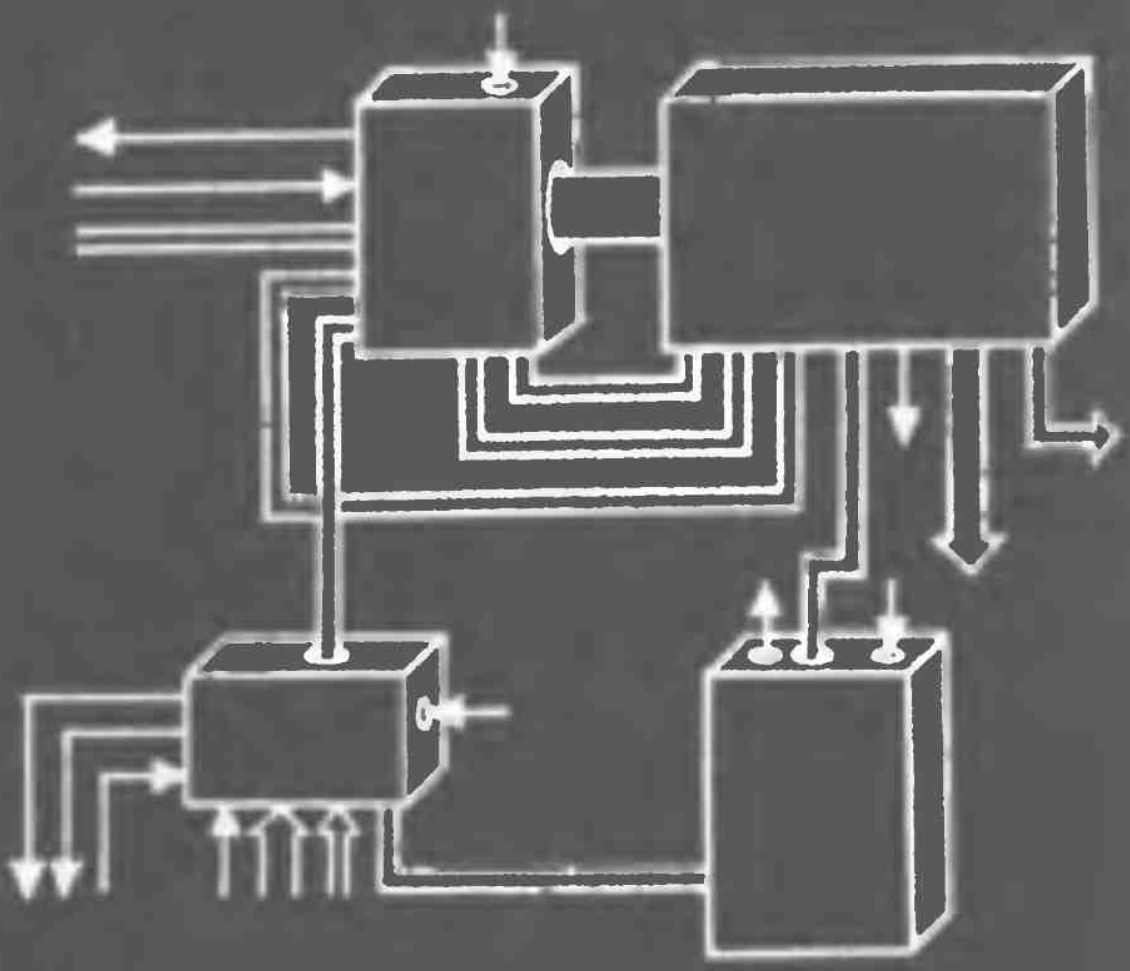
Э. КЛИНГМАН



ПРОЕКТИРОВАНИЕ СПЕЦИАЛИЗИРОВАННЫХ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

ББК 32.973
К49
УДК 681.3

Клингман Э.

К49 Проектирование специализированных микропроцессорных систем: Пер. с англ. — М.: Мир, 1985.— 363 с., ил.

В книге американского автора рассматриваются вопросы проектирования цифровых систем с использованием устройств с микропрограммным управлением и разрядно-модульной организацией. Приведены примеры построения микропроцессорных систем, предназначенных для решения различных технических задач.

Для специалистов в области вычислительной техники.

$К\frac{2405000000\text{-}408}{041(01)\text{-}84}$ 164-85, ч. 1

ББК 32.973
6Ф7.3

Редакция литературы по информатике и электронике

ПРЕДИСЛОВИЕ К РУССКОМУ ИЗДАНИЮ

В течение последних лет в нашей стране вышло несколько десятков книг советских и зарубежных авторов по вопросам проектирования и использования микропроцессоров и микропроцессорных систем. Интерес к литературе по этой тематике не ослабевает и в настоящее время.

Автор данной книги Э. Клингман известен советскому читателю по изданной в СССР монографии[1]. Новая книга Э. Клингмана знакомит главным образом со специальными вопросами создания цифровых систем на основе устройств с микропрограммным управлением и разрядно-модульной организацией. Значительное внимание автор уделяет вопросам, которые в литературе по микропроцессорным системам освещены лишь в незначительной степени.

Все большее распространение при разработке специализированных вычислительных систем приобретают матричные логические схемы и устройства с микропрограммным управлением и разрядно-модульной организацией. Такой набор средств характеризуется гибкостью и универсальностью. Использование указанных средств позволяет создавать системы высокого быстродействия, архитектура которых учитывает специфику решаемой задачи (точность вычислений, характер и структуру данных и т. п.).

Одной из особенностей книги является попытка автора дать вводные положения теории машин с конечным числом состояний, определить ее связь с традиционной теорией автоматов и использовать этот формальный аппарат для изучения матричных логических схем. Принципам построения и использования матричных логических схем уделено большое внимание. Способ формального описания логических схем матричного типа согласуется с описанием машин с конечным числом состояний. Используемые при этом понятия и система обозначений могут повлиять на развитие аппарата формального описания проектов цифровых систем.

Большое внимание автор уделяет устройствам управления выполнением программы. Их рассмотрение он начинает с абстрактного представления на уровне матричных логических схем. Подробно описаны широко распространенные устройства данного типа.

Ценность материала книги в значительной мере определяется представленными в ней примерами проектирования реальных систем: системы обработки изображений с использованием метода цифровой фильтрации, основанного на быстром преобразовании Фурье; вычислительной системы, в которой реализуются операции над числами, представленными в форме с плавающей точкой;

---

[1] Klingman E. E. Microprocessor Systems Design, Prentice-Hall, Englewood Cliffs, 1977. — Клингман Э. Проектирование микропроцессорных систем. — М.: Мир, 1980. В дальнейшем под обозначением «Кл., 1980» будет подразумеваться данная книга. — *Прим. перев.*

системы передачи изображений с применением процедуры адаптивного кодирования. Описанные принципы проектирования систем различного назначения вполне универсальны. Рассмотрение этих примеров принесет пользу как проектировщикам специализированных цифровых систем, так и тем, кто только приступил к освоению техники проектирования с применением устройств с микропроцессорным управлением и разрядно-модульной организацией.

Книга предназначена для специалистов, связанных по роду деятельности с проектированием микропроцессорных систем. Она может быть также полезной студентам старших курсов и аспирантам, специализирующимся в области проектирования систем обработки данных на базе микропроцессоров.

Л. В. Шабанов

ПРЕДИСЛОВИЕ

Как известно, при проектировании цифровых систем использование последних достижений в данной области часто игнорируется. В данной работе излагаются вопросы, которые должен знать проектировщик, намеревающийся использовать логические схемы матричного типа или устройства с микропрограммным управлением и разрядно-модульной организацией при разработке архитектуры цифровых вычислительных систем. Желательно, чтобы читатель, приступая к изучению этой книги, ознакомился с моей предыдущей работой (Кл., 1980). Книга адресована студентам и инженерам, специализирующимся в области проектирования систем и желающим изучить особенности разработки систем с микропрограммным управлением.

Для проектировщика, отдающего все силы своему призванию, деятельность в минувшем десятилетии, несомненно, была весьма плодотворной. Ведь обновление вычислительных средств происходит каждые 3—5 лет.

В настоящее время бо́льшая часть изданий, относящихся к устройствам с микропрограммным управлением и разрядно-модульной организацией, содержит лишь краткое изложение отдельных вопросов и несколько примеров, иллюстрирующих работу простейших систем. В данной книге представлены результаты детального проектирования ряда систем. Обычно такие системы содержат центральный процессор (ЦП), эмулирующий работу мини-ЭВМ (обычно это мини-ЭВМ фирмы Data General Nova или ЭВМ PDP-11 фирмы DEC). Подобный подход имеет то преимущество, что связывает между собой принцип микропрограммирования и традиционной архитектуры ЭВМ. Кроме того, он приводит и к более рациональному использованию компонентов системы.

Поскольку фирма IBM разработала технологию изготовления ЦП Системы 370 на кристалле в виде вентильной матричной сверхбольшой интегральной схемы (СБИС), стала очевидной принципиальная возможность эмуляции любой мини-ЭВМ с помощью СБИС. Следовательно, отпадает необходимость в использовании множества кристаллов для построения ЦП.

Истинная ценность устройств с микропрограммным управлением и разрядно-модульной организацией и логических схем матричного типа определяется их гибкостью и универсальностью. Четкое разделение функций и модульная организация этих устройств позволяют с их помощью легко и эффективно реализовать практически любой тип логики.

Наиболее подходящим применением устройств с разрядно-модульной организацией и логических схем матричного типа являются системы специального назначения.

Подход к проектированию ЦП с учетом экономического фактора определяет в большинстве случаев выбор микропроцессора Z8000 фирмы Zilog или M68000 фирмы Motorola, либо какой-нибудь мини-ЭВМ или большой универсальной ЭВМ. Однако существуют такие проекты, для реализации которых затраты, связанные с использованием устройств с микропрограммным управ-

лением и разрядно-модульной организацией и логических схем матричного типа, являются оправданными вследствие исключительно высокого быстродействия, достигаемого за счет настройки архитектуры ЭВМ на наиболее эффективную реализацию задачи.

В предлагаемой читателю книге рассматриваются отдельные блоки, входящие в микропроцессорную систему. Использование этих блоков иллюстрируется на трех примерах архитектуры ЭВМ специального назначения.

В гл. 1 содержатся общие положения о цифровых вычислительных системах, причем принципы построения таких систем излагаются на примере проектирования простейшего процессора.

В гл. 2 описываются стандартные блоки, которые могут быть использованы в качестве процессорных элементов систем с микропрограммным управлением.

В гл. 3 включен вспомогательный материал, который на начальном этапе изучения проблемы может быть опущен. Рассматриваемые здесь концепции являются универсальными и широко распространенными, в связи с чем изложены здесь кратко. Приводится определение машин с конечным числом состояний и описываются различные классы этих машин.

В гл. 4 рассматриваются концепции построения и использования логических схем матричного типа. В этой области микропроцессорной техники за последние годы наблюдается существенный прогресс. При описании схем сделана попытка применить такую систему символических обозначений, которая может служить некоторым обобщенным языком проектирования цифровых вычислительных систем. В области технологии электронных схем в минувшее десятилетие произошла настоящая промышленная революция, и значительных изменений в 80-е годы здесь не ожидается. Высокая эффективность технологии изготовления электронных схем приводит к тому, что разработчики оказываются буквально «заваленными» элементной базой, включающей тысячи высокопроизводительных стандартных блоков. Это требует создания и использования адекватного формального описания, которое могло бы применяться при проектировании архитектуры ЭВМ как на уровне вентилей, так и на уровне сверхбольших интегральных схем. Хотелось бы надеяться, что введенные в гл. 4 понятия и символические обозначения логических схем матричного типа помогут решить данную проблему.

Рассматриваемые здесь устройства, построенные на логических схемах матричного типа, включают в себя все более возрастающее количество малых интегральных схем. Способ формального описания логических схем матричного типа, разработанный в данной главе, далее применяется к различным классам машин, изученным в гл. 3. Для представления машины класса 2 (счетчика) используются карты Карно и описывается построение ряда счетчиков на логических схемах матричного типа. Последний раздел главы на начальном этапе изучения проблемы может быть опущен.

В гл. 5 рассматриваются устройства управления выполнением программы, начиная с их абстрактного представления на уровне логических схем матричного типа. Подробно описывается ряд широко распространенных устройств управления. С этой главой необходимо ознакомиться, прежде чем приступать к любой из оставшихся глав книги.

Гл. 6 содержит пример проектирования архитектуры системы. Описывается задача фильтрации двумерных сигналов и приводятся соответствующие уравнения. В качестве математической модели работы фильтра служит быстрое преобразование Фурье. Структура, полученная в результате математического анализа, используется далее для построения архитектуры, которая может быть реализована на базе устройств с микропрограммным управлением и разрядно-модульной организацией. Разработка системы обработки изображений выполнялась по заказу командования ракетными войсками США.

В гл. 7 приводится пример детального проектирования архитектуры вычислительной системы, использующей десятичную арифметику с плавающей точкой. Это наиболее завершенный и детально проработанный из рассматриваемых в книге проектов. Здесь освещены все аспекты, относящиеся к реаль-

ному проектированию систем на базе микропроцессоров с разрядно-модульной организацией.

В гл. 8 описывается еще один пример организации обработки двумерных сигналов. Здесь рассматривается система передачи изображений, построенная с использованием оригинальной адаптивной схемы кодирования, производящей сжатие визуальных данных на одном конце линии передачи и восстанавливающей исходное изображение на ее другом конце. Система построена на базе машины, разработанной автором для японской фирмы Ricoh. (Я весьма признателен фирме за любезное разрешение включить в книгу информацию о данной системе.) Машина представляет собой великолепный пример построения реальной системы с помощью устройств с микропрограммным управлением и разрядно-модульной организацией. Фирма Roger Woodward разработала систему, основанную на рассматриваемом здесь прототипе.

В заключение хотелось бы отметить, что знакомство с изложенным в книге материалом поможет читателю изучить вопросы проектирования СБИС. В книге не рассматриваются физические процессы, протекающие в электронных схемах, однако большое внимание уделено формальному описанию проектирования матричных логических схем, которые в настоящее время все шире используются в процессе создания однородных транзисторных структур в СБИС. В процессе проектирования СБИС возникают две основные проблемы: 1) выбор материала и принципов физической реализации устройства и 2) разработка архитектуры устройства. При использовании λ-правил и средств машинного проектирования бóльшая часть электрических характеристик оказывается «скрытой» от разработчика архитектуры вычислительной системы. Фирмы — производители интегральных схем на кристаллах используют в качестве исходных данных архитектурные решения. Эти решения являются основой разработки микроэлектронных устройств, построенных на СБИС. Вопрос о выборе метода проектирования — применение разрядно-модульной организации или использование СБИС — все в большей степени приобретает экономический характер. В том случае, если должно быть изготовлено ограниченное число вычислительных систем, часто оказывается целесообразным применение устройств с микропрограммным управлением и разрядно-модульной организацией, тогда как при большом планируемом выпуске разрабатываемой системы более подходящим является использование СБИС.

Э. Клингман

ГЛАВА 1

ОСНОВЫ ЦИФРОВЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Формальные аналоги человеческого мозга, известные под названием «машины для распознавания образов», осуществляют отображение любых сигналов, появляющихся на их входе, в определенные группы или классы сигналов в соответствии с заданными критериями распознавания (эти критерии могут быть самомодифицируемыми) или согласно определенной схеме классификации. На диаграмме состояний входные сигналы системы формально представляются точками, а классы, к которым относятся эти сигналы, — зонами или областями. Таким образом, класс является функцией принадлежности набора точек, представляющих входные сигналы, определенной группе сигналов.

При проектировании систем процесс разделения множества возможных входных сигналов на группы в соответствии со схемой их классификации (рис. 1.1) известен под названием *анализа*. Определение уровня разрешения сигналов и построение с учетом этого параметра сетки значений, в определенные ячейки которой попадают точки, соответствующие входным сигналам, называются *моделированием*, или *отображением*, причем конкретный выбор шага этой сетки определяет реализацию системы. Процесс выбора блоков, из которых компонуется система, и разработки интерфейса между блоками обычно называют *синтезом*. В добавление к указанным этапам проектирования разработчик вычислительной системы должен также установить, какие цели должны быть достигнуты для обеспечения заданных характеристик и по каким параметрам могут быть приняты компромиссные решения с целью *оптимизации* проекта.

Как показывают мои наблюдения, строго организованное проектирование имеет место главным образом на бумаге. Однако описание процедуры такого проектирования позволяет осуществлять в ходе разработки корректирующую обратную связь и тем самым приносит реальную пользу. Хотя совершенствование проектов, о которых идет речь в данной книге, происходило большей частью параллельно, их описание должно вестись по-

Рис. 1.1. Первый этап анализа — классификация.

следовательно. Я пытался найти для проектируемых устройств формальные аналоги, чтобы представить эти устройства в виде упорядоченных структур.

Проектирование сложных систем — это неустойчивый эволюционный процесс. При рассмотрении развития структуры системы во времени выясняется, что это развитие происходит не плавно, а импульсивно. Рассматриваемый подход к проектированию называется *проектированием снизу вверх*. Хотя в каждом случае цель проектирования представляется вполне определенной, первоначальный вариант системы оказывается слишком сложным. В то же время имеется слишком много неясных вопросов, чтобы можно было полностью использовать принцип *проектирования сверху вниз*, при котором система строится путем перехода от наиболее общих абстрактных представлений к реализации аппаратного и программного обеспечения. В таких случаях пытаются установить самые общие принципы построения системы, а далее начинают заниматься их реализацией, развивая эти принципы в ходе разработки и связывая их с другими принципами и подходами. Подобную методику проектирования обычно называют *поэтапным усовершенствованием*.

Таким образом, идеальным методом является проектирование, выполняемое по принципу сверху вниз, при котором производится переход от формулировки проблемы в самом общем виде к конкретной реализации системы. Однако с практической точки зрения наиболее реалистическим является подход, заключающийся в выделении в системе отдельных подсистем, структура которых определяется сущностью решаемой задачи, и последующем изучении этой структуры с учетом блоков, которые могут быть применены для ее реализации как в данный момент, так и в будущем. Такой подход является проектированием по принципу снизу вверх, поскольку проектирование ведется, начиная с элементов, находящихся в нижней части «дерева проекта».

Реализацию системы определяют предъявляемые к ней конкретные требования. Исходя из этих требований, выраженных в терминах стоимости, габаритов, производительности, времени отклика и т. п., разработчик должен осуществить определенный выбор среди возможных архитектур, языков и технологий. Мы постараемся в этой книге рассмотреть принципы, применимые ко всем различным реализациям микропроцессорных систем, и

привести примеры проектирования, достаточно подробно иллюстрирующие используемые средства и процедуры для конкретных реализаций. Аппаратная реализация обычно базируется на семействах логических схем, использующих технологию ТТЛ (транзисторно-транзисторная логика) и *n*-МОП (*n*-канальная логика на металлооксидных полупроводниках), предоставляющую наибольшие возможности по созданию различных функциональных блоков и обеспечивающую наилучшие стоимостные показатели.

Следует подчеркнуть, что, хотя мы попытаемся применить и пояснить на примерах только основные принципы проектирования цифровых вычислительных систем, разработчик, стремящийся добиться успеха, должен быть знаком с огромным диапазоном устройств, которые могут применяться в процессе проектирования. Бурный рост новых устройств, начавшийся в конце 60-х годов и продолжавшийся в 70-е годы, стал еще более интенсивным в 80-е годы. В отличие от других областей техники, в которых разработчику, для того чтобы быть в курсе важнейших событий и оставаться на уровне развития техники, достаточно изучения научных журналов, в области проектирования вычислительных систем необходимо также учитывать ситуацию на рынке.

ОСНОВНЫЕ ФУНКЦИИ

Возможность представления систем в терминах выполняемой ими работы (т. е. путем определения выходных сигналов, выдаваемых в ответ на определенные входные сигналы) позволяет рассматривать понятие «функции системы» абстрактно, в качестве формального аналога режимов функционирования системы, которые физически могут быть реализованы большим числом различных способов. В то же время возможность конструирования устройств, осуществляющих распознавание входных сигналов в соответствии со схемой их классификации, позволяет создавать реальные аналоги формальных систем, представляемых наборами точек на диаграмме состояния. В частности, как показано на рис. 1.2, могут быть построены цифровые вычислительные системы, являющиеся физическими аналогами формальных числовых и логических систем. В соответствии с наиболее распространенной схемой классификации сигналы, используемые в цифровых системах, делятся на три группы: высокого уровня, низкого уровня и недопустимые, или запрещенные. Устройства цифровой обработки данных, использующие подобную классификацию сигналов, показаны на рис. 1.3.

Можно сконструировать схемы с двумя устойчивыми состояниями, осуществляющие распознавание непрерывных шумовых

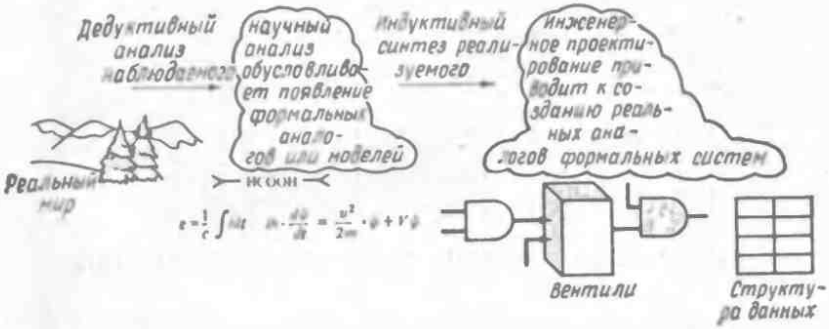


Рис. 1.2. Процесс развития моделей цифровой обработки данных.

сигналов, поступающих на их входы, и формирующие на выходах двоичные дискретные сигналы. Выходные сигналы представляют собой подмножество возможных, определенных заранее входных сигналов. Следовательно, достоверные входные сигналы, поступающие в систему, отображаются в достоверные выходные сигналы; последние подобным же образом используются в качестве достоверных входных сигналов на следующих уровнях системы.

В оставшейся части данной главы для изложения основополагающих концепций, относящихся к типовым процессам обработки данных, используются широко известные базовые схемы вентилей. Логические операции, выполняемые этими схемами, их графическое изображение и соответствующие им таблицы истинности представлены на рис. 1.4.

Используя базовые схемы вентилей в качестве элементарных блоков, можно построить системы произвольной сложности. При таком подходе к проектированию (на уровне вентилей) системы быстро становятся неуправляемыми, так что возникает настоятельная потребность в упрощении их структуры. Более простые по структуре системы строятся из блоков, включающих множество вентилей. Простейшие такие блоки реализуют специ-

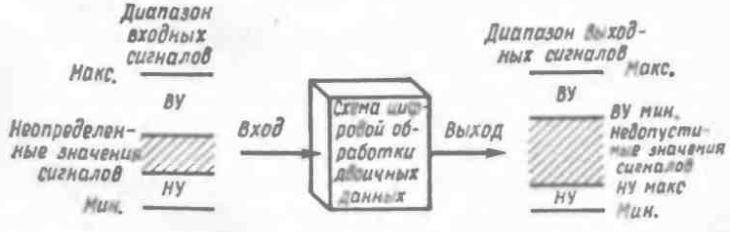


Рис. 1.3. Схема классификации сигналов в электронных устройствах цифровой обработки данных. (ВУ — сигналы высокого уровня, НУ — сигналы низкого уровня.)

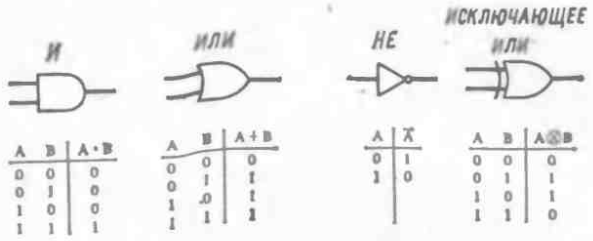


Рис. 1.4. Вентили И, ИЛИ, НЕ, ИСКЛЮЧАЮЩЕЕ ИЛИ.

фические функции, указанные выше, т. е. представляют собой логические схемы на вентилях или определенные комбинации таких логических схем. В зависимости от числа вентилей, входящих в состав этих блоков, последние классифицируются следующим образом:

Количество вентилей	Тип блока
10—100	МИС (интегральные схемы малого уровня интеграции)
100—1000	СИС (интегральные схемы среднего уровня интеграции)
1000—100 000	БИС (интегральные схемы большого уровня интеграции)
100 000 и более	СБИС (интегральные схемы сверхбольшого уровня интеграции)

В гл. 3 мы исследуем современный подход к проектированию блоков определенного функционального назначения, основанный на использовании упорядоченной совокупности вентилей, известной под названием *логической схемы матричного типа* и изображенной символически на рис. 1.5. Такой способ физической реализации функционального блока приводит каждый раз к существенным затратам на проектирование и изготовление устройства. По этой причине существует и используется конечное, хотя и большое, число устройств подобного типа. В повседневной деятельности существует практически неограниченное многообразие функций, выполнение которых с помощью цифровой вычислительной техники является экономически допустимым и целесообразным. В связи с этим желательно иметь функциональные блоки более общего назначения, из которых можно было бы собирать конкретные системы с учетом их особенностей. Обычным решением данной проблемы является использование программируемого процессора, обрабатывающего данные.

Пусть, например, требуется построить функциональный блок, позволяющий извлекать числа из блока регистров, складывать несколько чисел и записывать сумму в определенный регистр.

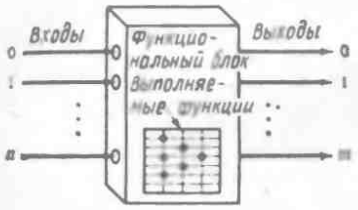


Рис. 1.5. Условное изображение логической схемы матричного типа, представляющей собой функциональный блок, построенный на вентилях.

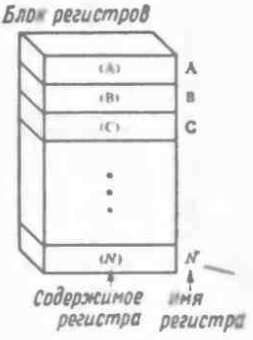


Рис. 1.6. Блок регистров общего назначения, используемый в качестве памяти.

Если обозначить регистры памяти символами A, B, C, а содержимое некоторого регистра B — как (B), то блок регистров схематично может быть изображен так, как показано на рис. 1.6. Использование вентилей для построения ячеек памяти (регистров) рассмотрено мною ранее (Кл., 1980).

Описанный здесь массив ячеек памяти является стандартным, так что необходимо выполнить проектирование соответствующего функционального блока. Вспомним, что теоретически любую конечную операцию можно представить в виде последовательности простейших операций, таких, как сложение со знаком, умножение со знаком, логические операции, хотя практически это может оказаться не осуществимо. Таким образом, можно использовать стандартное арифметическо-логическое устройство (АЛУ), например устройство 74181 фирмы Texas Instruments, вместе с блоком регистров памяти, причем произвольные операции над данными, хранимыми в памяти, могут выполняться в виде последовательности простейших операций, реализуемых АЛУ.

Операция C←[(A)+(B)] может быть реализована с помощью следующей последовательности микроопераций:

Поместить (B) на один из входов АЛУ.

Поместить (A) на другой вход АЛУ.

Выполнить в АЛУ сложение (A) и (B).

Записать выходные данные АЛУ в регистр C.

Группу регистров, используемых для размещения данных, называют *блоком*, или *набором регистров*. Тракты передачи электрических сигналов, соединяющие блок регистров с АЛУ или другими устройствами, носят название *шины*. В реальных

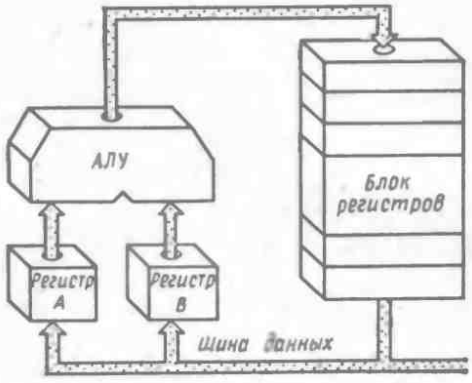


Рис. 1.7. Схема соединения шины данных, АЛУ и регистров.

микроэлектронных устройствах шина представляет собой нанесенный на плату электропроводный канал.

Схема соединения регистров памяти и АЛУ изображена на рис. 1.7. Операции над данными выполняются во времени последовательно. К каждой базовой элементарной операции производится обращение как к микрооперации. Операции, подлежащие выполнению в процессоре, определяются микрокомандами, поступающими последовательно от источника микрокоманд (обычно из памяти, в которой хранятся микрокоманды). Микрокоманды задают последовательность сигналов, инициирующих работу отдельных компонентов (управляющих сигналов), которые обеспечивают переход системы из одного конечного состояния в другое. Мы будем предполагать наличие управляющих сигналов всегда, когда они являются необходимыми, стремиться формировать рациональную последовательность управляющих сигналов и представлять ее в виде, приемлемом для проектирования устройства.

В упомянутой выше книге (Кл., 1980) рассматривался принцип двоичного кодирования-декодирования данных и описывались блоки, осуществляющие кодирование или декодирование поступающей в них информации. Работа, выполняемая процессором общего назначения, определяется потоком поступающих в него команд (представленных в двоичном коде). Концепции использования команд, их декодирования и выполнения достаточно просты, хотя аппаратная реализация команд является относительно сложной. Подсистема вычислительной машины, осуществляющая прием, декодирование и выполнение последовательности команд, или программы, называется *центральным процессором* (ЦП). Для иллюстрации основных принципов функционирования этой подсистемы выполним проектирование простого процессора на уровне вентилей. Чтобы принципы, ле-

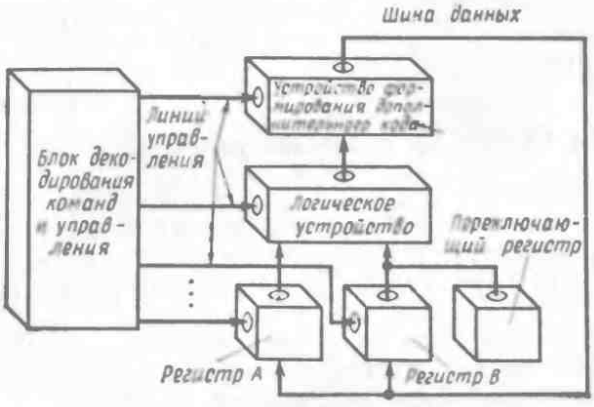
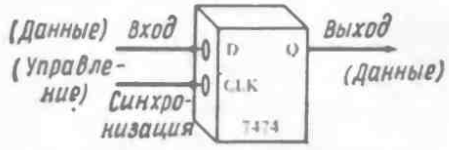


Рис. 1.8. Структура простейшего процессора.

