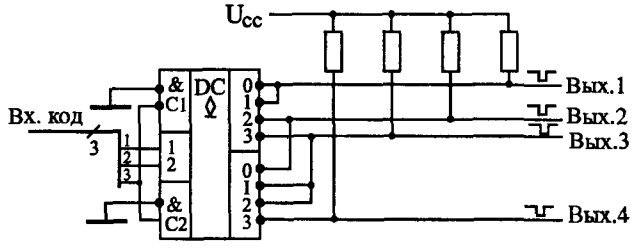


Рис. 3.8. Объединение выходов дешифраторов с ОК.

Шифраторы применяются гораздо реже, чем дешифраторы. Это связано с более специфической областью их применения. Значительно меньше и выбор микросхем шифраторов в стандартных сериях. В отечественных сериях шифраторы имеют в названии буквы ИВ.

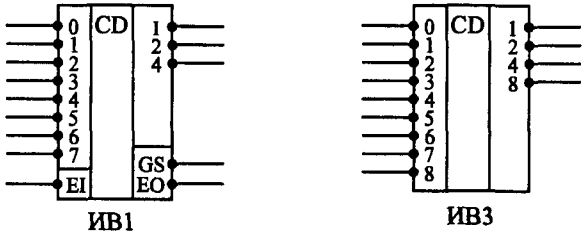


Рис. 3.9. Микросхемы шифраторов.

На рис. 3.9 показаны для примера две микросхемы шифраторов: ИВ1 и ИВ3. Первая имеет 8 входов и 3 выхода (шифратор 8–3), а вторая — 9 входов и 4 выхода (шифратор 9–4). Все входы шифраторов инверсные (активные входные сигналы — нулевые). Все выходы шифраторов тоже инверсные, то есть

формируется инверсный код. Микросхема ИВ1 помимо 8 информационных входов и 3 разрядов выходного кода (1, 2, 4) имеет инверсный вход разрешения -EI, выход признака прихода любого входного сигнала -GS, а также выход переноса -EO, позволяющий объединять несколько шифраторов для увеличения разрядности.

Таблица истинности шифратора ИВ1 приведена ниже (табл. 3.2).

Таблица 3.2. Таблица истинности шифратора ИВ1

Входы									Выходы				
-EI	0	1	2	3	4	5	6	7	-GS	4	2	1	-EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	0	1	1
0	X	X	X	X	X	0	1	1	0	0	1	0	1
0	X	X	X	X	0	1	1	1	0	0	1	1	1
0	X	X	X	0	1	1	1	1	0	1	0	0	1
0	X	X	0	1	1	1	1	1	0	1	0	1	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	0	1	1	1	1

Из таблицы видно, что на выходах кода 1, 2, 4 формируется инверсный двоичный код номера входной линии, на который приходит отрицательный входной сигнал. При одновременном поступлении нескольких входных сигналов формируется выходной код, соответствующий входу с наибольшим номером. То есть старшие входы имеют приоритет перед младшими. Поэтому такой шифратор называется приоритетным. При отсутствии входных сигналов (вторая строчка таблицы) формируется выходной код 111. Единичный сигнал -EI (первая строчка) запрещает работу шифратора (все выходные сигналы устанавливаются в единицу). На выходе -GS вырабатывается нуль при приходе любого входного сигнала, что позволяет, в частности, отличить ситуацию прихода нулевого входного сигнала от ситуации отсутствия любых входных сигналов. Выход -EO становится активным (нулевым) при отсутствии входных сигналов, но при разрешении работы шифратора сигналом -EI.

Стандартное применение шифраторов состоит в сокращении количества сигналов. Например, в случае шифратора ИВ1 информация о восьми входных сигналах сворачивается в три выходных сигнала. Это очень удобно, например, при передаче сигналов на большие расстояния. Правда, входные сигналы не должны приходить одновременно. На рис. 3.10 показаны стандартная схема включения шифратора и временные диаграммы его работы.

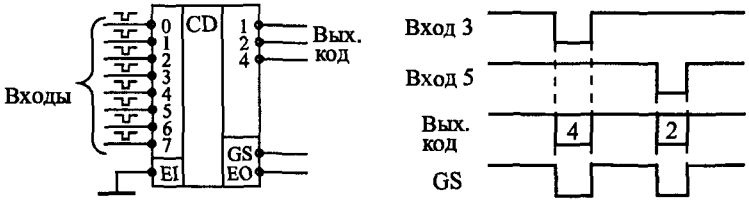


Рис. 3.10. Стандартное включение шифратора.

Инверсия выходного кода приводит к тому, что при приходе нулевого входного сигнала на выходе формируется не нулевой код, а код 111, то есть 7. Точно так же при приходе, например, третьего входного сигнала на выходе формируется код 100, то есть 4, а при приходе пятого выходного сигнала — код 010, то есть 2.

Наличие у шифраторов входов -EI и -EO позволяет увеличивать количество входов и разрядов шифратора, правда, с помощью дополнительных элементов на выходе. На рис. 3.11 показан пример построения шифратора 16–4 на двух микросхемах шифраторов ИВ1 и трех элементах 2И-НЕ (ЛА3).

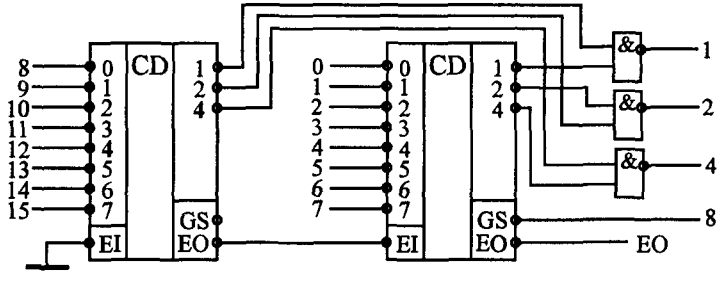


Рис. 3.11. Шифратор 16–4 на двух шифраторах 8–3.

Одновременное или почти одновременное изменение сигналов на входе шифратора приводит к появлению периодов неопределенности на выходах. Выходной код может на короткое время принимать значение, не соответствующее ни одному из входных сигналов. Поэтому в тех случаях, когда входные сигналы могут приходить одновременно, необходима синхронизация выходного кода, например, с помощью разрешающего сигнала -EI, который должен приходить только тогда, когда состояние неопределенности уже закончилось.

Задержка шифратора от входа до выхода кода примерно в полтора раза превышает задержку логического элемента, а задержка до выхода -GS — примерно в два раза больше. Точные величины задержек микросхем можно найти в справочниках.

3.2. Мультиплексоры

Мультиплексоры (английское Multiplexer) предназначены для поочередной передачи на один выход одного из нескольких входных сигналов, то есть для их мультиплексирования. Количество мультиплексируемых входов называется количеством каналов мультиплексора, а количество выходов называется числом разрядов мультиплексора. Например, 2-канальный 4-разрядный мультиплексор имеет 4 выхода, на каждый из которых может передаваться один из двух входных сигналов. А 4-канальный 2-разрядный мультиплексор имеет 2 выхода, на каждый из которых может передаваться один из четырех входных сигналов. Число каналов мультиплексоров, входящих в стандартные серии, составляет от 2 до 16, а число разрядов — от 1 до 4, причем чем больше каналов имеет мультиплексор, тем меньше у него разрядов.

Управление работой мультиплексора (выбор номера канала) осуществляется с помощью входного кода адреса. Например, для 4-канального мультиплексора необходим 2-разрядный управляющий (адресный) код, а для 16-канального — 4-разрядный код. Разряды кода обозначаются 1, 2, 4, 8 или A0, A1, A2, A3. Мультиплексоры бывают с выходом 2С и с выходом 3С. Выходы мультиплексоров бывают прямыми и инверсными. Выход 3С позволяет объединять выходы мультиплексоров с выходами других микросхем, а также получать двунаправленные и мультиплексированные линии. Некоторые микросхемы мультиплек-

соров имеют вход разрешения/запрета С (другое обозначение — S), который при запрете устанавливает на прямом выходе нулевой уровень.

На рис. 3.12 показаны для примера несколько микросхем мультиплексоров из состава стандартных серий. В отечественных сериях мультиплексоры имеют код типа микросхемы КП. На схемах микросхемы мультиплексоров обозначаются буквами MS.

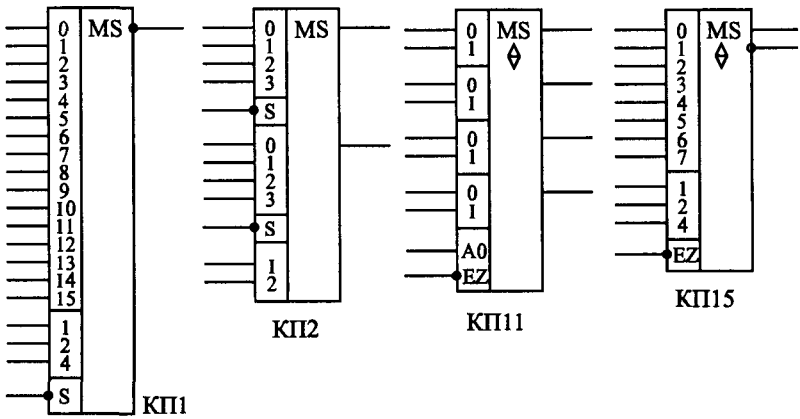


Рис. 3.12. Примеры микросхем мультиплексоров.

Ниже в качестве примера приведена таблица истинности одноразрядного 8-канального мультиплексора с выходами 3С (КП15) (табл. 3.3).

Таблица 3.3. Таблица истинности 8-канального мультиплексора

Входы				Выходы	
4	2	1	-EZ	Q	-Q
X	X	X	1	Z	Z
0	0	0	0	D0	-D0
0	0	1	0	D1	-D1
0	1	0	0	D2	-D2
0	1	1	0	D3	-D3
1	0	0	0	D4	-D4
1	0	1	0	D5	-D5
1	1	0	0	D6	-D6
1	1	1	0	D7	-D7

В таблице сигналы на входах 0...7 обозначены D0...D7, Q — прямой выход, -Q — инверсный выход, Z — третье состояние выхода. При единице на входе -EZ оба выхода находятся в третьем состоянии. При нуле на входе -EZ выходной сигнал на прямом выходе повторяет состояние входного сигнала, номер которого задается входным кодом на входах 1, 2, 4. Сигнал на инверсном выходе противоположен по полярности сигналу на прямом выходе.

На рис. 3.13 приведена временная диаграмма работы 4-канального мультиплексора. В зависимости от входного кода на выход передается один из четырех входных сигналов. При запрещении работы на выходе устанавливается нулевой сигнал вне зависимости от входных сигналов.

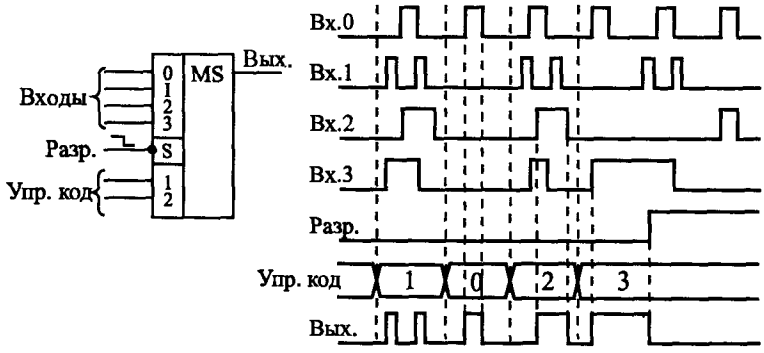


Рис. 3.13. Временная диаграмма работы 4-канального мультиплексора с разрешением.

Микросхемы мультиплексоров можно объединять для увеличения количества каналов. Например, два 8-канальных мультиплексора легко объединяются в 16-канальный с помощью инвертора на входах разрешения и элемента 2И-НЕ для смешивания выходных сигналов (рис. 3.14). Старший разряд кода будет при этом выбирать один из двух мультиплексоров. Точно так же из двух 16-канальных мультиплексоров можно сделать 32-канальный. Если нужно большее число каналов, то необходимо вместо инвертора включать дешифратор, на который подаются старшие разряды кода. Выходные сигналы дешифратора будут выбирать один из мультиплексоров.

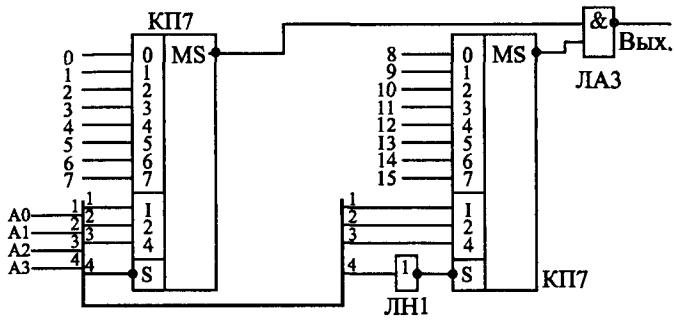


Рис. 3.14. Объединение мультиплексоров для увеличения количества каналов.

Состояния неопределенности, сопровождающиеся короткими паразитными импульсами, могут возникать на выходе мультиплексоров при почти одновременном изменении входных сигналов. Здесь возможны две ситуации. Во-первых, управляющий код может переключиться сразу после изменения передаваемого в данный момент на выход входного сигнала или сразу перед изменением входного сигнала, который будет передавать на выход следующий код. Во-вторых, сигналы (разряды) управляющего кода могут переключаться не одновременно, что приведет к кратковременной передаче на выход входного сигнала, не соответствующего ни одному из значений кода. В любом случае в момент переключения каналов сигнал на выходе мультиплексора не определен (рис. 3.15).

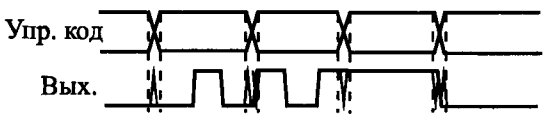


Рис. 3.15. Неопределенные состояния на выходе мультиплексора.

Чтобы избежать состояния неопределенности, лучше всего задавать состояние управляющего кода еще до начала работы схемы (до прихода входных сигналов) и в дальнейшем его не менять. Если же это невозможно, то необходима синхронизация, стробирование выходного сигнала, то есть его разрешение только тогда, когда все переходные процессы, связанные с изменением кода, уже закончились. Правда, обычно применять стро-

бирование довольно непросто, так как мультиплексор, как правило, должен без изменений передавать любой входной сигнал.

Задержки выходного сигнала мультиплексора по входам управляющего (адресного) кода примерно в два раза превышают задержки логических элементов, а по информационным входам — примерно в полтора раза. Точные величины задержек можно найти в справочниках.

3.3. Компараторы кодов

Микросхемы компараторов кодов (английское Comparator) применяются для сравнения двух входных кодов и выдачи на выходы сигналов о результатах этого сравнения (о равенстве или неравенстве кодов). На схемах компараторы кодов обозначаются двумя символами равенства: «= =». Код типа микросхемы компаратора кода в отечественных сериях — СП.

Примером микросхемы компаратора кодов может служить СП1 — четырехразрядный компаратор кодов, сравнивающий величины кодов и выдающий информацию о том, какой код больше, или о равенстве кодов (рис. 3.16).

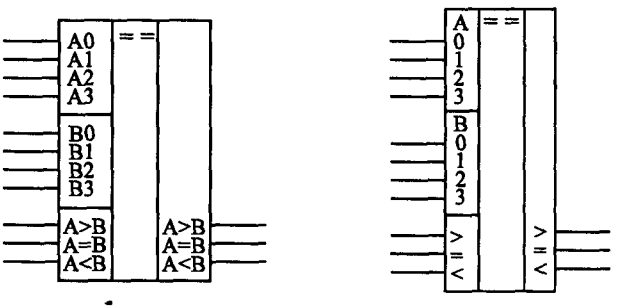


Рис. 3.16. Четырехразрядный компаратор кодов СП1 (два варианта обозначения).

Помимо восьми входов для сравниваемых кодов (двух четырехразрядных кодов, обозначаемых A0...A3 и B0...B3) компаратор СП1 имеет три управляющих входа для наращивания разрядности (A>B, A<B, A=B) и три выхода результирующих сигналов (A>B, A<B, A=B). Для удобства на схемах управляющие входы и выходы иногда обозначают просто «>», «<» и «=». Нулевые разряды кодов (A0 и B0) — младшие, третьи разряды (A3 и B3) — старшие.

Таблица 3.4. Таблица истинности компаратора СП1

Входы сравниваемых кодов				Входы наращивания			Выходы		
A3,B3	A2,B2	A1,B1	A0,B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	1	0	0
A3<B3	X	X	X	X	X	X	0	1	0
A3=B3	A2>B2	X	X	X	X	X	1	0	0
A3=B3	A2<B2	X	X	X	X	X	0	1	0
A3=B3	A2=B2	A1>B1	X	X	X	X	1	0	0
A3=B3	A2=B2	A1<B1	X	X	X	X	0	1	0
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	1	0	0
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	1	0	0	1	0	0
A3=B3	A2=B2	A1=B1	A0=B0	0	1	0	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	X	X	1	0	0	1
A3=B3	A2=B2	A1=B1	A0=B0	1	1	0	0	0	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	0	1	1	0

Таблица истинности компаратора кодов (табл. 3.4) кажется на первый взгляд довольно сложной, но на самом деле все просто.

Если используется одиночная микросхема, то для ее правильной работы достаточно подать единицу на вход A=B, а состояния входов A<B и A>B не важны: на них можно подать как нуль, так и единицу. Назначение выходов понятно из их названия, а полярность выходных сигналов положительная (активный уровень — единица). Если микросхемы компараторов кодов каскадируются (объединяются) для увеличения числа разрядов сравниваемых кодов, то выходные сигналы микросхемы, обрабатывающей младшие разряды кода, нужно подать на одноименные входы микросхемы, обрабатывающей старшие разряды кода (рис. 3.17).

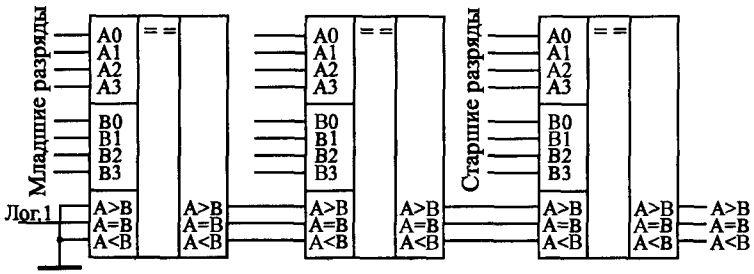


Рис. 3.17. Каскадирование компараторов кодов.

В зарубежные стандартные серии входят также и 8-разрядные компараторы, сравнивающие два кода по величине (то есть имеющие выходы «=», «>» и «<»). Примером может служить микросхема SN74AS885.

Одно из основных применений компараторов кодов состоит в селектировании входных кодов. В этом случае достаточно иметь информацию только о совпадении кодов на входах компаратора, а не о соотношении их величин. Интересующий нас код (эталонный) подается на один вход компаратора, а изменяющийся код (входной) — на другой вход. Используется только выход равенства кодов A=B. Для подобных применений выпускаются и специальные компараторы, определяющие только совпадение кодов. Примерами могут служить 8-разрядные микросхемы SN74ALS520, SN74ALS521, DC102A фирмы Dionics (отечественный аналог — КР559СК1), а также 6-разрядная микросхема DM8136 фирмы National Semiconductors (отечественный аналог — КР559СК2).

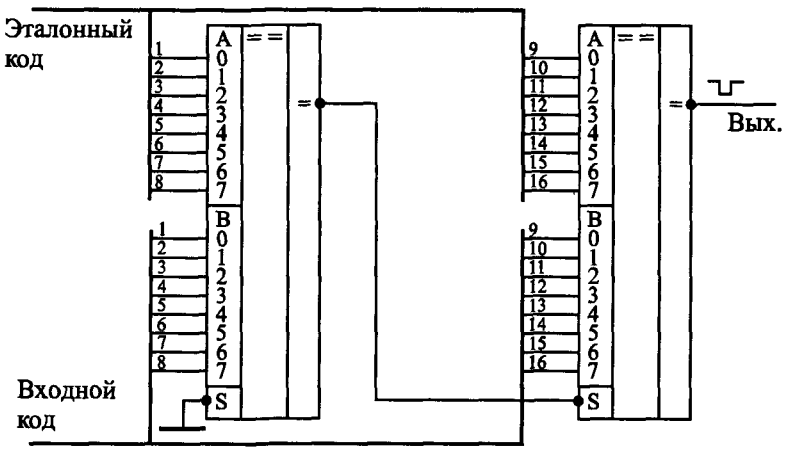


Рис. 3.18. Селектирование 16-разрядных кодов.

На рис. 3.18 показано применение компараторов SN74ALS521 для селектирования 16-разрядных кодов. Инверсный сигнал с выхода первой микросхемы подается на инверсный вход разрешения второй микросхемы, выходной сигнал которой (отрицательный) говорит о совпадении входного и эталонного кодов.

Неопределенные состояния на выходах компараторов кодов могут возникать при любом изменении любого из двух входных кодов. Это связано с неодновременным изменением разрядов кодов (рис. 3.19). На всех выходах компаратора СП1 могут появляться короткие паразитные импульсы. Чтобы устранить их влияние на дальнейшую часть схемы, применяется синхронизация и стробирование. Но для этого надо точно знать момент изменения входных кодов, что далеко не всегда возможно.

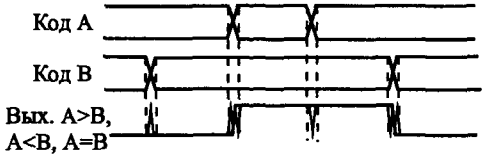


Рис. 3.19. Неопределенные состояния на выходах компаратора при изменении входных кодов.

При применении компараторов надо также учитывать, что при каскадировании задержки микросхем суммируются, и объединенный компаратор, состоящий из n микросхем, будет в n раз медленнее одиночного. Задержки компараторов кодов по входам разрядов кодов примерно вчетверо больше задержек логических элементов, а по входам расширения — примерно втрое. Так что эти микросхемы довольно медленные по сравнению с другими комбинационными микросхемами. Точные значения задержек можно найти в справочниках.

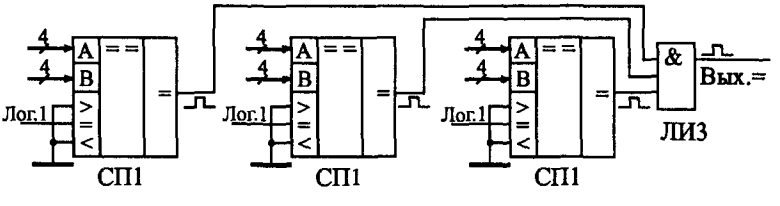


Рис. 3.20. Уменьшение задержки при каскадировании компараторов.

Если нам важен только факт равенства или неравенства входных кодов, то увеличить быстродействие при объединении компараторов можно, если подавать их выходные сигналы на элемент И (рис. 3.20). В этом случае суммарная задержка схемы

превысит задержку одного компаратора всего лишь на задержку элемента И. При применении компараторов с инверсным выходом (например, SN74ALS521) нужно использовать элемент ИЛИ с нужным числом входов.

При необходимости сравнения кодов не только на совпадение, но еще и по величине, такого простого решения не существует. Но эта задача встречается гораздо реже.

3.4. Сумматоры

Микросхемы сумматоров (английское Adder), как следует из их названия, предназначены для суммирования двух входных двоичных кодов. То есть выходной код равен арифметической сумме двух входных кодов. Например, если один входной код 0111 (число 7), а второй 0101 (число 5), то суммарный код на выходе будет 1100 (12=7+5). Сумма двух двоичных чисел с числом разрядов N может иметь число разрядов (N+1). Например, при суммировании чисел 13 (1101) и 6 (0110) получается число 19 (10011). Поэтому количество выходов сумматора на единицу больше количества разрядов входных кодов. Этот дополнительный (старший) разряд называется выходом переноса.

На схемах сумматоры обозначаются буквами SM. В отечественных сериях код, обозначающий микросхему сумматора, — ИМ.

Сумматоры бывают одноразрядные (для суммирования двух одноразрядных чисел), двухразрядные (суммируют двухразрядные числа) и четырехразрядные (суммируют четырехразрядные числа). Чаще всего применяют именно 4-разрядные сумматоры. На рис. 3.21 показаны для примера 2-разрядный и 4-разрядный сумматоры. Микросхема ИМ6 отличается от ИМ3 только повышенным быстродействием и номерами используемых выводов микросхемы, функция же выполняется та же самая.

Помимо выходных разрядов суммы и выхода переноса сумматоры имеют вход расширения (другое название —вход переноса) С для объединения нескольких сумматоров с целью увеличения разрядности. Если на этот вход приходит единица, то выходная сумма увеличивается на единицу, если же приходит нуль, то выходная сумма не увеличивается. Если используется одна микросхема сумматора, то на ее вход расширения С необходимо подать нуль.

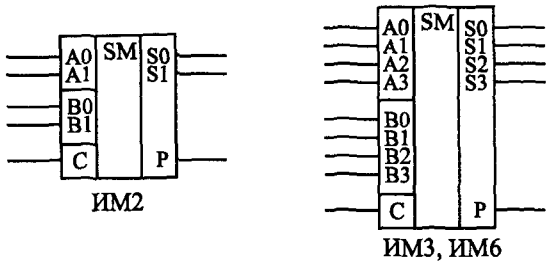


Рис. 3.21. Примеры микросхем сумматоров.

В качестве примера ниже приведена полная таблица истинности 2-разрядного сумматора ИМ2 (табл. 3.5). Как видно из таблицы, выходной 3-разрядный код (P, S1, S0) равен сумме входных 2-разрядных кодов (A1, A0) и (B1, B0), а также сигнала C. Нулевые разряды — младшие, первые разряды — старшие. Полная таблица истинности 4-разрядного сумматора будет чрезмерно большой, поэтому она не приводится. Но суть работы остается точно такой же, как и в случае 2-разрядного сумматора.

Таблица 3.5. Таблица истинности микросхемы 2-разрядного сумматора ИМ2

Входы				Выходы					
				C=0			C=1		
A1	A0	B1	B0	P	S1	S0	P	S1	S0
0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0	1	0
0	0	1	0	0	1	0	0	1	1
0	0	1	1	0	1	1	1	0	0
0	1	0	0	0	0	1	0	1	0
0	1	0	1	0	1	0	0	1	1
0	1	1	0	0	1	1	1	0	0
0	1	1	1	1	0	0	1	0	1
1	0	0	0	0	1	0	0	1	1
1	0	0	1	0	1	1	1	0	0
1	0	1	0	1	0	0	1	0	1
1	0	1	1	1	0	1	1	1	0
1	1	0	0	0	1	1	1	0	0
1	1	0	1	1	0	0	1	0	1
1	1	1	0	1	0	1	1	1	0
1	1	1	1	1	1	0	1	1	1

Сумматоры могут использоваться также для суммирования чисел в отрицательной логике (когда логической единице соответствует электрический нуль, и наоборот логическому нулю соответствует электрическая единица). Но в этом случае входной сигнал переноса С также становится инверсным, поэтому при использовании одной микросхемы сумматора на вход С надо подать электрическую единицу (высокий уровень напряжения). Инверсным становится и выходной сигнал переноса Р, низкий уровень напряжения на нем (электрический нуль) соответствует наличию переноса. То есть получается, что сумматор абсолютно одинаково работает как с положительной, так и с отрицательной логикой.

Рассмотрим пример. Пусть нам надо сложить два числа 5 и 7 в отрицательной логике. Числу 5 в положительной логике соответствует двоичный код 0101, а в отрицательной логике — код 1010. Числу 7 в положительной логике соответствует двоичный код 0111, а в отрицательной — код 1000. При подаче на вход сумматора кодов 1010 (десятичное число 10 в положительной логике) и 1000 (десятичное число 8 в положительной логике) получаем сумму 10+8=18, то есть код 10010 в положительной логике. С учетом входного сигнала переноса С=1 (то есть отсутствие входного переноса в отрицательной логике) выходной код сумматора получится на единицу больше: 18+1=19, то есть 10011. При отрицательной логике это будет соответствовать числу 01100, то есть 12 при отсутствии выходного переноса. В результате получили: 5+7=12.

Сумматор может вычислять не только сумму, но и разность входных кодов, то есть работать вычитателем. Для этого вычитаемое число надо просто поразрядно проинвертировать, а на вход переноса С подать единичный сигнал (рис. 3.22).

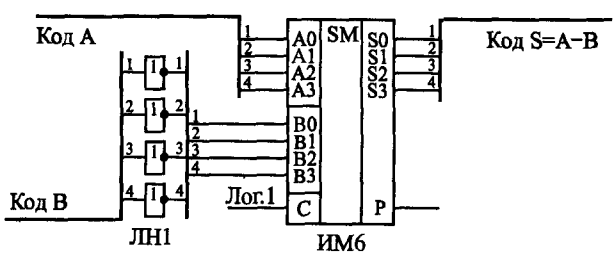


Рис. 3.22. Четырехразрядный вычитатель на сумматоре ИМ6 и инверторах ЛН1.

Например, пусть нам надо вычислить разность между числом 11 (1011) и числом 5 (0101). Инвертируем поразрядно число 5 и получаем 1010, то есть десятичное 10. Сумматор при суммировании 11 и 10 даст 21, то есть двоичное число 10101. Если сигнал С равен 1, то результат будет 10110. Отбрасываем старший разряд (выходной сигнал Р) и получаем разность 0110, то есть 6.

Еще пример. Пусть надо вычислить разность между числом 12 (1100) и числом 9 (1001). Инвертируем поразрядно 9, получаем 0110, то есть десятичное 6. Находим сумму 12 и 6, получаем 18, а с учетом С = 1 получаем 19, то есть двоичное 10011. В четырех младших разрядах имеем 0011, то есть десятичное 3.

Каскадировать сумматоры для увеличения разрядности очень просто. Сигнал с выхода переноса сумматора, обрабатывающего младшие разряды, нужно подать на вход переноса сумматора, обрабатывающего старшие разряды (рис. 3.23). При объединении трех 4-разрядных сумматоров получается 12-разрядный сумматор, имеющий дополнительный 13 разряд (выход переноса Р).

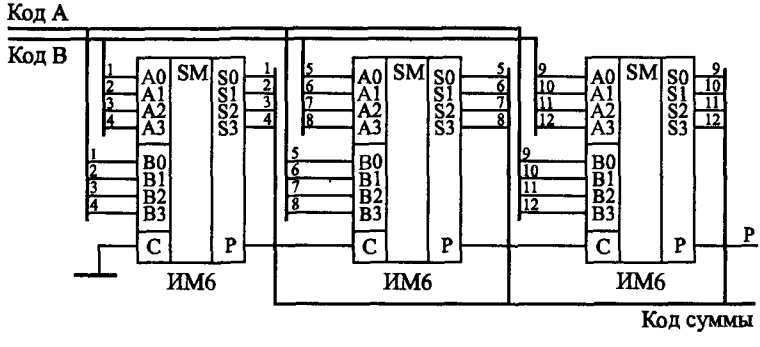


Рис. 3.23. Каскадирование сумматоров ИМ6 для увеличения разрядности.

Неопределенные состояния на выходах сумматора могут возникать при любом изменении любого из входных кодов (рис. 3.24). Выходной код суммы может принимать в течение короткого времени значения, никак не связанные с входными кодами, а на выходе переноса могут появляться короткие паразитные импульсы. Это связано прежде всего с неодновременным изменением разрядов входных кодов. Чтобы избежать влияния этих не-

определенных состояний на дальнейшую схему, необходимо предусматривать синхронизацию или стробирование выходных сигналов. Но для этого надо иметь информацию о моментах изменения входных кодов, которая имеется далеко не всегда.

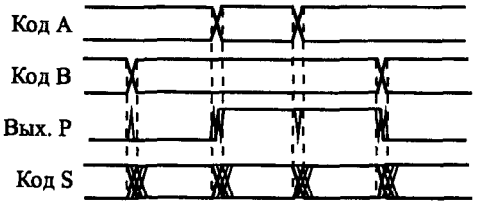


Рис. 3.24. Неопределенные состояния на выходах сумматора при изменении входных кодов.

Задержки сумматора ИМ6 от входов до выходов суммы примерно вдвое превышают задержку логического элемента, а от входов до выхода переноса — примерно в полтора раза. Задержки сумматора ИМ3 больше задержек ИМ6 почти вдвое. Поэтому в схемах, где важно быстродействие, лучше использовать ИМ6. Особенно это важно при каскадировании для увеличения разрядности, так как там задержки отдельных микросхем суммируются. Точные величины задержек можно найти в справочниках.

3.5. Преобразователи кодов

Микросхемы преобразователей кодов (английское Converter) служат для преобразования входных двоичных кодов в выходные двоично-десятичные и наоборот — входных двоично-десятичных кодов в выходные двоичные. Они используются довольно редко, так как применение двоично-десятичных кодов ограничено узкой областью, например, они применяются в схемах многоразрядной десятичной индикации. К тому же при правильной организации схемы часто можно обойтись без преобразования в двоично-десятичный код, например, выбирая счетчики, работающие в двоично-десятичном коде. Преобразование двоично-десятичного кода в двоичный встречается еще реже.

На схемах микросхемы преобразователей обозначаются буквами X/Y. В отечественных сериях преобразователи имеют обозначения ПР.

Кроме того, надо учесть, что любые преобразования параллельных кодов, даже самые экзотические, могут быть легко реализованы на микросхемах постоянной памяти нужного объема. Обычно это намного удобнее, чем брать стандартные микросхемы преобразователей кодов.

В стандартные серии входят две микросхемы преобразователей кодов: ПР6 для преобразования двоично-десятичного кода в двоичный и ПР7 для преобразования двоичного кода в двоично-десятичный (рис. 3.25). Обе микросхемы имеют выходы ОК, поэтому к ним надо присоединять нагрузочные резисторы величиной около 1 кОм, но для удобства в дальнейших схемах эти резисторы не показаны. Обе микросхемы имеют также вход разрешения выхода -ЕО, при нулевом уровне на котором все выходы активны, а при единичном — переходят в состояние единицы. Преобразователь ПР6 имеет дополнительные выходы A, B, C, не участвующие в основном преобразовании.

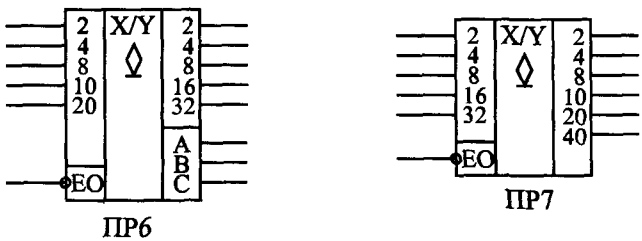


Рис. 3.25. Микросхемы преобразователей кодов.

Таблицы истинности преобразователей просты (табл. 3.6 и 3.7). Например, двоично-десятичный код без младшего разряда на входе ПР6 преобразуется в двоичный код без младшего разряда на выходе ПР6. Младший разряд не участвует в преобразовании, он непосредственно передается со входа на выход. Одна микросхема ПР6 обрабатывает входные коды в диапазоне от 0 (двоично-десятичный код 00 000) до 39 (код 11 1001).

Точно так же двоичный код без младшего разряда на входе ПР7 преобразуется в двоично-десятичный код без младшего разряда на выходе ПР7. Одна микросхема ПР7 может обрабатывать входные коды в диапазоне от 0 (двоичный код 000000) до 63 (код 111111). Младшие разряды входных кодов передаются на выход без обработки в обход микросхемы, так как они оди-

наковые как в двоичном, так и в двоично-десятичном кодах. Простейшие схемы включения одиночных микросхем ПР6 и ПР7 приведены на рис. 3.26.

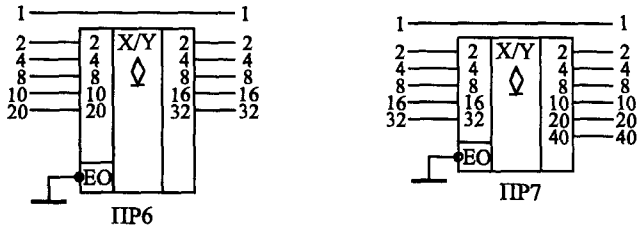


Рис. 3.26. Простейшее включение одиночных преобразователей кода ПР6 и ПР7.

Таблица 3.6. Таблица истинности преобразователя ПР6

Входы						Выходы				
-EO	20	10	8	4	2	32	16	8	4	2
1	X	X	X	X	X	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	0	1	0	0	1	0	0	1	1	0
0	0	1	0	1	0	0	0	1	1	1
0	0	1	0	1	1	0	1	0	0	0
0	0	1	1	0	0	0	1	0	0	1
0	1	0	0	0	0	0	1	0	1	0
0	1	0	0	0	1	0	1	0	1	1
0	1	0	0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	1	1	0	1
0	1	0	1	0	0	0	1	1	1	0
0	1	1	0	0	0	0	1	1	1	1
0	1	1	0	0	1	1	0	0	0	0
0	1	1	0	1	0	1	0	0	0	1
0	1	1	0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0	0	1	1

Таблица 3.7. Таблица истинности преобразователя ПР7

Входы						Выходы					
-ЕО	32	16	8	4	2	40	20	10	8	4	2
1	X	X	X	X	X	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	1	1	0	0	1	0	1	0
0	0	1	0	0	0	0	0	1	0	1	1
0	0	1	0	0	1	0	0	1	1	0	0
0	0	1	0	1	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	0	0	0	1
0	0	1	1	0	0	0	1	0	0	1	0
0	0	1	1	0	1	0	1	0	0	1	1
0	0	1	1	1	0	0	1	0	1	0	0
0	0	1	1	1	1	0	1	1	0	0	0
0	1	0	0	0	0	0	1	1	0	0	1
0	1	0	0	0	1	0	1	1	0	1	0
0	1	0	0	1	0	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1	1	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	0	1
0	1	0	1	1	0	1	0	0	0	1	0
0	1	0	1	1	1	1	0	0	0	1	1
0	1	1	0	0	0	1	0	0	1	0	0
0	1	1	0	0	1	1	0	1	0	0	0
0	1	1	0	1	0	1	0	1	0	0	1
0	1	1	0	1	1	1	0	1	0	1	0
0	1	1	1	0	0	1	0	1	0	1	1
0	1	1	1	0	1	1	0	1	1	0	0
0	1	1	1	1	0	1	1	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	1

Каскадировать преобразователи ПР6 и ПР7 для увеличения разрядности также несложно. Для преобразования двоично-десятичных кодов от 0 до 99 достаточно двух микросхем ПР6 (рис. 3.27), а для преобразования двоичных кодов от 0 до 255

требуется три микросхемы ПР7 (рис. 3.28). Если надо преобразовывать двоично-десятичные коды до 999, то понадобится 6 микросхем ПР6, а для преобразования двоичных кодов до 511 потребуется 4 микросхемы ПР7. На всех выходах микросхем необходимо включать нагрузочные резисторы.

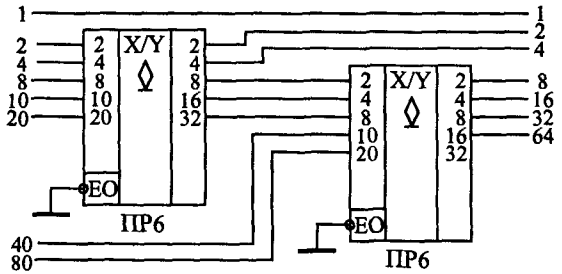


Рис. 3.27. Преобразователь двоично-десятичного кода от 0 до 99 в двоичный код.

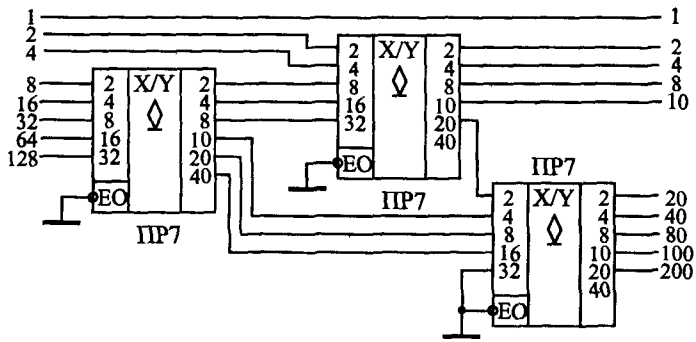


Рис. 3.28. Преобразователь двоичного кода от 0 до 255 в двоично-десятичный код.

Наличие дополнительных выходов A, B, C у микросхемы ПР6 позволяет преобразовывать двоично-десятичный код от 0 до 9 в код дополнения до 9 или до 10 (рис. 3.29). То есть сумма входного и выходного кода в этом случае равна соответственно 9 или 10. Например, при входном коде 6 на выходе схемы *а* будет код 3, а на выходе схемы *б* — код 4. В схеме *б* при входном коде 0 на выходе также формируется код 0. Как и все остальные выходы микросхемы ПР6, выходы A, B, C имеют тип ОК, поэтому к ним необходимо

присоединять нагрузочные резисторы, для удобства не показанные на схеме. Такие схемы «дополнителей» применяются редко, поэтому о них упоминают не во всех справочниках и учебниках, но иногда подобные функции бывают довольно удобны.

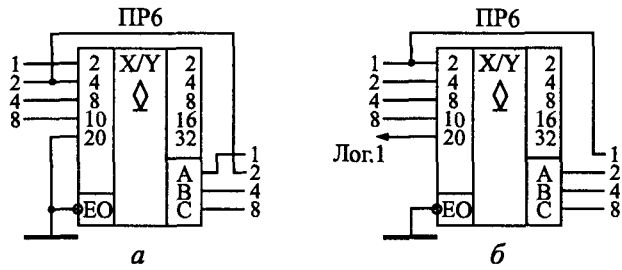


Рис. 3.29. Преобразователи входного кода в дополнение до 9 (*а*) и в дополнение до 10 (*б*).

Задержки преобразователей кодов примерно вдвое превосходят задержки логических элементов. Точные величины задержек можно найти в справочниках.

3.6. Одновибраторы и генераторы

Одновибраторы и генераторы вообще-то нельзя отнести к комбинационным микросхемам. Они занимают промежуточное положение между комбинационными микросхемами и микросхемами с внутренней памятью. Их выходные сигналы не определяются однозначно входными сигналами, как у комбинационных микросхем. Но в то же время они и не хранят информацию длительное время.

Одновибраторы (ждущие мультивибраторы, английское название Monostable Multivibrator) представляют собой микросхемы, которые в ответ на входной сигнал (логический уровень или фронт) формируют выходной импульс заданной длительности. Длительность этого импульса определяется внешними времязадающими резисторами и конденсаторами. То есть можно считать, что у одновибраторов есть внутренняя память, но эта память хранит информацию о входном сигнале строго заданное время, а потом информация исчезает. На схемах одновибраторы обозначаются буквами G1.

В стандартные серии микросхем входят одновибраторы двух основных типов (отечественное обозначение функции микросхемы — АГ):

- Одновибраторы без перезапуска (АГ1 — одиночный одновибратор, АГ4 — два одновибратора в одном корпусе);
- Одновибраторы с перезапуском (АГ3 — два одновибратора в одном корпусе).

Разница между этими двумя типами одновибраторов иллюстрируется рис. 3.30. Одновибратор без перезапуска не реагирует на входной сигнал до окончания своего выходного импульса. Одновибратор с перезапуском начинает отсчет нового времени выдержки T с каждым новым входным сигналом независимо от того, закончилось ли предыдущее время выдержки. В случае когда период следования входных сигналов меньше времени выдержки T, выходной импульс одновибратора с перезапуском не прерывается. Если период следования входных запускающих импульсов больше времени выдержки одновибратора T, то оба типа одновибраторов работают одинаково.

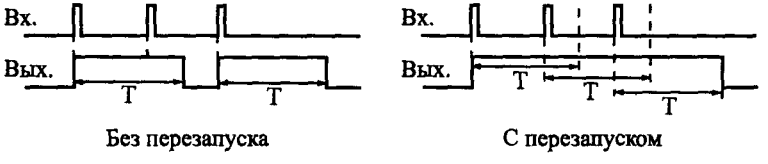


Рис. 3.30. Принцип работы одновибраторов без перезапуска и с перезапуском.

На рис. 3.31 приведены обозначения микросхем одновибраторов стандартных серий. Микросхемы АГ3 и АГ4 отличаются друг от друга только тем, что АГ3 работает с перезапуском, а АГ4 — без перезапуска.

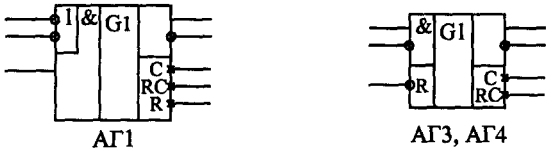


Рис. 3.31. Микросхемы одновибраторов.

Микросхемы имеют входы запуска, объединенные по И и ИЛИ, прямые и инверсные выходы, а также выводы для подключения внешних времязадающих цепей (резисторов и конденсаторов). Запускаются все одновибраторы по фронту результирующего входного сигнала. Использованная логика объединения входов микросхем позволяет осуществить их запуск как по положительному, так и по отрицательному фронту входного сигнала (рис. 3.32 и 3.33).

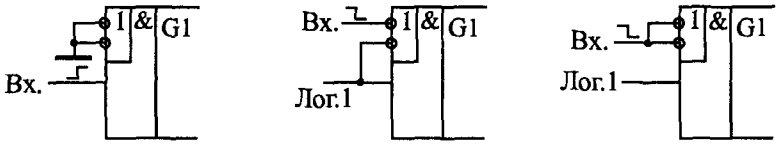


Рис. 3.32. Варианты запуска одновибратора АГ1.

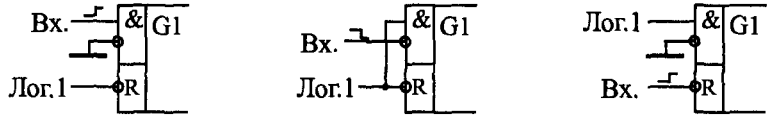


Рис. 3.33. Варианты запуска одновибраторов АГ3 и АГ4.

На неиспользуемые входы при этом надо подавать сигналы логического нуля или логической единицы. Можно также использовать остающиеся входы для разрешения или запрещения входного запускающего сигнала.

Одновибраторы АГ3 и АГ4 имеют также дополнительный вход сброса -R, логический нуль на котором не только запрещает выработку выходного сигнала, но и прекращает его действие. Вход -R можно также использовать для запуска одновибратора.

Таблицы истинности одновибраторов приведены ниже (табл. 3.8 и 3.9). В этих таблицах инверсные входные сигналы обозначены -A, -A1, -A2, прямые входные сигналы обозначены B, а прямой и инверсный выходные сигналы — соответственно Q и -Q.

Стандартное включение одновибраторов предполагает подключение внешнего резистора и внешнего конденсатора (рис. 3.34).

Таблица 3.8. Таблица истинности одновибратора АГ1

Входы			Выходы	
-A1	-A2	B	Q	-Q
0	X	1	0	1
X	0	1	0	1
X	X	0	0	1
1	1	X	0	1
1	↓	1	⎍	⎍
↓	1	1	⎍	⎍
↓	↓	1	⎍	⎍
0	X	↑	⎍	⎍
X	0	↑	⎍	⎍

Таблица 3.9. Таблица истинности одновибраторов АГ3 и АГ4

Входы			Выходы	
-R	-A	B	Q	-Q
0	X	X	0	1
X	1	0	0	1
X	X	0	0	1
1	0	↑	⎍	⎍
1	↓	1	⎍	⎍
↑	0	1	⎍	⎍

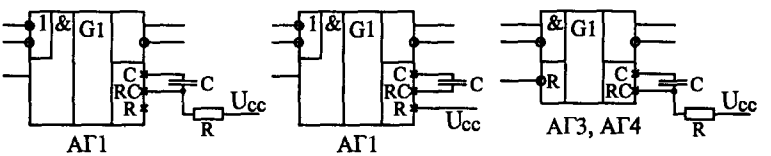


Рис. 3.34. Стандартные схемы включения одновибраторов.

Для одновибратора АГ1 длительность выходного импульса можно оценить по формуле: T = 0,7RC. Эта формула работает при величине сопротивления резистора в пределах от 1,5 кОм до 43 кОм. Емкость конденсатора может быть любой. Внутри микросхемы имеется внутренний резистор сопротивлением около 2 кОм, подключенный к выводу R, поэтому можно включать одновибратор без внешнего резистора, подключая вывод R к шине питания. Повторный запуск одновибратора невозможен сразу после оконча-

ния выходного импульса, до повторного запуска обязательно должен пройти интервал t = C (если емкость измеряется в нанофарадах, то временной интервал получается в микросекундах).

Для одновибраторов АГ3 и АГ4 длительность импульса можно оценить по формуле: T = 0,32C(R + 0,7), где сопротивление резистора измеряется в килоомах. Сопротивление резистора может находиться в пределах от 5,1 кОм до 51 кОм, емкость конденсатора — любая. Перезапуск одновибратора возможен только в том случае, когда интервал между входными запускающими импульсами больше 0,224C (если емкость измеряется в нанофарадах, то временной интервал — в микросекундах).

Наиболее распространенные применения одновибраторов следующие (рис. 3.35):

а) увеличение длительности входного импульса;
б) уменьшение длительности входного импульса;
в) деление частоты входного сигнала в заданное число раз;
г) формирование сигнала огибающей последовательности входных импульсов.

Для увеличения или уменьшения длительности входного сигнала (*а* и *б*) надо всего лишь подобрать сопротивление резистора и емкость конденсатора, исходя из требуемой длительности выходного сигнала. В этом случае можно использовать одновибратор любого типа: как с перезапуском, так и без перезапуска.

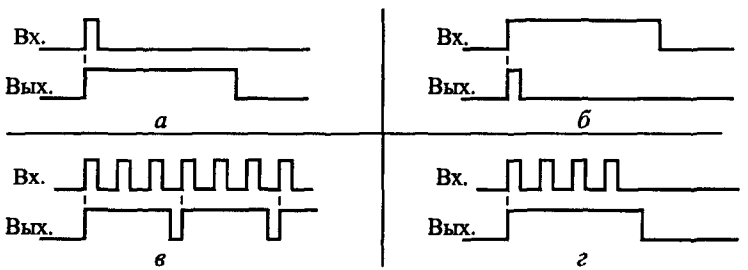


Рис. 3.35. Стандартные применения одновибраторов.

Для деления частоты входных импульсов в заданное число раз (*в*) используется только одновибратор без перезапуска. При этом надо выбрать такую длительность выходного сигнала, что-

бы одновибратор пропускал нужное количество входных импульсов. Например, если требуется разделить на 3 частоту входных импульсов f, то длительность выходного сигнала одновибратора надо выбрать в пределах от 2/f до 3/f. При этом одновибратор будет пропускать два входных импульса из каждых трех.

Для формирования огибающей входного сигнала (*г*) используется только одновибратор с перезапуском. При этом длительность его выходного импульса должна быть выбрана такой, чтобы каждый следующий входной сигнал перезапускал одновибратор. Если частота входного сигнала равна f, то длительность выходного сигнала одновибратора должна быть не меньше, чем 1/f.

Еще одно важное применение одновибратора состоит в подавлении дребезга контактов кнопки. Одновибратор с большим временем выдержки (порядка нескольких десятых долей секунды) надежно подавляет паразитные импульсы, возникающие из-за дребезга контактов, и формирует идеальные импульсы на любое нажатие кнопки (рис. 3.36).

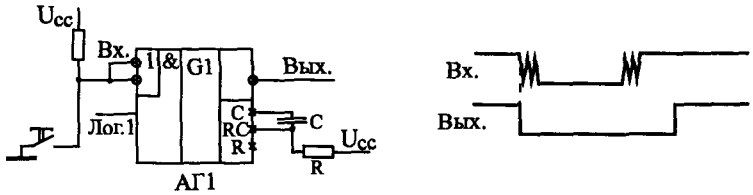


Рис. 3.36. Использование одновибратора для подавления дребезга контактов кнопки.

Для этого можно использовать как одновибратор с перезапуском, так и одновибратор без перезапуска (на рисунке). Можно также подобрать время выдержки так, что одновибратор будет давать один импульс по нажатию кнопки, а другой импульс — по отпусканию кнопки. Иногда это бывает удобнее.

Одновибраторы можно также использовать для построения генераторов (мультивибраторов) прямоугольных импульсов с различными значениями длительности импульсов и паузы между ними. При этом два одновибратора замыкаются в кольцо так, что каждый из них запускает другой после окончания своего выходного импульса (рис. 3.37). Один одновибратор формирует дли-

тельность импульса, а другой определяет паузу между импульсами. Изменяя номиналы резисторов и конденсаторов, можно получить нужные соотношения длительностей импульса и паузы.

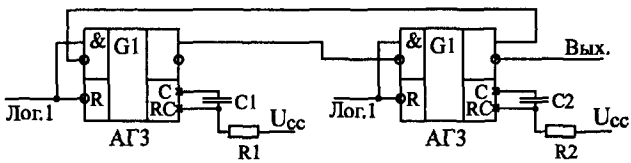


Рис. 3.37. Генератор импульсов на двух одновибраторах.

Таким образом, одновибраторы довольно легко позволяют решать самые разные задачи. Однако, применяя одновибраторы, надо всегда помнить, что длительность их выходных импульсов нельзя задать очень точно, ведь одновибратор имеет аналоговые цепи. На длительность выходного импульса одновибратора влияют разбросы номиналов резисторов и конденсаторов, температура окружающей среды, старение элементов, помехи по цепям питания, другие факторы. Поэтому применение одновибраторов надо по возможности ограничивать только теми случаями, когда время выдержки надо задавать с не слишком высокой точностью (погрешность не менее 20–30%).

Любую функцию одновибратора может выполнить синхронное тактируемое устройство (на основе кварцевого генератора, триггеров, регистров, счетчиков), причем выполнить гораздо точнее и надежнее. И ему не нужно никаких дополнительных времязадающих элементов (резисторов и конденсаторов). Количество одновибраторов, использованных в схеме, как правило, обратно пропорционально уровню мастерства разработчика этой схемы.

Задержки запуска одновибраторов примерно в два-три раза превосходят задержку логического элемента. Точные величины задержек можно найти в справочниках.

Помимо одновибраторов в стандартные серии включены также специализированные **генераторы** (мультивибраторы, английское Multivibrator). Обозначаются они на схемах буквой G. В отечественных сериях этот тип микросхемы кодируется буквами ГГ. Например, микросхема ГГ1 представляет собой два генератора в одном корпусе.

Микросхемы генераторов используют довольно редко, чаще применяют генераторы на инверторах или на триггерах Шмитта, описанные во второй главе. Однако в некоторых случаях генераторы ГГ1 не могут быть заменены ничем. Дело в том, что они допускают изменение частоты выходных импульсов с помощью уровней двух входных управляющих напряжений. Поэтому они называются также «генераторы, управляемые напряжением» или ГУН. Эффект изменения частоты можно использовать, например, в системах автоподстройки частоты (АПЧ) или в устройствах с частотной модуляцией (ЧМ).

Стандартная схема включения генератора ГГ1 приведена на рис. 3.38. Генератор имеет выводы для подключения внешнего конденсатора C1 и C2, к которым можно также подключать кварцевый резонатор, но при этом уже нельзя управлять частотой. Имеется два входа управления частотой U1 и U2, а также вход разрешения -E, при подаче на который логической единицы генерация прекращается и на выходе F устанавливается единица.

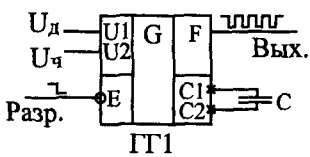


Рис. 3.38. Схема включения генератора ГГ1.

Один из входов управления (U1) обычно называется диапазонным или $U_д$, а другой (U2) — входом управления частоты или $U_ч$. При увеличении напряжения $U_ч$ частота увеличивается, при увеличении напряжения на входе $U_д$ частота уменьшается. Рекомендуемый диапазон изменения напряжения $U_д$ составляет от 2 до 4,5 В, а диапазон изменения $U_ч$ — от 0 до 5 В. В зависимости от напряжения $U_д$ меняется диапазон изменения частоты из-за изменения напряжения $U_ч$. Например, при $U_д = 2$ В и изменении $U_ч$ от 1 до 5 В частота изменяется примерно на 15%, а при $U_д = 4$ В — приблизительно в 4 раза.

Частота выходного сигнала ГГ1 определяется также внешним конденсатором, например, при $U_д = U_ч = 2$ В и при C = 1 мкФ частота будет около 100 Гц, а при C = 100 пФ — порядка

10 МГц. Максимально возможное значение частоты генератора составляет около 80 МГц. В справочниках приводятся графики зависимости частоты выходного сигнала ГГ1 от уровней управляющих напряжений и номинала внешнего конденсатора. Однако точно определить значение частоты по этим графикам невозможно, в любом случае требуется подстройка. К тому же наличие в схеме аналоговых узлов делает генератор ГГ1 чувствительным к разбросу номиналов конденсаторов, изменению температуры окружающей среды, старению элементов, помехам по цепям питания и к другим факторам. Именно поэтому использование этих генераторов крайне ограничено.

И последнее. В микросхеме ГГ1 существует взаимное влияние двух генераторов друг на друга, хотя в ней и приняты меры по снижению этого влияния. Поэтому не рекомендуется использовать одновременно два генератора одной микросхемы в режиме генерации частоты, управляемой напряжением.

Глава 4
ПРИМЕНЕНИЕ ТРИГГЕРОВ И РЕГИСТРОВ

Триггеры и регистры являются простейшими представителями цифровых микросхем, имеющих внутреннюю память. Если выходные сигналы логических элементов и комбинационных микросхем однозначно определяются их текущими входными сигналами, то выходные сигналы микросхем с внутренней памятью зависят также еще и от того, какие входные сигналы и в какой последовательности поступали на них в прошлом. То есть они помнят предысторию поведения схемы. Именно поэтому их применение позволяет строить гораздо более сложные и интеллектуальные цифровые устройства, чем в случае простейших микросхем без памяти. Микросхемы с внутренней памятью называются еще последовательными или последовательностными в отличие от комбинационных микросхем.

Триггеры и регистры сохраняют свою память только до тех пор, пока на них подается напряжение питания. То есть их память относится к типу *оперативной* памяти (в отличие от *постоянной* памяти и *перепрограммируемой постоянной* памяти, которым отключение питания не мешает сохранять информацию). После выключения питания и его последующего включения триггеры и регистры переходят в случайное состояние, то есть их выходные сигналы могут устанавливаться как к уровню логической единицы, так и к уровню логического нуля. Это необходимо учитывать при проектировании схем.

Большим преимуществом триггеров и регистров перед другими типами микросхем с памятью является их максимально высокое быстродействие (то есть минимальные времена задержек срабатывания и максимально высокая допустимая рабочая частота). Именно поэтому триггеры и регистры иногда называют также *сверхоперативной* памятью. Однако недостатком триггеров и регистров является то, что объем их внутренней памяти очень мал, они могут хранить только отдельные сигналы, биты (триггеры) или отдельные коды, байты, слова (регистры).

Триггер можно рассматривать как одноразрядную ячейку памяти, а регистр — как многоразрядную ячейку памяти, состоящую из несколько триггеров, соединенных параллельно (обычный, параллельный регистр) или последовательно (сдвиговый регистр или, что то же самое, регистр сдвига).

4.1. Триггеры

4.1.1. Принцип работы и разновидности триггеров

В основе любого триггера (английское — Trigger или Flip-Flop) лежит схема из двух логических элементов, которые охвачены положительными обратными связями (то есть сигналы с выходов подаются на входы). В результате подобного включения схема может находиться в одном из двух устойчивых состояний, причем находиться сколь угодно долго, пока на нее подано напряжение питания.

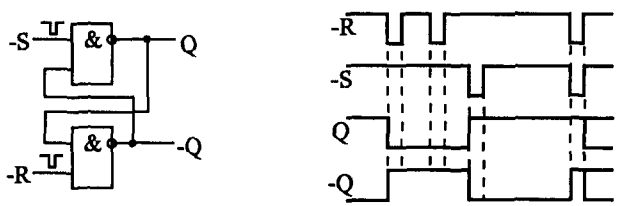


Рис. 4.1. Схема триггерной ячейки.

Пример такой схемы (так называемой триггерной ячейки) на двух двухвходовых элементах И-НЕ представлен на рис. 4.1. У схемы есть два инверсных входа: -R — сброс (от английского Reset), и -S — установка (от английского Set), а также два выхода: прямой выход Q и инверсный выход -Q.

Для правильной работы схемы отрицательные импульсы должны поступать на ее входы не одновременно. Приход импульса на вход -R переводит выход -Q в состояние единицы, а так как сигнал -S при этом единичный, выход Q становится нулевым. Этот же сигнал Q поступает по цепи обратной связи на вход нижнего элемента. Поэтому даже после окончания импульса на входе -R состояние схемы не изменяется (на Q остается нуль, на -Q остается единица). Точно так же при приходе им-

пульса на вход -S выход переходит Q в единицу, а выход -Q — в нуль. Оба этих устойчивых состояния триггерной ячейки могут сохраняться сколь угодно долго, пока не придет очередной входной импульс, то есть схема обладает памятью.

Если оба входных импульса придут строго одновременно, то в момент действия этих импульсов на обоих выходах будут единичные сигналы, а после окончания входных импульсов выходы случайным образом попадут в одно из двух устойчивых состояний. Точно так же случайным образом будет выбрано одно из двух устойчивых состояний триггерной ячейки при включении питания. Временная диаграмма работы триггерной ячейки показана на рисунке.

Таблица истинности схемы приведена ниже (табл. 4.1).

Таблица 4.1. Таблица истинности триггерной ячейки

Входы		Выходы	
-R	-S	Q	-Q
0	1	0	1
1	0	1	0
1	1	Без изменения	
0	0	Не определено	

В стандартные серии цифровых микросхем входит несколько типов микросхем триггеров, различающихся методами управления, а также входными и выходными сигналами. На схемах триггеры обозначаются буквой Т. В отечественных сериях микросхем триггеры имеют наименование ТВ, ТМ и ТР в зависимости от типа триггера. Наиболее распространены три типа триггеров (рис. 4.2):

- RS-триггер (обозначается ТР) — самый простой, но редко используемый триггер (*а*);
- JK-триггер (обозначается ТВ) имеет самое сложное управление, также используется довольно редко (*б*);
- D-триггер (обозначается ТМ) — наиболее распространенный тип триггера (*в*).

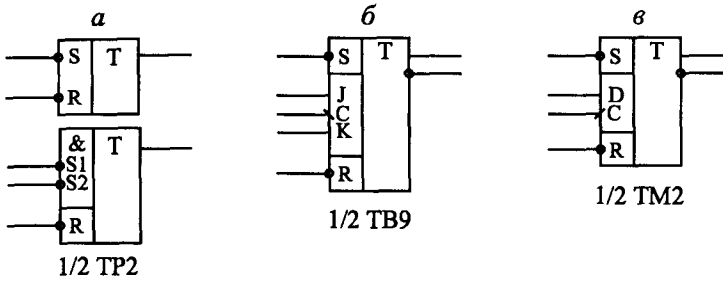


Рис. 4.2. Триггеры трех основных типов.

Примером RS-триггера является микросхема ТР2, в одном корпусе которой находятся четыре RS-триггера. Два триггера имеют по одному входу -R и -S, а два других триггера — по одному входу -R и по два входа -S1 и -S2, объединенных по функции И. Все триггеры имеют только по одному прямому выходу. RS-триггер практически ничем не отличается по своим функциям от триггерной ячейки, рассмотренной ранее (см. рис. 4.1). Отрицательный импульс на входе -R перебрасывает выход в нуль, а отрицательный импульс на входе -S (или на любом из входов -S1 и -S2) перебрасывает выход в единицу. Одновременные сигналы на входах -R и -S переводят выход в единицу, а после окончания импульсов триггер попадает случайным образом в одно из своих устойчивых состояний. Таблица истинности триггера ТР2 с двумя входами установки -S1 и -S2 представлена ниже (табл. 4.2).

Таблица 4.2. Таблица истинности RS-триггера ТР2

Входы			Выход
-S1	-S2	-R	Q
1	1	1	Без изменения
X	0	1	1
0	X	1	1
1	1	0	0
X	0	0	Не определен
0	X	0	Не определен

JK-триггер значительно сложнее по своей структуре, чем RS-триггер. Он относится к так называемым тактируемым триггерам, то есть он срабатывает по фронту тактового сигнала. Примером может служить показанная на рис. 4.2 микросхема ТВ9, имеющая в одном корпусе два JK-триггера со входами сброса и установки -R и -S. Входы -R и -S работают точно так же, как и в RS-триггере, то есть отрицательный импульс на входе -R устанавливает прямой выход в нуль, а инверсный — в единицу, а отрицательный импульс на входе -S устанавливает прямой выход в единицу, а инверсный — в нуль.

Однако состояние триггера может быть изменено не только этими сигналами, но и сигналами на двух информационных входах J и K и синхросигналом C. Переключение триггера в этом случае происходит по отрицательному фронту сигнала C (по переходу из единицы в нуль) в зависимости от состояний сигналов J и K. При единице на входе J и нуле на входе K по фронту сигнала C прямой выход устанавливается в единицу (обратный — в нуль). При нуле на входе J и единице на входе K по фронту сигнала C прямой выход устанавливается в нуль (обратный — в единицу). При единичных уровнях на обоих входах J и K по фронту сигнала C триггер меняет состояние своих выходов на противоположные (это называется счетным режимом).

Таблица 4.3. Таблица истинности JK-триггера ТВ9

Входы					Выходы	
-S	-R	C	J	K	Q	-Q
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	Не определено	
1	1	1—>0	1	0	1	0
1	1	1—>0	0	1	0	1
1	1	1—>0	0	0	Не изменяется	
1	1	1—>0	1	1	Меняется на противоположное	
1	1	1	X	X	Не изменяется	
1	1	0	X	X	Не изменяется	
1	1	0—>1	X	X	Не изменяется	

Все это видно из таблицы истинности триггера ТВ9 (табл. 4.3) и временной диаграммы его работы (рис. 4.3).

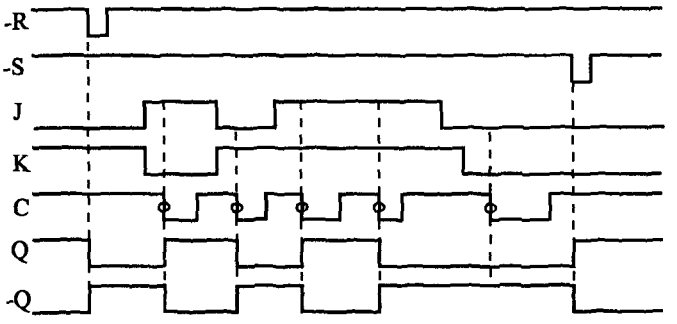


Рис. 4.3. Временная диаграмма работы JK-триггера ТВ9.

Наконец, самый распространенный D-триггер занимает, можно сказать, промежуточное положение между RS-триггером и JK-триггером. Помимо общих для всех триггеров входов установки и сброса -S и -R он имеет один информационный вход D (вход данных) и один тактовый вход C. Примером может служить показанная на рис. 4.2 микросхема ТМ2, содержащая в одном корпусе два D-триггера с прямыми и инверсными выходами.

Таблица 4.4. Таблица истинности D-триггера ТМ2

Входы				Выходы	
-S	-R	C	D	Q	-Q
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	Не определено	
1	1	0—>1	1	1	0
1	1	0—>1	0	0	1
1	1	0	X	Не меняется	
1	1	1	X	Не меняется	
1	1	1—>0	X	Не меняется	

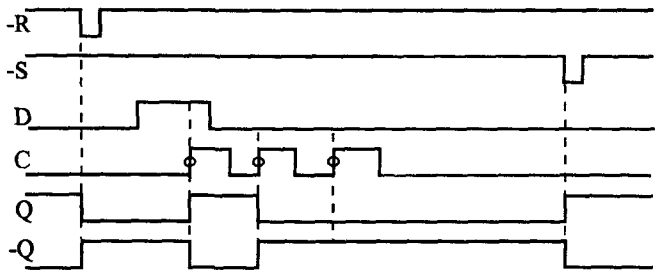


Рис. 4.4. Временная диаграмма работы D-триггера ТМ2.

Тактируется триггер (то есть изменяет свое состояние) по положительному фронту сигнала С (по его переходу из нуля в единицу) в зависимости от состояния входа данных D. Если на входе D единичный сигнал, то по фронту сигнала С прямой выход триггера устанавливается в единицу (инверсный — в нуль). Если же на входе D нулевой сигнал, то по фронту сигнала С прямой выход триггера устанавливается в нуль (инверсный — в единицу).

Таблица истинности триггера ТМ2 представлена выше (табл. 4.4), а временная диаграмма работы показана на рис. 4.4.

Остановимся на работе D-триггера чуть подробнее, так как он наиболее часто используется. При этом многие замечания, высказываемые здесь относительно D-триггера, будут верны и для других типов триггеров.

Прежде всего отметим, что все приведенные временные диаграммы относятся к первому уровню представления, к уровню логической модели. Конечно же, в реальности все триггеры имеют временные задержки установки выходных сигналов, а также предъявляют определенные временные требования к входным сигналам, при нарушении которых любой триггер будет работать неустойчиво или же не будет работать вообще. Это учитывается на втором уровне представления (в модели с временными задержками).

Например, как уже отмечалось, входные сигналы -R и -S не должны приходить одновременно, иначе состояние триггера будет неопределенным. Длительность сигналов -R и -S также не должна быть слишком малой, иначе триггер может на них не среагировать. Сигнал -R должен начинаться с оп-

ределенной задержкой после окончания сигнала -S и наоборот. В первом приближении можно считать, что минимально допустимые временные интервалы между входными сигналами должны равняться 1—2 задержкам логического элемента соответствующей серии.

Точно так же не должна быть слишком малой длительность тактового сигнала C (как положительного импульса, так и отрицательного импульса), иначе триггер может переключаться неустойчиво. Это требование универсально для всех микросхем, срабатывающих по фронту входного сигнала. Принципиально важна и величина временного сдвига (задержки) между установлением сигнала D и рабочим (положительным) фронтом сигнала C. Этот сдвиг также не должен быть слишком малым. Не должен быть чрезмерно малым и сдвиг между окончанием сигналов -R и -S и рабочим фронтом сигнала C. Повышенные требования предъявляются также к длительности фронта тактового сигнала C, которая не должна быть слишком большой. Это требование также универсально для всех микросхем, срабатывающих по фронту входного сигнала.

Одним словом, чем сложнее микросхема, тем важнее для нее становятся ограничения второго уровня представления, тем выше требования к разработчику по учету временных задержек и длительностей сигналов. Правда, требования эти не слишком разнообразны и не слишком жестки, поэтому, раз и навсегда усвоив их, можно проектировать любые схемы без грубых ошибок. Самое главное, что надо запомнить, состоит в следующем: цифровые схемы не любят слишком коротких входных сигналов и слишком малых задержек между входными сигналами, функционально связанными между собой. Ориентир здесь очень простой — величина задержки логического элемента данной серии. Поэтому для более быстрых серий ограничения будут менее жесткими, а для более медленных серий — более жесткими.

Несколько слов о величинах задержек микросхем триггеров.

Несмотря на свою достаточно сложную внутреннюю структуру микросхемы триггеров являются одними из самых быстрых. Задержка срабатывания триггера обычно не превышает 1,5–2 задержек логического элемента. Причем задержки по входам -R и -S чуть меньше, чем по тактовому входу C. В некоторых сериях JK-триггеры несколько быстрее, чем D-триггеры, в других — наоборот. Важный параметр триггера — максималь-

ная частота тактового сигнала C. Для ее приблизительной оценки можно придерживаться следующего простого правила: период тактового сигнала C не должен быть меньше величины задержки переключения триггера по входу C.

4.1.2. Основные схемы включения триггеров

Говоря об областях применения триггеров, мы будем рассматривать исключительно D-триггеры, так как в большинстве случаев RS- и JK-триггеры могут быть заменены D-триггерами без ухудшения каких бы то ни было параметров схемы. Примеры такой замены показаны на рис. 4.5.

RS-триггер получается из D-триггера, если в D-триггере не использовать входы C и D, например, соединить их с общим проводом (*а*).

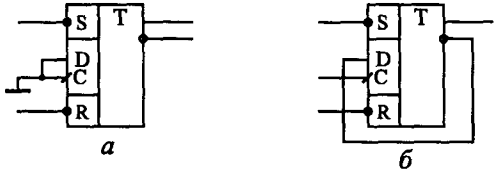


Рис. 4.5. Включение D-триггера для замены RS-триггера (а) и JK-триггера в счетном режиме (б).

Сложнее обстоит дело с заменой JK-триггера, в котором предусмотрено больше возможностей, чем в D-триггере. Однако обычно два информационных входа JK-триггера не так уж и нужны. А что касается счетного режима, в котором, пожалуй, наиболее часто работают JK-триггеры, то он легко реализуется на D-триггере в результате объединения информационного входа D с инверсным выходом (*б*). При этом по каждому положительному фронту сигнала C триггер будет менять свое состояние на противоположное: нуль на прямом выходе будет сменяться единицей и наоборот. То есть частота входного сигнала триггера будет меньше частоты входного тактового сигнала C в два раза.

Отметим также, что для реализации счетного режима чаще всего используются не триггеры, а счетчики, которые будут рассмотрены в следующей главе.

Особенности триггеров обуславливают наиболее широкий диапазон схем их включения для решения самых разных задач.

Например, с помощью триггера (любого типа) очень просто и эффективно решается задача устранения влияния дребезга контактов механических переключателей (рис. 4.6). Правда в данном случае необходим тумблер (или кнопка) с тремя выводами, один из которых попеременно подключается к двум другим. При этом первый же отрицательный импульс на входе -R перебрасывает триггер в состояние нуля, а первый же отрицательный импульс на входе -S — в состояние единицы. Последующие же импульсы на обоих этих входах, вызванные дребезгом контактов, уже никак не влияют на триггер. Нижнее (по рисунку) положение выключателя соответствует нулю на выходе триггера, а верхнее — единице.

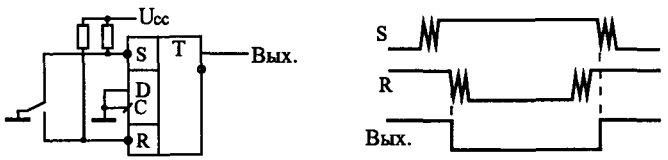


Рис. 4.6. Подавление дребезга контактов выключателя с помощью триггера.

Основное применение триггеры находят в тех случаях, когда надо сформировать сигнал, длительность которого соответствует длительности какой-то выполняемой операции, какого-то продолжительного процесса в схеме. Выходной сигнал триггера при этом может разрешать этот самый процесс, а может информировать остальные узлы устройства о том, что процесс идет (или, как говорят, служить *флагом* процесса). Например, в схеме на рис. 4.7 в начале процесса (операции) по сигналу Старт триггер перебрасывается в единицу, а в конце процесса (операции) по сигналу Стоп триггер перебрасывается обратно в нуль.

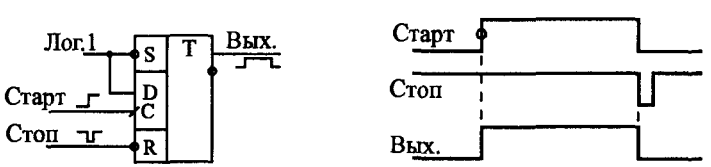


Рис. 4.7. Использование триггера в качестве флага процесса.

Для сигналов Старт и Стоп можно, конечно, использовать входы триггера -R и -S. Однако более правильным и универсальным решением будет выбор пары входов C и -R или C и -S, что предотвратит неоднозначность поведения триггера при одновременном приходе сигналов Старт и Стоп. Если используются входы C и -R, то на вход D надо подать единицу, а если применяются входы C и -S, то на вход D надо подать нуль. Такое решение удобно еще и тем, что в качестве одного из сигналов Старт и Стоп может выступать не уровень, а фронт. Именно этот фронт (в нужной полярности) и надо подать в этом случае на тактовый вход триггера C.

Вторая важнейшая область применения триггеров — это синхронизация сигналов.

Например, триггер позволяет наиболее просто избавиться от паразитных коротких импульсов на выходах комбинационных схем, возникающих при почти одновременном изменении нескольких входных сигналов (рис. 4.8). Для синхронизации в данном случае необходимо иметь синхросигнал (синхропереход), сопровождающий входные информационные сигналы (входной код) и задержанный относительно момента изменения этих сигналов на время $t_з$, большее задержки комбинационной схемы. При подаче этого синхроимпульса на вход C триггера, а выходного сигнала комбинационной микросхемы (Вых.1) на вход D триггера на выходе триггера получаем сигнал (Вых.2), полностью свободный от паразитных импульсов.

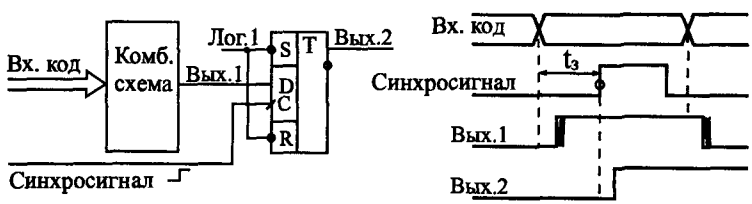


Рис. 4.8. Синхронизация с помощью триггера.

Более того, в случае, когда входной код комбинационной схемы изменяется регулярно, периодически, фронт синхросигнала может даже совпадать с моментом изменения входного кода (рис. 4.9).

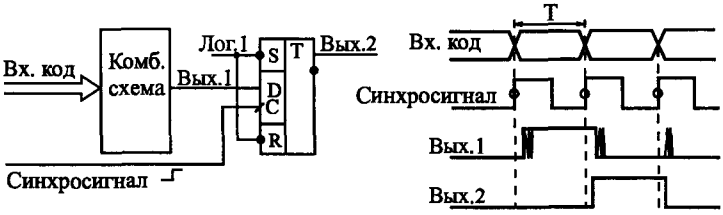


Рис. 4.9. Синхронизация с помощью триггера при периодическом изменении входного кода.

При этом за счет конечной величины задержки комбинационной схемы сигнал на вход C триггера будет поступать раньше, чем начнет изменяться сигнал на его входе D. Поэтому паразитные импульсы в триггер не запишутся. Правда, в данном случае сигнал на выходе триггера (Вых.2) будет задержан на период следования входных кодов T (или, что то же самое, на период синхросигнала) относительно выходного сигнала комбинационной схемы (Вых.1).

При проектировании цифровых схем, работающих по тактам единого тактового генератора, часто возникает необходимость синхронизовать с работой схемы какой-то внешний сигнал. То есть требуется обеспечить, чтобы этот сигнал (асинхронный по отношению ко всей остальной схеме) изменялся по тактам тактового генератора, как и все остальные сигналы схемы (стал бы синхронным всей остальной схеме). В этом тоже может помочь триггер.

Рассмотрим самый простой пример. Пусть необходимо с помощью внешнего сигнала разрешать и запрещать прохождение сигнала непрерывно работающего тактового генератора. В случае обычного RC-генератора эта задача иногда может быть решена довольно просто путем запуска и остановки генератора (см. рис. 2.39). Однако далеко не всегда допускается останавливать работу тактового генератора, от которого работает вся схема. В случае же кварцевого генератора его остановка и запуск вообще никогда не используются, так как такой генератор начинает работать после разрешения с задержкой, равной нескольким периодам тактовой частоты, причем количество этих периодов не постоянно. Поэтому будем считать, что тактовый генератор работает постоянно, а по внешнему управляющему сигналу мы будем разрешать или запрещать прохождение его выходных импульсов (рис. 4.10).

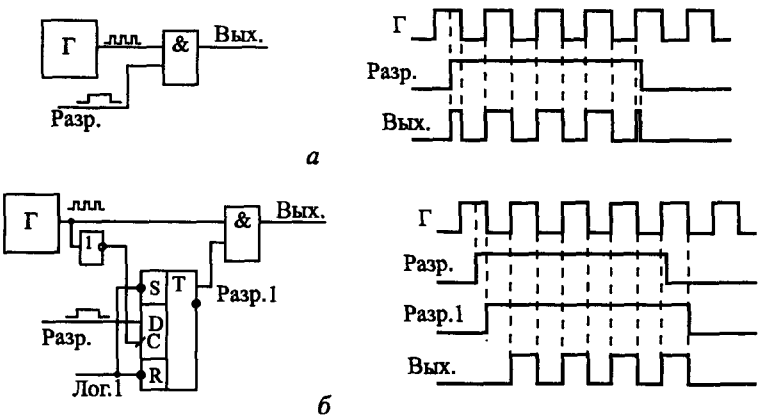


Рис. 4.10. Синхронизация сигнала разрешения.

В простейшем случае (*а*) для пропускания и запрещения импульсов тактового генератора Г используется логический элемент 2И. При этом вполне возможна ситуация прохождения на выход схемы импульсов неполной длительности или даже предельно коротких, нестабильно появляющихся импульсов, которые могут вносить неопределенность в работу остальной схемы.

Применение синхронизирующего триггера (*б*) обеспечивает прохождение на выход пропускающего элемента 2И только импульсов полной длительности. Разрешающий сигнал, проходя через триггер, который тактируется разрешаемым сигналом, становится синхронным с тактовым сигналом и гарантирует прохождение на выход обязательно целого количества тактовых импульсов, целого количества периодов тактового генератора.

Триггеры позволяют также строить линии задержки цифровых сигналов, для чего несколько триггеров соединяется в последовательную цепочку, причем все они тактируются единым тактовым сигналом С. Такое включение позволяет, например, одновременно обрабатывать комбинационными схемами несколько последовательных во времени состояний какого-то одного сигнала.

В качестве примера на рис. 4.11 приведена схема, которая выделяет во входном сигнале трехтактовую последовательность 010. Цепочка из трех триггеров Т1, Т2 и Т3, тактируемых единым синхросигналом, запоминает три последовательных со-

стояния входного сигнала. Например, если на выходе триггера Т2 будет зафиксировано состояние входного сигнала в N-м такте, то на выходе триггера Т1 будет состояние входного сигнала в такте (N+1), а на выходе триггера Т3 — в такте (N-1). Из-за конечной величины задержки переключения триггеров в каждый следующий триггер входной сигнал будет переписываться еще до того, как он изменит свое значение вследствие переключения предыдущего триггера.

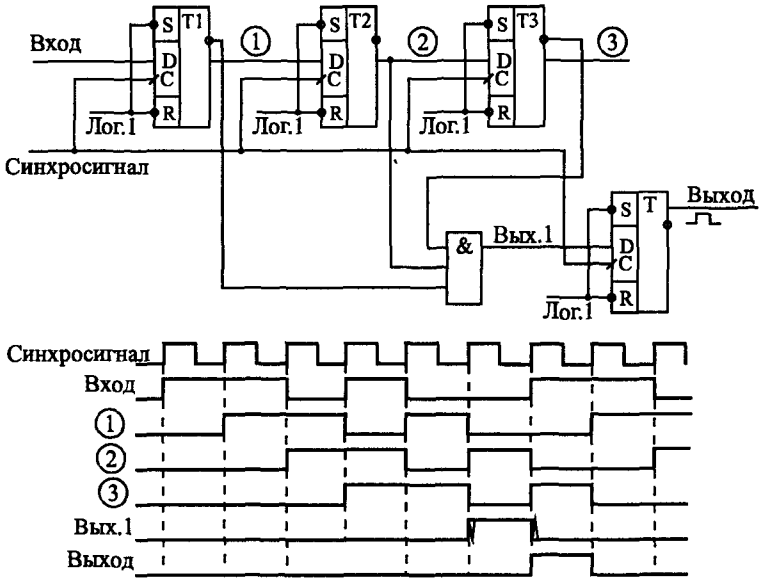


Рис. 4.11. Выделение 3-тактовой последовательности тактов во входном сигнале.

Подавая выходные сигналы триггеров (прямые или инверсные в зависимости от нужных уровней) на элемент И с нужным числом входов, можно зафиксировать любую трехтактовую последовательность во входном сигнале. Для предотвращения появления паразитных импульсов в выходном сигнале (они возможны, так как входные сигналы элемента И изменяются почти одновременно) применяется выходной триггер Т, тактируемый тем же самым общим синхросигналом. На выходе триггера Т получаем единичный сигнал, соответствующий последователь-

ности 010 во входном сигнале. Правда, этот выходной сигнал будет задержан относительно конца выделяемой последовательности 010 на два такта.

Конечно, применение триггеров не ограничивается рассмотренными примерами, все области их применения трудно даже перечислить. Мы же рассмотрим здесь еще несколько примеров использования триггеров.

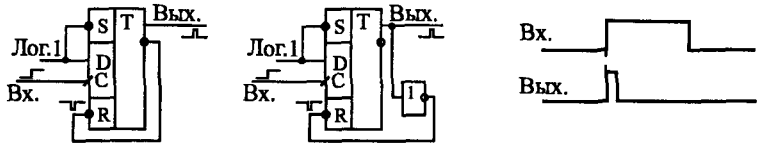


Рис. 4.12. Формирователь короткого импульса по фронту входного сигнала.

D-триггер позволяет довольно просто формировать выходной короткий импульс по фронту входного сигнала. Для этого даже не нужно никаких времязадающих RC-цепочек. Длительность выходного импульса определяется задержкой срабатывания триггера. Формирователь короткого импульса по положительному фронту входного сигнала (рис. 4.12) образуется путем подачи выходного сигнала триггера на вход сброса.

По положительному фронту на входе С триггер перебрасывается в единицу, но выходной сигнал триггера по цепи обратной связи тут же сбрасывает его обратно в нуль. Преимуществом данной схемы является то, что триггер имеет как прямой, так и инверсный выходы, поэтому мы получаем как положительный короткий импульс, так и отрицательный. В некоторых случаях в цепь этой обратной связи надо включать дополнительный инвертор для устойчивой работы схемы. Например, триггеры серии К155 не требуют инвертора, а триггеры серии КР1533 — требуют.

Применение триггеров совместно с другими микросхемами часто позволяет избежать появления паразитных коротких импульсов, обеспечить надежную и уверенную работу схемы. Например, на рис. 4.13 представлена схема, различающая короткие и длинные импульсы, приходящие на ее вход. Такая схема позволяет применять одну линию связи для передачи двух сигналов разного назначения, что бывает очень удобно при связи устройств, находящихся на большом расстоянии.

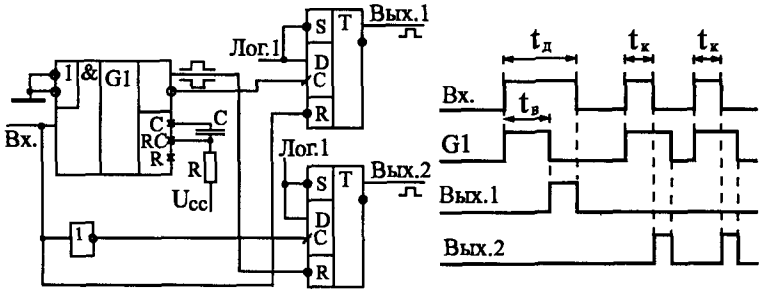


Рис. 4.13. Схема разделения коротких и длинных входных импульсов.

На вход схемы поступают короткие импульсы (длительностью $t_к$) и длинные импульсы (длительностью $t_д$). Конечно, на передающем конце надо обеспечить, чтобы эти импульсы формировались по очереди и с не слишком малой задержкой друг относительно друга. На выходе схемы формируются два сигнала, один из которых соответствует приходу короткого входного импульса, а другой — приходу длинного входного импульса.

Для различения входных импульсов используется одновибратор АГ1 с временем выдержки $t_в$, большим $t_к$, но меньшим $t_д$. Применение одновибратора в данном случае оправдано, так как требуемая точность времени выдержки невысока (считаем, что длительности импульсов различаются существенно). Выходные сигналы схемы формируются с помощью двух триггеров, а не простых двухвходовых логических элементов, что полностью исключает появление паразитных импульсов на фронтах.

Принцип работы схемы ясен из приведенной временной диаграммы. Одновибратор запускается по переднему фронту входного сигнала. Выходной сигнал Вых.1, соответствующий приходу длинного импульса, начинается по заднему фронту импульса одновибратора, а заканчивается по окончанию длинного входного импульса. Выходной сигнал Вых.2, соответствующий приходу входного короткого импульса, начинается по заднему фронту входного импульса, а заканчивается с окончанием импульса одновибратора.

Триггеры можно также использовать для обработки периодических последовательностей входных сигналов. При этом триггер, тактируемый кварцевым генератором, может очень точно различать частоты следования входных импульсов, то

есть выполнять функцию простейшего цифрового фильтра. Такие схемы выгодно отличаются от схем с одновибраторами и времязадающими RC-цепочками возможностью полностью интегрального исполнения и отсутствием какой бы то ни было настройки.

Простейший пример подобной обработки состоит в формировании огибающей входного сигнала. То есть при приходе входного сигнала заданной частоты выходной сигнал должен быть равен единице, а при отсутствии входного сигнала — нулю. Эта задача, как уже отмечалось (см. рис. 3.35,*г*), может быть решена с помощью одновибратора с перезапуском (типа АГ3). Однако применение триггеров значительно увеличивает точность срабатывания и позволяет работать с частотами, близкими к предельным для данного типа триггеров. Схема формирования огибающей состоит всего лишь из двух триггеров, тактируемых внешним синхросигналом (рис. 4.14). В данном случае предполагается, что частоты входного сигнала и тактового сигнала равны между собой.

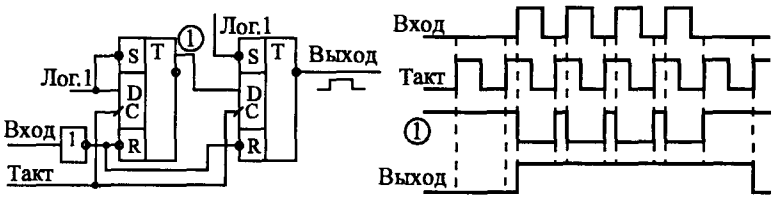


Рис. 4.14. Формирователь сигнала огибающей входного сигнала на триггерах.

Триггеры включены как двухтактная линия задержки с общим тактовым сигналом С и со сбросом входными сигналами. Самый первый входной импульс последовательности инициирует начало действия выходного сигнала, то есть переключение выхода в состояние логической 1, а заканчивается действие выходного сигнала, то есть возврат выхода в состояние логического 0, через 1–2 периода тактового сигнала после окончания входной последовательности (в зависимости от временного сдвига входного сигнала относительно тактового сигнала). Схема работает с входным сигналом любой частоты, большей половины частоты тактового сигнала (например, при тактовой частоте 10 МГц входной сигнал должен иметь частоту, боль-

шую 5 МГц). То есть за половину периода входной частоты не должно приходить больше одного положительного фронта тактового сигнала.

Этот же формирователь огибающей можно использовать в более сложных схемах. Примером может служить фильтр, который позволяет разделить две частоты входного сигнала, пропустить более высокочастотный сигнал и отсечь более низкочастотный (рис. 4.15).

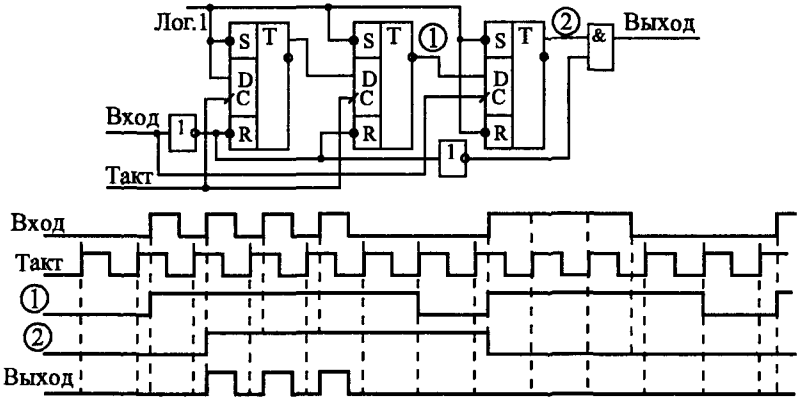


Рис. 4.15. Фильтр для пропускания высокочастотных сигналов на триггерах.

Фильтр состоит из трех триггеров и элемента 2И, работающего в режиме пропускания положительных входных импульсов. Два триггера (левые на схеме) образуют формирователь огибающей. Третий (правый на схеме) триггер выдает сигнал пропускания в случае, когда сигнал огибающей непрерывен, то есть когда частота входного сигнала составляет больше половины тактовой частоты. Если в момент прихода положительного фронта входного сигнала сигнал огибающей на выходе второго триггера нулевой, то пропускающий сигнал на выходе третьего триггера также нулевой, и импульсы не проходят на выход. При этом первый входной импульс пропускаемого сигнала на выход не проходит. Цепочка из двух инверторов компенсирует задержку срабатывания третьего триггера, она задерживает входной сигнал перед подачей его на вход выходного пропускающего элемента 2И.

Таким образом, фильтр надежно пропускает входные сигналы с частотой, большей половины тактовой частоты, и надежно задерживает сигналы с частотой, меньшей четверти тактовой частоты. Например, при тактовой частоте 10 МГц фильтр будет пропускать сигналы с частотой выше 5 МГц, и задерживать сигналы с частотой ниже 2,5 МГц. С частотами входного сигнала от 2,5 до 5 МГц работа фильтра не будет стабильной, она будет зависеть от временного сдвига между входным сигналом и тактовым сигналом.

Наконец, последняя схема на триггерах, которую мы рассмотрим, предназначена для кодирования входного сигнала в манчестерский код (или код Манчестер-II). Этот код широко используется при передаче сигналов на большие расстояния, в частности в локальных сетях.

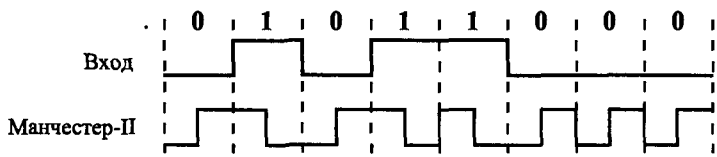


Рис. 4.16. Манчестерское кодирование.

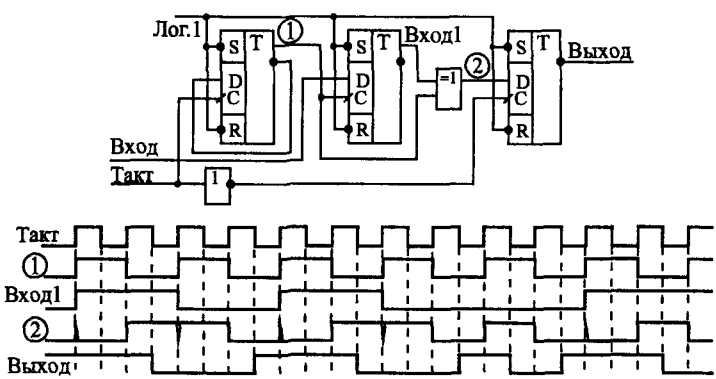


Рис. 4.17. Кодировщик манчестерского кода на триггерах.

Суть манчестерского кодирования иллюстрируется рис. 4.16. Входной сигнал представляет собой последовательность битов равной длительности. В каждом такте передается один бит ин-

формации. Манчестерский код заменяет единичный информационный бит на отрицательный переход в центре битового интервала, а нулевой информационный бит — на положительный переход в центре битового интервала. Таким образом, в центре каждого битового интервала сигнала в манчестерском коде обязательно имеется фронт (положительный или отрицательный), который может быть использован приемником этого сигнала для синхронизации приема каждого информационного бита. Поэтому манчестерский код называется самосинхронизирующимся кодом.

Кодировщик (он же шифратор) манчестерского кода (рис. 4.17) включает в себя элемент Исключающее ИЛИ, который, собственно, и производит кодирование, а также три триггера для синхронизации. Один триггер (левый на схеме) работает в счетном режиме, деля частоту тактового сигнала в два раза. Один триггер (центральный) синхронизирует входной информационный сигнал с тактовым сигналом половинной частоты. Наконец, последний, третий триггер (правый) синхронизирует выходной сигнал для устранения в нем паразитных коротких импульсов в моменты изменения входного сигнала. Он фиксирует выходной сигнал элемента Исключающее ИЛИ (уже готовый манчестерский код) через четверть периода после изменения входного сигнала Вход 1 (по отрицательному фронту исходного тактового сигнала).

4.2. Регистры

Регистры (английское Register) представляют собой, по сути, несколько D-триггеров (обычно от 4 до 16), соединенных между собой тем или иным способом. Поэтому принципиальной разницы между ними и отдельными D-триггерами не существует. Правда, триггеры, входящие в состав регистров, не имеют такого количества разнообразных управляющих входов, как одиночные триггеры.

На схемах регистры обозначаются буквами RG. В отечественных сериях микросхем регистрам соответствуют буквы ИР. Все регистры делятся на две большие группы (рис. 4.18):

- параллельные регистры;
- регистры сдвига (или сдвиговые регистры).

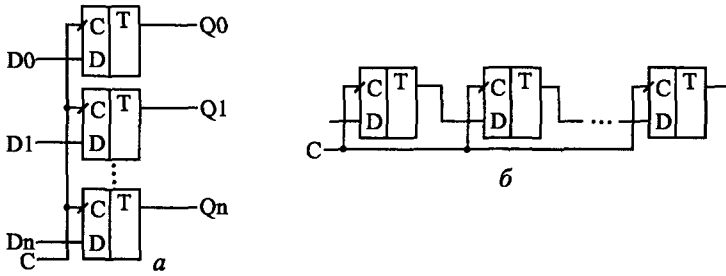


Рис. 4.18. Структура параллельного регистра (*а*) и сдвигового регистра (*б*).

Существуют и регистры других типов, но они применяются гораздо реже, чем параллельные и сдвиговые, так как имеют узкоспециальное назначение.

В параллельных регистрах (*а*) каждый из триггеров имеет свой независимый информационный вход (D) и свой независимый информационный выход. Тактовые входы (C) всех триггеров соединены между собой. В результате параллельный регистр представляет собой многоразрядный, многовходовый триггер.

В сдвиговых регистрах (*б*) все триггеры соединены в последовательную цепочку (выход каждого предыдущего триггера соединен со входом D следующего триггера). Тактовые входы всех триггеров (C) объединены между собой. В результате такой триггер может рассматриваться как линия задержки, входной сигнал которой последовательно перезаписывается из триггера в триггер по фронту тактового сигнала C. Информационные входы и выходы триггеров могут быть выведены наружу, а могут и не выводиться в зависимости от функции, выполняемой регистром.

Параллельные регистры в свою очередь делятся на две группы:

- регистры, срабатывающие по фронту управляющего сигнала C (или тактируемые регистры);
- регистры, срабатывающие по уровню управляющего сигнала C (или стробируемые регистры).

Чаще всего в цифровых схемах используются регистры, управляемые фронтом (то есть тактируемые), однако и стробируемые регистры имеют свой круг задач, в которых их ничто не может заменить.

4.2.1. Регистры, срабатывающие по фронту

Принцип действия регистров, срабатывающих по фронту тактового сигнала, ничем не отличается от принципа действия D-триггера. По положительному фронту тактового сигнала C каждый из выходов регистра устанавливается в тот уровень, который был в этот момент на соответствующем данному выходу входе D, и сохраняется таковым до прихода следующего положительного фронта сигнала C. То есть если триггер запоминает один сигнал (один двоичный разряд, один бит), то регистр запоминает сразу несколько (4, 6, 8, 16) сигналов (несколько разрядов, битов). Память регистра сохраняется до момента выключения питания схемы.

В стандартные серии входит несколько типов параллельных регистров, срабатывающих по фронту (рис. 4.19). Различаются они количеством разрядов, наличием или отсутствием инверсных выходов, наличием или отсутствием входа сброса (-R) или разрешения записи (-WE), а также типом выходных каскадов (2C или 3C) и соответственно наличием или отсутствием входа разрешения -EZ. Иногда на схемах тактовый вход C обозначается WR — сигнал записи в регистр.

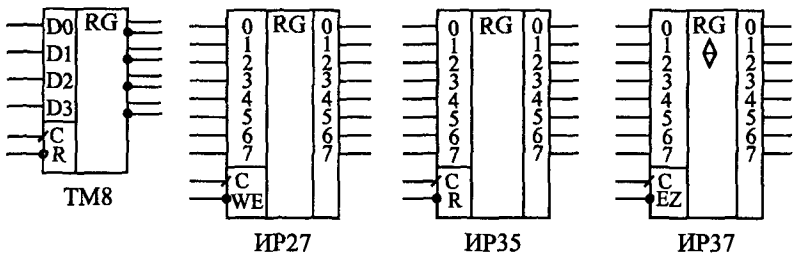


Рис. 4.19. Параллельные регистры стандартных серий, срабатывающие по фронту.

Большинство регистров имеют восемь разрядов, то есть запоминают один байт информации. Регистр TM8 в справочниках обычно называется счетверенным D-триггером (он и в наименовании несет буквы TM), хотя он вполне может рассматриваться и как регистр, так как тактовый вход C и вход сброса -R у всех четырех триггеров объединены между собой.

Таблицы истинности регистров очень просты и не отличаются принципиально от таблицы истинности D-триггеров. Отличие от триггеров появляется только в случае наличия у регистра дополнительных управляющих входов разрешения записи -WE и разрешения выхода -EZ. В качестве примеров ниже приведены таблицы истинности регистра ИР27 и регистра ИР37 (табл. 4.5. и 4.6 соответственно). По переходу тактового сигнала С из 0 в 1 (положительный фронт) оба регистра записывают в себя входную информацию.

Таблица 4.5. Таблица истинности регистра ИР27

Входы			Выходы
-WE	C	D	Q
0	0—>1	0	0
0	0—>1	1	1
0	0	X	Не меняется
0	1	X	Не меняется
1	X	X	Не меняется

Таблица 4.6. Таблица истинности регистра ИР37

Входы			Выходы
-EZ	C	D	Q
0	0—>1	0	0
0	0—>1	1	1
0	0	X	Не меняется
0	1	X	Не меняется
1	X	X	Z

Все регистры, имеющие выход с тремя состояниями, обеспечивают повышенную нагрузочную способность. Задержка переключения регистров примерно соответствует задержке переключения триггеров. Все временные ограничения, накладываемые на входные сигналы в случае триггеров (см. разд. 4.1.1), справедливы и для входных сигналов регистров. Например, не должна быть слишком малой длительность сигнала С, а также не должна быть слишком малой задержка между установлением

сигнала D и приходом положительного фронта сигнала C. Иначе работа регистра может быть нестабильной или даже неправильной.

Одно из основных применений регистров состоит в хранении требуемого кода в течение нужного времени. Если для работы остальной части схемы необходимо иметь входной код, который можно легко изменять, то для этого как раз подходит регистр.

На рис. 4.20 показана типичная схема включения регистра для хранения кода и временная диаграмма его работы. Код на входе регистра может изменяться произвольным образом, но в тот момент, когда этот код принимает необходимое значение, на вход C триггера подается синхросигнал (строб), который записывает код в регистр. Этот код будет храниться в регистре до прихода следующего строба. Причем важно и то, что все разряды выходного кода регистра будут переключаться одновременно даже в том случае, когда разряды входного кода переключаются не одновременно. Главное, чтобы к приходу положительного фронта строба (сигнала C) все разряды входного кода приняли нужное, устойчивое значение.

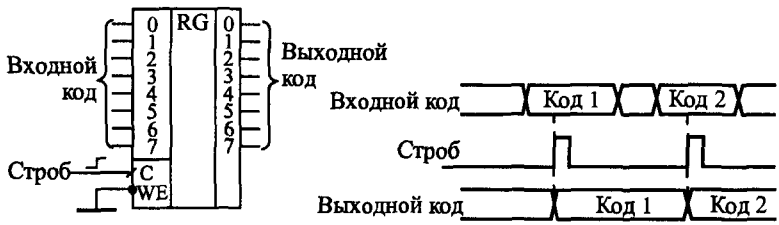


Рис. 4.20. Хранение кода в параллельном регистре.

Еще одно важнейшее применение регистров связано с запоминанием нескольких последовательных значений изменяющегося входного кода. Это позволяет, например, сравнивать предыдущее значение кода с последующим значением этого же кода или производить арифметические операции над несколькими последовательными значениями одного и того же кода. То есть регистр в данном случае выступает как элемент линии задержки, хранящей в себе историю поведения входного кода.

Для примера на рис. 4.21 показана схема вычисления разности двух последовательных значений входного кода. Такая задача возникает в частности при цифровой обработке аналоговых сигналов. Последовательные значения входного 4-разрядного кода сопровождаются тактовым сигналом, по положительному фронту которого производится запись в два последовательно включенных регистра. Когда на выходе регистра RG1 присутствует N-е значение входного кода, на выходе регистра RG2 будет (N-1)-е значение этого же кода.

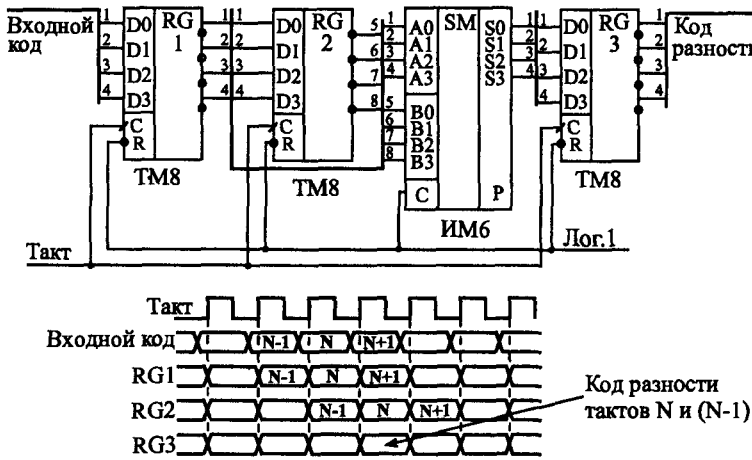


Рис. 4.21. Схема вычисления разности значений кодов в двух последовательных тактах.

Подавая эти два кода с выходов регистров на 4-разрядный сумматор, включенный в режиме вычитания (см. рис. 3.22), мы получаем на выходе сумматора код разности между N-м значением и (N-1)-м значением. В данном случае очень удобен регистр ТМ8, имеющий инверсные выходы. Для обеспечения строго одновременного изменения выходных сигналов сумматора можно включить дополнительный выходной регистр RG3, тактируемый тем же самым общим тактовым сигналом. Правда, код разности при этом будет задержан на один такт.

Регистры также широко используются для организации конвейерной обработки, позволяющей существенно повысить тактовую частоту работы схемы. Ускорение при этом достигается

за счет распараллеливания работы нескольких последовательно включенных узлов схемы.

Пусть, например, последовательность входных кодов, следующих с периодом T, поступает на вход цепочки из двух узлов, производящих обработку или преобразование этих кодов (рис. 4.22). Узлы эти могут представлять собой комбинационные микросхемы (например сумматоры) или более сложные устройства, включающие в себя микросхемы счетчиков или микросхемы памяти. Главное состоит в том, что выходные сигналы этих узлов выставляются не мгновенно, а в течение какого-то конечного времени, величина которого определяется внутренним строением узла. Пусть задержка установления выходного кода первого узла равняется t_1, а задержка установления выходного кода второго узла составляет t_2. Очевидно, что период следования входных кодов T не должен быть меньше, чем сумма этих двух задержек:

$$T > t_1 + t_2.$$

Иначе код на выходе цепочки может никогда не принять устойчивого значения, так как переходный процесс предыдущего такта будет сменяться переходным процессом следующего такта. То есть быстродействие узлов накладывает жесткое ограничение на тактовую частоту.

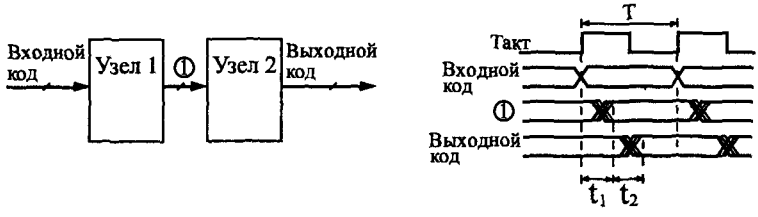


Рис. 4.22. Работа последовательной цепочки двух узлов.

Однако можно обойти это ограничение, если воспользоваться принципом конвейера, заставить узлы работать не последовательно, а параллельно. Это достигается включением между узлами регистра, тактируемого входным тактовым сигналом. Еще один регистр целесообразно включить на входе второго узла, что обеспечит длительность устойчивого кода на выходе всего устройства, равную длительности периода тактового сигнала T

(рис. 4.23). В результате ограничение на период тактового сигнала становится более мягким: T не должно быть меньше максимальной из двух величин t_1 и t_2 с добавлением времени задержки регистра:

$$T > \max\{t_1, t_2\} + t_{RG}.$$

То есть к следующему фронту тактового сигнала должен закончить свою работу самый медленный из узлов, и тогда его выходной код будет записан в регистр правильно.

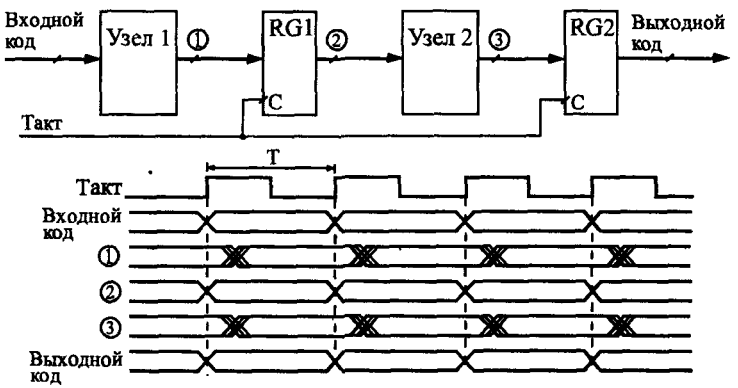


Рис. 4.23. Конвейерная обработка с помощью регистров.

Точно так же можно построить конвейер на любое количество последовательно включенных узлов. Конечно, в результате введения конвейера происходит задержка выполнения полной функции устройства на число тактов, равное числу введенных регистров. Однако в том случае, когда необходимо обрабатывать большие последовательности входных кодов, эта задержка наблюдается только один раз — в самом начале последовательности, а затем уже она не имеет значения.

Регистры могут также применяться в составе вычислителей, выполняя функцию накопителя результата вычислений. В данном случае мы уже имеем дело с более сложной обработкой информации, чем в случае чисто комбинационных схем. С каждым тактом в регистре обновляется содержимое, являющееся результатом математической обработки входного кода и результата предыдущего вычисления.

Рассмотрим два примера схем таких вычислителей.

Первая схема известна как **накапливающий сумматор**, применяющийся, например, в цифровых генераторах аналоговых сигналов. В самом названии схемы отражена ее функция: она суммирует и накапливает результат. Накапливающий сумматор (рис. 4.24) состоит из сумматора и выходного регистра, охваченных обратной связью.

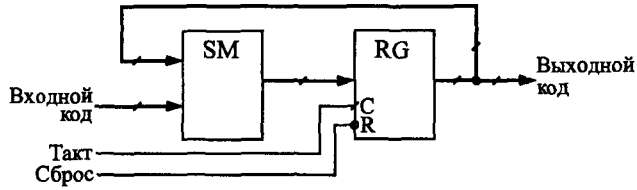


Рис. 4.24. Структура накапливающего сумматора.

Таким образом, на один вход сумматора подается код с выходов регистра, а на другой вход — входной код. В результате с каждым следующим фронтом тактового сигнала в регистр записывается код суммы входного кода с предыдущим содержимым регистра, с предыдущей суммой. Например, если входной код равен 3, а в регистре записан код 6, то в следующем такте в регистр будет записан код 9 (то есть 6+3), в следующем такте — код 12 (то есть 9+3) и т. д. Получается, что на выходе накапливающего сумматора формируется равномерно увеличивающийся двоичный код, и шаг этого увеличения можно менять. В данном случае удобно применять регистр со сбросом, например ИР35.

Отметим три особенности накапливающего сумматора.

Во-первых, когда выходной код достигает максимальной величины (становится больше 2^n, где n — количество разрядов регистра), происходит переполнение схемы и возобновление ее работы с минимальных значений кода. Однако совсем не обязательно в следующем цикле работы будут повторены те же значения кода, что и в предыдущем. Например, пусть n равняется 4, то есть максимальное число на выходе регистра равно 1111 в двоичном коде или 15 в десятичном коде. Пусть входной код равен 3. Тогда после начального сброса регистра выходной код будет нарастать так: 0, 3, 6, 9, 12, 15, 2, 5, 8, 11, 14, 1, 4, ... Это

происходит потому, что суммирование чисел 15 и 3 даст 18 или в двоичном коде 10010, а так как мы работаем только с младшими четырьмя разрядами, у нас получится 0010 или 2.

Во-вторых, особенность накапливающего сумматора состоит в том, что при больших значениях входного кода (больших половины максимально возможной величины) он может рассматриваться как накапливающий вычитатель. Пусть, например, входной код 4-разрядного сумматора равен 15 (1111 в двоичном коде), а в регистре записано число 13 (1011 в двоичном коде). В следующем такте в регистр запишется сумма 1101 + 1111 = 11100, а без старшего разряда — 1100, то есть 12. То есть выходной код уменьшился на единицу.

Наконец, в третьих, совсем не обязательно, чтобы шаг нарастания выходного кода накапливающего сумматора был целым числом (то есть 0, 1, 2, 3, ...). Если в качестве выходного кода берутся не все, а только старшие разряды регистра, то шаг нарастания вполне может быть дробным, например, 0,5, 1,25 или 3,75. Не вошедшие в выходной код разряды будут иметь вес 2^{-1} (то есть 0,5), 2^{-2} (то есть 0,25) и т. д. Правда результат суммирования в выходном коде будет представлен с точностью до целых чисел. При этом возможна ситуация, когда в течение нескольких тактов код на выходе не меняется, например, при входном коде 0,5 выходной код будет меняться один раз на два такта, а при входном коде 0,25 — один раз на четыре такта.

На рис. 4.25 показана схема 8-разрядного накапливающего сумматора на двух микросхемах сумматоров ИМ6 и одном регистре ИР35. В качестве выходного кода используется только 6 старших разрядов с выхода регистра, поэтому задание шага приращения возможно с точностью до 0,25. Максимально возможная частота тактового сигнала может быть определена по формуле:

$$T > t_{SM} + t_{RG},$$

где T — период тактового сигнала, t_{SM} — задержка 8-разрядного сумматора, а t_{RG} — задержка регистра.

Последний пример применения регистров, который мы рассмотрим, — это вычислитель максимального значения входного кода. Такой вычислитель, например, может применяться в схемах цифровых осциллографов для измерения амплитуды входного аналогового сигнала.

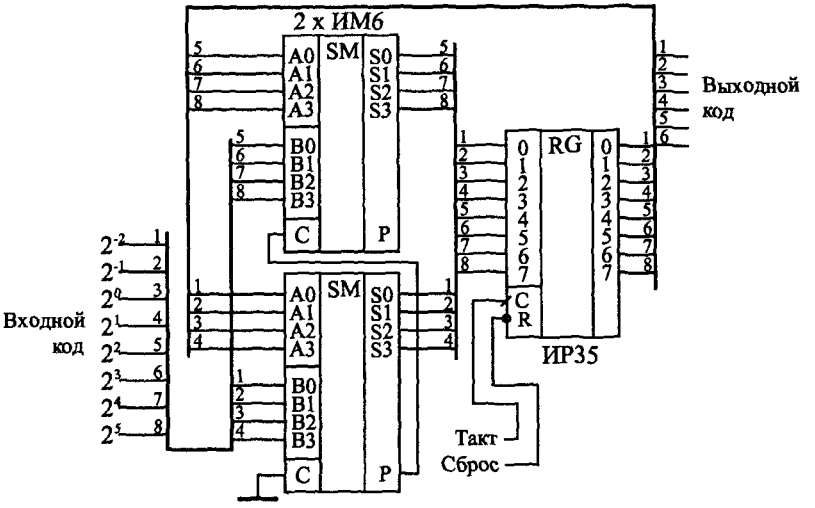


Рис. 4.25. Накапливающий сумматор.

Пусть мы имеем последовательность входных кодов, и нам необходимо выявить экстремальный (то есть максимальный или минимальный) код из всей этой последовательности. Эта задача решается довольно просто путем применения компаратора кодов и регистра, охваченных обратной связью (рис. 4.26).

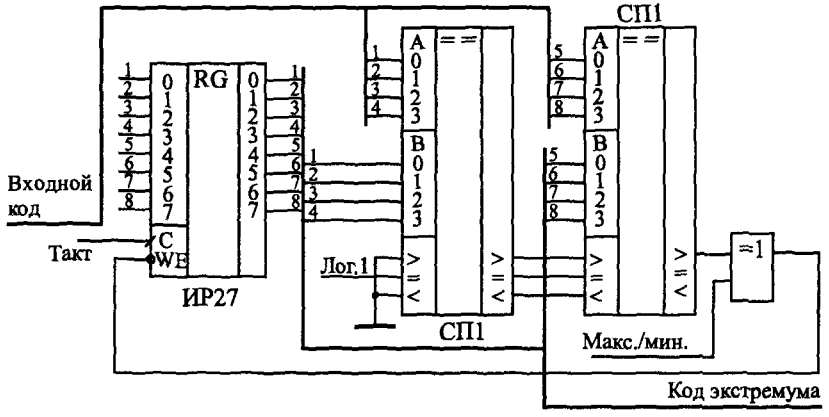


Рис. 4.26. Вычислитель экстремального значения входного кода.

В данном случае удобно использовать регистр со входом разрешения записи (ИР27). В регистре сохраняется код экстремума (максимума или минимума), 8-разрядный компаратор, составленный из двух микросхем СП1, сравнивает содержимое регистра и текущее значение входного кода. Элемент Исключающее ИЛИ выполняет функцию управляемого инвертора, выбирая режим вычисления максимума (единица на управляющем входе) или минимума (нуль на управляющем входе).

Допустим, мы вычисляем максимум. При этом запись в регистр текущего значения входного кода будет производиться только в том случае, когда это текущее значение больше числа, содержащегося в регистре. На выходе «>» компаратора кодов будет тогда сигнал логической единицы, а на входе разрешения записи регистра -WE — сигнал логического нуля. Если же текущее значение входного кода меньше кода, содержащегося в регистре, запись не производится. После окончания входной последовательности кодов (или после окончания одного ее периода при периодической последовательности) в регистре останется максимальное значение входного кода.

Аналогично вычисляется и минимум, только в данном случае в регистр будет записываться не только код, меньший числа в регистре, но и код, равный этому числу. Понятно, что на конечный результат вычисления это никак не повлияет.

4.2.2. Регистры, срабатывающие по уровню

Параллельные регистры, срабатывающие по уровню стробирующего сигнала (или, как их еще называют, регистры-защелки, английское Latch), можно рассматривать как некий гибрид между буфером и регистром. Когда сигнал на стробирующем входе единичный, такой регистр пропускает через себя входные информационные сигналы, а когда стробирующий сигнал становится равен нулю, регистр переходит в режим хранения последнего из пропущенных значений входных сигналов.

Применение таких регистров сильно ограничено, хотя иногда они довольно удобны. В некоторых схемах они могут успешно заменять регистры, срабатывающие по фронту, а в других схемах их применение вместо регистров, срабатывающих по фронту, недопустимо.

В стандартных сериях микросхем регистры, срабатывающие по уровню, представлены гораздо меньше, чем регистры, срабатывающие по фронту. На рис. 4.27 в качестве примеров показаны две микросхемы этого типа: 4-разрядного регистра ТМ7 и 8-разрядного регистра ИР22. Стробирующие входы С нередко на схемах обозначают Е (от английского Enable — разрешение) для того, чтобы не путать их с тактовыми входами D-триггеров.

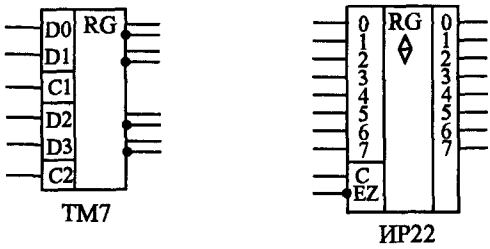


Рис. 4.27. Регистры, срабатывающие по уровню.

Микросхему ТМ7 (и близкую к ней ТМ5) часто называют набором триггеров, но ее можно рассматривать и как регистр. Микросхема состоит из четырех триггеров, стробирующие входы которых С соединены попарно, то есть можно говорить о двух двухразрядных регистрах-защелках. Входы С1 и С2 микросхемы управляют каждый двумя разрядами данных. Все триггеры имеют как прямые, так и инверсные выходы, что иногда очень удобно. Таблица истинности микросхемы ТМ7 приведена ниже (табл. 4.7).

Таблица 4.7. Таблица истинности регистра ТМ7

Входы		Выходы	
D	C	Q	-Q
0	1	0	1
1	1	1	0
0	0	Не меняется	
1	0	Не меняется	

При единице на входе С выходные сигналы повторяют входные, то есть регистр работает как обычный буфер с прямыми и инверсными выходами. При нуле на входе С на выходе ре-

гистра постоянно хранится та входная информация, которая была в момент прихода отрицательного фронта сигнала С. Однако говорить, что регистр ТМ7 срабатывает по отрицательному фронту сигнала С, неверно, так как информация на выходе меняется не только по этому фронту, но и в момент изменения входных сигналов при С = 1.

Регистр ИР22 отличается от ТМ7 тем, что имеет выходы с тремя состояниями (и соответственно вход разрешения всех выходов -EZ) и тем, что всеми восемью разрядами управляет один стробирующий сигнал С. Суть работы от этого не изменяется. При единице на входе С регистр работает как буфер-повторитель, а при нуле на входе С — хранит ту информацию, которая была на входе в момент отрицательного фронта сигнала С. Выходы у регистра ИР22 только прямые. Как и все регистры с тремя состояниями выхода, регистр ИР22 имеет повышенную нагрузочную способность. Таблица истинности регистра ИР22 приведена ниже (табл. 4.8).

Таблица 4.8. Таблица истинности регистра ИР22

Входы			Выход
-EZ	C	D	Q
0	1	1	1
0	1	0	0
0	0	X	Не меняется
1	X	X	Z

Величины задержек триггеров, срабатывающих по уровню, в 1,5—2 раза превышают задержки D-триггеров. Для правильной работы микросхем положительный импульс на входе С не должен быть слишком коротким, а задержка между изменением информации на входе D и отрицательным фронтом сигнала С не должна быть слишком малой. Информация на входе D не должна слишком быстро сниматься после отрицательного фронта сигнала С.

Основное применение регистра, срабатывающего по уровню стробирующего сигнала, состоит в запоминании на какое-то заданное время входного кода, причем в остальное время выходной код регистра должен повторять входной (рис. 4.28). Стро-

бирующий сигнал C в этом случае должен быть отрицательным на все время запоминания, и запоминаться будет входной код регистра в момент отрицательного (переднего) фронта сигнала C. Подобная функция бывает, например, необходима при построении устройств сопряжения для компьютеров. Регистр, по сути, продлевает во времени необходимое значение входного кода, в остальное время работая как повторитель.

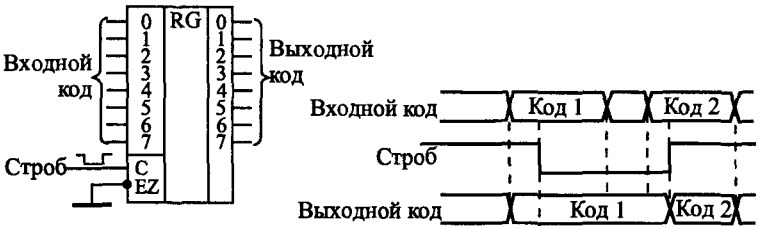


Рис. 4.28. Продление длительности входного кода с помощью регистра-защелки.

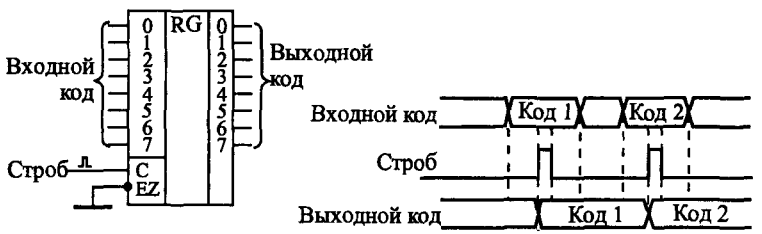


Рис. 4.29. Использование регистра-защелки для замены регистра, срабатывающего по фронту.

В ряде случаев регистры данного типа могут успешно заменять регистры, срабатывающие по фронту. Например, такая замена возможна в случае необходимости запоминания входного кода по сигналу C до момента прихода следующего сигнала C (рис. 4.29).

Сигнал C в данном случае должен быть коротким положительным импульсом, причем он обязательно должен быть «вложен» в запоминаемый входной код, то есть начинаться *после* начала (момента установления) кода, а заканчиваться *до* конца (момента снятия) кода (это так называемый вложенный цикл).

По переднему фронту сигнала С регистр перейдет в режим пропускания входного кода, а по заднему — в режим его хранения. Поэтому записываемый код на выходе регистра появится по положительному фронту сигнала С, то есть точно так же, как и в случае регистра, срабатывающего по фронту.

Однако подобная замена регистра, срабатывающего по фронту, на регистр, срабатывающий по уровню, возможна далеко не всегда. Некоторые схемы в принципе не могут работать с регистром-защелкой даже при очень коротком сигнале на входе С. Примером может служить уже упоминавшаяся схема накапливающего сумматора, которая работает исключительно с регистром, срабатывающим по фронту. Ведь при единичном сигнале на входе С регистр-защелка тут же перейдет в состояние пропускания входного кода, и в результате замкнется лавинообразная обратная связь: код с выхода регистра будет складываться со входным кодом бесконечное число раз (рис. 4.30). Конечно, при коротком импульсе на входе С этот неуправляемый процесс быстро прекратится, но что за информация в результате останется в регистре после окончания сигнала С, предсказать невозможно.

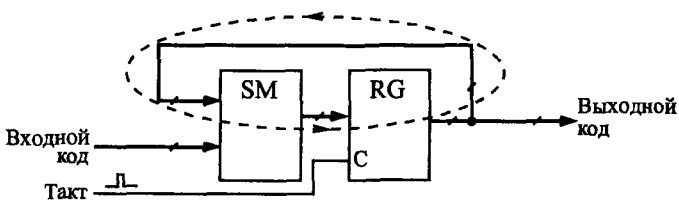


Рис. 4.30. Лавинообразная обратная связь в накапливающем сумматоре с регистром-защелкой.

И таких схем, принципиально не допускающих применения регистра-защелки, довольно много. Именно поэтому применение их сильно ограничено, а выбор микросхем в стандартных сериях невелик.

4.2.3. Сдвиговые регистры

Регистры сдвига или сдвиговые регистры (английское Shift Register) представляют собой, как уже отмечалось, последовательно соединенную цепочку триггеров. Основной режим

их работы — это сдвиг разрядов кода, записанного в эти триггеры, то есть по тактовому сигналу содержимое каждого предыдущего триггера переписывается в следующий по порядку в цепочке триггер. Код, хранящийся в регистре, с каждым тактом сдвигается на один разряд в сторону старших разрядов или в сторону младших разрядов, что и дало название регистрам данного типа.

С названиями направлений сдвига в сдвиговых регистрах часто возникает путаница. Сдвиг бывает двух видов: вправо (основной режим, который есть у всех сдвиговых регистров) и влево (этот режим есть только у некоторых, реверсивных сдвиговых регистров). Названия эти отражают внутреннюю структуру регистров сдвига (рис. 4.31) и перезапись сигналов последовательно по цепочке триггеров. При этом триггеры, вполне естественно, нумеруются слева направо, например, от 0 до 7 (или от 1 до 8) для 8-разрядных регистров. В результате сдвиг информации регистром вправо представляет собой сдвиг в сторону разрядов, имеющих большие номера, а сдвиг информации регистром влево — это сдвиг в сторону разрядов, имеющих меньшие номера.

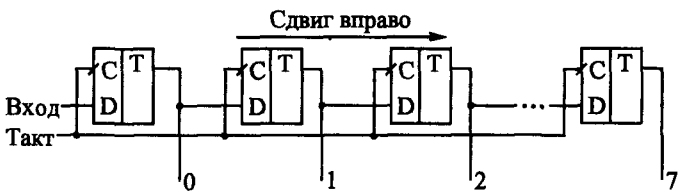


Рис. 4.31. Направление сдвига в сдвиговых регистрах.

Однако, как известно, в любом двоичном числе слева расположены старшие разряды, а справа — старшие разряды. Поэтому сдвиг двоичного числа вправо будет сдвигом в сторону младших разрядов, а сдвиг влево — сдвигом в сторону старших разрядов. Это противоречие не чей-то злой умысел, просто так сложилось исторически, и об этом надо помнить разработчику цифровой аппаратуры.

В стандартные серии цифровых микросхем входит несколько типов сдвиговых регистров, отличающихся возможными режимами работы, режимами записи, чтения и сдвига, а также ти-

пом выходных каскадов (2С или 3С). Большинство регистров сдвига имеет восемь разрядов. На рис. 4.32 представлены для примера четыре типа микросхем регистров сдвига.

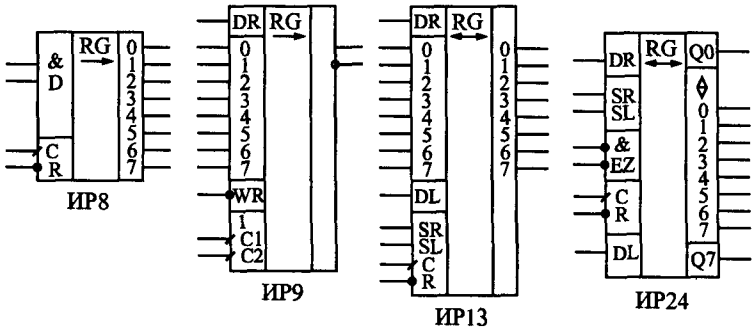


Рис. 4.32. Сдвиговые регистры.

Регистр ИР8 — наиболее простой из регистров сдвига. Он представляет собой 8-разрядную линию задержки, то есть имеет только один информационный вход, на который подается последовательная сдвигаемая информация (точнее, два входа, объединенных по функции 2И), и восемь параллельных выходов. Сдвиг в сторону выходов со старшими номерами осуществляется по переднему фронту тактового сигнала С. Имеется также вход сброса -R, по нулевому сигналу на котором все выходы регистра сбрасываются в нуль. Таблица истинности регистра ИР8 приведена ниже (табл. 4.9).

Таблица 4.9. Таблица истинности регистра сдвига ИР8

Входы				Выходы			
-R	C	D1	D2	Q0	Q1	...	Q7
0	X	X	X	0	0	...	0
1	0	X	X	Не меняются			
1	1	X	X	Не меняются			
1	0—>1	1	1	1	Q0	...	Q6
1	0—>1	0	X	0	Q0	...	Q6
1	0—>1	X	0	0	Q0	...	Q6

Регистр ИР9 выполняет функцию, обратную регистру ИР8. Если ИР8 преобразует входную последовательную информацию в выходную параллельную, то регистр ИР9 преобразует входную параллельную информацию в выходную последовательную. Однако суть сдвига не меняется, просто в регистре ИР9 все внутренние триггеры имеют выведенные параллельные входы, и только один, последний триггер имеет выход (причем как прямой, так и инверсный). Запись входного кода в регистр производится по нулевому сигналу на входе -WR. Сдвиг осуществляется по положительному фронту на одном из двух тактовых входов С1 и С2, объединенных по функции 2ИЛИ. Имеется также вход расширения DR, сигнал с которого в режиме сдвига перезаписывается в младший разряд сдвигового регистра. Таблица истинности регистра ИР9 приведена ниже (табл. 4.10).

Таблица 4.10. Таблица истинности регистра сдвига ИР9

Входы			Функция
-WR	C1	C2	
0	X	X	Параллельная запись
1	1	X	Хранение
1	X	1	Хранение
1	0	0—>1	Сдвиг
1	0—>1	0	Сдвиг

Как и все остальные сдвиговые регистры, регистры ИР8 и ИР9 допускают каскадирование, то есть совместное включение для увеличения разрядности. На рис. 4.33 показано объединение трех регистров ИР8, а на рис 4.34 — совместное включение трех регистров ИР9. В обоих случаях в результате объединения получается 24-разрядный сдвиговый регистр. При этом увеличение разрядности не приводит к увеличению задержки сдвига, так как тактовые входы всех используемых регистров объединяются параллельно. В случае регистров ИР8 входной последовательный код преобразуется в 24-разрядный выходной параллельный код. В случае регистров ИР9 входной 24-разрядный параллельный код преобразуется в выходной последовательный код.

Регистр ИР13 соединяет в себе возможности регистров ИР8 и ИР9. Он имеет как восемь входов для параллельной записи, так и соответствующие им восемь выходов параллельной информации. Сдвиг осуществляется по положительному фронту тактового сигнала С, причем сдвиг возможен как в сторону старших разрядов (вправо), так и в сторону младших разрядов (влево). Для наращивания разрядности у регистра ИР13 имеются последовательные информационные входы DR и DL, сигналы с которых вводятся соответственно в младший и в старший разряды. Предусмотрен сброс всех выходов регистра в нуль по нулевому сигналу на входе -R.

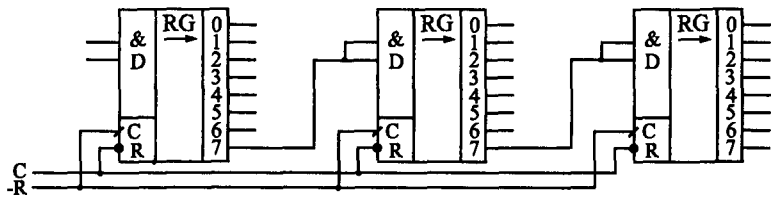


Рис. 4.33. Соединение регистров ИР8 для увеличения разрядности.

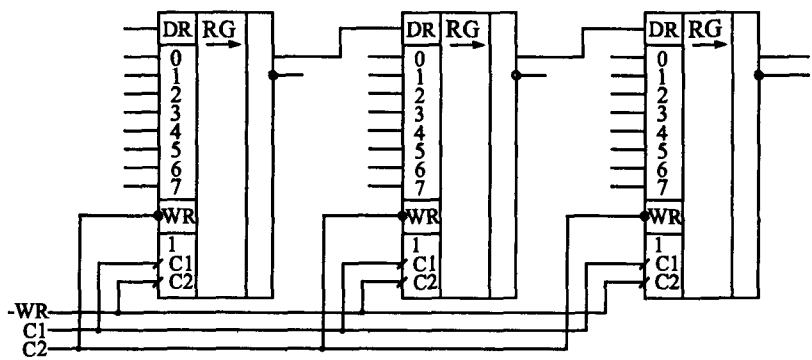


Рис. 4.34. Соединение регистров ИР9 для увеличения разрядности.

Режим работы регистра ИР13 определяется двумя управляющими входами SR и SL. При единице на входе SR и нуле на входе SL по фронту сигнала С происходит сдвиг в сторону старших разрядов. При нуле на входе SR и единице на входе SL по фронту сигнала С происходит сдвиг в сторону младших разрядов. При обоих единичных сигналах на входах SR и SL по

фронту сигнала C происходит параллельная загрузка информации в регистр. Все это видно из таблицы истинности регистра ИР13 (в табл. 4.11).

Отметим, что регистр ИР13 применяется заметно реже, чем более простые регистры ИР8 и ИР9, так как задач, в которых были бы нужны все возможности регистра ИР13 не так уж много, а управление работой регистра ИР13 довольно сложное.

Таблица 4.11. Таблица истинности регистра ИР13

Входы				Функция
C	-R	SR	SL	
X	0	X	X	Сброс
0—>1	1	1	0	Сдвиг вправо
0—>1	1	0	1	Сдвиг влево
0—>1	1	0	0	Хранение
0—>1	1	1	1	Параллельная запись

Наконец, последний сдвиговый регистр, который мы рассмотрим подробнее, это регистр ИР24. По своим возможностям он близок к ИР13, однако его главной особенностью является двунаправленная параллельная шина данных. То есть одни и те же выводы микросхемы используются как для параллельной записи информации в регистр, так и для параллельного чтения информации из регистра. При этом двунаправленные выводы данных имеют повышенную нагрузочную способность. Это позволяет легко сопрягать регистр ИР24 с многоразрядными микросхемами памяти и с двунаправленными буферами. Поэтому применяется данный регистр чаще, чем ИР13.

Регистр ИР24 обеспечивает сдвиг информации в обоих направлениях. Имеются входы расширения DR и DL, а также выходы расширения Q0 и Q7, что позволяет легко наращивать разрядность. Отличие выходов Q0 и Q7 от нулевого и седьмого разрядов данных состоит в том, что Q0 и Q7 — однонаправленные, то есть в любом режиме работы выдают информацию с выходов внутренних триггеров младшего и старшего разрядов. Тактируется регистр положительным фронтом сигнала C. Предусмотрен сброс регистра нулевым сигналом на входе -R.

Режим работы микросхемы определяется сигналами на управляющих входах SR и SL.

При единичном сигнале на SR и нулевом сигнале на SL по положительному фронту сигнала С происходит сдвиг информации вправо (в сторону разрядов с большими номерами). Запись в разряд 0 производится при этом со входа расширения DR.

При единичном сигнале на SL и нулевом сигнале на SR по положительному фронту сигнала С происходит сдвиг информации влево (в сторону разрядов с меньшими номерами). Запись в разряд 7 производится при этом со входа расширения DL.

При обоих нулях на входах SR и SL регистр переходит в режим хранения. Во всех этих случаях разряды данных работают как вход или как выход в зависимости от сигналов -EZ.

При обеих единицах на входах SR и SL по положительному фронту С в регистр записывается параллельный код, причем разряды данных переходят в состояние приема независимо от сигналов -EZ. Таблица истинности регистра ИР24 приведена ниже (табл. 4.12).

Таблица 4.12. Таблица истинности регистра ИР24

Входы				Функция
-R	C	SR	SL	
0	X	X	X	Сброс
1	0—>1	1	0	Сдвиг вправо
1	0—>1	0	1	Сдвиг влево
1	0—>1	1	1	Параллельная запись
1	X	0	0	Хранение

Объединяя два регистра ИР24, легко получить 16-разрядный сдвиговый регистр с сохранением всех возможностей одной микросхемы (рис. 4.35). Точно так же можно объединять и большее количество микросхем.

Главное применение всех регистров сдвига состоит в преобразовании параллельного кода в последовательный и наоборот. Такое преобразование используется, например, при передаче информации на большие расстояния (в информационных сетях), при записи информации на магнитные носители, при работе с телевизионными мониторами и с видеокамерами, а также во многих других случаях.

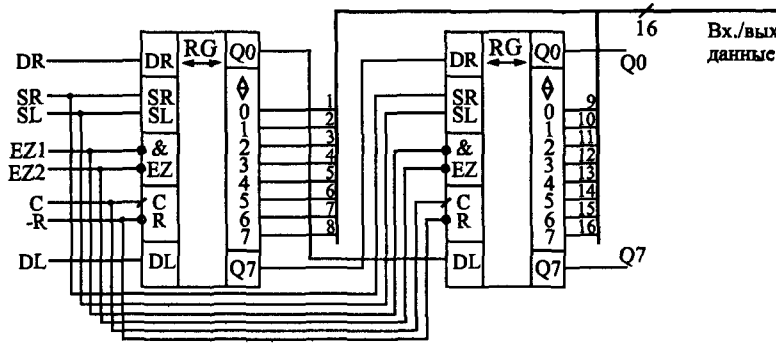


Рис. 4.35. Объединение регистров ИР24 для увеличения разрядности.

В качестве примера на рис. 4.36 показана простейшая схема передачи цифровой информации в последовательном коде по двум линиям: информационной и синхронизующей. Такая передача позволяет сократить количество соединительных проводов, а также упростить защиту передаваемых данных от действия внешних электромагнитных помех, правда, ценой снижения скорости передачи.

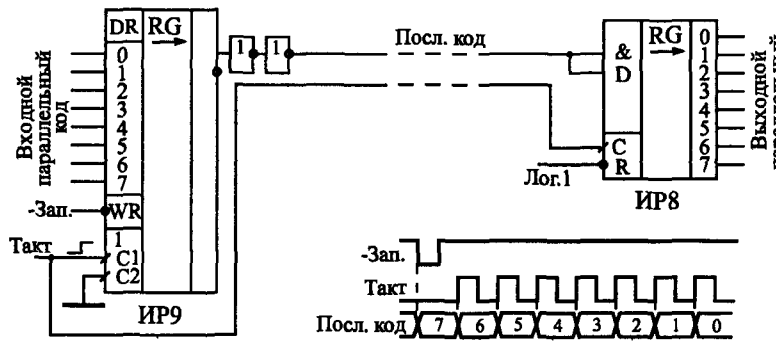


Рис. 4.36. Последовательная передача информации с помощью регистров сдвига.

На передающем конце (слева на рисунке) с помощью сдвигового регистра ИР9 входной параллельный 8-разрядный код преобразуется в последовательность разрядов данных, следующих с частотой тактового сигнала. На приемном конце (справа

на рисунке) с помощью сдвигового регистра ИР8 эта последовательность разрядов данных снова преобразуется в параллельный код. Оба регистра тактируются одним и тем же тактовым сигналом, который передается по линии связи параллельно с последовательностью данных. Для увеличения надежности передачи информационный сигнал дополнительно задерживается относительно фронта тактового сигнала с помощью цепочки из двух инверторов.

Первый бит последовательного входа (со входа 7 регистра ИР9) начинает передаваться с началом сигнала записи -Зап. Следующие разряды передаются с каждым следующим положительным фронтом тактового сигнала С. Последним передается сигнал со входа 0. В регистр ИР8 разряды последовательного кода записываются в том же самом порядке, в каком они были в регистре ИР9. По окончании передачи первый переданный сигнал данных окажется в разряде 7 шины данных регистра ИР8, а последний переданный сигнал данных — в разряде 0.

Следующее применение сдвиговых регистров состоит в организации всевозможных линий задержек, особенно имеющих значительное количество каскадов. С помощью сдвиговых регистров можно обеспечить задержку любого входного сигнала на целое число тактов. Правда, надо учитывать, что длительность входного сигнала (и любого его элемента) будет также передаваться по линии задержки с точностью до одного такта. Такие линии задержки могут применяться для сравнения нескольких последующих тактов входного сигнала, для выполнения арифметических операций с несколькими тактами входного сигнала, для других подобных целей. Работа линии задержки на регистре сдвига иллюстрируется рис. 4.37.

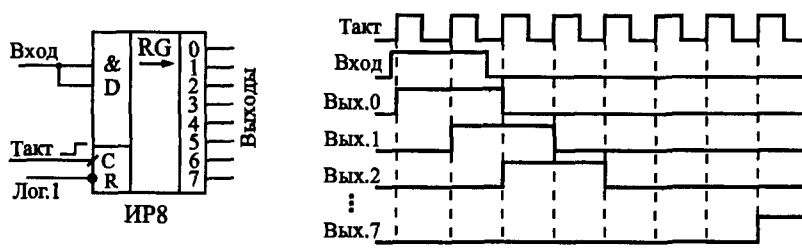


Рис. 4.37. Линия задержки входного сигнала на регистре сдвига.

Сдвиговые регистры могут также применяться для формирования импульсов заданной длительности, причем длительность импульса может задаваться управляющим кодом, то есть быть программно управляемой. На рис. 4.38 приведена возможная схема такого формирователя.

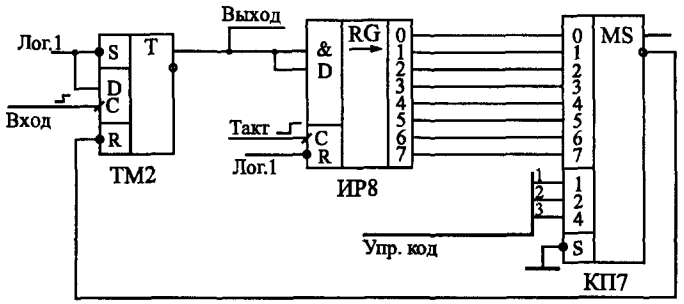


Рис. 4.38. Формирователь импульсов с длительностью, задаваемой управляющим кодом.

В исходном состоянии (до прихода положительного фронта входного сигнала) триггер сброшен в нуль, на всех выходах регистра сдвига нули, на инверсном выходе мультиплексора единица. На мультиплексор подан управляющий код, определяющий длительность выходного сигнала. При поступлении положительного фронта входного сигнала триггер перебрасывается в единицу (начало действия выходного импульса), и этот единичный сигнал начинает последовательно сдвигаться регистром сдвига по каждому фронту тактового сигнала.

Пусть управляющий код равен 5. Тогда в тот момент, когда на выходе 5 сдвигового регистра появится единица, она будет передана на выход мультиплексора КП7 с инверсией. При этом нулевой сигнал на входе -R триггера сбросит триггер в нуль, что соответствует окончанию действия выходного импульса.

Таким образом, длительность выходного сигнала будет определяться управляющим кодом. Погрешность установки этой длительности равна одному периоду тактового сигнала и зависит от временного сдвига между фронтом входного сигнала и фронтом ближайшего к нему тактового импульса. Чем больше длительность выходного сигнала, тем меньше относительная погрешность установки его точности. Например, при управ-

ляющем коде 0 длительность выходного сигнала может быть от 0 до T, где T — период тактового сигнала. А при управляющем коде 7 длительность выходного сигнала будет от 7T до 8T. При этом мы не учитываем задержек триггера, сдвигового регистра и мультиплексора.

Сдвиговые регистры могут также использоваться для умножения и деления двоичных чисел на 2^n, где n — целое число, большее нуля. Сдвиг двоичного числа вправо (в сторону младших разрядов) на один разряд равносилен делению на 2. Сдвиг двоичного числа влево (в сторону старших разрядов) на один разряд равносилен умножению на 2. Для того чтобы сдвиговый регистр умножал и делил двоичный код, надо всего лишь записать этот код в регистр и сдвинуть его нужное количество раз вправо или влево. Наиболее удобен для этого регистр ИР13. При этом необходимо, чтобы в освободившиеся разряды вводились нули, то есть на входы расширения DR и DL регистра нужно подать нулевые сигналы.