жащие в основе проектирования, не исчезали за техническими подробностями, желательно использовать простейшую схему. Для решения поставленной задачи достаточно рассмотреть 1-разрядный процессор (рис. 1.8). В состав процессора входят два регистра памяти и логическое устройство, способное выполнять операции над двумя 1-битовыми словами данных. Логическое устройство включает устройство преобразования данных в дополнительный код, формирующее такой код из данных, появляющихся на выходе логического устройства. Тракты данных обеспечивают передачу информации как из регистров в логическое устройство, так и в обратном направлении; причем предусмотрены средства управления работой этих трактов. Управляющие сигналы генерируются в устройстве декодирования команд и управления. Процессор также включает переключающий регистр, играющий роль входного порта. Регистры устройства состоят из триггеров, фиксирующих данные на входных линиях при поступлении управляющего или синхронизирующего сигнала. В данном случае наиболее подходящими являются D-триггеры. Сигнал на выходе D-триггера 7474 (производства фирмы Texas Instruments) будет совпадать с сигналом на входе при поступлении переднего фронта синхроимпульса (рис. 1.9).

Для простоты мы сначала рассмотрим устройство, позволяющее выполнять логические операции И и ИЛИ над двумя 1-битовыми элементами данных с преобразованием или без преобразования результата операции в дополнительный код. Структура такого устройства показана на рис. 1.10. Здесь функции, реализуемые вентилями, изображаются с помощью переключателей, установленных в трактах передачи данных. Действие управляющих сигналов условно обозначено ключами в соответ-



D	Q
t_n	t_n+1
0	0
1	1

Рис. 1.9. D-триггер и соответствующая таблица истинности.

ствующих линиях. Набор управляющих сигналов приведен в табл. 1.1. Электрическое управление трактом передачи сигналов может быть осуществлено с помощью вентилей так, как это изображено на рис. 1.11.

Работа рассматриваемого нами устройства синхронизируется таким образом, что его переход из одного состояния в другое производится по сигналам генератора синхроимпульсов. При поступлении переднего фронта этих импульсов вход синхронизируемого регистра запирается, а на его выходе появляется сигнал. Как следует из рис. 1.10, этот выходной сигнал может быть быстро передан через тракты системы и вновь появиться на входе в течение интервала времени прохождения переднего фронта синхроимпульса, измеряемого наносекундами. Данная ситуация обычно требует тщательного анализа и учета временны́х соотношений. Однако этого анализа можно избежать путем добавления к логическому устройству фиксатора выходных данных и использования дополнительной фазы синхроимпульсов для фиксации выходных данных логического устройства. Синхроимпульсы дополнительной фазы могут быть сформированы из синхроимпульсов основной фазы, при поступлении которых

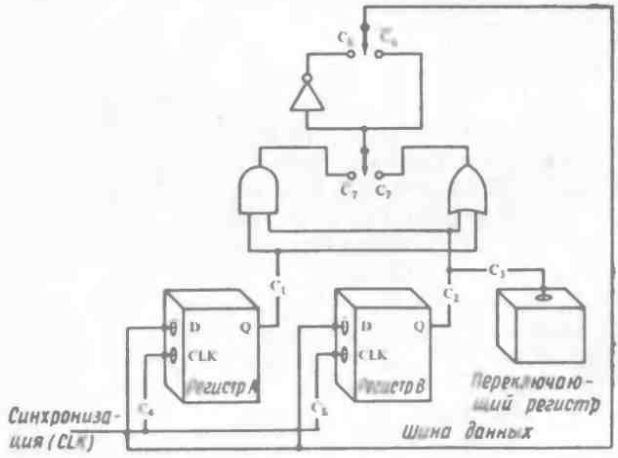


Рис. 1.10. Структура 1-разрядного процессора.

Таблица 1.1. Сигналы управления 1-разрядного процессора

Управляющий сигнал	Функция
C_1	Соединить выход регистра А с логическим устройством через порт 1
C_2	Соединить выход регистра В с логическим устройством через входной порт 2
C_3	Соединить переключающий регистр с логическим устройством через порт 2
C_4	Загрузить в регистр А данные, поступающие с шины
C_5	Загрузить в регистр В данные, поступающие с шины
C_6	Сформировать дополнительный код выходных данных логического устройства и поместить его на шину данных
C_7	Выполнить с помощью логического устройства следующие операции: И (сигнал C_7 отсутствует) ИЛИ (сигнал C_7 поступил)

производится загрузка информации в регистры с шины данных и формирование последовательности управляющих сигналов. Таким образом, теперь мы имеем:

1) фазу синхронизации, во время которой инициируется очередной цикл работы устройства благодаря появлению соответствующих сигналов на управляющих вентилях, — $РН_I$;

2) фазу синхронизации, во время которой производится фиксация результата выполнения операции на шине данных, — $РН_{DB}$;

3) фазу синхронизации, во время которой производится фиксация данных, поступающих с шины данных, в соответствующем регистре, — $РН_{REG}$.

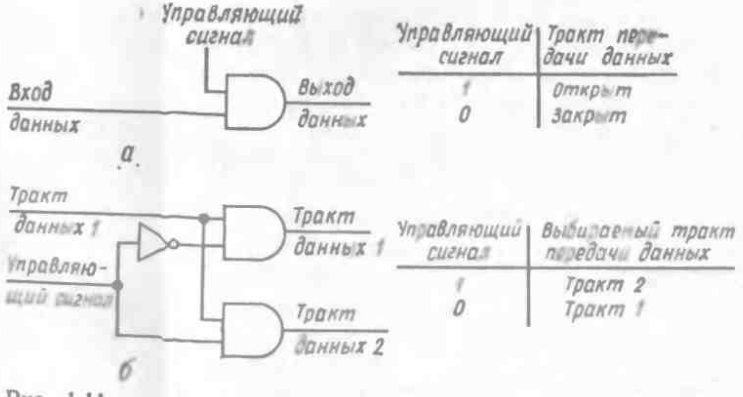


Рис. 1.11. *а* — Схема открытия/закрытия тракта передачи данных; *б* — схема управления выбором тракта передачи данных.

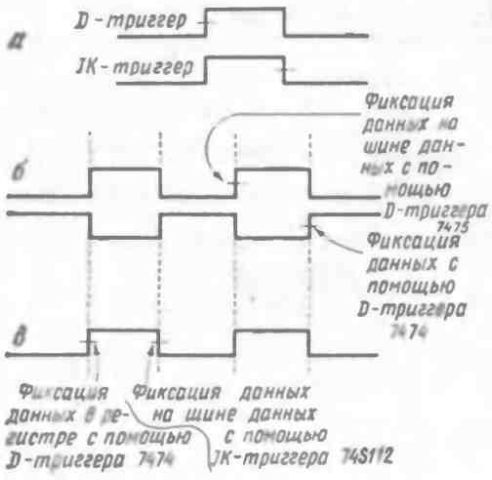


Рис. 1.12. *а* — Фиксация сигналов при использовании D- и JK-триггеров. [Показаны точки, соответствующие моментам фиксации выходных данных при использовании D-триггера 7474, фиксирующего данные при прохождении положительного фронта синхроимпульса (т. е. увеличении уровня сигнала с низкого до высокого), и JK-триггера 74S112, фиксирующего данные при прохождении отрицательного фронта синхроимпульса (уменьшении уровня сигнала с высокого до низкого).] *б* — фиксация входных и выходных данных при использовании инвертированных синхроимпульсов. [Представлены фазы синхроимпульсов, необходимые для фиксации сигналов в регистрах в надлежащие моменты времени при использовании D-триггеров 7474.] *в* — использование однофазной синхронизации с двумя типами фиксаторов сигналов. [Одной фазы синхроимпульсов достаточно, если для фиксации данных на шине данных и в регистрах используются D-триггер 7474, фиксирующий данные при прохождении положительного фронта синхроимпульса, и триггер 74S112, фиксирующий данные при прохождении отрицательного фронта синхроимпульсов.]

Целесообразно также рассмотреть процесс синхронизации работы устройства с учетом используемого в системе способа фиксации сигналов. D-триггер 7474 срабатывает при достижении определенного уровня напряжения сигналов, и его работа не зависит непосредственно от времени прохождения фронта синхроимпульсов. Когда в линии синхронизации установлен высокий или низкий уровень сигнала, триггер не реагирует на сигнал, поступающий на его вход D. Информация с входа D-триггера передается на выход Q во время поступления положительного фронта синхроимпульса, причем после превышения сигнала на входе линии синхронизации порогового значения вход D-триггера блокируется. Для обеспечения второй фазы синхронизации исходный синхроимпульс инвертируется, и, в то время как положительный фронт исходного синхроимпульса (CLOCK) используется для запирания одних элементов, положительный фронт инвертированного синхроимпульса ($\overline{\text{CLOCK}}$) осуществляет запирание других элементов. Заметим, однако, что JK-триггер, в котором данные передаются на выход Q во время прохождения отрицательного фронта синхроимпульса, может служить для фиксации данных на шине данных при использовании только одной фазы синхроимпульсов. Для этой цели подходит реагирующий на прохождение отрицательного фронта

синхроимпульса JK-триггер 74S112. Фиксация сигналов при использовании D- и JK-триггеров показана на рис. 1.12 с выделением отдельных фаз синхронизации.

ПОСЛЕДОВАТЕЛЬНОЕ ВЫПОЛНЕНИЕ КОМАНД

Последовательное выполнение процессором операций по обработке данных предполагает использование генератора синхронизирующих импульсов для организации последовательной передачи команд на линии управления. Причем с целью предотвращения фиксации данных в регистрах во время выдачи новой команды (т. е. при смене команд) желательно формировать на-

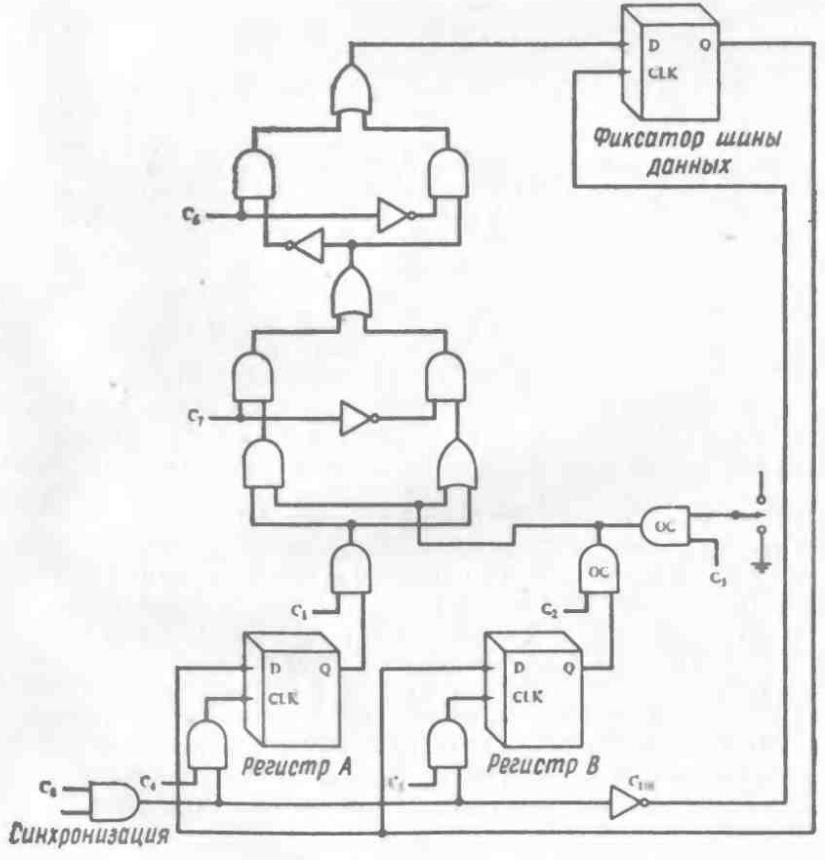


Рис. 1.13. Структура 1-разрядного процессора, включающая вентили и цепи синхронизации.

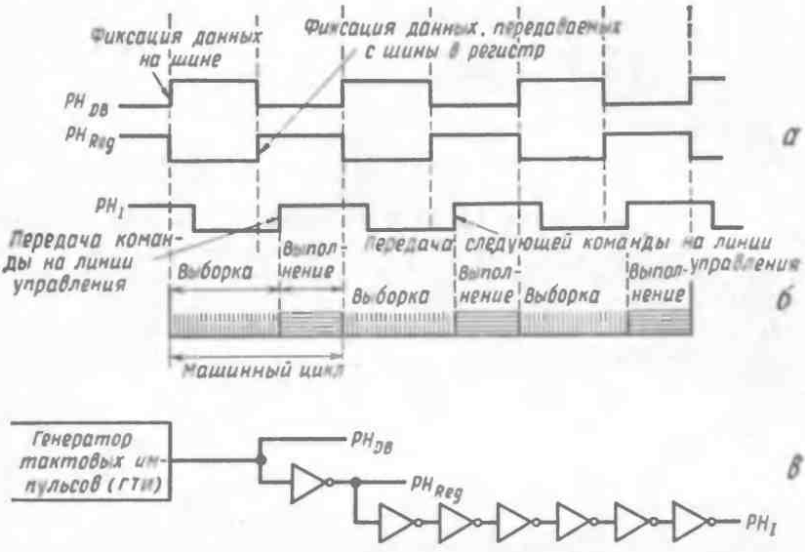


Рис. 1.14. *а* — **Фазы синхроимпульсов при использовании фиксаторов, построенных только на D-триггерах;** *б* — **фазы выборки и выполнения команд;** *в* — **простейшая схема формирования требуемых фаз синхроимпульсов.**

ряду с основной фазой синхроимпульсов еще одну, дополнительную фазу синхронизации. Эта фаза синхронизации может быть получена из основной путем сдвига по фазе или задержки синхроимпульсов. Задержка может быть осуществлена просто путем размещения нескольких инверторов между линиями основных и сдвинутых по фазе (задержанных) синхроимпульсов. Задержка обеспечивает хранение данных в соответствующем регистре перед выдачей новой команды. Такой способ синхронизации работы устройства показан на рис. 1.13.

Хотя средства обеспечения последовательного выполнения команд до сих пор еще не рассматривались, основные особенности синхронизации машин последовательного действия очевидны. Выполнение команд включает две фазы: выборки (извлечения) команды и собственно ее выполнения.

Соотношение между этими фазами и фазами синхронизации показано на рис. 1.14. Большинство систем разрабатывается так, что фазы выборки и выполнения команд не перекрываются; однако системы с более развитой логикой допускают такое перекрытие, уменьшая тем самым длительность цикла выполнения команды[1]).

[1]) Имеется в виду длительность цикла, усредненная по множеству последовательно выполняемых команд. — *Прим. перев.*

АЛГОРИТМЫ

Исходным положением при проектировании 1-разрядного процессора является тот факт, что самые сложные процедуры могут быть представлены в виде последовательности простейших операций. Например, умножение двух чисел может быть выполнено с помощью определенной последовательности операций сложения, и, таким образом, устройство, включающее схему сложения (сумматор) и соответствующие команды, может быть использовано для операции умножения. Подобным образом последовательность управляющих сигналов, обеспечивающая выполнение операции ИСКЛЮЧАЮЩЕЕ ИЛИ, может быть сформирована с помощью более простых логических схем, имеющихся в 1-разрядном процессоре. Целью дальнейшего обсуждения является составление алгоритма выполнения операции ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым любого регистра и содержимым переключающего регистра и записи результата операции в любой регистр. В схеме на рис. 1.13 предусмот-

Таблица 1.2. Операции 1-разрядного процессора

Операция	Управляющие сигналы (активным является высокий уровень сигнала)	Описание
NOP	C_8	Нет операции
LDA	C_4	Загрузка данных с шины в регистр A
LDB	C_5	Загрузка данных с шины в регистр B
ATD	C_1	Пересылка содержимого регистра A на шину данных
BTD	C_2	Пересылка содержимого регистра B на шину данных
STD	C_3	Пересылка содержимого регистра S на шину данных
CMA	C_1C_6	Формирование дополнительного кода содержимого регистра A и его пересылка на шину данных
CMB	C_2C_6	Формирование дополнительного кода содержимого регистра B и его пересылка на шину данных
CMS	C_3C_6	Формирование дополнительного кода содержимого регистра S и его пересылка на шину данных
ANB	$C_1C_2C_7$	Операция (A) И (B) с размещением результата на шине данных
ANS	$C_1C_3C_7$	Операция (A) И (S) с размещением результата на шине данных
ORB	C_1C_2	Операция (A) ИЛИ (B) с размещением результата на шине данных
ORS	C_1C_3	Операция (A) ИЛИ (S) с размещением результата на шине данных

Таблица 1.3. Коды операций и операндов и установка управляющих сигналов

Описание операции	Мнемоническое обозначение	КОП	Управляющие сигналы	Операнд	Код	Управляющие сигналы
Нет операции	NOP	000	C_8	Не используется	00	Не используются
Загрузка регистра А	LDA	001	C_4	(A)	01	S_1
Загрузка регистра В	LBA	010	C_5	(B)	10	S_2
Передача непосредственная	TXD	011	—	(S)	11	S_3
Формирование дополнительного кода	COM	100	C_6			
Логическое И с содержимым регистра А	AND	101	C_1C_7			
Логическое ИЛИ с содержимым регистра А	OR	110	C_1			

рен дополнительный сигнал управления C_8 с целью обеспечения фиксации выходных данных. При высоком уровне сигнала C_8 синхроимпульсы в процессор не поступают, никакой операции не выполняется (точнее говоря, выполняется операция NO OPERATION — НЕТ ОПЕРАЦИИ).

Прежде чем пытаться написать последовательность команд, реализующих логическую функцию ИСКЛЮЧАЮЩЕЕ ИЛИ, составим список всех команд 1-разрядного процессора, изображенного на рис. 1.13 (табл. 1.2). Приведем также коды операций и операндов основных команд (табл. 1.3). Каждая из семи команд закодирована двоичным числом, называемым *кодом операции* (КОП). Заметим, что (А) и (В) не рассматриваются как операнды команды ЗАГРУЗИТЬ РЕГИСТР. Здесь перечислены только те команды, которые мы будет использовать в процессе проектирования.

Запись типичной команды показана ниже. Команда разделена на поля, содержащие группы битов. Команда, представленная в двоичном коде, называется *машинной командой* или *командой машинно-ориентированного языка*. Символическая форма (мнемоническое обозначение) кода операции является записью команды на *языке ассемблера*. Этот язык позволяет программисту составлять последовательности команд (программы) на языке, более естественном для человека. Чтобы было возможно использовать язык ассемблера для написания программ, должен существовать *транслятор* ассемблера — специальная программа,

осуществляющая преобразование символических команд языка ассемблера в команды, представленные на машинном языке, физически передаваемые в процессор для выполнения операций. (Эти вопросы обсуждались более подробно в Кл., 1980.)

	Поле КОП	Поле операнда
СОМ А	100	01

Очевидно, что коды операций присвоены нами определенным командам в основном произвольно, в связи с чем возникает вопрос, как можно подобные двоичные числа использовать для выполнения требуемых операций. Ответ на этот вопрос также очевиден — необходимо декодирование команд, заключающееся в том, что машинная команда, состоящая из кода операции и операндов, передается на вход дешифратора команд, причем на выходе дешифратора формируются необходимые управляющие сигналы. Дешифратор команд, использующий декодирующую СИС типа 1 из 8 с низким уровнем сигнала, соответствующим активному состоянию выходных линий, показан на рис. 1.15. В данной книге значительное внимание уделяется различным способам реализации подобных дешифраторов команд.

Дешифраторы типа 1 из 8 являются устройствами с предварительной установкой исходного состояния. Это означает, что включению устройства всегда предшествует его выключение, т. е. тем самым обеспечивается отсутствие перекрытия команд. Соединив обозначенные соответствующим образом линии сигналов управления (рис. 1.13) с выходными линиями дешифратора (рис. 1.15), получим систему, представленную на рис. 1.16. Необходимо также разработать устройство управления последова-

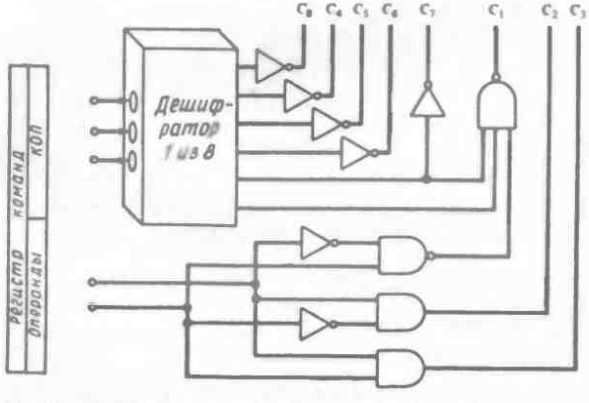


Рис. 1.15. Простейший дешифратор команд 1-разрядного процессора.

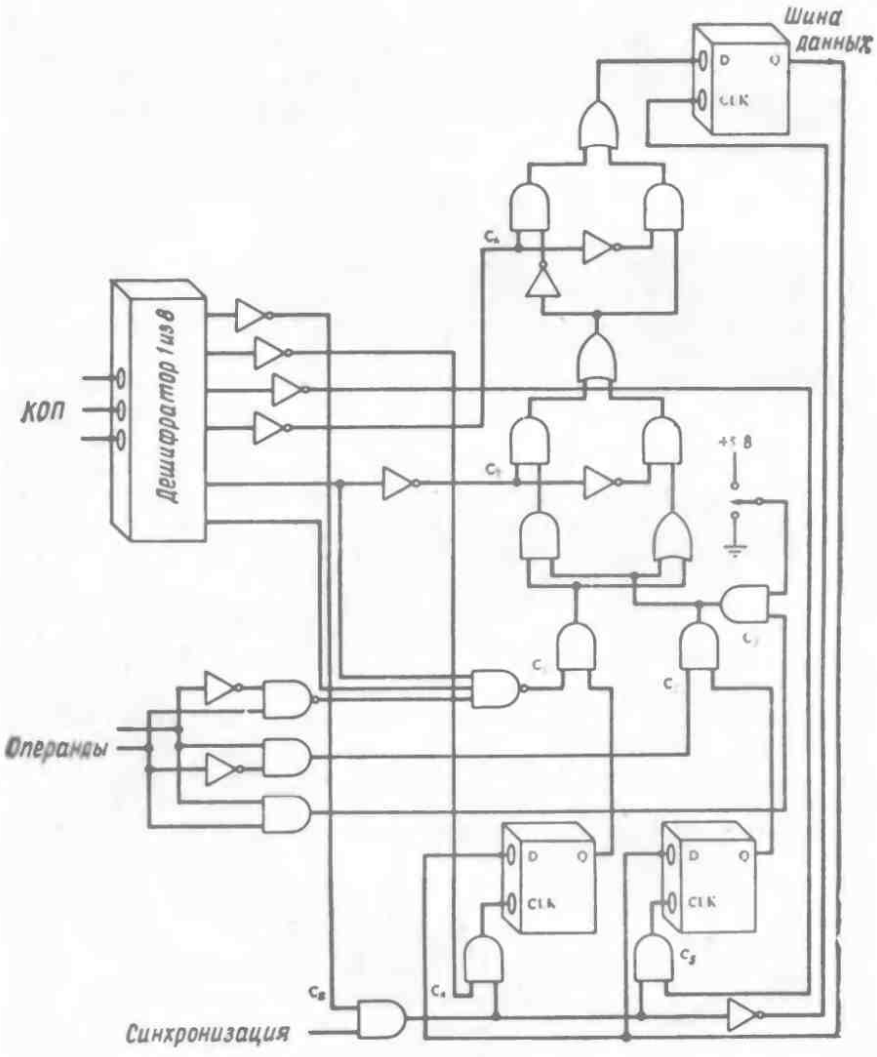


Рис. 1.16. Одноразрядный ЦП, включающий блок регистров памяти, логическое устройство и дешифратор команд.

тельностью выполнения команд, но механизм декодирования команд по существу представляет собой простейший вариант классической схемы декодирования управляющей информации. Реализация дешифратора команд в 1-разрядном процессоре может ввести читателя в заблуждение. Большинство ЭВМ манипулирует словами данных, содержащими 16 бит и более, имеет примерно 12 регистров, базовый набор команд (100 ко-

Таблица 1.4. Таблица управляющих сигналов для набора команд 1-разрядного процессора

Команда на языке ассемблера	Машинный код (КОП)	Управляющие сигналы (активным является высокий уровень сигнала)							
		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
NOP	000 00	0	0	0	0	0	0	0	1
LDA	001 00	0	0	0	1	0	0	0	0
LDB	010 00	0	0	0	0	1	0	0	0
TXD(A)	011 01	1	0	0	0	0	0	0	0
TXD(B)	011 10	0	1	0	0	0	0	0	0
TXD(S)	011 11	0	0	1	0	0	0	0	0
COM(A)	100 01	1	0	0	0	0	1	0	0
COM(B)	100 10	0	1	0	0	0	1	0	0
COM(S)	100 11	0	0	1	0	0	1	0	0
AND(B)	101 10	1	1	0	0	0	0	1	0
AND(S)	101 11	1	0	1	0	0	0	1	0
IOR(B)	110 10	1	1	0	0	0	0	0	0
IOR(S)	110 11	1	0	1	0	0	0	0	0

манд или более), как правило, с возможностью объединения нескольких команд с целью формирования многих сотен других команд. Проектирование логических схем, необходимых для декодирования команд и управления трактами передачи сигналов, становится исключительно сложной и трудоемкой задачей, что на практике сдерживает появление более мощных ЭВМ. В схеме дешифратора другого типа используются данные, приведенные в табл. 1.4. В соответствии с табл. 1.4 дешифратор команд представляет собой простое устройство, на вход которого поступает команда, а на выход — управляющие сигналы. Заметим, что неоднозначность в этом случае отсутствует. Все команды и управляющие слова[1] не повторяются. Следовательно, код операции каждой команды можно интерпретировать как адрес управляющего слова и хранить управляющие слова в постоянном запоминающем устройстве (ПЗУ). Табл. 1.4 построена исходя из 5-битового кода операции команды, которому соответствуют 32 адреса, и управляющих слов, каждое из которых содержит 8 бит. Таким образом, ПЗУ, состоящее из 32 слов по 8 бит, в которое записана надлежащая информация, может использоваться в качестве дешифратора команд. Построенный таким образом блок дешифратора команд изображен на рис. 1.17. Появление на рынке дешевых ПЗУ вызвало глубокие изменения в технике декодирования команд. Этот вопрос будет рассмотрен в гл. 5.

[1] Под управляющим словом здесь понимают совокупность значений управляющих сигналов, обеспечивающих выполнение операции, задаваемой командой. — *Прим. перев.*

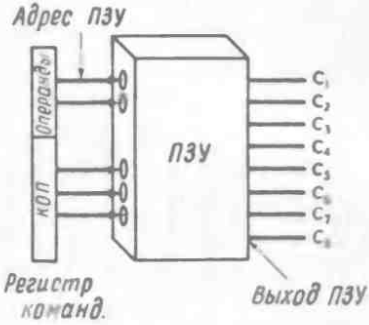


Рис. 1.17. Дешифратор команд на ПЗУ, соответствующий табл. 1.4.

АЛГОРИТМЫ РАБОТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ может быть реализована путем выполнения процессором в определенной последовательности операций И, ИЛИ, НЕ. Обратившись к таблицам истинности, приведенным на рис. 1.18, можно убедиться, что операцию ИСКЛЮЧАЮЩЕЕ ИЛИ можно представить через другие логические операции следующим образом:

$$A \otimes B = \overline{A} \cdot B + A \cdot \overline{B}$$

Такое представление определяет тем самым и способ реализации операции исходя из возможностей аппаратных средств. Предположим, необходимо выполнить операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым переключающего регистра и содержимым регистра А и поместить результат операции в регистр В. Последовательность команд процессора, используемых для выполнения подобной процедуры, называют *программой*. В данном случае программа выполняет алгоритм «вычисления» функции ИСКЛЮЧАЮЩЕЕ ИЛИ. В большинстве случаев в программах реализуются более сложные алгоритмы обработки данных.

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

A	$\overline{A}$
0	1
1	0

A	B	A⊗B
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 1.18. Условное изображение и таблицы истинности основных логических операций.

Перепишем алгоритм вычисления функции ИСКЛЮЧАЮЩЕЕ ИЛИ, используя скобки для указания последовательности выполнения операций и символ S для обозначения содержимого переключающего регистра:

$$A \otimes S = (((\overline{A}) \cdot S) + (A \cdot (\overline{S})))$$

Сначала выполняются операции наиболее глубокого уровня вложения, затем операции следующего уровня и т. д. до тех пор, пока не будут раскрыты все скобки выражения. Если две операции находятся на одном уровне вложения, выбор последовательности их выполнения может быть сделан произвольно либо следует учитывать другие факторы. Поскольку в процессе вычислений появляются промежуточные результаты, необходимо предусмотреть временную память для их размещения. Регистр B может выполнять роль такой временной памяти.

При определении порядка следования операций необходимо установить, какая из операций $\overline{A} \cdot S$ или $A \cdot \overline{S}$ должна выполняться первой. Начнем с операции $A \cdot \overline{S}$, сохраняя тем самым неизменным содержимое регистра A. В дальнейшем при выполнении операции $\overline{A} \cdot S$ регистр A освобождается и может использоваться в качестве регистра временной памяти. Это обусловлено тем обстоятельством, что в переключающий регистр не может загружаться информация, поступающая с шины данных, и, следовательно, этот регистр не может использоваться в качестве временной памяти для хранения промежуточных результатов. Если бы этого ограничения не существовало, решение задачи носило бы «симметричный» характер, т. е. нельзя было бы отдать предпочтение выполнению первой какой-либо из этих двух операций. Итак, для вычисления выражения

$$A \otimes S = \overline{A} \cdot S + A \cdot \overline{S}$$

требуется следующая последовательность операций:

$S \rightarrow \overline{S}$	$\rightarrow (B)$	A = исходное содержимое регистра A
$(B) \cdot A \rightarrow A \cdot \overline{S}$	$\rightarrow (B)$	(A) = временное (с промежуточными результатами) содержимое регистра A
$A \rightarrow \overline{A}$	$\rightarrow (A)$	
$(A) \cdot S \rightarrow \overline{A} \cdot S$	$\rightarrow (A)$	
$(A) + (B) \rightarrow \overline{A} \cdot S + A \cdot \overline{S}$	$\rightarrow (B)$	

Команды, реализующие данную последовательность операций, приведены в табл. 1.5. Необходимо также предусмотреть средства последовательной передачи команд на вход ЦП. На рис. 1.19 показана схема последовательной выборки команд из

Таблица 1.5. Программа вычисления функции ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым регистра А и переключающего регистра

Мнемоническое обозначение команды	Операнд	Результат операции
COM	S	$S \rightarrow \overline{S}$
LDB		$\overline{S} \rightarrow (B)$
AND	B	$A \cdot \overline{S}$
LDB		$A \cdot \overline{S} \rightarrow (B)$
COM	A	$A \rightarrow \overline{A}$
LDA		$\overline{A} \rightarrow (A)$
AND	S	$\overline{A} \cdot S$
LDA		$\overline{A} \cdot S \rightarrow (A)$
IOR	B	$\overline{A} \cdot S + A \cdot \overline{S}$
LDB		$A \times S \rightarrow (B)$

памяти. Мы опускаем проблему инициирования выполнения последовательности команд, полагая, что это выполнение начнется с надлежащей команды.

ИЗМЕНЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ВЫПОЛНЕНИЯ КОМАНД

Заметим, что в нашем процессоре не были предусмотрены средства для управления ходом выполнения программы, позволяющие изменять последовательность выполнения команд. Поскольку эффективность работы ЭВМ общего назначения определяется главным образом именно этой возможностью, мы обсудим средства, с помощью которых данную возможность можно реализовать.