Наконец, последнее применение сдвигового регистра, которое мы рассмотрим, — это генератор случайной последовательности сигналов или случайной последовательности кодов. Строго говоря, последовательности будут не полностью случайные, а квазислучайные, то есть будут периодически повторяться, но период этот довольно большой. Случайные последовательности сигналов и кодов широко применяются в тестирующей аппаратуре, в генераторах шума, в логических игровых устройствах.

Задача состоит в том, чтобы выходной сигнал или код менял свое состояние случайно (или почти случайно). Сигнал должен случайно переключаться из 0 в 1 и из 1 в 0, а код должен случайно принимать значения из диапазона от 0 до (2^N-1), где N — число разрядов кода (например, от 0 до 255 при 8-разрядном коде). Псевдослучайные последовательности имеют то преимущество перед истинно случайными, что они предсказуемые и периодические, но в этом же и их недостаток.

Структура генератора квазислучайной последовательности на сдвиговом регистре очень проста (рис. 4.39). Она представляет собой регистр сдвига с параллельными выходами (например, ИР8), несколько (минимум два) выходных сигналов которого, объединены с помощью элемента Исключающее ИЛИ, с выхода которого сигнал подается на вход регистра, замыкая схему в кольцо. Схема тактируется сигналом с частотой f_T.

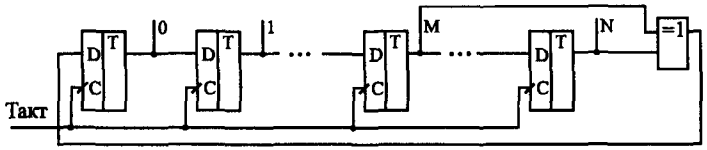


Рис. 4.39. Структура генератора псевдослучайной последовательности.

Выбор номеров разрядов для подключения обратной связи представляет собой непростую задачу, но существуют справочные таблицы, в которых они приведены. В любом случае одна из точек подключения — выход старшего разряда. В табл. 4.13 приведены точки подключения обратной связи для регистров сдвига с разным количеством разрядов N (номера разрядов считаются от нуля).

Таблица 4.13. Точки подключения обратной связи

N	7	8	15	16	24	31
Выходы	6, 5	7, 6, 4, 2	14, 13	15, 13, 12, 10	23, 22, 21, 16	30, 17

Из таблицы видно, что выгоднее брать число разрядов не кратное 8, например, 7, 15 или 31. В этом случае для обратной связи используются всего лишь два выхода, то есть достаточно одного двухвходового элемента Исключающее ИЛИ.

Период выходной последовательности генератора составляет $(2^N - 1)$ тактов, где N — количество разрядов регистра сдвига. За это время каждое из возможных значений выходного кода (кроме одного) встречается один раз. Количество единиц в выходном сигнале больше количества нулей на единицу.

Выходной код 000...0 представляет собой запрещенное состояние, так как он блокирует работу генератора, воспроизводя сам себя снова и снова. Но в то же время получиться такой нулевой код может только сам из себя, поэтому достаточно обеспечить, чтобы его не было при включении питания схемы.

Частоты в спектре выходного сигнала будут следовать с интервалом $f_T/(2^N-1)$, а огибающая спектра будет практически постоянной до частоты $0{,}25f_T$, то есть шум до этой частоты можно считать белым (спад в 3 дБ происходит на частоте $0{,}45\ f_T$).

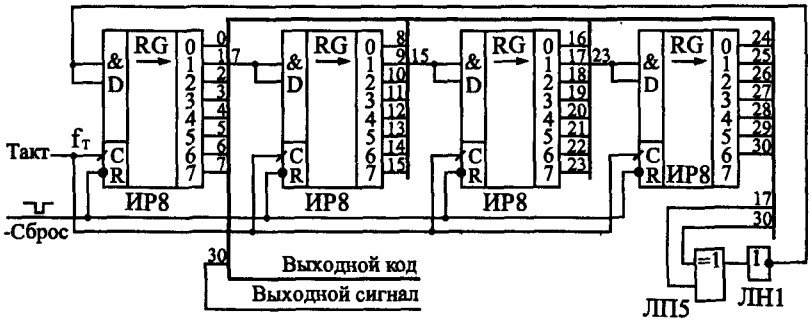


Рис. 4.40. 31-разрядный генератор псевдослучайной последовательности на регистрах сдвига.

На рис. 4.40 показана практическая схема генератора псевдослучайной последовательности на 31-разрядном сдвиговом регистре. Обратная связь осуществляется с выходов 30 и 17 регистра через двухвходовой элемент Исключающее ИЛИ с инвертором. Из-за применения инвертора запрещенным состоянием генератора является код 1111...1 (а не код 000...0), который в данном случае исключается очень просто — начальным сбросом регистров в нуль при включении питания по сигналу -Сброс. Генератор выдает квазислучайную последовательность 31-разрядных кодов со всех выходов регистра, а также квазислучайную последовательность нулей и единиц на любом из выходов регистра. Такой генератор использовала известная фирма Hewlett–Packard в своем генераторе шума.

Глава 5
ПРИМЕНЕНИЕ СЧЕТЧИКОВ

Счетчики представляют более высокий, чем регистры, уровень сложности цифровых микросхем, имеющих внутреннюю память. Хотя в основе любого счетчика лежат те же самые триггеры, которые образуют и регистры, но в счетчиках триггеры соединены более сложными связями, в результате чего их функции сложнее, и на их основе можно строить более сложные устройства, чем на регистрах. Точно так же, как и в случае регистров, внутренняя память счетчиков — оперативная, то есть ее содержимое сохраняется только до тех пор, пока включено питание схемы. С выключением питания память стирается, а при новом включении питания схемы содержимое памяти будет произвольным, случайным, зависящим только от конкретной микросхемы, то есть выходные сигналы счетчиков будут произвольными.

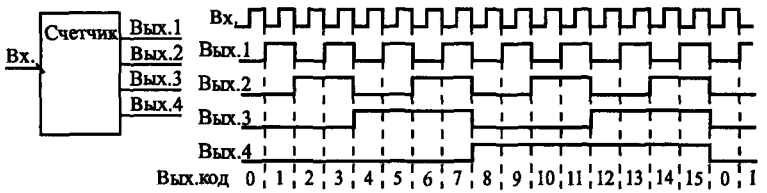


Рис. 5.1. Работа 4-разрядного двоичного счетчика.

Как следует из самого названия, счетчики предназначены для счета входных импульсов. То есть с приходом каждого нового входного импульса двоичный код на выходе счетчика увеличивается (или уменьшается) на единицу (рис. 5.1). Срабатывать счетчик может по отрицательному фронту входного (тактового) сигнала (как на рисунке) или по положительному фронту входного сигнала. Режим счета обеспечивается использованием внутренних триггеров, работающих в счетном режиме. Выходы счетчика представляют собой как раз выходы этих триггеров. Каждый выход счетчика представляет собой разряд двоичного кода, причем разряд, переключающийся чаще других (по каждому входному импульсу), будет младшим, а разряд, переключающийся реже других — старшим.

Счетчик может работать на увеличение выходного кода по каждому входному импульсу, это основной режим, имеющийся во всех счетчиках, он называется режимом прямого счета. Счетчик может также работать на уменьшение выходного кода по каждому входному импульсу, это режим обратного или инверсного счета, предусмотренный в счетчиках, называемых реверсивными. Инверсный счет бывает довольно удобен в схемах, где необходимо отсчитывать заданное количество входных импульсов.

Большинство счетчиков работают в обычном двоичном коде, то есть считают от 0 до $(2^N - 1)$, где N — число разрядов выходного кода счетчика. Например, 4-разрядный счетчик в режиме прямого счета будет считать от 0 (код 0000) до 15 (код 1111), а 8-разрядный — от 0 (код 0000 0000) до 255 (код 1111 1111). После максимального значения кода счетчик по следующему входному импульсу переключается опять в 0, то есть работает по кругу. Если же счет инверсный, то счетчик считает до нуля, а дальше переходит к максимальному коду 111...1.

Имеются также двоично-десятичные счетчики, предельный код на выходе которых не превышает максимального двоично-десятичного числа, возможного при данном количестве разрядов. Например, 4-разрядный двоично-десятичный счетчик в режиме прямого счета будет считать от 0 (код 0000) до 9 (код 1001), а затем снова от 0 до 9. А 8-разрядный двоично-десятичный счетчик будет считать от 0 (код 0000 0000) до 99 (код 1001 1001). При инверсном счете двоично-десятичные счетчики считают до нуля, а со следующим входным импульсом переходят к максимально возможному двоично-десятичному числу (то есть 9 для 4-разрядного счетчика, 99 для 8-разрядного счетчика). Двоично-десятичные счетчики удобны, например, при организации десятичной индикации их выходного кода. Применяются они гораздо реже обычных двоичных счетчиков.

По быстродействию все счетчики делятся на три большие группы:

- асинхронные (или последовательные) счетчики;
- синхронные счетчики с асинхронным переносом (или параллельные счетчики с последовательным переносом);
- синхронные (или параллельные) счетчики.

Принципиальные различия между этими группами проявляются только на втором уровне представления, на уровне модели с временными задержками. Причем больше всего различия эти проявляются при каскадировании счетчиков. Наибольшим быстродействием обладают синхронные счетчики, наименьшим — асинхронные счетчики, наиболее просто управляемые среди других. Каждая группа счетчиков имеет свои области применения, на которых мы и остановимся.

5.1. Асинхронные счетчики

Асинхронные счетчики строятся из простой цепочки JK-триггеров, каждый из которых работает в счетном режиме. Выходной сигнал каждого триггера служит входным сигналом для следующего триггера. Поэтому все разряды (выходы) асинхронного счетчика переключаются последовательно (отсюда название — последовательные счетчики), один за другим, начиная с младшего и кончая старшим. Каждый следующий разряд переключается с задержкой относительно предыдущего (рис. 5.2), то есть, вообще говоря, асинхронно, не одновременно с входным сигналом и с другими разрядами.

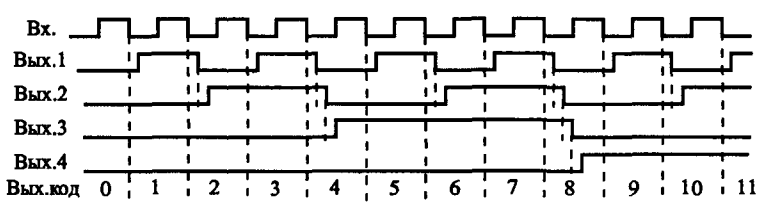


Рис. 5.2. Временная диаграмма работы 4-разрядного асинхронного счетчика.

Чем больше разрядов имеет счетчик, тем большее время ему требуется на полное переключение всех разрядов. Задержка переключения каждого разряда примерно равна задержке триггера, а полная задержка установления кода на выходе счетчика равна задержке одного разряда, умноженной на число разрядов счетчика. Легко заметить, что при периоде входного сигнала, меньшем полной задержки установления кода счетчика, правильный код на выходе счетчика просто не успеет установиться, поэтому такая ситуация не имеет смысла. Это накладывает же-

сткие ограничения на период (частоту) входного сигнала, причем увеличение, к примеру, вдвое количества разрядов счетчика автоматически уменьшает вдвое предельно допустимую частоту входного сигнала.

Таким образом, если нам нужен выходной код асинхронного счетчика, то есть все его выходные сигналы (разряды) одновременно, то должно выполняться следующее неравенство:

$$T > Nt_з,$$

где T — период входного сигнала, N — число разрядов счетчика, $t_з$ — время задержки одного разряда.

Надо еще учесть, что за период входного сигнала должно успеть сработать устройство (узел), на которое поступает выходной код счетчика, иначе счетчик просто не нужен, поэтому ограничение на частоту входного сигнала обычно бывает еще жестче.

В составе стандартных серий цифровых микросхем асинхронных счетчиков немного. В качестве примера на рис. 5.3. приведены три из них: четырехразрядный двоично-десятичный счетчик ИЕ2, четырехразрядный двоичный счетчик ИЕ5 и восьмиразрядный двоичный счетчик ИЕ19 (он же сдвоенный четырехразрядный счетчик).

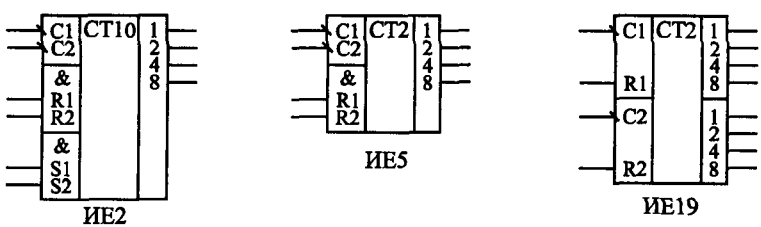


Рис. 5.3. Асинхронные счетчики стандартных серий.

У всех этих счетчиков управление работой очень простое: имеются лишь входы сброса в нуль или входы установки в 9 (только у ИЕ2). Все асинхронные счетчики работают по отрицательному фронту входного сигнала С (или, что то же самое, по заднему фронту положительного входного сигнала). У всех трех счетчиков выделены две независимые части, что увеличивает возможности их применения. При объединении этих двух час-

тей получается счетчик максимальной разрядности. Выходы счетчиков обозначают на схемах 0, 1, 2, 3, ... (как номера разрядов выходного двоичного кода) или 1, 2, 4, 8, ... (как веса каждого разряда двоичного кода).

Счетчик ИЕ2 имеет две части: один триггер (вход С1, выход 1) и три триггера (вход С2 и выходы 2, 4, 8). Таким образом, он состоит из одноразрядного счетчика и трехразрядного счетчика. Одиночный триггер работает в обычном счетном режиме, изменяя свое состояние по каждому отрицательному фронту сигнала С1, то есть делит частоту входного сигнала на 2. Три оставшихся триггера включены таким образом, чтобы считать до 5, то есть делить входную частоту сигнала С2 на 5. После достижения кода 4 (то есть 100) на выходах 2, 4 и 8 этот трехразрядный счетчик по следующему отрицательному фронту сигнала С2 сбрасывается в нуль. В результате при объединении выхода 1 микросхемы со входом С2 мы получаем 4-разрядный двоично-десятичный счетчик, делящий частоту входного сигнала С1 на 10, сбрасывающийся в нуль после достижения на выходах 1, 2, 4, 8 кода 9 (то есть 1001) по отрицательному фронту сигнала С1.

Счетчик ИЕ2 имеет два входа асинхронного сброса в нуль R1 и R2, объединенных по функции И, а также два входа установки в 9 — S1 и S2, также объединенных по функции И, причем установка в 9 блокирует установку в нуль. Наличие этих входов сброса и установки позволяет строить на базе счетчика ИЕ2 делители частоты с разными коэффициентами деления. Правда, этот счетчик используется довольно редко, значительно реже, чем другие асинхронные счетчики ИЕ5 и ИЕ19.

Таблица истинности асинхронного счетчика ИЕ2 при соединенном выходе 1 и входе С2 (при 4-разрядном выходном коде) приведена ниже (табл. 5.1), а состояния выходов при счете входных импульсов по тактам представлены в табл. 5.2.

Счетчик ИЕ5 точно так же, как и ИЕ2, имеет две части: один триггер (одноразрядный счетчик) со входом С1 и выходом 1 и три триггера (трехразрядный счетчик) со входом С2 и выходами 2, 4, 8. Оба счетчика двоичные, то есть первый считает до двух, а второй — до 8. При объединении входа С2 с выходом 1 получается 4-разрядный двоичный счетчик, считающий до 16. Счет производится по отрицательному фронту входных сигналов С1 и С2. Предусмотрена возможность сброса счетчика в нуль по сигналам R1 и R2, объединенным по функции И.

Таблица истинности счетчика ИЕ5 при соединении входа С2 и входа 1 (при 4-разрядном выходном коде) приведена ниже (табл. 5.3).

Таблица 5.1. Таблица истинности счетчика ИЕ2

Входы					Выходы			
C1	R1	R2	S1	S2	8	4	2	1
X	1	1	0	X	0	0	0	0
X	1	1	X	0	0	0	0	0
X	X	X	1	1	1	0	0	1
1—>0	X	0	X	0	Счет			
1—>0	0	X	0	X	Счет			
1—>0	0	X	X	0	Счет			
1—>0	X	0	0	X	Счет			

Таблица 5.2. Состояния выходов счетчика ИЕ2 при счете входных импульсов

Такт	Вых.8	Вых.4	Вых.2	Вых.1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

Таблица 5.3. Таблица истинности счетчика ИЕ5

Входы			Выходы			
C1	R1	R2	8	4	2	1
X	1	1	0	0	0	0
1—>0	0	X	Счет			
1—>0	X	0	Счет			

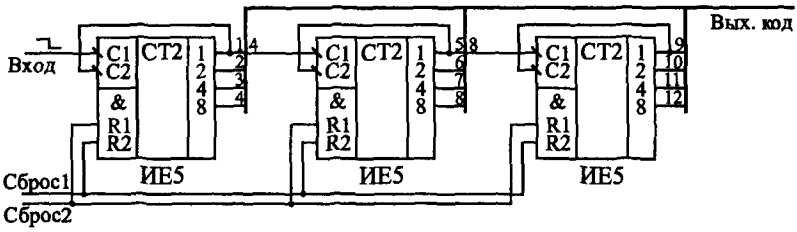


Рис. 5.4. Объединение трех счетчиков ИЕ5 для увеличения разрядности.

Объединять счетчики ИЕ5 для увеличения разрядности (каскадировать) очень просто: выход 8 предыдущего счетчика (выдающего более младшие разряды) нужно соединить со входом C1 следующего счетчика (выдающего более старшие разряды). На рис. 5.4 показано соединение трех счетчиков ИЕ5 для получения 12-разрядного асинхронного счетчика со сбросом в нуль. Точно так же можно объединять и счетчики ИЕ2, добавляя при этом входы общей установки счетчика в код 99...9. Однако при объединении надо помнить, что добавление каждого нового разряда увеличивает общую задержку переключения полученного счетчика. Многоразрядный асинхронный счетчик может получиться неприемлемо медленным.

Счетчик ИЕ19 можно считать сдвоенным вариантом счетчика ИЕ5. Он включает в себя два идентичных независимых друг от друга 4-разрядных асинхронных счетчика, каждый из которых имеет свой счетный вход C и свой вход сброса R. Считают оба счетчика, входящие в микросхему, по отрицательному фронту на своих входах C1 и C2. Сбрасываются они единичными сигналами на своих входах сброса R1 и R2.

Счетчики, входящие в микросхему ИЕ19, можно использовать самостоятельно, но можно и объединить их для получения 8-разрядного асинхронного счетчика с выходами 1, 2, 4, 8, 16, 32, 64, 128. Для такого объединения достаточно соединить выход 8 первого счетчика со счетным входом C2 второго счетчика. Если соединить два счетчика ИЕ19 (рис. 5.5), то получится уже 16-разрядный асинхронный двоичный счетчик. При этом выход 8 второго счетчика соединяется со счетным входом C1 первого счетчика. Однако и в данном случае каждый следующий разряд переключается с задержкой после переключения предыдущего.

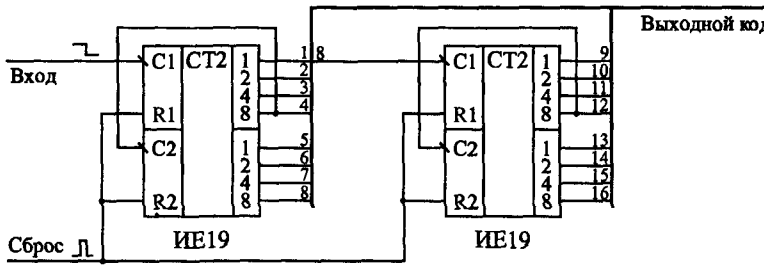


Рис. 5.5. Объединение двух счетчиков ИЕ19 для увеличения разрядности.

Основное применение асинхронных счетчиков состоит в построении всевозможных делителей частоты, то есть устройств, выдающих выходной сигнал с частотой, в несколько раз меньшей, чем частота входного сигнала. В данном случае нас интересует не выходной код счетчика, то есть не все его разряды одновременно, а только один разряд, поэтому взаимные задержки отдельных разрядов не играют роли, полная задержка переключения счетчика не имеет значения. Простейший пример делителя частоты на два — это триггер в счетном режиме или счетчик, выходным сигналом которого является выход первого, младшего разряда.

При построении делителей частоты иногда важна не только частота выходного сигнала, но и его форма, его скважность, то есть отношение периода следования импульсов к длительности этих импульсов. В таких случаях чаще всего требуется *меандр*, то есть цифровой сигнал со скважностью, равной двум (длительность импульсов равна длительности паузы между ними). Получить меандр из любого сигнала довольно просто: надо использовать дополнительный делитель частоты на 2, правда при этом частота выходного сигнала уменьшится еще вдвое.

Простейший пример такого делителя частоты на десять приведен на рис. 5.6. В делителе использован счетчик ИЕ2, у которого одноразрядный внутренний счетчик включен после трехразрядного внутреннего счетчика. Трехразрядный счетчик делит частоту входного сигнала на 5, но выходные импульсы имеют скважность, не равную двум (она равна 5). Одноразрядный счетчик делит частоту еще вдвое и одновременно формирует меандр. Задержки переключения разрядов счетчика относительно друг друга на рисунке не показаны (применяем первый уровень представления, логическую модель).

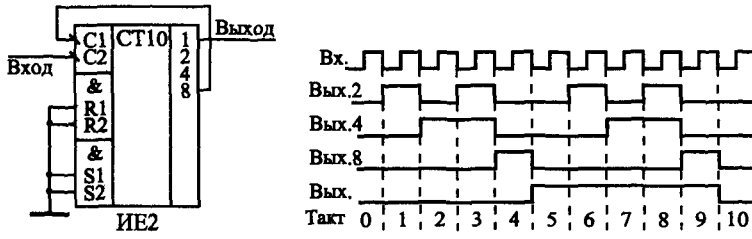


Рис. 5.6. Делитель частоты на 10, выдающий меандр.

Иногда возникает задача деления частоты входного сигнала в произвольное число раз (не в 10 и не в 2^n, что легко обеспечивается самой структурой стандартных счетчиков). В этом случае можно организовать сброс счетчика при достижении им требуемого кода путем введения обратных связей.

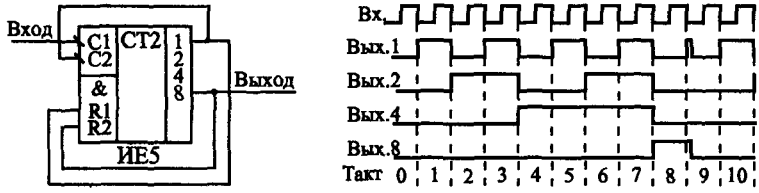


Рис. 5.7. Делитель частоты на 9 с обратными связями.

Например, на рис. 5.7 показан простейший делитель частоты на 9 на основе счетчика ИЕ5. При достижении его выходным кодом значения 9 (то есть 1001) счетчик автоматически сбрасывается в нуль по входам R1 и R2, и счет начинается снова. В результате частота выходного сигнала в 9 раз меньше частоты входного сигнала. При этом скважность выходного сигнала не равна двум. Временная диаграмма показана на рисунке для первого уровня представления (без учета временных задержек).

Если в числе, на которое надо делить частоту больше двух единиц (например, 15, то есть 1111, или 13, то есть 1101), то для формирования сигнала сброса надо использовать элементы 2И, 3И или 4И для объединения всех выходов, равных единице. В результате можно построить делитель входной частоты в любое число раз от 2 до 2^N, где N — число разрядов используемого счетчика. Правда, при организации обратных связей надо учи-

тывать ограничение на быстродействие счетчика. Все разряды, используемые для обратной связи, должны успеть переключиться за один период входного сигнала. Скважность выходного сигнала может принимать в данном случае самые разные значения, например, выходной сигнал может представлять собой очень короткие импульсы.

На асинхронных счетчиках можно строить также управляемые делители частоты, то есть такие делители, выходная частота которых определяется управляющим кодом. На рис. 5.8 показан делитель на 2^n, где n — целое. Восьмиразрядный счетчик ИЕ19 работает по входному сигналу с тактовой частотой f_T, а выходной 8-входовой мультиплексор КП7 передает на выход схемы один из 7 разрядов счетчика или же входной сигнал. Выбор номера канала производится входным управляющим 3-разрядным кодом. Например, при тактовой частоте f_T = 10 МГц, то есть при периоде входного сигнала 100 нс период выходного сигнала может составлять 100 нс, 200 нс, 400 нс, 800 нс, 1,6 мкс, 3,2 мкс, 6,4 мкс, 12,8 мкс.

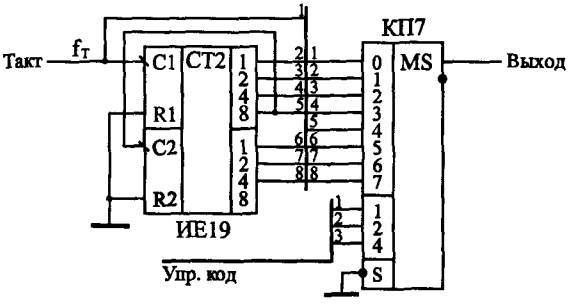


Рис. 5.8. Управляемый делитель частоты на асинхронном счетчике.

В момент переключения управляющего кода на выходе схемы могут появиться нежелательные короткие импульсы, так как никакой синхронизации управляющего кода не предусмотрено. Поэтому схема должна работать так: сначала задается входной управляющий код, а уже потом разрешается работа той схемы, на которую поступает сформированный нашей схемой выходной сигнал. В этом случае никаких проблем не будет. Не играют роли в данном случае и задержки переключения разрядов счетчика, так как всегда используется только один его разряд. Главное, чтобы с частотой f_T переключался первый разряд счетчика.

Конечно, применение асинхронных счетчиков не ограничивается только делителями частоты. В случаях, когда высокого быстродействия не требуется, когда переходные процессы на выходах счетчика не имеют значения (при правильной синхронизации), асинхронные счетчики вполне могут заменить более быстрые синхронные счетчики. Доля таких задач составляет около 20% от общего числа.

Если же включить на выходе асинхронного счетчика выходной параллельный регистр (рис. 5.9), то можно обеспечить одновременное переключение всех выходных разрядов счетчика.

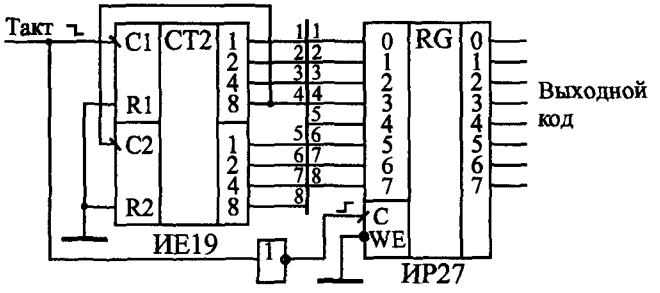


Рис. 5.9. Включение выходного регистра для одновременного переключения разрядов выходного кода.

Данная схема будет работать правильно, если период следования входных тактовых импульсов будет больше, чем время установления всех разрядов счетчика (в нашем случае — 8-разрядного счетчика ИЕ19). Инвертор необходим, так как счетчик срабатывает по отрицательному фронту входного сигнала, а регистр — по положительному фронту. Хотя данное решение устраняет главный недостаток асинхронного счетчика — неодновременность установления его выходных разрядов, однако второй недостаток — большая задержка установления выходного кода — сохраняется. Его устранить невозможно, можно только перейти на другие, более быстрые счетчики.

В заключение данного раздела надо отметить, что асинхронные счетчики, как и другие цифровые схемы, предъявляют требования к длительности входных сигналов. Например, не должны быть слишком короткими сигналы на тактовых входах и на входах сброса и установки. Не должны быть слишком затяну-

тыми фронты входных сигналов. Тактовые сигналы и сигналы сброса не должны приходить со слишком малыми задержками друг относительно друга.

5.2. Синхронные счетчики с асинхронным переносом

Синхронные (или параллельные) счетчики характеризуются тем, что все их разряды в пределах одной микросхемы переключаются одновременно, параллельно. Это достигается существенным усложнением внутренней структуры микросхемы по сравнению с простыми асинхронными счетчиками. В результате полная задержка переключения синхронного счетчика примерно равна задержке одного триггера, то есть синхронные счетчики гораздо быстрее асинхронных, причем их быстродействие не падает с ростом количества разрядов выходного кода (конечно, до определенных пределов).

Управление работой синхронного счетчика гораздо сложнее, чем в случае асинхронного счетчика, а количество разрядов синхронных счетчиков обычно не превышает четырех. Поэтому синхронные счетчики не всегда могут успешно конкурировать с асинхронными счетчиками, особенно при невысоких требованиях к быстродействию. Зато и возможностей у синхронных счетчиков, как правило, гораздо больше, чем у асинхронных, например, они обеспечивают параллельную запись информации в счетчик и инверсный режим счета.

Для объединения нескольких синхронных счетчиков с целью увеличения числа их разрядов (для каскадирования) используется специальный выходной сигнал переноса. В зависимости от принципов формирования этого сигнала переноса и от принципов его использования синхронные (параллельные) счетчики делятся на счетчики с асинхронным (последовательным) переносом и счетчики с синхронным (параллельным) переносом (или полностью синхронные счетчики).

Синхронные счетчики с асинхронным переносом занимают промежуточное положение по быстродействию между асинхронными счетчиками и полностью синхронными счетчиками. Управление их работой проще, чем у синхронных счетчиков, но сложнее, чем у асинхронных. Работают данные счетчики по положи-

тельному фронту входного сигнала (или, что то же самое, по заднему фронту отрицательного сигнала). Основная суть их работы сводится к следующему: все разряды одного счетчика переключаются одновременно, но при каскадировании счетчиков каждый следующий счетчик (дающий более старшие разряды) переключается с задержкой относительно предыдущего счетчика (дающего более младшие разряды). То есть задержка переключения многоразрядного счетчика увеличивается в данном случае не с каждым новым разрядом (как у асинхронных счетчиков), а с каждой новой (например, 4-разрядной) микросхемой.

Сигнал переноса у этих счетчиков при прямом счете вырабатывается тогда, когда все разряды равны единице (достигнут максимальный код) и когда приходит входной сигнал. Поэтому сигнал переноса, повторяющий входной сигнал, будет задержан относительно входного сигнала. И именно этот сигнал переноса используется в качестве входного для следующего счетчика при каскадировании. То есть входной сигнал второго счетчика задержан относительно входного сигнала первого счетчика, входной сигнал третьего счетчика задержан относительно входного сигнала второго счетчика и т. д.

Временная диаграмма 4-разрядного синхронного счетчика с асинхронным переносом показана на рис. 5.10. Из рисунка видно, что разряды переключаются одновременно по положительному фронту входного сигнала (с некоторой задержкой), а отрицательный сигнал переноса также задержан относительно входного отрицательного импульса. Понятно, что переключение разрядов счетчика, работающего с этим сигналом переноса в качестве входного, будет происходить с дополнительной задержкой относительно переключения разрядов данного счетчика.

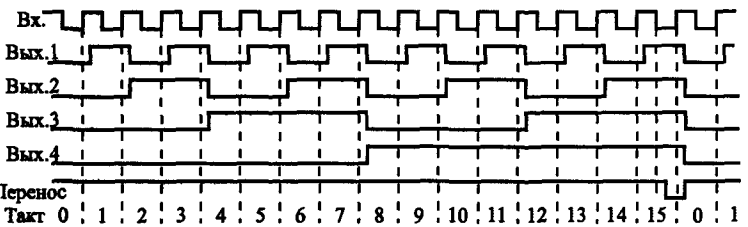


Рис. 5.10. Временная диаграмма работы синхронного счетчика с асинхронным переносом.

Примерами синхронных счетчиков с асинхронным переносом могут служить двоично-десятичный счетчик ИЕ6 и двоичный счетчик ИЕ7 (рис. 5.11). Они полностью идентичны по своим возможностям и назначениям входов и выходов, но только счетчик ИЕ6 считает от 0 до 9, а счетчик ИЕ7 — от 0 до 15. Оба счетчика реверсивные, обеспечивают как прямой счет (по положительному фронту на входе +1), так и обратный счет (по положительному фронту на входе –1). При прямом счете отрицательный сигнал переноса вырабатывается на выходе >15 (ИЕ7) или >9 (ИЕ6). При обратном (инверсном) счете отрицательный сигнал переноса вырабатывается на выходе <0 после достижения выходным кодом значения 0000. Имеется возможность сброса счетчика в нуль положительным сигналом на входе R, а также возможность параллельной записи в счетчик кода со входов D1, D2, D4, D8 по отрицательному сигналу на входе -WR. При параллельной записи информации счетчики ведут себя как регистры-защелки, то есть выходной код счетчика повторяет входной код, пока на входе -WR присутствует сигнал нулевого уровня.

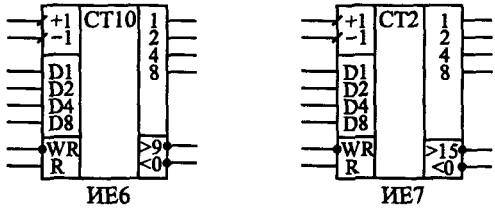


Рис. 5.11. Синхронные счетчики с асинхронным переносом.

Вход параллельной записи обозначается иногда на схемах также -L, -C, а выходы переноса обозначаются также -CR и -BR.

Режимы работы счетчиков ИЕ6 и ИЕ7 представлены в табл. 5.4.

После сброса счетчик начинает счет по положительным фронтам на счетных входах от нулевого кода. После параллельной записи счет начинается от числа, записанного в счетчик. После переполнения счетчика ИЕ7 (достижения кода 1111) при прямом счете вырабатывается отрицательный сигнал переноса >15, повторяющий входной отрицательный импульс на входе +1 с задержкой. После достижения кода 0000 при обратном счете выра-

батывается отрицательный сигнал переноса <0, повторяющий входной отрицательный импульс на входе –1 с задержкой. Точно так же работает и счетчик ИЕ6, но у него переполнение будет возникать в режиме прямого счета при достижении кода 1001.

Таблица 5.4. Таблица режимов работы счетчиков ИЕ6 и ИЕ7

Входы				Режим работы
R	-WR	+1	–1	
1	X	X	X	Сброс в нуль
0	0	X	X	Параллельная запись
0	1	1	1	Хранение
0	1	0	0	Хранение
0	1	0—>1	1	Прямой счет
0	1	1	0—>1	Обратный счет

Входные сигналы счета, записи и сброса не должны быть слишком короткими. Не должен быть слишком малым временной сдвиг между сигналами на входах D1 — D8 и сигналом записи как в начале импульса записи, так и в его конце (сигнал записи -WR должен начинаться после установления входного кода, а заканчиваться — до снятия входного кода).

Объединение счетчиков ИЕ7 и ИЕ6 для увеличения разрядности (каскадирование) осуществляется очень просто: нужно выходы переноса младших счетчиков (дающих младшие разряды выходного кода) соединить со счетными входами старших счетчиков (дающих старшие разряды выходного кода). На рис. 5.12 показана организация 12-разрядного счетчика на трех микросхемах ИЕ7. Этот счетчик может считать как на увеличение (прямой счет), так и на уменьшение (обратный счет). Возможны также сброс и параллельная запись в счетчики входного кода. Разряды каждого следующего счетчика будут переключаться одновременно, но с задержкой относительно переключения разрядов предыдущего счетчика. Точно так же объединяются и счетчики ИЕ6.

Если необходимо использовать все выходные разряды многоразрядного счетчика одновременно (как единый код), то необходимо выполнение следующего условия:

$$T > (N-1)t_{зп} + t_{зс},$$

где T — период входного сигнала, N — число объединенных микросхем счетчиков, $t_{зп}$ — время задержки переноса одного счетчика, $t_{зс}$ — время задержки счета (переключения выходного кода) одного счетчика.

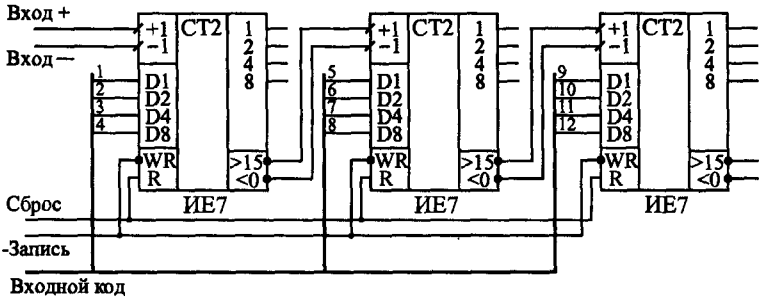


Рис. 5.12. Объединение счетчиков ИЕ7 для увеличения разрядности.

Применение синхронных счетчиков с асинхронным переносом очень многообразно. Например, они могут делить частоту входного сигнала, считать входные импульсы, формировать пачки импульсов, измерять длительность временного интервала, формировать сигналы заданной длительности, измерять частоту входных импульсов, последовательно переключать входные и выходные каналы, формировать сложные последовательности сигналов, перебирать адреса памяти и многое другое. Мы рассмотрим лишь несколько наиболее типичных примеров.

В качестве делителя частоты входного сигнала синхронные счетчики с асинхронным переносом очень удобны, так как в них сочетается сравнительно высокая скорость работы с довольно простым управлением. Удобно также и то, что у них имеется режим обратного счета. На этих счетчиках можно строить делители частоты с произвольно изменяемым с помощью входного кода коэффициентом деления. Такие делители находят, например, широкое применение в аналого-цифровых системах, работающих с аналоговыми сигналами разной частоты.

Простейший пример 8-разрядного делителя частоты на счетчиках ИЕ7 показан на рис. 5.13.

На вход счетчиков подается 12-разрядный управляющий код, определяющий коэффициент деления входной частоты. Этот код записывается в счетчики по сигналу переноса <0

старшего счетчика. С этого кода начинается затем счет на уменьшение. Когда счетчики отсчитают количество входных импульсов, равное входному коду, снова выработается сигнал переноса старшего счетчика и снова запишет входной код в счетчики. Коэффициент деления будет равен (N+1) при входном коде N. Отрицательный выходной сигнал будет по форме повторять входной, но с полной задержкой переноса, а его частота будет меньше частоты входного сигнала в (N+1) раз. При 12-разрядном входном коде максимальный коэффициент деления составит 4096, а минимальный — 1.

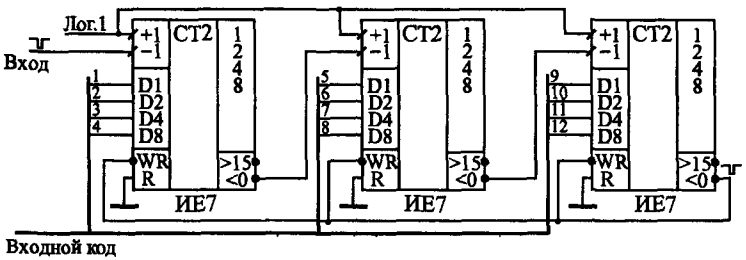


Рис. 5.13. Делитель частоты с коэффициентом деления, задаваемым входным кодом.

Чтобы сформулировать условия правильной работы данного делителя частоты, надо прежде всего отметить, что запись входного кода в счетчики производится отрицательным уровнем сигнала -WR, то есть передним фронтом входного отрицательного импульса, а счет производится положительным фронтом сигнала -1, то есть задним фронтом входного отрицательного импульса. Отсюда следует, что входной импульс должен быть достаточно коротким. Если он записывает код в счетчики своим передним фронтом, он уже не должен своим задним фронтом переключать счетчики по входу -1. Поэтому длительность входного отрицательного импульса не должна превышать полного времени переключения счетчиков и записи в них входного кода. В нашем случае это три задержки переноса и задержка записи в счетчик.

Если частота входного сигнала большая (например, больше 10 МГц), то нормальная длительность входного сигнала получается сама собой. Но частота входного сигнала не может быть и слишком большой. Иначе в процессе записи счетчик пропустит один из входных импульсов или даже несколько. То есть от переднего

фронта входного отрицательного сигнала до заднего фронта следующего входного отрицательного сигнала должны успеть сработать все счетчики и должна произойти запись в счетчики (суммарное время задержки опять же включит в себя сумму задержек переноса всех счетчиков и задержку записи). То есть ограничения на входную частоту будет тем жестче, чем больше счетчиков мы объединяем для увеличения количества разрядов. В данном случае важно именно количество примененных микросхем, а не количество используемых разрядов, как у асинхронных счетчиков.

Для решения часто встречающейся на практике задачи подсчета количества пришедших входных импульсов необходимо всего лишь объединить несколько микросхем счетчиков с целью получения требуемого числа разрядов. Например, если количество входных импульсов не превышает 255, то достаточно двух 4-разрядных счетчиков, если оно не больше 65535, то надо объединить уже четыре 4-разрядных счетчика. Так как в этом случае нас интересуют все выходные разряды одновременно, необходимо обеспечить, чтобы за период входных импульсов переключались все микросхемы счетчиков.

Обеспечить одновременность переключения всех выходных разрядов счетчика при счете входных импульсов можно, как и в случае асинхронных счетчиков, за счет включения выходного параллельного регистра, срабатывающего по фронту (рис. 5.14). Данное решение довольно универсально, оно может использоваться в самых разных ситуациях, когда необходим весь выходной код счетчика целиком. Код на выходе регистра будет удерживаться в течение всего периода входных импульсов. Правда, быстродействие счетчика от этого не повышается.

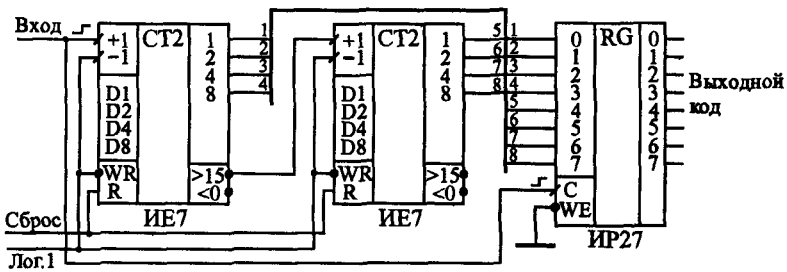


Рис. 5.14. Включение выходного регистра для одновременного переключения разрядов выходного кода.

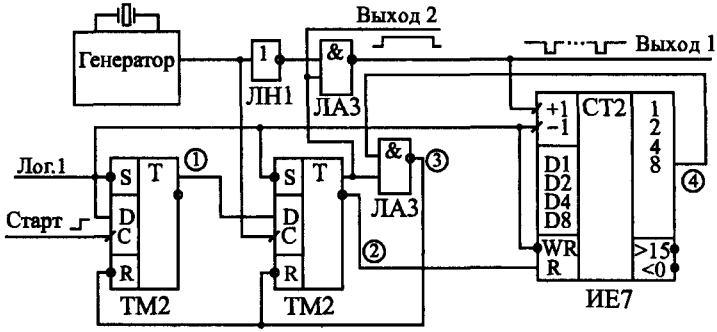


Рис. 5.15. Формирователь пачки из восьми импульсов.

Формирование пачки (группы) входных импульсов с заданным количеством импульсов — довольно распространенная задача. Например, такое формирование необходимо при организации обмена информацией в последовательном коде. Если в качестве преобразователя параллельного кода в последовательный используется 8-разрядный регистр сдвига, то ему в качестве синхросигнала необходима пачка из восьми импульсов. Схема формирователя такой пачки импульсов показана на рис. 5.15, а временная диаграмма ее работы — на рис. 5.16.

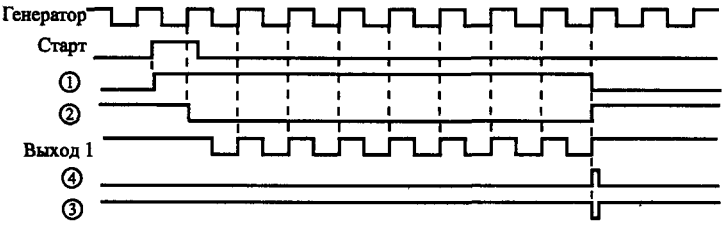


Рис. 5.16. Временная диаграмма работы формирователя пачки импульсов.

По сигналу Старт (положительный фронт) переключается первый триггер, использующийся для синхронизации. По первому положительному фронту тактового сигнала с генератора переключается второй триггер, разрешающий прохождение импульсов с генератора на выход через элемент 2И-НЕ, а также разрешающий работу счетчика ИЕ7.

После того как на Выход1 схемы пройдут восемь отрицательных импульсов, на выходе 8 счетчика выработается едини-

ца, что приведет к сбросу в исходное нулевое состояние обоих триггеров (коротким отрицательным импульсом на выходе нижнего по рисунку элемента 2И-НЕ) и к запрету прохождения импульсов на выход. Работа формирователя возобновится после следующего сигнала Старт.

На основе счетчиков довольно просто строить формирователи временных интервалов с длительностью, задаваемой внешним кодом. Такие формирователи находят широкое применение, например, в различных измерительных устройствах. Так как формирователь временных интервалов обычно работает с кварцевым тактовым генератором, возможны два подхода к его построению.

При первом подходе входной стартовый импульс синхронизируется с тактовым сигналом, в результате чего выходной импульс заданной длительности может начаться не сразу после стартового импульса, а через какое-то время, меньшее периода тактового сигнала. Длительность формируемого временного интервала будет в этом случае абсолютно точно известна и будет равна целому числу периодов тактового генератора. Именно так было сделано в предыдущей рассмотренной нами схеме (сигнал Выход 2 на рис. 5.15 как раз и будет формируемым сигналом с заданной длительностью).

При втором подходе выходной импульс заданной длительности начинается сразу после входного сигнала, но длительность его может отличаться от заданной на какое-то время, меньшее периода тактового сигнала. Иногда это более приемлемое решение, особенно при больших длительностях выходного сигнала, значительно больших, чем период тактового сигнала. Схема формирователя временного интервала, построенного в соответствии с этим вторым подходом, показана на рис. 5.17.

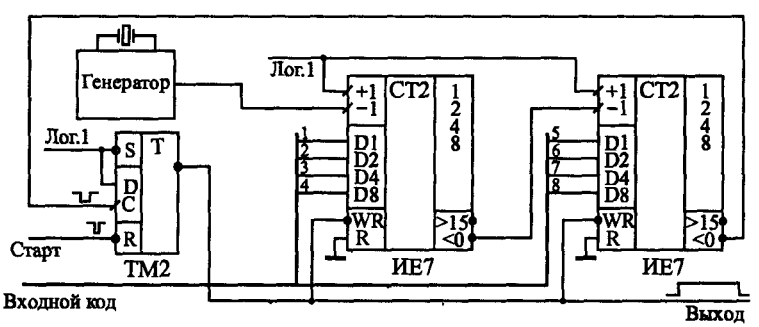


Рис. 5.17. Формирователь временного интервала.

Работа схемы начинается с подачи короткого отрицательного импульса -Старт. Он перебрасывает триггер, который разрешает работу счетчиков снятием сигнала параллельной записи -WR. По отрицательному фронту входного сигнала начинается положительный выходной сигнал заданной длительности. Счетчики начинают считать на уменьшение кода по положительным фронтам тактового сигнала с генератора. Когда они досчитают до нуля, вырабатывается сигнал переноса, перебрасывающий триггер в исходное состояние. Работа схемы возобновится после следующего сигнала -Старт.

Если входной код равен 1, то длительность выходного сигнала составит от T до 2T, где T — период тактового сигнала. Если входной код равен N (до 255), то длительность выходного сигнала составит от NT до (N+1)T в зависимости от момента прихода входного сигнала по отношению к тактовому сигналу. Абсолютная погрешность выдержки длительности выходного сигнала в любом случае не превышает периода тактового сигнала T.

Эту же самую схему вполне можно использовать в тех случаях, когда необходимо получить убывающий код от заданного числа до нуля. При этом сигнал с выхода триггера будет только внутренним сигналом схемы, а выходными сигналами схемы будут выходные разряды счетчиков.

Иногда бывает необходимо сформировать импульс требуемой длительности, но одновременно иметь не убывающий, а возрастающий код (от нуля до заданного значения). В таком случае схема получится несколько сложнее. Пример возможного решения формирователя импульса заданной длительности показан на рис. 5.18.

По сигналу Старт (положительный фронт) перебрасывается левый по рисунку триггер, который начинает формировать выходной сигнал и разрешает работу счетчика (снимая сигнал сброса R). Счетчик считает на увеличение по положительным фронтам тактового сигнала от нуля. Когда выходной код счетчика достигает величины входного кода, срабатывает правый по рисунку триггер, завершающий процесс формирования выходного сигнала. Счетчик сбрасывается в нуль, правый триггер по следующему фронту попадает в исходное состояние. Новый цикл начнется с приходом следующего сигнала Старт.

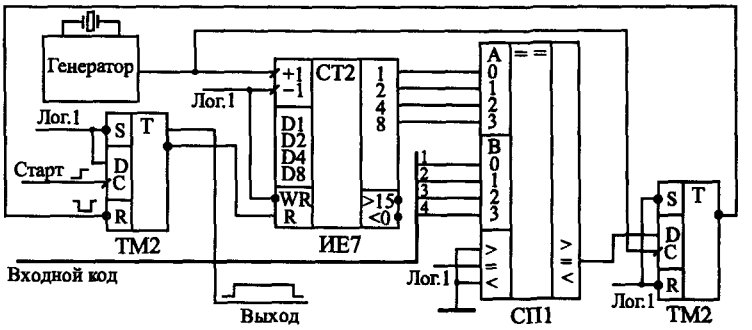


Рис. 5.18. Формирователь импульса заданной длительности (вариант с нарастающим кодом).

Если входной код равен 1, то длительность выходного сигнала составит от T до 2T, где T — период тактового сигнала генератора. Если входной код равен N, то длительность выходного сигнала будет равна от NT до (N+1)T в зависимости от временного сдвига между сигналом Старт и тактовым сигналом. В любом случае абсолютная погрешность времени выдержки выходного сигнала не превысит периода тактового сигнала T.

Счетчики также широко применяются в различных измерителях длительности входных сигналов. Для этого они отсчитывают импульсы тактового кварцевого генератора в течение длительности входного сигнала. После окончания входного сигнала в счетчике остается код, пропорциональный длительности этого сигнала. Пример практической схемы такого измерителя показан на рис. 5.19.

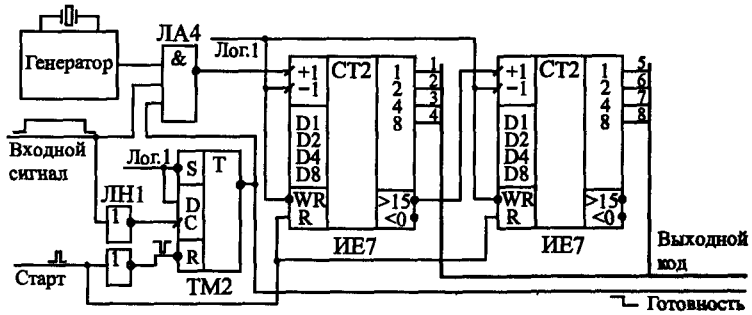


Рис. 5.19. Измеритель длительности входного сигнала.

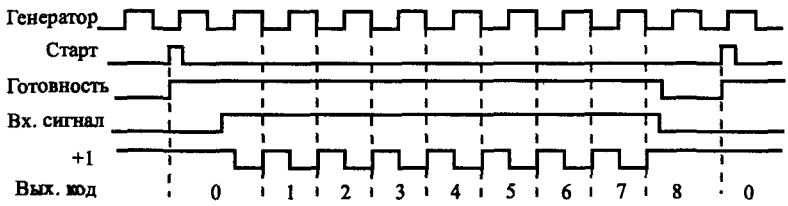


Рис. 5.20. Временная диаграмма работы измерителя длительности входного сигнала.

Работа схемы начинается по короткому управляющему импульсу Старт, который сбрасывает счетчик в нуль и переводит всю схему в режим счета, разрешая прохождение сигнала с тактового генератора на вход +1 счетчика при положительном входном сигнале. С началом входного сигнала импульсы с генератора поступают на вход счетчика, и счетчик их считает. После окончания входного сигнала поступление импульсов на вход счетчика прекращается, триггер перебрасывается в исходное состояние и сообщает отрицательным фронтом на своем инверсном выходе о готовности выходного кода (сигнал Готовность). Работа схемы возобновится по следующему импульсу Старт. Временная диаграмма работы измерителя длительности входного сигнала приведена на рис. 5.20.

Выходной код N измерителя связан с длительностью входного сигнала t простым соотношением:

$$t = NT,$$

где T — период тактового сигнала. Абсолютная погрешность измерения не превышает величины ±T. Поэтому для уменьшения относительной погрешности измерения необходимо увеличивать частоту тактового генератора и увеличивать разрядность счетчика.

Счетчики также применяются и для измерения частоты входного цифрового сигнала.

Частоту входного сигнала можно измерить двумя путями: косвенным, то есть измерением периода входного сигнала (по принципу, рассмотренному только что) и вычислением затем частоты (по формуле: $f_{вх} = 1/T_{вх}$) или же прямым измерением частоты. Первый метод требует вычислений с помощью компьютера или микроконтроллера, второй не требует никаких до-

полнительных вычислений. Поэтому мы рассмотрим здесь реализацию метода прямого измерения частоты.

В соответствии с этим методом необходимо сформировать временное окно с заданной длительностью t_o, в течение которого надо сосчитать количество N периодов входного сигнала T (рис. 5.21). В этом случае будет выполняться соотношение:

$$t_o = NT \text{ или } f = N/t_o,$$

где f — это частота входного сигнала, равная 1/T. То есть частота входного сигнала пропорциональна коду N, а коэффициент пропорциональности равен $1/t_o$. Если, например, выбрать t_o = 1 с, то код N будет равен частоте входного сигнала в герцах, а при t_o = 1 мс код N будет равен частоте входного сигнала в килогерцах.

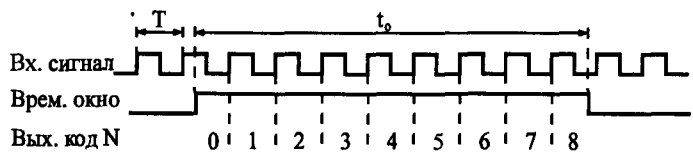


Рис. 5.21. Измерение частоты входного сигнала прямым методом.

Если длительность временного окна — строго постоянная величина, то погрешность измерения частоты будет определяться только погрешностью подсчета кода N. Абсолютная погрешность подсчета кода N не превысит единицы, а относительная погрешность не будет более 1/N. Понятно, что для увеличения точности измерения частоты надо увеличивать N, то есть необходимо увеличивать длительность временного окна t_o. Однако при этом автоматически увеличивается время измерения.

Схема измерителя частоты (рис. 5.22) практически не отличается от схемы измерителя длительности входного сигнала (рис. 5.19). Только в данном случае в качестве измеряемого сигнала будет использоваться сигнал временного окна, а в качестве тактового сигнала — входной сигнал. Для формирования сигнала временного окна можно применить схему рис. 5.15 (сигнал Выход 2), которая обеспечивает постоянную длительность выходного сигнала.

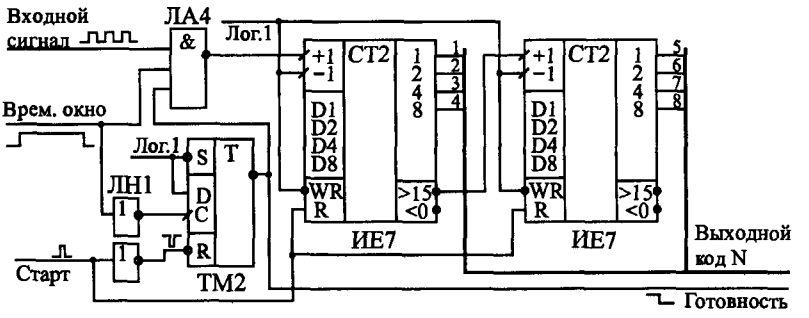


Рис. 5.22. Измеритель частоты входного сигнала прямым методом.

Еще одно широко распространенное применение счетчиков — последовательное переключение (сканирование) нескольких устройств, узлов, индикаторов, каналов передачи и т. д. Имеется, например, группа устройств, которые должны по тем или иным причинам работать не одновременно, а по очереди, так, что в каждый момент активным является только одно устройство, причем очередь эта замкнута в кольцо, и после последнего устройства начинает работать первое. Или же имеется несколько каналов связи (входных или выходных линий), которые надо так же по очереди подключать к одному выходу (при выходных каналах) или к одному входу (при входных каналах).

Во всех подобных случаях опрос, переключение, сканирование может производить счетчик с нужным числом разрядов. Счетчик с числом разрядов n может обслуживать 2^n устройств (или каналов).

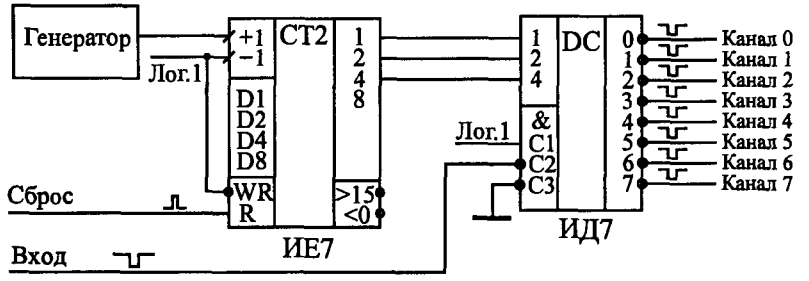


Рис. 5.23. Схема последовательного переключения выходных каналов.

В качестве первого примера рассмотрим схему переключения выходных каналов (рис. 5.23). Она последовательно, по очереди, циклически коммутирует один входной сигнал на восемь выходов, для чего используются счетчик, тактируемый сигналом задающего генератора, и дешифратор, работающий в качестве демультиплексора. Каждый из выходных каналов активен (то есть подключен) в течение одного периода тактового сигнала, а затем пассивен (то есть отключен) в течение семи периодов тактового сигнала. Предусмотрена возможность начального сброса схемы с помощью сигнала Сброс.

Используя данную схему, надо учитывать, что в момент переключения каналов может искажаться (обрезаться) выходной сигнал. Поэтому лучше всего обеспечить, чтобы входной сигнал приходил только тогда, когда переключения каналов не производится. Или на время передачи вообще останавливать процесс перебора каналов путем запрета прохождения импульсов с генератора на вход счетчика, а после окончания передачи снова разрешать последовательный перебор каналов.

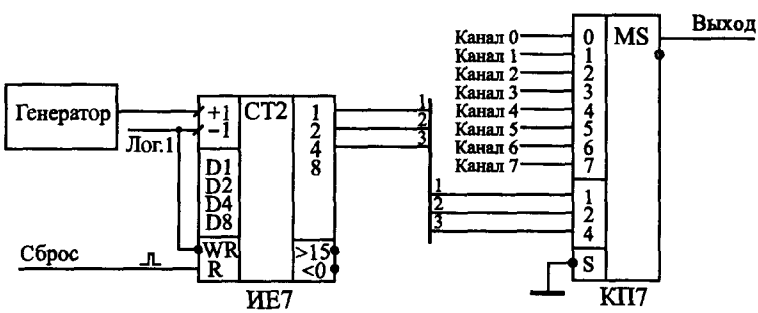


Рис. 5.24. Схема последовательного переключения входных каналов.

Второй пример, который мы рассмотрим, это схема, решающая обратную задачу — переключение входных каналов (рис. 5.24). Данная схема последовательно, циклически передает один из восьми входных сигналов на выход. Как и в предыдущем случае, перебор каналов осуществляется счетчиком, тактируемым сигналом с генератора. Непосредственно коммутация сигналов производится мультиплексором, на адресные входы которого подаются три разряда счетчика. Предусмотрена возможность начального сброса схемы с помощью сигнала Сброс.

В момент переключения каналов здесь также возможно искажение (обрезание) коммутируемых сигналов. Поэтому желательно обеспечить передачу сигналов в момент, когда переключения каналов нет. Или же надо останавливать процесс перебора каналов на время приема сигнала из выбранного канала путем запрета прохождения тактовых импульсов на вход счетчика, а затем снова запускать перебор каналов.

Еще одно применение счетчиков из этой же области состоит в организации так называемой динамической индикации.

Суть динамической индикации состоит в следующем. Если используется табло из нескольких индикаторов (одиночных светодиодов, светодиодных семисегментных индикаторов, светодиодных матричных индикаторов и т. д.), то совсем не обязательно, чтобы все эти индикаторы горели постоянно, одновременно. Можно зажигать их по очереди, что существенно сократит потребляемый всей схемой ток питания. Например, если в каждый момент времени горит только один индикатор из имеющихся восьми, то ток потребления индикаторов сократится в восемь раз. Учитывая, что каждый светящийся светодиод требует тока порядка 1—5 мА, такой подход может дать большой выигрыш, особенно в случае матричных индикаторов, содержащих несколько десятков светодиодов. А инерционность человеческого глаза приводит к тому, что вспышки света с частотой больше 20 Гц воспринимаются как непрерывное свечение. Так что при достаточной частоте перебора индикаторов глазу не будет заметно последовательное их включение.

На рис. 5.25 приведен пример схемы динамической индикации на восьми индикаторах. Для последовательного перебора индикаторов применяется счетчик, соединенный с дешифратором. Выходные сигналы дешифратора используются в качестве сигналов разрешения свечения для индикаторов. Частота сигнала тактового генератора, с которым работает счетчик, должна составлять не менее 160 Гц, чтобы каждый индикатор загорался не реже, чем с частотой 20 Гц. При этом нельзя также выбирать слишком большую частоту тактового генератора, так как в моменты переключения ток потребления микросхем сильно возрастает из-за паразитных емкостей, и при большой частоте весь эффект снижения потребления может сойти на нет.

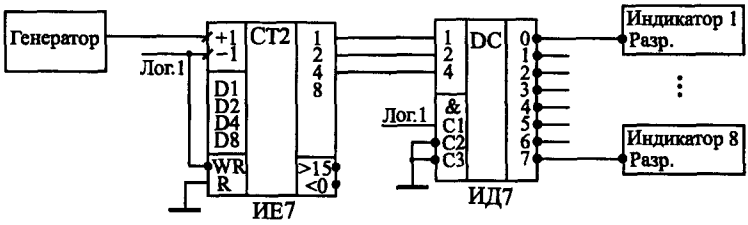


Рис. 5.25. Схема динамической индикации на восьми индикаторах.

Счетчики часто используют также для организации всевозможных таймеров, часов, то есть схем счета времени, выходной код которых необходимо время от времени читать. Для этого на вход счетчика подается сигнал образцовой частоты с кварцевого генератора. При этом возникает следующая проблема. Если чтение происходит в тот момент, когда счетчики переключаются, то с выходов счетчиков может быть считан случайный код, который не соответствует ни предыдущему установившемуся значению, ни последующему установившемуся значению. Можно, конечно, на время чтения кода остановить счет, но тогда ход часов собьется.

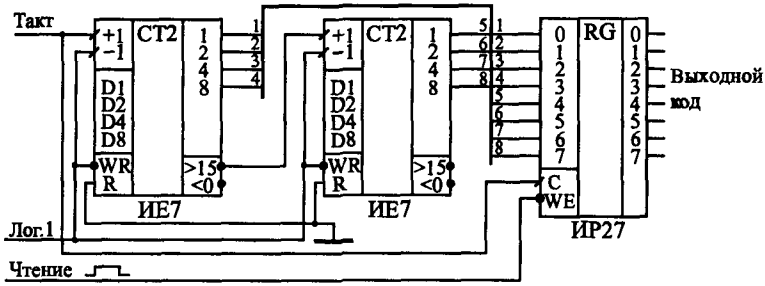


Рис. 5.26. Схема таймера с чтением выходного кода.

Пример решения данной проблемы приведен на рис. 5.26. Здесь выходной код счетчика на каждом такте записывается в выходной регистр с разрешением записи ИР27. А в момент чтения кода (при положительном сигнале Чтение) запись в регистр запрещается. В результате в течение всей длительности сигнала Чтение выходной код схемы будет неизменным, хотя счетчик будет продолжать считать без всяких помех, и ход часов не собьется.

Интересная особенность счетчиков ИЕ6 и ИЕ7 состоит в том, что они могут работать не только в режиме счета, но и в режиме повторителя входных сигналов данных. В режиме параллельной записи в счетчик при нулевом сигнале на входе -WR выходные сигналы счетчика будут повторять любые изменения входных сигналов данных, то есть счетчик работает по сути как регистр, срабатывающий по уровню стробирующего сигнала. В ряде случаев такая особенность очень удобна, так как она позволяет существенно упростить аппаратуру.

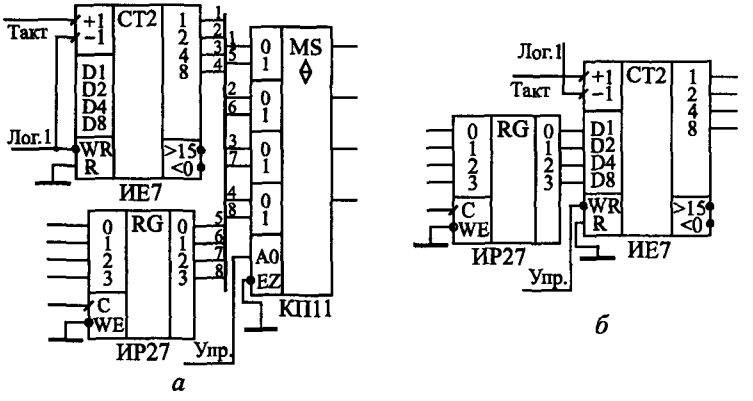


Рис. 5.27. Варианты мультиплексирования выходного кода счетчика с применением мультиплексора (*а*) и без него (*б*).

Пусть, например, необходимо выдавать на вход схемы один из двух входных кодов: код со счетчика, или код с регистра (то есть требуется мультиплексирование двух кодов). Эту задачу можно решить, применяя двухканальный мультиплексор (рис. 5.27,*а*), а можно решить проще — подавая код с регистра на входы данных счетчика и переводя в нужный момент счетчик в режим параллельной записи (рис. 5.27,*б*). В обоих случаях переключение кодов, подаваемых на выход схемы, производится сигналом Упр. Правда, во втором случае счетчик возобновляет свой счет (после снятия сигнала записи -WR) с кода, записанного в регистр. Если это неприемлемо, то можно воспользоваться входом сброса счетчика в нуль R.

И в заключение данного раздела мы рассмотрим две более сложные схемы на основе счетчиков. Это генератор прямоугольных импульсов с изменяемой частотой и длительностью импуль-

са и быстродействующий высокоточный измеритель частоты входного сигнала с большим диапазоном измеряемых частот.

Генерация прямоугольных импульсов — это довольно часто встречающаяся задача, в частности при разработке, отладке, тестировании электронной аппаратуры. От генератора прямоугольных импульсов требуется выдача импульсов заданной длительности при заданной паузе между импульсами (или, что то же самое, формирование импульсов заданной длительности и частоты следования). Желательно, чтобы диапазон изменения длительности импульсов и пауз между ними был как можно шире. Желательно также, чтобы был предусмотрен режим разового запуска (то есть остановка генерации после окончания одного выходного импульса) и автоматического запуска (то есть генерация периодической последовательности импульсов до прихода внешней команды остановки).

Предлагаемая здесь схема генератора не претендует, конечно, на рекордные характеристики, но она вполне может стать реальным удобным инструментом для разработчика цифровой аппаратуры, особенно если управление генератором поручить компьютеру с установленной на нем развитой сервисной управляющей программой. Благодаря своей простоте и наглядности схема эта может служить образцом для разработки более сложных генераторов импульсов, например, имеющих более высокое быстродействие, больший диапазон изменения длительности импульсов и их частоты, обеспечивающих генерацию импульсов с разной амплитудой и полярностью.

В основе генератора (рис. 5.28) — два 16-разрядных счетчика, выполненных на основе микросхем ИЕ7. Один из этих счетчиков (нижний на схеме) отсчитывает длительность выходного импульса, другой (верхний на схеме) — отсчитывает длительность паузы. Коды длительности импульса и паузы подаются соответственно на входы данных верхнего и нижнего счетчиков (эти коды могут храниться, например, в регистрах, не показанных на схеме). Счетчики импульса и паузы работают по очереди, что определяется управляющими сигналами на их входах параллельной записи -WR, которые также запрещают прохождение на входы -1 тактовых импульсов с помощью элементов 2И-НЕ. Эти управляющие сигналы поступают с прямого и инверсного выходов триггера ТМ2, на входы -R и -S которого подаются сигналы переноса с выходов <0 обоих счетчиков.

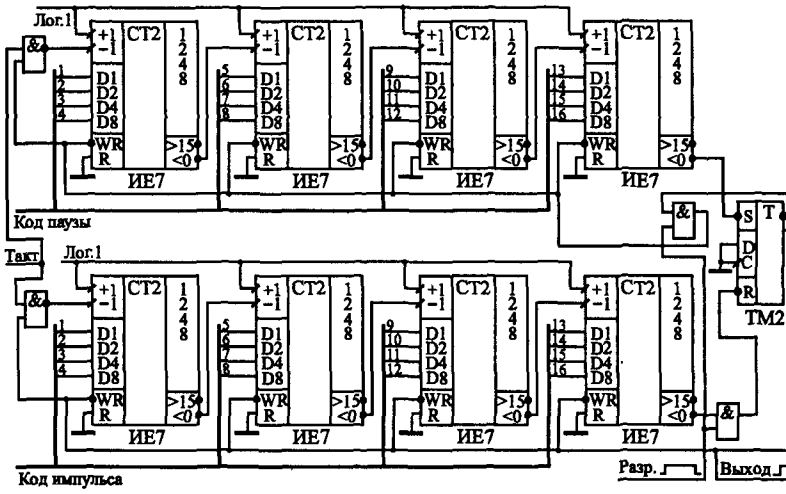


Рис. 5.28. Счетчики длительности импульса и паузы для генератора прямоугольных импульсов.

В результате, когда один счетчик считает, другой находится в режиме параллельной записи и не считает. После того как считающий счетчик досчитает до нуля, он перебросит выходной триггер, который переведет этот счетчик в состояние параллельной записи, запретит поступление на его вход тактовых импульсов и разрешит считать другому счетчику. Описанная последовательность действий повторится уже для другого счетчика. И этот процесс будет повторяться до тех пор, пока разрешена генерация.

В данном случае смело можно одновременно использовать как вход -R, так и вход -S триггера, так как сигналы, приходящие на них, гарантированно разнесены во времени. Сигнал с прямого выхода триггера служит выходным сигналом всего генератора в целом. Разрешается генерация положительным сигналом Разр. Когда генерация запрещена (нулевой сигнал Разр.) триггер сброшен в нуль по входу -R, и оба счетчика находятся в состоянии параллельной записи. Поэтому генератор всегда начинает работу с отработки паузы заданной длительности, а потом отрабатывает выходной импульс заданной длительности.

Сформулируем условия правильной работы данной схемы.

Во-первых, как и в случае управляемого делителя частоты (см. рис. 5.13), перевод счетчиков из режима счета в режим па-

раллельной записи осуществляется передним (отрицательным) фронтом тактового отрицательного импульса, а счет производится задним (положительным) фронтом отрицательного тактового импульса. Поэтому отрицательный тактовый импульс должен быть достаточно коротким. Один и тот же тактовый импульс не должен своим передним фронтом менять режим счетчиков, а задним фронтом переключать счетчики по входу -1. Длительность тактового отрицательного импульса не должна превышать полного времени переключения режимов счетчиков, включающего в себя четыре задержки переноса счетчиков, задержку переключения выходного триггера и задержку элементов 2И и 2И-НЕ.

Во-вторых, частота тактового сигнала не должна быть слишком большой, чтобы за время переключения режимов на вход -1 не пришел еще один положительный фронт тактового сигнала. Иначе этот фронт будет потерян. То есть от момента отрицательного фронта тактового импульса до момента положительного фронта следующего тактового импульса схема должна успеть полностью закончить переключение режимов счетчиков.

Пусть, например, мы хотим выбрать максимальную тактовую частоту 10 МГц (период T_T = 100 нс). Посмотрим, можно ли использовать микросхемы счетчиков серии КР1533. Для счетчиков КР1533ИЕ7 задержка сигнала переноса составляет не более 18 нс. Для четырех микросхем задержка переноса составит 72 нс. Тогда на сумму задержек триггера, элемента 2И и элемента 2И-НЕ остается не более 28 нс. Следовательно, если мы возьмем эти элементы из более быстрых серий (например, КР531 или КР1531), мы легко удовлетворим это требование.

При величине кода импульса N длительность импульса $T_И$ составит (N+1) • T_T. При величине кода паузы M длительность паузы $T_П$ составит (M+1) • T_T. Период выходных импульсов $T_{вых}$ будет равен (M+N+2) • T_T. Коды M и N могут принимать значения от 0 до 65535. То есть минимальная длительность импульса и паузы равна T_T, максимальная длительность импульса и паузы равна 65536 T_T, минимальная длительность периода выходного сигнала равна $2T_T$, а максимальная — 131072 T_T. Например, при тактовой частоте 10 МГц максимальный период выходного сигнала будет равен 13,1072 мс, а минимальный — 200 нс.