Изменение последовательности выполнения команд программным путем организуется путем проверки и обнаружения выполнения определенных условий. Для этой цели используются два типа команд: *условного пропуска* (SKP) и *условного перехода* (JMP) (рис. 1.20). Обсудим кратко каждую из них.

Команда условного пропуска. Данная команда обычно выполняется следующим образом: проверяется наличие одного из двух возможных результатов предыдущей обработки данных. При обнаружении первого результата выполняется очередная команда; при обнаружении второго результата очередная команда пропускается и выполняется следующая команда. Пропускаемая команда обычно представляет собой команду безусловного перехода.

Команда условного перехода. Выполняется проверка одного из двух возможных результатов предыдущей обработки данных. Обнаружение одного из результатов приводит к продолжению выполнения нормальной последовательности команд. При обна-

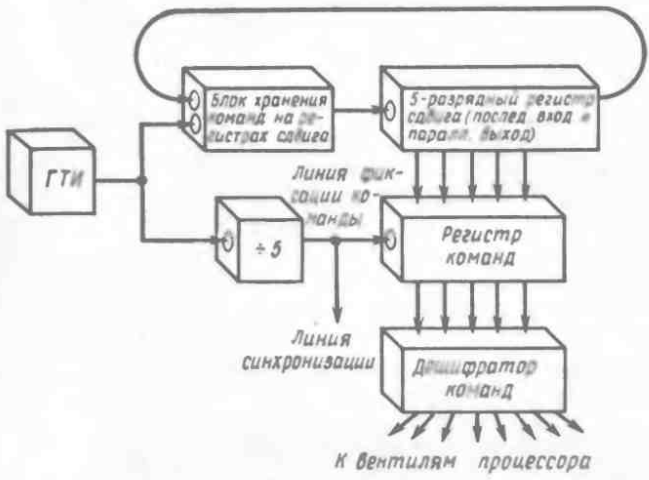


Рис. 1.19. Схема последовательной выборки команд.

ружении другого результата осуществляется переход к новой ячейке памяти, содержащей другую команду. С этой команды продолжается последовательное выполнение команд. Заметим, что команда условного перехода должна содержать адрес перехода, или должны быть предусмотрены определенные средства определения местоположения новой последовательности команд.

В качестве примера рассмотрим проверку равенства содержимого переключающего регистра содержимому регистра А, которая состоит в следующем:

выполнить операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым регистров А и S. Если содержимое этих двух регистров одинаково, то результатом операции, записываемым в регистр В, будет 0, в противном случае результат равен 1. Теперь необходима команда, которая проверяла бы содержимое регистра В и в зависимости от результата проверки выполняла или не выполняла переход по определенному адресу (или пропуск

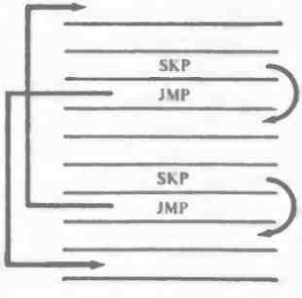


Рис. 1.20. Последовательность выполнения команд, включающая нормальную выборку следующей команды, а также пропуски команд и условные переходы.

Содержимое первого 2-битового поля КОП модифицированного набора команд

JMP	Команда, выполняемая после данной
00	Следующая
01	Расположенная по адресу, задаваемому 5-битовым полем адреса перехода команды условного перехода
1X	Расположенная по адресу, задаваемому 6-битовым полем адреса перехода команды безусловного перехода

Рис. 1.21. Формат команды модифицированного набора команд, включающий поле указания команды перехода и адрес перехода.

команды). Добавлением этой возможности завершается проектирование нашего 1-разрядного процессора.

Чтобы использовать команду пропуска нетривиальным способом, требуется команда безусловного перехода (пропуск которой обычно и организуется). Поэтому, вместо того чтобы разрабатывать обе команды — пропуска и перехода, мы реализуем только возможности команды перехода, причем в дополнение к команде условного перехода включаем в набор команд процессора также и команду безусловного перехода.

Один из простейших возможных способов модификации описанного выше набора команд с целью включения в него двух команд перехода заключается в добавлении 2 бит к полю кода операции. Назовем эти биты *полем указания команды перехода* (JMP). Обычно подобное увеличение длины команды является нежелательным. Однако в данном случае использование коротких команд представляет определенную проблему. Команда перехода должна содержать поле адреса перехода, указывающее адрес команды, которая должна выполняться следующей. Поэтому использование небольшого количества битов обычной 5-битовой команды для представления адреса перехода сводит к минимуму диапазон действия команды перехода[1]). Полученный в результате данной модификации не вполне обычный формат машинной команды показан на рис. 1.21.

Модификация, связанная с обеспечением нового 7-битового формата команд, состояла в расширении регистра сдвига с последовательным входом и параллельными выходами с 5 до 7 разрядов. Формирование синхроимпульсов происходит теперь путем деления частоты генератора тактовых импульсов на 7,

[1]) Имеется в виду диапазон изменения адреса перехода, задаваемого в команде перехода. — *Прим. перев.*

а не на 5. Информация в регистре команд фиксируется при поступлении заднего фронта седьмого по счету синхроимпульса. При этом производится задержка таким образом, что, во-первых, сигналы на параллельных выходах регистра сдвига успевают стабилизироваться, а, во-вторых, информация в регистрах ЦП оказывается зафиксированной не задержанными синхроимпульсами, и результаты выполнения предыдущей команды могут использоваться схемой управления. При передаче команды перехода в регистр команд биты поля JMP кода операции обеспечивают (будучи ненулевыми) передачу управления новой последовательности команд, если выполняется безусловный переход (JUN) либо условный переход (JMP), и результат проверки условия (C) имеет значение «истинно». Это условие может быть представлено в виде

$$JUN + (JMP \cdot C) = J$$

Здесь C — проверяемый бит условия.

Если значение J истинно, содержимое поля адреса перехода команды перехода записывается в регистр адреса перехода. При выполнении условного перехода могут использоваться только 5 бит адреса, тогда как при выполнении безусловного перехода адрес содержит 6 бит, обеспечивая тем самым диапазон действия команды перехода 0—64 (64 — предельное значение адреса перехода).

Адрес перехода помещается в компаратор для сравнения с содержимым счетчика адреса, реализованного на регистре. Передача команд и синхроимпульсов в ЦП блокируется командой перехода за счет флажка J, выполняющего сброс триггера установки. Этот триггер блокирует указанную передачу до тех пор, пока не появится соответствующий сигнал на выходе компаратора, что в свою очередь произойдет только тогда, когда на выходе счетчика адреса появится надлежащий адрес. Требуемая команда передается в регистр команд, который запирается задержанным синхроимпульсом, передача которого только что была разрешена в результате появления сигнала на выходе триггера установки. Задержка, вызванная прохождением сигналов через 6-разрядный счетчик, компаратор, триггер установки и несколько инверторов, определяет параметр M — *временной допуск* (погрешность) системы. Этот параметр должен иметь такое значение, чтобы регистр сдвига не успел начать сдвиг текущей команды в 7-разрядном регистре до момента ее запирания задержанным синхроимпульсом. Требование, чтобы регистр сдвига выполнял операцию сдвига для каждой команды, находящейся между командой перехода и адресуемой командой, является основным недостатком последовательной организации хранения команд в памяти.

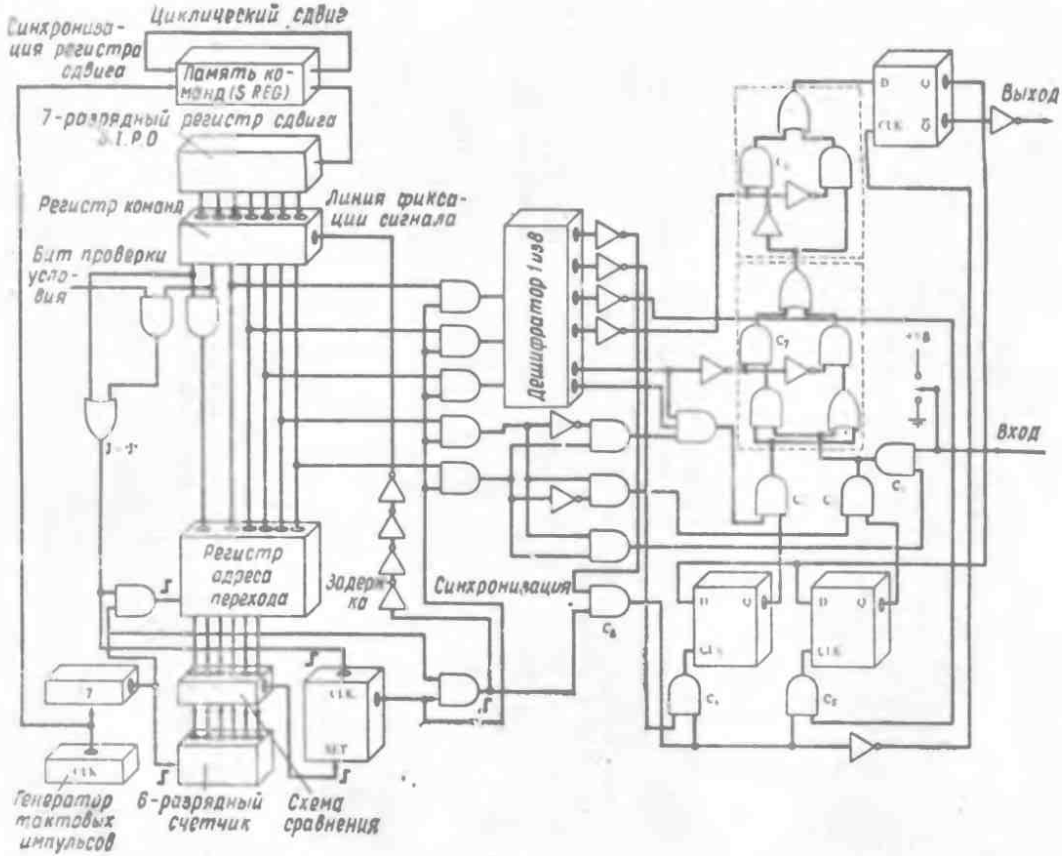


Рис. 1.22. Полная схема 1-разрядной ЭВМ.

ЗАКЛЮЧЕНИЕ

Технология цифровой обработки данных, базирующаяся на представлении данных в двоичной системе счисления и булевой логике, интенсивно развивается. Логические схемы, первоначально реализуемые с помощью механических реле, а затем электронно-вакуумных приборов и транзисторов, в настоящее время изготавливаются с использованием вентилей, количество которых превышает миллион на одно устройство. Эта глава представляет собой краткое введение в структуру подсистем обработки данных, изложенное на примере проектирования 1-разрядной ЭВМ (ее полная схема показана на рис. 1.22). Анализ структуры проводился на уровне описания работы отдельных вентилей. В гл. 2 базовая структура расширяется, причем рассмотрение ведется на уровне регистров. Принципы управления, о которых речь шла в данной главе, будут обсуждены более подробно в гл. 5.

ГЛАВА 2

ПРОЦЕССОРНЫЕ ЭЛЕМЕНТЫ

Одноразрядный процессор, описанный в гл. 1, содержит все основные компоненты, необходимые для обработки данных: функциональный блок[1], имеющий по крайней мере два входных канала; выходной канал, располагающий набором ячеек или регистров памяти для записи результатов выполнения операций; тракты передачи данных из этих регистров на вход функционального блока; схему синхронизации работы устройства. С помощью имеющихся в распоряжении разработчика дополнительных, ранее не используемых блоков возможности процессора можно расширить, доведя разрядность обрабатываемых данных до четырех. Основным следствием перехода к 4-разрядному процессору является выполнение значительно большего числа операций с двумя 4-битовыми словами данных, чем это можно было сделать с двумя 1-битовыми словами. Базовая схема 4-разрядного процессора показана на рис. 2.1.

Для выполнения операций над 4-битовыми словами данных хорошо подходит арифметическо-логическое устройство 74181 фирмы Texas Instruments, описанное ранее (Кл., 1980). Базовая структура устройства может быть расширена путем увеличения числа фиксаторов выходных данных: вместо простого 4-разрядного фиксатора, обеспечивающего сохранение текущих данных на выходе АЛУ, может использоваться подсистема памяти для хранения нескольких слов — *внутренний блок регистров ЦП*. Использование 16 регистров в свою очередь требует включения четырех управляющих линий выбора регистра, с помощью которых определяют регистр, в который должны быть загружены данные с выхода АЛУ. Предусматривается также инвертирование синхроимпульсов с целью предотвращения нарушения синхронизации передачи сигналов во времени («гонок» сигналов), что позволяет осуществлять извлечение данных из

[1] Имеется в виду блок, непосредственно выполняющий операции по обработке данных. — *Прим. перев.*

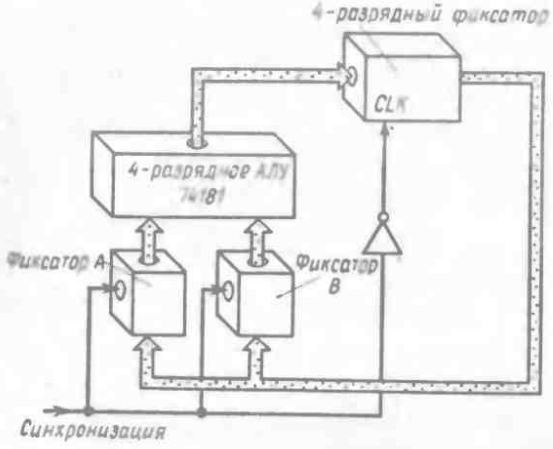


Рис. 2.1. Структура 4-разрядного процессора.

блока регистров, их модификацию и повторную загрузку в блок регистров.

При высоком уровне сигнала на линии синхронизации на выходы фиксаторов А и В передаются данные с их входов, причем данные на выходах фиксируются (сохраняются) и после того, как на линии синхронизации будет восстановлен низкий уровень сигнала.

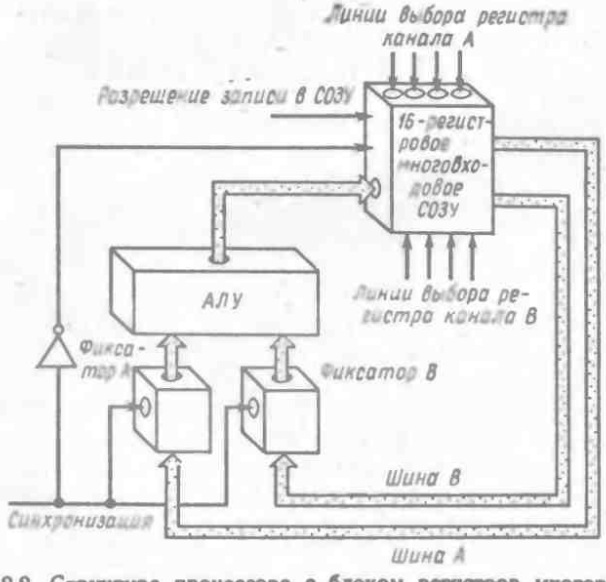


Рис. 2.2. Структура процессора с блоком регистров многовходового СОЗУ.

Дальнейшее развитие структуры процессора состоит в использовании многовходового регистрового сверхоперативного запоминающего устройства (СОЗУ), позволяющего одновременно считывать из блока регистров два слова данных. Последнее требует подключения к такому 16-регистровому СОЗУ еще четырех управляющих линий выбора регистра. Разработка логической структуры процессора завершается включением двух 4-разрядных шин данных, по которым можно одновременно загружать одно слово данных, выбираемое из блока регистров, в фиксатор А, а другое — в фиксатор В (рис. 2.2). Хотя многовходовое СОЗУ позволяет производить доступ к двум регистрам[1])

Рис. 2.3. Временна́я диаграмма загрузки регистров СОЗУ.

Загрузка фиксаторов А и В содержимым регистра СОЗУ, выбираемого соответственно сигналами линий управления А и В, производится при высоком уровне напряжения на линии синхронизации; загрузка регистра СОЗУ, выбираемого сигналами линий управления В, данными, поступающими из выходного порта АЛУ, производится при низком уровне напряжения на линии синхронизации и поступлении сигнала разрешения записи в СОЗУ.

в то время, когда осуществляется загрузка третьего, мы будем избегать такого большого числа линий управления, полагая, что сигналы управляющих линий В будут определять как регистр СОЗУ, в который загружаются данные, так и регистр, содержимое которого передается на шину данных В. Будем также считать, что для записи данных в регистр, выбираемый сигналами управления линий В, в СОЗУ должен быть передан сигнал разрешения, тогда как чтение данных из этого регистра может производиться в любой момент времени. Таким образом, линии управления В обеспечивают загрузку и считывание данных из СОЗУ, а линии управления А — только считывание данных из регистра СОЗУ на шину данных. Временна́я диаграмма, изображенная на рис. 2.3, построена в предположении, что в СОЗУ поступил сигнал разрешения и в него может записываться информация. Хотя предполагается, что данные могут выводиться из процессора благодаря их фиксации на выходных шинах, здесь мы имеем замкнутую систему, в которой отсутствуют средства многократного ввода данных. В 1-разрядном процессоре переключение источника входных данных АЛУ позволило выбирать в качестве входных данных либо выходные данные фиксатора В, либо данные, поступающие из переключающего регистра. Такой выбор одного из двух битов представ-

[1]) Для передачи их содержимого на шины данных. — *Прим. перев.*

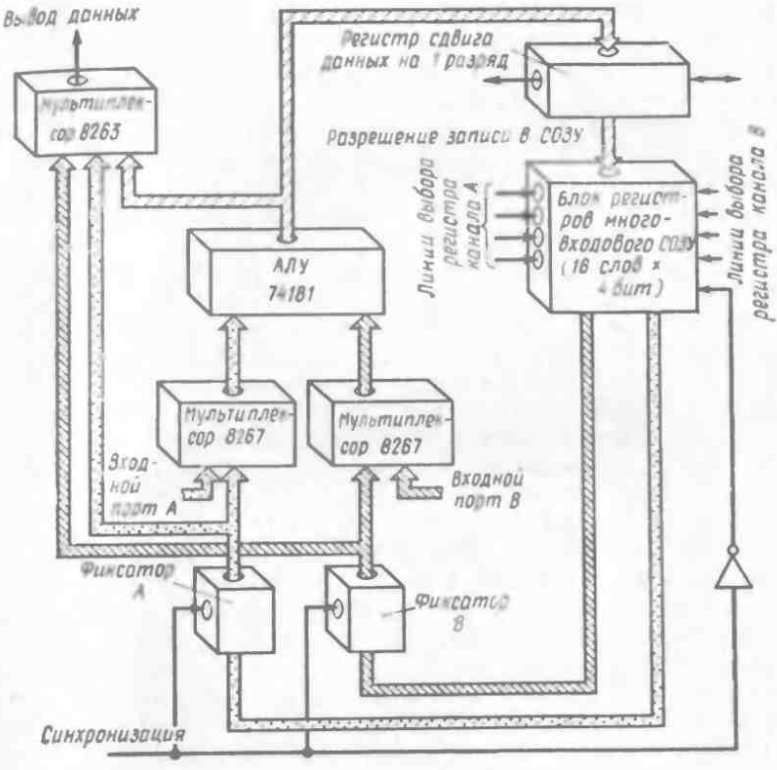


Рис. 2.4. Структура процессора с мультиплексорами в трактах передачи данных.

ляет собой простейшую форму мультиплексирования. Следовательно, чтобы получить аналогичный эффект при работе с 4-битовыми данными, можно включить 4-разрядный мультиплексор типа 8267 фирмы Signetics. Поскольку нет особых причин для оказания предпочтения входному порту В АЛУ, другой такой же мультиплексор (типа 8267) следует поместить на входе порта А. Кроме того, на выход устройства будут передаваться не просто данные с шин А и В, а данные, поступающие либо с выхода АЛУ, либо с выходов фиксаторов А или В. Это легко реализуется путем подачи сигналов с этих трех выходов в 4-разрядный мультиплексор 8263 с тремя входами. Полученная в результате структура процессора показана на рис. 2.4. Здесь устройство формирования дополнительного кода, расположенное в 1-разрядном процессоре на выходе логического устройства, заменено 4-разрядным регистром сдвига, позволяющим сдвигать 4-битовое слово данных на один разряд влево или вправо.

Таким образом, выходные данные АЛУ перед их загрузкой в СОЗУ могут сдвигаться на 1 разряд влево или вправо. В данной системе СОЗУ используется в качестве временной памяти и его не следует путать с основной памятью, в которой размещаются команды и данные программы. Внутренний блок регистров, связанный с АЛУ, используется для хранения операндов и результатов, относящихся к текущей операции, выполняемой АЛУ, и его часто называют *сверхоперативной памятью*.

Перейдя к обработке 4-битовых слов данных и используя симметричным образом стандартные 4-разрядные блоки, мы расширили структуру нашего исходного 1-разрядного устройства. По причине, которая вскоре станет понятной, нарушим теперь симметрию структуры путем включения в нее специальной схемы. Модификация состоит в добавлении канала передачи данных с выхода АЛУ на его вход. Чтобы это сделать, необходимо использовать определенный входной порт АЛУ. Для этой цели мы произвольно выберем входной порт B. Новый канал будет включать еще один 4-разрядный регистр, называемый регистром Q, а также регистр сдвига, с помощью которого осуществляется сдвиг на 1 разряд данных, поступающих с выхода АЛУ, перед их загрузкой в регистр Q. Кроме того, предусматривается обратная связь для передачи данных с выхода регистра Q на вход регистра сдвига данного канала. Однако для реализации этой обратной связи необходимо включить в данный тракт передачи данных еще один 4-разрядный мультиплексор (MUX) с двумя входами. Полученная в результате структура процессора показана на рис. 2.5. Выходные цепи процессора на рисунке не показаны, а добавленные в схему блоки заключены в прямоугольник, изображенный штриховой линией.

Выбор в качестве АЛУ устройства 74181 с входом и выходом цепи переноса позволяет обрабатывать данные с длиной слова, кратной 4. На схеме процессора на рис. 2.6 на эту возможность указывают каналы ввода-вывода сигналов, расположенные с обеих сторон регистра сдвига, размещенного перед регистром Q. Таким образом, построенная ранее структура процессора может быть расширена для обеспечения обработки данных с любой длиной слова, кратной 4. Другими выходами АЛУ являются выходы признаков нулевого результата операции, переполнения, генерирования и передачи сигналов переноса. Последние используются вместе со схемой ускоренного переноса (carry-look-ahead device) для повышения скорости счета с переносом в системе с разрядностью обрабатываемых слов данных, равной $4 \times n$. Такой схемой ускоренного переноса является устройство 74182, детально описываемое в документации фирмы-изготовителя. Ускоренный перенос используется только для повышения быстродействия, не являясь функционально необходимым, и далее обсуждению не подлежит. На схеме рис. 2.6

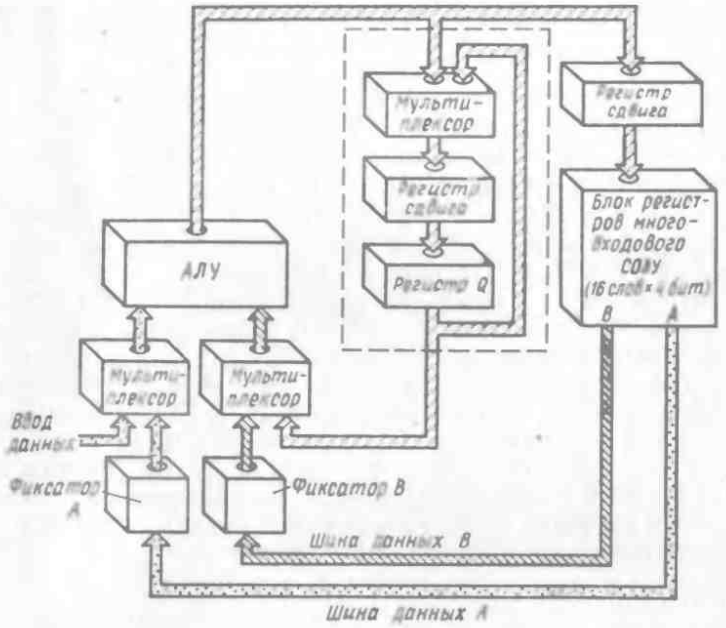


Рис. 2.5. Добавление в ЦП канала обратной связи с регистром сдвига.

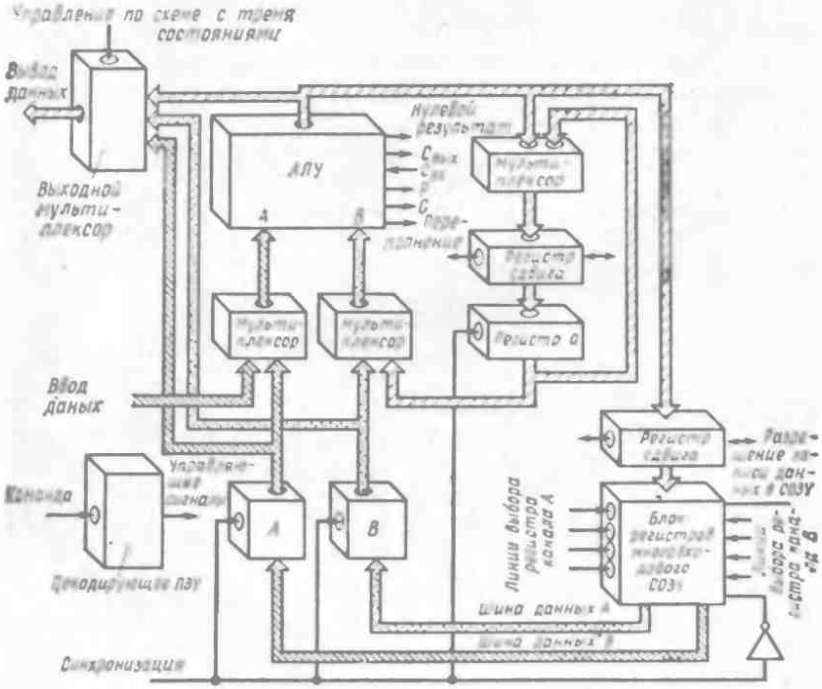


Рис. 2.6. Структура процессора с разрядностью обрабатываемых слов, кратной 4.

изображен выходной порт, а также линия, управляющая работой порта по схеме с тремя состояниями.

Как мог догадаться читатель, на рис. 2.6 представлена реальная интегральная схема — это схема микропроцессора 6701 фирмы Monolithis Memories. Микропроцессор выполнен на БИС с транзисторно-транзисторной логикой (ТТЛ), заменяющей собой 1000 вентилей. Для реализации подобной схемы на СИС потребовалось бы 25 корпусов СИС с ТТЛ при обеспечении потребляемой мощности ~7 Вт, тогда как потребление мощности микропроцессором 6701 составляет всего лишь 0,9 Вт. Устройство заключено в корпус с 40 выводами, занимает меньшую площадь, чем эквивалентные СИС с ТТЛ, на 194 см² и имеет на 375 выводов меньше. Последний фактор имеет очень большое значение, поскольку соединения в этих местах представляют собой самые ненадежные (подверженные разрушению) точки системы. Резкое сокращение количества выводов существенно повышает надежность системы.

ПРОЦЕССОРНЫЙ ЭЛЕМЕНТ Am2901 ФИРМЫ ADVANCED MICRO DEVICES

ОСНОВНОЕ УПРАВЛЕНИЕ

Несмотря на то что описанные в предыдущем разделе процессорные элементы (ПЭ) 6701 фирмы Monolithic Memories были разработаны раньше, стандартом для промышленности является 4-разрядный ПЭ Am2901, созданный фирмой Advanced Micro Devices (рис. 2.7).

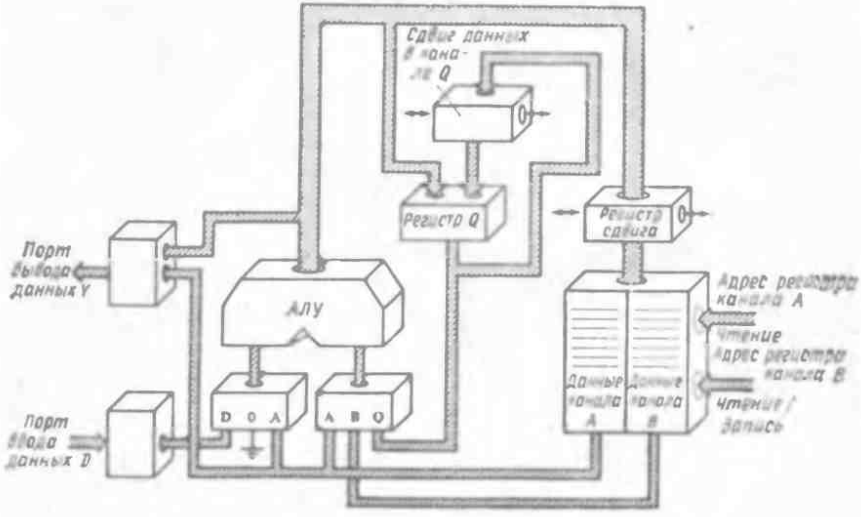


Рис. 2.7. Структура ПЭ Am2901.

Таблица 2.1. Основные операции, выполняемые АЛУ устройства Am2901

Управляющий код	Операция	Символическое обозначение
0	R плюс S	R+S
1	S минус R	S—R
2	R минус S	R—S
3	R ИЛИ S	$R \vee S$
4	R И S	$R \wedge S$
5	$\overline{R}$ И S	$\overline{R} \wedge S$
6	R ИСКЛЮЧАЮЩЕЕ ИЛИ S	$R \forall S$
7	R ИСКЛЮЧАЮЩЕЕ НЕ-ИЛИ S	$\overline{R \forall S}$

Таблица 2.2. Управление выбором источников операндов АЛУ устройства Am2901

Управляющий код	Источники операндов АЛУ	
	R	S
0	A	Q
1	A	B
2	0	Q
3	0	B
4	0	A
5	D	A
6	D	Q
7	D	0

Для управления работой ПЭ Am2901 используются следующие управляющие линии:

1. *Линии выбора регистра канала А.* Сигналы этих четырех адресных линий обеспечивают выбор в 16-регистровом СОЗУ регистра, содержимое которого будет передано на шину А при появлении на входной линии синхронизации высокого уровня сигнала.

Таблица 2.3. Управление выбором адресата АЛУ устройства Am2901

Управляющий код	Операция СОЗУ		Операция регистра Q		Выход Y	Регистр сдвига СОЗУ		Регистр сдвига Q	
	Сдвиг	Загрузка	Сдвиг	Загрузка		СОЗУ$_0$	СОЗУ$_3$	Q_0	Q_3
0	X	Не производится	Не производится	F[1]→Q	F	X	X	X	X
1	X	То же	X	Не производится	F	X	X	X	X
2	Не производится	F→B	X	То же	A	X	X	X	X
3	То же	F→B	X	» »	F	X	X	X	X
4	Вправо	F/2→B	Вправо	Q/2→Q	F	F_0	D_3	Q_0	D_3
5	»	F/2→B	X	Не производится	F	F_0	D_3	Q_0	X
6	Влево	2F→B	Влево	2Q→Q	F	D_0	F_3	D_0	Q_3
7	»	2F→B	X	Не производится	F	D_0	F_3	X	Q_3

[1]) F — выход АЛУ.

2. *Линии выбора регистра канала В.* Сигналы указанных четырех адресных линий обеспечивают выбор в 16-регистровом СОЗУ регистра, содержимое которого будет передано на шину В при появлении на входной линии синхронизации высокого уровня сигнала. Входные данные СОЗУ загружаются в выбранный с помощью этих линий регистр СОЗУ во время прохождения заднего фронта синхроимпульсов.

3. *Линии указания операции АЛУ.* Сигналы этих трех линий задают одну из восьми возможных арифметических или логических операций, выполняемых АЛУ. Для модификации операции может использоваться линия ввода сигнала переноса C_n. Основные операции, выполняемые АЛУ, приведены в табл. 2.1.

4. *Линии указания источника операндов АЛУ.* Сигналы этих трех линий задают источники операндов, обрабатываемых в АЛУ (табл. 2.2).

5. *Линии указания места размещения результата операции (адресата) АЛУ.* Сигналы этих трех линий задают адресат результата выполнения операции в АЛУ (табл. 2.3).

Микрокоманда ПЭ 2901 содержит следующие поля, определяющие основные управляющие сигналы:

БИТЫ СОСТОЯНИЯ И ДОПОЛНИТЕЛЬНОЕ УПРАВЛЕНИЕ

В добавление к основным управляющим сигналам в ПЭ Am2901 используется еще ряд дополнительных линий управления.

1. *Линия ввода синхроимпульсов.* Эта линия используется для управления фиксацией и внутренней передачей данных через вентили устройства. При низком уровне сигнала на линии синхронизации данные, поступающие на вход блока регистров, записываются в регистр, определяемый сигналами линии выбора регистра канала В (при наличии сигнала разрешения записи). Предыдущее содержимое регистра сохраняется во вспомогательных фиксаторах на выходах СОЗУ, что обеспечивает взаимную синхронизацию выполнения основных и вспомогательных операций со стеком регистров.

1 — Данные передаются в регистр Q. Фиксаторы A и B открыты — происходит запись в фиксаторы содержимого выбранных регистров СОЗУ.
2 — Фиксаторы A и B заперты — в них сохраняются данные, появившиеся последними на выходе СОЗУ. Новые данные записываются в регистр, определяемый сигналами линии выбора регистра канала B.
3 — Загружается регистр Q. Данные фиксируются в регистре СОЗУ, определяемом сигналами линии выбора регистра канала B. В фиксаторы A и B передаются данные с выхода СОЗУ.

2. *Линии входа и выхода цепи переноса* (C_n и C_{n+4}). Эти линии обычно используются при выполнении арифметических операций АЛУ. Так, управление вводом бита переноса позволяет осуществить приращение содержимого определенного регистра при выполнении определенных операций. Управление выводом бита переноса производится обычным образом.

3. *Линия признака состояния выхода АЛУ* F. Сигнал этой линии показывает, равен или не равен нулю результат выполнения текущей операции в АЛУ.

4. *Линии вывода сигналов ускоренного переноса* P и G могут использоваться вместе со схемой ускоренного переноса так, как это описывается в следующем разделе.

5. *Линии* Q_0 *и* Q_3 (самого младшего и самого старшего по значимости битов регистра Q) служат для выполнения операций сдвига в регистре Q (см. табл. 2.3).

6. *Линии* $СОЗУ_0$ *и* $СОЗУ_3$ (самого младшего и самого старшего по значимости битов регистра сдвига СОЗУ) используются при выполнении операции сдвига в регистре сдвига СОЗУ (см. табл. 2.3).

7. *Линия признака переполнения OVR.* Сигнал этой линии может проверяться при выполнении любой арифметической операции.

Основным назначением регистра Q ПЭ Am2901 (так же, как и устройства 6701) является реализация алгоритмов операций умножения и деления, хотя в ряде случаев этот регистр может использоваться в качестве дополнительного аккумулятора.

ОПЕРАЦИИ И ИСТОЧНИКИ ОПЕРАНДОВ АЛУ

В документации на ПЭ Am2901 описываются все возможные источники операндов для каждой операции, выполняемой АЛУ. Для пользователя более удобными могут оказаться табл. 2.4 и 2.5, в которых операции АЛУ сгруппированы по основным

Таблица 2.4. Логические операции устройства Am2901

КОП (восьмеричное представление) I_{543}	I_{210}	Название группы операций	Операция
4	0	И	$A \wedge Q$
4	1		$A \wedge B$
4	5		$D \wedge A$
4	6		$D \wedge Q$
3	0	ИЛИ	$A \vee Q$
3	1		$A \vee B$
3	5		$D \vee A$
3	6		$D \vee Q$
6	0	ИСКЛЮЧАЮЩЕЕ ИЛИ	$A \veebar Q$
6	1		$A \veebar B$
6	5		$D \veebar A$
6	6		$D \veebar Q$
7	0	ИСКЛЮЧАЮЩЕЕ НЕ-ИЛИ	$\overline{A \veebar Q}$
7	1		$\overline{A \veebar B}$
7	5		$\overline{D \veebar A}$
7	6		$\overline{D \veebar Q}$
7	2	Инвертирование	$\overline{Q}$
7	3		$\overline{B}$
7	4		$\overline{A}$
7	7		$\overline{D}$
6	2	Передача	Q
6	3		B
6	4		A
6	7		D
3	2	Передача	Q
3	3		B
3	4		A
3	7		D
4	2	Запись нуля	0
4	3		0
4	4		0
4	7		0
5	0	Маскирование	$\overline{A} \wedge Q$
5	1		$\overline{A} \wedge B$
5	5		$\overline{D} \wedge A$
5	6		$\overline{D} \wedge Q$

Таблица 2.5. Арифметические операции устройства Am2901

КОП (восьмеричное представление) I_{345}	I_{210}	$C_n=0$ (низкий уровень сигнала) Название группы операций	Операция	$C_n=1$ (высокий уровень сигнала) Название группы операций	Операция
0	0	Сложение	A+Q	Сложение+1	A+Q+1
0	1		A+B		A+B+1
0	5		D+A		D+A+1
0	6		D+Q		D+Q+1
0	2	Передача	Q	Увеличение на 1	Q+1
0	3		B		B+1
0	4		A		A+1
0	7		D		D+1
1	2	Уменьшение на 1	Q—1	Передача	Q
1	3		B—1		B
1	4		A—1		A
2	7		D—1		D
2	2	Формирование обратного кода	—Q—1	Формирование дополнительного кода	—Q
2	3		—B—1		—B
2	4		—A—1		—A
1	7		—D—1		—D
1	0	Вычитание в обратном коде	Q—A—1	Вычитание в дополнительном коде	Q—A
1	1		B—A—1		B—A
1	5		A—D—1		A—D
1	6		Q—D—1		Q—D
2	0		A—Q—1		A—Q
2	1		A—B—1		A—B
2	5		D—A—1		D—A
2	6		D—Q—1		D—Q

функциям обработки данных. Заметим, что сигналы линии входа цепи переноса — C_n при выполнении логических операций не используются.

ПРОЦЕССОРНЫЙ ЭЛЕМЕНТ Am2903 ФИРМЫ ADVANCED MICRO DEVICES

Примерно четыре года спустя после создания ПЭ Am2901 был разработан его усовершенствованный вариант Am2903. Незначительные изменения, произведенные в архитектуре, позволяют неограниченно расширять набор регистров при сохранении двухадресной системы команд и возможности выполнения операций сдвига данных. Был модифицирован дешифратор команд с целью включения операций умножения со знаком и

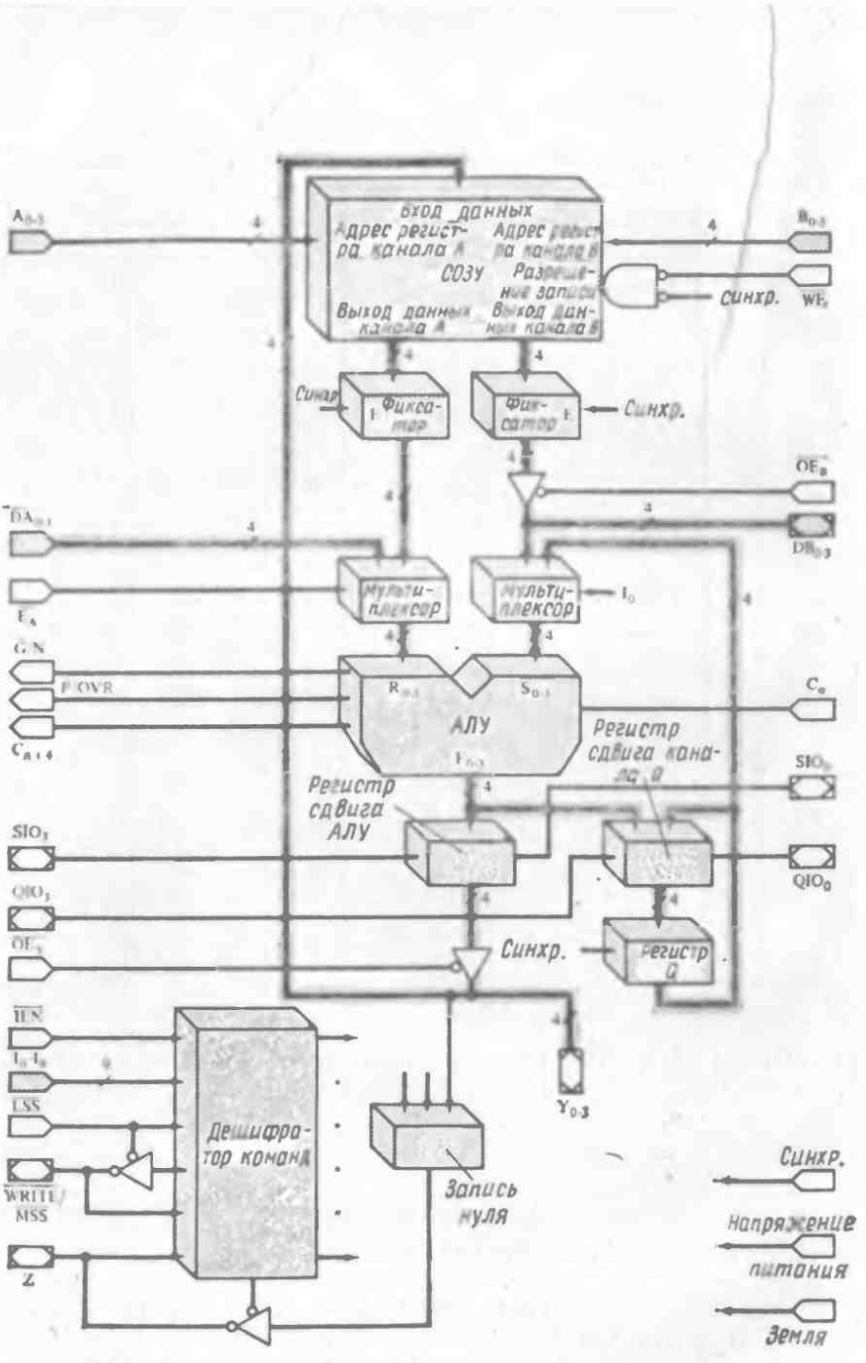


Рис. 2.8. Структура ПЭ Am2903 (*С разрешения* Advanced Micro Devices.)

без знака, деления со знаком с использованием алгоритма без восстановления остатка и нормализации. Введение последней операции позволяет выполнять операции над числами с плавающей точкой. Эти операции выполняются за несколько тактов работы устройства с использованием флажков состояния для указания на завершение операции.

Процессорный элемент Am2903 (рис. 2.8), так же как и ПЭ Am2901, располагает двухвходовым СОЗУ с возможностью одновременного использования обоих входных портов. Данные со входа передаются в выходные фиксаторы СОЗУ при высоком уровне сигнала линии синхронизации и сохраняются в фиксаторах после того, как этот уровень становится низким. Возможности ввода данных в СОЗУ ПЭ Am2903 расширены по сравнению с устройством Am2901. В последнем в СОЗУ всегда загружаются данные, передаваемые с выхода АЛУ. Данные, поступающие из внешних источников, перед вводом в СОЗУ должны пройти через АЛУ. В ПЭ Am2903 эта возможность сохраняется за счет использования входов прямого ввода данных DA_{0-3}, причем добавлены аналогичные входы DB_{0-3}. Однако наряду с такой возможностью предусмотрен прямой тракт передачи поступающих в устройство данных на входы СОЗУ. Этот тракт реализуется путем подключения выхода регистра сдвига АЛУ к входу СОЗУ. Этот выход соединен, как и в ПЭ Am2901, с выходами Y устройства, однако управление работой выходных линий Y по схеме с тремя состояниями позволяет использовать их для прямого ввода данных в СОЗУ, когда выход регистра сдвига АЛУ находится в состоянии высокого импеданса.

В ПЭ Am2903 могут выполняться операции как арифметического, так и логического сдвига данных. Во время арифметического сдвига бит знака S (старший значащий бит) остается на своем месте, а остальные биты сдвигаются. Во время операции логического сдвига данные рассматриваются как число без знака; при этом все биты сдвигаются вправо или влево.

Как будет видно в дальнейшем, обработка старших по значимости разрядов слова данных отличается от обработки остальных разрядов. Следовательно, должны быть предусмотрены средства для указания относительного положения разрядов слова, обрабатываемых каждой секцией. Для этой цели в ПЭ Am2903 используются два вывода — LSS (least significant

slice) и MSS (most significant slice), сигналы на которых определяют секции процессора с разрядно-модульной организацией, обрабатывающие самые младшие по значимости, самые старшие по значимости и промежуточные разряды слова данных. Как будет показано ниже при обсуждении операций сдвига, различные операции выполняются по-разному в зависимости от относительного положения секции в структуре процессора.

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ УСТРОЙСТВА

Процессорный элемент Am2903 выполняет в два раза больше операций, чем ПЭ Am2901: семь арифметических и девять логических операций с двумя операндами (табл. 2.6). Источники данных, выбираемых в качестве операндов, приведены в табл. 2.7.

Если сигналы на входах кода операции I_4, I_3, I_2, I_1 имеют низкий уровень, то устройство выполняет специальные операции. К этим операциям относятся:

1. *Нормализация слова нормальной и двойной длины.* Нормализация слова нормальной длины осуществляется с помощью регистра Q. Операция нормализации заключается в сдвиге битов слова влево до тех пор, пока два старших по значимости

Таблица 2.6. Операции АЛУ устройства Am2903[1)]

I_4	I_3	I_2	I_1	КОП (шестнадцатеричное представление)	Операции АЛУ
Н	Н	Н	Н	0	I_0=Н Специальные операции I_0=В F_i=В
Н	Н	Н	В	1	$F=S-R-1+C_n$
Н	Н	В	Н	2	$F=R-S-1+C_n$
Н	Н	В	В	3	$F=R+S+C_n$
Н	В	Н	Н	4	$F=S+C_n$
Н	В	Н	В	5	$F=S+C_n$
Н	В	В	Н	6	$F=R+C_n$
Н	В	В	В	7	$F=R+C_n$
В	Н	Н	Н	8	$F_i=\overline{H}$
В	Н	Н	В	9	$F_i=\overline{R_i}$ И S_i
В	Н	В	Н	A	$F_i=R_i$ ИСКЛЮЧАЮЩЕЕ НЕ-ИЛИ S_i
В	Н	В	В	B	$F_i=R_i$ ИСКЛЮЧАЮЩЕЕ ИЛИ S_i
В	В	Н	Н	C	$F_i=R_i$ И S_i
В	В	Н	В	D	$F_i=R_i$ НЕ-ИЛИ S_i
В	В	В	Н	E	$F_i=R_i$ НЕ-И S_i
В	В	В	В	F	$F_i=R_i$ ИЛИ S_i

[1)] Н — низкий уровень сигнала; В — высокий уровень сигнала; i=0÷3.

Таблица 2.7. Источники операндов АЛУ устройства Am2903[1])

$\overline{E}_A$	I_0	$\overline{OE}_B$	Операнд R АЛУ	Операнд S АЛУ
Н	Н	Н	Выход А СОЗУ	Выход В СОЗУ
Н	Н	В	Выход А СОЗУ	DB_{0-3}
Н	В	Х	Выход А СОЗУ	Регистр Q
В	Н	Н	DA_{0-3}	Выход В СОЗУ
В	Н	В	DA_{0-3}	DB_{0-3}
В	В	Х	DA_{0-3}	Регистр Q

[1]) Н — низкий уровень сигнала; В — высокий уровень сигнала; Х — значение сигнала не влияет на выбор источников операндов АЛУ.

бита не будут иметь противоположные значения[1]). Старший по значимости бит воспринимается как бит знака; за ним следуют подразумеваемая точка, отделяющая целую часть числа, и старший по значимости бит двоичного числа. Разряды сдвинутого влево числа заполняются справа нулями.

Ненормализованное положительное 16-разрядное число

Нормализованное положительное 16-разрядное число

В приведенном примере показана нормализация 16-разрядного слова нормальной длины при использовании четырех ПЭ Am2903. При нормализации слов двойной длины часть слова с младшими по значимости разрядами записывается в регистр Q, а часть слова со старшими по значимости разрядами — в выбранный регистр СОЗУ. Интерфейс между секциями процессора, обрабатывающими разные группы разрядов числа, различен для этих двух типов нормализации. Схема соединений приводится в документации по применению устройства, выпускаемой фирмой-изготовителем. Нормализация используется для преобразования чисел с фиксированной точкой в числа с плавающей точкой, а также для выполнения операций деления.

2. *Увеличение на 1 или 2.* Операция используется главным образом в машинах с байто- и слово-ориентированной структу-

[1]) Например, один бит равен 0, а другой бит равен 1. — *Прим. перев.*

рой данных, когда адреса 16-битовых слов данных представляются четными числами, а адреса 8-битовых элементов данных (байтов) — произвольными числами.

3. *Преобразование числа, представленного в виде абсолютной величины со знаком, в дополнительный код.* Двумя наиболее часто используемыми представлениями двоичных чисел являются дополнительный код и абсолютная величина числа со знаком (Кл., 1980). Процессорные элементы Am2903 могут быть соединены таким образом, что такое преобразование будет выполняться как одна из специальных операций устройства.

4. *Умножение без знака.* В ПЭ Am2903 реализация этой операции предполагает выполнение ранее описанных базовых операций сдвига и сложения. Для умножения чисел с разрядностью, кратной 4 (4n×4n), требуется 4n такта работы устройства. При выполнении операции предполагается, что регистр R_0 СОЗУ был предварительно очищен и будет использоваться для размещения самых старших по значимости битов частичных произведений (8n бит) и что множимое записывается в регистр R_1, а множитель — в регистр R_2. Операция реализуется путем

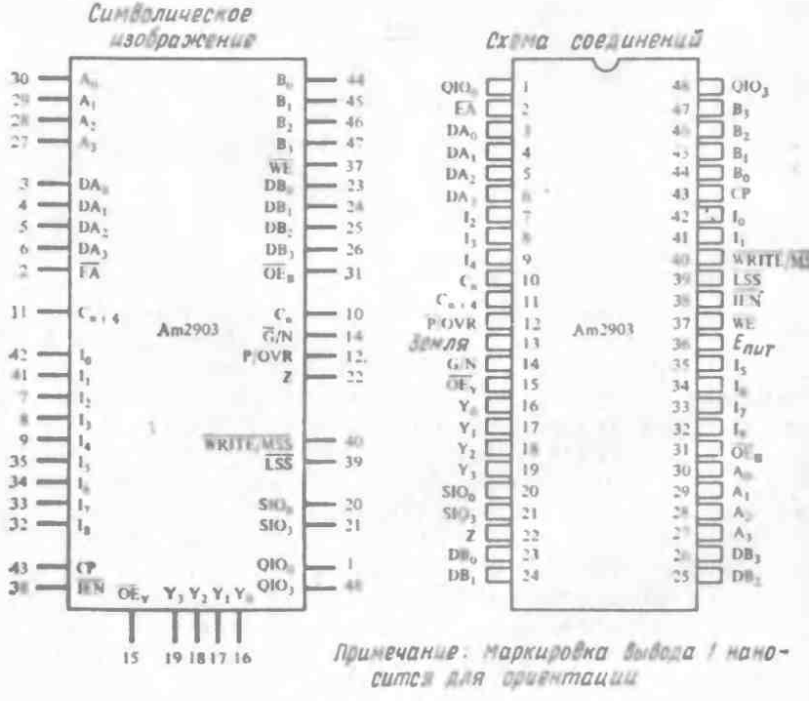


Рис. 2.9. Выводы ПЭ Am2903. (*С разрешения* Advanced Micro Devices.)

Таблица 2.8. Назначение выводов ПЭ Am2903

Вывод	Назначение
A_{0-3}	4 входа адреса регистра СОЗУ, содержимое которого передается в выходной порт А СОЗУ
B_{0-3}	4 входа адреса регистра СОЗУ, содержимое которого передается в выходной порт В СОЗУ; в этот регистр записываются входные данные при поступлении на вывод $\overline{WE}$ сигнала разрешения записи и низком уровне сигнала на линии синхронизации (вывод СР)
$\overline{WE}$	Вход сигнала разрешения записи данных в СОЗУ. При низком уровне сигнала данные, находящиеся в порте ввода-вывода Y, записываются в СОЗУ во время появления низкого уровня сигнала на линии синхронизации СР. При высоком уровне сигнала запись данных в СОЗУ запрещена
DA_{0-3}	Ввод в устройство Am2903 4-битовых данных. Эти выводы могут быть выбраны в качестве одного из источников операндов АЛУ; DA_0 — вывод самого младшего по значимости бита данных
$\overline{EA}$	Вход сигнала управления. При высоком уровне сигнала осуществляется выбор в качестве источника операнда R АЛУ выводов DA_{0-3}, а при низком уровне — выхода А СОЗУ
DB_{0-3}	Ввод-вывод в устройство 4-битовых данных. В зависимости от уровня сигнала на выводе $\overline{OE}_B$ на выводы DB_{0-3} могут непосредственно записываться данные из выходного порта В СОЗУ либо эти выводы могут быть выбраны в качестве источника операнда S АЛУ
$\overline{OE}_B$	Вход сигнала управления. При низком уровне сигнала разрешается передача данных из выходного порта В СОЗУ на выводы DB_{0-3}, при высоком уровне сигнала буфера выходного порта В, управляемого по схеме с тремя состояниями, СОЗУ запираются
C_n	Ввод сигнала переноса в АЛУ ПЭ2903
Z	Ввод-вывод по схеме с открытым коллектором. Появление высокого уровня сигнала на этом выводе обычно указывает на то, что сигналы на всех выходах Y_{0-3} имеют низкий уровень. Для некоторых специальных операций вывод Z используется в качестве входного.
SIO_0 SIO_3	Входы-выходы цепи сдвига данных в двух направлениях, осуществляемого регистром сдвига АЛУ. При сдвиге в сторону старших по значимости разрядов слова вывод SIO_0 используется в качестве входного, а SIO_3 — в качестве выходного. При сдвиге в сторону младших по значимости разрядов слова вывод SIO_3 используется в качестве входного, а SIO_0 — в качестве выходного
QIO_0 QIO_3	Входы-выходы цепи сдвига данных в двух направлениях, осуществляемого регистром сдвига канала Q, используемые так же, как и выводы SIO_0 и SIO_3

Продолжение

Вывод	Назначение
$\overline{LSS}$	Вход сигнала управления. Низкий уровень сигнала настраивает устройство на работу в качестве секции процессора с разрядно-модульной организацией (построенного на базе ПЭ Am2903), обрабатывающей младшие по значимости разряды слова (LSS), и разрешает передачу сигнала $\overline{WRITE}$ на вывод $\overline{WRITE}/\overline{MSS}$. При высоком уровне сигнала на выводе $\overline{LSS}$ устройство настраивается на работу в качестве секции процессора, обрабатывающей любые промежуточные или старшие по значимости разряды слова, и запирает выходной буфер сигнала $\overline{WRITE}$
$\overline{WRITE}/\overline{MSS}$	При низком уровне сигнала на выходе $\overline{LSS}$ на данном выводе появляется выходной сигнал $\overline{WRITE}$; сигнал $\overline{WRITE}$ имеет низкий уровень при выполнении команды, осуществляющей запись данных в СОЗУ. При высоком уровне сигнала на выводе $\overline{LSS}$ вывод $\overline{WRITE}/\overline{MSS}$ используется в качестве входного; высокий уровень сигнала на данном выводе настраивает устройство на работу в качестве секции процессора, обрабатывающей промежуточные группы разрядов слова (IS — intermediate slice), а низкий уровень сигнала — на работу в качестве секции процессора, обрабатывающей старшие разряды слова (MSS)
I_{9-0}	9 входов кода операции, используемых для указания операции, подлежащей выполнению в ПЭ Am2903
$\overline{IEN}$	Вход сигнала разрешения операции. При низком уровне сигнала разрешается передача выходного сигнала $\overline{WRITE}$ и допускается запись данных в регистр Q и триггер сравнения знака. При высоком уровне сигнала на выводе IEN принудительно устанавливается высокий уровень выходного сигнала $\overline{WRITE}$, причем содержимое регистра Q и триггера сравнения знака сохраняется неизменным
C_{n+4}	Появление сигнала на этом выводе означает появление выходных данных в цепи переноса АЛУ ПЭ Am2903
$\overline{G}/N$	Сигнал на этом выводе, имеющем многоцелевое назначение, указывает на генерирование бита переноса в секциях процессора, обрабатывающих младшие по значимости и промежуточные группы разрядов слова (сигнал $\overline{G}$), а также определяет знак результата выполнения операции в АЛУ секции процессора, обрабатывающей старшие по значимости разряды слова (сигнал N)
P/OVR	Сигнал на этом выводе, имеющем многоцелевое назначение, указывает на передачу данных по цепи переноса в секциях процессора, обрабатывающих младшие по значимости и промежуточные группы разрядов слова (сигнал $\overline{P}$), а также на появление переполнения при формировании дополнительного кода в секции процессора, обрабатывающей старшие по значимости разряды слова (сигнал OVR)

Продолжение

Вывод	Назначение
Y_{0-3}	4 вывода, предназначенные для ввода-вывода данных в ПЭ Am2903. При соответствующем уровне сигнала управления на выводе $\overline{OE}_Y$ разрешается передача данных на эти выводы с выхода регистра сдвига АЛУ. Данные выводы могут также использоваться в качестве входа при непосредственной записи в СОЗУ данных, поступающих в устройство
$\overline{OE}_Y$	Вход сигнала управления. При низком уровне сигнала разрешается передача данных с выхода регистра сдвига АЛУ на выводы Y_{0-3}; при высоком уровне сигнала запираются буферы выходов Y_{0-3}, управляемых по схеме с тремя состояниями
CP	Вход синхронизации ПЭ Am2903. Синхронизация работы регистра Q и триггера сравнения знака производится при поступлении переднего фронта синхроимпульсов на вывод CP. При поступлении сигнала разрешения записи $\overline{WE}$ и низком уровне сигнала на входе CP производится запись данных в СОЗУ устройства

передачи множителя в регистр Q и выполнения команды сложения числовых данных без знака 4n раз. Подпрограмма умножения может быть составлена из двух управляющих слов (микрокоманд), выполнение которых организуется с помощью внешнего (по отношению к ПЭ Am2903) счетчика и устройства управления (например, с помощью устройства Am2910, описываемого в гл. 5).

5. *Умножение со знаком.* Алгоритм умножения чисел, представленных в дополнительном коде, реализуется путем выполнения операций сложения числовых данных со знаком в течение 4n—1 тактов и последующего формирования дополнительного кода результата операции за один такт.

6. *Деление с формированием дополнительного кода.* ПЭ Am2903 может выполнять операции деления с обычной и повышенной точностью, когда абсолютная величина делителя превышает величину делимого. Схема соединений секций процессора и код микроопераций для реализации деления с нормальной и повышенной точностью приводятся в документации по использованию ПЭ Am2903, предоставляемой фирмой-изготовителем.

СХЕМА ВЫВОДОВ УСТРОЙСТВА

Схема выводов ПЭ Am2903 показана на рис. 2.9. Эту схему следует рассматривать вместе с описанием назначения выводов, содержащимся в табл. 2.8. Схемы соединений секций процессо-

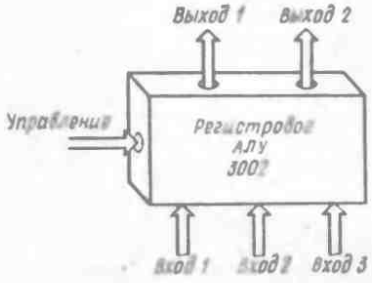


Рис. 2.10. Управление и ввод-вывод данных в РАЛУ 3002.

ра с разрядно-модульной организацией слишком многочисленны для того, чтобы приводить их в данной книге, — все эти схемы содержатся в документации фирмы-изготовителя. Для получения более детальной информации об использовании выводов следует обратиться к таблицам, помещенным в указанной документации.

ПРОЦЕССОРНЫЙ ЭЛЕМЕНТ 3002

Регистровое АЛУ (РАЛУ)[1] 3002, выпускаемое фирмами Intel и Signetics, было разработано для тех же целей, что и ПЭ Am2901 и Am2903. Процессорный элемент 3002 отличается от ПЭ Am2901 главным образом разрядностью обрабатываемых данных и возможностями системы ввода-вывода. В 4-разрядных ПЭ Am2901 ввод и вывод данных осуществляется через любой из имеющихся портов. Процессорный элемент 3002 (рис. 2.10)

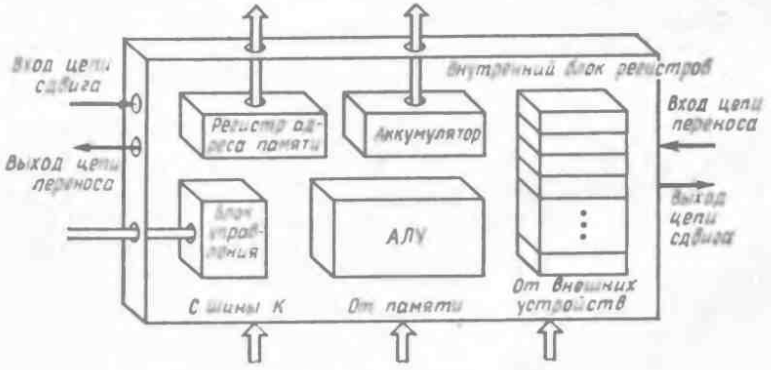


Рис. 2.11. Компоненты ПЭ 3002.

[1] Термином «регистровое АЛУ» автор называет модуль процессора, включающий АЛУ и внутренний блок регистров. — *Прим. перев.*

является 2-разрядным, но имеет три входных и два выходных порта, в связи с чем часто возникает потребность в нескольких внешних трактах передачи данных.