Для расширения диапазона изменения периода выходного сигнала можно применить управляемый делитель тактовой частоты. Другой возможный путь — наращивание разрядности счетчиков — приводит к снижению максимально допустимой тактовой частоты, так как обязательно вызывает увеличение задержек переключения счетчиков. К тому же, как правило, нет необходимости задавать длительность периода выходного сигнала, скажем, в 1 секунду с абсолютной погрешностью 100 нс (относительная погрешность — 10^{-7}). Гораздо важнее обеспечить стабильность частоты и периода выходного сигнала. Поэтому применение управляемого делителя частоты тактового сигнала не ухудшает характеристик генератора. Схема управления генератором прямоугольных импульсов с делителем частоты показана на рис. 5.29.

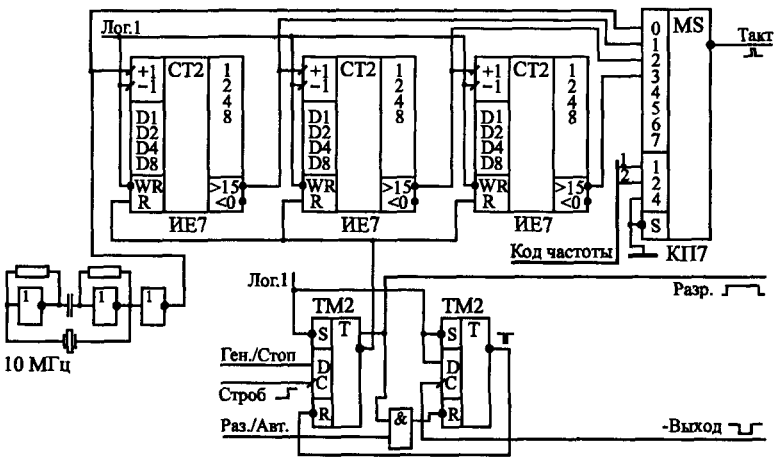


Рис. 5.29. Схема управления и делитель частоты для генератора прямоугольных импульсов.

Делитель частоты работает с кварцевым генератором с частотой 10 МГц и включает в себя три делителя на 16 на счетчиках ИЕ7. На выход мультиплексора (сигнал Такт) проходит один из сигналов с периодом 100 нс, 1,6 мкс, 25,6 мкс, 409,6 мкс. Длительность сигнала Такт не превышает половины периода сигнала с частотой 10 МГц, то есть 50 нс, что обеспечивает правильную работу счетчиков импульса и паузы (см. рис. 5.28). Выбор тактовой частоты осуществляется 2-разрядным кодом частоты. При за-

прете генерации все счетчики сбрасываются в нуль, это увеличивает точность привязки момента начала генерации к моменту подачи команды на начало генерации.

Схема управления генератором прямоугольных импульсов, также показанная на рис. 5.29, включает в себя два триггера ТМ2 и логический элемент 2И (ЛИ1).

Левый по рисунку триггер вырабатывает сигнал разрешения генерации Разр. В этот триггер необходимо записать единицу для разрешения генерации или нуль для остановки генерации. Запись в триггер входного сигнала Ген./Стоп производится передним фронтом сигнала Строб.

Правый по рисунку триггер служит для организации разового запуска генератора. Переключение режима разового или автоматического запуска производится управляющим сигналом Раз./-Авт. При автоматическом запуске (нуль на входе Раз./-Авт.) данный триггер не работает, он всегда находится в нулевом состоянии и дает уровень логической единицы на своем инверсном выходе. При разовом запуске (единица на входе Раз./-Авт.) правый триггер переходит в рабочий режим сразу после начала генерации (положительный сигнал Разр.). После окончания генерации первого выходного импульса на инверсном выходе генератора (инверсный выход триггера на рис. 5.28) появляется положительный перепад, который перебрасывает правый триггер на рис. 5.29. В результате он своим выходным сигналом сбрасывает левый триггер, что приводит к остановке генерации (так как сигнал Разр. становится нулевым). После этого схема снова готова к разовому запуску генерации. Временные диаграммы работы схемы в режимах автоматического и разового запуска показаны на рис. 5.30.

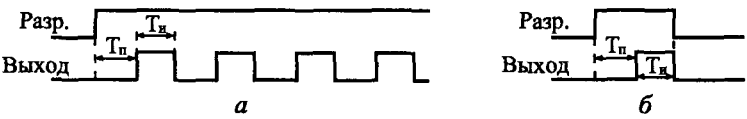


Рис. 5.30. Режимы работы генератора импульсов: автоматический (*а*) и разовый (*б*).

Асинхронность (независимость) момента прихода команды на начало передачи и сигнала задающего кварцевого генератора приводит к тому, что длительность первой паузы может ока-

заться на 100 нс меньше, чем она задана кодом паузы. Но это не слишком существенно, так как гораздо важнее длительность выходного импульса. Все последующие импульсы и паузы выдерживаются точно.

Абсолютная погрешность установки длительностей импульса $T_И$ и паузы и $T_П$ составляет половину периода тактового сигнала $T_Т$. Относительная погрешность установки этих величин составляет соответственно 0,5/N и 0,5/M. Понятно, что при малых величинах N и M погрешность будет большой (в пределе — даже 50%). Но при больших величинах длительностей импульса и паузы относительная погрешность не превышает 0,5/4096, то есть 0,012%.

Таким образом, рассмотренный генератор может формировать импульсы длительностью от 100 нс с паузой между импульсами от 100 нс. Максимально возможная длительность импульса составляет $2^{16} \cdot 2^{12} \cdot 100$ нс = 26,84 с. Такой же может быть и пауза. Правда отношение длительности импульса к длительности паузы (или длительности паузы к длительности импульса) не может превышать 65536. Величина периода выходного сигнала генератора может достигать 53,69 с.

Теперь рассмотрим вторую схему.

Задача измерения частоты следования входных прямоугольных импульсов также часто встречается как в чисто цифровых системах, так и в аналого-цифровых системах. Как уже упоминалось, существует два традиционных метода измерения частоты (рис. 5.31): один метод предполагает измерение периода $T_{ВХ}$ путем подсчета тактовых импульсов с периодом $T_Т$ в течение $T_{ВХ}$ и дальнейшее вычисление частоты по формуле: $f_{ВХ} = 1/T_{ВХ}$ (*а*), а другой метод прямо измеряет частоту $f_{ВХ}$ путем подсчета входных импульсов в течение временного окна $t_О$ (*б*).

Относительная погрешность и того и другого метода не превышает величины 1/N, где N — полученный в результате подсчета код. Понятно, что первый метод дает хорошую точность только для низких частот $f_{ВХ}$ (то есть для больших $T_{ВХ}$ и соответственно больших N). Второй метод дает хорошую точность только для больших частот $f_{ВХ}$ или в случае большого временного окна $t_О$ (то есть для больших N). В первом случае для увеличения точности необходимо увеличивать тактовую частоту, во втором случае — увеличивать длительность временного окна.

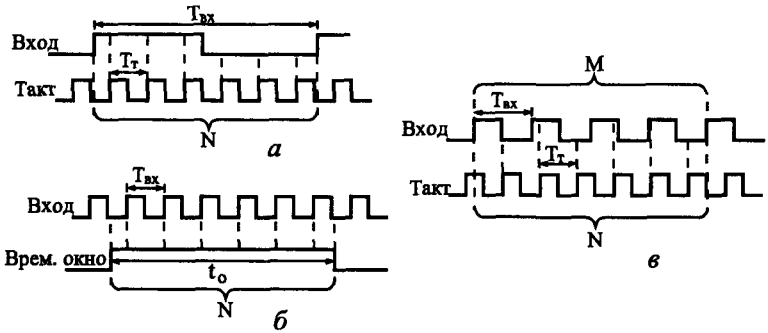


Рис. 5.31. Методы измерения частоты: через период (*а*), прямой (*б*) и комбинированный (*в*).

Время измерения частоты по первому методу составляет $T_{ВХ}$. Для второго метода оно постоянно и равно длительности временного окна t_O.

Поэтому желательно было бы соединить достоинства обоих методов, чтобы частота $f_{ВХ}$ измерялась бы достаточно быстро и с заданной точностью (с погрешностью, не меньшей заданной). Это возможно при использовании комбинированного метода (рис. 5.31,*в*). При данном методе импульсы тактовой частоты с периодом T_T подсчитываются в течение M полных периодов входного сигнала. При этом количество сосчитанных импульсов N определяет точность измерения (относительная погрешность не превышает 1/N). Значит, необходимо обеспечить, чтобы N было достаточно большим, например, при N>100 относительная погрешность не превысит 1%, а при N>1000 она будет меньше 0,1%. Обеспечить достаточную величину N можно простым выбором числа M.

Недостаток данного комбинированного метода состоит в том, что измеренное значение частоты необходимо вычислять. Так как при этом методе выполняется равенство: $MT_{ВХ} = NT_T$, следовательно, $f_{ВХ} = M/(NT_T)$. Однако при использовании компьютера или микроконтроллера такое вычисление не представляет особого труда. Зато данный комбинированный метод позволяет измерять частоту входного сигнала в широком диапазоне быстро и с заданной точностью. Поэтому мы подробно рассмотрим практическую реализацию именно этого метода.

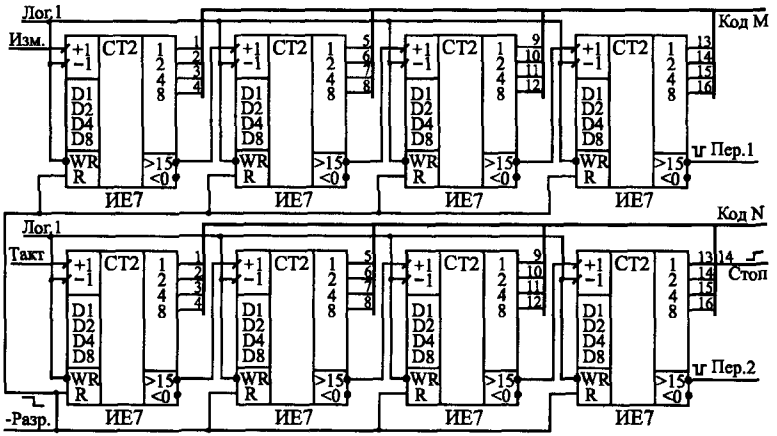


Рис. 5.32. Счетчики измерителя частоты входного сигнала.

В основе схемы измерителя частоты по комбинированному методу (рис. 5.32) — два 16-разрядных счетчика на основе микросхем ИЕ7, одновременно работающих в режиме прямого счета. На тактовый вход одного счетчика (верхнего на схеме) подается измеряемый сигнал Изм., на тактовый вход второго (нижнего на схеме) счетчика — тактовый сигнал образцовой частоты Такт. Выходные коды обоих счетчиков (соответственно M и N) используются после окончания измерения для вычисления значения частоты входного сигнала.

Работа счетчиков разрешается отрицательным сигналом -Разр. по фронту (например, положительному) входного сигнала. После окончания измерения по такому же фронту входного сигнала поступление сигналов Изм. и Такт запрещается. То есть счет производится в течение целого числа периодов входного сигнала.

Выход Стоп (положительный фронт) говорит о том, что код N достиг достаточной величины (в нашем случае — 8192), и, следовательно, можно останавливать измерение (но только по ближайшему фронту входного сигнала). То есть код N в конце измерения будет не менее 8192, и поэтому погрешность измерения частоты входного сигнала не превысит 1/8192 или 0,012%.

Для правильной работы схемы частота входного сигнала должна быть не более тактовой частоты $f_T = 1/T_T$ и не менее $f_T/65536$. Если она будет слишком малой, то наступит переполнение нижнего счетчика (выработается сигнал переноса -Пер.2).

Если же она будет слишком большой, то наступит переполнение верхнего счетчика (выработается сигнал переноса -Пер.1). Например, при тактовой частоте 10 МГц измеряемая частота входного сигнала может находиться в пределах от 152,6 Гц до 10 МГц.

Полное время измерения будет изменяться в пределах от $8192T_T$ до $(8192T_T + 2T_{BX})$. Один период T_{BX} может прибавляться к времени измерения из-за того, что после разрешения измерения счет начинается не сразу, а только с приходом фронта входного сигнала. Второй период T_{BX} может прибавляться за счет того, что счет заканчивается не сразу после достижения кодом N величины 8192, а только с приходом нужного (положительного) фронта входного сигнала. Максимальное время измерения в любом случае не превышает $65536T_T$ для всех измеряемых частот.

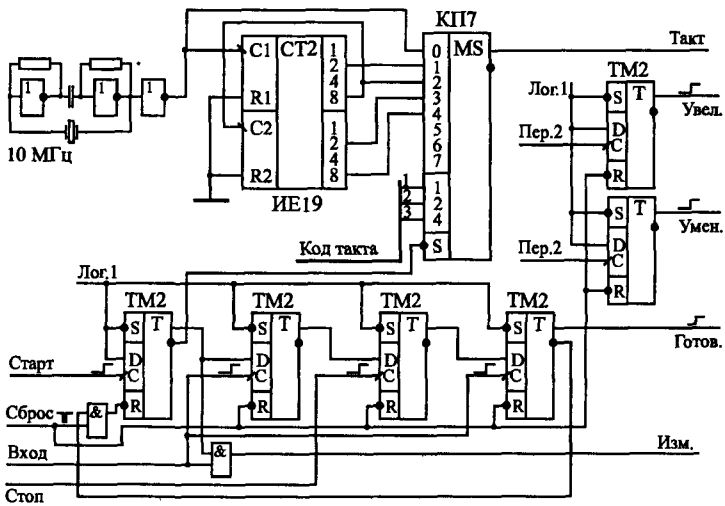


Рис. 5.33. Делитель частоты и схема управления для измерителя частоты входного сигнала.

Для увеличения диапазона измеряемых частот можно применить предварительный управляемый делитель частоты (рис. 5.33). Он обеспечивает выбор период тактового сигнала из ряда: 100 нс, 400 нс, 1,6 мкс, 6,4 мкс и 25,6 мкс с помощью кода такта. В результате применения этого делителя при минимальной тактовой частоте возможно измерение частоты входного сигнала до 0,6 Гц. Естественно, переход на каждый следующий диапазон измеряе-

мых частот может увеличить время измерения в 4 раза, но точность измерения в любом случае останется прежней.

Схема управления измерителем частоты, также показанная на рис. 5.33, включает в себя цепочку из четырех последовательно срабатывающих триггеров (ТМ2). Перед началом измерения все эти триггеры сбрасываются в нуль сигналом -Сброс.

Первый триггер перебрасывается в единицу по сигналу начала измерения Старт (положительный фронт). При этом разрешается прохождение подсчитываемых импульсов Изм. и Такт на вход счетчиков рис. 5.32. Одновременно разрешается работа второго триггера.

Второй триггер перебрасывается в единицу по положительному фронту входного сигнала. Тем самым он с помощью сигнала со своего инверсного выхода разрешает работу счетчиков (сигнал -Разр.). Одновременно разрешается работа третьего триггера.

Третий триггер перебрасывается в единицу по сигналу Стоп (то есть при достижении кодом N числа 8192). Он разрешает работу четвертого триггера.

Наконец, четвертый триггер перебрасывается по положительному фронту входного сигнала и сигналом со своего инверсного выхода сбрасывает первый триггер. Поступление сигналов Изм. и Такт прекращается. Выходной сигнал четвертого триггера служит флагом готовности выходных кодов N и M, которые необходимо прочитать для дальнейшего вычисления частоты. Перед новым измерением надо подать сигнал Сброс.

Кроме четырех управляющих триггеров в схему управления введены еще два триггера (справа на рисунке), выходные сигналы которых служат флагами переполнения и показывают после окончания измерения, правильно ли сработал измеритель частоты. Перед началом измерения оба эти триггера сбрасываются по сигналу Сброс. Если частота входного сигнала в нужных пределах, то оба триггера останутся в нуле. Если частота входного сигнала очень большая, то сработает верхний по рисунку триггер по входному сигналу переноса Пер.1 (см. рис. 5.32) и выдаст сигнал Увел., говорящий о том, что надо поднять частоту тактового сигнала (если это возможно). Если же частота входного сигнала слишком мала, то сработает нижний по рисунку триггер по входному сигналу переноса Пер.2 (см. рис. 5.32) и выдаст сигнал Умен., говорящий о том, что надо уменьшить частоту тактового сигнала (если возможно).

5.3. Синхронные счетчики

Синхронные (или параллельные) счетчики представляют собой наиболее быстродействующую разновидность счетчиков. Наращивание их разрядности при соблюдении определенных условий не приводит к увеличению полной задержки срабатывания. То есть можно считать, что именно синхронные счетчики работают как идеальные счетчики, все разряды которых срабатывают одновременно, параллельно. Задержка срабатывания счетчика в этом случае примерно равна задержке срабатывания одного триггера. Достигается такое быстродействие существенным усложнением внутренней структуры микросхемы.

Вместе с тем недостатком синхронных счетчиков является более сложное управление их работой по сравнению с асинхронными счетчиками и с синхронными счетчиками с асинхронным переносом. Поэтому синхронные счетчики целесообразно применять только в тех случаях, когда действительно требуется очень высокое быстродействие, очень высокая скорость переключения разрядов. Иначе усложнение схемы управления может быть не оправдано.

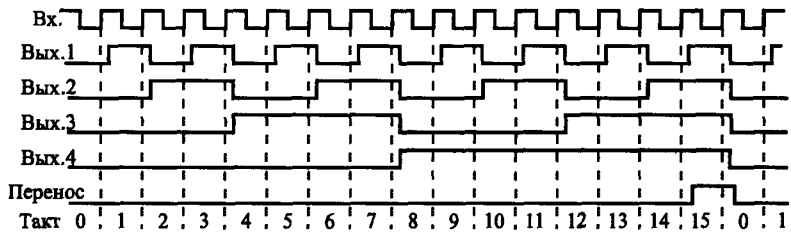


Рис. 5.34. Временная диаграмма работы синхронных двоичных счетчиков.

Временная диаграмма работы синхронного счетчика (рис. 5.34) отличается от временной диаграммы синхронного счетчика с асинхронным переносом способом формирования сигнала переноса, используемого при каскадировании счетчиков для увеличения разрядности. Сигнал переноса CR (от английского Carry) вырабатывается в данном случае тогда, когда все выходы счетчика устанавливаются в единицу (при прямом счете) или в нуль (при обратном, инверсном счете). Входной тактовый сигнал в образовании сигнала переноса при этом не участвует.

При каскадировании (совместном включении для увеличения разрядности), например, двух счетчиков тактовые входы С обоих счетчиков объединяются, а сигнал переноса первого счетчика подается на вход разрешения счета (ECT) второго счетчика. В результате второй счетчик будет считать каждый шестнадцатый входной тактовый импульс (так как он будет срабатывать только при переносе от первого счетчика). Выходные сигналы второго счетчика будут переключаться по фронту общего тактового сигнала одновременно с выходными сигналами первого счетчика. Условием правильной работы будет в данном случае следующее: за период тактового сигнала должен успеть выработаться сигнал переноса первого счетчика.

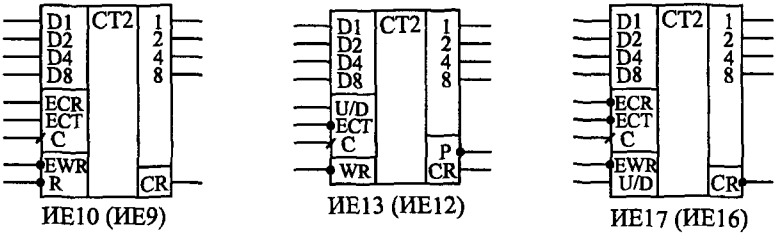


Рис. 5.35. Синхронные счетчики стандартных серий.

В стандартные серии микросхем входят несколько разновидностей синхронных (параллельных) счетчиков (рис. 5.35). Различаются они способом счета (двоичные или двоично-десятичные, реверсивные или не реверсивные), управляющими сигналами (наличием или отсутствием сигнала сброса). Все счетчики считают по положительному фронту тактового сигнала, все имеют выход переноса CR и входы расширения для каскадирования. Все счетчики имеют возможность параллельной записи информации.

Счетчики ИЕ9 и ИЕ10 отличаются друг от друга только тем, что ИЕ9 — двоично-десятичный, а ИЕ10 — двоичный. Микросхемы имеют вход асинхронного сброса -R, по нулевому уровню на котором все выходы счетчика сбрасываются в нуль. Счет (только прямой) производится по положительному фронту на тактовом входе С. Параллельная запись осуществляется синхронно, по положительному фронту на тактовом входе С при установленном в нуль сигнале разрешения записи -EWR. Сигна-

лы ECR (Enable Carry — разрешение переноса) и ECT (Enable Count — разрешение счета) используются при каскадировании микросхем. Разница между этими сигналами в том, что сигнал ECR не только запрещает счет, как сигнал ECT, но еще и запрещает выработку сигнала переноса CR. Счет идет при единичных сигналах на обоих входах ECT и ECT и при единичном сигнале на входе -EWR. Положительный сигнал переноса CR вырабатывается при максимально возможном коде на выходах счетчика (15 для ИЕ10 и 9 для ИЕ9) и при положительном сигнале на входе ECR. Режимы работы счетчиков ИЕ9 и ИЕ10 представлены в табл. 5.5.

Таблица 5.5. Режимы работы счетчиков ИЕ9 и ИЕ10

Входы					Режим
-R	-EWR	ECR	ECT	C	
0	X	X	X	X	Сброс
1	0	X	X	0—>1	Параллельная запись
1	1	0	X	X	Хранение
1	1	X	0	X	Хранение
1	1	1	1	0—>1	Прямой счет

Счетчики ИЕ12 (двоично-десятичный) и ИЕ13 (двоичный) отличаются от ИЕ9 и ИЕ10 тем, что они реверсивные, то есть допускают как прямой, так и обратный счет. Кроме того, у них несколько другое управление. Считают они также по положительному фронту тактового сигнала C при нулевом уровне на входе разрешения счета ECT. Прямой счет осуществляется при нулевом уровне на входе управления U/D, обратный — при единичном уровне на входе U/D. Переключение уровней на входах U/D и ECT допускается только при положительном сигнале на тактовом входе C. Сброс счетчиков ИЕ12 и ИЕ13 в нуль не предусмотрен, зато имеется возможность асинхронной параллельной записи информации по нулевому уровню сигнала параллельной записи -WR.

Положительный сигнал на выходе параллельного переноса CR появляется при достижении максимального кода (15 для ИЕ13 и 9 для ИЕ12) при прямом счете или при достижении нулевого кода при обратном (инверсном) счете. Имеется также

выход последовательного переноса P, отрицательный импульс на котором вырабатывается при положительном сигнале CR и повторяет отрицательный импульс на тактовом входе C (аналогично рассмотренным ранее счетчикам ИЕ6 и ИЕ7).

Режимы работы счетчиков ИЕ12 и ИЕ13 представлены в табл. 5.6.

Таблица 5.6. Режимы работы счетчиков ИЕ12 и ИЕ13

Входы				Режим
-WR	U/D	-ECR	C	
0	X	X	X	Параллельная запись
1	X	1	X	Хранение
1	0	0	0—>1	Прямой счет
1	1	0	0—>1	Обратный счет

Микросхемы ИЕ16 (двоично-десятичный счетчик) и ИЕ17 (двоичный счетчик) отличаются от рассмотренных синхронной параллельной записью по фронту тактового сигнала C, возможностью прямого и обратного счета и отсутствием сигнала сброса в нуль.

Срабатывают счетчики ИЕ16 и ИЕ17 по положительному фронту тактового сигнала C. При нулевом уровне на входе разрешения записи -EWR по фронту сигнала C в счетчик записывается информация со входов данных D1, D2, D4, D8. При единичном уровне на входе -EWR по положительному фронту сигнала C происходит счет. Направление счета определяется входом U/D: при единице на этом входе счет прямой, при нуле — обратный. Имеются два входа расширения: вход разрешения счета -ECT и вход разрешения переноса -ECR. Различаются эти два входа тем, что сигнал -ECR не только запрещает счет, как сигнал -ECT, но еще и запрещает выработку сигнала переноса. Переключение уровней на входах U/D, -ECT и -ECR надо производить только при единичном уровне на тактовом входе C.

Отрицательный сигнал переноса -CR (синхронный) вырабатывается при достижении на выходах счетчика максимального кода (15 для ИЕ7 или 9 для ИЕ16) при прямом счете или нулевого кода при обратном счете.

Режимы работы счетчиков ИЕ16 и ИЕ17 приведены в табл. 5.7.

Таблица 5.7. Режимы работы счетчиков ИЕ16 и ИЕ17

Входы					Режим
-EWR	U/D	-ECT	-ECR	C	
0	X	X	X	0—>1	Параллельная запись
1	1	0	0	0—>1	Прямой счет
1	0	0	0	0—>1	Обратный счет
1	X	1	X	X	Хранение
1	X	X	1	X	Хранение

Возможности применения синхронных (параллельных) счетчиков очень широки. Достаточно сказать, что они без всяких проблем могут заменить во всех схемах как асинхронные (последовательные) счетчики, так и синхронные счетчики с асинхронным (последовательным) переносом. При необходимости достижения максимального быстродействия они имеют большие преимущества по сравнению со всеми другими счетчиками. Их выходной код устанавливается одновременно при любом количестве разрядов без применения дополнительных выходных регистров (которые требовались в случае асинхронных счетчиков и синхронных счетчиков с асинхронным переносом).

Мы рассмотрим здесь всего несколько схем, иллюстрирующих характерные особенности именно синхронных счетчиков.

Сначала остановимся на методах каскадирования счетчиков. В отличие от других типов счетчиков синхронные счетчики можно соединять различными способами, причем способ соединения различен для разного количества микросхем. В качестве примера возьмем микросхемы ИЕ17.

При объединении двух счетчиков (рис. 5.36) никаких проблем не возникает: выход переноса -CR младшего счетчика соединяется со входом разрешения счета старшего счетчика -ECT. На входы -ECR обоих счетчиков подается нулевой уровень. Условие правильной работы будет простым и легко выполнимым: период тактового сигнала C не должен быть меньше, чем задержка выработки сигнала переноса CR.

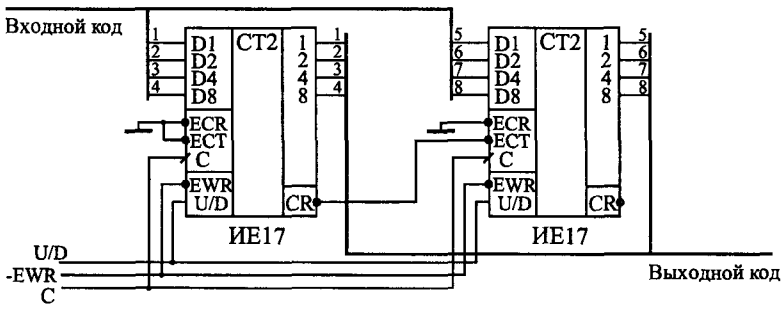


Рис. 5.36. Объединение двух счетчиков ИЕ17.

При объединении трех счетчиков ситуация несколько усложняется (рис. 5.37). Сигнал с выход переноса первого счетчика подается на входы -ECT второго и третьего счетчиков. Сигнал с выход переноса второго счетчика подается на вход -ECR третьего счетчика. В результате третий счетчик будет считать только тогда, когда имеется перенос как у первого счетчика, так и у второго счетчика. На рисунке для простоты не показано подключение входных и выходных сигналов, не участвующих в каскадировании.

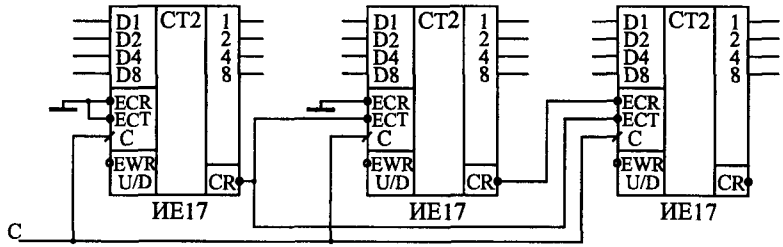


Рис. 5.37. Объединение трех счетчиков ИЕ17.

Условие правильной работы схемы остается тем же, что и в случае двух счетчиков: период тактового сигнала C не должен быть меньше задержки выработки сигнала переноса CR.

При объединении четырех (и более) счетчиков уже возникает проблема, так как у старших счетчиков не остается свободных управляющих входов для собирания всех сигналов переноса более младших счетчиков. Поэтому в данном случае используется способность входного сигнал -ECR запрещать выходной

сигнал переноса -CR (рис. 5.38). На четвертый и последующие счетчики подаются уже не сигналы переноса со всех предыдущих счетчиков, а только с первого и с предыдущего. На рисунке для простоты не показано подключение входов и выходов, не участвующих непосредственно в каскадировании.

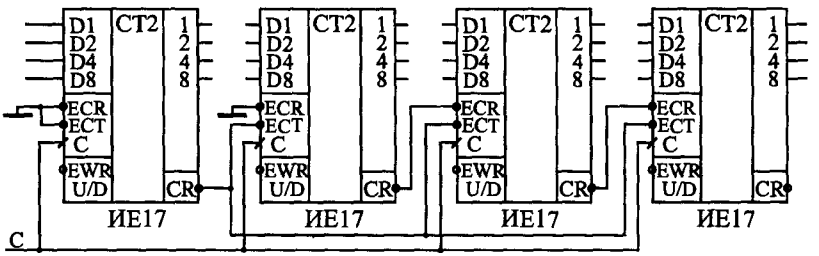


Рис. 5.38. Объединение четырех счетчиков ИЕ17.

При таком включении происходит уже накапливание задержек сигналов переноса. Максимальной задержка будет для сигнала переноса второго счетчика. Условие правильной работы всех счетчиков будет следующее: период тактового сигнала C не должен быть меньше, чем максимальная суммарная задержка сигналов переноса до входа последнего счетчика. При объединении четырех счетчиков в эту максимальную задержку входят задержка сигнала переноса -CR микросхемы относительно фронта сигнала C и задержка сигнала переноса -CR относительно сигнала -ECR. При объединении пяти счетчиков добавится еще одна задержка сигнала переноса -CR относительно сигнала -ECR и т. д. Поэтому с увеличением количества объединяемых счетчиков будет снижаться допустимая тактовая частота.

При необходимости объединения большого количества счетчиков (большего четырех) можно избежать накопления суммарной задержки переноса, включив на входах старших счетчиков -ECT логические элементы ИЛИ с нужным числом входов. Эти элементы должны собирать все сигналы переноса с более младших счетчиков, то есть на их выходах должен быть нуль тогда, когда сигналы -CR всех предыдущих счетчиков нулевые. При этом, правда, в суммарную задержку переноса, которая не должна превышать периода тактового сигнала C, войдут задержки этих самых элементов ИЛИ.

В любом случае при выполнении условия правильной работы счетчиков схема будет работать как идеальный счетчик, то есть все разряды многокаскадного счетчика будут переключаться одновременно.

А теперь рассмотрим некоторые схемы на основе синхронных счетчиков.

Управляемый делитель частоты с коэффициентом пересчета, задаваемым входным кодом, реализуется на синхронных счетчиках довольно просто (рис. 5.39). Сигнал переноса -CR старшего счетчика подается на вход разрешения записи -EWR. Счетчики работают в режиме обратного счета (на вход U/D подан сигнал логического нуля).

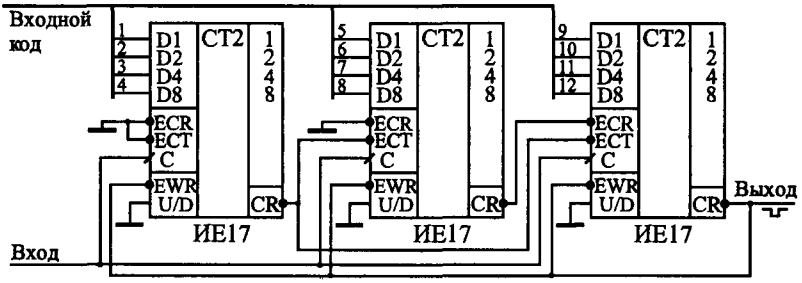


Рис. 5.39. Управляемый делитель частоты.

При достижении всеми счетчиками нулевого кода вырабатывается сигнал переноса -CR, переводящий счетчики в режим параллельной записи входного управляющего кода. Следующим положительным фронтом тактового сигнала C входной код записывается в счетчики. Это приводит к новому циклу счета от входного кода до нуля.

Коэффициент пересчета делителя частоты равен (N+1), где N — входной код, который может принимать значения от 1 до $(2^n - 1)$, где n — количество разрядов кода. Условие правильной работы делителя частоты следующее: период тактового сигнала не должен быть меньше полной задержки переноса. Длительность выходного сигнала делителя частоты равна периоду тактовой частоты.

Следующая схема — формирователь временного интервала заданной длительности (рис. 5.40) демонстрирует, как надо ис-

пользовать выходной сигнал переноса синхронных счетчиков при необходимости организации разового (не периодического) цикла работы.

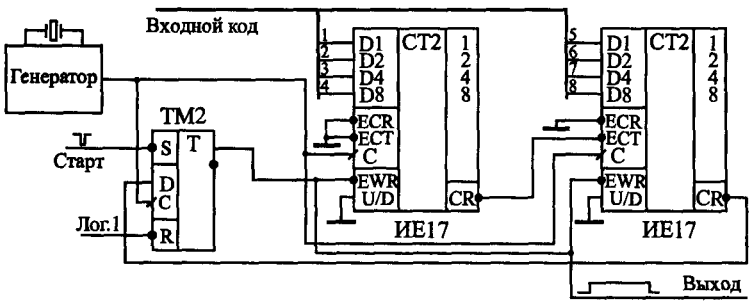


Рис. 5.40. Формирователь интервала заданной длительности.

Работа формирователя начинается по короткому отрицательному импульсу -Старт, перебрасывающему управляющий триггер в единицу и начинающему выходной сигнал. Положительный сигнал с выхода триггера переводит 8-разрядный синхронный счетчик из режима параллельной записи входного кода в режим счета (по входу -EWR). Счет на уменьшение идет по положительным фронтам тактового сигнала с генератора. Когда счетчик досчитает до нуля, следующим положительным фронтом тактового сигнала нулевой сигнал переноса -CR будет записан в триггер. Тем самым будет завершен выходной сигнал, а счетчик будет переведен в режим параллельной записи. Следующий цикл работы формирователя начнется по сигналу -Старт.

В данном случае триггер, обрабатывающий сигнал переноса работает синхронно со счетчиками, так как тактируется тем же (положительным) фронтом единого тактового сигнала. Длительность выходного сигнала будет находиться в интервале от NT до (N+1)T, где T — период тактового сигнала с генератора, а N — входной код от 0 до 255.

Посмотрим, как на синхронных счетчиках можно построить генератор прямоугольных импульсов с регулируемыми длительностью импульса и длительностью паузы, который был рассмотрен в предыдущем разделе (см. рис. 5.28 и 5.29). Будем ориентироваться на достижение максимального быстродействия, то есть на максимально возможную тактовую частоту.

Схема управления будет мало отличаться от схемы рис. 5.29, поэтому мы остановимся только на схеме счетчиков импульса и паузы. Выберем разрядность обоих этих счетчиков равной 16. Тогда схема счетчиков импульса и паузы (рис. 5.41) будет включать в себя восемь микросхем счетчиков ИЕ17 и выходной триггер, а также логические элементы 4ИЛИ-НЕ для уменьшения задержек переноса. В данном случае очень удобно брать JK-триггер, так как он имеет два информационных входа и тактовый вход.

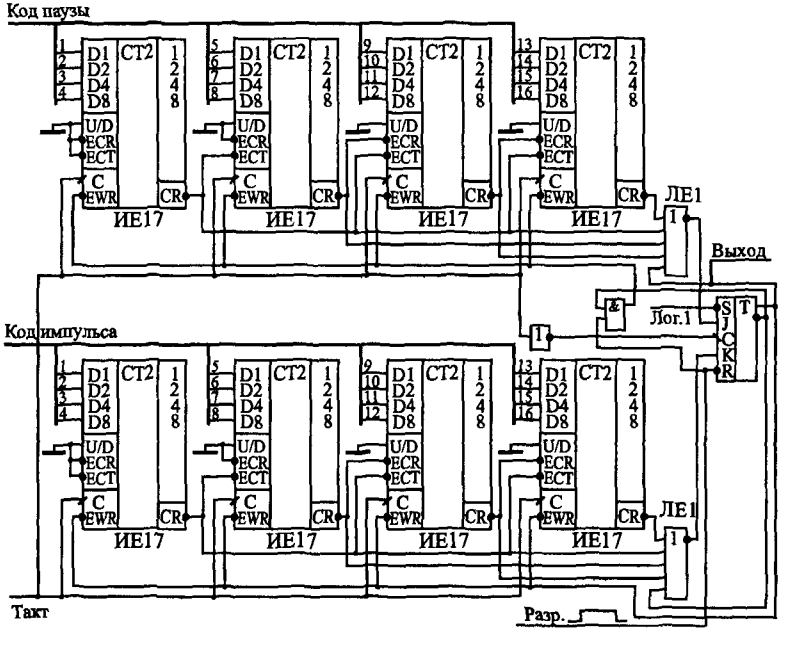


Рис. 5.41. Синхронные счетчики импульса и паузы для генератора прямоугольных импульсов.

Триггер тактируется отрицательным фронтом сигнала C, а счетчики — положительным фронтом, поэтому для обеспечения синхронной работы всей схемы по одному фронту тактового сигнала сигнал на вход C триггера подается через инвертор.

Суть работы схемы остается прежней: 16-разрядные счетчики импульса и паузы работают по очереди, что определяется управляющими сигналами с выходов триггера (прямого и ин-

версного). Счетчики считают на уменьшение (в режиме инверсного счета) от кода, параллельно записанного в них, до нуля.

До начала работы (сигнал Разр. нулевой) оба счетчика находятся в состоянии параллельной записи и записывают в себя код импульса и паузы. После прихода положительного сигнала разрешения генерации Разр. начинает счет верхний на схеме счетчик (счетчик паузы).

Когда счетчик паузы досчитывает до нуля, его сигнал переноса записывается в триггер по входу J и перебрасывает выход триггера в единицу, что переводит счетчик паузы из состояния счета в состояние параллельной записи и запрещает поступления сигнала на вход J. Одновременно переходит в состояние счета нижний на схеме счетчик (счетчик импульса), который, в свою очередь досчитав до нуля, перебрасывает триггер в нуль по входу К. Этот процесс периодически повторяется, пока разрешена генерация (то есть сигнал Разр. положительный).

Сформулируем условия правильной работы схемы.

Во-первых, за период тактового сигнала должен успеть полностью сработать 16-разрядный счетчик, выполненный на четырех микросхемах синхронных счетчиков. То есть сигнал на входы -ECR и -ECT последнего счетчика должны успеть придти до следующего фронта тактового сигнала.

Во-вторых, за период тактового сигнала должна успеть сработать цепочка из инвертора (ЛН1), триггера (ТВ11) и элемента 2И (ЛИ1). Это более мягкое требование, чем предыдущее, если, конечно, взять перечисленные элементы из быстродействующих серий КР531 или КР1531.

Рассмотренный переход на синхронные счетчики позволяет повысить максимальную частоту тактового сигнала генератора прямоугольных импульсов по меньшей мере вдвое (до 20 МГц) по сравнению со схемой на синхронных счетчиках с асинхронным переносом.

Наконец, последнее применение синхронных счетчиков, которое мы рассмотрим, связано с их возможностью параллельной записи по фронту тактового сигнала. То есть в режиме параллельной записи счетчик представляет собой регистр, срабатывающий по фронту тактового сигнала. Благодаря этой особенности, при объединении нескольких счетчиков их выходные коды можно последовательно считывать с выходов по-

следнего в цепочке, старшего счетчика (рис. 5.42). Счетчики в данном случае образуют своеобразный многоразрядный сдвиговый регистр.

Режим работы схемы определяется управляющим сигналом Счет/Сдвиг. При высоком уровне этого сигнала счетчики находятся в режиме прямого счета по фронту сигнала Такт. При низком уровне сигнала счетчики переходят в режим последовательного счета 12-разрядного счетчика через четыре разряда правого на схеме счетчика. Первым читается состояние старшего счетчика, последним — младшего. Сдвиг выходного кода происходит по положительному фронту тактового сигнала. После трех импульсов тактового сигнала во все три счетчика оказывается записанным нулевой код, то есть схема готова к режиму прямого счета.

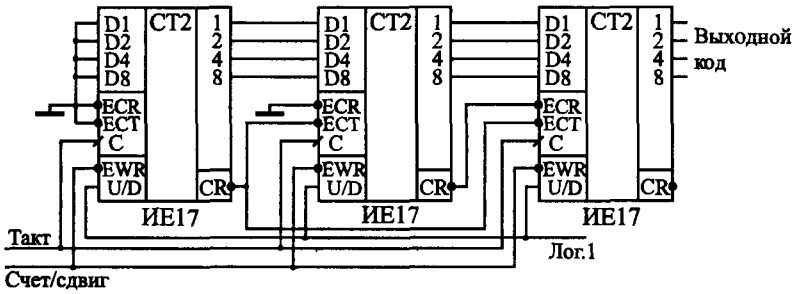


Рис. 5.42. Последовательное чтение выходного кода многокаскадного счетчика.

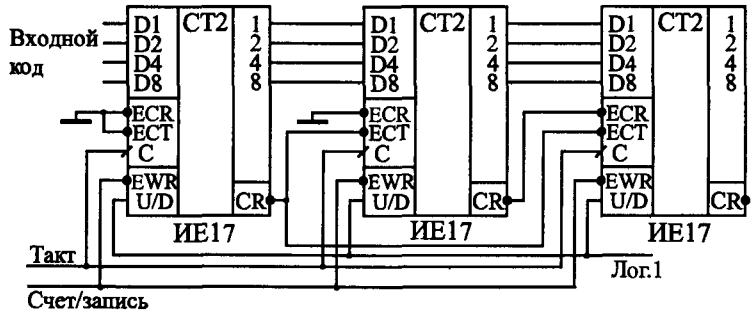


Рис. 5.43. Последовательная запись в счетчики исходного состояния.

И точно такая же последовательная перезапись информации из счетчика в счетчик позволяет с помощью 4-разрядных входных кодов записать исходное состояние нескольких последовательно соединенных счетчиков (рис. 5.43).

Перед началом работы схема переводится в состояние параллельной записи нулевым уровнем сигнала Счет/запись. При этом 4-разрядные коды, которые надо записать во все счетчики, по очереди подаются на вход первого (младшего) счетчика и сдвигаются по направлению к старшему счетчику по положительному фронту тактового сигнала С. Для записи всех трех счетчиков необходимо подать три тактовых импульса подряд. Причем первым надо записывать код, предназначенный для старшего (правого на схеме) счетчика, а последним — код, предназначенный для младшего (левого на схеме) счетчика.

Глава 6
ПРИМЕНЕНИЕ МИКРОСХЕМ ПАМЯТИ

Микросхемы памяти (или просто память, или запоминающие устройства — ЗУ, английское — Memory) представляют собой группу еще более сложных цифровых микросхем по сравнению с микросхемами, рассмотренными ранее. Память — это всегда очень сложная структура, включающая в себя множество элементов. Правда, внутренняя структура памяти регулярная, большинство элементов одинаковые, связи между элементами сравнительно простые, поэтому функции, выполняемые микросхемами памяти, не слишком сложные.

Память, как и следует из ее названия, предназначена для запоминания, хранения каких-то массивов информации, проще говоря, наборов, таблиц, групп цифровых кодов. Каждый код хранится в отдельном элементе памяти, называемом ячейкой памяти. Основная функция любой памяти как раз и состоит в выдаче этих кодов на выходы микросхемы по внешнему запросу. А основной параметр памяти — это ее объем, то есть количество кодов, которые могут в ней храниться, и разрядность этих кодов.

Для обозначения количества ячеек памяти используются следующие специальные единицы измерения:

- 1К — это 1024, то есть 2^{10} (читается «кило-» или «ка-»), примерно равно одной тысяче;
- 1М — это 1048576, то есть 2^{20} (читается «мега-»), примерно равно одному миллиону;
- 1Г — это 1073741824, то есть 2^{30} (читается «гига-»), примерно равно одному миллиарду.

Принцип организации памяти записывается следующим образом: сначала пишется количество ячеек, а затем через знак умножения (косой крест) пишется разрядность кода, хранящегося в одной ячейке. Например, организация памяти 64К × 8 означает, что память имеет 64К (то есть 65536) ячеек и каждая

ячейка — восьмиразрядная. А организация памяти 4М × 1 означает, что память имеет 4М (то есть 4194304) ячеек, причем каждая ячейка имеет всего один разряд. Общий объем памяти измеряется в байтах (килобайтах — Кбайт, мегабайтах — Мбайт, гигабайтах — Гбайт) или в битах (килобитах — Кбит, мегабитах — Мбит, гигабитах — Гбит).

В зависимости от способа занесения (записи) информации и от способа хранения информации микросхемы памяти разделяются на следующие основные типы:

- Постоянная память (ПЗУ — постоянное запоминающее устройство, ROM — Read Only Memory — память только для чтения), в которую информация заносится один раз на этапе изготовления микросхемы. Такая память называется еще масочным ПЗУ. Информация в памяти не пропадает при выключении ее питания, поэтому ее еще называют энергонезависимой памятью.
- Программируемая постоянная память (ППЗУ — программируемое ПЗУ, PROM — Programmable ROM), в которую информация может заноситься пользователем с помощью специальных методов (ограниченное число раз). Информация в ППЗУ тоже не пропадает при выключении ее питания, то есть она также энергонезависимая.
- Оперативная память (ОЗУ — оперативное запоминающее устройство, RAM — Random Access Memory — память с произвольным доступом), запись информацию в которую наиболее проста и может производиться пользователем сколько угодно раз на протяжении всего срока службы микросхемы. Информация в памяти пропадает при выключении ее питания.

Существует множество промежуточных типов памяти, а также множество подтипов, но указанные типы самые главные, принципиально отличающиеся друг от друга. Хотя разница между ПЗУ и ППЗУ с точки зрения разработчика цифровых устройств, как правило, не так уж велика, но в отдельных случаях, например, при использовании так называемой флэш-памяти (flash-memory), представляющей собой ППЗУ с многократным электрическим стиранием и перезаписью информации, эта разница действительно чрезвычайно важна. Можно считать, что флэш-память занимает промежуточное положение между ОЗУ и ПЗУ.

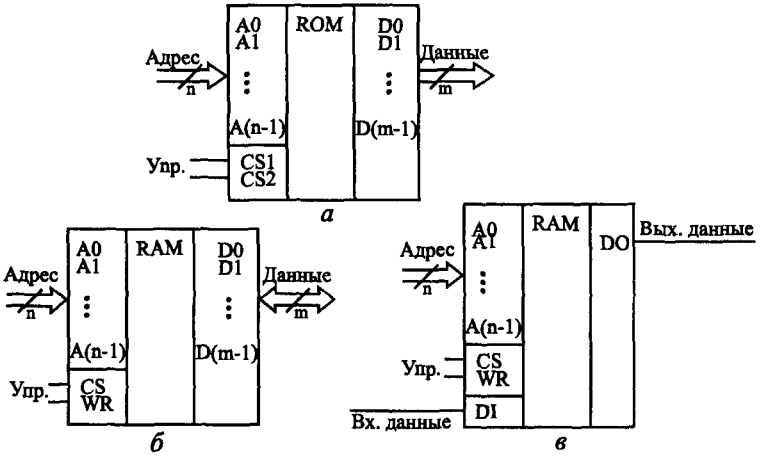


Рис. 6.1. Микросхемы памяти: ПЗУ (*а*), ОЗУ с двунаправленной шиной данных (*б*), ОЗУ с раздельными шиами входных и выходных данных (*в*).

В общем случае любая микросхема памяти имеет следующие информационные выводы (рис. 6.1):

- Адресные выводы (входные), образующие шину адреса памяти. Код на адресных линиях представляет собой двоичный номер ячейки памяти, к которой происходит обращение в данный момент. Количество адресных разрядов определяет количество ячеек памяти: при количестве адресных разрядов n количество ячеек памяти равно 2^n.
- Выводы данных (выходные), образующие шину данных памяти. Код на линиях данных представляет собой содержимое той ячейки памяти, к которой производится обращение в данный момент. Количество разрядов данных определяет количество разрядов всех ячеек памяти (обычно оно бывает равным 1, 4, 8, 16). Как правило, выходы данных имеют тип выходного каскада ОК или 3С.
- В случае оперативной памяти помимо выходной шины данных может быть еще и отдельная входная шина данных, на которую подается код, записываемый в выбранную ячейку памяти. Другой возможный вариант — совмещение входной и выходной шин данных, то есть двунаправленная шина данных, направление передачи информации по которой определяется

управляющими сигналами. Двунаправленная шина применяется обычно при использовании 4-разрядной шины данных или шин данных еще большей разрядности.

- Управляющие выводы (входные), которые определяют режим работы микросхемы. В большинстве случаев у памяти имеется вход выбора микросхемы CS (их может быть несколько, объединенных по функции И). У оперативной памяти также обязательно есть вход записи WR, активный уровень сигнала на котором переводит микросхему в режим записи.

Мы в данной главе не будем, конечно, изучать все возможные разновидности микросхем памяти, для этого не хватит целой книги. К тому же эта информация содержится в многочисленных справочниках. Микросхемы памяти выпускаются десятками фирм во всем мире, поэтому даже перечислить все их не слишком просто, не говоря уже о том, чтобы подробно рассматривать их особенности и параметры. Мы всего лишь рассмотрим различные схемы включения типичных микросхем памяти для решения наиболее распространенных задач, а также методы проектирования некоторых узлов и устройств на основе микросхем памяти. Именно это имеет непосредственное отношение к цифровой схемотехнике. И именно способы включения микросхем мало зависят от характерных особенностей той или иной микросхемы той или иной фирмы.

6.1. Постоянная память

В данном разделе мы не будем говорить о флэш-памяти, так как область ее применения пока не слишком широка. Мы ограничимся только микросхемами ПЗУ и ППЗУ, информация в которые заносится раз и навсегда (на этапе изготовления или же самим пользователем). Мы также не будем рассматривать здесь особенности оборудования для программирования ППЗУ (так называемых программаторов), принципы их построения и использования, это отдельная большая тема. Мы будем считать, что нужная нам информация может быть записана в ПЗУ или ППЗУ, а когда, как, каким способом она будет записана, нам не слишком важно. Все эти допущения позволят нам сосредоточиться именно на схемотехнике узлов и устройств на основе ПЗУ и ППЗУ (для простоты будем называть их в дальнейшем просто ПЗУ).

Упомянем здесь только, что ППЗУ делятся на репрограммируемые или перепрограммируемые ПЗУ (РПЗУ, EPROM — Erasable Programmable ROM), то есть допускающие стирание и перезапись информации, и однократно программируемые ПЗУ. В свою очередь РПЗУ делятся на ПЗУ, информация в которых стирается электрическими сигналами (EEPROM — Electrically Erasable Programmable ROM), и на ПЗУ, информация в которых стирается ультрафиолетовым излучением через специальное прозрачное окошко в корпусе микросхемы (собственно EPROM — Erasable Programmable ROM). Запись информации в любые ППЗУ производится с помощью подачи определенных последовательностей электрических сигналов (как правило, повышенного напряжения) на выводы микросхемы.

Фирмами-производителями цифровых микросхем выпускается немало самых разнообразных ПЗУ и ППЗУ. Различаются микросхемы постоянной памяти разным объемом (от 32 байт до 8 Мбайт и более), разной организацией (обычно количество разрядов данных бывает 4, 8 или 16), способами управления (назначением управляющих сигналов), типами выходных каскадов (обычно ОК или 3С), разным быстродействием (обычно задержка составляет от единиц до сотен наносекунд). Но суть всех микросхем ПЗУ остается одной и той же: имеется шина адреса, на которую надо подавать код адреса нужной ячейки памяти, имеется шина данных, на которую выдается код, записанный в адресуемой ячейке, и имеются входы управления, которые разрешают или запрещают выдачу информации из адресуемой ячейки на шину данных.

На рис. 6.2 представлены для примера несколько простейших и типичных микросхем постоянной памяти.

Микросхема К155РЕ3 (аналог — N8223N) представляет собой однократно программируемое ППЗУ с организацией 32 × 8. Исходное состояние (до программирования) — все биты всех ячеек нулевые. Для программирования (записи информации) используется специальный программатор, подающий на разряды данных импульсы высокого напряжения. Тип выходных каскадов — открытый коллектор, то есть обязательно надо включать на выходах резисторы, подсоединенные к шине питания. Имеется один управляющий вход -CS, при положительном уровне сигнала на котором на всех выходах устанавливаются единицы.

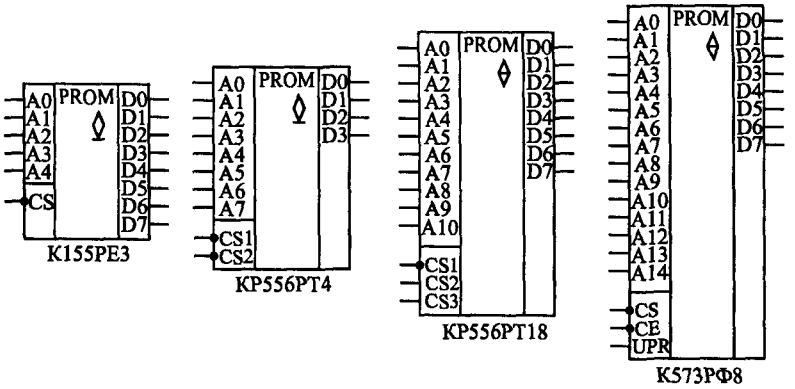


Рис. 6.2. Примеры микросхем ППЗУ отечественного производства.

Микросхема КР556РТ4 (аналог — I3601) — это также однократно программируемая постоянная память с организацией 256 × 4. Исходное состояние (до программирования) — все биты всех ячеек нулевые. Тип выходных каскадов — ОК. Два управляющих входа -CS1 и -CS2 объединены по принципу И, то есть для разрешения работы микросхемы (для перевода выходов в активное состояние) оба эти сигнала должны быть нулевыми. Для записи информации в микросхему используется программатор.

Микросхема КР556РТ18 (аналог — HM76161) также является однократно программируемым ППЗУ и имеет организацию 2К × 8. Тип выходов микросхемы — 3С. Имеются три управляющих входа: один инверсный -CS1, два других — прямые CS2 и CS3, объединенных по функции И. Выходы данных переходят в активное состояние при нулевом уровне на -CS1 и при единичных уровнях на CS2 и CS3. Если входы управления используются для подачи управляющих сигналов (то есть выходы могут переходить в третье состояние), то на выходах надо включать нагрузочные резисторы, подключенные к шине питания. Исходное состояние микросхемы (до программирования) — все биты всех ячеек единичные.

Наконец, микросхема К573РФ8 (аналог — I27256) — это пример памяти РПЗУ с ультрафиолетовым стиранием информации. Чтобы перепрограммировать память, необходимо ее стереть, для чего в течение некоторого времени (обычно

несколько минут) надо облучать микросхему через окошко в корпусе ультрафиолетовым излучением (можно использовать медицинский кварцевый облучатель). Стертая микросхема имеет все биты установленные в единицу. Затем проводится процедура записи с помощью программатора, несколько отличающегося от программаторов однократно программируемых микросхем. Управляющие входы -CS и -CE должны быть установлены в нуль для перевода выходов микросхемы в активное состояние. Имеется специальный вход UPR для подачи программирующего высокого напряжения, на который при чтении информации из микросхемы надо подавать напряжение питания. Тип выходных каскадов — 3С. Микросхемы этого типа самые медленные, их задержки самые большие.

Основные временные характеристики микросхем ПЗУ — это две величины задержки. Задержка выборки адреса памяти — это время от установки входного кода адреса до установки выходного кода данных. Задержка выборки микросхемы — это время от установки активного разрешающего управляющего сигнала CS до установки выходного кода данных памяти. Задержка выборки микросхемы обычно в несколько раз меньше задержки выборки адреса.

Содержимое ПЗУ обычно изображается в виде специальной таблицы, называемой картой прошивки памяти. В таблице показывается содержимое всех ячеек памяти, причем в каждой строке записывается содержимое 16 (или 32) последовательно идущих (при нарастании кода адреса) ячеек памяти. При этом, как правило, используется 16-ричное кодирование.

Пример карты прошивки ПЗУ с организацией 256 × 8 показан в табл. 6.1 (все биты всех ячеек считаются установленными в единицу). Пользоваться таблицей очень просто. Например, для того, чтобы посмотреть содержимое ячейки памяти с 16-ричным адресом 8A, надо взять строку таблицы с номером 80 и столбец таблицы с номером A (данная ячейка в таблице выделена жирным шрифтом).

Любые микросхемы ПЗУ легко можно включать так, чтобы уменьшать или увеличивать количество адресных разрядов, то есть уменьшать или увеличивать количество используемых ячеек памяти. И то и другое часто требуется при построении схем цифровых устройств.

Таблица 6.1. Пример карты прошивки ПЗУ

Адрес	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
20	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
30	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
40	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
50	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

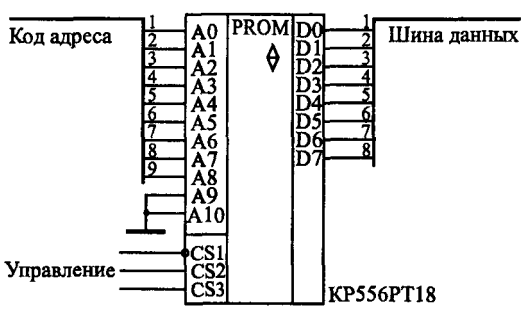


Рис. 6.3. Уменьшение количества адресных разрядов ПЗУ.

Для уменьшения количества адресных разрядов необходимо на нужное число старших адресных входов подать нулевые сигналы. Каждый отключенный таким образом адресный разряд уменьшает количество ячеек ПЗУ вдвое. Например, на рис. 6.3 показано, как из микросхемы с организацией 2К × 8 сделать

микросхему 512 × 8. Два старших разряда адреса памяти отключены (на них поданы нулевые сигналы). Использоваться будут только младшие (верхние в таблице прошивки) 512 ячеек памяти, и только их надо будет программировать. Конечно, гораздо лучше подобрать микросхему именно с тем количеством ячеек, которое действительно необходимо в данной схеме, но это, к сожалению, возможно не всегда.

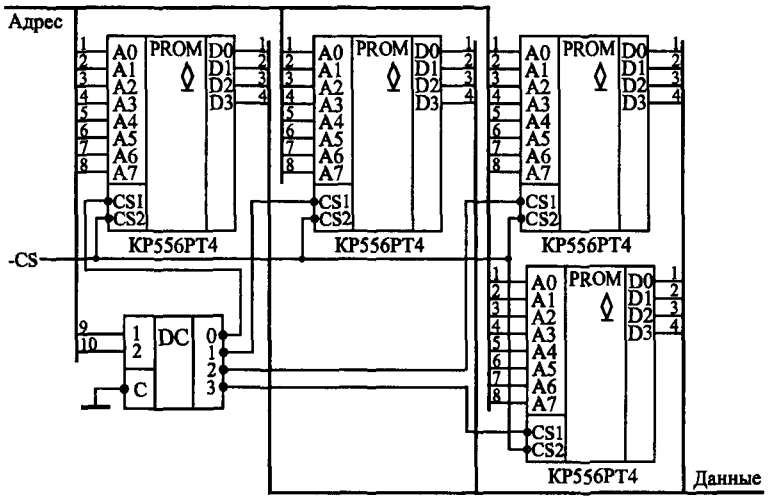


Рис. 6.4. Увеличение количества адресных разрядов ПЗУ с помощью дешифратора.

Задача увеличения количества адресных разрядов ПЗУ встречается значительно чаще задачи уменьшения количества адресных разрядов. В результате такого увеличения возрастает объем ПЗУ, объемы отдельных микросхем суммируются. Для увеличения адресных разрядов обычно применяются микросхемы дешифраторов (рис. 6.4). Младшие разряды шины адреса при этом подаются на объединенные адресные входы всех микросхем, а старшие — на управляющие (адресные) входы дешифратора. Выходные сигналы дешифратора разрешают работу всегда только одной микросхемы памяти. В результате на общую шину данных всех ПЗУ выдает свою информацию только одна микросхема. На рисунке для простоты не показаны выходные резисторы с разрядов данных на шину питания, подключе-

ние которых чаще всего необходимо, так как тип выходов данных микросхем ПЗУ — это ОК или 3С.

В результате подобного объединения микросхем ПЗУ может увеличиться время выборки адреса полученного единого ПЗУ. В данном случае (см. рис. 6.4) оно будет равно максимальной из двух величин: времени выборки адреса одной микросхемы и суммы двух задержек: задержки дешифратора и задержки выборки микросхемы ПЗУ.

Если надо объединить две микросхемы (то есть добавить всего один разряд адресной шины), то можно обойтись без дешифратора, подавая на вход -CS одной микросхемы прямой дополнительный сигнал адреса, а на вход -CS другой микросхемы — этот же сигнал с инверсией. Применение дешифратора 3—8 позволяет объединить 8 микросхем ПЗУ (добавить три адресных разряда), а применение дешифратора 4—16 добавляет четыре адресных разряда, объединяя 16 микросхем ПЗУ.

Часто возникает также задача увеличения количества разрядов данных. Для этого необходимо всего лишь объединить одноименные адресные входы нужного количества микросхем ПЗУ, а выходы данных ПЗУ не объединяются, а образуют код с большим числом разрядов. Например, при объединении таким образом двух микросхем с организацией 8К × 8 можно получить ПЗУ с организацией 8К × 16.

6.1.1. ПЗУ как универсальная комбинационная микросхема

Одно из самых распространенных применений микросхем ПЗУ — замена ими сложных комбинационных схем. Такое решение позволяет существенно упростить проектируемое устройство и снизить количество используемых комбинационных микросхем, а также иногда уменьшить потребляемый ток и увеличить быстродействие схемы.

Суть предлагаемого подхода сводится к следующему. Если рассматривать адресные входы микросхемы ПЗУ как входы комбинационной схемы, а разряды данных — как выходы этой комбинационной схемы, то можно сформировать *любую* требуемую таблицу истинности данной комбинационной схемы. Для этого всего лишь надо составить таблицу прошивки ПЗУ, соответствующую нужной таблице истинности. В этом случае не надо ни подбирать логические элементы, ни оптимизировать

их соединения, ни думать о том, можно ли вообще построить требуемую комбинационную схему из стандартных микросхем. Важно только, чтобы количество требуемых входов не превышало количества адресных разрядов ПЗУ, а количество требуемых выходов не превышало разрядности шины данных ПЗУ.

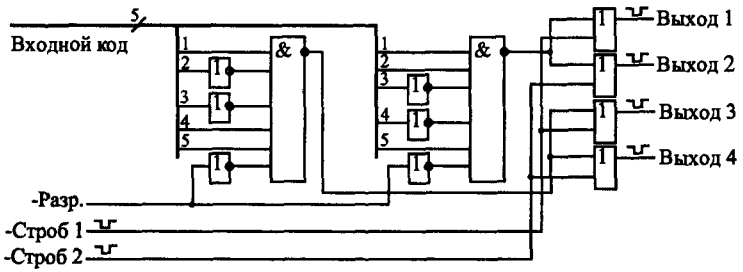


Рис. 6.5. Пример комбинационной схемы, заменяемой на ПЗУ.

В качестве примера рассмотрим довольно сложную комбинационную схему (рис. 6.5), имеющую восемь входов и четыре выхода. Функция схемы сводится к следующему. Прежде всего она распознает два различных 5-разрядных входных кода (11001 и 10011) в случае, когда на входе разрешения -Разр. присутствует нулевой сигнал. А при приходе сигналов -Строб 1 и -Строб 2 схема выдает на выход отрицательные импульсы. Причем первый выходной сигнал вырабатывается в случае, когда входной код равен 11001 и пришел сигнал -Строб 1, второй выходной сигнал — при том же коде, но по входному сигналу -Строб 2. Третий и четвертый выходные сигналы вырабатываются при входном коде 10011 и при приходе соответственно управляющих сигналов -Строб 1 и -Строб 2. То есть логика работы довольно сложная, и разнообразных логических элементов требуется немало.

Но всю эту схему можно заменить всего лишь одной микросхемой ПЗУ, например типа РТ4, имеющей 8 адресных входов и 4 выхода данных (рис. 6.6). При этом пять разрядов входного кода подаются на младшие разряды адреса ПЗУ (A0...A4), входной сигнал -Разр. — на адресный вход A5, сигнал -Строб 1 — на вход A6, сигнал -Строб 2 — на вход A7. Младший разряд данных памяти D0 используется для первого выходного сигнала, D1 — для второго выходного сигнала, D2 — для третьего вы-

ходного сигнала, D3 — для четвертого выходного сигнала. Микросхема ПЗУ всегда выбрана (управляющие сигналы -CS1 и -CS2 — нулевые). На выходах данных памяти включены резисторы, так как тип выходов микросхемы РТ4 — ОК.

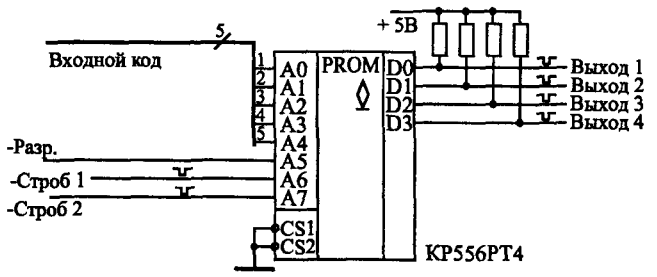


Рис. 6.6. Включение ПЗУ для замены комбинационной схемы, показанной на рис. 6.5.

Составим карту прошивки ПЗУ. Активные выходные сигналы — нулевые, а пассивные — единичные. Значит, в большинстве ячеек ПЗУ будут записаны коды F (то есть все выходные сигналы пассивны). Активному (нулевому) первому выходному сигналу при пассивных остальных будет соответствовать двоичный код данных 1110 (16-ричный код — E), активному второму выходному сигналу будет соответствовать двоичный код 1101 (16-ричный — D), активному третьему выходному сигналу — двоичный код 1011 (или B), активному четвертому выходному сигналу — двоичный код 0111 (или 7). То есть только содержимое четырех ячеек памяти будет отличаться от F.

Например, код E будет записан в ячейку с таким адресом, значения пяти младших разрядов которого (A0...A4) равны селектируемому входному коду 11001, разряда A5 — нулю (сигнал -Разр. активен), разряда A6 — нулю (сигнал -Строб 1 активен) и разряда A7 — единице (сигнал -Строб 2 пассивен). То есть получаем двоичный код адреса 10011001 (или в 16-ричном коде 99). Точно так же код D будет записан в ячейку с адресом 01011001 (то есть 16-ричное 59), код B — в ячейку с адресом 10010011 (то есть 93), а код 7 — в ячейку с адресом 01010011 (то есть 53). Получившаяся карта прошивки ПЗУ приведена в табл. 6.2. Она полностью совпадает с таблицей истинности заменяемой комбинационной схемы.

Таблица 6.2. Карта прошивки ПЗУ для замены комбинационной схемы

Адрес	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
20	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
30	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
40	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
50	F	F	F	7	F	F	F	F	F	D	F	F	F	F	F	F
60	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
70	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
80	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
90	F	F	F	B	F	F	F	F	F	E	F	F	F	F	F	F
A0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
B0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
C0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
D0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
E0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

Может показаться, что такое использование микросхемы ПЗУ чересчур расточительно, избыточно, но это не так. Гораздо важнее, что схема сильно упрощается. Если же взять комбинационную схему с более сложной таблицей истинности, то возможности ПЗУ будут использованы полнее. К тому же большое достоинство такого решения состоит в том, что при необходимости изменения логики работы комбинационной схемы потребуется всего лишь перепрошивка ПЗУ, а не проектирование новой схемы из логических элементов. Задержка ПЗУ при замене комбинационной схемы любой сложности остается одной и той же, она равна задержке выборки адреса микросхемы ПЗУ. При сложной заменяемой комбинационной схеме ПЗУ может оказаться даже быстрее.

Однако использование ПЗУ для замены комбинационных схем имеет и свои довольно серьезные недостатки. Дело в том, что микросхемы ПЗУ еще больше, чем комбинационные микро-

схемы, чувствительны к моменту изменения входных сигналов (адресных разрядов). На выходах данных микросхем ПЗУ при любом изменении входного кода адреса могут появляться короткие паразитные импульсы. Поэтому лучше всего использовать ПЗУ для замены комбинационных схем, которые работают в статическом режиме, и в которых короткие импульсы не имеют значения.

Можно также применять методы синхронизации выходных сигналов ПЗУ (рис. 6.7) с помощью управляющих сигналов выбора микросхемы CS (*а*) или же с помощью выходных триггеров и регистров (*б*). Суть синхронизации состоит в том, что выходные сигналы ПЗУ надо разрешать или фиксировать с помощью синхросигнала только тогда, когда все переходные процессы внутри микросхемы, вызванные сменой кода адреса, уже закончились, и паразитные импульсы на выходах гарантированно отсутствуют.

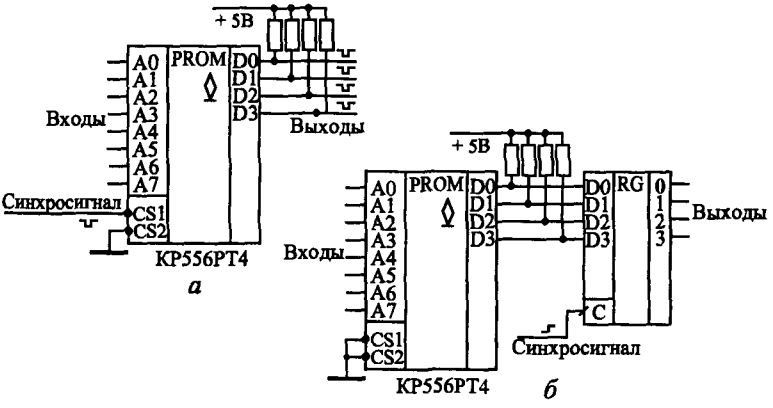


Рис. 6.7. Методы синхронизации выходных сигналов ПЗУ с помощью сигнала CS (*а*) и выходного регистра (*б*).

Микросхемы ПЗУ могут заменять собой любые комбинационные микросхемы: дешифраторы, шифраторы, компараторы кодов, сумматоры, мультиплексоры, преобразователи кодов и т. д. Однако при подобной замене всегда стоит подумать, не лучше ли использовать уже готовые микросхемы, чем изготавливать новые (программировать ПЗУ). Микросхемы ПЗУ могут оказаться медленнее стандартных комбинационных микросхем

и потреблять больший ток питания. К тому же они могут потребовать использования выходных резисторов, если микросхемы имеют выходы ОК или при использовании входов CS у микросхем с выходами 3С. Другое дело, когда ПЗУ выполняет функцию, отличающуюся от функции стандартной комбинационной микросхемы. Простейший пример — дешифратор с положительными, а не с отрицательными (как в стандартных сериях) активными выходными сигналами.

В общем случае ПЗУ можно рассматривать как преобразователь входного кода (кода адреса) в выходной код (код данных) по произвольному закону, задаваемому разработчиком. Это позволяет не только преобразовывать друг в друга различные стандартные коды, но и выполнять множество других функций, например использовать ПЗУ как простейший табличный вычислитель. Для этого надо на адресные разряды ПЗУ подать код входного числа (аргумента), а на выходах разрядов данных получить код выходного числа (функции). Такой табличный вычислитель имеет очень высокое быстродействие по сравнению с другими типами вычислителей (время вычисления функции равно задержке выборки адреса ПЗУ).

В качестве простейшего примера рассмотрим вычислитель для возведения в квадрат 4-разрядного двоичного числа (рис. 6.8). Вычислитель выполнен на микросхеме ПЗУ типа РЕ3, у которого использованы четыре разряда адреса и восемь разрядов данных. Он позволяет получать двоичные коды квадратов любых чисел в диапазоне от 0 (или в двоичном коде 0000, в 16-ричном коде 0) до 15 (или в двоичном коде 1111, в 16-ричном коде F), которые принимают значения от 0 (или в двоичном коде 00000000, в 16-ричном — 00) до 225 (или в двоичном коде 11100001, в 16-ричном — E1).

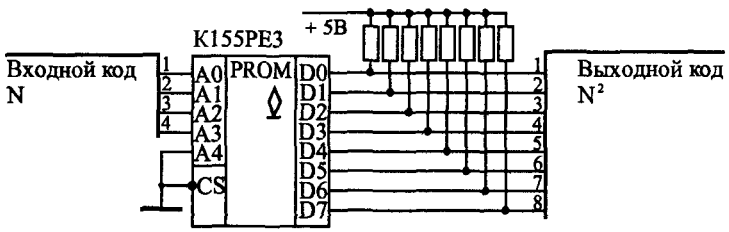


Рис. 6.8. Вычислитель квадратов входных чисел.