Процессорный элемент 3002 может применяться в системах различной конфигурации, однако обычно выход 1 используется в качестве источника адреса памяти, а выход 2 — для вывода данных. На входы 1, 2 и 3 обычно поступают данные из управляющего ПЗУ, памяти и внешних устройств соответственно. Как следует из названия устройства, регистровое АЛУ содержит внутренний блок регистров и собственно АЛУ. Кроме того, блок управления модуля выполняет декодирование сигналов, поступающих на вход сигналов управления, и выбирает регистры с исходными данными (операндами), регистр — адресат операции и тип операции, которая должна быть выполнена АЛУ.

Компоненты регистрового АЛУ приведены на рис. 2.11. Для облегчения пользования документацией, поставляемой фирмой-изготовителем устройства, применяется стандартная терминология.

АЛУ ПЭ 3002 может выполнять следующие операции: арифметические операции в дополнительном коде, увеличение и уменьшение на 1, сдвиг влево, сдвиг вправо, И (AND), ИЛИ (OR), ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR), инвертирование (NOT). Во время выполнения арифметических операций в выходной цепи переноса появляется бит переноса (рис. 2.11).

РАЗРЯДНО-МОДУЛЬНАЯ ОРГАНИЗАЦИЯ ПРОЦЕССОРА

Процессорный элемент 3002 является 2-разрядным. Это означает, что через порты ввода-вывода, внутренние тракты данных, регистры и АЛУ передаются данные, состоящие из 2 бит. Поскольку разрядность практически всех процессоров превышает величину 2, ПЭ 3002 соединяют друг с другом в многослойную структуру типа «сэндвич» (рис. 2.12), обеспечивая тем самым разрядность обрабатываемых данных, кратную 2.

Разрядно-модульная организация процессора позволяет выполнять параллельную обработку 2-битовых полей слова данных. При этом слово может содержать любое количество битов, кратное 2. При реализации многих операций, например сложения, вычитания или сдвига, требуется передача дополнительной информации: битов переноса, битов заема, а также битов, передаваемых в следующую секцию процессора в результате сдвига. Таким образом, в любых машинах с разрядно-модульной организацией должны быть предусмотрены выводы для передачи между секциями битов информации такого рода. Поскольку указанные выше операции выполняются блоком АЛУ процессора, линии передачи дополнительной информации подключены именно к этим блокам ПЭ 3002 (рис. 2.13). При выполнении ариф-

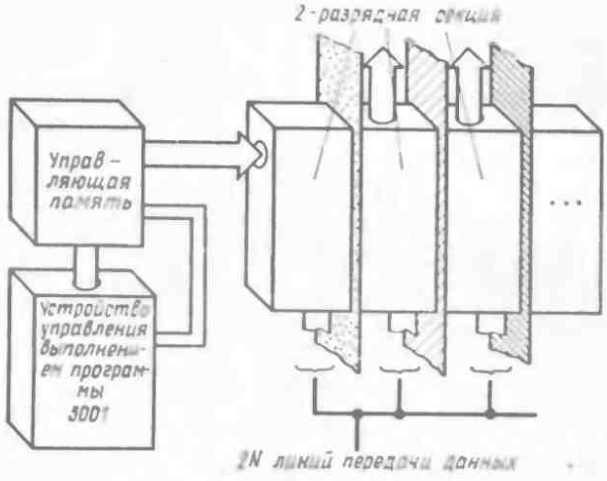


Рис. 2.12. N 2-разрядных секций, образующих 2N-разрядный процессор.

метических операций бит переноса появляется на выходе цепи переноса. Во время логических операций выполняется операция ИЛИ над данными длиной в слово, выбранными в качестве операнда (с использованием маски, передаваемой, как будет показано ниже, по шине K), причем результат операции появляется на выходе цепи переноса, что обеспечивает возможность тестирования бита и обнаружения нулевого значения.

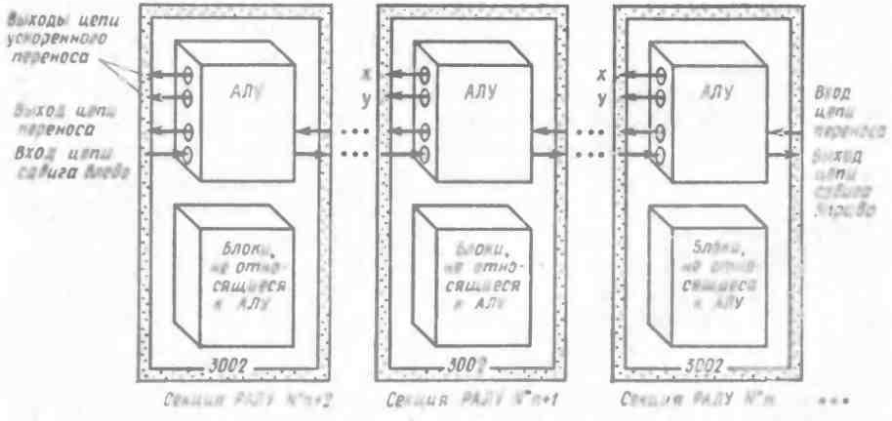


Рис. 2.13. Цепи переноса, сдвига и ускоренного перенюса.

ОРТОГОНАЛЬНОСТЬ ОПЕРАЦИЙ ПЕРЕНОСА И СДВИГА, ВЫПОЛНЯЕМЫХ В ПЭ

Операции, связанные с переносом битов в старшие разряды, и операции сдвига являются взаимоисключающими, или ортогональными, в том смысле, что при выполнении операций сдвига никогда не происходит переноса, а при выполнении операций, вызывающих перенос, не производится сдвига битов слова данных. Таким образом, хотя имеются два канала ввода и вывода дополнительной информации, в любой момент времени между секциями процессора передается не более 1 бит такой информации. По этой причине управление работой выходов осуществляется по схеме с тремя состояниями, причем неиспользуемый в данный момент времени выход поддерживается в состоянии высокого импеданса. Таким образом, выходы цепей переноса и сдвига вправо могут быть объединены и информация может передаваться по одной линии. Точно так же можно объединить и входы цепи переноса и сдвига влево, поскольку эти два сигнала никогда не используются в секциях процессора одновременно.

УСКОРЕННЫЙ ПЕРЕНОС В ПРОЦЕССОРЕ С РАЗРЯДНО-МОДУЛЬНОЙ ОРГАНИЗАЦИЕЙ

Не относящаяся к АЛУ часть ПЭ на рис. 2.13 изображена в виде единого блока. В АЛУ предусмотрены выходы цепи ускоренного переноса, используемые для быстрой передачи сигнала переноса с помощью такого устройства, как схема ускоренного переноса (СУП) 3003 фирмы Intel, показанная на рис. 2.14. Эти выходы, построенные на комбинационных схемах, уменьшают время, обычно затрачиваемое на передачу сигнала переноса, за счет параллельной передачи битов переноса вместо передачи их по цепочке. Сигнал разрешения передачи битов переноса используется для запирания СУП 3003 на время выполнения операций сдвига вправо. Этот сигнал может выдаваться дешифратором команд при декодировании микрокоманды сдвига слова данных вправо или задаваться битом кода микрооперации.

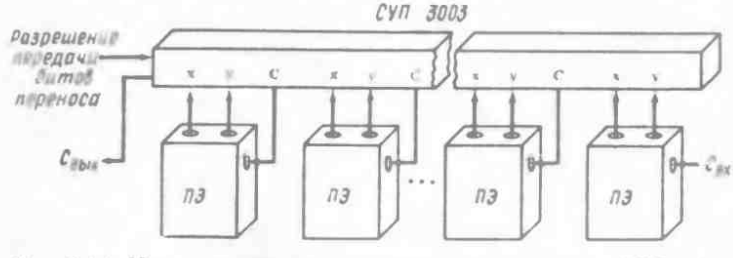


Рис. 2.14. Использование схемы ускоренного переноса 3003, обеспечивающей управление работой до 8 РАЛУ 3002.

ТРАКТЫ ПЕРЕДАЧИ ДАННЫХ ПЭ 3002

Внутренние тракты передачи данных ПЭ 3002 показаны на рис. 2.15. На обоих входах АЛУ используется мультиплексирование входных данных, поступающих из различных источников. Такими источниками для канала А являются аккумулятор, ОЗУ и внутренний набор регистров. Для канала В источниками служат аккумулятор, внешние устройства (ВУ) и шина К (шина маски данных). Содержимое аккумулятора может использоваться в качестве исходного операнда АЛУ либо в канале А, либо в канале В. Другими источниками входных данных могут быть внешний по отношению к процессору набор регистров (ОЗУ) или внутренний блок регистров (СОЗУ) — для канала А и внешние устройства и шина К — для канала В. Код, посту-

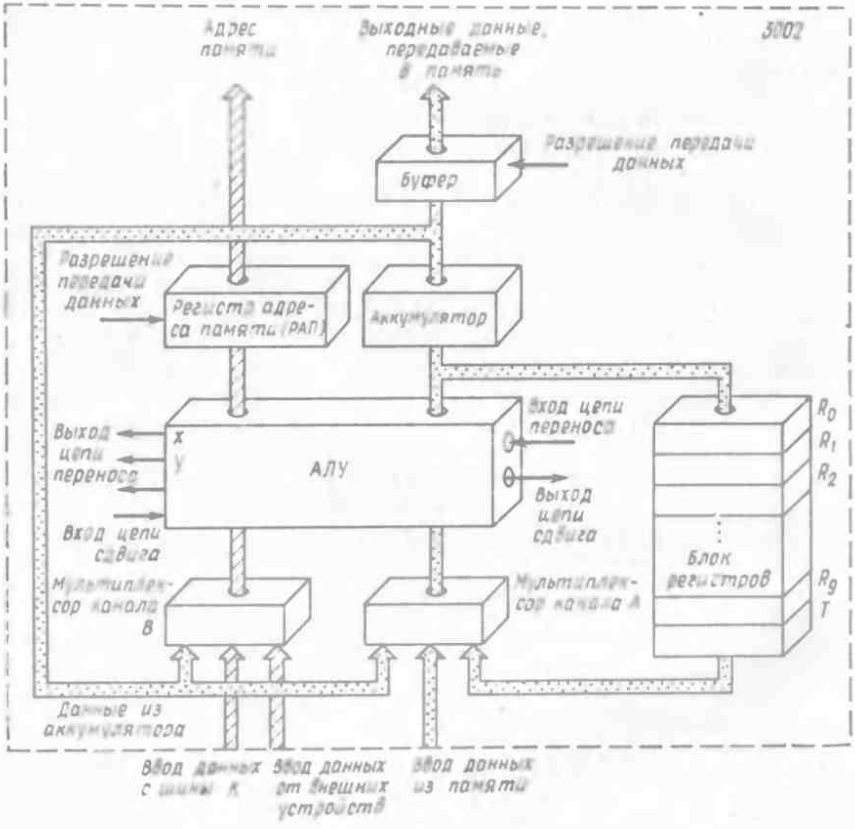


Рис. 2.15. Тракты передачи данных в ПЭ 3002.

пающий с шины К, логически умножается на содержимое выбранного входа канала В. Для большинства операций это умножение применяется с целью выборки данных из аккумулятора с использованием маски, состоящей из битов, равных только 1 или только 0. На шину К данные обычно поступают из поля К-маски микрокоманды. Данная шина может использоваться для загрузки в устройство 3002 константы. Использование маски, передаваемой по шине К, для тестирования посредством операции ИЛИ данных длиной в слово (с передачей сигнала на выход цепи переноса) обеспечивает гибкое маскирование и проверку значений отдельных битов данных. Набор операций устройства 3002 приведен в табл. 7.2.

МИКРОПРОЦЕССОРЫ 74S481 ФИРМЫ TEXAS INSTRUMENTS

Архитектура микропроцессора 74S481 фирмы Texas Instruments идентична архитектуре семейства микропроцессоров 9900, разработанных этой же фирмой, с командами типа память — память[1] (см. Кл., 1980). В соответствии со сложившимися традициями в вычислительных системах используются относительно низкоскоростная основная память и сверхбыстродействующая внутренняя буферная память процессора. В середине 70-х годов специалисты фирмы Texas Instruments решили, что быстродействия полупроводниковой основной памяти достаточно для выполнения любых вычислительных операций, и разработали архитектуру микро-ЭВМ, в которой блок рабочих регистров процессора размещается в основной памяти. Основное преимущество такой архитектуры связано с обеспечением возможности управления прерываниями или переключением программ. При классической архитектуре появление прерывания, приводящее к перехвату управления процессором на короткий интервал времени, требует выполнения процедуры обслуживания прерывания, заключающейся в запоминании содержимого внутреннего блока регистров процессора для последующего восстановления. В архитектуре с командами типа память — память набор регистров процессора представляет собой часть ячеек основной памяти и внутри процессора хранится только указатель используемого в данный момент набора регистров. Таким образом, во время прерываний должны запоминаться только текущие указатели наборов регистров процессора, что занимает существенно меньше времени, чем сохранение непосредственно содержимого этих наборов регистров.

---

[1] То есть без использования в процессоре блока внутренних регистров (СОЗУ). — *Прим. перев.*

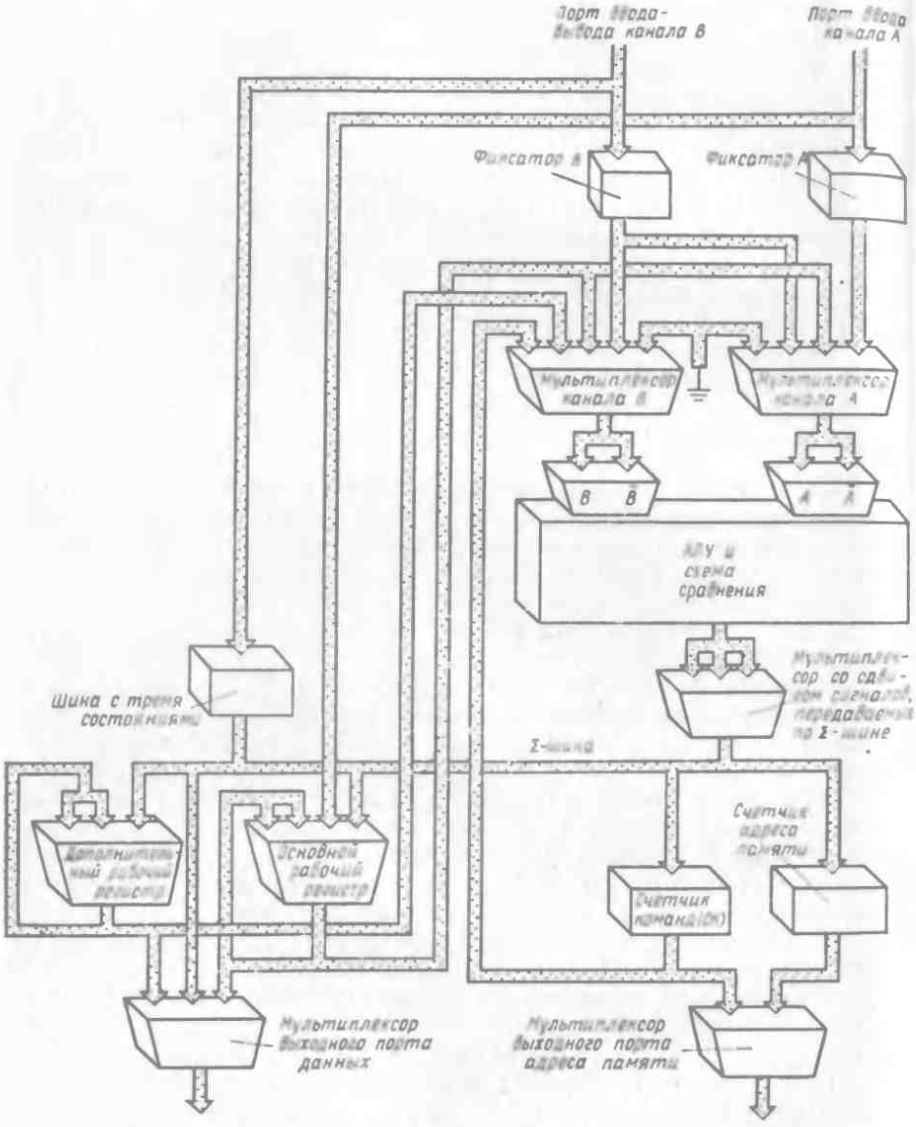


Рис. 2.16. Структура ПЭ 74S481 фирмы Texas Instruments.

Процессорный элемент 74S481 представляет собой 4-разрядную секцию — модуль процессора с разрядно-модульной организацией, — располагающую обычными линиями ввода и вывода битов переноса, линиями ввода и вывода битов, передаваемых между секциями процессора в результате сдвига данных, ли-

ниями вывода сигнала переполнения, линиями передачи сигналов ускоренного переноса.

Необычная особенность устройства заключается в использовании единственного вывода для определения относительной позиции данной секции в многосекционном процессоре (обработка самых младших по значимости, промежуточных и самых старших по значимости групп разрядов слов данных) за счет распознавания трех уровней напряжения на данном выводе (0÷0,8 В; 1,8÷3 В и 3,6÷5 В соответственно).

Процессорный элемент имеет по два параллельно работающих входных и выходных порта. Кроме того, один из входных портов может использоваться в качестве выходного. Основные тракты передачи данных внутри ПЭ 74S481 показаны на рис. 2.16. Блок управления БИС располагает 10 линиями выбора операции; кроме того, работа блока зависит от сигнала на входе цепи переноса АЛУ. Предполагается, что в секциях процессора, обрабатывающих самые старшие или самые младшие по значимости разряды слова, две из этих линий при выполнении определенных операций могут использоваться для ввода-вывода данных.

Секционные процессоры 74S481 являются исключительно гибкими устройствами и позволяют выполнять 24 780 операций. Эти операции описаны в таблицах, приводимых в документации, издаваемой фирмой Texas Instruments. Большая часть операций реализуется за счет параллельного управления работой различных элементов процессора. Это позволяет одновременно выполнять составные операции следующего типа:

1. Выбор функции АЛУ.
2. Управление сдвигом.
3. Выбор адресата результата операции с обновлением адреса или цикла.
4. Передача адреса и данных в память.

Составные операции, выполняемые за 1-тактовый цикл, синхронизируются однофазными синхроимпульсами (рис. 2.17).

Как показано на рис. 2.16, ПЭ 74S481 содержит основной и дополнительный рабочие регистры, обеспечивающие выполнение операций арифметического, логического и циклического

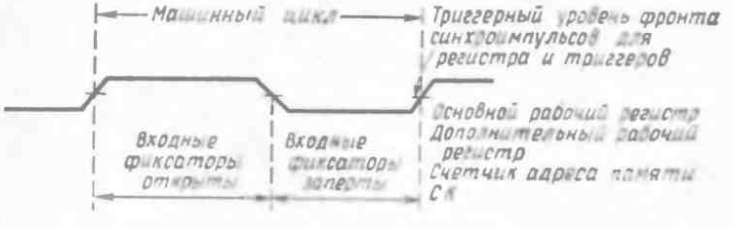


Рис. 2.17. Цикл работы устройства ПЭ 74S481.

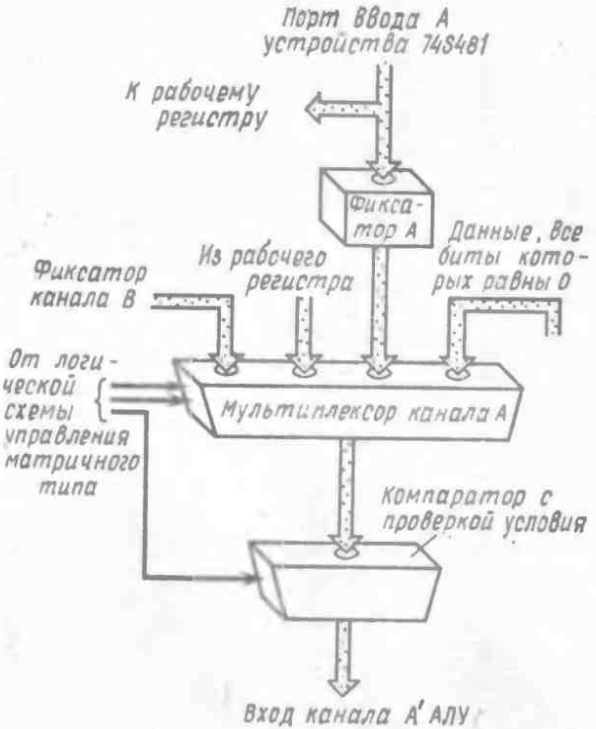


Рис. 2.18. Источники операнда канала А' АЛУ.

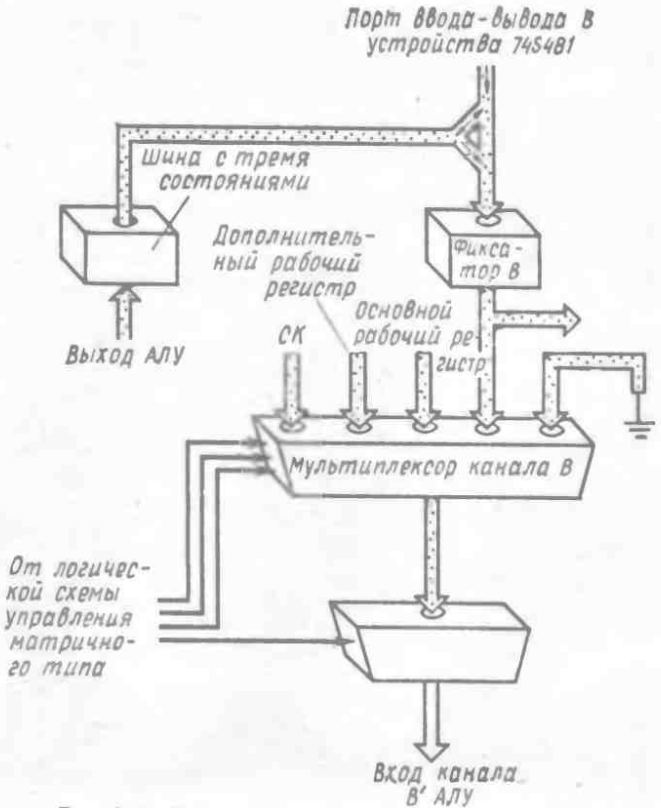


Рис. 2.19. Источники операнда канала В' АЛУ.

сдвига слов данных нормальной и двойной длины с сохранением бита знака числа. Соединения между регистрами отдельных секций процессора и управление их работой подробно описаны в документации фирмы.

На входы АЛУ могут передаваться операнды из разных источников (рис. 2.18). На вход A′ АЛУ может передаваться исходный или дополнительный код данных, поступающих из следующих источников: 1) фиксатора входа канала A; 2) фиксатора входа канала B; 3) рабочего регистра; 4) входа данных, все биты которых равны 0.

На вход B′ АЛУ может передаваться исходный или дополнительный код данных, поступающих из следующих источников (рис. 2.19): 1) фиксатора входа канала B; 2) шины результата операции (выхода АЛУ); 3) основного рабочего регистра; 4) дополнительного рабочего регистра; 5) счетчика команд; 6) входа данных, все биты которых равны 0.

ЗАКЛЮЧЕНИЕ

Одной из задач проектирования цифровых вычислительных устройств является обеспечение выполнения арифметических и логических операций над данными. Данные и промежуточные результаты обычно помещаются в буферную память процессора, более быстродействующую и более дорогую, чем основная память. Стремление к универсальности привело к созданию ПЭ — специальных устройств, построенных на быстродействующих БИС. Поскольку главным источником обрабатываемых данных является основная память, ПЭ обычно содержат в той же самой БИС регистры адреса памяти, а также входные и выходные порты данных, хотя существует по крайней мере одна реализация системы, в которой функция интерфейса процессора с памятью выполняется с помощью отдельного устройства.

Основными целями применения рассмотренных в данной главе ПЭ является достижение высокой производительности и гибкости системы. Первое становится возможным благодаря использованию технологии интегральных микросхем, обеспечивающей высокое быстродействие, например, такой, как технология ТТЛ-схем с диодами Шоттки или ЭСЛ-схем (схем с эмиттерно-связанной логикой), а также соответствующему выбору архитектуры. Вторая цель обеспечивается разрядно-модульной организацией процессора, а также выбором рациональной архитектуры системы. За счет высокого уровня параллелизма доступа к секциям процессора можно выполнять до 24 780 различных операций по обработке данных.

МАШИНЫ СОСТОЯНИЙ И ТЕОРИЯ АВТОМАТОВ

При проектировании цифровых электронных устройств используется огромное число комбинаций все расширяющегося набора различных элементов. Это, с одной стороны, обеспечивает большую гибкость проектирования, а с другой — может привести к возникновению сложных задач. Значительное количество вариантов реализаций систем обусловливает развитие большого числа формальных моделей систем и средств их описания. Настоящая глава знакомит с теорией машин состояний и различными средствами их описания.

МАШИНЫ СОСТОЯНИЙ

Большинство систем, представляющих интерес для проектировщика, поддаются как наблюдению, так и управлению. Для наблюдения за системой используются ее выходы (выходные переменные), а управление системой производится через ее входы (входные переменные). Простейшая модель подобной системы условно изображена на рис. 3.1. Один из возможных способов выявления внутренней структуры машины состоит в поиске отношения между входными и выходными переменными, или передаточной функции $H = I/Q$. Мы будем исследовать несколько классов структур систем возрастающей сложности. Наше изложение схоже с принятым в книге К. Р. Клэра[1].

МАШИНЫ КЛАССА 0

Если использовать понятие передаточной функции H, то машина простейшей структуры будет иметь передаточную функцию, выражаемую константой, $I = HQ$, где H — константа.

---

[1] Clare Ch. R., Designing Logic Systems Using State Machines, McGraw-Hill, New York, 1973.

Рис. 3.1. Простейшая модель машины.

Для машины с несколькими входами и выходами передаточная функция является фиксированной матрицей. Такое же основное представление применяется и в тех случаях, когда выходные переменные являются функциями входных переменных, т. е. $I=H(Q)$. Здесь все наблюдаемые переменные являются функциями только входных переменных, а состояния системы не являются наблюдаемыми. *Состояние системы* — это минимальная информация, которая необходима для предсказания поведения системы. Машины, поведение которых может быть предсказано исключительно по значениям входных переменных, называются *машинами класса 0.* На рис. 3.2 представлена такая машина; здесь передаточная функция обозначена символом f. Примерами машин класса 0 являются логические элементы И, ИЛИ, НЕ и схема полусумматора. Условные обозначения этих логических схем приведены на рис. 3.3.

СРЕДСТВА ОПИСАНИЯ МАШИН СОСТОЯНИЙ

Знание обозначений логических схем позволяет представить поведение каждой машины класса 0, изображенной на рис. 3.3; используемые символические обозначения удобны, однако по своей природе не являются описательными. Простейшим описательным представлением поведения машины служит *таблица истинности,* в которой каждой возможной входной комбинации

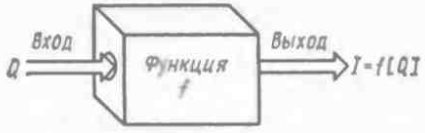


Рис. 3.2. Машина класса 0.

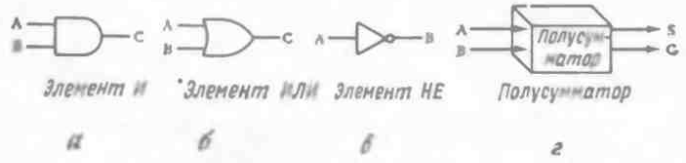


Рис. 3.3. Условные обозначения некоторых логических элементов — машин класса 0.

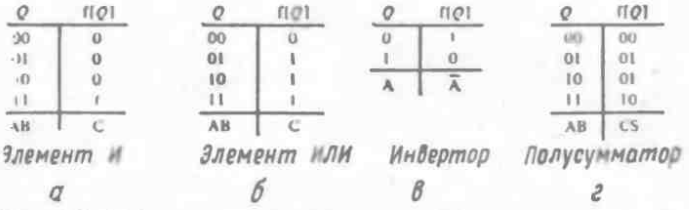


Рис. 3.4. Таблицы истинности — описательное представление машин класса 0.

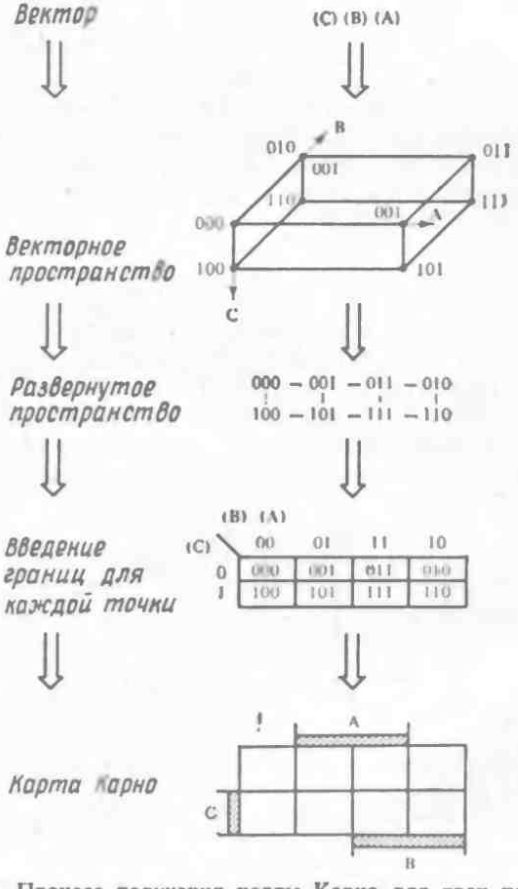
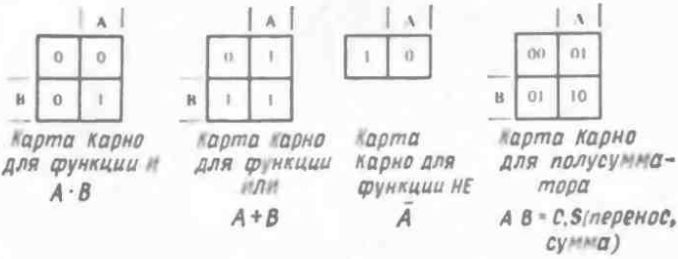


Рис. 3.5. Процесс получения карты Карно для трех переменных.

значений переменных ставится в соответствие значение выходной переменной. Таблицы, описывающие поведение машин класса 0, представлены на рис. 3.4.

Таблицы истинности могут быть преобразованы в специальные эквивалентные представления, известные как *карты Карно*. В карте Карно элементы таблицы интерпретируются как двоичные коды. Таким образом, эти коды, отличающиеся только значением одной переменной, определяются положением, которое представляется n-кой, или вектором в n-мерном пространстве. Эти n-ки могут быть отображены в двумерное пространство посредством процесса развертывания (рис. 3.5).

Используя таблицы истинности, приведенные на рис. 3.4, можно легко составить карты Карно для схем И, ИЛИ, НЕ и полусумматора.



Карта Карно для функции И $A \cdot B$

Карта Карно для функции ИЛИ $A+B$

Карта Карно для функции НЕ $\bar{A}$

Карта Карно для полусумматора $A\,B = C,S$ (*перенос, сумма*)

С целью минимизации числа элементов в цифровых схемах выполняют преобразование соответствующих булевых уравнений. Один из методов их преобразования основан на использовании карт Карно.