Карта прошивки ПЗУ вычислителя квадратов (табл. 6.3) будет очень проста: код данных в каждой ячейке равен квадрату кода адреса этой ячейки. Используется всего 16 ячеек памяти, содержимое остальных 16 ячеек не имеет значения (что обозначено в таблице XX).

Таблица 6.3. Карта прошивки ПЗУ-вычислителя квадратов

Адрес	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	01	04	09	10	19	24	31	40	51	64	79	90	A9	C4	E1
10	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

На такой же точно схеме (рис. 6.8) можно сделать и вычислитель квадратов в двоично-десятичном коде. В этом случае будет использовано всего лишь десять ячеек памяти с адресами от 0 до 9, а в ячейках будут записаны их квадраты от 0 до 81.

Недостаток любого табличного вычислителя на ПЗУ — это необходимость увеличения вдвое требуемого объема памяти при увеличении разрядности входного числа на единицу. Например, при 8-разрядных входных числах требуется ПЗУ с количеством ячеек 256, при 16-разрядных входных числах — ПЗУ с количеством ячеек 64К, а при 32-разрядных входных числах — ПЗУ с количеством ячеек 4Г. Такие большие объемы очень трудно реализовать на серийно выпускаемых микросхемах. Поэтому табличные вычислители на ПЗУ обычно строятся только для разрядности входных чисел не более 16.

Одно из наиболее распространенных применений ПЗУ как преобразователя кодов — это построение на их основе всевозможных индикаторов, отображающих на экране буквы и цифры. ПЗУ в данном случае переводит код (номер) буквы или цифры в ее изображение. Конечно, в данном случае заменить ПЗУ комбинационной схемой совершенно невозможно, так как букв и цифр очень много, а их изображения очень разнообразны.

Простейший пример данного применения ПЗУ — это управление знаковым семисегментным индикатором, знакомым всем по калькуляторам, кассовым аппаратам, электронным часам, весам и т. д. В семисегментных индикаторах изображение всех цифр от 0 до 9 строится всего из семи сегментов (отрезков линий) (рис. 6.9).

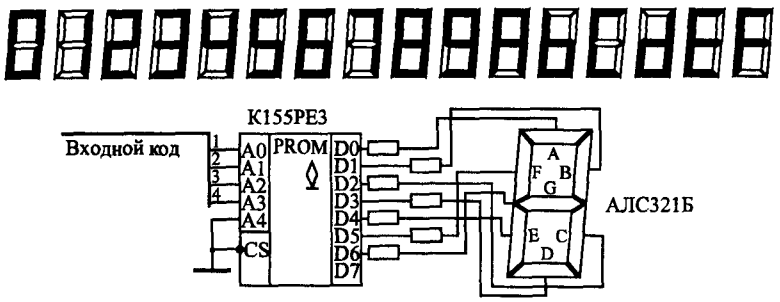


Рис. 6.9. Дешифратор знакового семисегментного индикатора на ПЗУ.

Чтобы отобразить в виде цифры 4-разрядный двоичный код, надо этот код преобразовать в 7-разрядный код, каждому разряду которого будет соответствовать один сегмент индикатора. То есть коду 0000 должно соответствовать изображение нуля (6 сегментов, расположенных по периметру), а коду 0001 — изображение единицы (два правых вертикальных сегмента). Для повышения универсальности индикатора удобно дополнить десять цифр еще и шестью буквами, использующимися в 16-ричном коде (A, B, C, D, E, F). Семь сегментов индикатора позволяют сделать и это, правда, изображения букв при этом получаются не слишком качественными.

ПЗУ типа РЕ3, используемое в качестве дешифратора индикатора, имеет 4 входа и 7 выходов (старший разряд адреса и старший разряд данных не используются). Карта прошивки ПЗУ приведена в табл. 6.4. Нулевой сигнал на каждом из выходов данных ПЗУ зажигает соответствующий ему сегмент.

Таблица 6.4. Карта прошивки ПЗУ для дешифратора знакового индикатора

Адрес	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	40	79	24	30	19	12	02	78	00	10	08	03	46	21	06	0E
10	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

ПЗУ позволяют также формировать и более сложные изображения букв и цифр — матричные. Такие изображения используются, например, в табло типа «бегущая строка», на экранах мониторов, в больших рекламных табло. Каждая буква,

цифра, другой знак располагается в данном случае на прямоугольной матрице, называемой знакоместом и состоящей из нескольких строк и нескольких столбцов точечных элементов изображения, которые могут зажигаться независимо друг от друга. Чем больше строк и столбцов в знакоместе, тем более качественное изображение букв и цифр можно получить. Минимально возможный размер знакоместа — 5 столбцов на 7 строк, то есть всего 35 элементов изображения.

ПЗУ в данном случае содержит в себе информацию об изображениях всех возможных букв и цифр (обычно этот набор включает в себя 256 символов). Но выходной код ПЗУ имеет мало разрядов, поэтому каждый такой код, соответствующий одному адресу, представляет себой информацию об изображении не целого символа, а только одной его строки (или столбца). Информация о целом символе занимает в ПЗУ столько ячеек, сколько в изображении символа имеется столбцов (или строк). Пример матричного знакогенератора на ПЗУ приведен на рис. 6.10.

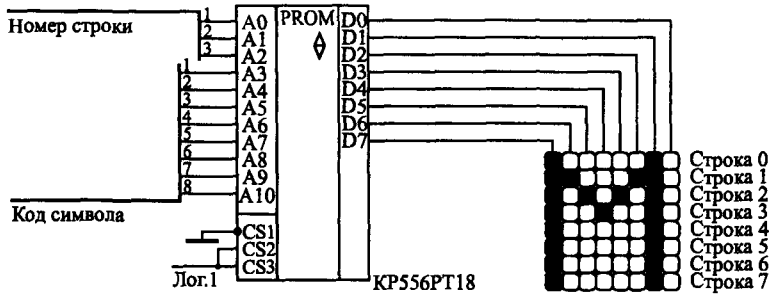


Рис. 6.10. Матричный знакогенератор на ПЗУ.

В данном случае используется знакоместная матрица из 8 строк и 8 столбцов. В каждую ячейку ПЗУ записывается код изображения одной из 8 строк одного из 256 символов. Изображение одного символа занимает 8 последовательно расположенных ячеек в ПЗУ. Для букв и цифр правый столбец знакоместа не используется, он служит для отделения знаков друг от друга. Он может и использоваться в случае специальных (например, графических) символов. В случае матричного светодиодного индикатора перебор строк может осуществляться 3-раз-

рядным счетчиком с дешифратором 3—8 на его выходе. В случае телевизионного монитора перебор строк осуществляется с помощью генератора вертикальной развертки изображения.

Составление карты прошивки такого ПЗУ непросто, оно обычно производится с помощью специальных программ на компьютере. Но принцип составления прост. Например, если активному (зажженному) элементу изображения соответствует единичный сигнал, то для нулевой строки символа «М», показанного на рисунке, в ПЗУ надо записать 10000010, для первой строки — код 11000110, для второй — код 10101010, для третьей — 10010010 и т. д.

6.1.2. ПЗУ в генераторах импульсных последовательностей

Следующее важнейшее применение ПЗУ — это построение генераторов сложных последовательностей цифровых импульсов. Такие генераторы широко используются в самых разных измерительных системах, в устройствах автоматики, в телевизионных системах, в схемах управления линейными или матричными индикаторами и т. д.

Задача в данном случае ставится следующим образом. Необходимо сформировать последовательность из нескольких сигналов различной длительности, сдвинутых относительно друг друга на различные временные интервалы. Причем последовательность эта может быть как разовой (однократно начинающейся по внешнему сигналу), так и периодической, непрерывно повторяющейся.

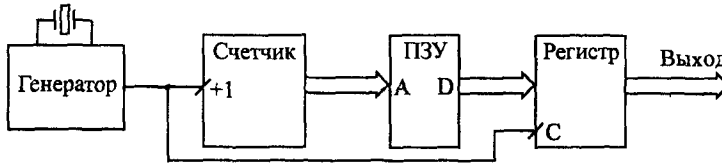


Рис. 6.11. Пример структуры генератора последовательностей сигналов на ПЗУ.

Наиболее распространенная структура генератора последовательностей выходных сигналов на ПЗУ включает в себя тактовый генератор нужной частоты, счетчик с требуемым числом разрядов, ПЗУ и выходной регистр (рис. 6.11). Счетчик перебри-

рает адреса ПЗУ, ПЗУ последовательно выдает на выходы данных все записанные в него коды. Выходной регистр, тактируемый тем же тактовым сигналом, что и счетчик, служит для предотвращения появления в выходных сигналах паразитных импульсов и для обеспечения одновременного переключения всех выходных сигналов (что особенно важно в случае, когда используются несколько параллельно включенных микросхем ПЗУ для увеличения разрядности шины данных).

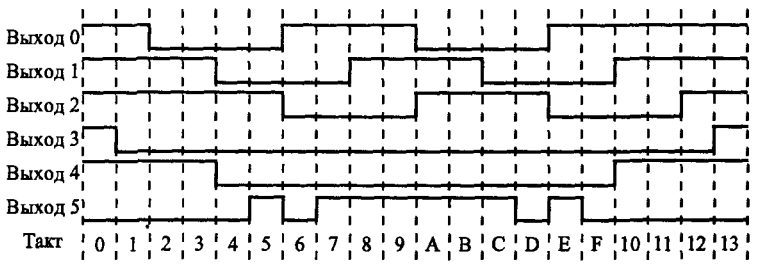


Рис. 6.12. Временная диаграмма формируемых выходных сигналов.

Рассмотрим пример. Пусть необходимо непрерывно формировать периодическую последовательность из шести выходных сигналов в соответствии с временной диаграммой рис. 6.12. Получить такую последовательность можно, конечно, с помощью комбинационных схем, включенных на выходе счетчика или с помощью множества одновибраторов, запускающих друг друга, но и то и другое решение чересчур громоздко и сложно как в проектировании, так и в настройке. Применение же ПЗУ значительно упрощает задачу. Достаточно провести несложные расчеты и составить карту прошивки ПЗУ.

Расчеты сводятся к следующему.

Прежде всего определяем минимально возможную тактовую частоту (с целью минимизации требуемого объема ПЗУ). Для этого надо выделить максимальный временной интервал (дискрет времени), который укладывается целое число раз во все временные сдвиги, задержки, длительности требуемой диаграммы. В нашем случае этот дискрет равен одному делению по оси времени. Например, если длительность этого деления равна 250 нс, то и период тактового сигнала надо выбирать 250 нс, то есть тактовая частота будет равна 4 МГц. Можно, конечно, вы-

брать ее и кратной 4 МГц, например 8 МГц, 12 МГц, но тогда потребуется вдвое или втрое больший объем ПЗУ. Если бы нам надо было формировать только три верхних сигнала (Выход 0, Выход 1, Выход 2), то период тактовой частоты можно было бы брать вдвое больше (в нашем примере — 500 нс), так как для этих сигналов все длительности кратны двум делениям.

Второй расчет сводится к определению количества ячеек и разрядности ПЗУ. Шесть выходных сигналов схемы требуют шести разрядов данных ПЗУ. Длительность последовательности равна 20 тактам (или 14 в 16-ричном коде), то есть не равна 2^n, поэтому счетчик придется сбрасывать в нуль через каждые 20 тактов, для чего потребуется еще один разряд данных ПЗУ. Итого потребуется 7 разрядов. А для перебора 20 тактов последовательности потребуется 5-разрядный счетчик, так как $2^4 = 16$ (недостаточно), а $2^5 = 32$ (достаточно). Значит, разрядность шины адреса ПЗУ также должна быть не менее пяти, то есть минимальные требования к организации ПЗУ — это 32 × 8, значит, подойдет микросхема ПЗУ типа РЕ3.

Наконец, третий расчет касается условий правильной работы схемы. Генератор последовательности будет работать правильно, если за период тактового сигнала успеют сработать счетчик и ПЗУ. То есть сумма задержки полного переключения счетчика и задержки выборки адреса ПЗУ не должна превышать периода тактового сигнала.

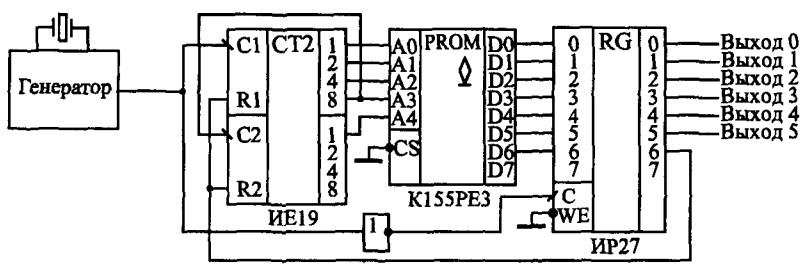


Рис. 6.13. Первый вариант схемы генератора последовательности сигналов на ПЗУ.

Таким образом, один из возможных вариантов схемы генератора последовательности импульсов (рис. 6.13) будет включать в себя тактовый генератор, пятиразрядный счетчик на ос-

нове ИЕ19, ПЗУ типа К155РЕ3 и 8-разрядный выходной регистр ИР27. Так как счетчик срабатывает по отрицательному фронту тактового сигнала, а регистр — по положительному фронту тактового сигнала, необходимо включить инвертор. На схеме для простоты не показаны резисторы на выходах данных типа ОК микросхемы ПЗУ.

Из схемы видно, что существует еще одно дополнительное условие правильной работы, связанное с использованием сигнала сброса. За период тактового сигнала должен успеть сработать регистр (записать в себя сигнал сброса с выхода ПЗУ), должен сброситься счетчик и должно выдать код ПЗУ. То есть сумма задержек регистра, сброса счетчика и выборки адреса ПЗУ не должна превышать периода тактового сигнала.

Теперь составим карту прошивки ПЗУ. Для этого сформируем таблицу значений всех семи выходных сигналов во всех двадцати рабочих тактах генератора последовательностей (табл. 6.5). Значения шести выходных сигналов (Выход 0 ... Выход 5) мы непосредственно берем из требуемой временной диаграммы (см. рис. 6.12). Хотя эти сигналы будут приходить на выход схемы с задержкой на один такт из-за наличия выходного регистра, но при периодической работе генератора последовательности это не имеет никакого значения.

Седьмой сигнал (Выход 6) используется в качестве сброса счетчика, он равен нулю от нулевого такта до 18 (в 16-ричном коде — 12) такта и равен единице в последнем девятнадцатом такте (в 16-ричном коде — 13). Этим единичным сигналом счетчик будет сбрасываться в нуль, то есть работа схемы после 19 такта будет возобновляться с нулевого такта.

Пусть нам требуется более высокое быстродействие схемы. Например, период тактового генератора должен быть равен 100 нс и даже меньше. В этом случае асинхронный счетчик типа ИЕ19 уже не подойдет из-за своего низкого быстродействия. Надо использовать 5-разрядный синхронный счетчик. Можно, конечно, включить две микросхемы 4-разрядных счетчиков типа ИЕ17, но можно обойтись и одним 4-разрядным счетчиком, если задействовать оставшийся свободным выход данных ПЗУ и оставшийся свободным разряд регистра (рис. 6.14). При этом без всякого ущерба для быстродействия можно использовать синхронный счетчик с асинхронным переносом ИЕ7, так как нужна всего одна микросхема счетчика.

Таблица 6.5. Карта прошивки ПЗУ генератора последовательности сигналов

Такт (адрес)	Вых.6	Вых.5	Вых.4	Вых.3	Вых.2	Вых.1	Вых.0	Код (данные)
0	0	0	1	1	1	1	1	1F
1	0	0	1	0	1	1	1	17
2	0	0	1	0	1	1	0	16
3	0	0	1	0	1	1	0	16
4	0	0	0	0	1	0	0	04
5	0	1	0	0	1	0	0	24
6	0	0	0	0	0	0	1	01
7	0	1	0	0	0	0	1	21
8	0	1	0	0	0	1	1	23
9	0	1	0	0	0	1	1	23
A	0	1	0	0	1	1	0	26
B	0	1	0	0	1	1	0	26
C	0	1	0	0	1	0	0	24
D	0	0	0	0	1	0	0	04
E	0	1	0	0	0	0	1	21
F	0	0	0	0	0	0	1	01
10	0	0	1	0	0	1	1	13
11	0	0	1	0	0	1	1	13
12	0	0	1	0	1	1	1	17
13	1	0	1	1	1	1	1	5F

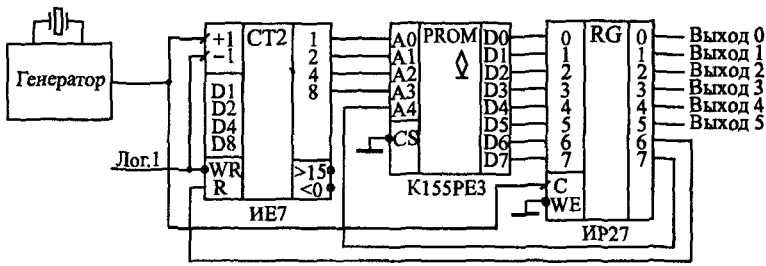


Рис. 6.14. Второй вариант схемы генератора последовательности с 4-разрядным синхронным счетчиком ИЕ7.

Карта прошивки ПЗУ генератора последовательностей остается той же самой, что и в предыдущем случае, но добавляется один (седьмой) разряд шины данных ПЗУ. Значение этого разряда должно быть равно нулю в пятнадцати тактах с нулевого до 14 (E в 16-ричном коде), в четырех тактах с 15 (F в 16-ричном коде) по 18 (12 в 16-ричном коде) его значение должно быть равным единице (код данных увеличится на 80), а в последнем 19 такте (13 в 16-ричном коде) — снова нулю. В результате адреса ПЗУ будут перебираться от 0 до 15 (старший разряд адреса — нулевой), затем от 16 до 19 (в старшем разряде адреса единица, а счетчик после переполнения считает с нуля), а затем снова от нулевого адреса (счетчик сбросится сигналом Выход 6, а старший разряд адреса памяти станет нулевым). Получающаяся в результате карта прошивки ПЗУ приведена в табл. 6.6.

Таблица 6.6. Карта прошивки ПЗУ для варианта схемы генератора последовательности импульсов

Адрес	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
00	1F	17	16	16	04	24	01	21	23	23	26	26	24	04	21	81
10	93	93	97	5F	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Конечно, рассмотренный пример довольно прост, при использовании ПЗУ большего объема генерируемые последовательности сигналов могут быть гораздо сложнее рассмотренной, и выходных сигналов может быть больше, но суть остается прежней: ПЗУ выдает любые последовательности выходных сигналов минимальными средствами. Причем генерируемые последовательности можно довольно просто изменять, заменяя ПЗУ на микросхемы с другой картой прошивки.

6.1.3. Микропрограммные автоматы на ПЗУ

Использование микропрограммных автоматов — это следующий шаг по пути усложнения «интеллекта» цифровых схем. На основе микропрограммных автоматов можно строить устройства, которые работают по довольно сложным алгоритмам, выполняют различные функции, определяемые входными сигналами, выдают сложные последовательности выходных сигна-

лов. При этом алгоритм работы микропрограммного автомата может быть легко изменен заменой прошивки ПЗУ.

В отличие от рассматривавшихся ранее устройств на «жесткой» логике, принцип работы которых однозначно определяется используемыми элементами и способом их соединения, микропрограммные автоматы с помощью одной и той же схемы могут выполнять самые разные функции. То есть они гораздо более гибкие, чем схемы на «жесткой» логике. К тому же проектировать микропрограммные автоматы с точки зрения схемотехники довольно просто. Недостатком любого микропрограммного автомата по сравнению со схемами на «жесткой» логике является меньшее предельное быстродействие и необходимость составления карты прошивки ПЗУ с микропрограммами, часто довольно сложными.

Наиболее распространенная структура микропрограммного автомата (рис. 6.15) включает в себя всего лишь три элемента: ПЗУ, регистр, срабатывающий по фронту, и тактовый генератор.

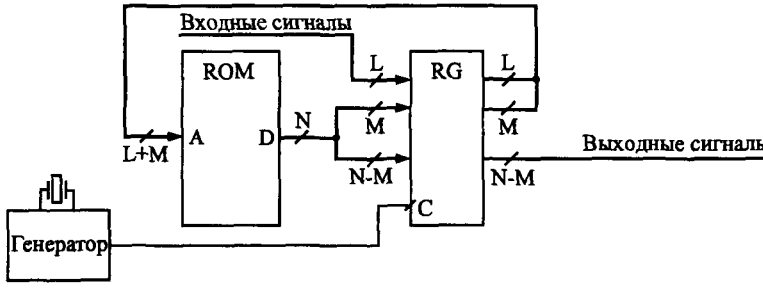


Рис. 6.15. Структура микропрограммного автомата.

ПЗУ имеет (L+M) адресных разрядов и N разрядов данных. Регистр используется с количеством разрядов (N+L). Разряды данных ПЗУ записываются в регистр по положительному фронту тактового сигнала генератора. Часть этих разрядов (M) используется для образования адреса ПЗУ, другая часть (N–M) служит для формирования выходных сигналов. Входные сигналы (L) поступают на входы регистра и используются совместно с частью выходных разрядов ПЗУ для получения адреса ПЗУ.

Схема работает следующим образом. В каждом такте ПЗУ выдает код данных, тем самым определяя не только состояние выходных сигналов схемы, но и адрес ПЗУ, который установит-

ся в следующем такте (после следующего положительного фронта тактового сигнала). На этот следующий адрес влияют также и входные сигналы. То есть в отличие от формирователя последовательности сигналов, рассмотренного в предыдущем разделе, в данном случае адреса могут перебираться не только последовательно (с помощью счетчика), но и в произвольном порядке, определяемом прошивкой ПЗУ, называемой микропрограммой.

Условием правильной работы схемы будет следующее. За один период тактового сигнала должны успеть сработать регистр и ПЗУ. То есть сумма задержки регистра и задержки выборки адреса ПЗУ не должна превышать периода тактового сигнала. Отметим также, что входные сигналы микропрограммного автомата нельзя подавать непосредственно на адресные входы ПЗУ (без регистра), так как их асинхронное (по отношению к тактовому сигналу) изменение может вызвать переходный процесс на выходах данных ПЗУ именно в тот момент, когда выходные сигналы ПЗУ записываются в регистр. В результате в регистр может записаться неверная информация, что нарушит работу всей схемы. Регистр же синхронизирует изменения входных сигналов с тактовым сигналом, в результате чего все разряды кода адреса ПЗУ меняется одновременно — по положительному фронту тактового сигнала. Регистр должен обязательно срабатывать по фронту, использование регистра-защелки не допускается, так как он может вызвать лавинообразный переходный процесс.

Возможности такой простой схемы оказываются очень большими. Например, микропрограммный автомат может выдавать последовательности выходных сигналов в ответ на определенное изменение входных сигналов. Он может также временно остановить выдачу выходных сигналов до прихода входных сигналов. Он может анализировать длительность входного сигнала и в зависимости от нее выдавать те или иные выходные сигналы. Он может также и многое другое.

Сформулируем несколько элементарных функций, из которых могут складываться алгоритмы работы микропрограммного автомата (рис. 6.16).

1. Последовательный перебор адресов ПЗУ (например, для выдачи последовательности выходных сигналов).

2. Периодическое повторение последовательности адресов ПЗУ (например, для повторения последовательности выходных сигналов).
3. Остановка в каком-то адресе ПЗУ (например, для ожидания изменения входного сигнала).
4. Временное отключение реакции на входные сигналы (например, для того, чтобы закончить отработку реакции на предыдущее изменение входных сигналов). Эта функция на рисунке не показана.

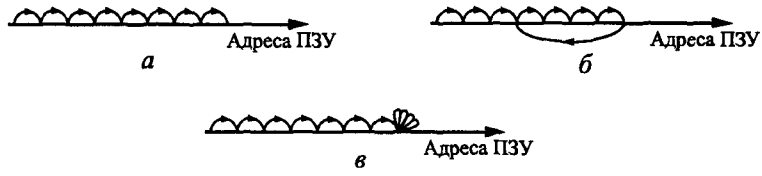


Рис. 6.16. Элементарные функции микропрограммного автомата: последовательный перебор (*а*), циклическое повторение (*б*) и остановка (*в*).

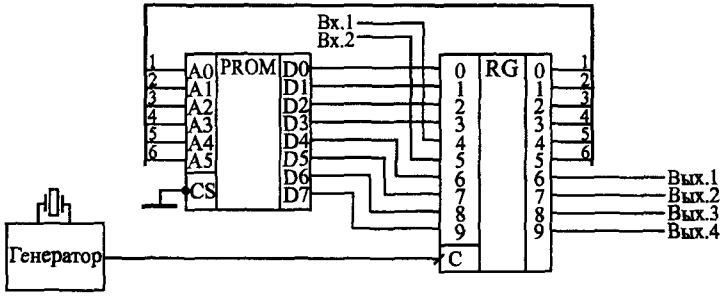


Рис. 6.17. Пример практической схемы микропрограммного автомата на ПЗУ.

В качестве примера рассмотрим выполнение некоторых элементарных функций микропрограммным автоматом, показанным на рис. 6.17.

ПЗУ имеет организацию 64 × 8 (это могут быть, например, две микросхемы РЕ3 или одна микросхема РТ18), количество разрядов регистра равно десяти (например, это может быть две микросхемы ИР27). Схема имеет два входных сигнала и четыре

выходных сигнала. Такая организация микропрограммного автомата хороша тем, что его микропрограмма (то есть карта прошивки ПЗУ) очень наглядна, легко составляется и читается.

Выходные разряды данных ПЗУ делятся на две группы по 4 сигнала: младшие идут на образование следующего адреса ПЗУ, старшие образуют четыре выходных сигнала. Входные сигналы поступают (через регистр) на два старших разряда адреса ПЗУ. Если мы изобразим карту прошивки ПЗУ в привычном для нас виде таблицы из четырех строк и 16 столбцов (табл. 6.7), то в этой таблице будут наглядно видны как состояние каждого из внутренних сигналов, так и реакция схемы на любой входной сигнал, а также состояния всех выходных сигналов в каждом такте.

Таблица 6.7. Карта прошивки ПЗУ 64 × 8 для микропрограммного автомата

Адрес	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
10	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
20	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
30	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Прежде всего, легко заметить, что выбор той или иной строки таблицы производится старшими разрядами кода адреса ПЗУ, то есть входными сигналами микропрограммного автомата. Таким образом, любое изменение входных сигналов приводит к переходу в карте прошивки на другую строку. Например, строка 00 будет соответствовать нулевым входным сигналам микропрограммного автомата, а строка 30 — единичным входным сигналам.

Теперь посмотрим на структуру 8-разрядного кода данных ПЗУ. Младшие четыре разряда этого кода (правый, младший знак 16-ричного кода в таблице) соответствуют четырем младшим разрядам кода адреса ПЗУ. Старшие четыре разряда кода данных ПЗУ (левый, старший знак 16-ричного кода в таблице) соответствуют четырем выходным сигналам микропрограммного автомата. То есть непосредственно по 16-ричному коду данных из таблицы можно сказать, во-первых, каким будет следующий адрес ПЗУ, и во-вторых, какими будут выходные сигналы автомата в следующем такте.

Рассмотрим пример микропрограммы (табл. 6.8), реализующей некоторые элементарные функции.

Таблица 6.8. Пример микропрограммы для схемы на рис. 6.17

Адрес	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF	00
10	11	22	33	44	55	66	77	88	99	55	BB	CC	DD	EE	FF	00
20	10	21	32	43	54	65	76	87	98	A9	BA	CB	DC	ED	FE	0F
30	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	00

Верхняя (нулевая) строка таблицы демонстрирует последовательный перебор адресов памяти при нулевых входных сигналах. Пусть, например, автомат находится в адресе 00. В ячейке с адресом 00 указан следующий адрес 1 (младший знак 16-ричного кода 11), то есть в следующем такте автомат перейдет в адрес 01 (считаем, что входные сигналы остаются нулевыми). Из адреса 01 автомат перейдет в адрес 02, так как в ячейке с адресом 01 указан следующий адрес 2. Точно так же из адреса 02 автомат перейдет в адрес 03 и так далее до адреса 0F, в котором указан следующий адрес 0, то есть в следующем такте автомат снова вернется в адрес 00. Затем цикл последовательного прохождения адресов первой строки повторится (если, конечно, входные сигналы останутся нулевыми). Четыре выходных сигнала автомата в данном случае повторяют код следующего адреса, то есть подобно 4-разрядному двоичному счетчику выдают постепенно нарастающий код.

Вторая сверху (первая) строка таблицы демонстрирует циклическое повторение группы тактов. Если, например, работа начинается с адреса 10, то микропрограммный автомат последовательно будет перебирать адреса 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, а затем вернется в адрес 15 и будет постоянно повторять группу адресов 15, 16, 17, 18, 19. В данном случае мы считаем, что входной сигнал Вх.1 постоянно равен единице, а входной сигнал Вх.2 постоянно равен нулю, что и соответствует второй сверху строке таблицы.

Посмотрим теперь, что будет делать автомат, если входной сигнал Вх.2 постоянно равен нулю, а входной сигнал Вх.1 меняет свое состояние с нуля на единицу и обратно. Такое переклю-

чение входных сигналов приводит к переходу между нулевой и первой строками таблицы с микропрограммой. Допустим, автомат начинает работу с адреса 00 (сигнал Вх.1 равен нулю). Идет последовательный перебор адресов нулевой строки. Даже если в течение первых девяти тактов сигнал Вх.1 будет меняться, на выходных сигналах автомата это никак не отразится, так как коды первых девяти адресов в нулевой и первой строках полностью совпадают. Поэтому можно сказать, что в этих первых девяти тактах мы отключили реакцию нашего автомата на входной сигнал Вх.1 за счет дублирования кодов.

Допустим теперь, что изначально нулевой входной сигнал Вх.1 стал равен единице в адресе 0A и далее не меняется. Это приведет к тому, что вместо адреса 0B автомат перейдет в адрес 1B. Затем он пройдет адреса 1C...1F, перейдет в адрес 10, дойдет до адреса 19 и начнет повторять цикл 15...19. Если же, например, в адресе 18 сигнал Вх.1 снова станет равен нулю, то вместо адреса 19 автомат попадет в адрес 09 и далее будет выполнять микропрограмму нулевой строки.

Третья сверху (вторая) строка таблицы, которая соответствует входным сигналам Вх.1 = 0, Вх.2 = 1, показывает пример остановки автомата в каждом адресе и ожидание прихода входного сигнала. Например, автомат находится в адресе 23. Следующим адресом в коде данной ячейки указан 3. Значит, если входные сигналы остаются неизменными, то автомат перейдет опять же в адрес 23 и будет оставаться в нем постоянно. Но если входной сигнал Вх.2 станет равным нулю, автомат перейдет в адрес 03 и начнет выполнять микропрограмму нулевой строки. То есть переключение входного сигнала Вх.2 из нуля в единицу при нулевом уровне Вх.1 останавливает выполнение микропрограммы нулевой строки до обратного перехода входного сигнала Вх.2 в нуль.

Наконец нижняя (третья) строка таблицы демонстрирует переход из любого адреса строки в нулевой адрес этой же строки. Пусть, например, выполняется микропрограмма нулевой строки (Вх.1 и Вх.2 — нулевые) и в адресе 06 оба входных сигнала переходят в единицу. Автомат попадает в адрес 37, а из него — в адрес 30, где и остается постоянно до изменения входных сигналов. Если затем оба входных сигнала снова станут нулевыми, то автомат перейдет в адрес 00 и начнет снова выполнять микропрограмму нулевой строки таблицы.

Иногда бывает необходимо перевести автомат в какой-то определенный адрес. Ведь при включении питания в регистре может оказаться произвольный код. Для такой инициализации автомата можно, например, использовать регистр, имеющий вход сброса -R, на который подается внешний сигнал, и тогда по этому сигналу автомат попадет в нулевой адрес. Но можно пойти и другим путем: составить микропрограмму таким образом, что автомат при включении питания сам перейдет за несколько тактов в нужный адрес независимо от того, в какой адрес он попал при включении питания.

Рассмотрим теперь пример проектирования простейшего микропрограммного автомата.

Пусть нам необходимо формировать с помощью микропрограммного автомата последовательность из трех выходных сигналов в ответ на положительный фронт одного входного сигнала (рис. 6.18).

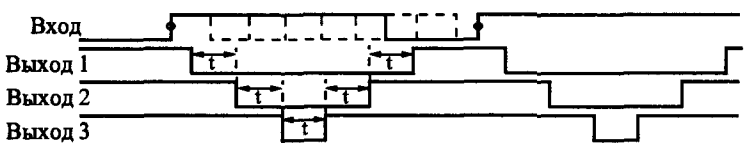


Рис. 6.18. Диаграмма работы проектируемого микропрограммного автомата.

Второй выходной сигнал «вложен» в первый, а третий выходной сигнал «вложен» во второй. Причем до тех пор, пока данная последовательность не закончится, входной сигнал может произвольно менять свое состояние, на работе схемы это никак не должно сказываться. А после окончания данной последовательности микропрограммный автомат снова должен ждать положительного фронта входного сигнала и начинать по нему выработку новой последовательности.

Все временные сдвиги в выходной последовательности (t) примем для простоты одинаковыми и равными 1 мкс. Это значительно облегчает выбор тактовой частоты микропрограммного автомата, она должна быть равной 1 МГц (период 1 мкс). В этом случае полная длина генерируемой последовательности составит всего 6 тактов (один такт соответствует переходу всех выходных сигналов в единицу).

Для формирования 6-тактовой последовательности необходимо не менее 3 адресных разрядов ПЗУ (так как $2^2 = 4$, а $2^3 = 8$). Еще один разряд адреса ПЗУ потребуется для входного сигнала схемы. Итого необходимо 4 адресных разряда ПЗУ.

Для формирования трех выходных сигналов требуется три разряда данных ПЗУ. Еще три разряда данных ПЗУ нужно для задания следующего адреса при отработке выходной последовательности. Итого необходимо шесть разрядов данных ПЗУ.

Посчитаем требуемую разрядность выходного регистра. ПЗУ выдает шесть разрядов данных. Имеется один входной сигнал. Итого регистр должен фиксировать 7 сигналов, значит, он должен быть 7-разрядным.

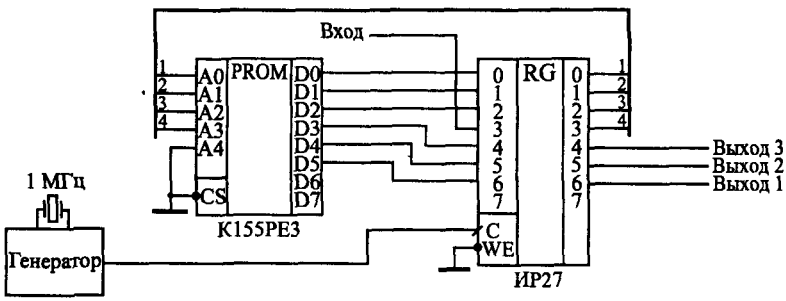


Рис. 6.19. Спроектированный микропрограммный автомат.

Итак, все расчеты закончены. Полученная схема микропрограммного автомата приведена на рис. 6.19. Она включает в себя тактовый генератор с частотой 1 МГц, микросхему ПЗУ типа РЕ3, у которой не используется один разряд адреса и два разряда данных, а также 8-разрядный регистр ИР27, у которого не используется один разряд. Для простоты на рисунке не показаны резисторы на выходах данных типа ОК микросхемы ПЗУ.

Теперь необходимо составить микропрограмму для спроектированной схемы. Она должна включать в себя ожидание положительного фронта входного сигнала, отработку последовательности выходных сигналов с временным отключением входного сигнала, ожидание нулевого уровня входного сигнала и переход на новое ожидание положительного фронта входного сигнала. Полученная микропрограмма с необходимыми комментариями представлена в табл. 6.9.

Таблица 6.9. Микропрограмма для спроектированного автомата

Адрес ПЗУ					Данные ПЗУ							Комментарий
Код	Вход	Данные			Вых.1	Вых.2	Вых.3	Сл. адрес			Код	
0	0	0	0	0	1	1	1	0	0	0	38	Ожидание полож. фронта Вх.
8	1	0	0	0	0	1	1	0	0	1	19	Пришел положительный фронт входного сигнала. Отработка последовательности. Входной сигнал равен единице.
9	1	0	0	1	0	0	1	0	1	0	0A	
A	1	0	1	0	0	0	0	0	1	1	03	
B	1	0	1	1	0	0	1	1	0	0	0C	
C	1	1	0	0	0	1	1	1	0	1	1D	
D	1	1	0	1	1	1	1	1	0	1	3D	Ожидание Вх.= 0
1	0	0	0	1	0	0	1	0	1	0	0A	Вх. стал равен нулю до окончания последовательности
2	0	0	1	0	0	0	0	0	1	1	03	
3	0	0	1	1	0	0	1	1	0	0	0C	
4	0	1	0	0	0	1	1	0	0	0	18	Переход на ожидание фронта
5	0	1	0	1	1	1	1	0	0	0	38	Переход на ожидание фронта
6	0	1	1	0	1	1	1	0	0	0	38	Не используемые ячейки
7	0	1	1	1	1	1	1	0	0	0	38	
E	1	1	1	0	1	1	1	0	0	0	38	
F	1	1	1	1	1	1	1	0	0	0	38	

Отключение реакции на входной сигнал достигается в микропрограмме за счет дублирования участка отработки выходной последовательности при входном сигнале, равном нулю, и при входном сигнале, равном единице. Если в конце выходной последовательности входной сигнал равен единице, то схема ожидает сначала перехода входного сигнала в нуль (отрицательный фронт), а затем уже ждет положительного фронта входного сигнала. Если в конце последовательности входной сигнал равен нулю, то схема сразу же переходит на ожидание положительного фронта входного сигнала.

Все неиспользуемые микропрограммным автоматом ячейки заполнены командами перехода в начальное состояние схемы с пассивными (единичными) выходными сигналами. Такое решение обеспечивает правильное начало работы микропрограммного автомата при любом начальном адресе (при любом начальном коде в регистре). Начальный сброс автомата не используется.

Последний микропрограммный автомат, который мы рассмотрим в данном разделе, предназначен для дешифрации (декодирования) последовательного кода Манчестер-II, применяющегося для последовательной передачи данных на большие расстояния, в частности в локальных сетях. Этот код уже упоминался в главе 4 (см. рис. 4.16 и 4.17). Там был рассмотрен кодировщик кода Манчестер-II.

Дешифрация (декодирование) этого кода гораздо сложнее кодирования (рис. 6.20). Она требует выделения «правильных», информационных фронтов в середине битовых интервалов (помечены на рисунке кружками) и отсечение «неправильных» фронтов между битовыми интервалами (помечены на рисунке крестиками). Для этого надо после первого (информационного) фронта в течение временного интервала 0,75T не реагировать на приходящие фронты входного сигнала, а затем снова обрабатывать любой приходящий фронт, снова выдерживать интервал 0,75T и т. д. При приходе «правильных», информационных фронтов (в середине битовых интервалов) необходимо формировать выходные синхросигналы, по которым фиксируется (в регистре) информация из сигнала в коде Манчестер-II.

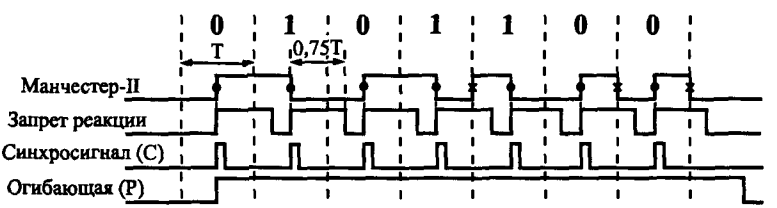


Рис. 6.20. Декодирование кода Манчестер-II.

Мы видим, что алгоритм декодирования довольно сложен. Его можно, конечно, выполнить с помощью одновибраторов и триггеров. Но можно воспользоваться и микропрограммным автоматом, который в отличие от одновибратора не требует настройки, не чувствителен к помехам и тактируется сигналом кварцевого генератора.

Схема такого микропрограммного автомата (рис. 6.21) очень проста, она включает в себя ПЗУ типа РЕ3, регистр ИР27 и

тактовый генератор с частотой, в 8 раз превышающей частоту прихода битов в коде Манчестер-II. Например, при длительности битового интервала 1 мкс (скорость передачи информации 1 Мбит/с) частота генератора должна быть равной 8 МГц. Схема практически не отличается от схемы, рассмотренной в предыдущем примере, хотя выполняемая ей функция гораздо сложнее.

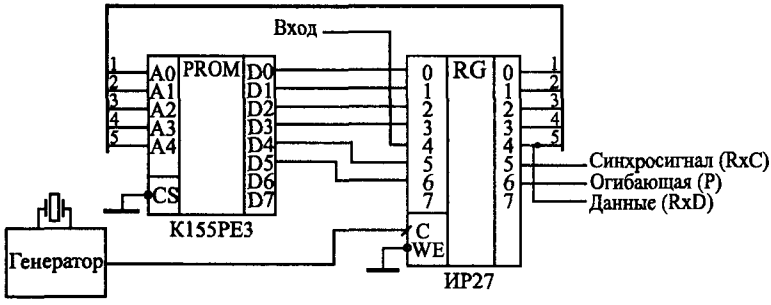


Рис. 6.21. Микропрограммный автомат для декодирования кода Манчестер-II.

Помимо синхросигнала (его обычно обозначают RxC или C) микропрограммный автомат выдает также сигнал огибающей информационного пакета (P), то есть сигнал, активный при наличии передачи информации в коде Манчестер-II и пассивный при отсутствии передачи информации (см. рис. 6.20). Сигнал C (RxC) стробирует запись сигнала данных RxD, представляющего собой просто входной сигнал в коде Манчестер-II, пропущенный через регистр.

Мы не будем подробно рассматривать составление микропрограммы для декодирования кода Манчестер-II, так как это заняло бы слишком много места. Достаточно только внимательно изучить табл. 6.10, содержащую микропрограмму с комментариями, чтобы понять, что в ней присутствуют и отключение реакции на входной сигнал, и выдержка временных интервалов (задержка), и переходы в заданные адреса, и остановки для ожидания положительного и отрицательного фронтов входного сигнала.

Таблица 6.10. Микропрограмма декодирования кода Манчестер-II (сигнал RxC обозначен в таблице С).

Адрес ППЗУ					Данные ППЗУ						Комментарий
Вх	Адрес				С	Р	След. адрес				
0	0	0	0	0	0	1	0	0	0	1	Задержка и ожидание положительного фронта входного сигнала
0	0	0	0	1	0	1	0	0	1	0	
0	0	0	1	0	0	1	0	0	1	1	
1	0	0	0	0	1	1	0	1	0	1	
1	0	0	0	1	1	1	0	1	0	1	Снятие Р и ожидание входного сигнала
1	0	0	0	0	1	1	0	1	0	1	Выставление RxC и переход на обработку положительного фронта входного сигнала
1	0	0	0	1	1	1	0	1	0	1	
1	0	0	1	0	1	1	0	1	0	1	
1	0	0	1	1	1	1	0	1	0	1	
1	0	1	0	0	1	1	0	1	0	1	
0	0	1	0	1	1	1	0	1	1	0	Снятие RxC и задержка с отключением входа
0	0	1	1	0	0	1	0	1	1	1	
0	0	1	1	1	0	1	1	0	0	0	
0	1	0	0	0	0	1	1	0	0	1	
0	1	0	0	1	0	1	0	0	0	0	Переход на ожидание положительного фронта
1	0	1	0	1	1	1	0	1	1	0	Снятие RxC и задержка с отключением входа
1	0	1	1	0	0	1	0	1	1	1	
1	0	1	1	1	0	1	1	0	0	0	
1	1	0	0	0	0	1	1	0	0	1	
1	1	0	0	1	0	1	1	0	1	0	Переход на ожидание отрицательного фронта
1	1	0	1	0	0	1	1	0	1	1	Задержка и ожидание отрицательного фронта входного сигнала
1	1	0	1	1	0	1	1	1	0	0	
1	1	1	0	0	0	1	1	1	0	1	
1	1	1	0	1	0	1	1	1	1	0	
1	1	1	1	0	0	0	1	1	1	0	Снятие Р и ожидание входного сигнала
0	1	0	1	0	1	1	0	1	0	1	Выставление RxC и переход на обработку отрицательного фронта входного сигнала
0	1	0	1	1	1	1	0	1	0	1	
0	1	1	0	0	1	1	0	1	0	1	
0	1	1	0	1	1	1	0	1	0	1	
0	1	1	1	0	0	0	0	1	0	0	Переход на ожидание положительного фронта

6.2. Оперативная память

Основное отличие оперативной памяти (RAM) от постоянной (ROM) состоит в возможности оперативного изменения содержимого всех ячеек памяти с помощью дополнительного управляющего сигнала записи WR. Каждая ячейка оперативной (*статической*) памяти представляет собой, по сути, регистр из триггерных ячеек, в который может быть записана информация и из которого можно информацию читать. Выбор того или иного регистра (той или иной ячейки памяти) производится с помощью кода адреса памяти. Поэтому при выключении питания вся информация из оперативной памяти пропадает (стирается), а при включении питания информация в оперативной памяти может быть произвольной.

Отметим, что существует также еще одна разновидность оперативной памяти, так называемая *динамическая* (в отличие от статической), в которой информация хранится не в регистрах (не в триггерных ячейках), а в виде заряда на конденсаторах. Эта память отличается более низкой стоимостью, меньшим быстродействием и необходимостью регулярной регенерации (Refresh — освежение) информации в ней (так как конденсаторы со временем разряжаются). Область применения динамической памяти гораздо уже, чем статической, в основном она применяется в качестве системной оперативной памяти компьютеров, где соображения стоимости выходят на первый план. Поэтому здесь мы о ней говорить не будем. Хотя многие особенности использования статической памяти относятся и к динамической памяти.

Во всех рассмотренных в предыдущем разделе схемах постоянная память в принципе может быть заменена оперативной, только карту прошивки в данном случае придется записывать в память каждый раз заново после включения питания. Точно так же многое из сказанного в данном разделе про оперативную память справедливо и для постоянной памяти, но только информацию в постоянной памяти менять невозможно. Однако существуют также и специфические области применения оперативной памяти, которым и будет уделено здесь особое внимание.

Как уже отмечалось, оперативная память бывает двух основных видов: с раздельными шинами входных и выходных данных (в основном, это одноразрядная память) и с двунаправ-

ленной (совмещенной) шиной входных и выходных данных (это многоразрядная память). Некоторые простейшие примеры микросхем памяти обоих этих видов приведены на рис. 6.22. Выходы данных микросхем памяти имеют тип ОК (довольно редко) или 3С. Управляющие сигналы — это сигнал выбора микросхемы CS (иногда их несколько), сигнал записи WR (обычно отрицательный) и иногда сигнал разрешения выхода OE.

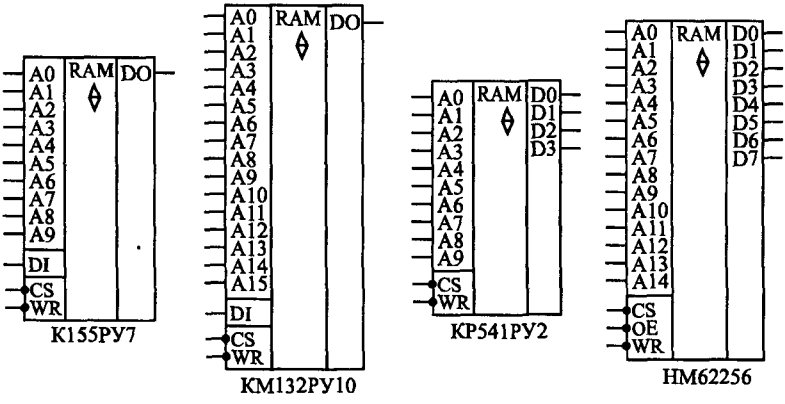


Рис. 6.22. Примеры микросхем статических ОЗУ.

Микросхема оперативной памяти К155РУ7 (аналог — F9342APC) имеет организацию 1К × 1 и раздельные входной и выходной сигналы данных. Выход микросхемы — типа 3С. Управление работой микросхемы производится двумя управляющими сигналами -CS и -WR. Режимы работы микросхемы приведены в табл. 6.11.

Таблица 6.11. Режимы работы оперативной памяти К155РУ7

Входы и выходы					Режим работы
-CS	-WR	A0...A9	DI	DO	
1	X	X	X	3С	Хранение
0	0	Адрес	0	3С	Запись 0
0	0	Адрес	1	3С	Запись 1
0	1	Адрес	X	Данные	Чтение

Микросхема КМ132РУ10 отличается от К155РУ7 в основном большим объемом (организация 64К × 1) и несколько меньшим быстродействием. Назначение управляющих сигналов и таблица режимов работы у этих микросхем совпадают.

Таблица 6.12. Режимы работы оперативной памяти КР541РУ2

Входы и выходы				Режим работы
-CS	-WR	A0...A9	DIO0...DIO3	
1	X	X	3С	Хранение
0	0	Адрес	0	Запись 0
0	0	Адрес	1	Запись 1
0	1	Адрес	Читаемые данные	Чтенне

Микросхема КР541РУ2 (аналог — IM7147L-3) относится к другой разновидности микросхем памяти. У нее четыре двунаправленных вывода данных типа 3С. Управляющие сигналы те же самые: -CS и -WR. Таблица режимов работы (табл. 6.12) также похожа на таблицу для одноразрядных микросхем. Главное отличие состоит в том, что в режиме записи на входах/выходах данных присутствует записываемая информация.

Микросхема HM62256 фирмы Hitachi отличается от КР541РУ2 прежде всего организацией (32К × 8) и управляющими сигналами (добавлен сигнал разрешения выхода -OE). Когда этот сигнал пассивен (равен единице), входы/выходы данных микросхемы находятся в состоянии 3С независимо от режима работы. Введение дополнительного сигнала позволяет более гибко управлять работой микросхемы. К тому же обычно в подобных микросхемах при пассивном сигнале -CS (равном единице) значительно уменьшается потребляемая мощность.

В настоящее время имеется огромный выбор микросхем памяти с разным объемом (от нескольких байт до нескольких мегабайт), с разным количеством разрядов (обычно 1, 4, 8, 16 разрядов), с разными методами управления, с разным потреблением и быстродействием. В каждом конкретном случае надо подбирать оптимальную память, в наибольшей степени удовлетворяющую требованиям решаемой задачи.

Таблицы режимов работы (таблицы истинности) микросхем памяти не дают достаточно информации для их практического

использования. Для микросхем памяти очень важны временные параметры (задержки сигналов относительно друг друга) и порядок выставления и снятия сигналов адреса, данных и управления. Всю эту информацию дают временные диаграммы циклов записи в память и чтения (считывания) из памяти, приводимые в справочниках. Самые главные временные параметры оперативной памяти следующие:

- время выборки адреса (задержка между изменением адреса и выдачей данных);
- время выборки микросхемы (задержка выдачи данных по выставлению сигнала -CS);
- минимальная длительность сигнала записи -WR;
- минимальная длительность сигнала -CS.

Всего же количество временных параметров может достигать двух-трех десятков, но мы не будем подробно останавливаться на этом, так как вся подобная информация имеется в многочисленных справочниках. Характерные величины всех временных параметров памяти составляет от единиц и даже долей наносекунд до десятков наносекунд.

Типичные временные диаграммы циклов записи и чтения приведены на рис. 6.23. Конкретные временные диаграммы для каждого типа памяти можно найти в справочниках.

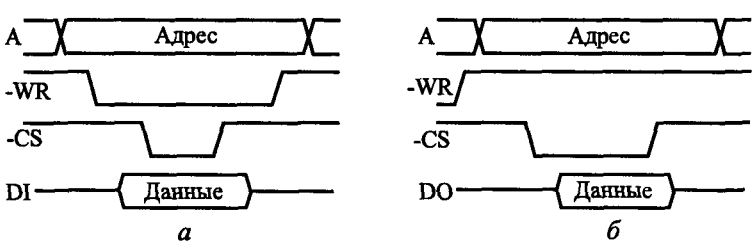


Рис. 6.23. Типичные временные диаграммы записи в память (*а*) и чтения из памяти (*б*).

Для записи информации в память надо выставить код адреса на адресных входах, выставить код записываемых в этот адрес данных на входах данных, подать сигнал записи -WR и подать сигнал выбора микросхемы -CS. Порядок выставления сигналов может быть различным, он может быть существенным или не-

существенным (например, можно выставлять или снимать -CS раньше или позже выставления или снятия -WR). Собственно запись обычно производится сигналом -WR или -CS, причем данные должны удерживаться в течение всего сигнала -WR (или -CS) и заданное время после его окончания.

Сигнал -CS у некоторых микросхем памяти допускается поддерживать активным (нулевым) для всех записываемых адресов и при этом подавать импульсы -WR для каждого адреса. Точно так же у некоторых микросхем допускается поддерживать активным (нулевым) сигнал записи -WR, но при этом подавать импульсы -CS.

В случае микросхем памяти с двунаправленной шиной данных необходимо использовать источник записываемых данных с выходом 3С или ОК, чтобы избежать конфликта данных, записываемых в память с данными, выдаваемыми из памяти в режиме чтения.

Для чтения информации из памяти надо выставить код адреса читаемой ячейки и подать сигналы -CS и -OE (если он имеется). Сигнал -WR в процессе чтения должен оставаться пассивным (равным единице). В некоторых микросхемах памяти (называемых *нетактируемыми*, например К155РУ7, КР541РУ2, НМ62256) можно держать активным (нулевым) сигнал -CS для всех читаемых адресов. В других микросхемах (называемых *тактируемыми*, например, КМ132РУ10, К537РУ8) необходимо подавать свой импульс -CS для каждого читаемого адреса. Понятно, что нетактируемые микросхемы гораздо удобнее в применении, чем тактируемые.

Микросхемы оперативной памяти довольно часто объединяются для увеличения разрядности данных или разрядности адреса.

На рис. 6.24 показано объединение четырех микросхем К155РУ7 для получения памяти с организацией 1К × 4. Точно так же могут быть объединены и микросхемы с двунаправленной шиной данных. Например, из четырех микросхем памяти с организацией 1К × 4 можно получить память с организацией 1К × 16.

Для увеличения количества адресных разрядов используются те же методы, что и в случае ПЗУ (см. рис. 6.4). Если объединяются всего две микросхемы памяти, то можно обойтись без применения дешифраторов, выбирающих одну из объединяемых микросхем.

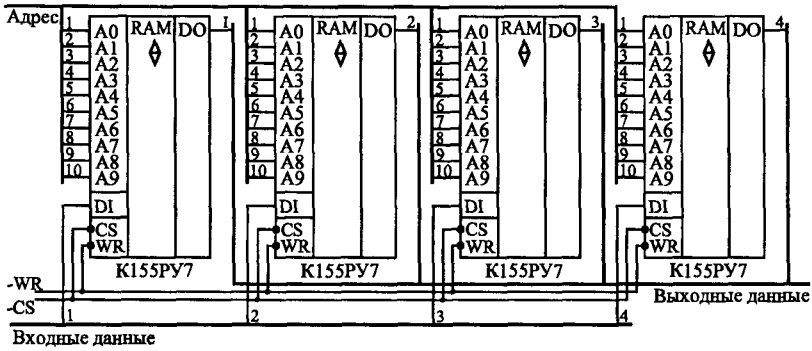


Рис. 6.24. Объединение микросхем памяти для увеличения разрядности шины данных.

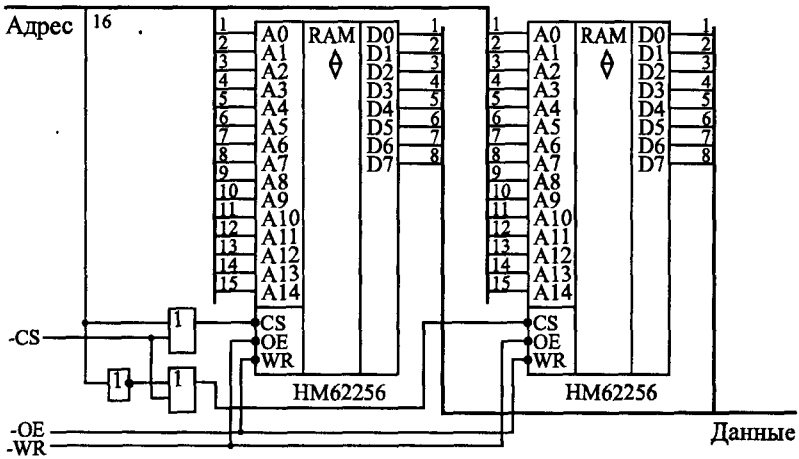


Рис. 6.25. Объединение микросхем памяти для увеличения разрядности шины адреса.

На рис. 6.25 показан вариант схемы объединения двух микросхем HM62256 для получения памяти с организацией 64К × 8. Дополнительный старший адресный разряд управляет прохождением сигнала -CS на одну из микросхем (при нулевом уровне на дополнительном адресном разряде сигнал -CS проходит на левую по рисунку микросхему, при единичном уровне — на правую микросхему).

Интересной особенностью микросхем оперативной памяти является возможность произвольного изменения порядка сигналов адресных разрядов без всяких последствий для функционирования памяти. Например, сигнал, поступающий на разряд A0 можно с тем же успехом подавать на A7, сигнал, приходящий на A7 подавать на A3, сигнал, приходящий на A3 подавать на A10 и т. д. Дело в том, что информация в оперативную память записывается по тем же самым адресам, по которым потом и читается. И перестановка адресных разрядов изменяет только номер ячейки, в которую записывается информация и из которой затем читается эта же информация. Такая взаимозаменяемость адресных входов оперативной памяти бывает полезной при проектировании разводки печатных плат. В случае ПЗУ это правило не работает, так как там информация в ПЗУ записана раз и навсегда, и читать ее надо по тем же адресам, по которым ее ранее записали.

6.2.1. ОЗУ для временного хранения информации

Главное применение микросхем оперативной памяти, непосредственно следующее из ее названия, — это временное хранение цифровой информации, всевозможных массивов кодов, таблиц данных, одиночных чисел и т. д. Цель такого хранения информации состоит в том, чтобы в любой момент иметь возможность быстро ее прочитать для дальнейшей обработки, записи в энергонезависимую память (в ПЗУ, на магнитные носители) или для другого использования. Записанная в оперативную память и непрочитанная затем информация не имеет смысла, так как при выключении питания она безвозвратно пропадет.

То есть временное хранение предполагает, что к памяти имеется возможность доступа от какого-то устройства или от какой-то другой части схемы как с операцией записи, так и с операцией чтения (считывания). В зависимости от того, в каком порядке может записываться или читаться информация, существуют две разновидности ОЗУ:

- ОЗУ с параллельным или произвольным доступом (это наиболее универсальная схема);
- ОЗУ с последовательным доступом (это более специфическая схема).

Параллельный или произвольный доступ наиболее прост и обычно не требует никаких дополнительных элементов, так как именно на этот режим непосредственно рассчитаны микросхемы памяти. В этом режиме можно записывать информацию в любой адрес ОЗУ и читать информацию из любого адреса ОЗУ в произвольном порядке. Однако параллельный доступ требует формирования довольно сложных последовательностей всех входных сигналов памяти. То есть для записи информации необходимо сформировать код адреса записываемой ячейки и только потом подать данные, сопровождаемые управляющими сигналами -CS и -WR (см. рис. 6.23). Точно так же необходимо подавать полный код адреса читаемой ячейки при операции чтения. Этот режим доступа чаще всего применяется в компьютерах и контроллерах, где самыми главными факторами являются универсальность и гибкость использования памяти для самых разных целей.

Последовательный доступ к памяти предполагает более простой порядок общения с памятью. В этом случае не надо задавать код адреса записываемой или читаемой ячейки, так как адрес памяти формируется схемой автоматически. Для записи информации надо всего лишь подать код записываемых данных и сопроводить его стробом записи. Для чтения информации надо подать строб чтения и получить читаемые данные. Автоматическое задание адреса при этом осуществляется внутренними счетчиками, меняющими свое состояние по каждому обращению к памяти. Например, десять последовательных циклов записи запишут информацию в десять последовательно расположенных ячеек памяти. Недостаток такого подхода очевиден: мы не имеем возможности записывать или читать ячейки с произвольными адресами в любом порядке. Зато существенно упрощается и ускоряется процедура обмена с памятью (запись и чтение). Мы будем в данном разделе рассматривать именно этот тип памяти, этот тип доступа.

Можно выделить три основных типа оперативной памяти с последовательным доступом:

- память типа «первый вошел — первый вышел» (FIFO, First In — First Out);
- память магазинного, стекового типа, работающая по принципу «последний вошел — первый вышел» (LIFO, Last In — First Out);
- память для хранения массивов данных.

Два первых типа памяти подразумевают возможность чередования операций записи и чтения в памяти. При этом память FIFO выдает читаемые данные в том же порядке, в котором они были записаны, а память LIFO — выдает читаемые данные в порядке, обратном тому, в котором они были записаны в память. Память FIFO можно сравнить со сдвиговым регистром, на выходе которого данные появляются в том же порядке, в котором они были в него записаны. А память LIFO обычно сравнивают с магазином для подачи патронов в автомате или пистолете, в котором первым выдается патрон, вставленный в магазин последним. Память с принципом LIFO используется в частности в компьютерах (стек), где она хранит информацию о параметрах программ и подпрограмм.

Для памяти FIFO требуется хранение двух кодов адреса (адрес для записи и адрес для чтения), для памяти LIFO достаточно одного кода адреса.

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

Рассмотрим несколько примеров схем, реализующих перечисленные типы памяти с последовательным доступом.

На рис. 6.26 представлена функциональная схема памяти типа FIFO на микросхемах с раздельными шинами входных и выходных данных. Адреса памяти задаются двумя счетчиками — счетчиком записи и счетчиком чтения, выходные коды которых мультиплексируются с помощью 2-канального мультиплексора. Запись данных осуществляется по стробу записи -Зап., чтение данных — по стробу чтения -Чт. Своим задним фронтом сигнал -Зап. переключает счетчик записи, а задний фронт сигнала -Чт. переключает счетчик чтения. В результате каждая следующая запись осуществляется в следующий по порядку адрес памяти. Точно так же каждое следующее чтение производится из следующего по порядку адреса памяти.

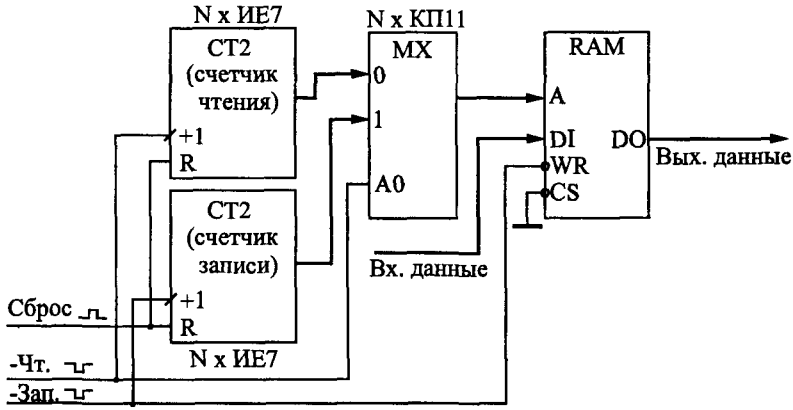


Рис. 6.26. Функциональная схема памяти типа FIFO

Перед началом работы необходимо сбросить счетчики в нуль сигналом Сброс. При отсутствии операций записи и чтения память находится в состоянии чтения (сигнал -WR равен единице), а на адресные входы памяти подается код адреса записи со счетчика записи. При подаче строба записи -Зап. производится запись входных данных по адресу из счетчика записи. Входные (записываемые) данные должны выставляться раньше начала сигнала -Зап., а заканчиваться после сигнала -Зап. При подаче строба чтения -Чт. мультиплексор переключается на передачу адреса чтения, и на выходе памяти появляется информация, считываемая из адреса чтения, задаваемого счетчиком чтения. Действительными выходные (читаемые) данные будут по заднему (положительному) фронту сигнала -Чт.

Запись начинается с нулевого адреса памяти и производится по последовательно нарастающим адресам. Точно так же чтение начинается с нулевого адреса памяти и производится по последовательно нарастающим адресам. Операции записи и чтения могут чередоваться. Временные диаграммы циклов записи и чтения показаны на рис. 6.27.

В качестве счетчиков можно использовать нужное количество микросхем ИЕ7, в качестве мультиплексора — микросхемы КП11, в качестве памяти — К155РУ7 или любые другие нетактируемые микросхемы памяти с раздельными шинами входных и выходных данных.

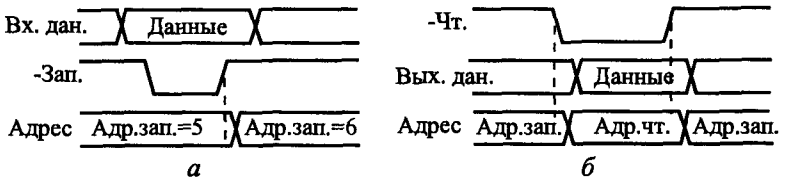


Рис. 6.27. Временные диаграммы циклов записи (*а*) и чтения (*б*) для памяти типа FIFO.

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

Функциональная схема памяти типа LIFO (рис. 6.28) проще по структуре, чем схема памяти FIFO, так как она содержит только один счетчик и не требует мультиплексирования. В данном случае считаем, что используется память с двунаправленной шиной входных/выходных данных.

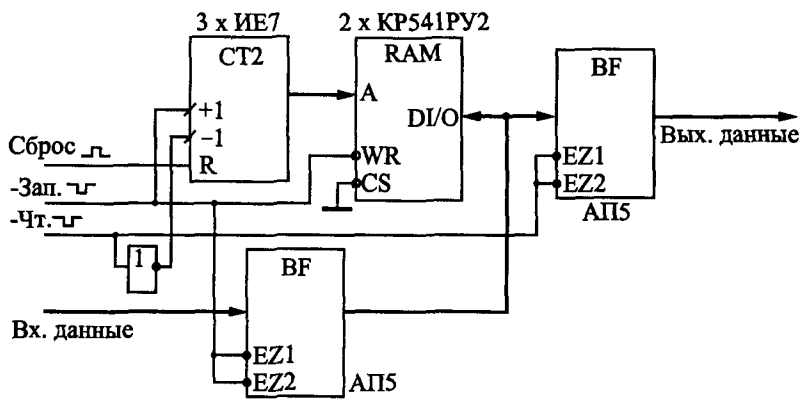


Рис. 6.28. Функциональная схема памяти типа LIFO.

Счетчик адреса необходим реверсивный, с раздельными тактовыми входами прямого и обратного счета (например, ИЕ7). После проведения цикла записи по заднему фронту сигнала -Зап. счетчик увеличивает свой выходной код (адрес памяти) на единицу. Перед проведением цикла чтения по переднему фронту сигнала -Чт. счетчик уменьшает свой выходной код на единицу. Такая организация перебора адресов позволяет организовать чтение из памяти в порядке, обратном порядку записи в память.

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

Исходное состояние счетчика в данной схеме вообще-то не слишком важно, так как не важен текущий адрес памяти, в который производится запись и из которого потом производится чтение. Однако в случае, когда используется начальный сброс счетчика в нулевое состояние (по сигналу Сброс), можно довольно просто организовать контроль за переполнением памяти LIFO из-за слишком большого количества записанной в нее информации. Для контроля переполнения можно использовать выходной сигнал переноса старшего счетчика (>15).

Временные диаграммы циклов записи и чтения приведены на рис. 6.29.

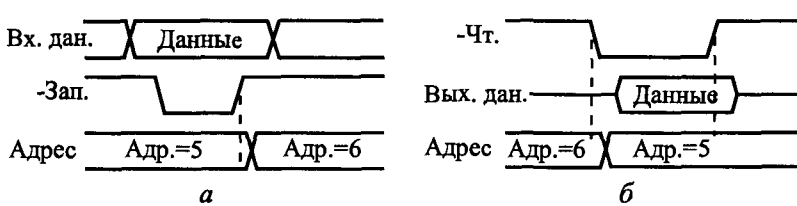


Рис. 6.29. Временные диаграммы циклов записи (*а*) и чтения (*б*) для памяти типа LIFO.

В цикле записи по сигналу -Зап. открывается входной буфер АП5 и входные (записываемые) данные поступают на входы/выходы памяти. Одновременно по этому же сигналу -Зап. память переходит в режим записи. В результате в текущий адрес памяти записываются входные данные, после чего адрес увеличивается на единицу. Входные данные должны начинаться до начала сигнала -Зап. и заканчиваться после его окончания.

В цикле чтения по сигналу -Чт. адрес уменьшается на единицу, после чего открывается выходной буфер АП5, выдающий на выход схемы читаемую из памяти информацию. Применение выходного буфера не обязательно, однако он предотвращает прохождение на шину выходных данных информации, записываемой в память в цикле записи. Выходные данные действительны по заднему фронту сигнала -Чт.

Условия правильной работы схемы следующие. Длительность сигнала записи должна быть не меньше минимально допустимой длительности сигнала -WR памяти. Период следования сигналов -Зап. не должен быть меньше суммы времени срабатывания счетчиков и длительности сигнала -Зап. Длительность сигнала чтения должна быть не менее суммы времени срабатывания счетчика, времени выборки адреса памяти и задержки выходного буфера данных. Период следования сигналов -Чт. также не должен быть меньше этой же суммы. Память должна быть нетактируемой, например КР541РУ2.

Наконец, третий тип памяти для временного хранения данных — память для хранения массивов данных. Рассмотрим вариант схемы такой памяти типа FIFO (рис. 6.30).

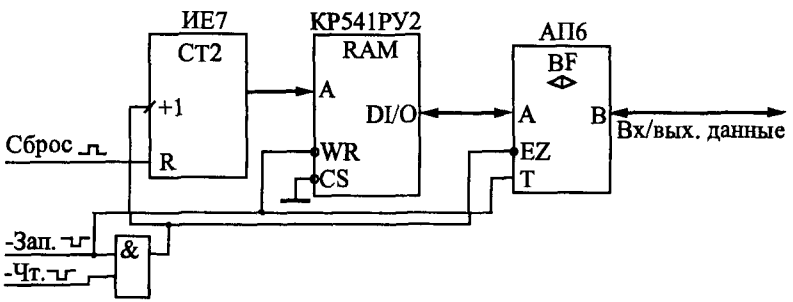


Рис. 6.30. Функциональная схема памяти для хранения массивов данных.

Адреса памяти в данном случае задаются одним единственным счетчиком, который работает в режиме только прямого счета. Перед началом работы необходимо сбросить счетчик (сигнал Сброс). Затем производится запись массива данных. При этом после каждого цикла записи по заднему фронту сигнала -Зап. выходной код счетчика увеличивается на единицу. После окончания записи всего массива снова надо сбросить счетчик в нуль (сигнал Сброс), а затем производить чтение массива, начиная с нулевого адреса. При этом после каждого цикла чтения по заднему фронту сигнала -Чт. выходной код счетчика по-прежнему увеличивается на единицу. В результате массив данных читается в том же порядке, в каком и был записан. Контроль за длиной записываемого и читаемого массивов возлагается на внешнее по отношению к приведенной схеме устройство.

Для данной схемы должна использоваться нетактируемая память (например, КР541РУ2) с двунаправленной шиной входных/выходных данных. Считаем, что данные подаются на схему и читаются из схемы также по двунаправленной шине данных. Между памятью и этой шиной включается двунаправленный буфер (типа АП6), который может понадобиться, например, для обеспечения большого выходного тока и малого входного тока со стороны внешней двунаправленной шины данных (это типичная ситуация при построении микропроцессорных и компьютерных систем). Буфер этот открывается на передачу данных в память по сигналу -Зап. (сигнал -EZ становится равным нулю, сигнал T также нулевой) и открывается для чтения данных из памяти по сигналу -Чт. (сигнал -EZ становится равным нулю, сигнал T равен единице).

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

6.2.2. ОЗУ как информационный буфер

Второе важнейшее применение микросхем оперативной памяти состоит в организации разнообразных информационных буферов, то есть буферной памяти для промежуточного хранения данных, передаваемых между двумя устройствами или системами. Суть информационного буфера состоит в следующем: передающее устройство записывает передаваемые данные в буфер, а принимающее устройство читает принимаемые данные из буфера (рис. 6.31).

Рис. 6.31. Включение буферной памяти.

Такое промежуточное хранение позволяет лучше согласовать работу устройств, участвующих в обмене данными, повысить их независимость друг от друга, согласовать скорости передачи и приема данных.

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

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

Рис. 6.32. Двунаправленный информационный буфер.

Информационные буферы бывают однонаправленными (входными или выходными) и двунаправленными (то есть входными и выходными одновременно — рис. 6.32). Например, буферная память сетевого адаптера двунаправленная, так как она буферирует как информацию, передаваемую в сеть из компьютера, так и информацию, принимаемую из сети в компьютер. Двунаправленные буферы всегда сложнее проектировать из-за большего количества потоков данных.

Информационные буферы могут обеспечивать периодический обмен между устройствами или непрерывный обмен между ними. Примером буфера с непрерывным режимом обмена может служить контроллер видеомонитора, информация из которого постоянно выдается на видеомонитор, но может изменяться по инициативе компьютера.

Информационные буферы с периодическим режимом обмена могут быть организованы по типу FIFO или по типу LIFO.

В случае FIFO массив данных читается из памяти одним устройством в том же порядке, в каком он был записан в память другим устройством. Выпускаются даже специальные микросхемы быстродействующей буферной памяти типа FIFO, которые не имеют адресной шины и представляют собой, по сути, многоразрядный сдвиговый регистр. В отличие от обычной микросхемы сдвигового регистра, где читать вдвигаемую информацию можно только тогда, когда она продвинется по всем ячейкам регистра, информацию с выходов буфера FIFO можно начинать читать с выходов сразу же после того, как она начала записываться в его входы. Но мы будем рассматривать здесь только буферы на обычных, традиционных микросхемах памяти как более универсальные.

В случае информационного буфера LIFO массив данных читается из памяти в порядке, противоположном тому, в котором он был записан в память. Такое решение иногда позволяет проще организовать схему перебора адресов памяти.

То есть разнообразие информационных буферов огромно. Мы же рассмотрим здесь всего три примера схем буферной памяти.

Первая схема — это простейший однонаправленный буфер с периодическим режимом обмена по принципу FIFO (рис. 6.33). Одно устройство записывает информацию в буфер, на другое устройство выдается информацию из буфера. Память всегда записывается полностью, по всем адресам, и читается также полностью. Строб записи -Зап. поступает в режиме записи с частотой, необходимой для записи, строб чтения -Чт. поступает при чтении с частотой, необходимой для чтения. Шины данных для записи и чтения в память в случае, показанном на рисунке, отдельные.

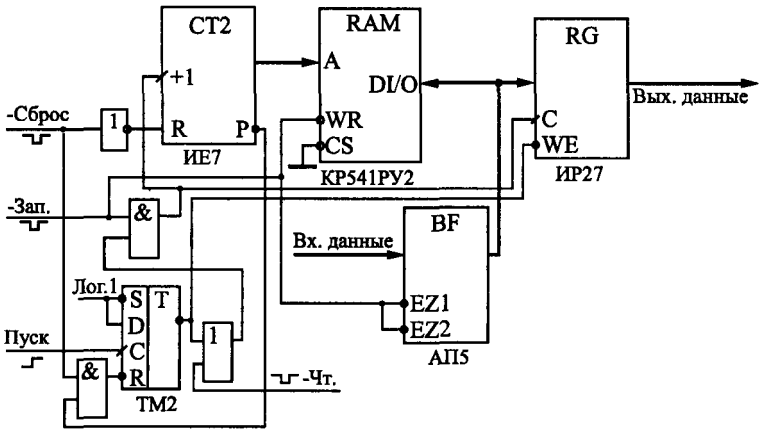


Рис. 6.33. Однонаправленный буфер типа FIFO.

При таких условиях необходим всего лишь один счетчик для перебора адресов памяти, причем счетчик, работающий только в режиме прямого счета и имеющий вход начального сброса в нуль.

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

После окончания процедуры записи устройство, производившее запись, разрешает чтение из памяти, устанавливая в единицу триггер положительным фронтом сигнала Пуск (нуль на инверсном выходе). При этом разрешается прохождение сигнала -Чт. Адреса памяти переключаются по заднему фронту сигнала -Чт., и по этому же фронту данные, читаемые из памяти, фиксируются в выходном регистре, срабатывающем по фронту (например ИР27). Выходной регистр выполняет две функции: он не пропускает на выход данные, записываемые в память (по сигналу -WE запрещается запись в триггер), а также обеспечивает одновременность изменения всех разрядов читаемых данных. Выходная информация из-за этого регистра задерживается на один период сигнала -Чт., что необходимо учитывать. Если взять регистр со входом сброса в нуль, то можно сделать, чтобы при процедуре записи в память на выходе схемы был нулевой код.

После окончания чтения всего объема памяти вырабатывается сигнал переноса счетчика -P, который снова переводит всю схему в режим записи, сбрасывая триггер в нуль (единица на инверсном выходе). После этого записывающее внешнее устройство снова может начинать процедуру записи в память.

Условия правильной работы схемы следующие. Длительность сигнала -Зап. не должна быть менее минимальной длительности сигнала -WR памяти. Период следования сигналов -Зап. не должен быть меньше суммы длительности сигнала -Зап. и задержки переключения счетчика. Период следования сигналов -Чт. не должен быть меньше суммы времени выборки адреса памяти и задержки переключения счетчика. Память должна быть нетактируемой (например, КР541РУ2).

Более сложную структуру имеет двунаправленный буфер с периодическим режимом обмена типа LIFO. Он позволяет выдавать и принимать массивы данных произвольной длины (а не фиксированной длины, как в предыдущем случае) с заданной скоростью. Такая задача возникает в частности при проектировании адаптеров локальных сетей. Несмотря на то, что данные читаются из буфера в порядке, обратном тому, в котором они были записаны в буфер, при обмене информацией между двумя буферами это никак не сказывается.

Пусть, например, устройство 1 передает информацию в устройство 2, а в качестве промежуточного устройства (устройство 3) выступает кабель сети (рис. 6.34).

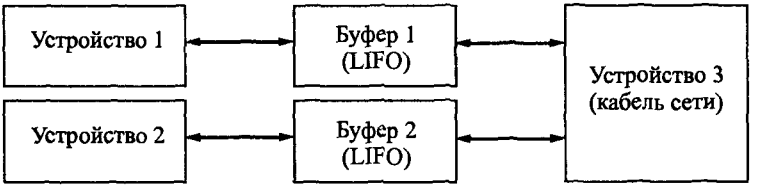


Рис. 6.34. Обмен между двумя устройствами через два буфера типа LIFO.

Устройство 1 записывает в буфер 1 массив в прямом порядке, буфер 1 выдает этот массив в устройство 3 (сеть) в обратном порядке, буфер 2 принимает массив из сети в обратном порядке, а устройство 2 читает принятую информацию опять же в прямом порядке. То есть читается информация в том же порядке, в каком она и записывалась. То же самое происходит и при передаче информации из устройства 2 в устройство 1.

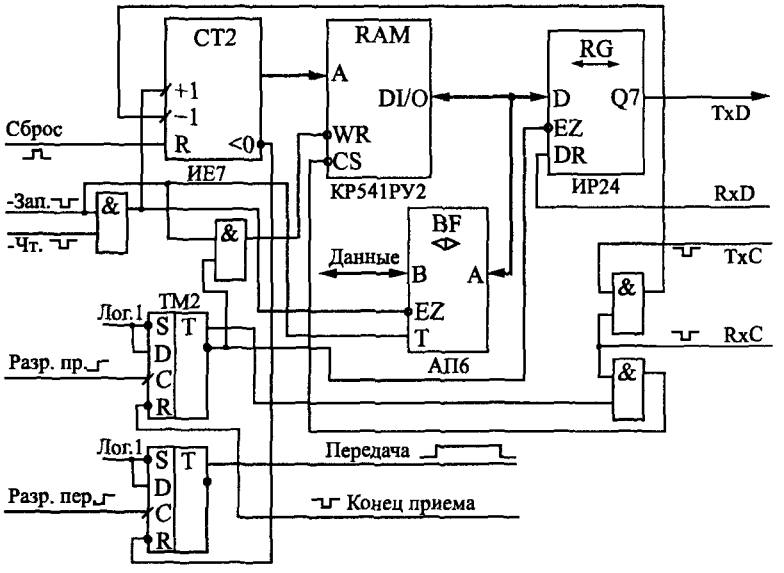


Рис. 6.35. Двунаправленный буфер типа LIFO.

Схема буфера LIFO (рис. 6.35) включает в себя помимо памяти и двунаправленного буфера реверсивный счетчик (типа ИЕ7) и реверсивный регистр сдвига (типа ИР24), служащий для

преобразования параллельного кода в последовательный при передаче в сеть и последовательного кода в параллельный при приеме из сети. Режимы работы буфера задаются двумя триггерами, один из которых разрешает режим передачи в сеть, а другой — режим приема из сети.

Перед началом работы оба триггера сброшены в нуль, счетчик также сброшен в нуль сигналом Сброс. Сначала в память записывается передаваемый в сеть массив данных. Запись производится сигналом -Зап., задний фронт которого увеличивает выходной код счетчика (адрес памяти) на единицу. После окончания записи массива сигналом Разр.пер. разрешается передача массива в сеть. В режиме передачи по сигналу строба передачи (TxC) перебираются адреса памяти в обратном порядке (счетчик работает в режиме обратного счета). Данные, читаемые из памяти, записываются в сдвиговый регистр и выдаются в сеть в последовательном коде (TxD). После того как счетчик досчитает до нуля, вырабатывается сигнал переноса <0, который сбрасывает в нуль триггер передачи. То есть в сеть выдается весь массив, записанный в память, независимо от его длины, причем массив выдается в обратном порядке.

В режиме приема информации из сети записывается единица в триггер разрешения приема по сигналу Разр.пр. Принимаемые из сети данные в последовательном коде RxD записываются в сдвиговый регистр, а из него уже в параллельном коде — в память. Запись производится по сигналу строба приема RxC, задним фронтом которого переключается счетчик, работающий в режиме инверсного счета. После окончания приема по сигналу «Конец приема» сбрасывается триггер разрешения приема. После этого производится чтение информации из памяти по сигналу -Чт. Задним фронтом этого сигнала переключается счетчик, работающий в режиме прямого счета. То есть массив читается в порядке, обратном тому, в котором он пришел из сети.

Условия правильной работы данной схемы аналогичны тем, что были сформулированы для предыдущих рассмотренных схем буферов. Сигналы стробов записи и чтения должны иметь такую длительность, чтобы осуществлять соответственно запись в память и чтение из памяти. Период следования этих сигналов должен быть таким, чтобы успевали производиться операции записи и чтения, а также успевал переключаться счетчик.

Наконец, последняя схема, которую мы рассмотрим, это буфер с непрерывным режимом работы. То есть с одним из устройств такой буфер общается непрерывно, а с другим — только в момент обращения со стороны этого устройства. В данном случае уже необходимо иметь два счетчика адреса памяти, выходные коды которых надо мультиплексировать с помощью мультиплексора.

Примем для простоты, что буфер однонаправленный и передающий, то есть одно устройство только записывает в память информацию (в нужные моменты), а на другое устройство постоянно выдается читаемая из всех подряд адресов памяти информация (рис. 6.36).

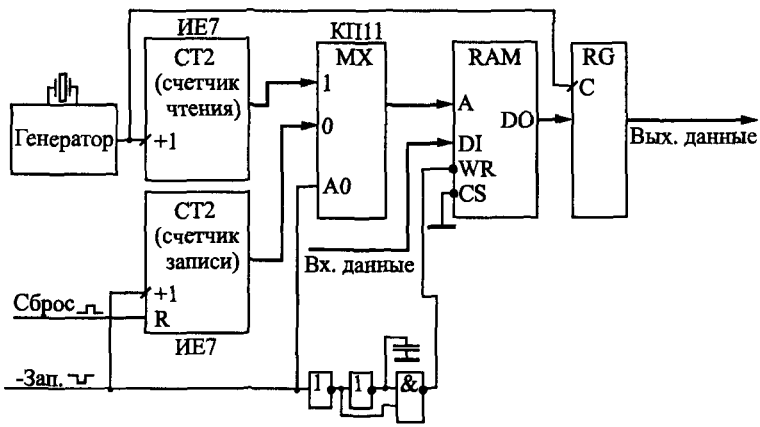


Рис. 6.36. Передающий буфер с непрерывным режимом работы.

Счетчик чтения непрерывно перебирает адреса памяти с частотой такового генератора. Читаемая из памяти информация записывается в выходной регистр и выдается на выход. В момент записи по сигналу -Зап. мультиплексор подает на адресные входы памяти выходной код счетчика записи. На память подается сигнал -WR, вложенный в сигнал -Зап. (он начинается после начала сигнала -Зап. и заканчивается раньше сигнала -Зап.). Это достигается применением цепочки из двух инверторов и элемента 2И-НЕ. Такая последовательность сигналов позволяет записать в память входные данные по адресу записи со счетчика записи и не изменять содержимое ячеек памяти с другими адресами.

Перед началом записи в память счетчик записи сбрасывается в нуль по сигналу Сброс. После каждой операции записи по заднему фронту сигнала -Зап. код на выходе счетчика записи увеличивается на единицу. То есть для записи информации во все ячейки памяти необходимо сбросить счетчик и произвести столько циклов записи, сколько ячеек имеется в памяти.

Условия правильной работы схемы следующие. Счетчики должны быть синхронными для быстрого переключения. Память должна быть нетактируемая и с раздельными входами и выходами данных. Емкость конденсатора должна быть такой, чтобы формируемый импульс -WR имел достаточную длительность для записи информации в память. За время действия сигнала -Зап. должен успеть сработать мультиплексор и должна записаться информация в память. Выходной регистр должен срабатывать по фронту. Длительность периода тактового сигнала должна быть не меньше суммы задержки выборки адреса памяти и задержки переключения счетчика чтения. За время действия сигнала -Зап. должна успеть записаться информация в память, и должен переключиться счетчик записи.

Недостаток приведенной организации буфера состоит в том, что при проведении цикла записи в память на выходе схемы будет не та информация, которая должна читаться из памяти в данный момент. Преодолеть этот недостаток можно двумя путями.

Первый путь состоит в осуществлении записи в память только в те моменты, когда выходная информация буфера не важна. Например, если речь идет о буфере контроллера видеомонитора, то запись в память можно производить только во время кадрового гасящего импульса, когда на экране ничего не отображается.

Второй путь более сложен. Он состоит в том, чтобы разделить во времени запись в память и чтение из памяти. Например, в первой половине такта (то есть периода тактового генератора) производится запись в память (если есть внешний сигнал записи), а во второй половине такта всегда производится чтение информации из памяти и запись ее в выходной регистр. Соответственно мультиплексор в первой половине периода подает на адресные входы памяти адрес записи, а во второй половине — адрес чтения. Временную привязку момента записи к ближайшей первой половине такта можно осуществить с помощью

микропрограммного автомата. При таком решении запись в память можно производить в любой момент без искажения читаемой информации, однако существенно (минимум вдвое) возрастают требования к быстродействию всех микросхем.

6.2.3. Улучшение параметров ОЗУ

При применении оперативной памяти часто встает задача улучшения ее характеристик. О совместном включении нескольких микросхем с целью увеличения разрядности шины адреса и шины данных уже говорилось. Здесь же мы остановимся на задаче повышения быстродействия памяти, то есть увеличения предельной тактовой частоты, с которой можно записывать информацию в память и читать информацию из памяти.

Наверное, самое распространенное и самое простое решение, позволяющее повысить быстродействие памяти, состоит в применении сдвиговых регистров. Сдвиговые регистры всегда имеют существенно большее быстродействие, чем память, так как они имеют гораздо более простую структуру. Частота следования тактовых импульсов этих регистров может достигать десятков и сотен мегагерц, тогда как память с такими параметрами найти трудно.

Увеличение быстродействия памяти достигается с помощью сдвиговых регистров очень просто: уменьшение в несколько раз разрядности шины данных памяти позволяет во столько же раз увеличить частоту записи информации в память или чтения информации из памяти.

Например, если необходимо в 8 раз увеличить частоту чтения информации из памяти, то надо соединить нужное количество микросхем памяти для увеличения разрядности шины данных в 8 раз, а затем применить на выходах данных схему (рис. 6.37) на основе 8-разрядного регистра сдвига. 8-разрядный код, читаемый из памяти, записывается в сдвиговый регистр, а затем сдвигается семь раз с частотой, в 8 раз большей, чем частота опроса памяти. И запись и сдвиг производятся одним тактовым сигналом с генератора. Восемь тактовых импульсов отсчитываются синхронным счетчиком. Для управления работой регистра сдвига применен элемент 3ИЛИ-НЕ, выдающий положительный импульс в течение первой 1/8 периода опроса памяти. Этот же сигнал используется как строб чтения из памяти (своим задним фронтом он переключает адреса памяти).

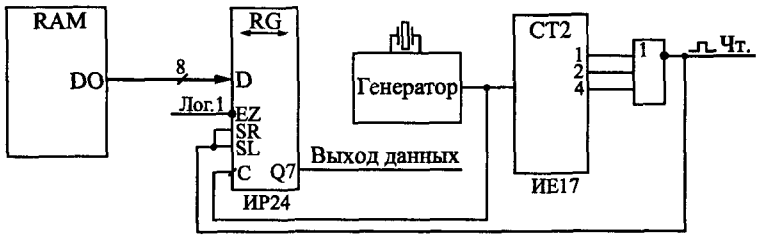


Рис. 6.37. Увеличение частоты чтения информации.

При необходимости увеличения частоты записи в память одного сдвигового регистра недостаточно. Дело в том, что информация в память записывается не по фронту сигнала, а по уровню, то есть записываемая информация должна сохраняться на входе памяти определенное время. Поэтому код с выхода сдвигового регистра необходимо перед записью в память переписать в параллельный регистр, где он будет затем храниться в течение всего периода записи в память.

Схема, показанная на рис. 6.38, ускоряет частоту записи в память в 4 раза.

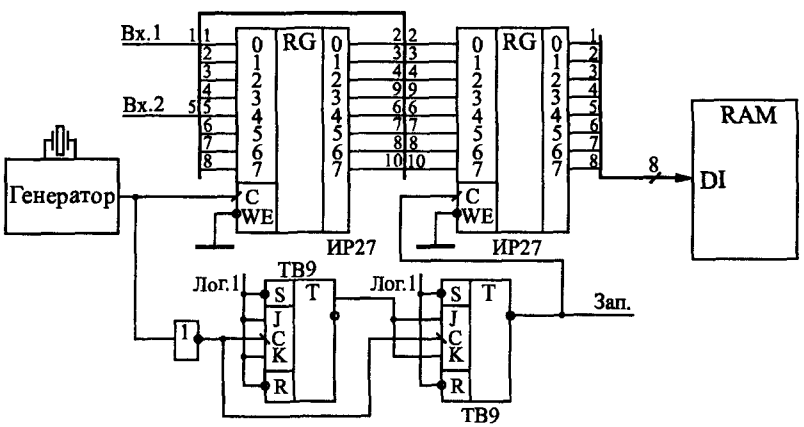


Рис. 6.38. Увеличение частоты записи информации.

В данном случае в качестве регистра сдвига удобно использовать обычный параллельный регистр, срабатывающий по фронту, у которого выходы трех разрядов соединены со входа-

ми следующих разрядов. При этом из одного 8-разрядного регистра мы получаем два 4-разрядных регистра сдвига. Входная 2-разрядная информация записывается в эти два 4-разрядных регистра сдвига, затем переписывается в параллельный регистр и только потом записывается в память. Для отсчета четырех импульсов тактового генератора применен 2-разрядный счетчик на двух JK-триггерах, включенных в счетном режиме, что позволяет несколько повысить быстродействие по сравнению со стандартными микросхемами счетчиков. Сигнал с выхода второго триггера записывает информацию в параллельный регистр, а также используется в качестве строба записи в память Зап.

Большой недостаток оперативной памяти состоит в том, что информация, записанная в нее, исчезает при выключении источника питания. Поэтому часто используется дополнительный источник питания (гальваническая батарея или аккумулятор), который питает при выключении источника питания только микросхемы памяти. В данном случае очень удобны микросхемы ОЗУ, выполненные по КМОП технологии, ток потребления которых в статическом режиме (при неизменных входных и выходных сигналах) очень мал (порядка единиц микроампер). В результате получается так называемая энергонезависимая оперативная память, содержимое которой может легко перезаписываться, но не пропадает при выключении питания, как в ПЗУ.

Схема энергонезависимой памяти (рис. 6.39) довольно проста, хотя и имеет ряд неочевидных особенностей.

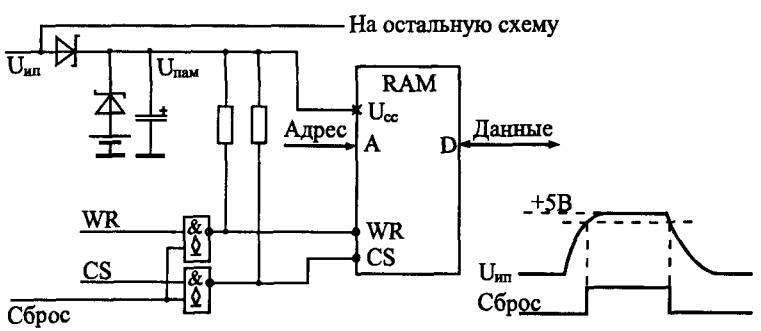


Рис. 6.39. Энергонезависимая оперативная память.

Дело в том, что управляющие сигналы памяти -WR и -CS имеют активный низкий уровень, а при выключении питания все входные сигналы памяти, естественно, станут нулевыми. Это приведет к искажению записанной в память информации. Поэтому необходимо обеспечить, чтобы при выключении питания сигналы на входах -WR и -CS были пассивными, то есть имели уровень логической единицы. Для этого обычно используются логические элементы с выходами ОК, нагрузочные резисторы которых присоединяются не к пропадающему напряжению питания памяти $U_{ип}$, а к сохраняющемуся напряжению питания памяти $U_{пам}$. Для получения напряжения $U_{пам}$ используется простая схема на двух диодах (лучше брать диоды Шоттки с меньшим падением напряжения), которая передает на выход $U_{пам}$ напряжение источника питания $U_{ип}$ (если питание включено) или напряжение от гальванической батареи 3—4,5 В (если питание выключено).

Для большей гарантии от пропадания информации во время переходных процессов (при постепенном нарастании $U_{ип}$ и при постепенном его уменьшении) необходимо управлять прохождением сигналов WR и CS на память с помощью управляющего сигнала Сброс. Этот сигнал равен нулю при напряжения $U_{ип}$ менее 4,7—4,8 В и равен единице при нормальном напряжении $U_{ип}$ = 5 В (временная диаграмма приведена на рисунке). В результате такого решения память отключается от остальной схемы при недостаточном напряжении питания (сигналы -WR и -CS равны единице) и подключается к остальной схеме при нормальном напряжении питания.

В заключение данной главы надо отметить, что в ней сознательно не рассмотрена одна из важнейших областей применения микросхем памяти (как постоянной, так и оперативной) — микропроцессорные системы и компьютерные системы. Дело в том, что говорить о применении памяти в этой области невозможно без изложения основ микропроцессорной и компьютерной схемотехники, а это отдельная большая тема, требующая специальной книги. К тому же изучение методов включения памяти, которые рассмотрены в данной главе, позволяет в дальнейшем довольно легко понять принципы применения памяти в любых возможных областях.

Глава 7

ПРИМЕНЕНИЕ МИКРОСХЕМ ЦАП И АЦП

Как уже отмечалось в первой главе, цифро-аналоговые преобразователи (ЦАП, DAC — Digital-to-Analog Converter) и аналого-цифровые преобразователи (АЦП, ADC — Analog-to-Digital Converter) главным образом применяются для сопряжения цифровых устройств и систем с внешними аналоговыми сигналами, с реальным миром. При этом АЦП преобразует аналоговые сигналы во входные цифровые сигналы, поступающие на цифровые устройства для дальнейшей обработки или хранения, а ЦАП преобразует выходные цифровые сигналы цифровых устройств в аналоговые сигналы (см. рис. 1.23).

ЦАП и АЦП применяются в измерительной технике (цифровые осциллографы, вольтметры, генераторы сигналов и т. д.), в бытовой аппаратуре (телевизоры, музыкальные центры, автомобильная электроника и т. д.), в компьютерной технике (ввод и вывод звука в компьютерах, видеомониторы, принтеры и т. д.), в медицинской технике, в радиолокационных устройствах, в телефонии и во многих других областях. При этом применение ЦАП и АЦП постоянно расширяется по мере перехода от аналоговых устройств к цифровым устройствам.

В качестве ЦАП и АЦП обычно применяются специализированные микросхемы, выпускаемые многими отечественными и зарубежными фирмами.

Сразу же надо отметить, что для грамотного и профессионального использования микросхем ЦАП и АЦП совершенно не достаточно знания цифровой схемотехники. Эти микросхемы относятся к аналого-цифровым, поэтому они требуют также знания аналоговой схемотехники, существенно отличающейся от цифровой. Практическое применение ЦАП и АЦП требует расчета аналоговых цепей, учета многочисленных погрешностей преобразования (как статических, так и динамических), знания характеристик и особенностей аналоговых микросхем (в первую очередь операционных усилителей) и многого другого, что далеко выходит за рамки этой книги. Существует обширная

литература, специально посвященная именно вопросам применения ЦАП и АЦП. Поэтому в данной главе мы не будем говорить о специфике выбора и принципах включения конкретных микросхем ЦАП и АЦП, мы будем рассматривать только основные особенности методов соединения ЦАП и АЦП с цифровыми узлами. Нас будет в первую очередь интересовать организация цифровых узлов, предназначенных для соединения с ЦАП и АЦП.

7.1. Применение ЦАП

В общем случае микросхему ЦАП можно представить в виде блока (рис. 7.1), имеющего несколько цифровых входов и один аналоговый вход, а также аналоговый выход.

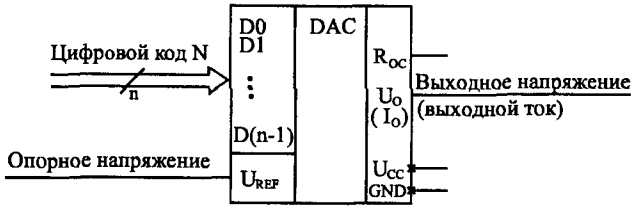


Рис. 7.1. Микросхема ЦАП.

На цифровые входы ЦАП подается n-разрядный код N, на аналоговый вход — опорное напряжение $U_{оп}$ (другое распространенное обозначение — U_{REF}). Выходным сигналом является напряжение $U_{вых}$ (другое обозначение — U_O) или ток $I_{вых}$ (другое обозначение — I_O). При этом выходной ток или выходное напряжение пропорциональны входному коду и опорному напряжению. Для некоторых микросхем опорное напряжение должно иметь строго заданный уровень, для других допускается менять его значение в широких пределах, в том числе и изменять его полярность (положительную на отрицательную и наоборот). ЦАП с большим диапазоном изменения опорного напряжения называется умножающим ЦАП, так как его можно легко использовать для умножения входного кода на любое опорное напряжение.

Кроме информационных сигналов микросхемы ЦАП требуют также подключения одного или двух источников питания и общего провода. Обычно цифровые входы ЦАП обеспечивают совместимость со стандартными выходами микросхем ТТЛ.

В случае, когда ЦАП имеет токовый выход, его выходной ток обычно преобразуется в выходное напряжение с помощью внешнего операционного усилителя и встроенного в ЦАП резистора R_{OC}, один из выводов которого выведен на внешний вывод микросхемы (рис. 7.2). Поэтому, если не оговорено иное, мы будем в дальнейшем считать, что выходной сигнал ЦАП — напряжение U_O.

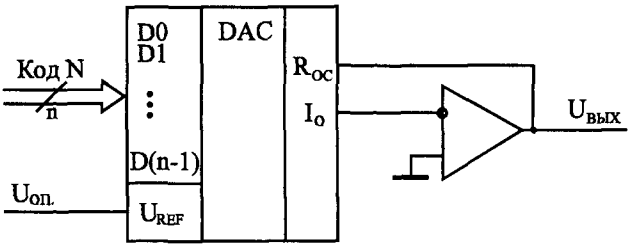


Рис. 7.2. Преобразование выходного тока ЦАП в выходное напряжение.

Суть преобразования входного цифрового кода в выходной аналоговый сигнал довольно проста. Она состоит в суммировании нескольких токов (по числу разрядов входного кода), каждый последующий из которых вдвое больше предыдущего. Для получения этих токов используются или транзисторные источники тока или резистивные матрицы, коммутируемые транзисторными ключами.

В качестве примера на рис. 7.3 показана схема реализации 4-разрядного (n = 4) цифро-аналогового преобразования на основе резистивной матрицы R-2R и ключей (в реальности используются ключи на основе транзисторов). Правому положению ключа соответствует единица в данном разряде входного кода N (разряды D0...D3). Операционный усилитель может быть как встроенным (в случае ЦАП с выходом по напряжению), так и внешним (в случае ЦАП с выходом по току).

Первым (самым левым на схеме) ключом коммутируется ток величиной $U_{REF}/2R$, вторым ключом — ток $U_{REF}/4R$, третьим ключом — ток $U_{REF}/8R$, четвертым ключом — ток $U_{REF}/16R$. То есть токи, коммутируемые соседними ключами, различаются вдвое, как и веса разрядов двоичного кода. Токи, коммутируемые всеми ключами, суммируются и преобразуют-

ся в выходное напряжение с помощью операционного усилителя с сопротивлением $R_{OC} = R$ в цепи отрицательной обратной связи.

При правом положении каждого ключа (единица в соответствующем разряде входного кода ЦАП) ток, коммутируемый этим ключом, поступает на суммирование. При левом положении ключа (нуль в соответствующем разряде входного кода ЦАП) ток, коммутируемый этим ключом, не поступает на суммирование.

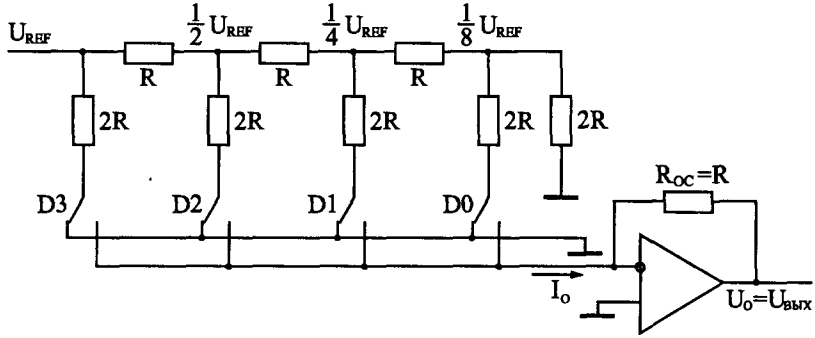


Рис. 7.3. 4-разрядное цифро-аналоговое преобразование.

Суммарный ток I_O от всех ключей создает на выходе операционного усилителя напряжение $U_O = I_O R_{OC} = I_O R$. То есть вклад первого ключа (старшего разряда кода) в выходное напряжение составляет $U_{REF}/2$, второго — $U_{REF}/4$, третьего — $U_{REF}/8$, четвертого — $U_{REF}/16$. Таким образом, при входном коде N = 0000 выходное напряжение схемы будет нулевым, а при входном коде N = 1111 оно будет равно $-15U_{REF}/16$.

В общем случае выходное напряжение ЦАП при $R_{OC} = R$ будет связано со входным кодом N и опорным напряжением U_{REF} простой формулой:

$$U_{ВЫХ} = -N \cdot U_{REF} 2^{-n},$$

где n — количество разрядов входного кода. Знак минус получается из-за инверсии сигнала операционным усилителем. Эту связь иллюстрирует также табл. 7.1.

Таблица 7.1. Таблица преобразования ЦАП в однополярном режиме

Входной код N	Выходное напряжение $U_{ВЫХ}$
000...000	0
000...001	$-2^{-n} U_{REF}$
...	...
100...000	$-2^{-1} U_{REF}$
...	...
111...111	$-(1-2^{-n}) U_{REF}$

Некоторые микросхемы ЦАП предусматривают возможность работы в биполярном режиме, при котором выходное напряжение изменяется не от нуля до U_{REF}, а от $-U_{REF}$ до $+U_{REF}$. При этом выходной сигнал ЦАП $U_{ВЫХ}$ умножается на 2 и сдвигается на величину U_{REF}. Связь между входным кодом N и выходным напряжением $U_{ВЫХ}$ будет следующей:

$$U_{ВЫХ} = U_{REF} (1 - N \cdot 2^{1-n}).$$

Эту ситуацию иллюстрирует табл. 7.2. Такое биполярное преобразование при возможности смены знака опорного напряжения называется также четырехквадрантным умножением (то есть и опорное напряжение, и выходное напряжение могут быть в данном случае как положительными, так и отрицательными).

Таблица 7.2. Таблица преобразования ЦАП в биполярном режиме

Входной код N	Выходное напряжение $U_{ВЫХ}$
000...000	U_{REF}
...	...
011...111	$2^{-n} U_{REF}$
100...000	0
...	...
111...111	$-(1-2^{1-n}) U_{REF}$

Микросхемы ЦАП, имеющиеся на рынке, различаются количеством разрядов (от 8 до 24), величиной задержки преобразования (от единиц наносекунд до единиц микросекунд), допус-

тимой величиной опорного напряжения (обычно — единицы вольт), величинами погрешностей преобразования и другими параметрами. Различаются они также технологией изготовления и особенностями внутренней структуры, что нередко накладывает ограничения на их использование. Поэтому выбирать микросхему ЦАП для конкретного применения необходимо с использованием подробной справочной информации, предоставляемой фирмами-изготовителями. Мы же будем говорить только об общих принципах включения ЦАП в цифровые схемы без учета их частных особенностей.

Иногда бывает необходимо уменьшить количество разрядов ЦАП. Для этого надо подать сигналы логического нуля на нужное число младших разрядов ЦАП (но никак не старших разрядов). На рис. 7.4 показано, как из 10-разрядного ЦАП можно сделать 8-разрядный ЦАП, подав нули на два младших разряда. Увеличение количества разрядов ЦАП представляет собой гораздо более сложную задачу, требующую построения сложных аналоговых схем, поэтому оно встречается довольно редко, значительно проще подобрать микросхему с нужным или с большим, чем нужно, количеством разрядов.

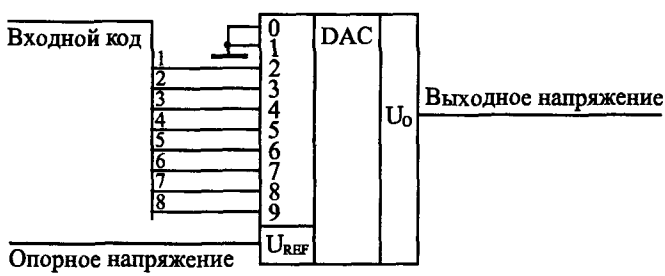


Рис. 7.4. Уменьшение разрядности ЦАП.

Основное применение микросхем ЦАП состоит в получении аналогового сигнала из последовательности цифровых кодов (рис. 7.5). Как правило, коды подаются на входы ЦАП через параллельный регистр, что позволяет обеспечить одновременность изменения всех разрядов входного кода ЦАП. При неодновременном изменении разрядов входного кода на выходе ЦАП появляются большие короткие импульсы напряжения, уровни которых не соответствуют ни одному из кодов.

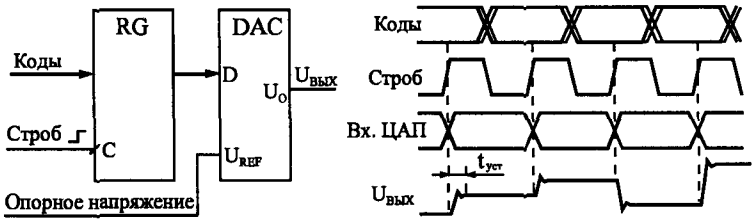


Рис. 7.5. Преобразование последовательности кодов в выходное напряжение.

Однако даже при одновременном изменении всех разрядов входного кода ЦАП уровень напряжения, соответствующий поданному коду, устанавливается не сразу, а за время установления ЦАП $t_{уст}$, что связано с неидеальностью внутренних элементов ЦАП. Выходной ток ЦАП, как правило, устанавливается значительно быстрее выходного напряжения, так как он не зависит от инерционности операционного усилителя. Понятно, что условие правильной работы ЦАП состоит в том, чтобы длительность сохранения входного кода была больше, чем время установления ЦАП $t_{уст}$, иначе выходной сигнал не успеет принять значение, соответствующее входному коду.

Если подавать коды на вход ЦАП редко, то приведенная на рис. 7.5 схема может использоваться, например, в управляемом источнике питания, выходное напряжение которого задается входным кодом. Правда, при этом необходимо еще обеспечить большой выходной ток источника питания, применив внешний усилитель тока.

Если же подавать коды на вход ЦАП с высокой частотой, то можно получить генератор (он же синтезатор) аналоговых сигналов произвольной формы. В этом случае коды, поступающие на ЦАП, называют кодами выборок (то есть мгновенных значений) генерируемого аналогового сигнала.

В простейшем случае в качестве источника входных кодов ЦАП можно использовать обычный двоичный счетчик (рис. 7.6). Выходное напряжение ЦАП будет возрастать при этом на величину $2^{-n}U_{REF}$ с каждым тактовым импульсом, формируя пилообразные выходные сигналы амплитудой U_{REF}. Длительность каждой ступеньки равна периоду тактового генератора T, а период всего выходного сигнала равен 2^nT. Количество ступенек в периоде выходного сигнала равно 2^n. Если в данной схеме исполь-

зовать синхронные счетчики с синхронным переносом, то входной регистр ЦАП не нужен, так как все разряды счетчика переключаются одновременно. Если же используются асинхронные счетчики или синхронные счетчики с асинхронным переносом, то входной регистр ЦАП необходим.

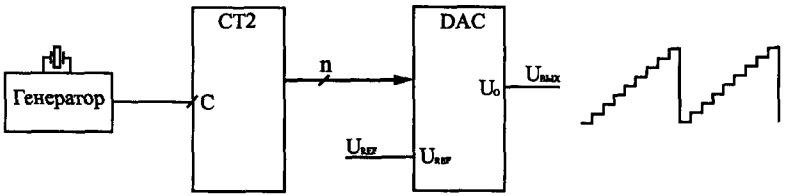


Рис. 7.6. Генератор пилообразного аналогового сигнала

Если же надо формировать аналоговые сигналы произвольной формы (синусоидальные, колоколообразные, шумовые, треугольные, импульсные и т. д.), то в качестве источника кодов, поступающих на ЦАП, необходимо использовать память, работающую в режиме чтения (рис. 7.7).

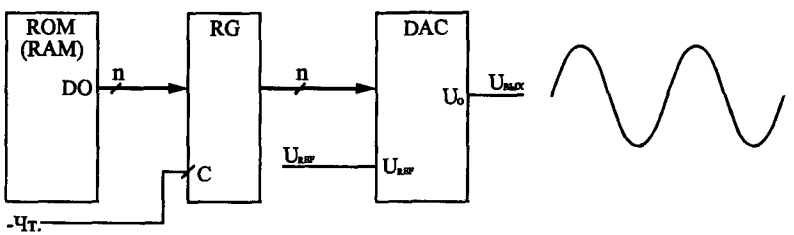


Рис. 7.7. Генерация сигналов произвольной формы.

Если память постоянная, то набор форм генерируемых сигналов задается раз и навсегда. Если же память оперативная, то строится однонаправленный информационный буфер с периодическим режимом работы, что позволит записывать в память коды для генерации самых разных сигналов. В обоих случаях входной регистр ЦАП необходим, информация в него записывается стробом чтения из памяти.

Как и в предыдущем случае, выходной сигнал ЦАП будет состоять из ступенек, высота которых кратна $2^{-n}U_{REF}$. Амплиту-

да выходного сигнала не превышает U_{REF}. Если адреса памяти перебираются счетчиком, то период выходного аналогового сигнала равен $2^m T$, где T — период тактового сигнала чтения из памяти -Чт., а m — количество адресных разрядов памяти.

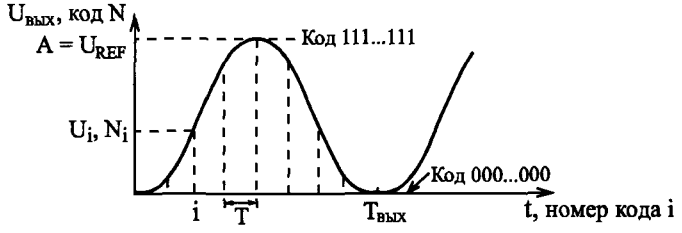


Рис. 7.8. Вычисление кодов выборок периодического сигнала.

Если надо вычислить коды выборок для генерации какого-то периодического сигнала, то необходимо его период разделить на 2^m частей и вычислить соответствующие 2^m значений этого сигнала U_i. Затем надо пересчитать значения сигнала в коды по формуле: $N_i = 2^n U_i/A$, где A — амплитуда сигнала, и взять ближайшее целое значение кода. Нулевое значение сигнала даст при этом нулевой код 000...000, максимальное значение сигнала (равное амплитуде A) даст максимальный код 111...111. В результате подачи этих кодов на ЦАП с периодом T будет генерироваться аналоговый сигнал требуемой формы с амплитудой, равной U_{REF}, и с периодом $T_{ВЫХ} = 2^m T$. Пример такого вычисления проиллюстрирован рис. 7.8.

Подробнее задача проектирования генератора аналоговых сигналов произвольной формы будет рассмотрена в следующей главе.

Преобразование цифровых кодов в аналоговый сигнал — это не единственное применение микросхем ЦАП. Они могут также использоваться для управляемой обработки аналоговых сигналов, например для усиления и ослабления аналоговых сигналов в заданное число раз. Для этого лучше всего подходят умножающие ЦАП, которые допускают изменение уровня опорного напряжения в широких пределах, в том числе и с изменением его знака. Таких микросхем ЦАП выпускается сейчас достаточно много с разным быстродействием и с разным количеством разрядов входного кода.

Самая простейшая схема — это цифровой аттенюатор (ослабитель) аналогового сигнала (рис. 7.9), часто применяемый для регулировки амплитуды выходного сигнала генератора на основе ЦАП.

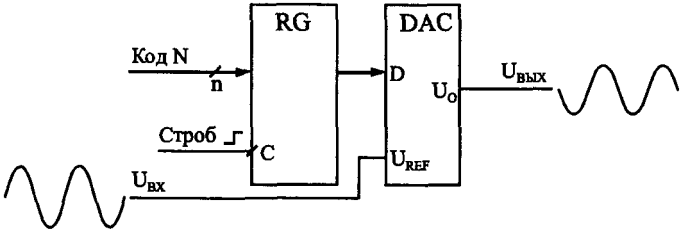


Рис. 7.9. Аттенюатор аналогового сигнала на ЦАП.

Схема практически ничем не отличается от схемы на рис. 7.5. Но есть два важных отличия: вместо постоянного опорного напряжения подается переменный аналоговый сигнал, а ЦАП должен быть обязательно умножающим. Выходной сигнал связан со входным простой формулой:

$$U_{ВЫХ} = -U_{ВХ} \cdot N \cdot 2^{-n},$$

то есть выходной сигнал пропорционален входному (с инверсией), а коэффициент пропорциональности определяется входным цифровым кодом N. Коэффициент пропорциональности изменяется в данном случае от нуля и почти до единицы с шагом, равным 2^{-n}.

Входной регистр ЦАП в данном случае также необходим, так как при неодновременном переключении разрядов входного кода на выходной сигнал ЦАП могут накладываться короткие импульсы значительной амплитуды. Требования к быстродействию ЦАП (к величине его времени установления) в данном включении не слишком высоки, так как амплитуду выходного сигнала обычно требуется менять нечасто. А частота входного аналогового сигнала может быть довольно большой, она никак не связана с временем установления ЦАП.

Существует также схема включения ЦАП, которую можно использовать как управляемый усилитель аналогового сигнала с коэффициентом усиления, задаваемым входным кодом N (рис. 7.10).

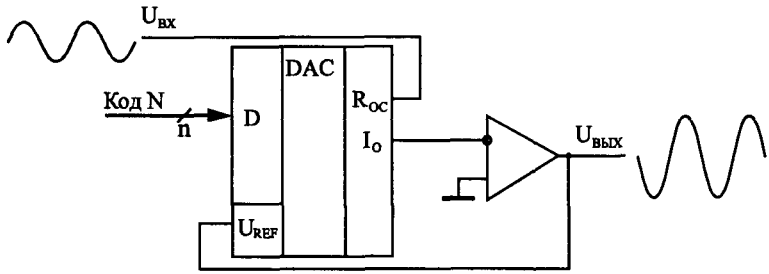


Рис. 7.10. Управляемый усилитель входного сигнала.

В этом случае выходной ток ЦАП равен величине $U_{ВХ}/R_{ОС}$, а так как в качестве опорного напряжения используется выходное напряжение, то получается, что выходное напряжение связано со входной формулой:

$$U_{ВЫХ} = -U_{ВХ} \cdot 2^n/N,$$

то есть коэффициент пропорциональности между выходным и входным напряжениями обратно пропорционален коду N. Код N может меняться в этом случае от 1 до $(2^n - 1)$, что соответствует коэффициенту усиления от примерно единицы до 2^n. Например, при 10-разрядном ЦАП коэффициент усиления схемы может достигать 1024.

Как и в предыдущем случае, скорость переключения ЦАП не очень важна, так как коэффициент усиления обычно не требуется переключать слишком часто. На схеме для простоты не показан входной регистр ЦАП, который опять же необходим, чтобы обеспечить одновременность переключения всех разрядов входного кода.

Используя последовательное включение схем рис. 7.9 и рис. 7.10, можно обеспечить приведение к стандартному уровню входного напряжения, изменяемого в очень широких пределах (рис. 7.11). Такая задача часто встречается в аналого-цифровых системах. Коэффициент передачи всей схемы будет равен отношению входных кодов обоих ЦАП N/M и может быть установлен с высокой точностью как в диапазоне от 0 до 1 (аттенюатор), так и в диапазоне от 1 до 2^n (усилитель). На схеме не показаны входные регистры обоих ЦАП, но они также нужны.

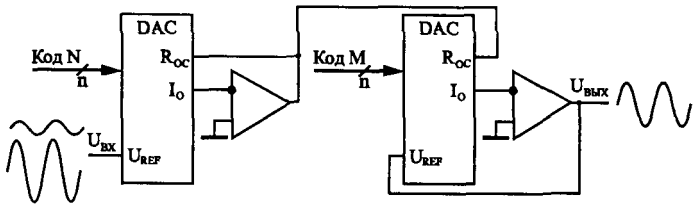


Рис. 7.11. Последовательное включение аттенюатора и усилителя.

Наконец, последняя схема с применением ЦАП, которую мы рассмотрим, — это схема сдвига аналогового сигнала на величину, задаваемую входным цифровым кодом. Сдвиг представляет собой, по сути, сложение аналогового сигнала с постоянным напряжением. Такая задача довольно часто встречается в аналого-цифровых системах.

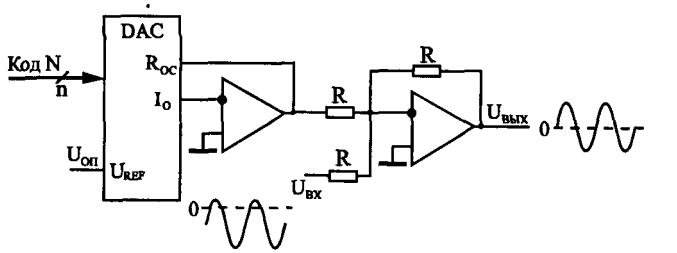


Рис. 7.12. Схема управляемого сдвига аналогового сигнала.

Схема сдвига (рис. 7.12) включает в себя преобразователь цифрового кода в выходное напряжение и аналоговый сумматор на операционном усилителе. Величина напряжения сдвига входного сигнала будет равна $U_{REF} \cdot 2^{-n} \cdot N$. Так как применяются два инвертирующих операционных усилителя, инверсии входного сигнала на выходе в данном случае не будет. Если нужен как положительный, так и отрицательный сдвиг, то необходимо применять ЦАП с биполярным выходным сигналом.

7.2. Применение АЦП

Микросхемы АЦП выполняют функцию, прямо противоположную функции ЦАП — преобразуют входной аналоговый сигнал в последовательность цифровых кодов. В общем случае микро-

схему АЦП можно представить в виде блока, имеющего один аналоговый вход, один или два входа для подачи опорного (образцового) напряжения, а также цифровые выходы для выдачи кода, соответствующего текущему значению аналогового сигнала (рис. 7.13).

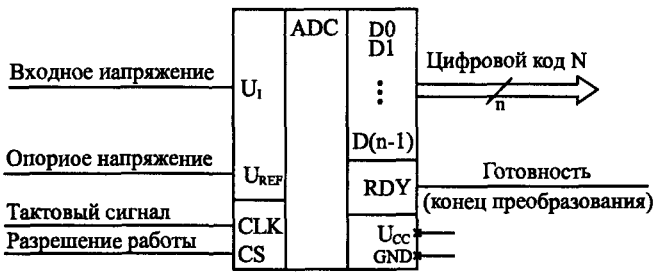


Рис. 7.13. Микросхема АЦП.

Часто микросхема АЦП имеет также входы для подачи тактового сигнала CLK, сигнала разрешения работы CS и выход для выдачи сигнала RDY, указывающего на готовность выходного цифрового кода. На микросхему подается одно или два питающих напряжения. В целом микросхемы АЦП сложнее, чем микросхемы ЦАП, их разнообразие заметно больше, и поэтому сформулировать для них общие принципы применения сложнее.

Опорное напряжение АЦП задает диапазон входного напряжения, в котором производится преобразование. Оно может быть постоянным или же допускать изменение в некоторых пределах. Иногда предусматривается подача на АЦП двух опорных напряжений с разными знаками, тогда АЦП способен работать как с положительными, так и с отрицательными входными напряжениями.

Выходной цифровой код N (n-разрядный) однозначно соответствует уровню входного напряжения. Код может принимать 2^n значений, то есть АЦП может различать 2^n уровней входного напряжения. Количество разрядов выходного кода n представляет собой важнейшую характеристику АЦП. В момент готовности выходного кода выдается сигнал окончания преобразования RDY, по которому внешнее устройство может читать код N.

Управляется работа АЦП тактовым сигналом CLK, который задает частоту преобразования, то есть частоту выдачи выход-

ных кодов. Предельная тактовая частота — второй важнейший параметр АЦП. В некоторых микросхемах имеется встроенный генератор тактовых сигналов, поэтому к их выводам подключается кварцевый генератор или конденсатор, задающий частоту преобразования. Сигнал CS разрешает работу микросхемы.

Выпускается множество самых разнообразных микросхем АЦП, различающихся скоростью работы (частота преобразования от сотен килогерц до сотен мегагерц), разрядностью (от 6 до 24), допустимыми диапазонами входного сигнала, величинами погрешностей, уровнями питающих напряжений, методами выдачи выходного кода (параллельный или последовательный), другими параметрами. Обычно микросхемы с большим количеством разрядов имеют невысокое быстродействие, а наиболее быстродействующие микросхемы имеют небольшое число разрядов. Область применения любой микросхемы АЦП во многом определяется использованным в ней принципом преобразования, поэтому необходимо знать особенности этих принципов. Для выбора и использования АЦП необходимо пользоваться подробными справочными данными от фирмы-производителя.

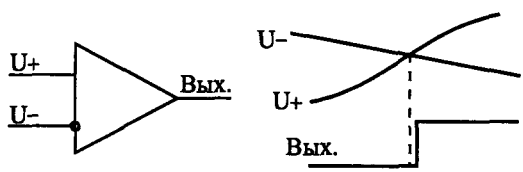


Рис. 7.14. Компаратор напряжения.

В качестве базового элемента любого АЦП используется компаратор напряжения (рис. 7.14), который сравнивает два входных аналоговых напряжения и в зависимости от результата сравнения выдает выходной цифровой сигнал нуль или единицу. Компаратор работает с большим диапазоном входных напряжений и имеет высокое быстродействие (задержка порядка единиц наносекунд).

Существует два основных принципа построения АЦП: последовательный и параллельный.

В последовательном АЦП входное напряжение последовательно сравнивается одним единственным компаратором с несколькими эталонными уровнями напряжения, и в зависимости

от результатов этого сравнения формируется выходной код. Наибольшее распространение получили АЦП на основе так называемого регистра последовательных приближений (рис. 7.15).

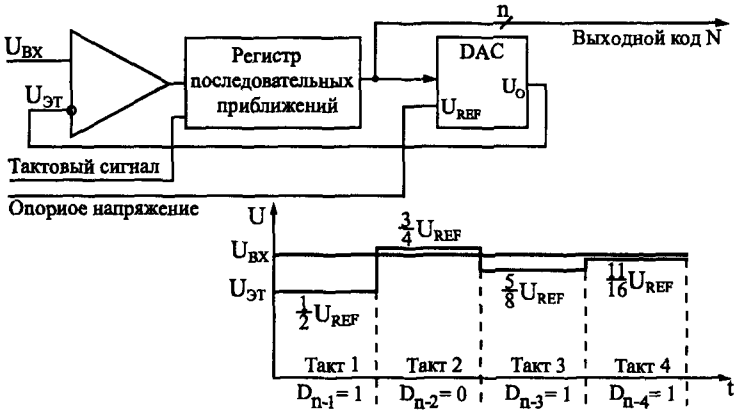


Рис. 7.15. АЦП последовательного типа.

Входное напряжение подается на вход компаратора, на другой вход которого подается ступенчато изменяющееся во времени эталонное напряжение. Выходной сигнал компаратора подается на вход регистра последовательных приближений, тактируемого внешним тактовым сигналом. Выходной код регистра последовательных приближений поступает на ЦАП, которое из опорного напряжения формирует меняющееся эталонное напряжение.

Регистр последовательных приближений работает так, что в зависимости от результата предыдущего сравнения выбирается следующий уровень эталонного напряжения по следующему алгоритму:

- В первом такте входной сигнал сравнивается с половиной опорного напряжения.
- Если входной сигнал меньше половины опорного напряжения, то на следующем такте он сравнивается с четвертью опорного напряжения (то есть половина опорного напряжения уменьшается на четверть). Одновременно в регистр последовательных приближений записывается старший разряд выходного кода, равный нулю.

- Если же входной сигнал больше половины опорного напряжения, то на втором такте он сравнивается с 3/4 опорного напряжения (то есть половина увеличивается на четверть). Одновременно в регистр последовательных приближений записывается старший разряд выходного кода, равный единице.
- Затем эта последовательность сравнений повторяется нужное число раз с уменьшением на каждом такте вдвое ступени изменения эталонного напряжения (на третьем такте — 1/8 опорного напряжения, на четвертом — 1/16 и т. д.). В результате опорное напряжение в каждом такте приближается к входному напряжению. Всего преобразование занимает n тактов. В последнем такте вычисляется младший разряд

Понятно, что этот процесс довольно медленный, требует нескольких тактов, причем в течение каждого такта должны успеть сработать компаратор, регистр последовательных приближений и ЦАП с выходом по напряжению. Поэтому последовательные АЦП довольно медленные, имеют сравнительно большое время преобразования и малую частоту преобразования.

Второй тип АЦП, АЦП параллельного типа, работает по более простому принципу. Все разряды выходного кода вычисляются в них одновременно (параллельно), поэтому они гораздо быстрее, чем последовательные АЦП. Правда, они требуют применения большого количества компараторов ($2^n - 1$), что вызывает чисто технологические трудности при большом количестве разрядов (например, при 12-разрядном АЦП требуется 4095 компараторов).

Схема такого АЦП (рис. 7.16) включает в себя резистивный делитель из 2^n одинаковых резисторов, который делит опорное напряжение на $(2^n - 1)$ уровней.

Входное напряжение сравнивается с помощью компараторов с уровнями, формируемыми делителем напряжения. Выходные сигналы компараторов с помощью шифратора преобразуются в n-разрядный двоичный код. Шифратор выдает на выход номер последнего из сработавших (то есть выдавших сигнал логической единицы) компараторов. Например, в случае 3-разрядного АЦП (на рисунке) при величине входного напряжения от 0 до 1/8 опорного напряжения выходной код будет 000, при входном напряжении от 1/8 до 2/8 опорного напряжения сработает первый компаратор, что даст выходной код 001, при входном напряжении от 2/8 до 3/8 опорного напряжения сработают компа-

раторы 1 и 2, что даст выходной код 010, и т. д. Процесс преобразования происходит в параллельном АЦП очень быстро, поэтому частота преобразования может достигать сотен мегагерц.

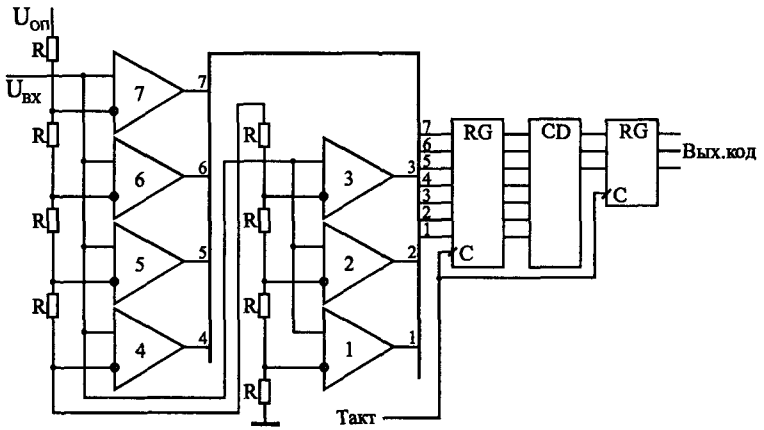


Рис. 7.16. 3-разрядный АЦП параллельного типа.

Для повышения быстродействия в параллельном АЦП иногда применяется конвейерный принцип: выходной код компараторов записывается в (2^n-1)-разрядный параллельный регистр, показанный на рисунке 7.16. Выходной код шифратора также записывается в n-разрядный параллельный регистр. Оба регистра в этом случае тактируются одним и тем же тактовым сигналом. Это снижает требования к быстродействию компараторов и шифратора. Правда, выходной код АЦП задерживается из-за таких регистров на два периода таковой частоты.

Громоздкость структуры параллельного АЦП приводит к тому, что в некоторых АЦП применяется смешанный параллельно-последовательный принцип. Это несколько снижает быстродействие подобного АЦП по сравнению с обычным параллельным АЦП, но зато позволяет получить большое число разрядов, не увеличивая количество компараторов до 2^n-1.

Для того чтобы АЦП любого типа работал с использованием всех своих возможностей, необходимо обеспечить согласование диапазона изменения входного аналогового сигнала с допустимым диапазоном (динамическим диапазоном) входного напряжения АЦП.

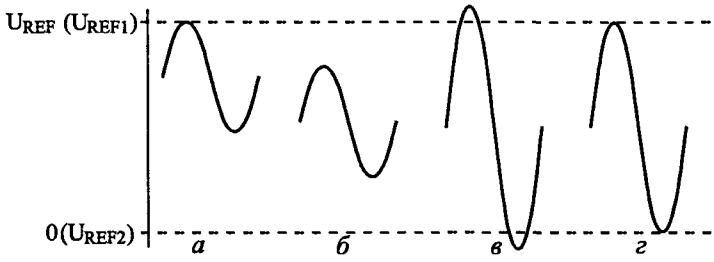


Рис. 7.17. Соотношение входного сигнала и динамического диапазона АЦП.

На рис. 7.17 показано четыре возможных случая соотношения динамического диапазона АЦП (от 0 до U_{REF} или от U_{REF1} до U_{REF2}) и входного сигнала. В случаях *а* и *б* входной сигнал меньше динамического диапазона, поэтому АЦП будет работать правильно, но не будет использовать всех своих возможностей. В случае *в* входной сигнал слишком большой, поэтому часть его значений не будет преобразована. Только в случае *г* АЦП действительно будет работать как n-разрядный и будет преобразовывать все значения входного сигнала. Для согласования входного сигнала с динамическим диапазоном АЦП можно применять усилители, аттенюаторы, схемы сдвига. В некоторых случаях согласование может быть достигнуто простым выбором величин опорных напряжений.

Иногда бывает необходимо уменьшить количество разрядов АЦП. В этом случае нужное количество младших разрядов выходного кода микросхемы просто не используется. На рис. 7.18 показано использование 10-разрядного АЦП в качестве 8-разрядного.

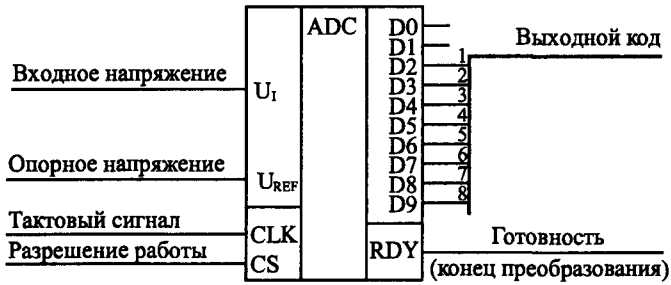


Рис. 7.18. Уменьшение количества разрядов выходного кода АЦП.

Обратная задача — увеличение разрядности АЦП — встречается чаще. Существует ряд типичных схемотехнических решений по объединению нескольких микросхем АЦП для увеличения количества разрядов выходного кода, но большинство этих решений требует сложных расчетов результирующих погрешностей преобразования и применения аналоговых узлов. Мы не будем их здесь рассматривать. Отметим только, что при возникновении задачи увеличения разрядности надо прежде всего попытаться найти микросхему с нужным количеством разрядов и только потом рассматривать возможности объединения нескольких микросхем АЦП.

Рассмотрим несколько типичных схем включения АЦП, используемых в аналого-цифровых системах.

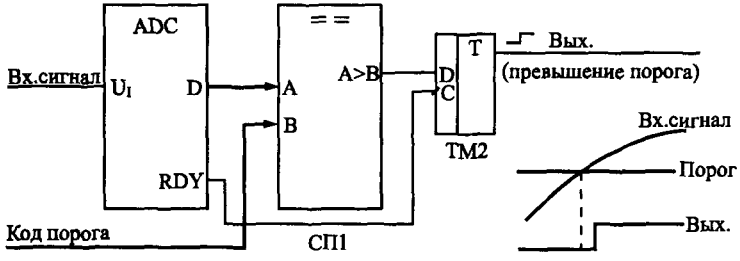


Рис. 7.19. Фиксатор превышения входным сигналом установленного порога.

Первая схема (рис. 7.19) предназначена для фиксации момента превышения входным аналоговым сигналом заданного порогового напряжения. Схема вырабатывает выходной сигнал (положительный фронт) тогда, когда входной аналоговый сигнал становится больше установленного уровня, причем уровень этот задается цифровым кодом порога. Код порога сравнивается с выходными кодами АЦП с помощью микросхемы компаратора кодов. Выходной сигнал компаратора кодов записывается в триггер по сигналу RDY с АЦП, что позволяет исключить влияние коротких импульсов, возникающих на выходе компаратора в момент изменения входных кодов. Применение этого триггера задерживает выходной сигнал на один такт.

Может показаться, что применение АЦП в данном случае не оправданно, избыточно. Но надо учитывать, что в аналого-цифровых системах АЦП, преобразующий входной сигнал в по-

следовательность кодов, как правило, уже есть, поэтому дополнительного АЦП не требуется, достаточно только включить компаратор кодов и триггер.

АЦП также применяется в схемах вычисления амплитуды входного аналогового сигнала. Для такого вычисления можно использовать уже рассмотренную схему вычислителя экстремального значения входного кода (см. рис. 4.26). В качестве источника последовательности входных кодов в данном случае выступает АЦП (рис. 7.20).

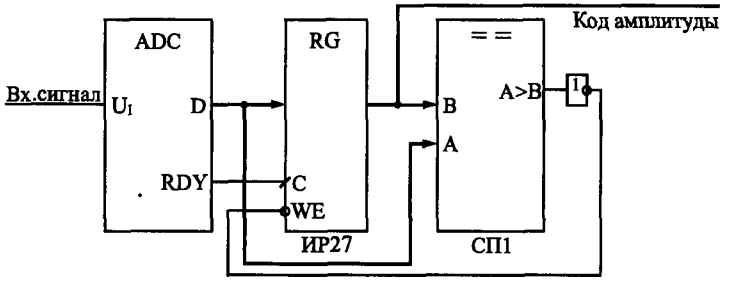


Рис. 7.20. Вычислитель амплитуды аналогового сигнала.

В регистр со входом разрешения записи записывается код с выхода АЦП по сигналу RDY в том случае, если текущее значение кода больше значения кода, записанного ранее в регистр. В результате уже после одного периода входного сигнала в регистре будет код амплитуды входного сигнала. За период преобразования АЦП должны успеть сработать компаратор кодов и регистр.

Если такой вычислитель амплитуды входного сигнала используется в составе сложной аналого-цифровой системе, в которой уже присутствует АЦП, непрерывно преобразующий входной сигнал в коды, то дополнительно требуются только цифровые микросхемы: компаратор кодов и регистр.

Наиболее часто встречающееся использование АЦП — это преобразование входного сигнала в поток кодов, причем коды эти обычно записываются в буферную память. В данном случае наиболее подходящим является однонаправленный буфер с периодическим режимом работы. То есть сначала в буферную память заносится массив кодов выборок входного сигнала, а затем этот массив читается для дальнейшей обработки. Именно так,

например, строится цифровой осциллограф, предназначенный для наблюдения аналоговых сигналов на экране.

Схема включения АЦП в этом случае показана на рис. 7.21. В качестве строба записи в буферную память используется сигнал RDY с АЦП. Подробнее организацию буфера мы уже рассматривали в предыдущей главе.

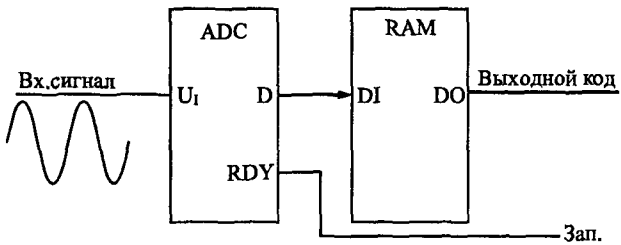


Рис. 7.21. Включение буферной памяти для запоминания кодов с выходов АЦП.

Конечно, в реальных аналого-цифровых устройствах все гораздо сложнее, в них требуются схемы синхронизации процесса записи со входным сигналом, схемы предварительной обработки аналогового сигнала, но суть остается той же — буферная память, записывающая последовательность кодов с выхода АЦП. Чем больше объем памяти, тем больший фрагмент входного аналогового сигнала она может запомнить. Например, если память имеет организацию 64К × 8 и работает с 8-разрядным АЦП, то при частоте преобразования АЦП 10 МГц буфер сможет хранить в себе фрагмент аналогового сигнала длительностью 6,5536 мс.

Наконец, последняя схема, которую мы рассмотрим (рис. 7.22), позволяет вдвое повысить быстродействие АЦП, точнее, поднять вдвое частоту записи кодов выборок входного сигнала в буферную память.

Идея схемы очень проста: используется два АЦП и два буфера, которые работают по очереди, например, четные выборки входного сигнала обрабатывает один АЦП со своим буфером, а нечетные — другой АЦП со своим буфером. В результате запоминание кодов входного сигнала осуществляется с частотой вдвое больше частоты преобразования каждого из АЦП. Например, если каждый АЦП и каждый буфер работают с частотой 10 МГц, то результирующая частота преобразования составит 20 МГц.

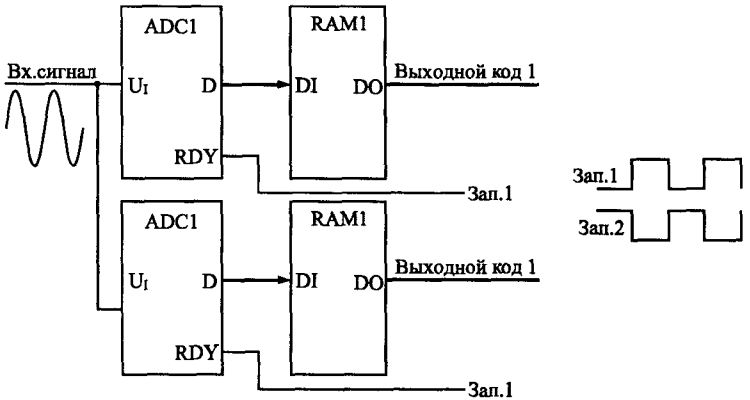


Рис. 7.22. Увеличение вдвое частоты преобразования входного сигнала с помощью двух АЦП с буферами.

Тактовые сигналы АЦП и сигналы RDY на выходах АЦП должны быть сдвинуты один относительно другого на половину периода тактового сигнала. Чтение зарегистрированных кодов из обоих буферов также должно быть организовано по очереди: первый код читается из первого буфера, второй — из второго, третий — опять из первого, четвертый — из второго и т. д. Объем обоих буферов в данном случае складывается. Например, при организации каждого буфера 64К × 8 результирующий буфер будет иметь организацию 128К × 8.

Пользуясь этим же принципом, можно повысить частоту обработки входного сигнала с помощью АЦП не только вдвое, но и втрое, в четыре раза и т. д. Необходимо только согласовать во времени работу соответственно трех, четырех и т. д. АЦП, у каждого из которых должна быть своя буферная память.

Помимо упомянутых здесь АЦП последовательно и параллельного типов существуют еще и АЦП с промежуточным преобразованием. В них входной аналоговый сигнал с помощью аналогового интегратора преобразуется во временной интервал между цифровыми импульсами или в частоту следования цифровых импульсов. Выходной цифровой код, соответствующий входному аналоговому сигналу формируется в результате измерения длительности временного интервала или частоты следования импульсов (рис. 7.23). Если используется выходная частота, то такой АЦП называется «преобразователем напряжение–частота» (ПНЧ).

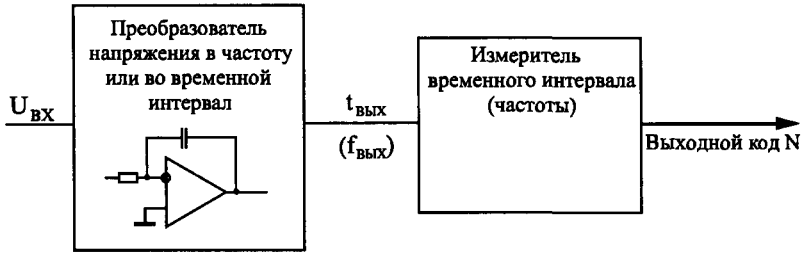


Рис. 7.23. АЦП с промежуточным преобразованием.

Такой подход позволяет с помощью сравнительно простых аппаратных средств получить высокую точность преобразования, не зависящую от многих параметров используемых компонентов и от характеристик окружающей среды. Измерение временных интервалов и частоты следования импульсов осуществляется простейшими цифровыми схемами, примеры которых приведены в главе 5. Измерения эти могут осуществляться с высокой точностью вследствие того, что существует очень хороший временной эталон — кварцевый генератор. Отметим, что достоинством ПНЧ является также возможность простой передачи его выходного цифрового сигнала на большие расстояния.

В конце главы надо еще раз отметить, что приведенные здесь схемы сильно упрощены, для их практической реализации необходимо знание не только цифровой схемотехники, но и аналоговой и аналого-цифровой схемотехники, а также знание особенностей конкретных микросхем ЦАП и АЦП, что не является предметом данной книги. Однако рассмотренные здесь ключевые принципы использования ЦАП и АЦП и их совместного включения с цифровыми схемами будут полезны любому разработчику.

Глава 8
ПРИМЕРЫ РАЗРАБОТКИ ЦИФРОВЫХ УСТРОЙСТВ

В предыдущих главах были рассмотрены базовые элементы цифровой схемотехники и простейшие приемы проектирования узлов на их основе. Но для разработки сложных устройств и систем всех этих знаний порой оказывается недостаточно.

Чтобы создать сложное устройство, необходимо еще владеть приемами системотехники, то есть уметь на основании анализа функций, которые должно выполнять устройство в целом, спроектировать его структуру, сформулировать принципы взаимодействия узлов, четко определить все задачи, которые должен решать каждый из узлов, выработать требования к отдельным узлам. Возможно, в результате всех этих шагов будет изменена сама первоначальная задача, будут переформулированы требования к создаваемому устройству, его месту в системе, принципам его взаимодействия с другими устройствами. И только потом, после всей этой предварительной работы уже можно переходить к разработке узлов, собственно к схемотехнике.

Приемы системотехники сформулировать, формализовать, описать, даже перечислить гораздо сложнее, чем приемы схемотехники. Да и пользы от такой формализации зачастую немного. Проектирование сложного, надежно работающего цифрового устройства с минимальными аппаратурными затратами сродни искусству и требует определенных способностей, даже таланта разработчика. Научить этому практически невозможно. Более того, попытка научить системотехнике может даже принести вред, так как ограничит творческие способности разработчика несколькими жесткими стандартными алгоритмами.

Однако можно показать несколько простейших примеров разработки, продемонстрировать последовательность шагов, которые необходимы при проектировании, которые могут встретиться в процессе создания устройств некоторых распространенных типов. Исходя из этих примеров разработчик может в дальнейшем попробовать по аналогии создать что-то свое, более или менее совершенное, но обязательно работоспособное.

Поэтому в данной главе как раз и будут подробно рассмотрены несколько простых примеров проектирования сравнительно сложных устройств на всех этапах: от анализа решаемой задачи до создания полной принципиальной схемы. Примеры эти, конечно, не отражают и малой доли всех реально встречающихся задач, но относятся к различным классам цифровых устройств.

8.1. Разработка клавиатуры

Различные клавиатуры с большим количеством клавиш (кнопок) широко используются в цифровых системах: в компьютерах, контроллерах, измерительных приборах, в бытовой технике. Основная задача любой клавиатуры довольно проста: она должна при любом нажатии на клавишу выдавать код номера этой клавиши и сигнал флага нажатия клавиши (строб этого кода). Получив этот сигнал флага, внешнее устройство читает код нажатой клавиши и предпринимает требуемые действия.