В современной экономике производства важными показателями являются время, затрачиваемое разработчиками на проектирование систем, и стоимость последних. Большие затраты времени на минимизацию числа элементов окупаются только в том случае, когда системы будут выпускаться в большом количестве. Проектировщику системы весьма полезно знание методов оптимизации, которые могут быть применены при разработке программного и аппаратного обеспечения систем. Существование алгоритма *табличного представления уравнений* (map-reading) позволяет проводить упрощение данного множества уравнений с помощью ЭВМ. Хотя первичными являются методы логического проектирования, проектировщики систем, имеющие доступ к программам упрощения логических выражений, должны расширять их использование на практике.

Алгоритмическая *машина состояний Клэра* схематически описывает функцию выходов и функцию переходов машины состояний. Символическим представлением состояния является блок состояния (рис. 3.6), который имеет вход и выход, имя

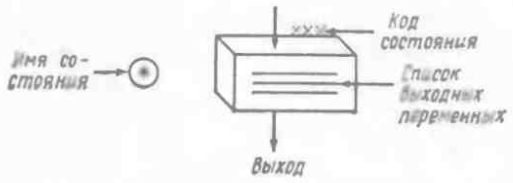
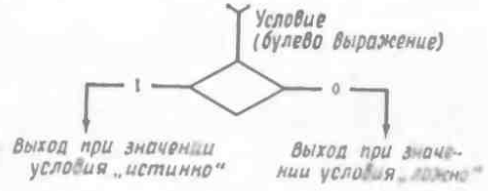


Рис. 3.6. Символическое обозначение состояния алгоритмической машины состояний Клэра.

состояния, код состояния, список выходных переменных для данного состояния. Если переход из одного состояния в другое зависит от каких-либо условий, определяемых значениями переменных, то используется логический блок, или переключатель, который имеет следующее графическое изображение



Те выходные переменные, которые зависят не только от состояния машины, но также и от входных переменных, представляют блоком условных переменных, который имеет следующее графическое изображение

Так как мы изучаем переходы из одного состояния в другое, наибольший интерес для нас представляют те графы, которые отображают связи некоторого состояния со следующими состояниями. Путь из некоторого состояния в следующее состояние будем называть *путем связи* или *связью*. Понятие *времени состояния* как периода от начала некоторого перехода до начала следующего перехода используется для построения основного блока (рис. 3.7), который представляет систему в течение времени состояния. Эквивалентные блоки алгоритмических машин состояний представлены на рис. 3.8.

Блок-схемы алгоритмических машин состояний, описывающие функции И, ИЛИ, НЕ и полусумматора, а также соответст-

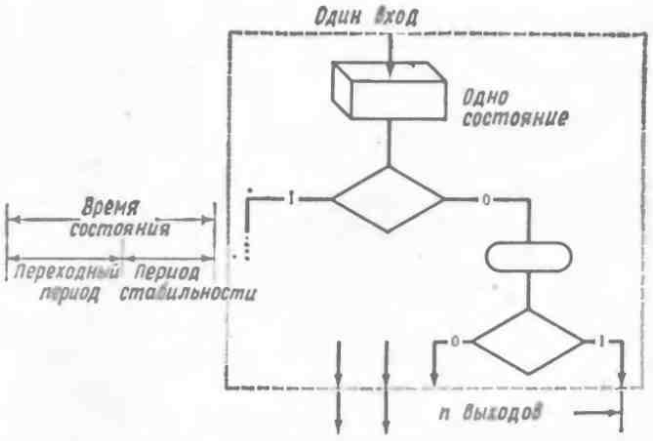


Рис. 3.7. Основной блок алгоритмической машины состояний.

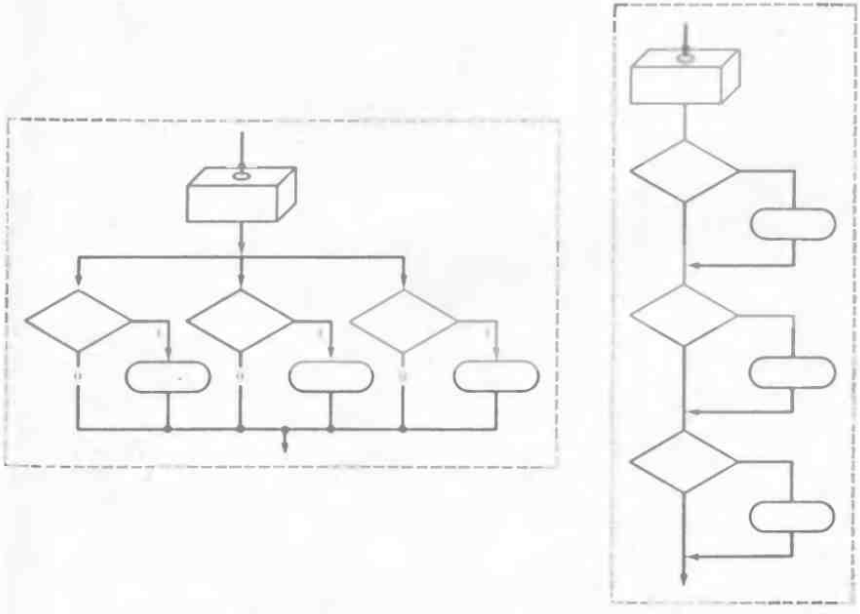


Рис. 3.8. Эквивалентные блоки алгоритмической машины состояний.

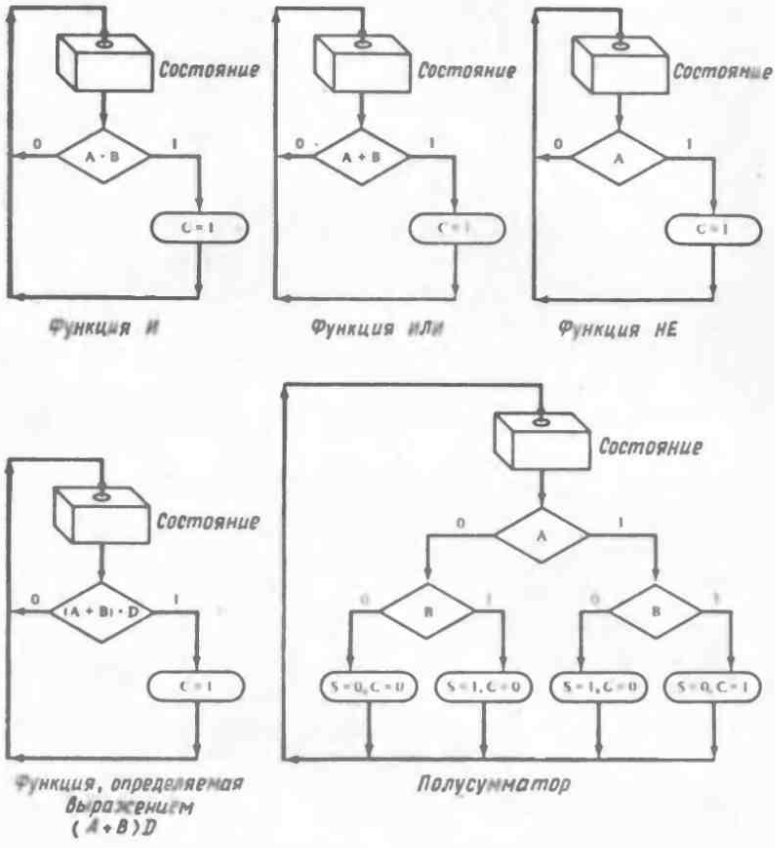


Рис. 3.9. Блок-схемы алгоритмических машин состояний для различных функций.

вующие булевы выражения, представлены на рис. 3.9. Рассмотрение этого рисунка позволяет сделать вывод, что алгоритмическая машина состояний представляет все булевы функции единообразно. Это как раз та сторона описания систем с использованием алгоритмических машин состояний, которая позволяет классифицировать и естественно развивать системы от простейших комбинационных схем до наиболее сложных машин и языковых систем. Все машины класса 0 (комбинационные логические схемы) могут быть без труда преобразованы из описания в виде алгоритмической машины состояний в табличное представление посредством перечисления условных (выходных) переменных для всех возможных комбинаций значений входных переменных. Читатель должен самостоятельно убедиться в этом

на примере полусумматора. Уравнение, соответствующее машине класса 0, имеет следующий вид:

$$I = f(Q).$$

Эта запись означает, что выходные переменные определяются входными переменными Q и структурой машины, представляемой функцией f.

МАШИНЫ КЛАССА 1

Введем в рассмотрение независимую переменную t, которая может быть либо стохастической, либо детерминированной. Если наступление события в момент t вообще непредсказуемо, то соответствующую систему называют «системой, управляемой событиями», или «событийной системой». В этом случае t является временем наступления события. Если события наступают регулярно, то время t обычно задается посредством некоторого счетчика. В этом случае t носит название «времени перехода состояния», или «времени состояния». Переход из начального состояния в конечное происходит в момент времени перехода состояния и отображается стрелкой, направленной из начального состояния в конечное. Таким образом, машину класса 1 можно представить следующим уравнением:

$$I \longleftarrow f(Q),$$

которое показывает, что выходные переменные являются функциями входных переменных (сигналов), задержанных на время одного состояния. Такая машина называется *машиной с элементом задержки,* и, следовательно, она должна иметь память или элемент памяти, способный запоминать входные сигналы, соответствующие предыдущему времени состояния. На рис. 3.10 изображены три эквивалентных представления машины класса 1.

Простым примером машины с задержкой является регистр или синхронизируемый фиксатор. Состояние 3-разрядного регистра представим вектором

$$X = [x_1 \quad x_2 \quad x_3],$$

а векторами $Q = [q_1 \; q_2 \; q_3]$ и $I = [i_1 \; i_2 \; i_3]$ обозначим соответственно входные и выходные переменные регистра.

Для того чтобы оперировать отдельными переменными, определим частичную функцию переходов, которая является подфункцией функции переходов и связана с определенной переменной, и используем для идентификации переменных обозначе-

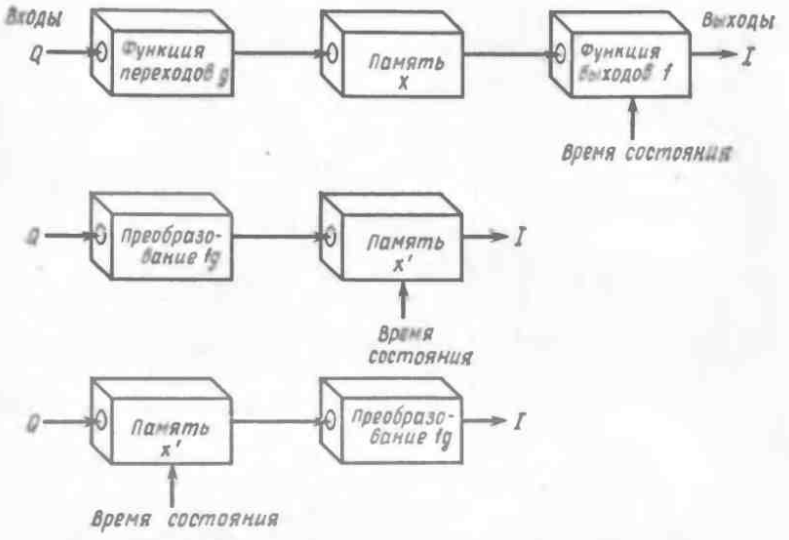
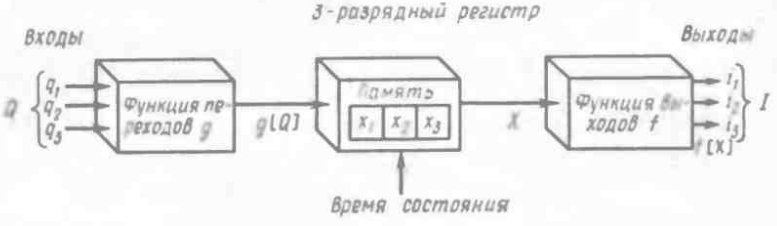


Рис. 3.10. Функционально эквивалентные представления машины класса 1.

ния с индексами. Тогда машина с задержкой может быть представлена следующим образом:



Таким образом, 3-разрядный регистр является машиной с задержкой и описывается следующими уравнениями:

$$g[Q] = \begin{bmatrix} gx_1[Q] = q_1 \\ gx_2[Q] = q_2 \\ gx_3[Q] = q_3 \end{bmatrix}, \qquad f[X] = \begin{bmatrix} fi_1[X] = x_1 \\ fi_2[X] = x_2 \\ fi_3[X] = x_3 \end{bmatrix}.$$

Эти уравнения можно записать следующим более простым способом:

$$g[Q] = \begin{bmatrix} x_1 \leftarrow q_1 \\ x_2 \leftarrow q_2 \\ x_3 \leftarrow q_3 \end{bmatrix}, \qquad f[X] = \begin{bmatrix} i_1 = x_1 \\ i_2 = x_2 \\ i_3 = x_3 \end{bmatrix}$$

или представить в эквивалентной векторной форме

$$g[Q] = [X \longleftarrow Q] \quad \text{и} \quad f[X] = [I = X].$$

Введение памяти и сигнала записи в память позволяет запоминать информацию и использовать ее впоследствии. Поэтому состояние машины может полностью или частично определяться предысторией процесса. Отметим, что в аналоговых измерительных системах обеспечивается запоминание состояния процесса путем обновления состояния памяти с помощью схем измерения и фиксации. Сигнал записи в память периодически обновляет состояние системы. Новое состояние может зависеть или не зависеть от старого состояния и (или) от текущих значений входных сигналов. Введение сигнала записи в память, кроме того, разделяет время состояния на два периода: переходный период и период стабильности. Сигналы на входе памяти должны оставаться неизменными в течение периода записи в память и могут изменяться в течение периода, когда сигнал записи в память не действует. Те сигналы, которые изменяются в течение периода записи, или стабильности, являются по отношению к системе *асинхронными*. Синхронные системы имеют равные времена состояний или синхронизируются внешним по отношению к системе источником сигнала. Асинхронные машины имеют время состояния, которое зависит только от внутренних задержек, связанных со структурой машины.

МАШИНЫ КЛАССА 2

С помощью обратной связи информация о состоянии машины может подаваться на вход машины состояний. Можно представить себе машину, у которой следующее состояние является функцией только текущего состояния и выходные переменные которой являются функциями только текущего состояния (рис. 3.11). Примером *машины состояний класса 2* может служить десятичный счетчик, в частности ТТЛ-устройство 7490 фирмы Texas Instruments. Представление такого счетчика в виде алгоритмической машины состояний дано на рис. 3.12. Машина класса 2 описывается уравнениями

$$X \longleftarrow g[X] \quad \text{и} \quad I = f[X].$$

МАШИНЫ КЛАССА 3

В десятичном счетчике реализуется циклическая детерминированная последовательность состояний, что характерно для машин класса 2. *Машины класса 3* являются обобщением машин класса 2. Они позволяют изменять последовательность состояний в ответ на входные сигналы или сигналы управления.

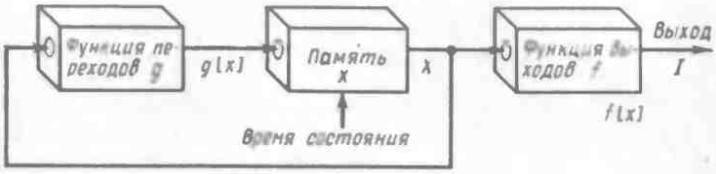


Рис. 3.11. Модель машины класса 2.

Обобщениями уравнений машин класса 2 являются следующие уравнения машин класса 3:

$$X \longleftarrow g[X, Q] \quad \text{и} \quad I = f[X].$$

Модель машины класса 3 показана на рис. 3.13.

МАШИНЫ КЛАССА 4

Обобщением машин третьего класса является машина, в которой выходные сигналы являются функциями как входных сигналов, так и текущего состояния. Соответствующие уравне-

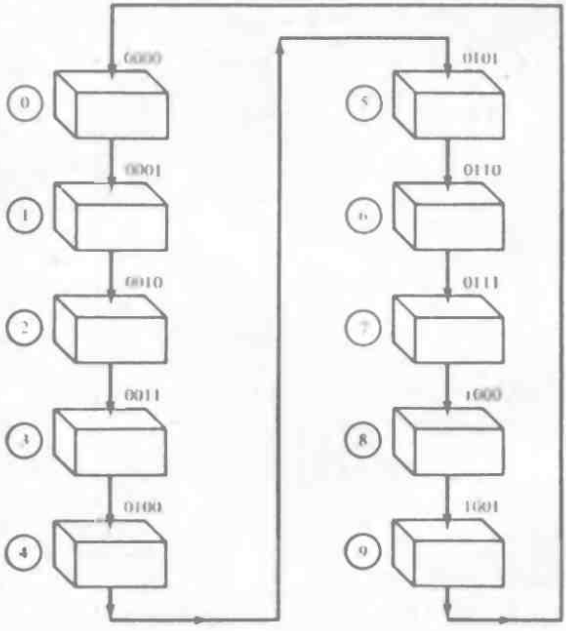


Рис. 3.12. Представление десятичного счетчика в виде алгоритмической машины состояний.

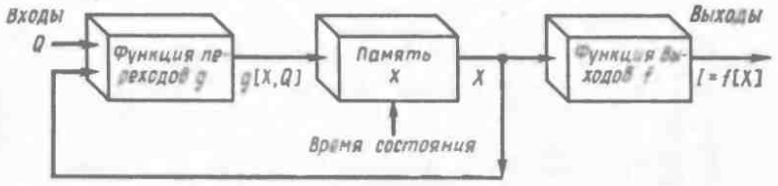


Рис. 3.13. Модель машины класса 3.

ния машин класса 4 имеют вид

$$X \longleftarrow g[X, Q] \quad \text{и} \quad I = f[X, Q].$$

В этом случае алгоритмическая машина состояний должна описываться, как показано на рис. 3.14, посредством функции выходов f.

Машина класса 4 является наиболее обобщенной. Модель машины состояний, представленную на рис. 3.14, можно изобразить иначе (рис. 3.15). Теперь нетрудно увидеть, что машина состояний класса 4 может быть сведена к известной универсальной машине (рис. 3.16).

УНИВЕРСАЛЬНАЯ МАШИНА

Универсальная машина (рис. 3.16) являлась предметом длительных научных исследований. В частности, в теории автоматов обсуждаются машины этого типа, определяемые следующим образом:

$$M = (X, Q, I, g, f),$$

где $Q = (q_1, q_2, \ldots, q_n)$ — входной алфавит,
$I = (i_1, i_2, \ldots, i_n)$ — выходной алфавит,
$X = (x_1, x_2, \ldots, x_n)$ — множество состояний,
с отображениями g и f, определяемыми следующим образом:
g: $Q \times X$ в X
f: $Q \times X$ в I автомат Мили,

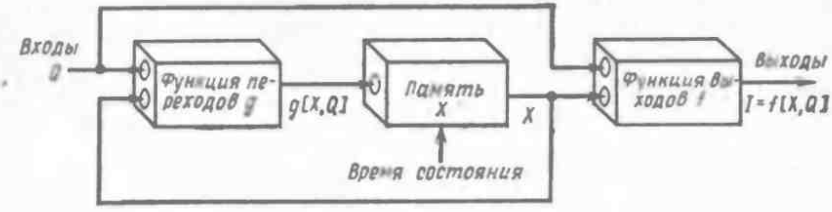


Рис. 3.14. Модель машины класса 4.

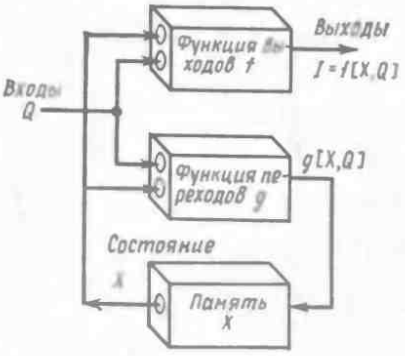


Рис. 3.15. Вариант изображения модели машины класса 4.

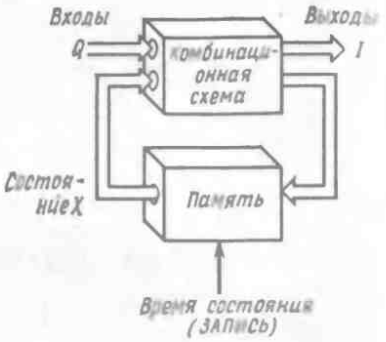


Рис. 3.16. Универсальная машина.

f: X в I автомат Мура.

Автомат Мили эквивалентен машине класса 4, а автомат Мура — машине класса 3.

ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О МАШИНАХ СОСТОЯНИЙ

Машина состояний, как мы видели, представляет собой обобщенную модель, на основе которой может быть представлена любая компонента или любой модуль некоторой логической системы. Если система такова, что в ней определено время состояния t, то функция следующего состояния g определяется следующим выражением:

$$X[(k+1)t] = g[X(kt), Q(kt)],$$

где X(kt) является состоянием системы в момент дискретного времени kt, а Q(kt) — вход системы в тот же момент времени. В обычной нотации оператор задержки обозначают стрелкой $\longrightarrow$ или $\longleftarrow$, указывающей направление замещения состояния:

$$X \longleftarrow g[X, Q].$$

Функция выходов f образует множество выходных сигналов или инструкций I на основании состояния системы и входных сигналов для каждого состояния. Основное преобразование описывается уравнением

$$I(kt) = f[X(kt), Q(kt)],$$

которое может быть записано в более краткой форме

$$I = f[X, Q].$$

В таблице 3.1 представлены данные пяти классов машин.

Таблица 3.1. Классы машин состояний

Класс машины	Функция	Характеристика класса
0	I = f[Q] X ← I	Комбинационный автомат
1	I = f[X] X ← g[Q]	Машина с задержкой
2	I = f[X]	Выход — функция состояния
	X ← g[X]	Прямой переход состояния
3	I = f[X]	Выход — функция состояния
	X ← g[X, Q]	Переход — функция состояния и входных сигналов
4	I = f[X, Q]	Выход — функция состояния и входных сигналов
	X ← g[X, Q]	Переход — функция состояния и входных сигналов

ОБЩЕЕ ПРЕДСТАВЛЕНИЕ КЛАССОВ МАШИН СОСТОЯНИЙ

Следующие уравнения справедливы как для линейных, так и для нелинейных систем:

$$X \longleftarrow g[X, Q] \quad \text{и} \quad I = f[X, Q].$$

Линейные системы обычно рассматриваются как две несвязанные системы, описываемые уравнениями:

$$X(t+1) = AX(t) + BQ(t),$$
$$I(t) = CX(t) + DQ(t),$$

где сложение может выполняться по модулю некоторого числа n. С целью дальнейшего изучения алгоритмических машин состояний и применения их для проектирования логических схем, читатель может обратиться к упомянутой выше книге Клэра.

ТЕОРИЯ АВТОМАТОВ

Целью использования аппарата алгоритмических машин состояний является главным образом упрощение системного описания, что обеспечивает реализацию систем с минимальным числом элементов. Как было показано, обобщенная машина состоя-

ний эквивалентна универсальной машине теории автоматов. Работы в теории автоматов, являющейся по существу математической теорией, направлены на решение фундаментальной задачи теории вычислений, которую можно сформулировать следующим образом: существует ли для определенного класса задач алгоритм, обеспечивающий решение за конечное число шагов?

Несмотря на то что теория автоматов имеет ограниченное применение в практике проектирования, ее результаты представляют интерес для разработчиков. Машина Тьюринга может быть описана набором выполняемых ею операций:

Ввод	Чтение входного символа Q
Проверка условия	Сравнение Q с внутренним состоянием машины X
Вывод	Вывод соответствующего выходного символа I
Выполнение перехода	Изменение внутреннего состояния X на новое состояние X′
Цикл	Повторение описанных выше шагов с новым входным символом Q′

Процессор часто представляют в форме, предложенной Тьюрингом, т. е. в виде ленты бесконечной протяженности, разделенной на ячейки, которые могут содержать символ X, в частности пустой символ. Лента проходит под головкой чтения/записи, способной воспринимать текущий символ и записывать новый символ, который, например, может совпадать с текущим символом. Входной символ определяется путем сравнения текущего символа с текущим состоянием машины. После записи символа машина выполняет переход в другое состояние, перемещение ленты вперед или назад. Затем процесс повторяется.

Хотя машина Тьюринга может быть сконструирована физически, она оказывается слишком простой и не имеет практического применения. Однако простота и универсальность математической модели машины Тьюринга во многих случаях позволяет использовать ее как идеальный аналитический аппарат.

МАТРИЧНЫЕ ЛОГИЧЕСКИЕ СХЕМЫ

В главе 3 были описаны и классифицированы машины состояний. В настоящей главе машина состояний обычно будет определяться как некоторая система или устройство, предназначенное для выполнения данной задачи. Это определение будет охватывать различные машины и системы, в частности биологические системы, системы теплопередачи, машины с микропрограммным управлением, обучающие машины и т. п. Любая теория, нацеленная на описание такого многообразия систем, должна исключать все частные свойства реальных систем, которые отвлекают внимание и не позволяют четко выявить основные элементы и структуру изучаемого объекта. Представление матричных логических схем, рассматриваемое в настоящей главе, является простым средством описания, которое дает возможность достаточно точно отображать поведение интересующих нас машин и систем.

ПРИНЦИПЫ ОПИСАНИЯ МАТРИЧНЫХ ЛОГИЧЕСКИХ СХЕМ

Обычно машины состоят из множества элементов, частей или подсистем. Состояние машины будет задаваться описанием машины с учетом ограничивающих условий. Для описания цифровых систем используются машины с дискретными, или квантованными, состояниями, которые противопоставляются аналоговым, или непрерывным, системам. Такой способ соответствует «общепараметрическому» описанию систем. Так, например, наиболее простым описанием системы может быть множество дискретных точек в «пространстве состояний» (рис. 4.1). Введение на этом множестве структуры связи обусловливает формирование топологического пространства. Если в рамках допустимой аппроксимации возможно выделить множество элементов, кото-

Рис. 4.1. Точки в «пространстве состояний».

рые не зависят от всех внешних элементов, то соответствующую систему называют *замкнутой*.

Существование некоторой машины определяется возможностью осуществления в ней устойчивого состояния. Если структура связи, относящаяся к множеству элементов, допускает возможность установления системы более чем в одно устойчивое состояние, то для системы возможно осуществление переходов из одного состояния в другое. Если же, кроме того, это множество имеет детерминированные или стохастические связи со средой, то макрозависимость этого множества от среды называется *поведением*, а структурное множество — *моделью*. Такие наиболее общие модели именуют *машинами состояний*. Способы описания поведения (переходов из состояния в состояние) систем обычно предполагают обозначение каждого состояния именем, меткой, признаком или числом. Если число элементов множества конечно, что свойственно всем реальным машинам, то состояние машины может быть обозначено с помощью упорядоченного множества точек. При этом исходная система отображается согласно некоторому предписывающему алгоритму или алгоритму упорядочивания. Такое отображение схематически изображено на рис. 4.2.

Вектор состояния эквивалентен упорядоченному множеству характеристических свойств, идентифицирующих «фактическое» состояние системы. Таким образом, вводится некоторый «язык», в терминах которого система может быть описана. Эти векторы состояний являются элементами модели поведения. Поведение системы представляется последовательностью векторов состояний, т. е. переходами обобщенной машины состояний. Такая последовательность может либо отображать историю поведения системы, либо предсказывать ее поведение, либо в общем слу-

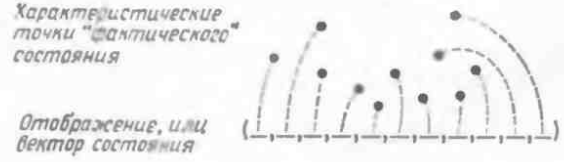


Рис. 4.2. Отображение пространства состояний в векторное пространство.

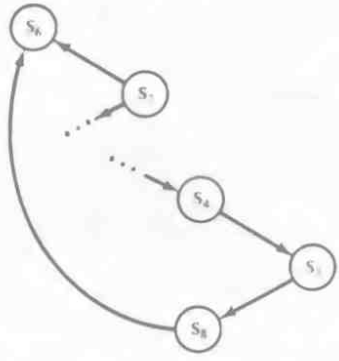


Рис. 4.3. Диаграмма последовательности состояний.

чае обеспечивать обе указанные возможности. Порядок последовательности состояний может быть представлен графически, например в виде диаграммы последовательности состояний (рис. 4.3).

Теперь предположим, что каждое состояние имеет единственное последующее состояние и что n состояний образуют некоторый цикл. Тогда поведение системы можно представить в табличной форме (рис. 4.4). Если данные таблицы упорядочить, то, как показано на рис. 4.5, вместо описания следующего состояния в ней может находиться указатель местонахождения в таблице следующего состояния.

Возможно, конечно, графическое изображение указателей в явном виде. Такое изображение выполнено на рис. 4.6 в виде стрелок. Подобная схема может быть представлена с использованием векторов состояний (рис. 4.7). Такое примитивное описание не является символьным, поскольку в данном случае используются фактические указатели вместо более абстрактных символов, которые указывают на следующее состояние в после-

Рис. 4.4. Таблица текущих и следующих состояний.

Номер или метка	Текущее состояние	Указатель следующего состояния
000	0 0 S_6 0 0 0 0 0	001
001	0 S_7 0 0 0 0 0 0	0
010		
011		
100	0 0 0 0 S_4 0 0 0	101
101	0 0 0 0 0 S_5 0 0	110
110	S_8 0 0 0 0 0 0 0	000

Рис. 4.5. Таблица состояний и указателей следующих состояний.

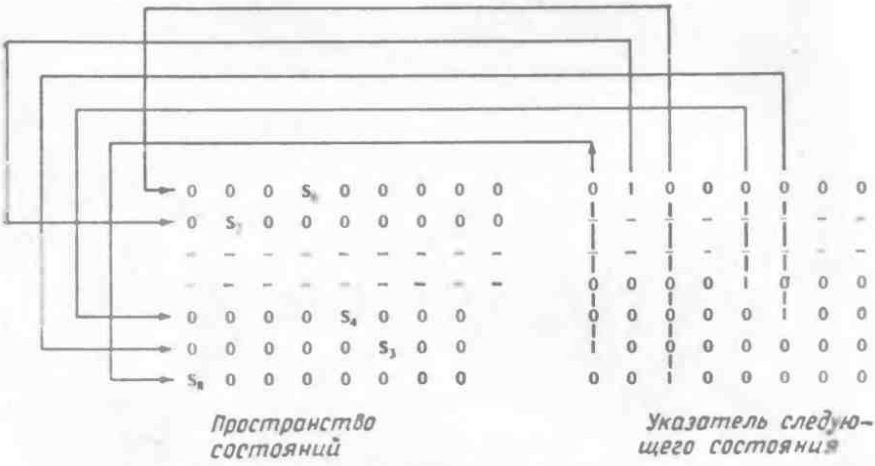


Рис. 4.6. Способ графического указания следующих состояний.

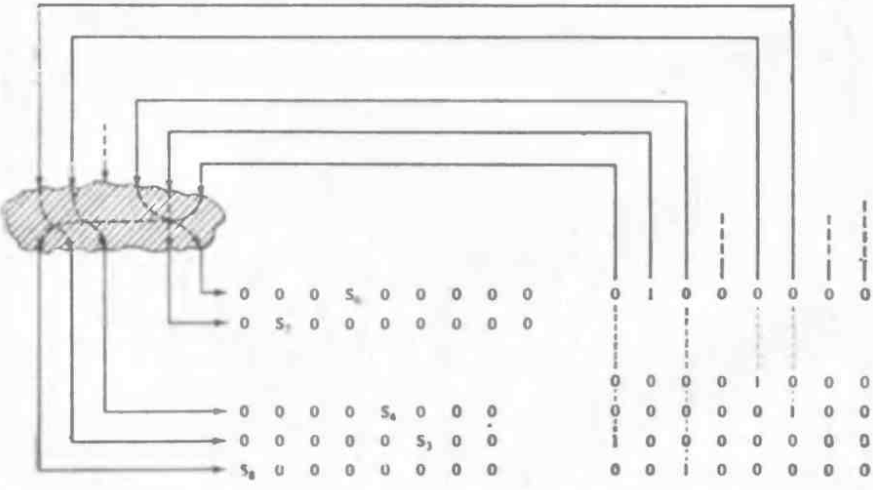


Рис. 4.7. Использование «области переходов» для выбора следующего состояния.

довательности. Представление состояний, данное на рис. 4.6, может быть выполнено более рациональным способом. Новый вариант представления, изображенный на рис. 4.7, отличается от предыдущего локализацией пересечений всех линий в области, которая выделена на рисунке специальным символом — темной зоной. Этот символ можно интерпретировать как некоторое представление процесса перехода, который действительно обусловливает осуществление изменения состояний системы. С использованием матричных алгебраических обозначений эволюция вектора состояния системы $X(t)$ может быть описана следующим матричным преобразованием:

$$X(t+dt) = U(t+dt, t)\,X(t).$$

Это преобразование определяет вектор состояния в момент времени $t+dt$, исходя из вектора состояния в момент времени t. Использованный нами символ (рис. 4.7) может здесь рассматриваться как графический эквивалент матрицы переходов $U(t+dt, t)$. Эту матрицу можно представить себе как некоторую «матрицу переключений».

Рис. 4.7 является избыточным в том смысле, что каждому состоянию соответствует отдельная стрелка, выходящая из области, отображающей процесс переходов, и, кроме того, каждое состояние представляется отдельным вектором состояния $(0, 0, 0, i, ..., 0)$. Рассматриваемая диаграмма может быть представлена проще с помощью матрицы «вход-выход» (рис. 4.8). Такие матрицы были введены в 1951 г. Уилксом в связи с развитием методов микропрограммирования.

Используя матрицу «вход-выход», можно получить еще одно матричное представление рассматриваемого преобразования (рис. 4.9). Для этого представления характерно явное изображение преобразования состояний с помощью связей, отмеченных жирными точками на определенных пересечениях. Таким образом, функция преобразования больше не выполняется в области перехода. Читатель должен самостоятельно убедиться в том, что

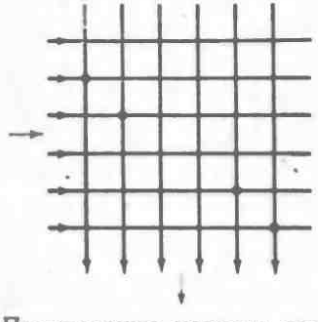


Рис. 4.8. Представление матрицы «вход-выход».

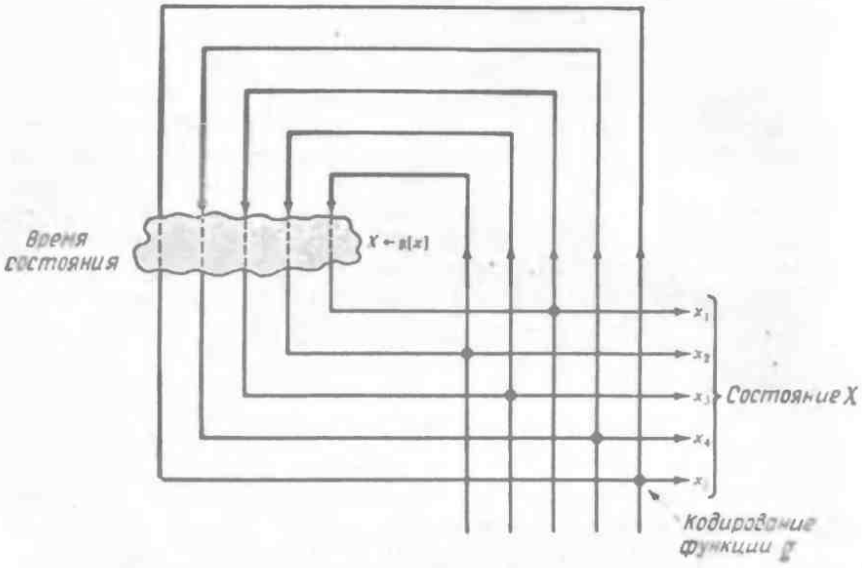


Рис. 4.9. Матричное представление преобразования текущего состояния в следующее состояние.

матричное представление, изображенное на рис. 4.9, и представление переходов в виде графов на рис. 4.10, эквивалентны.

Если матрице «вход-выход» соответствуют представленные ниже соотношения, то очевидно, что она является просто другим описанием комбинационной схемы. В гл. 3 мы рассматривали классы машин и показали, что комбинационные схемы представляют собой машины класса 0 и описываются соотношениями

$$I = f(Q) \quad \text{и} \quad X \longleftarrow 1.$$

Если в матричной логической схеме каждый узел помечен, например, так, как показано на рис. 4.11, то соединения можно интерпретировать согласно рис. 4.12. Таким образом, на выход-

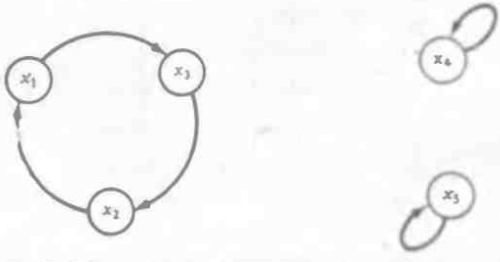


Рис. 4.10. Представление переходов с помощью графов.

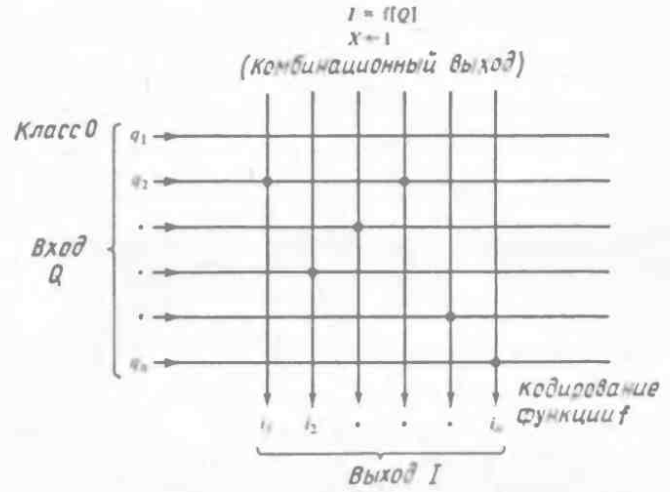


Рис. 4.11. Представление машины состояний класса 0 матричной логической схемой.

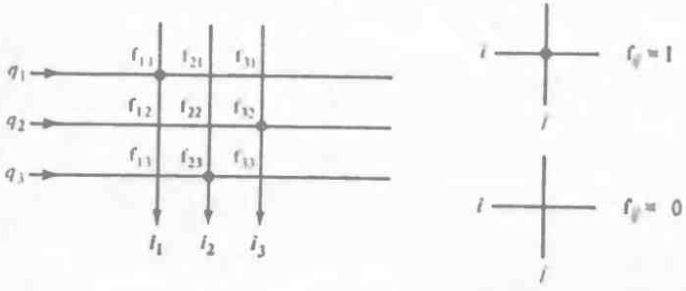


Рис. 4.12. Условные обозначения, используемые в матричных схемах.

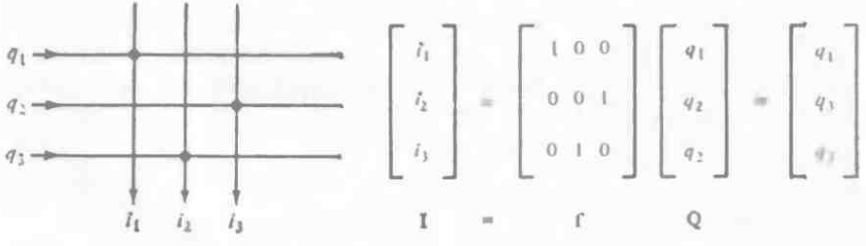


Рис. 4.13. Матричная логическая схема и эквивалентное преобразование с булевыми матрицами.

ных линиях получаются логические значения, определяемые логической операцией ИЛИ по всем узлам. Первой линии, например, будет соответствовать выражение

$$i_1 = f_{11}q_1 + f_{12}q_2 + f_{13}q_3.$$

В общем случае для k-й выходной линии будет справедливо выражение

$$i_k = \sum_{j=1}^{j=n} f_{kj}q_j.$$

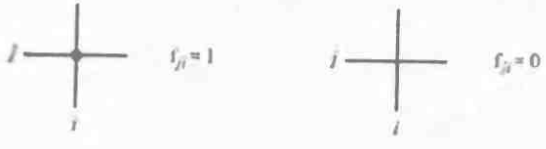
Таким образом, рассматриваемый формализм — «матричная логика» — явным образом отображает матричную операцию над входным вектором, результатом которой является выходной вектор (рис. 4.13).

РЕАЛИЗАЦИЯ «МАТРИЧНОЙ ЛОГИКИ»

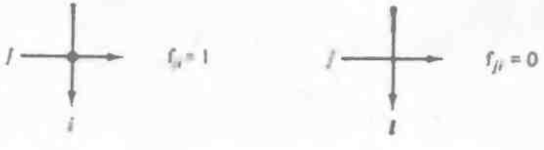
Выражение для выходной линии матрицы «вход-выход» имеет следующий вид:

$$i_k = f_{k1}q_1 + f_{k2}q_2 + \cdots + f_{kn}q_n,$$

где q_1 является входной, а i_k — выходной переменными. Математически это можно интерпретировать таким образом:



В другой интерпретации указывается направление соответствующих линий:



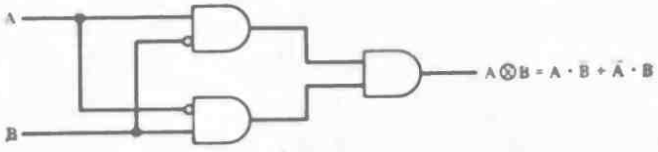
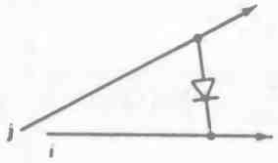


Рис. 4.14. Классическая схема ИСКЛЮЧАЮЩЕЕ ИЛИ, реализованная с помощью элементов И, ИЛИ и НЕ.

Однонаправленный характер этих соединений может быть физически реализован с помощью диодных элементов:



По этой причине логические матрицы часто называют *диодными матрицами.*

КОМБИНАЦИОННЫЕ ЛОГИЧЕСКИЕ СХЕМЫ

В гл. 3 была рассмотрена возможность представления комбинационной логической схемы машиной состояний класса 0. Теперь изучим матричные логические схемы, эквивалентные известным базовым логическим элементам. На рис. 4.14 изображена схема, выполняющая операцию ИСКЛЮЧАЮЩЕЕ ИЛИ. Эта схема является примером классической реализации указанной логической функции с помощью элементов И, ИЛИ и НЕ.

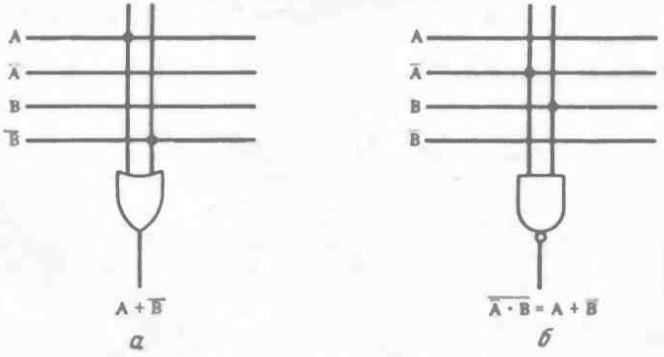


Рис. 4.15. Матрицы ИЛИ (*а*) и И-НЕ (*б*).

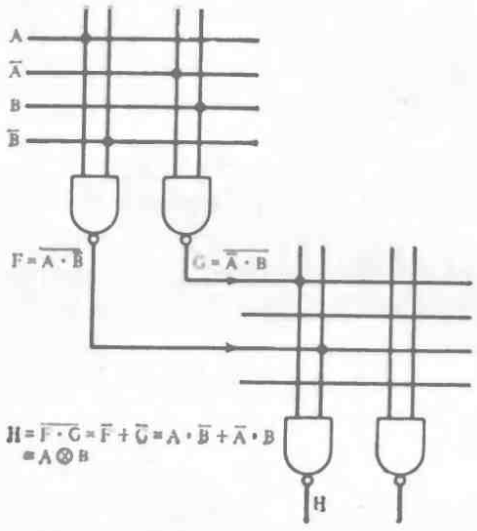


Рис. 4.16. Двухуровневая матричная логическая схема, реализующая функцию ИСКЛЮЧАЮЩЕЕ ИЛИ двух переменных.

В матричной логике выходу матричной схемы соответствует комбинационная логическая функция входных переменных, выражаемая либо только через операции И, либо только через операции ИЛИ (рис. 4.15). Булева функция общего вида содержит как операцию И, так и операцию ИЛИ. Следовательно, она не

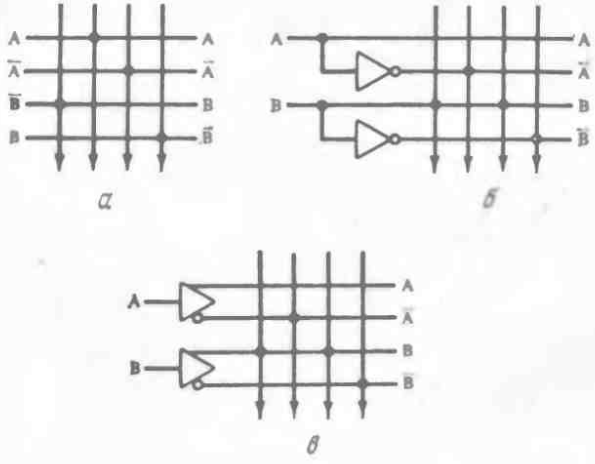


Рис. 4.17. Виды матричных логических схем: *а* — элементарная матрица; *б* — типичная матричная логическая схема; *в* — схема, эквивалентная схеме *б*.

может быть представлена ни матрицей типа ИЛИ, ни матрицей типа И-НЕ. Однако это не приводит к возникновению больших трудностей. Дело в том, что, согласно принципу дуальности (двойственности), выход матрицы можно рассматривать как функцию И или как функцию ИЛИ соответствующих входных значений. Следовательно, если использовать две матрицы, то одна из них может соответствовать функции И, а вторая — функции ИЛИ. Как изображено на рис. 4.16, выход первой матрицы может быть входом второй матрицы. Первую матрицу называют *матрицей логического умножения*, поскольку обычно на ее выходе получается логическое произведение входных переменных. Вторая матрица называется *матрицей логического сложения*, так как обычно на ее выходе получается логическая сумма входных переменных. Поскольку булевы функции могут быть представлены в виде суммы произведений переменных, двухуровневая матрица, обеспечивающая выполнение логического умножения и последующее суммирование, например двухуровневая матрица типа И-НЕ — ИЛИ, будет пригодна для реализации булевой функции входных переменных.

Прежде чем перейти к рассмотрению более сложных представлений логических матриц, упростим используемые символические обозначения. Очень часто в булевых выражениях содержатся инверсные члены, а так как при реализации логических схем основным ограничением является общее количество внешних выводов, то для их уменьшения целесообразно вводить в устройство только переменные, имеющие значение «истинно», и инверсные значения переменных получать в самом устройстве. Использование такого приема позволяет в некоторых случаях вдвое сократить необходимое число внешних выводов. Пример такой реализации матричной логической схемы изображен на рис. 4.17.

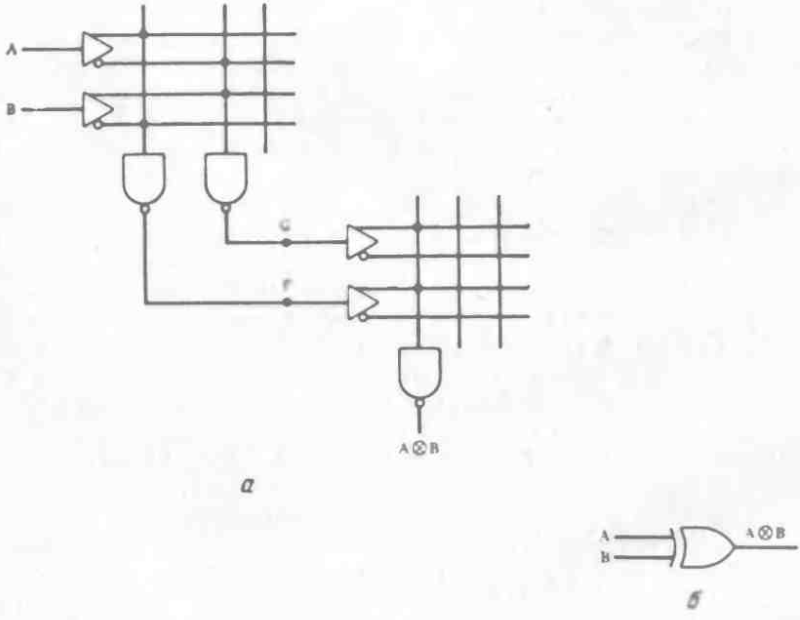
Вторым важным вопросом является выбор рациональной системы обозначений. Сложные матричные схемы содержат много линий. Если при этом используются стандартные правила изображения схем, согласно которым каждый вход логической схемы представляется отдельно, то затрудняется процесс изображения и восприятия сложных схем. Поэтому вместо стандартного изображения логического элемента (рис. 4.18, *а*) используется эквивалентное изображение (рис. 4.18, *б*).

Матричная схема, реализующая функцию ИСКЛЮЧАЮЩЕЕ ИЛИ, приведена на рис. 4.16. На рис. 4.19, *а* представлено изображение этой матричной схемы с использованием новых обозначений. Даже при использовании упрощенных правил изображения матричных логических схем представление функции ИСКЛЮЧАЮЩЕЕ ИЛИ получается довольно сложным по сравнению со стандартным эквивалентным изображением

а *б*

Рис. 4.18. Стандартное обозначение схемы И (*а*) и ее эквивалентное обозначение с использованием матрицы (*б*).

(рис. 4.19, *б*). Однако, подобно описаниям алгоритмической машины состояний, описание всех машин класса 0 с использованием матричной логики является одинаковым. Преимущества матричной логики определяются прежде всего регулярностью структуры соответствующих матричных схем, что делает их пригодными для автоматизированного проектирования и производства. При использовании матричной логики вместо других логических систем, основанных на применении схем малой и средней степени интеграции (1—300 логических элементов на кристалле), достигается сокращение числа выводов, уменьшение количества корпусов и потребляемой мощности. При проектировании сверхбольших интегральных схем преимущественно применяется матричная логика.



а *б*

Рис. 4.19. Матричная логическая схема, реализующая функцию ИСКЛЮЧАЮЩЕЕ ИЛИ (*а*) и обозначение элемента ИСКЛЮЧАЮЩЕЕ ИЛИ (*б*).

ПРОГРАММИРУЕМЫЕ МАТРИЦЫ

Проектировать цифровые системы с помощью матричной логики предложил Уилкс. Первым практическим применением БИС с матричной логикой явилось их использование фирмой Texas Instruments при разработке различных вычислительных устройств. Эти устройства не являлись программируемыми, поскольку соединения в матрице производились посредством металлизации по методу масочного программирования, и пользователь не мог сам определять логические операции, выполняемые матричной схемой. Фирма National Semiconductor впервые освоила выпуск матричных схем, проектируемых с использованием метода масочного программирования по схеме, задаваемой пользователем. Такие матричные логические схемы стали называть *программируемыми логическими матрицами (ПЛМ)*. Пользователь предоставляет фирме-изготовителю схему соединений, согласно которой реализуется программируемая логическая матрица для проектируемого устройства. Схематическое изображение программируемой логической матрицы дано на рис. 4.20.

Использование логических матриц с масочным программированием оказывается экономически оправданным при массовом производстве, однако при ограниченном выпуске устройств такая технология приводит к существенным временны́м и денежным затратам. Поэтому применение матриц с масочным программированием на стадии опытного конструирования ока-

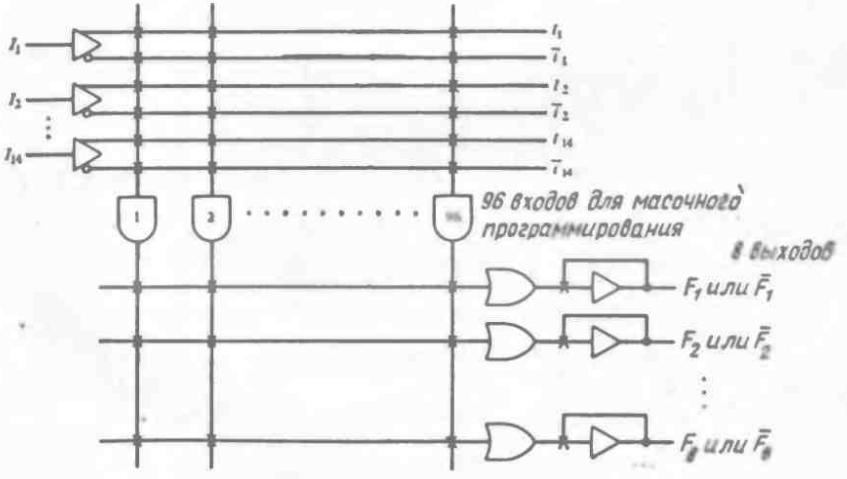


Рис. 4.20. Первая стандартная схема программируемой логической матрицы DM7575 фирмы National Semiconductor.

зывается слишком расточительным. Неоспоримыми достоинствами обладают *логические матрицы с программируемым полем (ЛМПП)*. Пользователи приобретают их «чистыми», т. е. не настроенными на выполнение определенных функций. Пользователь в данном случае не выдает фирме-изготовителю заказ на программирование матрицы и соответствующую схему, а программирует их сам с помощью относительно дешевой системы программирования. Существуют развитые семейства или классы устройств, основанные на программируемых логических матрицах, которые классифицируют на основании методов и принципов программирования логических матриц. Базисная структура, общая для типичных логических программируемых матриц, представлена на рис. 4.21.

В двухуровневой матричной структуре пользователь может программировать либо матрицу типа И, либо матрицу типа ИЛИ, либо обе матрицы. В частном случае одна из матриц может быть фиксированной. Три основных класса логических устройств с программируемыми матрицами представлены на рис. 4.22. Все эти устройства могут программироваться либо пользователем с помощью системы программирования, либо фирмой-изготовителем по заказу пользователя.

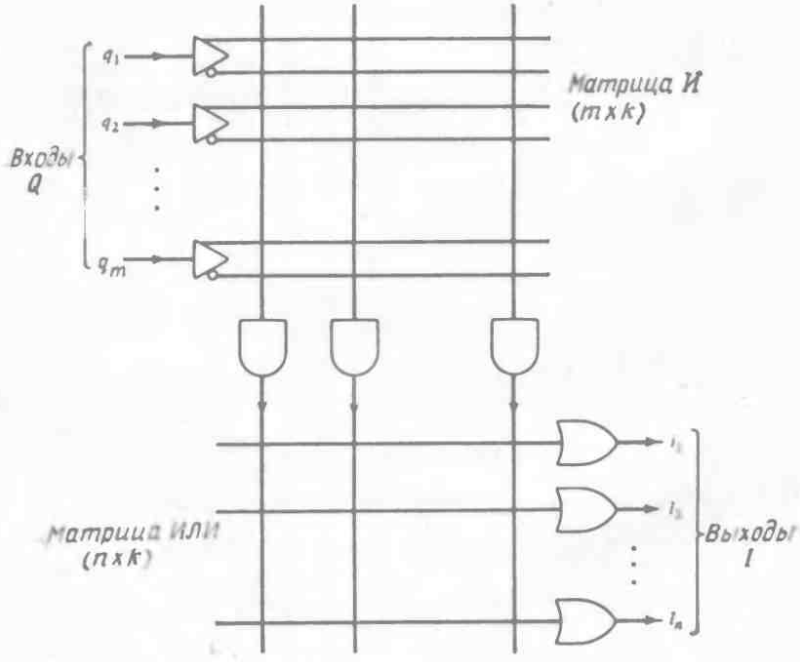


Рис. 4.21. Схема двухуровневой логической матрицы И—ИЛИ.

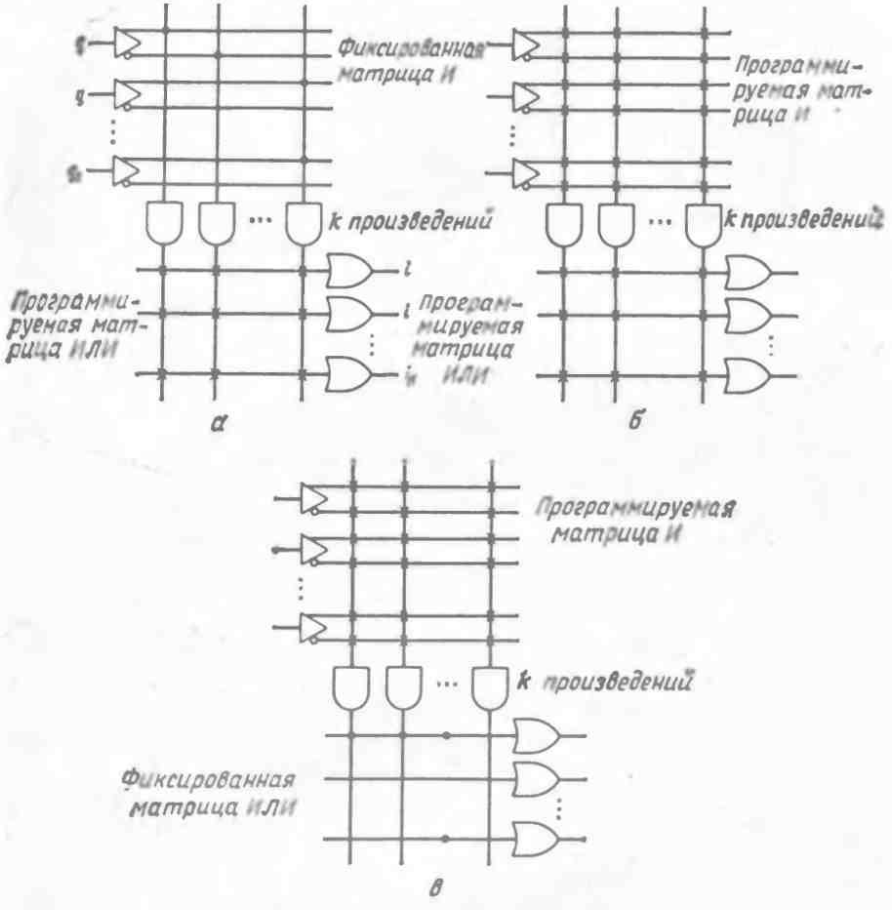


Рис. 4.22. Три основных класса устройств, построенных на программируемых логических матрицах: *а* — программируемое ПЗУ с фиксированной матрицей И и программируемой матрицей ИЛИ; *б* — логическая матрица с программируемым полем — матрицы И и ИЛИ программируемые; *в* — программируемая матричная логическая схема с программируемой матрицей И и фиксированной матрицей ИЛИ.

Кроме указанных трех основных классов устройств, существуют еще два подкласса: *программируемая матрица вентилей (ПМВ)* — в этой схеме отсутствует матрица типа ИЛИ, а также *программируемый мультиплексор (ПМ)*, в котором матрица типа ИЛИ фиксированна и частично фиксированна матрица типа И. В следующей ниже таблице представлены все упомянутые выше устройства:

Устройство	Матрица типа И	Матрица типа ИЛИ
ПЗУ/ППЗУ[1]	Фиксированная	Программируемая
ЛМПП	Программируемая	»
ПМВ	»	Отсутствует
ПМ	Фиксированная/Программируемая	Фиксированная
Программируемая матричная схема	Программируемая	Фиксированная

[1] ППЗУ — программируемое постоянное запоминающее устройство.

ДИОДНЫЕ ЛОГИЧЕСКИЕ МАТРИЦЫ

Однонаправленная проводимость диодных элементов позволяет соединить с их помощью две идентичные линии таким образом, что одна из них может однозначно идентифицироваться как входная линия, а другая — как выходная линия. При таком соединении сигналы, поступающие на входные линии, воздействуют на выходную линию, однако обратное воздействие невозможно. Схема, изображенная на рис. 4.23, *а*, поясняет принцип действия диодной матрицы. На горизонтальную линию через резистор R подается напряжение V_{cc}, кроме того, горизонтальная линия связывается посредством диодов с каждой входной вертикальной линией. Если на некоторой входной вертикальной линии установится низкий уровень напряжения, то прямое сопротивление соответствующего диода станет столь малым по сравнению с сопротивлением резистора R, что все напряжение будет падать на резисторе R, и на выходной горизонтальной линии установится низкое напряжение. Это справедливо в отношении каждой входной линии. Следовательно, логическое значение сигнала на выходной горизонтальной линии будет определяться выражением

$$\text{Логическое значение на выходе} = \overline{A} + \overline{B} + \overline{C} + \cdots$$

Согласно закону де Моргана, это выражение можно записать следующим образом:

$$\text{Логическое значение на выходе} = A \cdot B \cdot C \cdots$$

Полученное выражение означает, что на выходе диодной матрицы вырабатывается значение логической функции И от входных сигналов.

Еще один вариант диодной матрицы изображен на рис. 4.23, *б*. Здесь входными являются горизонтальные линии, а вертикальная линия будет играть роль выходной. Если на каком-либо входе появится высокий уровень напряжения, то пря-

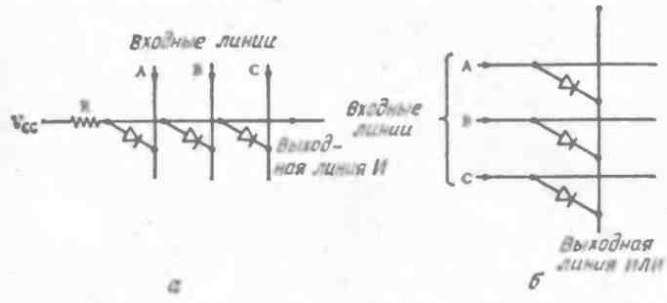


Рис. 4.23. Диодные логические соединения типа И (*а*) и ИЛИ (*б*).