Главная задача при проектировании клавиатуры состоит в минимизации аппаратурных затрат и в обеспечении надежного срабатывания в любой ситуации. Существует масса схемотехнических решений этой задачи, от примитивных до сложнейших. Клавиатуры могут быть механическими, квазисенсорными или сенсорными, клавиатуры могут иметь жесткую логику работы или быть интеллектуальными, даже допускать перепрограммирование. Мы будем в качестве примера рассматривать самую простую механическую клавиатуру с жесткой логикой работы.

Количество клавиш полноразмерной клавиатуры компьютера превышает сотню, поэтому мы будем проектировать клавиатуру на максимальное количество клавиш, равное 128. Естественно, клавиатура должна иметь защиту от дребезга механических контактов и должна корректно обрабатывать ситуацию одновременного нажатия нескольких клавиш. Примем, например, что при одновременном нажатии нескольких клавиш клавиатура должна выдавать код только одной из них. Примем также, что максимально возможный темп нажатия клавиш на клавиатуре не должен превышать 20 нажатий в секунду (это довольно много). Таким образом, основные требования к проектируемому устройству сформулированы. Начнем разработку.

Очень часто удобным и эффективным приемом разработки является начало разработки устройства «с конца». То есть про-

ектирование начинается, исходя из требуемого результата, из тех сигналов, которые устройство должно выдавать вовне и принимать извне. И только в конце проектирования разрабатывается та часть устройства, которая выполняет требуемую функцию. Такой подход гарантирует, что разработанное устройство не будет чрезмерно избыточным, не будет делать ничего лишнего, а также то, что оно корректно будет взаимодействовать с другими устройствами и системами. Этот принцип проектирования не универсален, порой выдержать его в течение всего процесса разработки трудно, но попробовать его применить к любому устройству никогда не помешает.

В нашем случае необходимо сначала определиться, что должна выдавать вовне клавиатура. Обычно это задается техническим заданием, но мы примем, что наша клавиатура должна выдавать 7-разрядный двоичный номер нажатой клавиши (так как $2^7 = 128$) и сопровождать его положительным сигналом флага нажатия. Сигнал флага и код клавиши должны сохраняться до тех пор, пока нажата клавиша. За это время (несколько миллисекунд) внешнее устройство должно успеть проанализировать сигнал флага и прочитать выходной код клавиатуры. Обычно данное требование является не слишком жестким.

Альтернативное решение — сохранение кода нажатой клавиши и сигнала флага до момента чтения выходного кода внешним устройством — конечно же, снижает требование к быстродействию читающего внешнего устройства, однако оно может привести к тому, что некоторые нажатия клавиш останутся без реакции, не будут обработаны.

Необходимо также определиться, как клавиатура будет вести себя при одновременном нажатии нескольких клавиш. Наиболее сложные, интеллектуальные клавиатуры выдают последовательно коды всех нажатых клавиш, запоминая их в буферной памяти. Но мы примем, что клавиатура должна выдавать только код одной из одновременно нажатых клавиш (первой по установленному порядку). Нажатия всех остальных клавиш одновременно с данной клавишей просто игнорируются.

При проектировании механической клавиатуры важно решить, как будет обрабатываться неизбежно присутствующий дребезг механических контактов клавиш. Его можно обрабатывать как внутри клавиатуры, так и вне ее (то есть перенести эту функцию на внешнее устройство). Оба этих подхода имеют свои преимущества. Но

наша клавиатура будет обрабатывать дребезг контактов самостоятельно. Принцип обработки выбираем очень простой: первое зафиксированное замыкание контактов клавиши считается началом нажатия, а конец нажатия определяется тогда, когда контакты будут разомкнуты в течение заданного интервала времени.

В результате временная диаграмма работы разрабатываемой клавиатуры может быть упрощенно представлена в виде рис. 8.1. Здесь сигнал флага начинается при фиксации единичного сигнала с клавиши (это может быть как во время дребезга, так и после его окончания). После выставления флага фиксируется выходной код клавиши. После отпускания клавиши (нулевой сигнал), через время задержки $t_{зад}$ снимается сигнал флага. Время задержки должно быть заведомо больше времени дребезга контактов. Выходной код может сохраняться после отпускания клавиши до следующего нажатия, а может и сниматься.

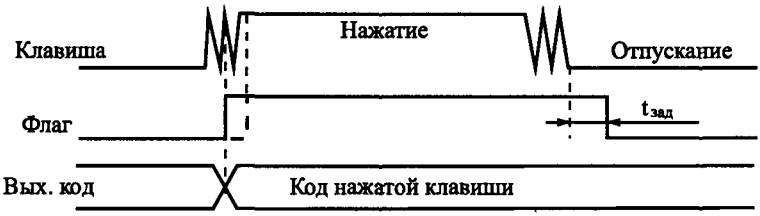


Рис. 8.1. Временная диаграмма работы клавиатуры.

Дальнейшая разработка невозможна без выбора принципа преобразования сигналов от нажатия клавиш в код номера нажатой клавиши.

Простейшим путем построения подобного преобразователя является использование приоритетных шифраторов (рис. 8.2).

Каждая клавиша дает свой логический сигнал, сигналы от всех клавиш преобразуются шифратором в код номера клавиши. Однако такой простейший подход хорош только при небольшом количестве клавиш (до 8 или до 16), так как при большом количестве входов приоритетный шифратор получается довольно сложным. При малом количестве клавиш дребезг контактов обычно устраняется отдельно для каждой клавиши с помощью RS-триггера (как это показано на рисунке). Это решение простое, но требующее больших аппаратурных затрат.

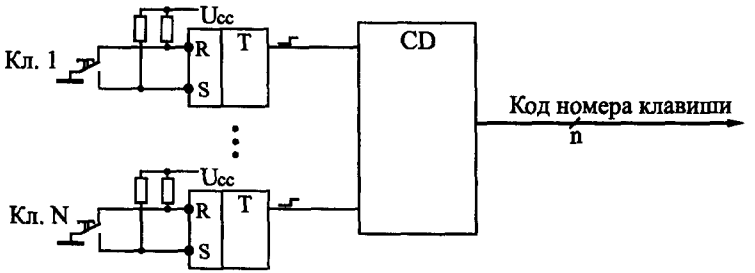


Рис. 8.2. Простейший преобразователь для клавиатуры.

Другим путем построения преобразователя является использование так называемой коммутационной матрицы, состояние которой периодически опрашивается с частотой тактового генератора. Коммутационная матрица представляет собой две группы пересекающихся проводников (строки и столбцы), во всех точках пересечения которых находятся клавиши. В данном случае каждая клавиша не формирует своего отдельного логического сигнала, а только коммутирует (соединяет) одну из строк матрицы с одним из ее столбцов.

Наиболее универсальная схема преобразователя, легко наращиваемая и достаточно простая, приведена на рис. 8.3.

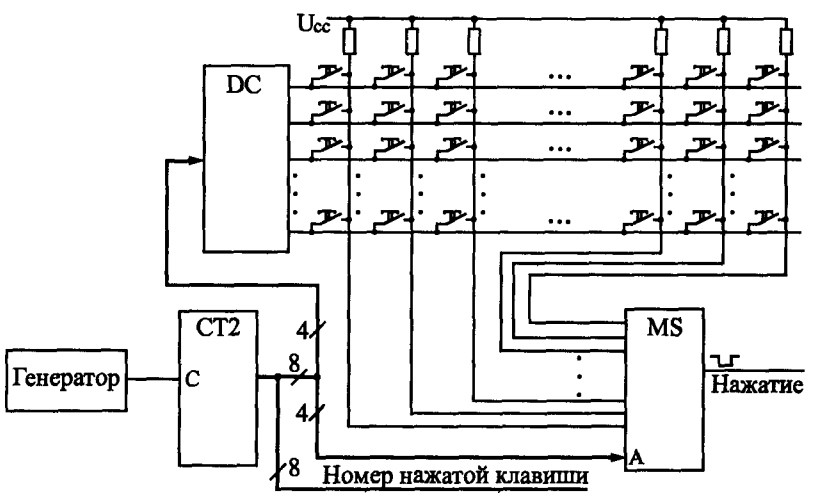


Рис. 8.3. Преобразователь с опросом всех клавиш.

Для опроса коммутационной матрицы используется счетчик, тактируемый генератором. Старшие разряды счетчика используются для выбора одной из строк матрицы с помощью дешифратора (на выбранную строку поступает сигнал логического нуля, на невыбранную — сигнал логической единицы). Младшие разряды счетчика используются для опроса столбцов матрицы с помощью мультиплексора. Сигнал с опрашиваемого столбца подается на выход мультиплексора. Признаком нажатия клавиши является нулевой сигнал на выходе мультиплексора. В этот момент на выходах счетчика присутствует код номера нажатой клавиши. Такая схема легко позволяет строить клавиатуры на большое количество клавиш (до 256, как на рисунке, и даже больше), однако она требует довольно большого времени для полного опроса клавиатуры (так как количество тактов опроса равно полному количеству клавиш).

Совмещение двух рассмотренных подходов позволяет строить достаточно большие клавиатуры с малыми аппаратурными затратами и малым временем опроса.

При таком комбинированном методе (рис. 8.4) также используется коммутационная матрица с клавишами на всех пересечениях строк и столбцов, но опрашиваются не все клавиши по очереди, а только строки (или столбцы) матрицы. Для опроса, как и в предыдущем случае, применяются генератор, счетчик и дешифратор. Положение же нажатой клавиши в строке (или в столбце) определяется с помощью шифратора. Код нажатой клавиши образуется из выходного кода счетчика (старшие разряды) и кода с выхода шифратора (младшие разряды).

В нашем случае клавиатура имеет 128 клавиш, то есть коммутационная матрица должна состоять из 16 строк, опрашиваемых дешифратором 4—16 (ИД3), и 8 столбцов, сигналы с которых обрабатываются шифратором 8—3 (ИВ1). Счетчик должен иметь 4 разряда (ИЕ7). Эти 4 разряда и 3 разряда с выхода шифратора дадут 7-разрядный номер нажатой клавиши. Полный цикл опроса клавиатуры будет занимать 16 тактов генератора (по числу строк). Признаком нажатия одной из клавиш будет отрицательный сигнал на выходе -GS шифратора. Если нажато несколько клавиш в разных строках, то обрабатываться будет та клавиша, строка которой будет опрошена первой. Если нажато несколько клавиш в одной строке, то шифратор выдаст код клавиши, соответствующей большему номеру входа. Надо также

учитывать, что шифратор ИВ1 выдает инверсный номер входа, на который пришел нулевой сигнал, эта особенность может потребовать применения трех выходных инверторов (на рисунке не показаны).

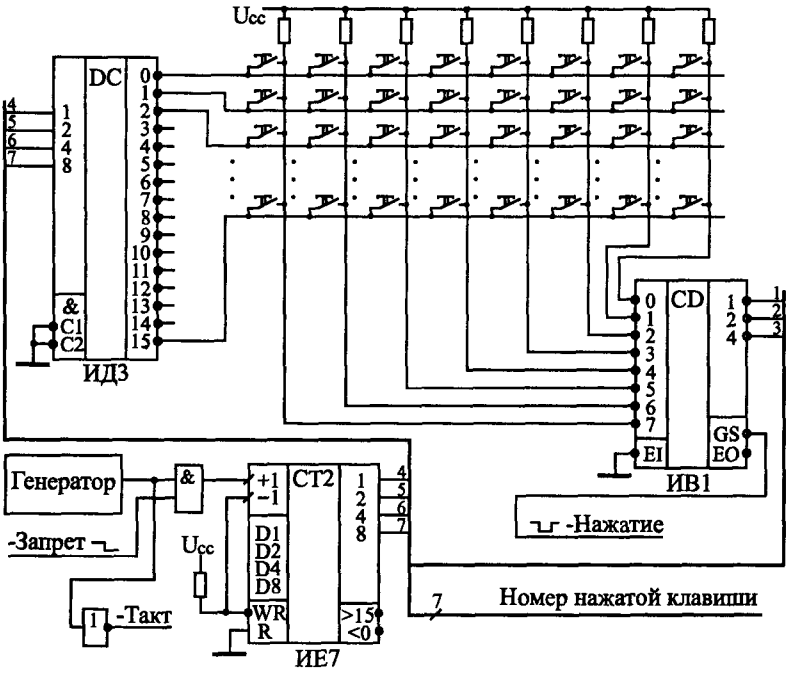


Рис. 8.4. Преобразователь с опросом строк клавиш.

Оценим, какой должна быть частота тактового генератора. Мы приняли, что максимальная скорость нажатия равна 20 раз в секунду. Значит, за 1/20 секунды надо успеть опросить всю клавиатуру, то есть все 16 строк. Таким образом, минимально допустимая тактовая частота составляет 16 • 20 = 320 Гц. Но надо заложить и запас на обработку дребезга контактов. Поэтому примем тактовую частоту опроса равной 400 Гц. Она может быть и больше, но чрезмерно увеличивать ее (например, выше 1 кГц) не стоит, так как при быстром переключении микросхем увеличивается потребляемый схемой ток. Понятно, что генератор должен быть не кварцевым, так как кварцевые резонаторы

на низкие частоты не выпускаются, а делитель частоты резко усложнит схему. К тому же точная выдержка частоты генератора в данном случае совершенно не нужна.

Выходной сигнал -Нажатие, конечно же, будет иметь короткие паразитные импульсы. Во-первых, они будут возникать из-за дребезга контактов нажатой в данный момент клавиши. Во-вторых, они могут возникать из-за переходных процессов при переключении счетчика и дешифратора. Эти паразитные импульсы надо исключать.

Чтобы исключить действие паразитных импульсов из-за переходных процессов при переключении счетчика и дешифратора, достаточно применить стробирование или тактирование сигнала -Нажатие в середине каждого тактового интервала. Для этого из схемы преобразователя надо вывести сигнал -Такт.

Исключение коротких выходных импульсов из-за дребезга контактов клавиш сложнее. Прежде всего, на время нажатия клавиши целесообразно остановить опрос строк с помощью сигнала -Запрет. Затем надо обработать сигнал -Нажатие по принципу, показанному на рис. 8.1. Будем считать, что при дребезге контактов длительность кратковременного размыкания не превышает периода тактового генератора (2,5 мс при тактовой частоте 400 Гц). Тогда задержка окончания сигнала флага нажатия (см. рис. 8.1) должна быть не менее одного периода тактового сигнала. Для выработки задержки можно использовать цепочку триггеров, тактируемых сигналом -Такт.

Схема выработки выходных сигналов клавиатуры приведена на рис. 8.5.

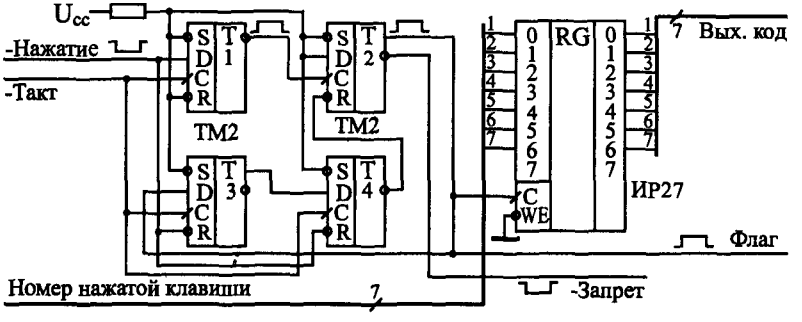


Рис. 8.5. Схема выработки выходных сигналов клавиатуры.

Инверсный выход триггера Т1 переключается в состояние логической 1 в середине тактового интервала (по положительному фронту сигнала -Такт), если сигнал -Нажатие — нулевой. Своим выходным сигналом триггер Т1 перебрасывает в единицу триггер Т2, который уже никак не связан с сигналом -Нажатие, не реагирует ни на какой дребезг этого сигнала. Выходной сигнал триггера Т2 используется в качестве сигнала флага нажатия клавиатуры. Инверсный сигнал с выхода триггера Т2 используется в качестве сигнала -Запрет, останавливающего опрос строк клавиатуры.

Цепочка триггеров Т3 и Т4, тактируемая сигналом -Такт, служит для задержки снятия сигнала флага после отпускания клавиши (когда сигнал -Нажатие становится равным единице). После установки флага в единицу сигнал флага начинает записываться по фронту сигнала -Такт в триггеры Т3 и Т4, но только в том случае, когда сигнал -Нажатие установлен в единицу. В результате на инверсном выходе триггера Т4 появится сигнал логического нуля при единичном значении сигнала -Нажатие в моменты двух последовательных положительных фронтов сигнала -Такт. Сигнал с выхода Т4 сбрасывает сигнал флага в нуль, после чего вся схема переходит в исходное состояние и ждет следующего нулевого сигнала -Нажатие.

Если кратковременное размыкание при дребезге контактов клавиш длится более 2,5 мс, то можно увеличить количество триггеров в последовательной цепочке (Т3 и Т4), что приведет к увеличению задержки снятия сигнала флага на целое число тактов генератора.

Таким образом, схема полностью разработана. Отметим, что низкая тактовая частота работы схемы позволяет нам не рассчитывать задержек микросхем, то есть использовать только первый уровень представления, логическую модель. А эффекты, связанные с переходными процессами при переключении микросхем, мы устранили, обеспечив временной сдвиг между тактовыми сигналами схемы преобразователя (рис. 8.4) и схемы выработки выходных сигналов (рис. 8.5) на половину периода генератора. Номиналы всех резисторов, примененных в схеме, должны быть около 1 кОм.

8.2. Разработка вычислителя контрольной суммы

Различные контрольные суммы широко применяются в цифровых устройствах и системах для контроля правильности хранения или передачи массивов информации. Суть этого метода контроля проста: к хранимому или передаваемому информационному массиву присоединяется небольшой контрольный код (обычно от 1 разряда до 32 разрядов), в котором в свернутом виде содержится информация обо всем массиве. При чтении или получении этого массива еще раз вычисляется тот же самый контрольный код по тому же самому алгоритму. Если этот вновь вычисленный код равен тому коду, который был присоединен к массиву, то считается, что массив сохранен или передан без ошибок. Логика здесь следующая: контрольный код (он же контрольная сумма) гораздо меньше контролируемого массива, поэтому вероятность искажения контрольной суммы гораздо меньше, чем вероятность искажения массива. Если же исказятся как массив, так и контрольная сумма, то вероятность того, что эти искажения не будут замечены при повторном подсчете контрольной суммы, крайне мала. Существует, правда, вероятность, что массив будет искажен в нескольких местах таким образом, что контрольная сумма от этих искажений никак не изменится, но такая вероятность также обычно мала.

Контрольные суммы применяются при хранении данных в памяти (оперативной и постоянной), при хранении данных на магнитных носителях (дисках, лентах), в локальных и глобальных сетях передачи информации. При использовании контрольной суммой для защиты хранимой информации можно установить наличие или отсутствие повреждений данного массива (файла, сектора на диске). При использовании контрольной суммы для защиты передаваемой по сети информации приемник может потребовать от передатчика повторной передачи искаженного массива.

Существует множество способов вычисления контрольной суммы, различающихся степенью сложности вычисления и надежностью выявления ошибок. Но наибольшее распространение получил в настоящее время так называемый «циклический метод контроля по избыточности» или CRC (Cyclic Redundancy Check), при котором применяется циклическая контрольная сумма.

Вычисляется циклическая контрольная сумма таким образом. Весь массив информации рассматривается как одно N-разрядное двоичное число, где N — количество бит во всех байтах массива. Для вычисления контрольной суммы это N-разрядное число делится на некоторое постоянное число (полином), выбранное специальным образом (но делится не просто, а по модулю 2). Частное от этого деления отбрасывается, а остаток как раз и используется в качестве контрольной суммы.

Мы не будем углубляться в математическое обоснование этого метода. Интересующиеся читатели могут обратиться к специальной литературе. Здесь же мы отметим только, что данный метод выявляет одиночные ошибки в массиве с вероятностью 100%, а любое другое количество ошибок с вероятностью, примерно равной $(1-2^{-n})$, где n — количество разрядов контрольной суммы (это верно только при условии, что N гораздо больше n, что, впрочем, почти всегда выполняется). Например, при n = 8 данная вероятность составит 0,996, для n = 16 она будет равна 0,999985, а для n = 32 она будет 0,9999999997672. То есть почти все ошибки будут выявляться.

А теперь кратко поясним, что такое деление по модулю 2. Пусть массив (последовательность бит) имеет следующий вид: 101111001110 (для простоты берем небольшую разрядность). В качестве делителя (называемого обычно полиномом) возьмем число 10011. Как оно выбирается? Оно должно делиться по модулю 2 без остатка только на единицу и само на себя (то есть это должно быть простое число в смысле деления по модулю 2). Разрядность полинома берется на единицу большая, чем требуемая разрядность контрольной суммы (остатка от деления). Так, чтобы получить 8-разрядный остаток (8-разрядную контрольную сумму), надо брать 9-разрядный полином. В нашем случае полином 5-разрядный, следовательно, остаток будет 4-разрядный. Для получения 8-разрядного остатка можно использовать, например, полином 1 0001 1101 или 11D в 16-ричном коде.

Деление по модулю 2 производится точно так же, как и привычное для нас деление «в столбик» (рис. 8.6), но вместо вычитания в данном случае используется поразрядное сложение по модулю 2, то есть каждый результирующий бит представляет собой функцию Исключающее ИЛИ от соответствующих битов слагаемых. Частное от деления нас не интересует, а остаток, равный в нашем примере 1000, и будет циклической контрольной суммой.

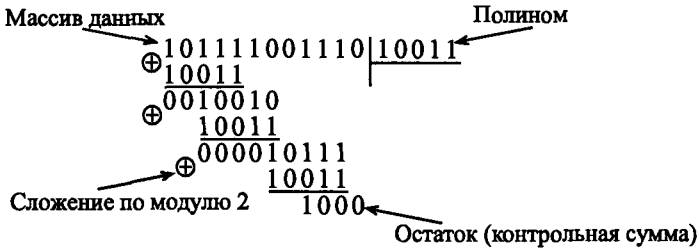


Рис. 8.6. Вычисление циклической контрольной суммы.

Как практически реализовать вычисление этого остатка (контрольной суммы)? Можно сделать это по приведенному здесь принципу деления в столбик (аппаратно или программно). Но в любом случае это довольно громоздко и медленно. Ускорить процесс вычисления можно, воспользовавшись табличным методом. Для этого составляется таблица чисел размером $2^n \times n$, где n — разрядность контрольной суммы. Принцип вычисления чисел в таблице очень прост (табл. 8.1).

Таблица 8.1. Таблица для табличного метода вычисления циклической контрольной суммы

Адрес в таблице	Данные в таблице (числа)
0	0
1	Остаток от деления числа 1 0000 0000 на полином
2	Остаток от деления числа 10 0000 0000 иа полином
3	Остаток от деления числа 11 0000 0000 на полином
4	Остаток от деления числа 100 0000 0000 на полнном
5	Остаток от деления чнсла 101 0000 0000 на полнном
...	...
255	Остаток от деления чнсла 1111 1111 0000 0000 на полнном

Числа представляют собой остаток от деления по модулю 2 числа с n конечными нулями (в нашем примере n = 8) и с n начальными разрядами, равными номеру числа (его адресу) в таблице. Деление производится на выбранный полином (в нашем случае — 9-разрядный). Таблица вычисляется один раз и хранится на диске или в ПЗУ.

Алгоритм вычисления контрольной суммы с помощью этой таблицы следующий (рассматриваем случай n = 8). Берем первый байт нашего информационного массива. Рассматриваем его как адрес в таблице (номер числа). Берем из таблицы число с полученным номером — получаем остаток O_1. Берем второй байт массива и складываем его по модулю 2 с остатком O_1. Полученное число используем как адрес в таблице. По этому адресу выбираем из таблицы остаток O_2. Берем третий байт массива, складываем его по модулю 2 с остатком O_2. Используя это число как адрес в таблице, выбираем из нее остаток O_3 и так продолжаем до последнего байта массива. Естественно, это будет гораздо быстрее, чем вычисление «в столбик».

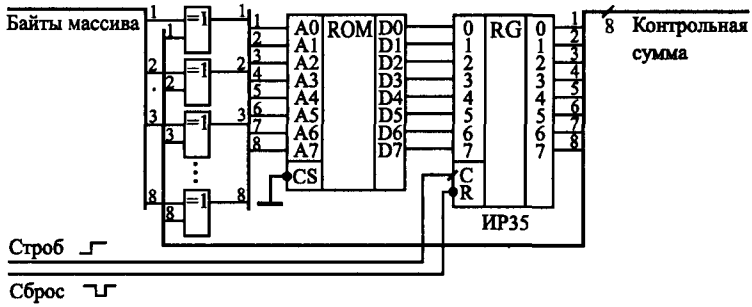


Рис. 8.7. Параллельный вычислитель 8-разрядной циклической контрольной суммы на ПЗУ.

Для реализации этого алгоритма с помощью цифровых схем требуется только ПЗУ с организацией $2^n \times n$ (256×8 при 8-разрядной контрольной сумме), n-разрядный регистр и n элементов Исключающее ИЛИ (рис. 8.7). В ПЗУ заносится таблица промежуточных остатков (табл. 8.1), на вход схемы подаются один за другим байты массива, сопровождаемые стробом. Адресом ПЗУ служит сумма по модулю 2 входных данных и содержимого выходного регистра, в который по сигналу строба записывается выходной код ПЗУ. Перед началом вычисления состояние регистра обнуляется. После окончания всего массива в регистре образуется циклическая контрольная сумма.

Недостаток данной схемы параллельного вычислителя очевиден: в случае большого числа разрядов контрольной суммы требуется очень большой объем ПЗУ (64К × 16 для 16-разряд-

ной суммы и 4Г × 32 для 32-разрядной суммы). Поэтому она применяется сравнительно редко. Зато параллельный вычислитель обладает высоким быстродействием (байты могут поступать с периодом, равным сумме задержки выходного регистра, времени выборки адреса ПЗУ и задержки элемента Исключающее ИЛИ).

Для многоразрядной контрольной суммы чаще применяется другой подход — вычисление в последовательном коде, при котором на вычислитель массив данных поступает последовательно, бит за битом. Последовательный вычислитель контрольной суммы представляет собой сдвиговый регистр с обратными связями от некоторых разрядов через сумматоры по модулю 2 (то есть элементы Исключающее ИЛИ). Полное количество разрядов регистра сдвига должно быть равно разрядности вычисляемой контрольной суммы (или, что то же самое, быть на единицу меньше разрядности используемого полинома). Место включения обратных связей однозначно определяется выбранным полиномом. Это очень похоже на генератор квазислучайной последовательности (см. раздел 4.2.3).

Количество точек включения обратной связи определяется количеством единиц в полиноме (единица в младшем разряде не учитывается), а номера разрядов сдвигового регистра, с которых берутся сигналы обратной связи, определяются номерами единичных разрядов в коде полинома. В отличие от генератора квазислучайного сигнала в данном случае необходимо смешать по функции Исключающее ИЛИ не только сигналы обратной связи, но и входной сигнал данных в последовательном коде.

На рис. 8.8 приведен пример последовательного вычислителя 16-разрядной циклической контрольной суммы при выбранном полиноме 1 0001 0000 0010 0001 или 11021 в 16-ричном коде (рекомендация МККТТ V.41). Так как в коде полинома три единицы (без младшего разряда), необходимо взять три точки включения обратной связи. При этом номера разрядов сдвигового регистра, к которым подключаются обратные связи, определяются положением единичных битов в полиноме. Перед началом работы сдвиговый регистр необходимо сбросить в нуль (сигнал -Сброс). Биты массива должны сопровождаться сигналом строба. После окончания массива в регистре будет циклическая контрольная сумма.

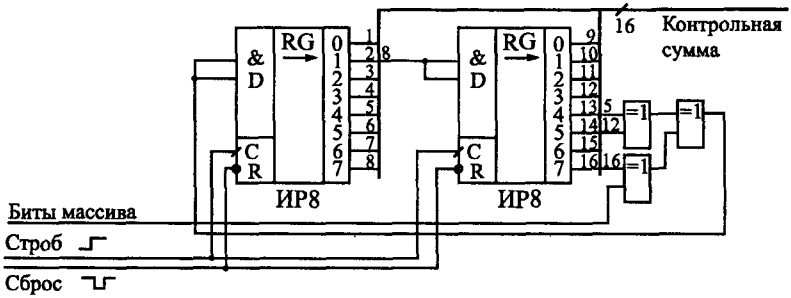


Рис. 8.8. Последовательный вычислитель 16-разрядной циклической контрольной суммы на регистре сдвига.

Может показаться, что такой последовательный вычислитель не слишком удобен из-за того, что данные массива должны подаваться на него в последовательном коде. Однако именно в последовательном коде передаются данные в информационных сетях, и в последовательном коде записываются данные на магнитные носители. Поэтому во всех подобных случаях последовательные вычислители подходят идеально.

Период поступления битов массива на последовательный вычислитель не должен быть меньше суммы задержки регистра сдвига и элементов Исключающее ИЛИ. В итоге предельная скорость вычисления циклической контрольной суммы оказывается значительно меньшей, чем в случае параллельного вычислителя. Это также недостаток данного метода вычисления.

8.3. Разработка логического анализатора

Логический анализатор — это контрольно-измерительный прибор, предназначенный для запоминания (фиксации) и последующего анализа (например, просмотра на экране) временных диаграмм большого количества цифровых сигналов. Логические анализаторы используются при динамической отладке различных цифровых устройств и систем, а также при контроле их работы. Совершенно незаменимы логические анализаторы при разработке и отладке различных микропроцессорных систем, контроллеров, компьютеров, где используется большое количество многоразрядных шин цифровых сигналов. Именно логические анализаторы позволяют разработчику увидеть те времен-

ные диаграммы, которые он рисует на бумаге при проектировании своего устройства, причем увидеть их в реальном масштабе времени, то есть посмотреть, как работает устройство на своей нормальной рабочей скорости.

Логический анализатор по своему назначению близок к осциллографу, так как он также позволяет наблюдать на экране временные диаграммы сигналов. Но существуют и существенные отличия логического анализатора от обычного (не цифрового) осциллографа:

- Логический анализатор работает только с цифровыми, то есть двухуровневыми (реже трехуровневыми) сигналами, а осциллограф — с аналоговыми сигналами, имеющими бесконечно большое число разрешенных уровней;
- Логический анализатор имеет большое количество входных линий (обычно от 16 до 64), то есть позволяют одновременно фиксировать множество входных сигналов, а осциллографы обычно позволяют одновременно увидеть не более четырех входных сигналов.
- Логический анализатор работает в режиме однократного запоминания временных диаграмм (как запоминающий осциллограф). То есть анализатор запоминает состояния входных сигналов в течение заданного времени (называемого окном регистрации), а затем дает возможность анализировать зафиксированные последовательности. Осциллограф же работает обычно в режиме непрерывной развертки, то есть он не запоминает формы входного сигнала и позволяет наблюдать только повторяющиеся, периодические сигналы.
- Логический анализатор предусматривает возможность так называемой предпусковой регистрации. Эта возможность предусматривается и в цифровых осциллографах, но ее нет в аналоговых осциллографах.

Рассмотрим подробнее, что такое предпусковая регистрация.

Процесс регистрации входных сигналов (или отображения их на экране в обычном осциллографе) всегда должен быть привязан к какому-то моменту времени, к какому-то внешнему событию, называемому запуском. Иначе разобраться в отображаемых сигналах будет совершенно невозможно. Например, в осциллографах моментом запуска обычно является момент пре-

вышения входным исследуемым сигналом установленного порога. Сигналом запуска может служить и специальный внешний синхронизирующий сигнал. В логических анализаторах в качестве запуска обычно используется момент появления на входах заданного уровня или заданной последовательности одного или нескольких входных сигналов.

В обычных осциллографах отображение формы входного сигнала (или входных сигналов) начинается в момент запуска, то есть на экране видно только то, что происходило со входными сигналами *после* момента запуска. Такая регистрация может быть названа послепусковой. Можно также сказать, что точка запуска всегда находится в начале окна регистрации (рис. 8.9).

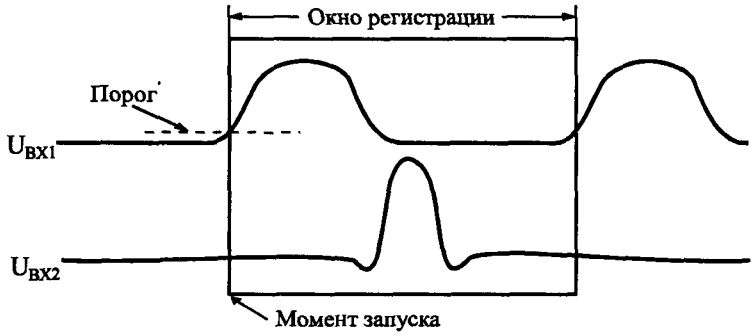


Рис. 8.9. Послепусковая регистрация в аналоговых осциллографах.

В логических анализаторах (и в цифровых осциллографах) существует возможность увидеть и зафиксировать не только то, что было *после* запуска, но еще и то, что происходило в течение определенного времени *до* момента запуска. Именно эта регистрация до момента запуска и называется предпусковой регистрацией. В этом случае точка запуска может находиться и в начале, и в середине, и в конце окна регистрации (рис. 8.10). Понятно, что такая возможность очень удобна, так как, выбирая величину длительности предпусковой регистрации, можно увидеть те события, временная привязка к началу которых затруднена или попросту невозможна. Длительность (глубина) предпусковой регистрации может быть постоянной (например, равной половине длительности окна регистрации) или переменной (то есть задаваться пользователем в пределах от нуля до полной дли-

тельности окна регистрации). При переменной глубине предпусковой регистрации точка запуска может располагаться в любой точке окна регистрации — от его начала до его конца.

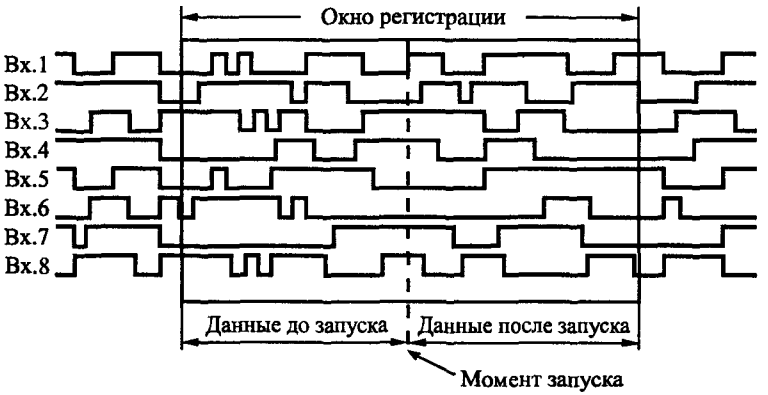


Рис. 8.10. Предпусковая регистрация в логических анализаторах и цифровых осциллографах.

С точки зрения схемотехники, логический анализатор представляет собой быстродействующую буферную оперативную память, работающую в периодическом режиме. Буфер этот однонаправленный. То есть сначала в буферную память с большой тактовой частотой последовательно записываются состояния нескольких входных сигналов, а затем эта информация последовательно читается из буфера. Таким образом, адреса буферной памяти могут перебираться одним и тем же счетчиком как в режиме записи, так и в режиме чтения. Структура таких буферов уже рассматривалась в разделе 6.2.2.

Главные особенности логического анализатора по сравнению со стандартной структурой информационного буфера на основе оперативной памяти следующие:

- большое число разрядов шины данных (то есть входных сигналов, каналов регистрации анализатора);
- необходимость обеспечения режима предпусковой регистрации;
- необходимость временной привязки процесса регистрации (записи в память) к состояниям входных сигналов (обеспечение запуска).

Первая из этих особенностей приводит к тому, что данные при чтении приходится считывать не все сразу, а по очереди (особенно при числе разрядов больше 32). Обычно данные требуется читать по 8 или по 16 разрядов. В результате усложняется та часть схемы буферной памяти, которая отвечает за чтение данных.

Вторая особенность требует существенного усложнения схемы счетчика, перебирающего адреса буферной памяти.

Наконец, третья особенность требует усложнения схемы управления работой информационного буфера.

Логические анализаторы делятся на синхронные (или анализаторы логических состояний) и асинхронные (или анализаторы временных диаграмм). Синхронные анализаторы работают от тактового генератора исследуемой схемы и фиксируют только временные сдвиги, кратные его периоду, а следовательно, выявляют только нарушения в логике работы схемы. Асинхронные анализаторы работают от собственного внутреннего тактового генератора, поэтому они позволяют измерять абсолютные значения временных сдвигов между сигналами и могут выявлять ошибки из-за неправильно рассчитанных задержек, из-за емкостных эффектов и т. д. Они обычно делаются гораздо более быстрыми, чем синхронные анализаторы (рассчитываются на предельно возможную частоту регистрации). В идеале логический анализатор должен обеспечивать оба эти режима работы, то есть работать как от своего внутреннего тактового генератора с разными тактовыми частотами, так и от внешнего тактового сигнала. То есть тактовый генератор анализатора должен быть также достаточно сложным.

Сформулируем исходные данные для проектирования логического анализатора. В данном случае нам важно не получить рекордные характеристики, а всего лишь продемонстрировать принципы разработки подобных схем на основе буферной памяти. Пусть количество входных линий анализатора (каналов регистрации) равно 32, количество регистрируемых состояний — 4096, максимальная тактовая частота — 10 МГц, тактовый генератор — внутренний с изменяемой частотой или внешний, запуск — по положительному или отрицательному фронту (синхропереходу) на одной из 8 входных линий, глубина предпусковой регистрации — задается программно. Будем также считать, что данные из памяти читаются порциями по 8 разрядов.

Таким образом, буферная оперативная память анализатора должна иметь объем 128 Кбит при организации 4К × 32. Помимо оперативной памяти анализатор должен включать в себя счетчик для перебора адресов с количеством разрядов не менее 12. В структуре анализатора должен быть также внутренний тактовый генератор с программно изменяемой частотой и возможностью подключения внешнего тактового сигнала. Наконец, необходимо наличие схемы запуска анализатора, которая будет выбирать одну из 8 входных линий и полярность синхроперехода (положительный или отрицательный фронт).

Память целесообразно выполнить на многоразрядных микросхемах ОЗУ (для снижения количества микросхем). Требования к быстродействию памяти в данном случае не слишком высоки (при максимальной тактовой частоте 10 МГц в течение 100 нс необходимо успеть переключить счетчик адресов и записать входную информацию в ОЗУ). Микросхем памяти, способных обеспечить такую скорость работы, достаточно много.

От счетчика адресов памяти требуется максимальное быстродействие (можно взять, например, микросхемы синхронных счетчиков КР531ИЕ17, которые достаточно легко каскадируются без потери быстродействия). Кроме простого перебора адресов счетчик должен также обеспечивать предпусковую регистрацию. Остановимся на этом несколько подробнее.

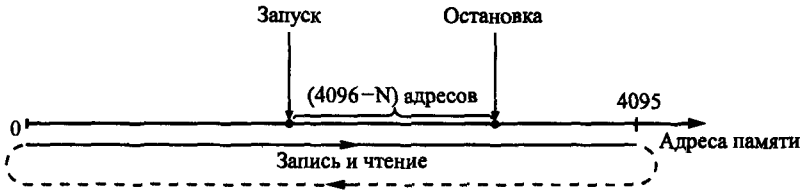


Рис. 8.11. Организация предпусковой регистрации.

Для того чтобы реализовать предпусковую регистрацию, необходимо обеспечить, непрерывную перезапись по кругу содержимого буферной памяти до момента прихода запуска (рис. 8.11).

То есть после записи последнего 4095 адреса надо записывать информацию по нулевому адресу. Если мы выбираем глубину предпусковой регистрации N тактов, то надо остановить

регистрацию через (4096–N) тактов после момента прихода запуска. После остановки регистрации надо считывать содержимое памяти, начиная с точки остановки, с перебором адресов в том же самом направлении, что и при регистрации. Проведя 4096 операций чтения содержимого памяти, мы получим информацию о состоянии входных сигналов в течение N тактов до запуска и (4096–N) тактов после запуска, то есть моменту прихода запуска будет соответствовать содержимое адреса памяти, считанного N-м.

Однако все произойдет именно таким образом только в том случае, если от момента начала регистрации до момента прихода запуска логический анализатор успеет зафиксировать N тактов. Иначе, остановив регистрацию через (4096–N), мы не перепишем всю память, и в части его адресов будет находиться предыдущая информация. Чтобы избежать этого, надо запретить реакцию на запуск в течение N тактов после начала регистрации (выдержать своеобразное «мертвое» время). А что будет, если запуск придет в течение этого самого «мертвого» времени? Если исследуемый процесс периодический (то есть все входные сигналы повторяются через какое-то время), то анализатор среагирует на следующий запуск после окончания «мертвого» времени. Если же исследуемый процесс однократный, не повторяющийся, то надо начать процесс регистрации заведомо раньше (на «мертвое» время или больше), чем начнется изучаемый процесс (например, если мы исследуем старт компьютера при включении питания).

В результате счетчики анализатора должны обеспечивать временную диаграмму, показанную на рис. 8.12.

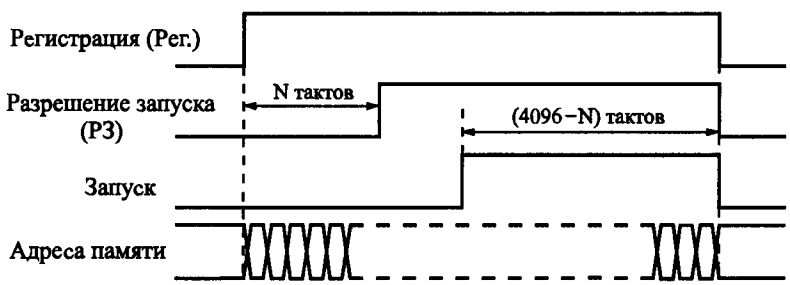


Рис. 8.12. Временная диаграмма работы счетчиков логического анализатора.

Адреса памяти начинают перебираться с началом регистрации. В течение N тактов после начала регистрации реакция на запуск запрещается, а затем разрешается. Через (4096–N) тактов после прихода запуска регистрация прекращается.

Отметим, что точно так же может быть реализована предпусковая регистрация в цифровом осциллографе. По сравнению с логическим анализатором в схему надо будет добавить только один или несколько АЦП и некоторые другие цифро-аналоговые узлы.

Спроектируем схему счетчиков, реализующую приведенную временную диаграмму.

Счетчик, перебирающий адреса памяти, должен быть 12-разрядным, так как $2^{12} = 4096$. Во время регистрации он должен работать в непрерывном режиме, реализуя постоянную перезапись по кругу всей буферной памяти. На этот же счетчик можно возложить функцию отсчета «мертвого» времени (N тактов). Но этот же счетчик не может отсчитывать еще и (4096–N) тактов после прихода запуска, так как запуск может прийти в любое момент после окончания «мертвого» времени. Для этого понадобится уже другой счетчик, причем также 12-разрядный.

Этот второй счетчик должен начинать работу только после прихода запуска (по сигналу Разрешение запуска) и должен отсчитывать всего (4096–N) тактов, после чего завершать регистрацию. То есть получается, что логический анализатор начинает регистрацию по внешнему управляющему сигналу, а заканчивает автоматически через (4096–N) тактов после запуска, о чем должен сообщать вовне сигнал флага окончания регистрации.

Таким образом, один 12-разрядный счетчик должен отсчитывать N тактов, а другой 12-разрядный счетчик должен отсчитывать (4096–N) тактов. Проще всего организовать такой режим, если в оба счетчика записывать до начала работы код N и задать первому счетчику инверсный режим счета, а второму — прямой режим счета. Перебор адресов памяти первым счетчиком начнется с адреса N и будет происходить на уменьшение (а не на увеличение, как на рис. 8.11), однако для работы буферной памяти это не имеет никакого значения.

Сигнал переноса первого счетчика появится через N тактов после начала регистрации. Этот сигнал должен разрешать ожидание запуска (сигнал РЗ на рис. 8.12). Когда же приходит запуск, то разрешается работа второго счетчика, начинающего считать с кода N на увеличение. В результате сигнал переноса второго

счетчика появится через (4096–N) тактов после начала его работы. Этот сигнал должен остановить процесс регистрации.

После окончания регистрации должен начаться процесс чтения из памяти. При этом первый счетчик должен перебирать адреса памяти по стробу чтения в том же направлении, что и при записи (то есть в режиме инверсного счета). 4096 последовательно произведенных циклов чтения позволит перебрать все 4096 адресов памяти, причем на N-м цикле чтения будет прочитан такт, в котором произошел запуск.

Схема счетчиков логического анализатора, реализующая описанный алгоритм, приведена на рис. 8.13.

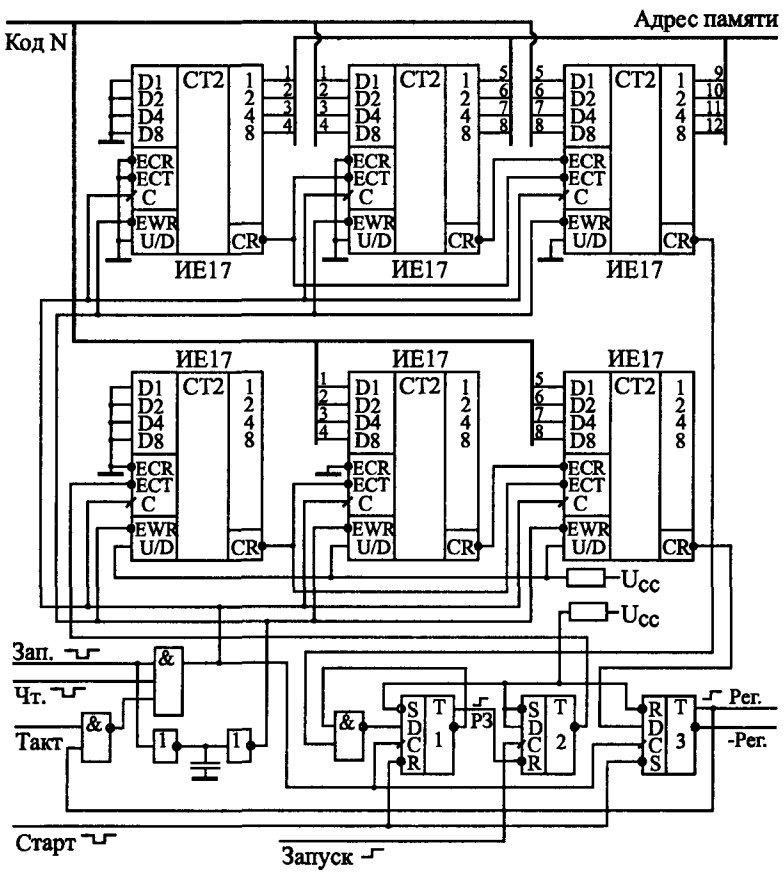


Рис. 8.13. Схема счетчиков логического анализатора.

Два 12-разрядных счетчика реализованы на шести микросхемах ИЕ17. Первый счетчик (верхний на схеме) работает в режиме инверсного счета, второй (нижний на схеме) — в режиме прямого счета. Перед началом работы в оба счетчика по внешнему сигналу -Зап. записывается код N. Причем значения четырех младших разрядов 12-разрядного кода N равны нулю, а записываются только 8 старших разрядов. Это приводит к тому, что глубина предпусковой регистрации может задаваться с точностью до 16 тактов и принимать значения из ряда: 0, 16, 32, 48, 64, ... , 4080. При записи на входы счетчиков -EWR и С поступают отрицательные сигналы, причем сигнал -EWR задержан относительно сигнала С на двух инверторах и (при необходимости) на конденсаторе. В результате положительный фронт сигнала С приходит тогда, когда сигнал -EWR равен нулю, что и требуется для записи.

На вход С обоих счетчиков могут приходить еще два сигнала: строб чтения из памяти -Чт. и тактовый сигнал Такт. Сигнал Такт приходит при регистрации (в режиме записи в память), а сигнал -Чт. поступает при чтении из памяти зарегистрированной информации.

После того как в оба счетчика записан код N необходимо начать регистрацию по внешнему сигналу -Старт. Этот сигнал сбрасывает в нуль триггеры 1 и 2 и устанавливает в единицу триггер 3. Выходной сигнал триггера 3 разрешает регистрацию (сигнал Рег. на рис. 8.12), то есть разрешает прохождение тактовых импульсов на входы счетчиков С. Это приводит к тому, что начинает работу первый счетчик, а второму счетчику работа запрещается по входу -ECT выходным сигналом триггера 2.

Первый счетчик, выходной код которого используется как адрес памяти, отсчитывает N тактов в инверсном режиме и вырабатывает сигнал переноса -CR. Этот сигнал перебрасывает в единицу триггер 1. Затем первый счетчик продолжает перебирать адреса памяти по кругу, а запись нулей в триггер 1 запрещается элементом 2И-НЕ на входе триггера. Выходной сигнал триггера 1 (сигнал РЗ на рис. 8.12) разрешает работу триггера 2 и тем самым разрешает реакцию схемы на сигнал запуска Запуск (положительный фронт).

Сигнал Запуск перебрасывает в единицу триггер 2, разрешая работу второго счетчика. Первый счетчик в это время продолжает считать. Второй счетчик начинает свой счет с кода N, досчитывает в режиме прямого счета до 4096 и своим сигналом пе-

реноса -CR перебрасывает в нуль триггер 3. Это приводит к запрету поступления тактовых импульсов на вход С счетчиков и к остановке регистрации.

После этого может начаться чтение записанной информации по стробу -Чт. По заднему (положительному) фронту этого сигнала первый счетчик будет перебирать адреса памяти в инверсном режиме. Второй счетчик также будет считать, но это не имеет никакого значения. После 4096 циклов чтения вся информация из памяти будет прочитана, и схема будет готова к новой регистрации.

Перейдем теперь к проектированию других узлов логического анализатора.

Как уже отмечалось, тактовый сигнал анализатора может быть как внутренним (от внутреннего тактового генератора), так и внешним (от исследуемой схемы). Для повышения универсальности анализатора целесообразно обеспечить его работу на нескольких тактовых частотах. Большие частоты будут использоваться для анализа быстрых процессов, а малые частоты — для анализа длительных процессов. Тактовая частота не должна при этом принимать слишком много значений. Вполне достаточно ряда нескольких частот, различающихся вдвое.

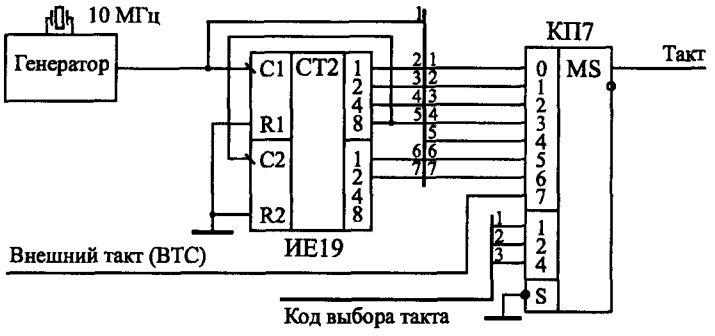


Рис. 8.14. Тактовый генератор логического анализатора

Пример схемы тактового генератора для логического анализатора приведен на рис. 8.14.

Тактовый генератор анализатора выполнен на кварцевом генераторе, 6-разрядном счетчике (ИЕ19) и 8-канальном мультиплексоре (КП7). Он может выдавать на выход ряд тактовых частот, различающихся в 2 раза (период 100, 200, 400, 800, 1600,

3200, 6400 нс) или внешний тактовый сигнал ВТС. То есть он позволяет реализовать как синхронный, так и асинхронный режим работы логического анализатора. Счетчик может быть применен асинхронный (ИЕ19), так как каждый его выход используется самостоятельно, независимо от других. Выбор канала мультиплексора (то есть вида тактового сигнала) осуществляется 3-разрядным кодом выбора такта, причем код 111 будет соответствовать внешнему тактовому сигналу.

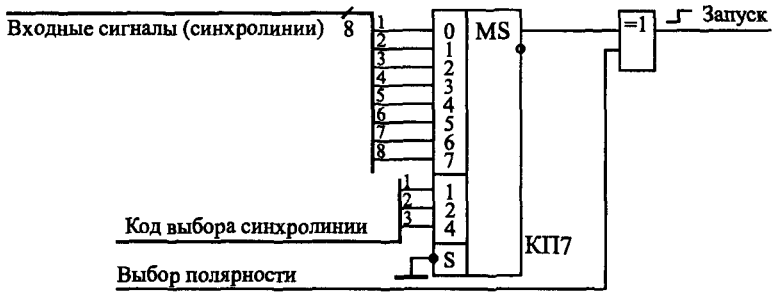


Рис. 8.15. Схема запуска логического анализатора.

Схема запуска анализатора должна обеспечивать выбор положительного или отрицательного фронта (синхроперехода) на одной из 8 входных линий анализатора. Выходным сигналом запуска является положительный фронт (см. рис. 8.13). Для выбора одного из восьми входных сигналов удобно использовать 8-канальный мультиплексор (КП7), а для выбора полярности перехода можно применить элемент Исключающее ИЛИ, включенный в режиме управляемого инвертора. Схема запуска получается очень простой (рис. 8.15). Выбор входного сигнала, по которому будет производиться запуск, осуществляется 3-разрядным управляющим кодом выбора синхролинии. Выбор полярности перехода производится внешним сигналом выбора полярности, причем единица на этом входе соответствует отрицательному фронту, а нуль — положительному фронту.

Наконец, последний узел логического анализатора — это память с буферами данных. Память должна иметь организацию 4К × 32, для чего придется использовать 4 микросхемы, так как обычно микросхемы оперативной памяти бывают 8-разрядные. Шина данных таких микросхем двунаправленная, поэтому тре-

буется применение буферов для данных. Память должна работать в режиме записи при регистрации и в режиме чтения при чтении информации, зарегистрированной логическим анализатором. Для упрощения схемы целесообразно использовать нетактируемые микросхемы памяти.

В режиме записи сигнал -CS памяти должен представлять собой отрицательные импульсы на каждый адрес памяти, а сигнал -WR должен быть постоянно активным (нулевым). Хотя некоторые микросхемы памяти (например, КР541РУ2) могут записывать информацию и при постоянных нулевых уровнях обоих сигналов -CS и -WR при изменении только адресов памяти. Использование таких микросхем еще более упрощает схему.

В режиме чтения сигнал -CS должен быть постоянно активным (нулевым), а сигнал -WR должен быть постоянно равен единице. Смена читаемой информации будет производиться только сменой адресов памяти.

Для чтения информации из памяти порциями по 8 разрядов надо применить четыре 8-разрядных однонаправленных буфера (типа АП5). Каждый из них будет открываться своим стробом чтения и выдавать на общую 8-разрядную шину данных по 8 разрядов читаемой из памяти информации. Таким образом, на чтение 32 разрядов из одного адреса памяти потребуется четыре цикла чтения из логического анализатора. Смена адреса памяти должна происходить после последнего из этих четырех циклов чтения. То есть для чтения всего объема памяти потребуется 16384 циклов чтения по 8 разрядов из логического анализатора.

Помимо буфера чтения необходимо применить также входной 32-разрядный буфер, который будет пропускать входные (регистрируемые) сигналы в память в режиме регистрации и будет закрываться после окончания регистрации. Назначение этого буфера состоит в том, чтобы не выдавать на входные линии логического анализатора читаемую из памяти по двунаправленной шине данных информацию. Этот буфер также можно построить на микросхемах однонаправленных буферов типа АП5.

В результате схема памяти логического анализатора будет иметь вид, показанный на рис. 8.16.

Объединение четырех микросхем памяти производится стандартным образом: объединяются одноименные разряды адреса, сигналы -CS и -WR всех микросхем. На входы -CS подается сигнал, равный нулю при отсутствии регистрации (нулевой

сигнал Рег.) и равный инверсному тактовому сигналу при регистрации (то есть отрицательному импульсу на каждый адрес памяти). Минимальная длительность импульса -WR равна в режиме записи половине периода тактового сигнала с частотой 10 МГц, то есть 50 нс, поэтому память должна успевать за это время записать информацию. На входы -WR подается сигнал -Рег., равный нулю при регистрации и единице при отсутствии регистрации. Если память может записывать информацию при постоянных нулевых сигналах -CS и -WR, то на вход -CS можно постоянно подать нулевой уровень.

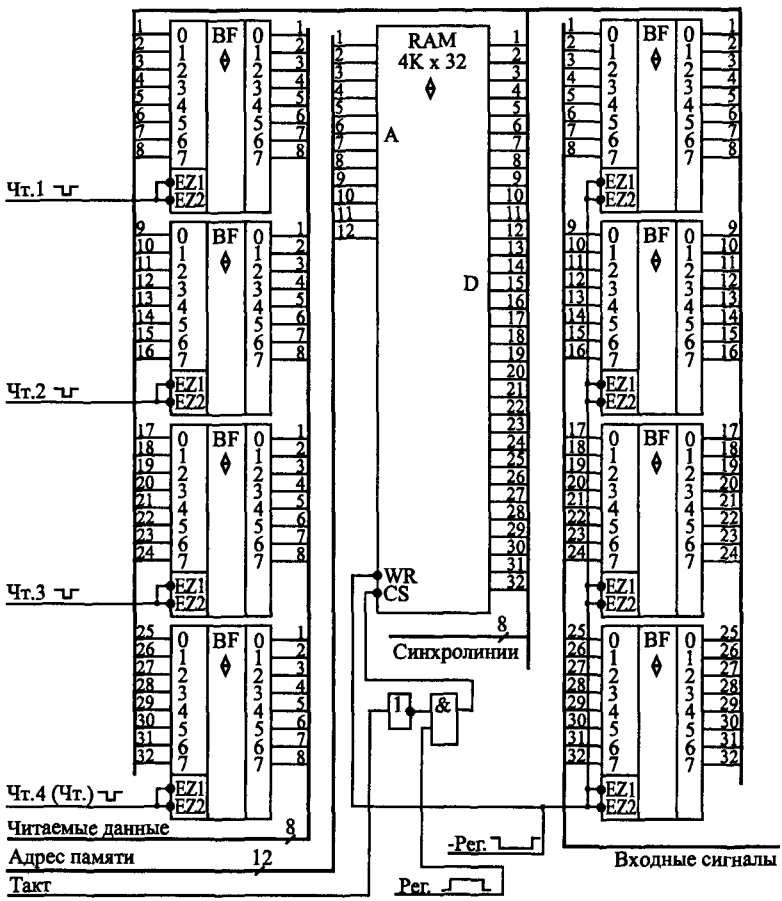


Рис. 8.16. Схема памяти логического анализатора.

Четыре микросхемы АП5 входного буфера (справа по рисунку) управляются сигналом -Рег., то есть они открываются на все время регистрации и закрываются, когда регистрации нет.

Четыре микросхемы АП5 буфера чтения открываются на чтение (выдают 8-разрядные читаемые данные) каждая свои стробом чтения: -Чт.1 ... -Чт.4. При этом сигнал -Чт.4 (его задний фронт) используется для переключения счетчиков адреса памяти при чтении (см. сигнал -Чт. на рис. 8.13). Для чтения информации из одного адреса памяти надо последовательно подать сигналы -Чт.1 ... -Чт.4, после чего адрес переключится на следующий.

В качестве синхролиний для запуска анализатора (см. рис. 8.15) используются восемь разрядов данных памяти. В режиме регистрации на них приходят 8 входных сигналов логического анализатора.

Таким образом, схема логического анализатора спроектирована полностью.

Сформулируем теперь порядок управления этой схемой. Управлять ей может, например, компьютер или контроллер, который будет отображать, обрабатывать и хранить зарегистрированные последовательности входных сигналов.

Перед началом регистрации необходимо записать в счетчики анализатора код N (количество тактов предпусковой регистрации) по сигналу записи -Зап. (см. рис. 8.13). Необходимо также установить 7-разрядный управляющий код, который определит режим работы анализатора. Три разряда этого кода задают тип тактового сигнала анализатора (см. рис. 8.14). Три разряда выбирают номер входного сигнала (из восьми возможных), фронт на котором будет служить запуском, а последний седьмой разряд определит полярность этого фронта, синхроперехода (см. рис. 8.15). После этого можно начинать регистрацию по сигналу -Старт (см. рис. 8.13). Анализатор отсчитает «мертвое» время, зафиксирует приход синхроперехода (запуск) и остановит регистрацию через нужное количество тактов после запуска. Узнать о том, что регистрация завершилась, можно исходя из анализа сигнала Рег. (см. рис. 8.13). Затем можно начинать чтение из буферной памяти анализатора по стробам -Чт.1 ... -Чт.4 (см. рис. 8.16). Когда информация из всех 4096 адресов памяти будет прочитана, анализатор снова будет готов к регистрации.

8.4. Разработка генератора аналоговых сигналов

Цифровые генераторы (или, как их еще называют, синтезаторы) аналоговых сигналов произвольной формы часто используются при отладке различных аналоговых и аналого-цифровых устройств и систем. Они позволяют не только получить сигналы разных стандартных и нестандартных форм, но и обеспечить высокую точность задания амплитуды и частоты сигнала, не достижимые в случае обычных аналоговых генераторов. Цифровые генераторы работают обычно под управлением компьютеров или контроллеров, что обуславливает большие удобства пользователя и широкие возможности по заданию разнообразных форм сигналов и по их хранению.

Мы будем разрабатывать довольно простой генератор, рассчитанный на звуковой диапазон частот выходного сигнала 20 Гц ... 20 кГц (период от 50 мкс до 50 мс). Генератор должен формировать сигналы произвольной формы с амплитудой, задаваемой управляющим кодом. Генератор должен работать в режиме автоматической (периодической) генерации, а также в режиме разовой генерации с остановкой генерации после окончания одного периода выходного сигнала. Управление работой генератора должно быть полностью цифровым.

Отметим, что в реальности сигналы сложной формы, как правило, бывают низкочастотными. Они встречаются, например, при виброиспытаниях, в медицинской технике, в сейсмической технике и т. д. Высокочастотные сигналы обычно имеют довольно простую форму, например, синусоидальную. Поэтому наш простой генератор, рассчитанный на невысокие частоты, будет, тем не менее, удовлетворять требованиям довольно широкого спектра применений.

Разработку генератора мы начнем «с конца», то есть с того выходного сигнала, который он должен формировать.

Как уже отмечалось в главе 7 (см. раздел 7.1), выходной сигнал ЦАП $U_{ЦАП}$ представляет собой ступенчатую функцию, которую можно представить в виде суммы идеального («гладкого») аналогового сигнала $U_{Вых}$ и пилообразного сигнала помехи $U_{ПОМ}$ (рис. 8.17).

Сигнал помехи $U_{ПОМ}$ имеет основную частоту, равную частоте поступления входных кодов на ЦАП. Для сглаживания ступенек выходного сигнала ЦАП и приближения его к идеаль-

ному сигналу $U_{ВЫХ}$ можно применить простой аналоговый фильтр низкой частоты (ФНЧ), который должен существенно ослаблять сигнал помехи, но не ослаблять полезный выходной сигнал генератора. В примере на рис. 8.17 частота полезного сигнала в 16 раз меньше частоты сигнала помехи, поэтому эта задача фильтрации не слишком сложна. Однако от генератора сигналов произвольной формы может понадобиться синтез выходных сигналов с крутыми фронтами (например, прямоугольных или пилообразных сигналов). В этом случае применение такого выходного фильтра низкой частоты может исказить выходные сигналы, затянув их фронты. Поэтому целесообразно предусмотреть два выхода генератора: один с низкочастотной фильтрацией, а другой без нее.

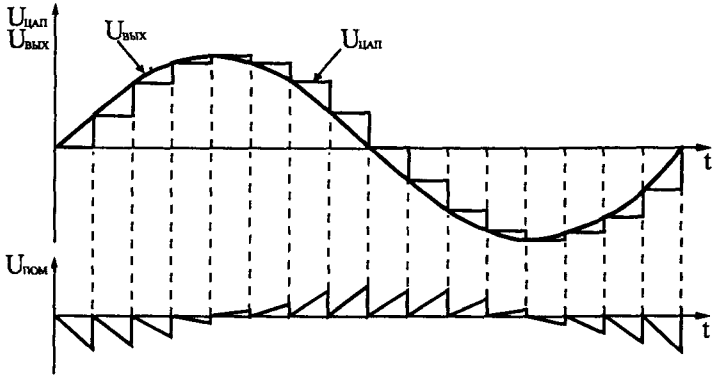


Рис. 8.17. Цифровая генерация аналогового сигнала.

Помимо фильтра низкой частоты выходной узел генератора сигналов должен содержать схему задания амплитуды выходного сигнала. В случае использования оперативной памяти для хранения кодов выборок выходного сигнала схема задания амплитуды может и отсутствовать. При этом в память необходимо заносить коды выборок сигнала с нужной амплитудой. Однако такой подход не слишком удобен, так как он требует пересчета всех кодов выборок для каждой новой амплитуды сигнала выбранной формы. Гораздо удобнее сделать так, чтобы в памяти всегда хранились коды выборок сигнала с максимально возможной амплитудой, а выходной сигнал с ЦАП ослаблялся управляемым аттенюатором в нужное количество раз.

В результате схема выходного узла генератора аналоговых сигналов будет включать в себя еще и управляемый аттенюатор, рассмотренный в разделе 7.1 (рис. 8.18).

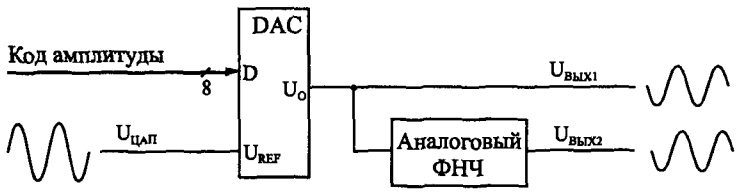


Рис. 8.18. Схема выходного узла генератора.

Аналоговый фильтр низких частот должен иметь коэффициент передачи в полосе пропускания, равный единице, и частоту среза, обеспечивающую эффективное подавление сигнал помехи. Тип схемы фильтра и его порядок не слишком важны. Для удобства пользователя целесообразно сделать фильтр неинвертирующим, чтобы выходные сигналы на обоих выходах генератора ($U_{ВЫХ1}$ и $U_{ВЫХ2}$) были одной полярности. Аттенюатор управляется 8-разрядным кодом амплитуды, что обеспечивает коэффициент деления сигнала от 1/256 до 1. Если амплитуда исходного сигнала $U_{ЦАП}$ равна 10 В, то амплитуда выходного сигнала ($U_{ВЫХ1}$ и $U_{ВЫХ2}$) может быть задана с точностью около 40 мВ. Увеличение разрядности кода амплитуды потребовало бы принятия специальных мер, так как слишком малые аналоговые сигналы сильно искажаются шумами и помехами по цепям питания. ЦАП необходимо применять умножающий с биполярным выходом, чтобы обрабатывать как положительные, так и отрицательные выходные сигналы.

Теперь переходим к проектированию собственно цифровой части генератора.

Как уже отмечалось в разделе 7.1, основной узел генератора должен представлять собой буферную оперативную память с периодическим режимом работы. Причем буфер этот должен быть однонаправленным. Перед началом работы в буфер заносится массив кодов выборок синтезируемого сигнала, а во время работы генератора адреса памяти опрашиваются в нужном темпе, и выходные коды памяти подаются на ЦАП, формирующий аналоговый сигнал $U_{ЦАП}$. Проблема состоит в выборе нужного

объема памяти и в способе перебора адресов для обеспечения нужной частоты выходного сигнала. Память может также быть постоянной (ПЗУ), если необходимо формировать одну или несколько постоянных форм сигналов. В этом случае операция записи в память исключается, но проблема выбора способа перебора адресов памяти остается.

Существует два основных способа перебора адресов памяти генератора аналоговых сигналов, каждый из которых имеет свои достоинства и недостатки.

Первый, простейший способ предусматривает перебор адресов памяти генератора с помощью обычного двоичного счетчика. В данном случае опрашиваются все адреса памяти подряд. Изменение частоты аналогового выходного сигнала генератора производится с помощью изменения тактовой частоты этого счетчика, для чего используется тот или иной управляемый делитель частоты опорного кварцевого генератора (рис. 8.19). Частота выходного сигнала будет определяться при таком решении по формуле: $f_{вых} = f_Г/(2^n N)$, где $f_Г$ — частота задающего кварцевого генератора, N — управляющий код делителя частоты, n — разрядность счетчика (разрядность шины адреса памяти).

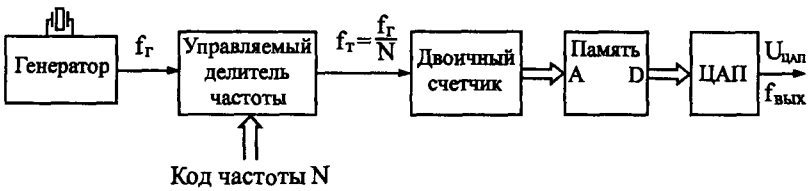


Рис. 8.19. Опрос памяти с помощью двоичного счетчика.

Главное достоинство данного подхода состоит в том, что при изменении частоты выходного сигнала не меняется точность воспроизведения формы выходного сигнала. Ведь точность воспроизведения формы аналогового сигнала зависит в первую очередь от количества выборок, приходящихся на период выходного сигнала, а в данном случае оно постоянно и равно количеству адресов памяти. Например, если память имеет 1К адресов, то выходной сигнал при любой частоте будет задаваться с помощью 1024 точек, он всегда будет иметь 1024 ступеньки.

Однако данное решение имеет и серьезные недостатки. Основной его недостаток состоит в том, что частота сигнала помехи в данном случае прямо пропорциональна частоте выходного аналогового сигнала генератора (она больше частоты выходного сигнала во столько раз, сколько адресов имеет память). Например, при 1К адресов памяти частота сигнала помехи в 1024 раз больше частоты выходного сигнала. И при изменении частоты выходного сигнала в 1000 раз также в 1000 раз будет изменяться частота сигнала помехи. Отфильтровать такую помеху переменной частоты чрезвычайно трудно, если не невозможно, так как требуется применение фильтра с частотой среза, изменяемой в очень широких пределах.

Другой существенный недостаток данного метода связан с высокими требованиями к быстродействию ЦАП. Например, если максимальная частота выходного аналогового сигнала генератора должна быть 20 кГц, а память имеет 1К адресов, то ЦАП должен успевать работать с частотой более 20 МГц, то есть иметь время установления менее 50 нс. При большей частоте выходного сигнала и при большем объеме памяти требования к быстродействию ЦАП будут еще выше. И с такой же скоростью должна работать буферная память, то есть требования к быстродействию памяти также велики.

Второй возможный способ перебора адресов памяти генератора аналоговых сигналов состоит в применении накапливающего сумматора с переменным шагом суммирования (рис. 8.20).

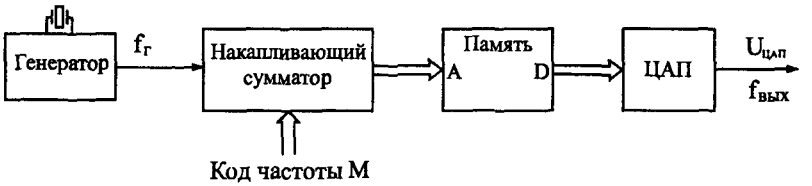


Рис. 8.20. Опрос памяти с помощью накапливающего сумматора.

В память, как и в предыдущем случае, заносится массив кодов выборок периода требуемого сигнала. Но при генерации опрашиваются не все адреса памяти подряд, а только адреса с шагом, задаваемым входным кодом накапливающего сумматора M (см. раздел 4.2.1). Чем больше этот шаг, тем быстрее будет

пройден весь объем памяти и тем больше будет частота выходного сигнала генератора. И соответственно, чем меньше шаг, тем больше времени потребуется на опрос всех адресов памяти, тем меньше будет частота выходного сигнала генератора.

При изменении шага опроса памяти изменяется и количество выборок на период выходного сигнала, что приводит к изменению точности воспроизведения формы сигнала. Количество выборок К на период выходного сигнала вычисляется по формуле: $K = 2^n/M$, где n — количество разрядов адреса памяти, М — управляющий код накапливающего сумматора. А частота выходного аналогового сигнала определяется формулой: $f_{ВЫХ} = f_Г M /2^n$, где $f_Г$ — частота задающего кварцевого генератора. То есть выходная частота прямо пропорциональна управляющему коду М, а не обратно пропорциональна, как в предыдущем случае.

Главное достоинство данного подхода состоит в том, что сигнал помехи на выходе всегда имеет одну и ту же частоту, равную частоте задающего кварцевого генератора $f_Г$, независимо от частоты выходного аналогового сигнала. Поэтому такую помеху легко отфильтровать, никакой перестройки частоты среза фильтра не требуется.

Другое важное достоинство данного решения состоит в том, что по мере роста частоты выходного сигнала генератор сам пропорционально уменьшает количество выборок на период выходного сигнала, поэтому требования к быстродействию ЦАП, формирующего выходной сигнал, не слишком жесткие. ЦАП может быть в несколько раз более медленным, чем в предыдущем случае, при такой же максимальной выходной частоте. Или, можно сказать и так, при том же самом ЦАП генератор может выдавать выходные сигналы с гораздо более высокой частотой. Точно так же снижаются и требования к быстродействию памяти. Это приводит к тому, что объем памяти в данном случае может быть гораздо больше, чем в предыдущем.

Но ничто не дается даром, поэтому данный метод имеет и существенный недостаток. С ростом частоты выходного сигнала его форма будет передаваться все более грубо, ступеньки будут все больше. На рис. 8.21 приведен пример воспроизведения формы синусоидального сигнала, записанного в память объемом 32К × 8 для двух разных шагов наращивания адреса М (количество выборок на период К = 16 и К = 48). Понятно, что точность воспроизведения формы сигнала сильно зависит от кода М. Это может

привести к тому, что некоторые фрагменты сигналов сложной формы могут быть пропущены. К тому же в случае, когда количество выборок на период выходного сигнала К не равно целому числу, периоды выходного сигнала будут несколько отличаться один от другого. Несколько смягчает этот недостаток уже упоминавшееся обстоятельство, что в природе сигналы сложной формы обычно низкочастотные, а именно низкочастотные сигналы воспроизводятся при данном методе наиболее точно.

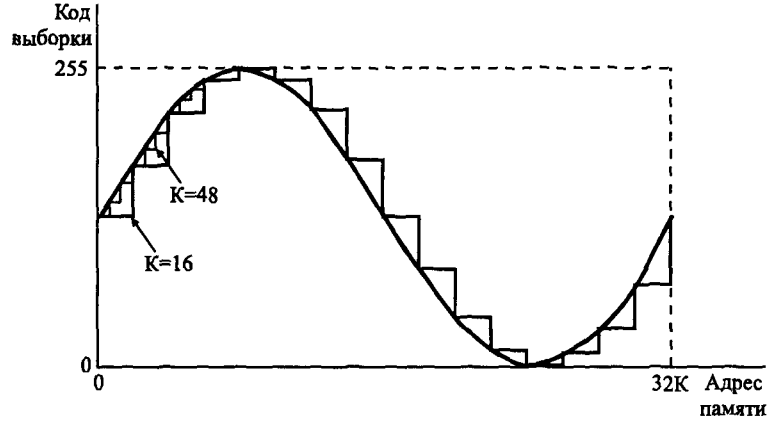


Рис. 8.21. Опрос памяти с разными шагами (количество выборок на период К = 16 и К = 48).

Исходя из всех этих соображений, останавливаем свой выбор именно на этом, втором методе.

Примем для дальнейшего проектирования, что минимальное количество выборок на период выходного сигнала будет равно 32, а максимальное будет равно количеству адресов памяти. Так как от генератора требуется большой диапазон выходных частот (частоты могут различаться в 1000 раз), объем памяти должен быть большим. Если минимальное количество выборок на период равно 32, то максимальное количество выборок на период потребуется в тысячу раз больше, то есть 32000. Поэтому количество адресов памяти не должно быть меньше 32000. Возьмем память с количеством адресов, равным 32К.

Количество разрядов данных памяти, определяющее точность задания величины выборок выходного сигнала, не стоит

брать слишком большим. Ведь на формируемый аналоговый сигнал будут накладываться помехи от цифровой части схемы, поэтому чрезмерно точное задание величин выборок выходного сигнала окажется попросту излишним. Поэтому выберем количество разрядов данных памяти равным 8, то есть память будет иметь организацию 32К × 8.

Спроектируем накапливающий сумматор для генератора аналоговых сигналов.

Как уже отмечалось, частота выходного аналогового сигнала прямо пропорциональна управляющему коду накапливающего сумматора М. Абсолютная погрешность установки частоты составит 0,5/М. Поэтому для малых частот погрешность установки частоты будет максимальной. Например, если коду М = 1 будет соответствовать частота 20 Гц, то следующее разрешенное значение частоты будет равно 40 Гц (при М = 2). Это не слишком удобно, хорошо бы иметь точность установки частоты не ниже хотя бы 10% во всем частотном диапазоне. Возьмем, например, абсолютную погрешность установки частоты 0,5 Гц. Значит, при М = 1 генератор должен выдавать частоту 1 Гц. Такие низкие частоты мы можем просто не использовать, зато частота 20 Гц (при М = 20) будет иметь точность установки 2,5%. Разрешенные значения частот вблизи 20 Гц составят при этом 19 Гц, 20 Гц, 21 Гц.

Выберем теперь величину тактовой частоты накапливающего сумматора (то есть частоты задающего кварцевого генератора). Максимальная частота выходного сигнала нашего генератора должна быть равна 20 кГц, при этом на период выходного сигнала должно приходиться 32 выборки. То есть тактовая частота накапливающего сумматора должна быть не менее 20 кГц • 32 = 640 кГц. Выберем с запасом тактовую частоту равной 1 МГц. Максимальная частота выходного аналогового сигнала при 32 выборках на период будет при этом составлять 1 МГц/32 = 31,25 кГц.

Количество разрядов накапливающего сумматора должно быть таким, чтобы он обеспечивал весь выбранный частотный диапазон. Нетрудно подсчитать, что нам потребуется 20-разрядный накапливающий сумматор (так как 2^{20} = 1048576), то есть при тактовой частоте 1 МГц минимальный период выходного сигнала составит 1048576 тактов или чуть более одной секунды, что примерно соответствует частоте выходного сигнала 1 Гц.

Если использовать 4-разрядные микросхемы полных сумматоров (ИМ3 или ИМ6), то для построения 20-разрядного сумматора потребуется 5 микросхем сумматоров. Для запоминания выходного кода сумматоров надо будет использовать три микросхемы 8-разрядных регистров, причем регистры эти должны быть со входом сброса (например ИР35) для начального сброса накапливающего сумматора.

Получившаяся в итоге схема накапливающего сумматора приведена на рис. 8.22. В качестве тактового сигнала в режиме генерации она использует сигнал с кварцевого генератора частотой 1 МГц (разрешающий сигнал Ген.), а в режиме записи в память кодов выборок — строб записи в память -Зап. На входы адреса памяти подаются сигналы 15 старших выходных разрядов накапливающего сумматора, а 5 младших разрядов накапливающего сумматора не используются. Код частоты М подается на 15 младших входных разрядов накапливающего сумматора, а на старшие 5 разрядов поданы нулевые сигналы. В результате при максимальном коде М = 32767 накапливающий сумматор будет переполняться за 32 такта (выходная частота 31,25 кГц), а при минимальном коде М = 1 — за 1048576 тактов (выходная частота около 1 Гц).

Перед началом записи в память накапливающий сумматор должен быть сброшен в нуль сигналом -Сброс НС. Во время записи в память каждый строб записи -Зап. должен увеличивать на единицу адрес памяти, поэтому код частоты М должен быть установлен в данном режиме равным 32 (двоичный код 100000).

Условия правильной работы накапливающего сумматора следующие. За период сигнала тактового генератора должны успеть сработать регистр и сумматор. В нашем случае это условие довольно легко выполняется, так как период сигнала тактового генератора 1 мкс. Но при построении более высокочастотных генераторов аналоговых сигналов требуется более высокая тактовая частота, и при этом может уже сказаться накопление задержек переноса пяти микросхем сумматоров. При тактовой частоте больше 10 МГц это уже может вызвать большие проблемы. Точно так же за период следования стробов записи в памяти -Зап. должны успевать срабатывать регистр и сумматоры. Это условие обычно значительно проще выполнить, чем первое.

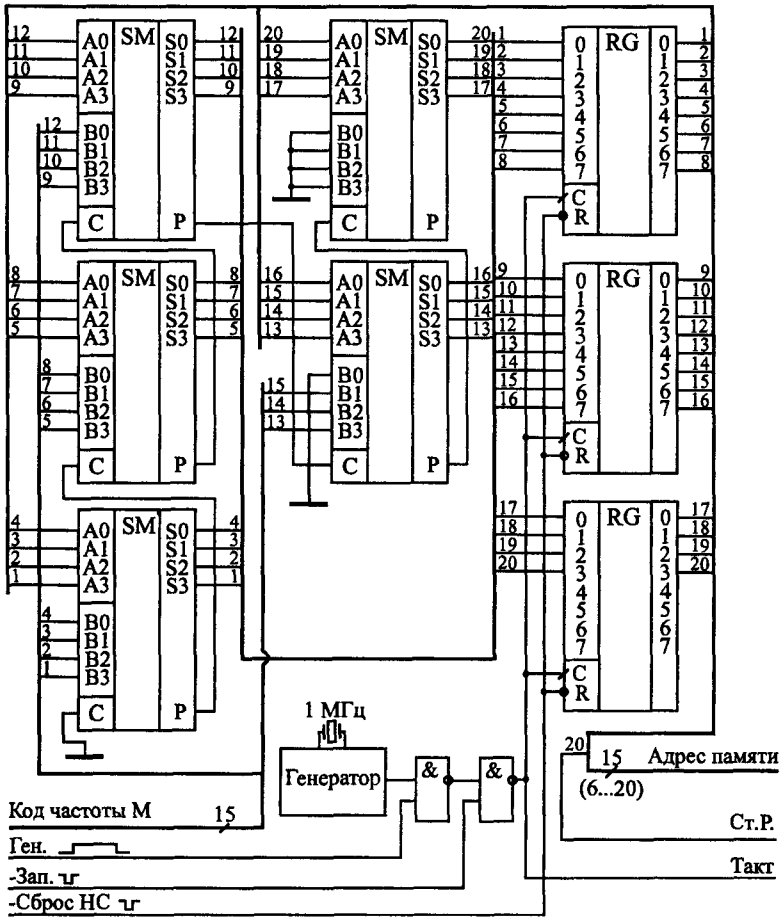


Рис. 8.22. Накапливающий сумматор генератора аналоговых сигналов.

Посмотрим, какой будет частота сигнала помехи, и какой должна быть частота среза выходного аналогового низкочастотного фильтра (см. рис. 8.18). При управляющем коде частоты М больше или равном 32 каждый тактовый импульс будет вызывать изменение адреса памяти. Поэтому частота помехи будет равна частоте тактового генератора (1 МГц). Это соответствует частоте выходного сигнала, большей 32 Гц. Однако нам надо обеспечить нижнюю частоту выходного аналогового сигнала 20 Гц.

Если код частоты М будет лежать в пределах от 16 до 31, то адрес памяти будет изменяться не реже одного раза на два такта тактового генератора. Частота помехи будет не менее 500 кГц. То есть при частоте выходного сигнала, большей 16 Гц, частота сигнала помехи будет в пределах от 500 кГц до 1 МГц. Максимальная частота выходного аналогового сигнала равна 31,25 кГц. Значит, частота среза фильтра должна быть такой, чтобы сильно ослаблять частоты, большие 500 кГц, но не искажать частоты, меньшие 31,25 кГц. Эти частоты различаются в 16 раз, поэтому фильтр построить не слишком сложно.

В результате мы получаем, что выбранная схема накапливающего сумматора обеспечивает диапазон частот выходного аналогового сигнала от 16 Гц до 31,25 кГц, причем погрешность установки частоты составляет 0,5 Гц во всем частотном диапазоне. Количество выборок сигнала на период будет изменяться от 32 на верхнем краю частотного диапазона до 32К на нижнем краю частотного диапазона. Это вполне удовлетворяет требованиям к генератору, сформулированным в начале данного раздела.

Переходим теперь к проектированию схемы управления для генератора аналоговых сигналов.

Схема управления генератора должна обеспечивать два режима работы: режим записи в память и режим генерации. Причем генерация может быть как автоматическая (периодическая), так и разовая. Эти режимы реализуются простой схемой на двух триггерах (рис. 8.23).

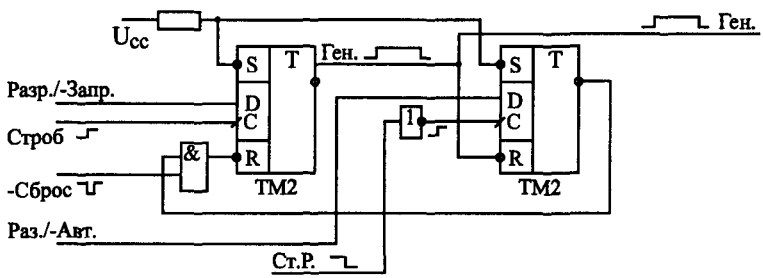


Рис. 8.23. Схема управления для генератора аналоговых сигналов.

Первый (левый на схеме) триггер служит для разрешения или запрещения генерации. По внешнему сигналу Строб (положительный фронт) в него записывается единица для разрешения генера-

ции или нуль для запрещения генерации. Выходной сигнал Ген. используется для разрешения тактовых импульсов накапливающего сумматора (см. рис. 8.22) и для управления остальной частью схемы. Перед началом работы генератора этот триггер сбрасывается в нуль внешним сигналом начального сброса -Сброс.

Второй (правый на схеме) триггер служит для организации режима разового запуска генератора. При запрете генерации этот триггер сброшен в нуль сигналом Ген. (единица на инверсном выходе). При разрешении генерации этот триггер срабатывает по отрицательному фронту на старшем разряде накапливающего сумматора (сигнал Ст.Р. со схемы на рис. 8.22), то есть по переполнению накапливающего сумматора, возникающему после окончания одного периода аналогового сигнала. Если внешний управляющий сигнал Раз./-Авт. установлен в нуль (автоматический запуск), то ничего не происходит, триггер остается сброшенным. Если же внешний сигнал Раз./-Авт. установлен в единицу (разовый запуск), то после окончания одного периода выходного аналогового сигнала генератора второй триггер перебросится в единицу (нуль на инверсном выходе) и сбросит тем самым первый триггер, запретив генерацию. Узнать об этом можно, анализируя флаг генерации — сигнал Ген. Для нового разрешения генерации надо снова записать единицу в первый триггер.

Наконец, последний узел генератора аналоговых сигналов — это память с ЦАП.

Прежде всего надо обеспечить, чтобы ЦАП, формирующий выборки аналогового сигнала по кодам из памяти, выдавал как положительные, так и отрицательные сигналы, то есть был биполярным. Это существенно повысит универсальность генератора. ЦАП должен формировать выходное напряжение (а не выходной ток), что позволит более просто обрабатывать выходной сигнал выходным узлом (*см.* рис. 8.18). Требования к быстродействию ЦАП в нашем случае невелики: коды всегда поступают на него с периодом в 1 мкс, значит, за это время ЦАП должен успеть установить свое выходное напряжение. Таких ЦАП существует довольно много.

Опорное напряжение ЦАП удобно выбирать равным 10 В, что обеспечит размах выходного сигнала от −10 В до +10 В. При этом шаг изменения выходного сигнала (минимально возможная высота ступеньки) составит 20 В / 256, то есть около 80 мВ. Но это только для сигнала максимальной амплитуды 10 В. Если

же требуется генерация сигнала с амплитудой 1 В (ослабление выходным аттенюатором в 10 раз), то шаг изменения выходного сигнала будет около 8 мВ.

Входной код ЦАП (то есть выходной код буферной памяти) должен фиксироваться в параллельном регистре, чтобы все разряды этого кода подавались на входы ЦАП одновременно. В момент отсутствия генерации на выходе ЦАП должно быть нулевое напряжение, поэтому данный регистр должен иметь вход сброса, на который подается сигнал Ген. Однако надо учитывать, что при биполярном выходе ЦАП нулевому уровню выходного сигнала соответствует не нулевой код 00000000, а код 10000000 (с единицей в старшем разряде). Поэтому регистр должен сбрасываться не в нуль, а именно в состояние 10000000. При этом просто поставить дополнительный инвертор на старший разряд кода нельзя, так как он внесет задержку, и старший разряд кода будет устанавливаться позже остальных разрядов, что может вызвать недопустимо большие выбросы выходного напряжения. Поэтому этот входной регистр ЦАП должен иметь как прямые, так и инверсные выходы (например, ТМ8), причем все разряды кроме старшего надо брать с прямых выходов регистра, а старший разряд — с инверсного выхода. Это обеспечит одновременное изменение всех разрядов кода. Для компенсации инверсии старшего разряда надо дополнительно проинвертировать сигнал старшего разряда на входе регистра.

Память выборок сигнала целесообразно использовать многоразрядную с совмещенной входной и выходной шинами данных, что позволит упростить схему. Микросхемы с организацией 32К × 8 выпускаются многими фирмами. Память лучше брать нетактируемую, чтобы в режиме чтения (при генерации) можно было постоянно подавать на вход -CS сигнал логического нуля. Быстродействие памяти не слишком критично, так как перебор адресов происходит довольно медленно. За период тактового сигнала (1 мкс) в режиме чтения должен успеть сработать регистр накапливающего сумматора, и память должна успеть выдать читаемый код (с задержкой выборки адреса).

Совмещенная шина входных/выходных данных памяти требует применения однонаправленного входного буфера (например, АП5), через который в режиме записи на память будут подаваться записываемые в память коды выборок генерируемого сигнала. Буфер должен открываться тем же сигналом, который подается на вход -WR памяти. Во время генерации буфер должен быть закрыт.

В результате схема буферной памяти с ЦАП для генератора аналоговых сигналов будет выглядеть так, как показано на рис. 8.24.

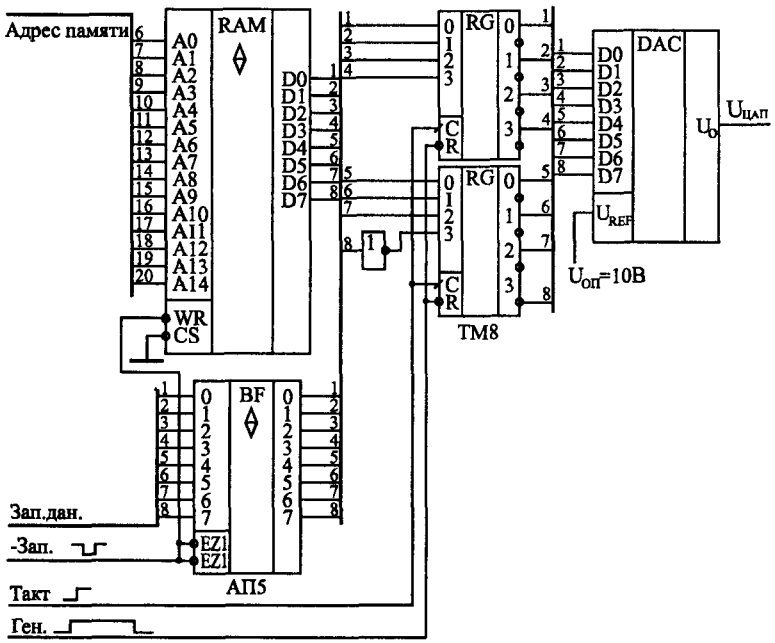


Рис. 8.24. Память и ЦАП генератора аналоговых сигналов.

Перед началом работы в память должны быть записаны коды выборок (8-разрядная шина Зап.дан.) по стробу -Зап. Данные должны выставляться до начала строба и сниматься после его окончания. Во время строба записи -Зап. память переходит в режим записи (сигнал -WR), а буфер открывается (сигналы -EZ1 и -EZ2). За счет задержки буфера записываемые данные снимаются со входов данных памяти позже, чем заканчивается сигнал -Зап. Поэтому данные записываются в память. По окончании сигнала -Зап. происходит смена адреса памяти (см. рис. 8.22). Всего должно быть проведено 32К циклов записи для полного заполнения памяти.

Когда начинается генерация (сигнал Ген.), адреса памяти перебираются накапливающим сумматором, а читаемая из них

информация записывается по сигналу Такт (см. рис. 8.22) в 8-разрядный регистр (две микросхемы ТМ8), а затем поступает на входы ЦАП. В результате выдача выборок выходного сигнала ($U_{ЦАП}$) задерживается на один такт относительно момента чтения из памяти, но эта задержка, как правило, не имеет никакого значения. После окончания генерации регистр сбрасывается в состояние 10000000, соответствующее нулю выходного сигнала $U_{ЦАП}$. Так как по сигналу начального сброса -Сброс (см. рис. 8.23) генерация запрещается, на выходе генератора в этот момент также будет нулевое напряжение.

Таким образом, схема генератора аналоговых сигналов полностью спроектирована.

Сформулируем теперь последовательность действий, которые надо предпринимать для управления работой генератора.

После включения питания надо подать сигнал начального сброса -Сброс (см. рис. 8.23), который запретит генерацию и обеспечит нулевой уровень выходного напряжения генератора.

Затем необходимо записать в память массив кодов выборок сигнала требуемой формы. Для этого код частоты надо задать равным 32 и сбросить накапливающий сумматор в нуль сигналом -СбросНС. После этого надо производить последовательную запись всех 32К кодов по шине записываемых данных Зап.дан., сопровождая их стробами записи -Зап.

После окончания записи в память можно запускать генерацию, но перед началом генерации надо сбросить накапливающий сумматор сигналом -СбросНС, задать режим запуска генерации (разовый или автоматический), а также установить код нужной выходной частоты (см. рис. 8.22 и 8.23). Кроме того, надо задать код амплитуды выходного сигнал (см. рис. 8.18). После этого надо подать положительный сигнал Разр./-Запр. и сопроводить его стробом (см. рис. 8.23). Если надо остановить автоматическую генерацию, то надо установить нулевой сигнал Разр./-Запр. и сопроводить его стробом. Если же генерация разовая, то узнать о том, продолжается она или уже закончилась, можно на основании анализа сигнала Ген. (см. рис. 8.23).

В заключение отметим, что управление разработанным генератором аналоговых сигналов лучше возложить на компьютер или управляющий интеллектуальный контроллер, что существенно упростит работу с ним.

ПРИЛОЖЕНИЕ

МИКРОСХЕМЫ, ПАРАМЕТРЫ, СИГНАЛЫ

Таблица П1. Основные параметры цифровых микросхем

Обозначение	Другое обозначение	Параметр
C_L	$C_Н$	Допустимая емкость нагрузки
F_M	$F_{МАКС}$	Максимальная частота переключения триггера
I_{CC}	$I_П$	Ток потребления
I_I	$I_{ВХ}$	Входной ток
I_O	$I_{ВЫХ}$	Выходной ток
I_{IL}	$I^0_{ВХ}$	Входной ток низкого уровня
I_{IH}	$I^1_{ВХ}$	Входной ток высокого уровня
I_{OL}	$I^0_{ВЫХ}$	Выходной ток низкого уровня
I_{OH}	$I^1_{ВЫХ}$	Выходной ток высокого уровня
t_{LH}	T^{01}	Длительность фронта сигнала при переходе из низкого уровня в высокий
t_{HL}	t^{10}	Длительность фронта сигнала при переходе из высокого уровня в низкий
t_{PLH}	$t^{01}_З$	Время задержки при переходе из низкого уровня в высокий
t_{PHL}	$t^{10}_З$	Время задержки при переходе из высокого уровня в низкий
t_{PZL}	$t^{Z0}_З$	Время задержки при переходе из третьего состояния в низкий уровень
t_{PZL}	$t^{Z1}_З$	Время задержки при переходе из третьего состояния в высокий уровень
U_{CC}	$U_П$	Напряжение питания
U_{IL}	$U^0_{ВХ}$	Входное напряжение низкого уровня
U_{IH}	$U^1_{ВХ}$	Входное напряжение высокого уровня
U_{OL}	$U^0_{ВЫХ}$	Выходное напряжение низкого уровня
U_{OH}	$U^1_{ВЫХ}$	Выходное напряжение высокого уровня

Таблица П2. Функциональное назначение цифровых микросхем стандартных серий

Обозначение	Аналог SN74	Функция
АГ1	121	Одновибратор без перезапуска
АГ3	123	Два одновибратора с перезапуском
АГ4	221	Два одновибратора без перезапуска
АП3	240	Два 4-разрядных буфера с 3С и инверсией
АП4	241	Два 4-разрядных буфера с 3С
АП5	244	Два 4-разрядных буфера с 3С
АП6	245	8-разрядный двунаправленный буфер с 3С
АП9	640	8-разрядный двунаправленный буфер с 3С
АП10	640	8-разрядный двунаправленный буфер с 3С и инверсией
АП14	465	8-разрядный буфер с 3С
АП15	466	8-разрядный буфер с 3С и инверсией
АП16	643	8-разрядный буфер с 3С
АП20	646	8-разрядный двунаправленный буфер с регистром и с 3С
ВА1	226	Схема сопряжения с магистралью
ВЖ1	630	16-разрядная схема контроля по коду Хемминга
ГГ1	124	Два генератора, управляемых напряжением
ИВ1	148	Приоритетный шифратор 8–3
ИВ3	147	Приоритетный шнфратор 9–4
ИД1	14	Двоично-десятичный дешифратор с высоковольтным выходом
ИД3	154	Дешифратор 4–16
ИД4	155	Сдвоенный дешнфратор 2–4
ИД5	156	Два дешнфратора 2–4 с ОК
ИД6	42	Двоично-десятичный дешифратор 3–8
ИД7	138	Дешифратор 3–8
ИД10	145	Двоично-десятичный дешифратор 3–8 с большнм выходным током
ИД14	139	Два дешифратора 2–4
ИД18	247	Дешифратор двоично-десятичного кода в код семисегментного индикатора
ИЕ2	90	4-разрядный двоично-десятичный счетчик
ИЕ4	92	Счетчик-делитель на 12
ИЕ5	93	4-разрядный двоичный счетчик
ИЕ6	192	4-разрядный реверсивный двоично-десятичный счетчик

Таблица П2. Продолжение

Обозначение	Аналог SN74	Функция
ИЕ7	193	4-разрядный реверсивный двоичный счетчик
ИЕ8	97	Делитель частоты с переменным коэффициентом деления
ИЕ9	160	4-разрядный синхронный двоично-десятичный счетчик с асинхронным сбросом
ИЕ10	161	4-разрядный синхронный двончный счетчик с асинхронным сбросом
ИЕ11	162	4-разрядный двоично-десятичный счетчнк с синхронным сбросом
ИЕ12	190	4-разрядный синхронный реверсивный десятичный счетчик
ИЕ13	191	4-разрядный синхронный реверсивный двоичный счетчик
ИЕ14	196	Счетчик-делитель на 2 и на 5
ИЕ15	197	4-разрядный асинхронный счетчик с предварительной установкой
ИЕ16	168	4-разрядный синхронный двоично-десятичный счетчик с параллельной загрузкой
ИЕ17	169	4-разрядный синхронный двоичный счетчик с параллельной загрузкой
ИЕ18	163	4-разрядный двоичный счетчик с синхронным сбросом
ИЕ19	393	Сдвоенный 4-разрядный двоичный счетчик
ИЕ20	390	Два двоично-десятичных счетчика со сбросом
ИМ1	80	1-разрядный полный сумматор
ИМ2	82	2-разрядный полный сумматор
ИМ3	83	4-разрядный полный сумматор
ИМ5	183	4-разрядный полный сумматор с ускоренным переносом
ИМ6	283	4-разрядный полный сумматор с ускоренным переносом
ИМ7	385	4-разрядный сумматор-вычитатель
ИП2	180	8-разрядная схема контроля четности
ИП3	181	АЛУ для двух 4-разрядных слов
ИП4	182	4-разрядная схема ускоренного переноса
ИП5	280	9-разрядная схема контроля четности
ИП6	242	Двунаправленный 4-разрядный буфер с инверсией
ИП7	243	Двунаправленный 4-разрядный буфер
ИП8	261	Параллельный умножитель 2 × 4 разряда
ИП9	384	8-разрядный последовательно-параллельный умножитель
ИР1	95	4-разрядный двунаправленный сдвиговый регистр

Таблица П2. Продолжение

Обозначение	Аналог SN74	Функция
ИР8	164	8-разрядный сдвиговый регистр с последовательным входом и параллельными выходамн
ИР9	165	8-разрядный сдвиговый регистр с параллельными входами и последовательным выходом
ИР10	166	8-разрядный сдвиговый регистр
ИР11	194	4-разрядный 2-направленный сдвнговый регистр
ИР12	195	4-разрядный 2-направленный сдвиговый регистр
ИР13	198	8-разрядный сдвиговый регистр
ИР15	173	4-разрядный регистр с 3С
ИР16	295	4-разрядный реверсивный сдвиговый регистр с выходами 3С
ИР22	373	8-разрядный регистр-защелка с 3С
ИР23	374	8-разрядный регистр с 3С
ИР24	299	8-разрядный двунаправленный реверсивный сдвиговый регистр с 3С
ИР25	395	4-разрядный сдвиговый регистр с 3С
ИР26	б70	Регнстровый файл 4 × 4 с 3С
ИР27	377	8-разрядный регистр с разрешением записи
ИР29	323	8-разрядный сдвиговый регистр с 3С
ИР30	259	8-разрядный регистр хранения с адресацией
ИР32	170	Регистровый файл 4 × 4 с ОК
ИР33	573	8-разрядный буферный регистр
ИР34	873	Два 4-разрядных регистра с 3С
ИР35	273	8-разрядный регистр со сбросом
ИР37	574	8-разрядный регистр с 3С
ИР38	874	Два 4-разрядных регистра с 3С
ИР40	533	8-разрядный регистр-защелка с 3С и ннверсией
ИР41	534	8-разрядный регистр с 3С и инверсией
КП1	150	16-канальный мультиплексор
КП2	153	Сдвоенный 4-канальный мультиплексор
КП5	152	8-канальный мультнплексор
КП7	151	8-канальный мультиплексор со стробированием
КП11	257	4-разрядный 2-канальный мультиплексор с 3С
КП12	253	2-разрядный 4-канальный мультиплексор

Таблица П2. Продолжение

Обозначение	Аналог SN74	Функция
КП13	298	4-разрядный 2-канальный мультиплексор со стробированием
КП14	258	4-разрядный 2-канальный мультиплексор с 3С с инверсией
КП15	251	8-канальный мультиплексор с 3С
КП16	157	4-разрядный 2-канальный мультиплексор
КП17	353	2-разрядный 4-канальный мультиплексор с 3С и инверсией
КП18	158	4-разрядный 2-канальный мультиплексор с инверсией
КП19	352	2-разрядный 4-канальный мультиплексор с инверсией
ЛА1	20	Два логических элемента 4И-НЕ
ЛА2	30	Логический элемент 8И-НЕ
ЛА3	00	Четыре логических элемента 2И-НЕ
ЛА4	10	Три логических элемента 3И-НЕ
ЛА6	40	Два логических элемента 4И-НЕ с повышенным выходным током
ЛА7	22	Два логических элемента 4И-НЕ с ОК и повышенным выходным током
ЛА8	01	Четыре логических элемента 2И-НЕ с ОК
ЛА9	03	Четыре логических элемента 2И-НЕ с ОК
ЛА10	12	Три логических элемента 3И-НЕ с ОК
ЛА11	26	Четыре логических элемента 2И-НЕ с ОК и повышенным выходным напряжением
ЛА12	37	Четыре логических элемента 2И-НЕ с повышенным выходным током
ЛА13	38	Четыре логических элемента 2И-НЕ с ОК и повышенным выходным током
ЛА16	140	Два логических элемента 4И-НЕ для работы на линию 50 Ом
ЛА19	134	Логическнй элемент 12И-НЕ с разрешением
ЛА21	1000	Четыре логических элемента 2И-НЕ с повышенным выходным током
ЛА22	1020	Два логических элемента 4И-НЕ с повышенным выходным током
ЛА23	1003	Четыре логических элемента 2И-НЕ с ОК и повышенным выходным током
ЛА24	1010	Три логических элемента 3И-НЕ с повышенным выходным током

Таблица П2. Продолжение

Обозначение	Аналог SN74	Функция
ЛД1	60	Два 4-входовых расширителя по ИЛИ
ЛЕ1	02	Четыре логических элемента 2ИЛИ-НЕ
ЛЕ2	23	Два логических элемента 4ИЛИ-НЕ со стробнрованием
ЛЕ3	25	Два логических элемента 4ИЛИ-НЕ со стробированием
ЛЕ4	27	Три логических элемента 3ИЛИ-НЕ
ЛЕ5	28	Четыре логических элемента 2ИЛИ-НЕ с повышенным выходным током
ЛЕ6	128	Четыре логических элемента 2ИЛИ-НЕ с повышенным выходным током
ЛЕ7	260	Два логических элемента 5ИЛИ-НЕ
ЛИ1	08	Четыре логических элемента 2И
ЛИ2	09	Четыре логических элемента 2И с ОК
ЛИ3	11	Три логических элемента 3И
ЛИ4	15	Три логических элемента 3И с ОК
ЛИ6	21	Два логических элемента 4И
ЛЛ1	32	Четыре логических элемента 2ИЛИ
ЛЛ3	136	Четыре двухвходовых логических элемента Исключающее ИЛИ с ОК
ЛН1	04	Шесть инверторов
ЛН2	05	Шесть инверторов с ОК
ЛН3	06	Шесть инверторов с ОК н повышенным выходным напряжением
ЛН4	07	Шесть буферных элементов с ОК
ЛН5	16	Шесть инверторов с ОК и повышенным выходным напряжением
ЛН6	366	Шесть инверторов с 3С и с управлением
ЛП4	17	Шесть буферных элементов с ОК и повышенным выходным напряжением
ЛП5	86	Четыре двухвходовых логических элемента Исключающее ИЛИ
ЛП8	125	Четыре буферных элемента с 3С и раздельным управлением
ЛП9	07	Шесть буферных элементов с ОК и повышенным выходным напряжением
ЛП10	365	Шесть буферных элементов с 3С
ЛП11	367	Шесть буферных элементов с 3С
ЛП12	136	Четыре двухвходовых логических элемента Исключающее ИЛИ с ОК

Таблица П2. Продолжение

Обозначение	Аналог SN74	Функция
ЛП16	1034	Шесть буферов с повышенным выходным током
ЛП17	1035	Шесть буферов с ОК и повышенным выходным током
ЛР1	50	Два элемента 2-2И-2ИЛИ-НЕ
ЛР3	53	Элемент 2-2-2-3И-4ИЛИ-НЕ
ЛР4	55	Элемент 4-4И-2ИЛИ-НЕ
ЛР9	64	Элемент 2-4-2-3И-ИЛИ-НЕ
ЛР10	65	Элемент 2-4-2-3И-ИЛИ-НЕ с ОК
ЛР11	51	Элементы 2-2И-2ИЛИ-НЕ и 2-3И-2ИЛИ-НЕ
ЛР13	54	Элемент 3-2-2-3И-4ИЛИ-НЕ
ПР6	184	Преобразователь двоично-десятичного кода в двоичный
ПР7	185	Преобразователь двоичного кода в двоично-десятичный
РП1	170	Регистровое ЗУ 4 × 4
РП3	172	Регистровое ЗУ 8 × 2 с ОК
РУ1	81	ОЗУ с организацией 4 × 4
РУ2	89	ОЗУ с организацией 16 × 4
РУ3	84	ОЗУ 4 × 4 с дополнительными входами записи
РУ5	130	ОЗУ с организацией 256 × 1
РУ9	289	ОЗУ с организацией 16 × 4
РУ10	225	ОЗУ с организацией 16 × 4
СП1	85	4-разрядный компаратор кодов
ТВ1	72	JK-триггер с элементом 3И на входе
ТВ6	107	Два JK-триггера
ТВ9	112	Два JK-триггера
ТВ10	113	Два JK-триггера
ТВ11	114	Два JK-триггера
ТВ15	109	Два JK-триггера
ТЛ1	13	Два триггера Шмитта с инверсией и элементом 4И на входе
ТЛ2	14	Шесть триггеров Шмитта с ннверсией
ТЛ3	132	Четыре триггера Шмитта с инверсией и элементом 2И на входе
ТМ2	74	Два D-триггера с прямыми и инверсными выходами
ТМ5	77	Четыре D-триггера типа «защелка»
ТМ7	75	Четыре D-триггера типа «защелка» с прямыми н инверсными выходами
ТМ8	175	Четыре D-триггера с прямыми н инверсными выходами
ТМ9	174	Шесть D-триггеров с общим синхровходом
ТР2	279	Два DS-триггера

Таблица П3. Типы микросхем семейства SN74

Номер SN74	Обозначение	Номер SN74	Обозначение	Номер SN74	Обозначение
00	ЛА3	51	ЛР11	130	РУ5
01	ЛА8	53	ЛР3	132	ТЛ3
02	ЛЕ1	54	ЛР13	134	ЛА19
03	ЛА9	55	ЛР4	136	ЛЛ3
04	ЛН1	60	ЛД1	136	ЛП12
05	ЛН2	64	ЛР9	138	ИД7
06	ЛН3	65	ЛР10	139	ИД14
07	ЛН4	72	ТВ1	140	ЛА16
07	ЛП9	74	ТМ2	141	ИД1
08	ЛИ1	75	ТМ7	145	ИД10
09	ЛИ2	77	ТМ5	147	ИВ3
10	ЛА4	80	ИМ1	148	ИВ1
11	ЛИ3	81	РУ1	150	КП1
12	ЛА10	82	ИМ2	151	КП7
13	ТЛ1	83	ИМ3	152	КП5
14	ТЛ4	84	РУ3	153	КП2
15	ЛИ4	85	СП1	154	ИД3
16	ЛН5	86	ЛП5	155	ИД4
17	ЛП4	89	РУ2	156	ИД5
20	ЛА1	90	ИЕ2	157	КП16
21	ЛИ6	92	ИЕ4	158	КП18
22	ЛА7	93	ИЕ5	160	ИЕ9
23	ЛЕ2	95	ИР1	161	ИЕ10
25	ЛЕ3	97	ИЕ8	162	ИЕ11
26	ЛА11	107	ТВ6	163	ИЕ18
27	ЛЕ4	109	ТВ15	164	ИР8
28	ЛЕ5	112	ТВ9	165	ИР9
30	ЛА2	113	ТВ10	166	ИР10
32	ЛЛ1	114	ТВ11	168	ИЕ16
37	ЛА12	121	АГ1	169	ИЕ17
38	ЛА13	123	АГ3	170	ИР32
40	ЛА6	124	ГГ1	170	РП1
42	ИД6	125	ЛП8	172	РП3
50	ЛР1	128	ЛЕ6	173	ИР15

Таблица П3. Продолжение

Номер SN74	Обозначение	Номер SN74	Обозначение	Номер SN74	Обозначение
174	ТМ9	245	АП6	384	ИП9
175	ТМ8	247	ИД18	385	ИМ7
180	ИП2	251	КП15	390	ИЕ20
181	ИП3	253	КП12	393	ИЕ19
182	ИП4	257	КП11	395	ИР25
183	ИМ5	258	КП14	465	АП14
184	ПР6	259	ИР30	466	АП15
185	ПР7	260	ЛЕ7	533	ИР40
190	ИЕ12	261	ИП8	534	ИР41
191	ИЕ13	273	ИР35	573	ИР33
192	ИЕ6	279	ТР2	574	ИР37
193	ИЕ7	280	ИП5	630	ВЖ1
194	ИР11	283	ИМ6	639	АП9
195	ИР12	289	РУ9	640	АП10
196	ИЕ14	295	ИР16	643	АП16
197	ИЕ15	298	КП13	646	АП20
198	ИР13	299	ИР24	670	ИР26
221	АГ4	323	ИР29	873	ИР34
225	РУ10	352	КП19	874	ИР38
226	ВА1	353	КП17	1000	ЛА21
240	АП3	365	ЛП10	1003	ЛА23
241	АП4	366	ЛН6	1010	ЛА24
242	ИП6	367	ЛП11	1020	ЛА22
243	ИП7	373	ИР22	1034	ЛП16
244	АП5	374	ИР23	1035	ЛП17
		377	ИР27		

Таблица П4. Соответствие зарубежных и отечественных серий цифровых микросхем

Серии SN74	Отечественные серии
74 ... J	КМ155
74 ... N	К155
74AC ... N	КР1554
74ALS ... N	КР1533
74F ... N	КР1531
74H ... N	К131
74L ... N	К134
74 LS ... J	КМ555
74LS ... N	К555
74S ... J	КМ531
74S ... N	КР531

Таблица П5. Обозначения сигналов и микросхем

Обозначение	Название	Назначение
&	And	Элемент И
=1	Exclusive Or	Элемент Исключающее ИЛИ
+1	—	Вход счета на увеличенне
–1	—	Вход счета на уменьшение
<—>	—	Двунаправленная передача
<—>	—	Двунаправленный сдвиг
<	—	Вход расширения сумматора «меньше»
< 0	—	Перенос (заем) счетчика при инверсном счете (на уменьшение)
>	—	Вход расширения сумматора «больше»
> 9	—	Перенос 4-разрядного двоично-десятичного счетчика при прямом счете
> 15	—	Перенос 4-разрядного двоичного счетчика при прямом счете
=	—	Вход расшнрения сумматора «равно»
0, 1, 2, 3, ...	—	Номера входных нли выходных разрядов кода
0V	—	Общий вывод

Таблица П5. Продолжение

Обозначение	Название	Назначение
1	Or	Элемент ИЛИ
1, 2, 4, 8 ...	—	Входы/выходы разрядов кода
A	Address	Адресные разряды
A0, A1,...	—	Разряды входного/выходного кода A
A=B	Parity	Вход или выход равенства кодов A и B
A>B, A<B	—	Входы или выходы сравнения кодов A и B
A, B, C,...	—	Входы и выходы различного назначения
ADC	Analog-to-Digital Converter	Аналого-цифровой преобразователь, АЦП
ALU	Arithmetic Logic Unit	Арифметическо-логическое устройство
B0, B1,...	.—	Разряды входного/выходного кода B
BF	Buffer	Буфер
BR	Borro	Заем
C	Clock	Тактовый сигнал (строб), сигнал разрешения
C	Carr	Вход переноса
C	Capacitor	Подключение внешнего конденсатора
CD	Coder	Шифратор
CE	Clock Enable	Разрешение тактового сигнала
CE	Chip Enable	Разрешение работы микросхемы
CEP	Count Enable Parallel	Вход параллельного наращивания разрядности счетчиков
CET	Count Enable Trickle	Вход наращивания разрядности счетчиков («трюковый»)
CLK	Clock	Тактовый вход
CLR	Clear	Очистка, сброс
CPU	Central Processor Unit	Центральный процессор
CR	Carr	Перенос
CRU	Carry lock ahead Unit	Схема ускоренного переноса
CT	Counter	Счетчик
CT10	2/10 Counter	Двоично-десятичный счетчик
CT2	Binary Counter	Двоичный счетчик
CT2/10	2/10 Counter	Двоично-десятичный счетчик

Таблица П5. Продолжение

Обозначение	Название	Назначение
CS	Chip Select	Выбор микросхемы
D	Data	Разряды данных, данные
DAC	Digital-to-Analog Converter	Цифро-аналоговый преобразователь, ЦАП
DC	Decoder	Дешифратор
DI	Data Input	Входные данные
DIO	Data Input/Output	Входные/выходные данные
DL	Data Left	Вход данных для сдвига влево
DO	Data Output	Выходные данные
DP	Data Parallel	Параллельные данные
DR	Data Right	Вход данных для сдвига вправо
DS	Data Serial	Последовательные данные
D/U	Down/Up	Переключение направления счета счетчиков
E	Enable	Разрешение
EC	Enable Count	Разрешение счета
ECR	Enable Carr	Разрешение переноса
ECT	Enable Count	Разрешение счета
EI	Enable Input	Разрешение входа
EIO	Enable Input/Output	Разрешение входа и выхода
EO	Enable Output	Разрешение выхода
EP	Enable P	Разрешение переноса
EWR	Enable Write	Разрешение записи
EZ	Enable Z-state	Разрешение третьего состояния
G	Generator	Генератор
G1	Generator	Одновибратор
I	Input	Вход
I/O	Input/Output	Вход/Выход
J	—	Вход записи нуля в JK-триггере
K	—	Вход записи единицы в JK-триггере
L	Load	Загрузка, запись
LOAD	Load	Загрузка, запись
LSB	Least Significant Bit	Младший значащий разряд

Таблица П5. Продолжение

Обозначение	Название	Назначение
M2	—	Схема контроля четности
MS	Multiplexer	Мультиплексор
MSB	Most Significant Bit	Старший значащий разряд
MUX	Multiplexer	Мультиплексор
O	Output	Выход
OE	Output Enable	Разрешение выхода
P	—	Выход переноса
PE	Parallel Enable	Разрешение параллельной загрузки
PROM	Programmable ROM	Программируемая постоянная память
P/S	Parallel/Serial	Переключение параллельный/последовательный режим
Q	Quit	Выход
R	Reset	Сброс (установка в нуль)
R	Resistor	Подключение внешнего резистора
RAM	Random Access Memory	Оперативная память, ОЗУ
RC	Resistor/Capacitor	Подключение внешнего резистора и конденсатора
RD	Read	Чтение
RE	Read Enable	Разрешение чтения
RG	Register	Регистр
ROM	Read Only Memory	Постоянная память, ПЗУ
R/W	Read/Write	Чтение/Запись
S	Set	Установка в единицу
S	Strobe	Стробирующий сигнал
S0, S1,...		Входы установки режима
S0, S1,...	Sum	Разряды выходного кода суммы
SE	Set Enable	Разрешение установки
SEMO	Set Mode	Установка режима
SL	Shift Left	Сдвиг влево
SM	Summater	Сумматор
SR	Synchronous Reset	Синхронный сброс

Таблица П5. Продолжение

Обозначение	Название	Назначение
SR	Shift Right	Сдвиг вправо
SUM	Summater	Сумматор
SYN	Synchro	Синхросигнал
T	Trigger	Триггер
TC	Terminal Count	Окончание счета
U/D	Up/Down	Переключение направления счета счетчиков
U_{CC}	—	Напряжение питания
V	—	Входы управления работой
WE	Write Enable	Разрешение записи
WR	Write	Запись
X/Y	—	Преобразователь кодов
Z	Z-state	Третье состояние выхода

СПИСОК ЛИТЕРАТУРЫ

1. Титце У., Шенк К. Полупроводниковая схемотехника: Справочное руководство. Пер. с нем. — М.: Мир, 1982. — 512 с.
2. Гнатек Ю. Р. Справочник по цифро-аналоговым и аналого-цифровым преобразователям: Пер. с англ. / Под ред. Ю. А. Рюжина. — М.: Радио и связь, 1982. — 420 с.: ил.
3. Горошков Б. И. Радиоэлектронные устройства: Справочник. — М.: Радио и связь, 1984. — 400 с.: ил.
4. Применение интегральных микросхем в электронной вычислительной технике: Справочник / Р. В. Данилов, С. А. Ельцова, Ю. П. Иванов и др.; Под ред. Б. Н. Файзулаева, Б. В. Тарабрина. — М.: Радио и связь, 1986. — 384 с.: ил.
5. Шило В. Л. Популярные цифровые микросхемы: Справочник. — М.: Радио и связь, 1987. — 352 с.
6. Полупроводниковые БИС запоминающих устройств: Справочник / Под ред. А. Ю. Гордонова и Ю. Н. Дьякова. — М.: Радио и связь, 1987. — 360 с.: ил.
7. Бородин С. М., Новиков Ю. В. Модуль логического анализатора для контрольно-измерительных систем на базе микроЭВМ // Микропроцессорные средства и системы. — 1987. — № 1. — с. 67–68.
8. Уильямс Г. Б. Отладка микропроцессорных систем: Пер. с англ. — М.: Энергоатомиздат, 1988 — 253 с.: ил.
9. Бородин С. М., Новиков Ю. В., Поддубный А. П., Томчук А. А. Средства отображения информации для микропроцессорных систем измерения, контроля и управления // Микропроцессорные средства и системы. — 1988. — № 3. — с. 76–79.
10. Интерфейсы систем обработки данных: Справочник / А. А. Мячев, В. Н. Степанов, В. К. Щербо; Под ред. А. А. Мячева. — М.: Радио и связь, 1989 — 416 с.: ил.
11. Овчинников В. В., Рыбкин И. И. Техническая база интерфейсов локальных вычислительных сетей. — М.: Радио и связь, 1989 — 272 с.: ил.
12. Новиков Ю. В. Универсальный параллельный интерфейс для модульных микропроцессорных систем измерения, контроля и управления // Микропроцессорные средства и системы. — 1989. — № 6. — с. 71–72.

13. Шевкопляс Б. В. Микропроцессорные структуры. Инженерные решения: Справочник. — 2-е изд. перераб. и доп. — М.: Радио и связь, 1990 — 512 с.: ил.
14. Большие интегральные схемы запоминающих устройств: Справочник / Под ред. А. Ю. Гордонова и Ю. Н. Дьякова. — М.: Радио и связь, 1990. — 288 с.: ил.
15. Воробьев Е. П., Сенин К. В. Интегральные микросхемы производства СССР и их зарубежные аналоги: Справочник. — М.: Радио и связь, 1990. — 352 с.: ил.
16. Чернега В. С., Василенко В. А., Бондарев В. Н. Расчет и проектирование технических средств обмена и передачи информации. — М.: Высшая школа, 1990. — 224 с.: ил.
17. Федорков Б. Г., Телец В. А. Микросхемы ЦАП и АЦП: функционирование, параметры, применение. — М.: Энергоатомиздат, 1990. — 320 с.: ил.
18. Новиков Ю. В. Функциональные модули контрольно-измерительных систем на базе микроЭВМ // Микропроцессорные средства и системы, 1990. — № 3. — с. 75–77.
19. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC: Пер. с англ. / Под ред. У. Томпкинса, Дж. Уэбстера. — М.: Мир, 1992 — 592 с.: ил.
20. Юшин А. М. Цифровые микросхемы для электронных устройств: Справочник. — М.: Высшая школа, 1993. — 176 с.: ил.
21. Логические ИС КР1533, КР1554: Справочник / И. И. Петровский, А. В. Прибыльский, А. А. Троян, В. С. Чувелев: В 2-х ч. — М.: БИНОМ, 1996.
22. The TTL Data Book. — Texas Instruments, 1997.
23. Новиков Ю. В., Калашников О. А., Гуляев С. Э. Разработка устройств сопряжения для персональных компьютеров типа IBM PC / Под общ. ред. Ю. В. Новикова. Практ. пособие. — М.: ЭКОМ, 1997 — 224 с.: ил.
24. Хоровиц П., Хилл У. Искусство схемотехники. Пер. с англ. 5-е изд. перераб. — М.: Мир, 1998. — 704 с.: ил.
25. Перельман Б. Л., Шевелев В. И. Отечественные микросхемы и зарубежные аналоги: Справочник. — М.: НТЦ Микротех, 1998. — 376 с.: ил.
26. Новиков Ю. В., Карпенко Д. Г. Аппаратура локальных сетей: функции, выбор, разработка / Под общ. ред. Ю. В. Новикова. — М.: ЭКОМ, 1998 — 288 с.: ил.

27. Мюллер С. Модернизация и ремонт персональных компьютеров / Пер. с англ. — М.: БИНОМ, 1998. — 944 с.: ил.
28. Гук М. Аппаратные средства IBM PC. Энциклопедия. — СПб: Питер Ком, 1999. — 816 с.: ил.
29. Бирюков С. А. Применение цифровых микросхем серий ТТЛ и КМОП. — М.: ДМК, 1999. — 240 с.: ил.
30. Новиков Ю. В., Кондратенко С. В. Локальные сети: архитектура, алгоритмы, проектирование. — М.: ЭКОМ, 2000 — 312 с.: ил.

СЛОВАРЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

2S (2-States Output) — выход с двумя активными состояниями, 2С, стандартный выход ТТЛ.
3S (3-States Output) — выход с тремя состояниями, 3С.
Adder — сумматор.
AND — логическая функция И.
AC (Alternating Current) — переменный ток.
ALU (Arithmetic and Logic Unit) — АЛУ, арифметико-логическое устройство.
ADC (Analog-to-Digital Converter) — АЦП, аналого-цифровой преобразователь.
ASCII (American Standard Code for Information Interchange) — стандартный американский код обмена символьной информацией.
BCD (Binary-Coded Decimal) — двоично-десятичный код.
BiCMOS (Bipolar Complementary Metal-Oxide-Semiconductor) — биполярно-полевая технология изготовления микросхем, сочетающая на одном кристалле биполярные и КМОП структуры.
Buffer — буфер.
Bus — шина, магистраль.
CAS (Column-Address Select) — сигнал выбора адреса столбца (в микросхемах динамической памяти).
Chip — микросхема, чип.
Clear — очистка, сброс в нуль.
Clock — тактовый, тактирующий сигнал.
CMOS (Complementary Metal-Oxide-Semiconductor) — комплементарная МОП технология (КМОП).
Coder (encoder) — шифратор, кодер.
Comparator — компаратор.
Converter — преобразователь.
CRC (Cyclic Redundancy Check) — циклический избыточный контроль, метод вычисления циклической контрольной суммы и сама эта сумма.
Counter — счетчик.

DAC (Digital-to-Analog Converter) — ЦАП, цифро-аналоговый преобразователь.
DC (Direct Current) — постоянный ток.
Decoder — дешифратор, декодер.
Delay — задержка.
DIC (Dual In-line Ceramic package) — керамический корпус микросхемы типа DIL.
DIL (Dual In-Line package) — корпус микросхемы с двухрядным вертикальным расположением выводов.
DIMM (Dual In-Line Memory Module) — модуль памяти с двусторонним расположением выводов.
DIP (Dual In-line Plastic package) — пластмассовый корпус микросхемы типа DIL.
DIP Switches — малогабаритные выключатели, смонтированные в корпусе типа DIP.
DRAM (Dynamic RAM) — динамическое ОЗУ.
Driver — выходной буфер, драйвер.
EEPROM (Electrically Erasable Programmable ROM) — ПЗУ с электрическим стиранием и возможностью программирования.
EPROM (Erasable Programmable ROM) — ПЗУ со стиранием (ультрафиолетовым излучением) и перезаписью информации (РПЗУ).
Female — разъем-розетка, гнездо.
FIFO (First In, First Out) — «первым вошел — первым вышел», один из способов организации ОЗУ с последовательным доступом.
Flash memory — разновидность памяти EEPROM, характеризующаяся высокой емкостью, малым потреблением и большим допустимым количеством циклов перезаписи, флэш-память.
Flip-flop — триггер.
Gate — логический элемент, вентиль.
GND (Ground) — общий провод схемы, «земля».
H (High) — высокий уровень сигнала, единичный уровень при положительной логике.
IC (Integrated Circuit) — интегральная микросхема, ИС.
Inverter — инвертор.
I/O (Input/Output) — ввод/вывод (В/В), вход/выход.

Jumper — съемная перемычка, соединяющая штыревые контакты на плате, джампер.
L (Low) — низкий уровень сигнала, нулевой уровень при положительной логике.
Latch — триггер (регистр) типа «защелка».
LCD (Liquid Crystal Display) — жидкокристаллический дисплей, индикатор.
Line driver — драйвер линии, буфер.
LSB (Least Significant Bit) — младший значащий бит (в байте или слове).
LSI (Large Scale Integration) — большая интегральная схема, БИС.
LVT (Low-Voltage Technology) — низковольтная технология микросхем (напряжение питания 3,3 В).
Male — разъем-вилка, штекер.
Master — ведущее, главное устройство, участвующее в обмене информацией, задатчик.
Monostable multivibrator — ждущий мультивибратор, одновибратор.
MSB (Most Significant Bit) — старший значащий бит (в байте или слове).
Multiplexer — мультиплексор.
Multivibrator — мультивибратор.
NAND — логическая функция И-НЕ.
Noninverter — повторитель.
NOR — логическая функция ИЛИ-НЕ.
NVRAM (Non-Volatile RAM) — энергонезависимое ОЗУ, сохраняющее информацию при отключении питания.
OC (Open-Collector Output) — выход микросхемы с открытым коллектором.
OR — логическая функция ИЛИ.
PAL (Programmable Array Logic) — программируемая логическая матрица, ПЛМ.
Parity — четность, паритет.
Plug — разъем типа вилка.
Preset — предварительная установка.
PROM (Programmable ROM) — программируемое ПЗУ, ППЗУ.
Pull-up Resistor — нагрузочный резистор, включаемый между выходом микросхемы и шиной питания.

RAM (Random Access Memory) — оперативное запоминающее устройство, ОЗУ.
RAS (Row-Address Select) — сигнал выбора адреса строки (в микросхемах динамической памяти).
Receiver — приемник, входной буфер.
Refresh — регенерация (в динамической памяти).
Register — регистр.
Reset — сигнал установки микросхемы или устройства в нулевое или исходное состояние.
ROM (Read-Only Memory) — постоянное запоминающее устройство, ПЗУ.
RxC (Received Clock) — принимаемый синхросигнал.
RxD (Received Data) — принимаемые данные.
Schmitt trigger — триггер Шмитта.
SDRAM (Synchronous Dynamic RAM) — синхронное динамическое ОЗУ.
Set — сигнал установки выхода в единичное состояние.
SIMM (Single In-Line Memory Module) — модуль памяти с однорядным расположением выводов.
SIP (Single In-line Package) — корпус микросхемы с однорядным расположением выводов.
Slave — ведомое, пассивное устройство, участвующее в обмене информацией, исполнитель.
Slot — щелевой разъем для подключения печатных плат с разъемом в виде печатных проводников, слот.
Socket — контактирующее устройство для установки микросхем на плату, сокет.
SRAM (Static RAM) — статическое ОЗУ.
Strobe — стробирующий сигнал, строб.
Terminator — оконечное согласующее устройство на линии связи (обычно — резистор).
TI, TII (Texas Instruments Inc.) — американская фирма, один из ведущих производителей цифровых микросхем малой и средней степени интеграции.
TR (Terminate Resistor) — нагрузочный резистор для линии связи.
Transceiver — приемопередатчик, трансивер, двунаправленный буфер.
Transmitter — передатчик, выходной буфер.
Trigger — триггер.

TTL (Transistor-Transistor Logic) — транзисторно-транзисторная (биполярная) логика, ТТЛ.
TTLS (Transistor-Transistor Logic Schottky) — транзисторно-транзисторная логика Шоттки, ТТЛШ.
TxC (Transmitted Clock) — передаваемый синхросигнал.
TxD (Transmitted Data) — передаваемые данные.
V — напряжение (Voltage), вольт (Volt).
VLSI (Very Large Scale Integration) — сверхбольшая интегральная схема (СБИС).
Waveform — форма сигнала.
Z (Z-state) — третье (высокоимпедансное) состояние выхода микросхемы.
ZIF (Zero Insertion Force) — разъем или сокет с нулевым усилием вставки.

2С — выход с двумя активными состояниями (нуль и единица), стандартный ТТЛ-совместимый выход.
3С — выход с тремя состояниями (два активных: нуль и единица, третье — пассивное, отключенное), а также само третье состояние выхода в отличие от двух активных состояний.

Адрес — закодированный номер, определяющий, куда передается информация или откуда она принимается.
Активный уровень сигнала — уровень, соответствующий приходу, наличию сигнала, то есть выполнению этим сигналом соответствующей ему функции.
АЛУ — арифметико-логическое устройство (ALU).
АПЧ — автоматическая подстройка частоты.
Асинхронный сигнал — сигнал, не привязанный по времени к внутренним процессам схемы, не синхронизованный со схемой.
Асинхронный (последовательный) счетчик — счетчик, выходные разряды которого переключаются по очереди, начиная с младшего.
АЦП — аналого-цифровой преобразователь (ADC), преобразующий величину входного аналогового сигнала в выходной цифровой код.
Байт — группа двоичных разрядов, битов (как правило, 8 бит), содержащая какой-то код.

Биполярный сигнал — сигнал, который может быть как положительным, так и отрицательным.

Бит (от англ. Binary Digit — двоичное число) — единица двоичной информации, разряд двоичного кода, принимающий значения 0 и 1.

БИС — большая интегральная схема (LSI).

Ввод данных — то же, что чтение, считывание, прием.

Внлка (штекер) — часть разъема, контакты которого входят в контакты розетки (гнезда).

Вывод данных — то же, что запись, передача.

Выводы микросхемы — металлические контакты на корпусе микросхемы для входных и выходных сигналов, подключения внешних элементов и подачи питания.

Временная диаграмма — графики зависимости от времени входных и выходных сигналов цифрового устройства в различных режимах работы.

Выборка — мгновенное значение аналогового сигнала, которому ставится в соответствие цифровой код.

Гига- (Г) — приставка для обозначения миллиарда, 10^9.

Данные — передаваемая в закодированном виде цифровая информация.

Двоичная система счисления — система счисления по модулю два, в которой разряды числа кодируются 0 или 1 и представляют собой степени числа 2.

Двунаправленная линия (шина) — линия (шина), по которой сигналы могут передаваться в обоих направлениях (по очереди).

Дешифрация — преобразование входного двоичного кода в номер выходного сигнала.

Дорожки — проводники на поверхности печатной платы, для передачи сигналов и подачи питания.

Единичный сигнал — то же, что положительный сигнал.

Задержка — временной сдвиг между входным и выходным сигналами устройства, узла, микросхемы.

Задний фронт сигнала (спад) — переход сигнала из активного уровня в пассивный.

Защелка — триггер или регистр, стробируемый уровнем сигнала и пропускающий входной сигнал на выход при активном управляющем сигнале (стробе).

ЗУ — запоминающее устройство, память.

Импульс — сравнительно короткий сигнал.

Инверсный выход — выход, выдающий сигнал инверсной полярности по сравнению со входным сигналом.

Инвертирование или инверсия сигнала — изменение полярности сигнала.

Интерфейс — соглашение об обмене между электронными устройствами. Включает в себя требования по электрическому, логическому и конструктивному сопряжению устройств.

ИС — интегральная микросхема, ИМС (IC), чип.

Кабель — один или несколько проводов в общей оболочке, используемые для передачи сигналов.

Карта — электронное устройство, выполненное на печатной плате.

КЗ — короткое замыкание.

Кило- (к) — приставка для обозначения тысячи, 10^3.

КМОП — комплементарная технология МОП (CMOS).

Код — информация, передаваемая несколькими двоичными разрядами, битами.

Контактные площадки — проводники на поверхности печатной платы, к которым припаиваются выводы микросхем.

Коэффициент разветвления — число входов, которое может быть подключено к данному выходу без нарушения его работы. Определяется отношением выходного тока к входному. Стандартная величина коэффициента разветвления при использовании микросхем одной серии равна 10.

Линия (линия связи) — проводник (электрический или оптоволоконный), передающий сигнал.

Меандр — сигнал со скважностью, равной двум, то есть длительность импульсов равна длительности паузы между ними.

Мега- (М) — приставка для обозначения миллиона, 10^6.

Микро- (мк) — приставка для обозначения одной миллионной доли, 10^{-6}.

Милли- (м) — приставка для обозначения одной тысячной доли, 10^{-3}.

МОП — полупроводниковая технология на основе полевых транзисторов типа «металл — окисел — полупроводник» (MOS).

Мультиплексирование — передача различных сигналов по одной линии (шине) в разные моменты времени.

Нагрузочная способность — параметр выхода микросхемы, характеризующий величину выходного тока, которую может выдать в нагрузку данный выход без нарушения его работы. Чаще всего нагрузочная способность прямо связана с коэффициентом разветвления.

Нано- (н) — приставка для обозначения одной миллиардной доли, 10^{-9}.

Нулевой сигнал — то же, что отрицательный сигнал.

Обратная связь — передача сигнала или его части с выхода схемы на ее вход или один из ее входов.

Обратный (инверсный) счет — счет на уменьшение выходного кода.

ОЗУ — оперативное запоминающее устройство, оперативная память (RAM).

ОК — выход с открытым коллектором.

Опорное напряжение — напряжение эталонного уровня, с которым сравнивается входной сигнал (в АЦП), или из которого формируется выходной сигнал (в ЦАП).

Отрицательная логика — система сигналов, в которой логической единице соответствует низкий уровень напряжения, а логическому нулю — высокий.

Отрицательный сигнал (сигнал отрицательной полярности, нулевой сигнал) — сигнал, активный уровень которого — логический нуль. То есть единица — это отсутствие сигнала, нуль — сигнал пришел.

Отрицательный фронт сигнала (спад) — переход сигнала из единицы (из высокого уровня) в нуль (в низкий уровень).

Пассивный уровень сигнала — уровень, в котором сигнал не выполняет никакой функции.

Перепад (переход) сигнала — переключение сигнала из нуля в единицу или из единицы в нуль, то же что фронт сигнала.

Передний фронт сигнала — переход сигнала из пассивного уровня в активный.

ПЗУ — постоянное запоминающее устройство, постоянная память (ROM).

Пико- (п) — приставка для обозначения одной триллионной доли, 10^{-12}.

ПЛИС — программируемая логическая интегральная микросхема.

ПЛМ — программируемая логическая матрица (PAL).

Погрешность абсолютная — разность между измеренной величиной и ее истинным значением (погрешность измерения); разность между сформированной величиной и требуемым ее значением (погрешность формирования, погрешность воспроизведения).

Погрешность относительная — отношение абсолютной погрешности к требуемому (или истинному) значению данной величины. Часто измеряется в процентах.

Положительная логика — система сигналов, в которой логической единице соответствует высокий уровень напряжения, а логическому нулю — низкий.

Положительный сигнал (сигнал положительной полярности, единичный сигнал) — сигнал, активный уровень которого — логическая единица. То есть нуль — это отсутствие сигнала, единица — сигнал пришел.

Положительный фронт сигнала (или просто фронт) — переход сигнала из нуля (из низкого уровня) в единицу (в высокий уровень).

Поляриость сигнала — уровень сигнала, соответствующий его активности. Положительной полярности соответствует активный единичный сигнал, отрицательной полярности — активный нулевой сигнал.

Помехи — паразитные сигналы, накладывающиеся на информационные сигналы и искажающие их. Помехи могут наводиться извне (электромагнитным полем), а также возникать в цепях питания.

Помехозащищенность — параметр, характеризующий величину входного сигнала помехи, который еще не может изменить состояние выходных сигналов. Определяется разницей между напряжением U_{IH} и порогом срабатывания (помехозащищенность единичного уровня), а также разницей между порогом срабатывания и U_{IL} (помехозащищенность нулевого уровня).

Порог срабатывания — уровень входного напряжения, выше которого сигнал воспринимается как единица, а ниже — как нуль. Для ТТЛ микросхем он примерно равен 1,3...1,4 В.

ППЗУ — программируемое ПЗУ (PROM).

Принципиальная схема — наиболее подробная схема электронного устройства с указанием всех элементов, связей, входов и выходов, выполненная в соответствии со стандартом.

Протокол — порядок обмена сигналами между цифровыми устройствами.

Прямой выход — выход, выдающий сигнал положительной полярности.

Прямой счет — счет на увеличение выходного кода.

Разрядность (кода, шины) — количество двоичных разрядов кода или количество цифровых сигналов для передачи кода по шине.

Разъем — разъемное контактирующее устройство из двух частей (розетка и вилка), служащее для передачи сигналов и питания электронных схем.

Реверсивный счетчик — счетчик, работающий как в режиме прямого счета, так и в режиме обратного (инверсного) счета.

Регенерация — периодическое восстановление, освежение информации, записанной в динамическую память.

Розетка (гнездо) — часть разъема, в контакты которого входят контакты вилки (штекера).

РПЗУ — репрограммируемое ПЗУ (EPROM), информация в котором стирается ультрафиолетовым излучением и может быть записана вновь.

СБИС — сверхбольшая интегральная схема (VLSI).

Слово (двоичное) — группа бит (обычно 16, 32 или 64 бита), состоящая из нескольких байт.

Синхронизация — обеспечение согласованной во времени работы нескольких устройств, например, по общему тактовому сигналу.

Синхронный (параллельный) счетчик — счетчик, все разряды которого переключаются одновременно, синхронно с тактовым сигналом

Синхросигнал — то же, что тактовый сигнал.

Скважность — отношение периода следования импульсов к длительности этих импульсов.

Сокет (Socket) — то же, что колодка, контактирующее устройство-гнездо, в которое устанавливается микросхема с возможностью простой ее замены.

Спад сигнала — то же, что задний фронт сигнала (обычно — отрицательный фронт).

Строб (стробирующий сигнал) — управляющий сигнал, который своим уровнем определяет момент выполнения элементом или узлом его функции. В более общем смысле строб — это любой синхронизирующий сигнал, тактовый сигнал.

Стробирование — согласование во времени работы узлов и устройств с помощью строба.

Структурная схема — упрощенная схема электронного устройства, показывающая только основные узлы и важнейшие связи между ними.

Схема — электронный узел, устройство, а также их изображение на чертеже.

Такт — то же что тактовый сигнал, а также период тактового сигнала.

Тактовый сигнал — управляющий сигнал, который своим фронтом определяет момент выполнения элементом или узлом его функции. Иногда то же, что и стробирующий сигнал.

Тера- (Т) — приставка для обозначения триллиона, 10^{12}.

Тетрада (полубайт, ниббл) — группа из четырех бит, кодируемая одним символом в 16-ричной системе счисления.

Точность — показатель близости функционирования данного устройства к идеалу, обычно измеряется с помощью величины погрешности.

ТТЛ — транзисторно-транзисторная логика и соответствующая ей полупроводниковая технология (TTL).

ТТЛШ — технология ТТЛ с диодами Шоттки (TTLS). Характеризуется более высоким быстродействием при той же потребляемой мощности.

Узел — часть электронного устройства, выполняющая четко выделенную функцию или несколько взаимосвязанных функций.

Устройство (электронное) — функционально (а иногда и конструктивно) законченная электронная схема.

Флэш-память (Flash Memory) — разновидность РПЗУ с электрическим стиранием информации и возможностью многократной перезаписи.

Фронт сигнала — переход сигнала из нуля в единицу или из единицы в нуль, иногда в более узком значении «передний положительный фронт».

Функциональная схема — не слишком подробная схема электронного устройства, показывающая подробно только схемы отдельных, принципиально важных узлов устройства.

ЦАП — цифро-аналоговый преобразователь (DAC), преобразующий входной цифровой код в величину аналогового сигнала (тока или напряжения).

Цепочка — последовательное соединение нескольких узлов (или устройств), при котором выходы предыдущего узла соединяются со входами последующего узла. Также цепочкой называют любое соединение электронных компонентов (например, RC цепочка, LC-цепочка).

Цикл — последовательность обмена сигналами, в течение которого выполняется только одна операция.

Чип — то же, что интегральная микросхема, ИМС.

Шина — группа сигнальных линий, объединенных по какому-то принципу, например, шиной называют сигналы, соответствующие всем разрядам какого-то двоичного кода (шина данных, шина адреса). Иногда шиной называют также провод питания («шина питания») и общий провод («шина земли»).

Шифрация — преобразование номера приходящего входного сигнала в выходной двоичный код.

Ячейка (памяти) — элемент памяти (одноразрядный или многоразрядный), который служит для хранения информационного кода и может быть выбран с помощью кода адреса памяти.

Учебное издание

Новиков Юрий Витальевич

ОСНОВЫ ЦИФРОВОЙ СХЕМОТЕХНИКИ

БАЗОВЫЕ ЭЛЕМЕНТЫ И СХЕМЫ. МЕТОДЫ ПРОЕКТИРОВАНИЯ

Руководитель проекта **Т. Г. Хохлова**
Ведущий редактор **В. М. Матвеев**
Художественный редактор **В. П. Григорьев**
Художник **П. Инфанте**
Технический редактор **Е. В. Денюкова**
Корректор **Е. Н. Клитина**
Оригинал-макет подготовлен **С. А. Янковой**

Лицензия ЛР № 010174 от 20.05.97г.

Подписано к печати 14.09.2001 г. Формат 60 × 90/16.
Бумага офсетная. Гарнитура Times New Roman. Печать офсетная.
Объем 12,00 бум. л. Усл.-печ. л. 24,00. Уч.-изд. л. 22,04.
Изд. №6/9801. Тираж 5000 экз. Заказ 4589

Издательство «Мир»
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций
107996, Москва, 1-й Рижский пер., 2.

Диапозитивы изготовлены в издательстве «Мир»

Издательство и редакция не несут ответственности
за содержание публикуемых рекламных объявлений,
а также не предоставляют информации о деятельности фирм.

Отпечатано с готовых диапозитивов
в ППП «Типография Наука»
121099, Москва, Шубинский пер., 6.

Налоговая льгота — общероссийский классификатор продукции
ОК-005-93, том 2; 953000 — книги, брошюры

СОВРЕМЕННАЯ СХЕМОТЕХНИКА

Ю. В. Новиков

ОСНОВЫ ЦИФРОВОЙ СХЕМОТЕХНИКИ

БАЗОВЫЕ ЭЛЕМЕНТЫ И СХЕМЫ
МЕТОДЫ ПРОЕКТИРОВАНИЯ

В этой книге содержится тот необходимый минимум знаний, который должен иметь и которым должен свободно и активно пользоваться каждый профессиональный разработчик цифровой аппаратуры. В то же время она позволяет освоить азбуку цифровой схемотехники даже тем читателям, которые имеют слабое представление об электронике вообще.

Для более подготовленного или по-настоящему заинтересованного читателя эта книга — руководство, которое призвано сформировать действительно хорошего проектировщика, способного оптимальным образом строить высокоэффективные цифровые системы самой различной степени сложности и четко представляющего себе взаимосвязь всех процессов в этих системах сверху донизу.

Из этой книги вы узнаете:

- Как функционируют и взаимодействуют между собой все основные типы цифровых микросхем — от самых простых до самых сложных.
- Какие модели и уровни представления цифровых микросхем используются при проектировании цифровых электронных систем.
- Каковы принципы выбора оптимальной стратегии построения цифровые устройств на основе самых различных микросхем.

Для студентов, преподавателей, профессионалов.

ISBN 5-03-003449-8