мое сопротивление диода станет малым и на выходной линии также установится высокий уровень напряжения. Следовательно, логическое значение сигнала на выходной линии будет определяться выражением

$$\text{Логическое значение на выходе} = A + B + C$$

Это выражение означает, что на соответствующем выходе диодной матрицы вырабатывается значение логической функции ИЛИ от входных сигналов. Вследствие симметрии диодной матрицы проектировщик может по-своему усмотрению считать вертикальные линии, т. е. столбцы матрицы, входными либо выходными линиями. Пример, иллюстрирующий такую возможность, представлен на рис. 4.24.

Одна из первых программируемых логических матриц, появившаяся на рынке в 1973 г., была идентична описанной выше логической матрице. Фирма Rockwell International разработала кремниевую интегральную схему на сапфировой подложке 15900, представляющую собой диодную матрицу типа И—ИЛИ,

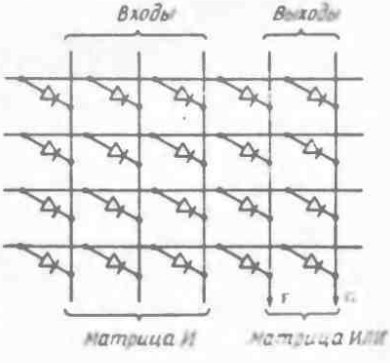


Рис. 4.24. Диодная логическая матрица типа И—ИЛИ.

которая имеет 128 строк и 46 столбцов. Некоторое число n вертикальных линий может играть роль входных линий; все оставшиеся вертикальные линии (46—n) или часть из них назнача-

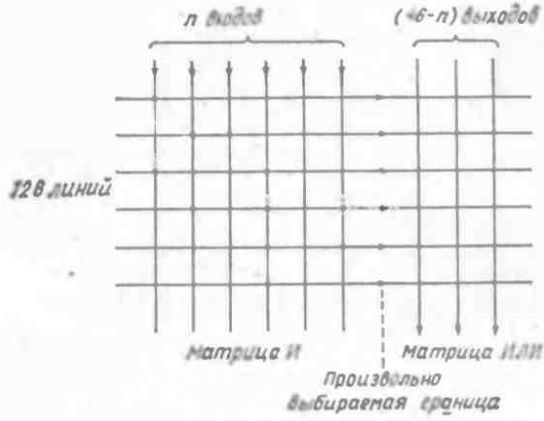


Рис. 4.25. Структура программируемой диодной логической матрицы фирмы Rockwell International.

ются выходными линиями. Такая необычайная гибкость распределения функций вертикальных линий матрицы не позволяет производить инвертирование значений вводимых переменных внутри самой схемы, и поэтому инверсные значения входных переменных должны подаваться отдельно на входы матрицы. На рис. 4.25 дано схематическое изображение описываемого устройства — программируемой диодной логической матрицы.

Напряжения на горизонтальных линиях описанных выше диодных матриц зависят от напряжений на всех входных вертикальных линиях, которые соединены с этими горизонтальными линиями посредством диодов. Чтобы разорвать связь между некоторой горизонтальной линией матрицы и некоторой входной вертикальной линией (или связь выходной вертикальной линии

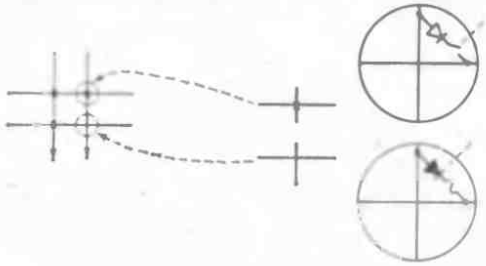


Рис. 4.26. Условные обозначения, используемые при изображении программируемых логических матриц с плавкими связями.

и некоторой горизонтальной линии), следует удалить из матрицы соответствующий диод. Эта операция выполняется на производстве с помощью лазерной установки, которая лазерным лучом физически разрушает подлежащие удалению диоды. Позже для обеспечения программирования диодных матриц стали использовать матрицы с плавкими связями, условные обозначения которых представлены на рис. 4.26.

Фирма Monolithic Memories выпускает программируемые

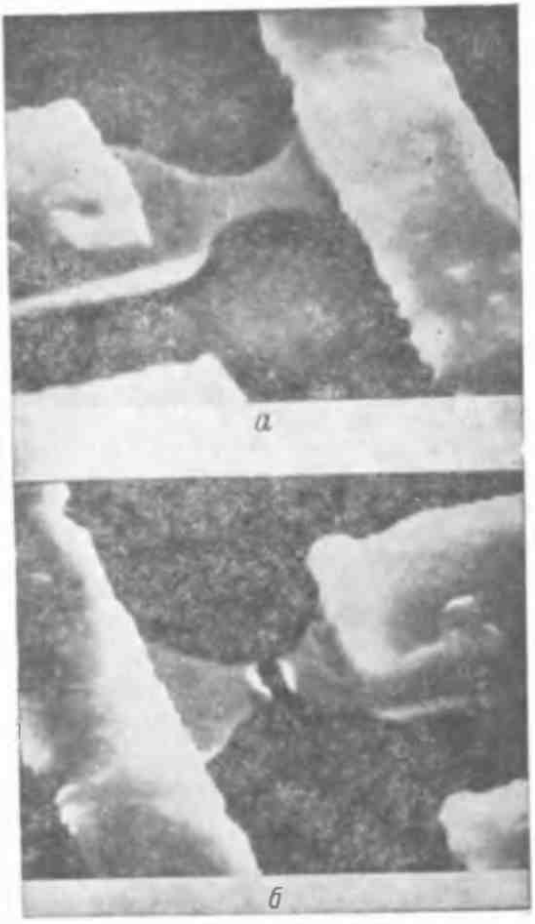


Рис. 4.27. Фотографии нерасплавленной (*а*) и расплавленной (*б*) плавких связей. *(С разрешения* Monolithic Memories, Inc.)

матричные схемы (ПМС), используя биполярную технологию, которая применяется для изготовления программируемых ПЗУ (на ТТЛ-схемах) с плавкими связями. Фотографии существующей и разрушенной плавких связей представлены на рис. 4.27. Плавкая связь представляет собой диодную связь между входной и выходной линиями матрицы. Если связь не нарушена, то диодное соединение существует, в противном случае выходная и входная линии не имеют никакого соединения. Для указания того, что все имеющиеся плавкие связи сохранены, используется следующее обозначение:

Линия терма-произведения, все плавкие связи которой уничтожены, всегда имеет высокий уровень напряжения. Так как

входные буферы ПЛМ и ПМС представляют для каждой переменной как значение «истинно», так и значение «ложно», линия терма-произведения, для которой сохранены связи со всеми входными линиями, всегда будет иметь низкий уровень напряжения. Таким образом, пользователь выполняет программирование матрицы посредством физического разрушения связей в заранее определенных узлах. Программирование производится с помощью специального программирующего устройства, которое по своим электрическим характеристикам должно соответствовать матричным схемам, подлежащим программированию.

ПРОГРАММИРУЕМЫЙ МУЛЬТИПЛЕКСОР

Матрица типа И — ИЛИ, содержащая фактически две матрицы — матрицу типа И и матрицу типа ИЛИ, в частном случае может использоваться и без матрицы типа ИЛИ. Если из матрицы типа И — ИЛИ исключена матрица типа ИЛИ, то оставшаяся матрица типа И представляет собой *программируемую матрицу вентилей*. Если же матрица типа ИЛИ сохраняется, то имеющаяся в этом случае конфигурация матриц может служить основой для построения *программируемого мультиплексора*. В моей ранее выпущенной книге (Кл., 1980) был описан 8-входовый мультиплексор 74151. Мультиплексор 29693 фирмы Raytheon, появившийся в 1977 г., был первым программируемым мультиплексором. Он состоял из четырех мультиплексоров 74LS151, каждый из которых по входу был связан с матрицей типа И с плавкими (программируемыми) связями, имеющей 10 входов и 8 выходов. Конструктивно мультиплексор 29693 реализуется в корпусе с двухрядным расположением 20 выводов; его условное изображение дано на рис. 4.28.

Более подробная схема программируемого мультиплексора представлена на рис. 4.29. Каждый из четырех простых мультиплексоров имеет один выход с тремя состояниями. Все выходы управляются сигналами, подаваемыми по общей линии.

В каждом мультиплексоре линии выбора канала ($S_0 - S_2$)

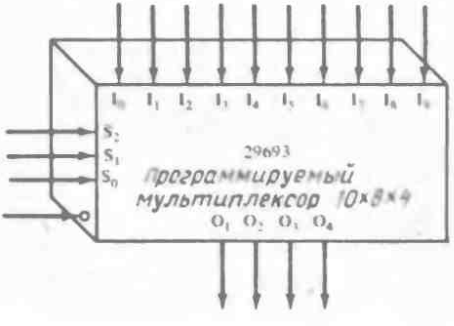


Рис. 4.28. Обозначение программируемого мультиплексора 29693 фирмы Raytheon.

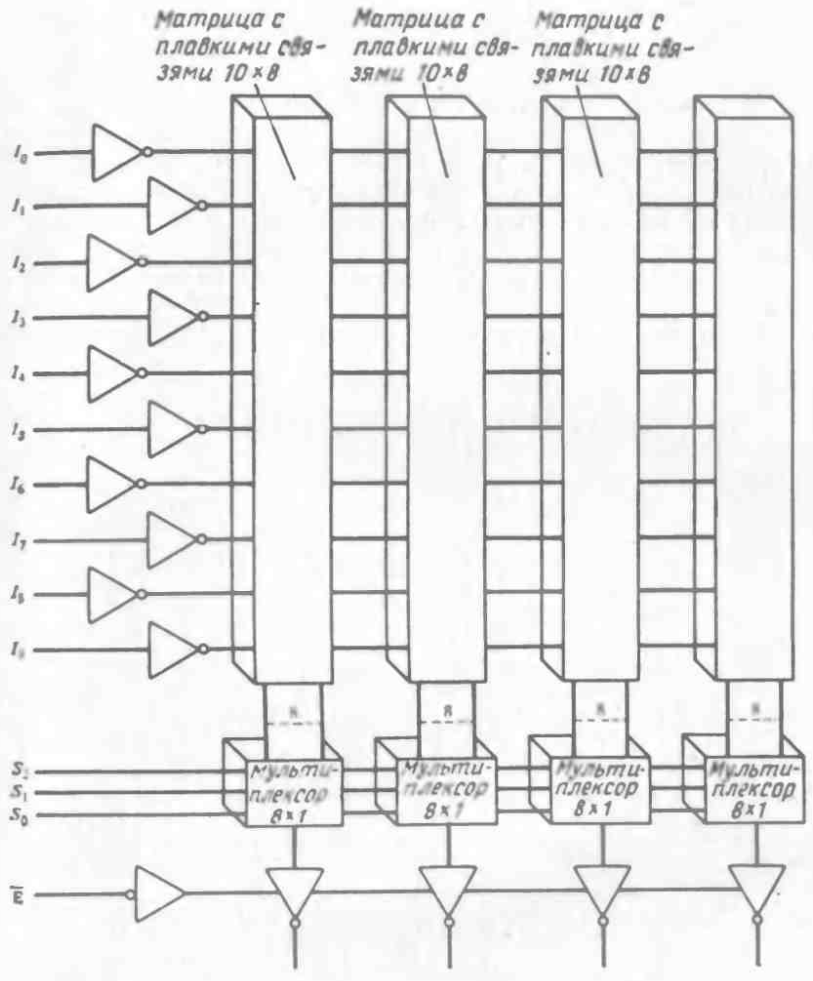


Рис. 4.29. Структурная схема 10-входового программируемого мультиплексора фирмы Raytheon.

используются для выбора одного из восьми входов мультиплексора. В исходном состоянии каждый вход 8-входового простого мультиплексора соединен плавкими связями со всеми 10 входами матрицы (т. е. входами программируемого мультиплексора), при этом на выходах матрицы получается логическое произведение входных переменных. Заметим, однако, что сигналы на каждом входе матрицы инвертируются и, следовательно, на входе простого мультиплексора фактически получается логическое значение функции ИЛИ от 10 входных переменных (сигна-

лов) программируемого мультиплексора; последнее утверждение непосредственно следует из теоремы де Моргана

$$\overline{A}\cdot\overline{B}\cdot\overline{C}=\overline{A+B+C}$$

Устройство программируется посредством удаления соответствующих плавких связей, что приводит к отключению входов матрицы от входов простых мультиплексоров. Обычно девять из десяти связей на каждой входной линии программируемого мультиплексора остаются разомкнутыми, но иногда для получения логических значений функции ИЛИ двух или большего количества входных переменных сохраняется соответствующее

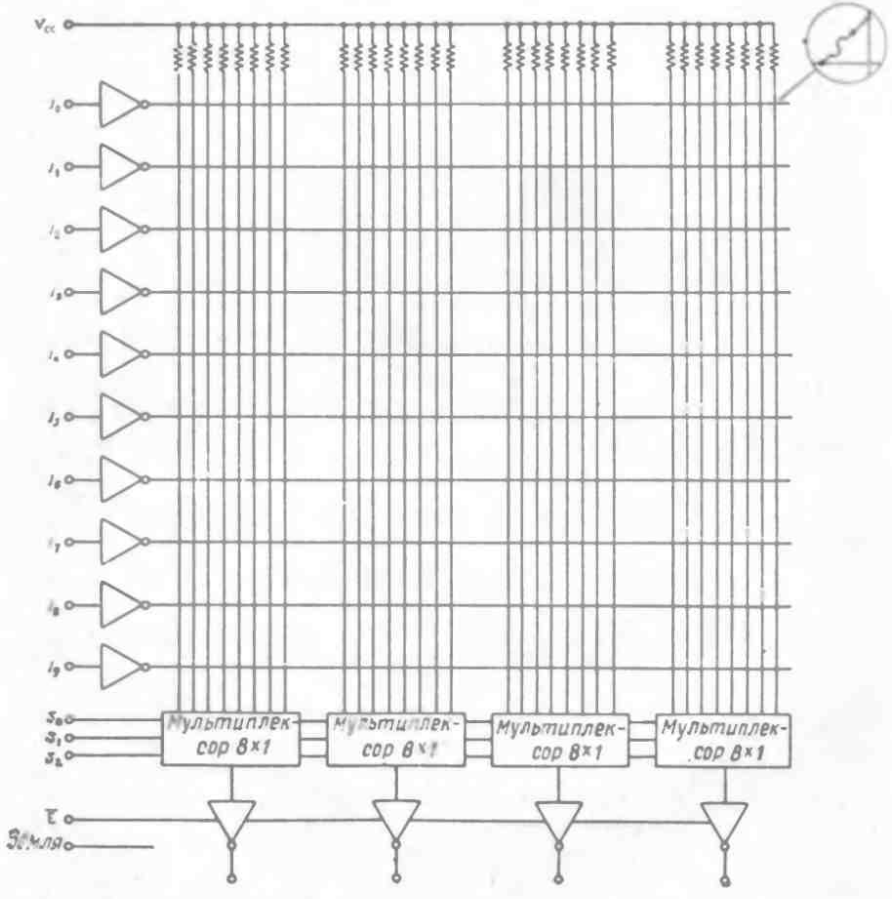


Рис. 4.30. Программируемый мультиплексор до выполнения операции программирования.

количество связей. Спецификации электрических и временных характеристик процесса программирования (физического уничтожения плавких связей) описываются в технической документации, поставляемой фирмами-изготовителями. Многие коммерческие установки для программирования ППЗУ имеют специально приспособленные средства для программирования программируемых мультиплексоров. Схема программируемого устройства, которое еще не было подвергнуто операции программирования, изображена на рис. 4.30; здесь использованы стандартные обозначения для матриц с плавкими связями.

ПРИМЕР ПРОГРАММИРУЕМОГО МУЛЬТИПЛЕКСОРА

Рассмотрим устройство, при проектировании которого Бреден использовал интегральную схему 29693, являющуюся программируемым мультиплексором. Известно, что для вычисления контрольных разрядов с целью обнаружения ошибок в последовательных потоках данных широко используются полиномы. Было определено несколько стандартных полиномов (CRCC16, MKKTT и LRCC16), схемы вычисления которых приведены на рис. 4.31.

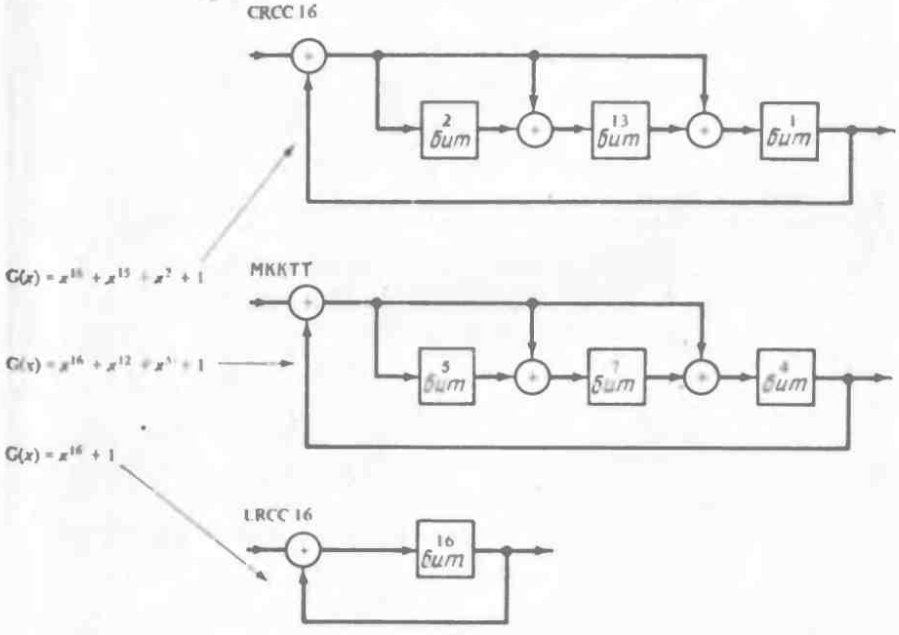


Рис. 4.31. Стандартные схемы вычисления полиномов, используемые для обнаружения ошибок.

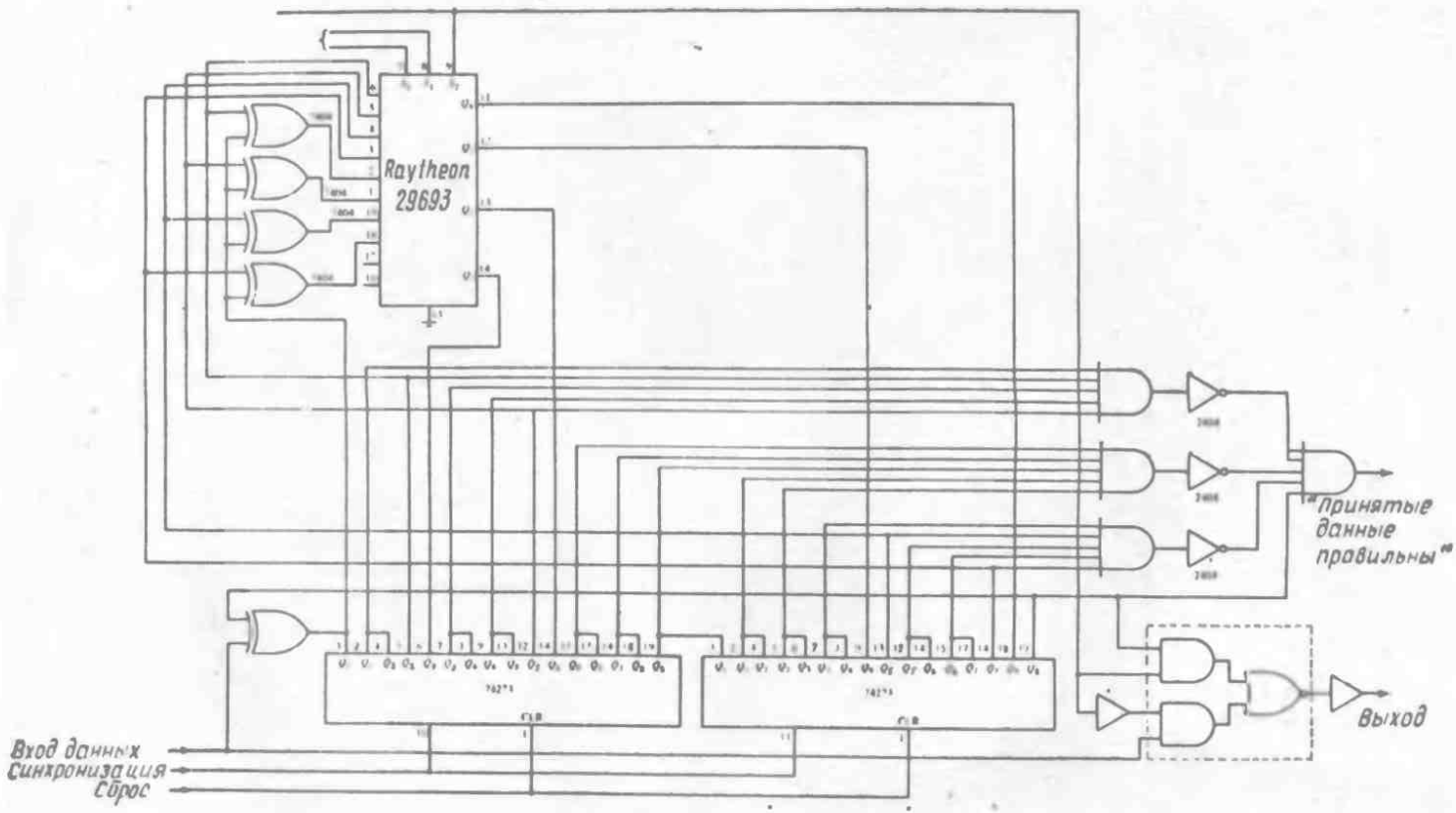


Рис. 4.32. Полиномиальный преобразователь Бредена, обеспечивающий обнаружение ошибок. (*С разрешения* Electronic Engineering Times.)

Для формирования 16-разрядного сдвигового регистра используются две интегральные схемы типа 74273, содержащие по восемь триггеров D-типа (рис. 4.32). В программируемом мультиплексоре 29693 выбор используемого полинома производится с помощью сигналов, поступающих на линии выбора S_0 и S_1. Для передаваемого слова данных с помощью схемы вычисления соответствующего полинома определяется контрольный бит, который добавляется к потоку данных, когда на входе S_2 появляется высокий уровень напряжения. Поступающие на вход устройства данные, содержащие 16 бит данных и контрольный бит, проверяют по используемому полиному. Если при передаче данных ошибка не произошла, то результатом деления будет 0. Сигнал «Принятые данные правильны» появится на выходе пятивходового элемента ИЛИ-НЕ, ему соответствует высокий уровень напряжения. В следующей таблице приведено соответствие значений логических сигналов на входах S_0, S_1 и S_2 мультиплексора и функций, выполняемых устройством обнаружения ошибок.

S_2	S_1	S_0	Функция
0	0	0	Прием или передача. Кодирование по полиному CRCC16
0	0	1	Прием или передача. Кодирование по полиному MKKTT
0	1	0	Прием или передача. Кодирование по полиному LRCC16
1	0	0	Передача. Определение контрольного символа по полиному CRCC16
1	0	1	Передача. Определение контрольного символа по полиному MKKTT
1	1	0	Передача. Определение контрольного символа по полиному LRCC16

Последовательности действий, выполняемых при передаче и приеме данных, сведены в следующую таблицу:

Передача	Прием
Выбор полинома (S_0, S_1)	Выбор полинома (S_0, S_1)
Очистка сдвигового регистра	Очистка сдвигового регистра
Передача данных ($S_2=0$)	Прием данных и контрольного бита ($S_2=0$)
Передача контрольного знака ($S_2=1$)	Контроль передачи (выход=1)

На рис. 4.33 изображена схема программируемого мультиплексора, прошедшего операцию программирования. Зачерненными кружками на схеме обозначены разрушенные связи.

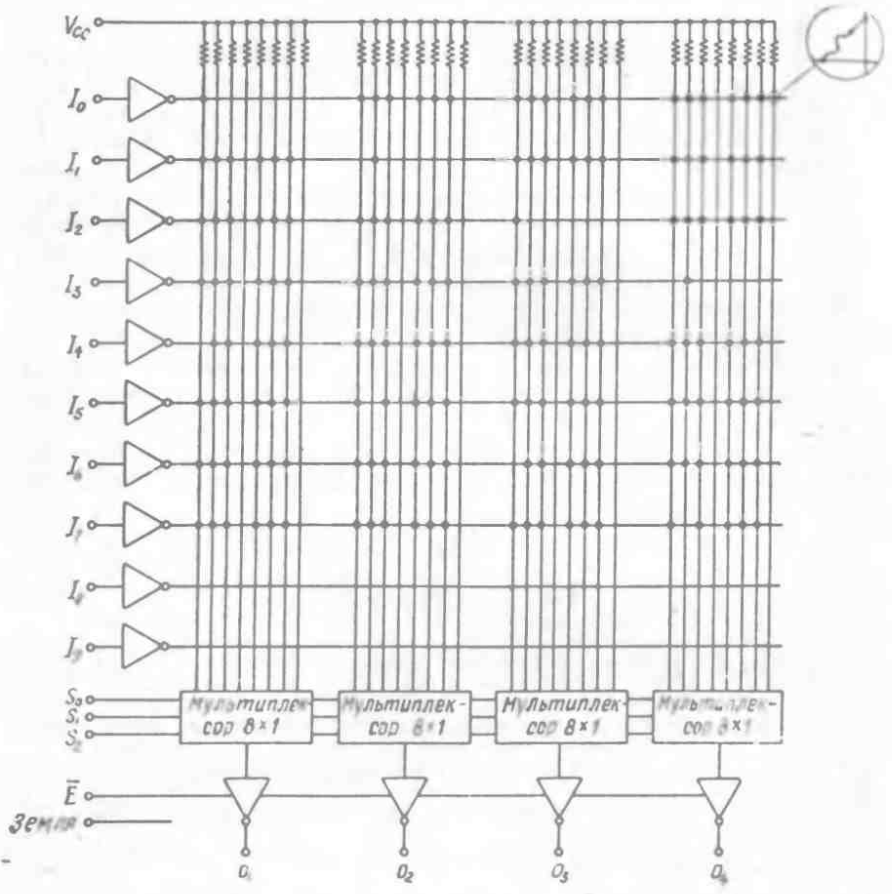


Рис. 4.33. Программируемый мультиплексор Бредена, прошедший операцию программирования и подготовленный к работе в схеме обнаружения ошибок. (*С разрешения* Electronic Engineering Times.)

Размеры матриц обусловлены конструктивными ограничениями электронных цифровых устройств. В связи с этим проектировщик может столкнуться со следующими ограничениями (см. рис. 4.34):

1) число входов (m) матрицы меньше числа входных переменных;

2) число имеющихся вертикальных линий (k) меньше, чем количество подлежащих вычислению термов;

3) число выходов (n) матрицы меньше требуемого.

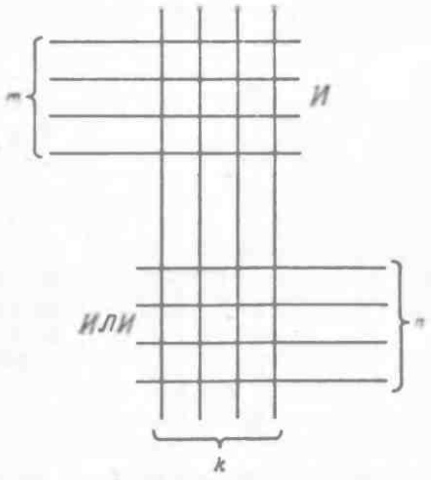


Рис. 4.34. Структура двухуровневой логической матрицы и параметры, характеризующие ее размеры.

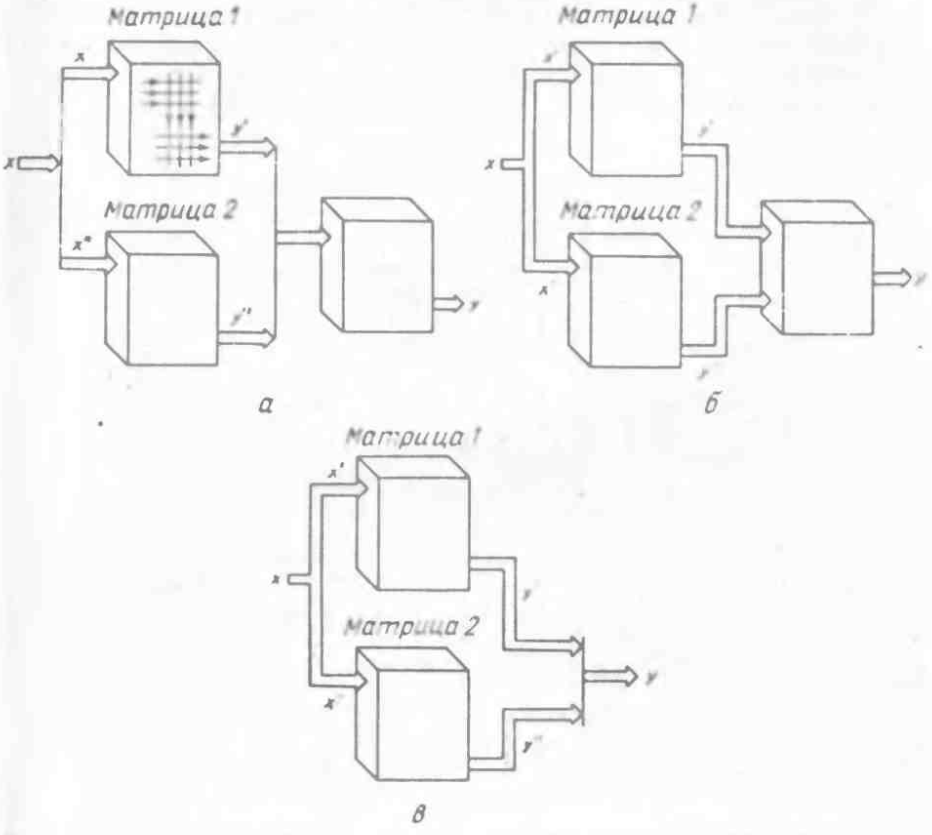


Рис. 4.35. Варианты расширения матричных схем: *а* — увеличение числа входов матрицы; *б* — увеличение количества вычисляемых термов; *в* — увеличение числа выходов матрицы.

На рис. 4.35 дано графическое пояснение принципа расширения. Общего решения задачи увеличения числа переменных состояния посредством увеличения количества матриц не существует.

ЛОГИЧЕСКИЕ УСТРОЙСТВА С ПРОГРАММИРУЕМЫМИ МАТРИЦАМИ

В 1978 г. фирма Monolithic Memories начала производить логические схемы с программируемыми матрицами; в семейство таких схем вошло примерно 15 различных устройств. Целью разработки семейства было уменьшение числа корпусов интегральных схем в проектируемой логической схеме от 5 или 10 до 1. Устройства семейства могут программироваться пользователем. Кроме того, они имеют выходы с тремя состояниями, регистровые выходы и программируемый ввод-вывод. Все эти устройства могут работать в составе микропроцессорных систем. Устройства семейства компонуются из нескольких базисных структур, подобных структуре, изображенной на рис. 4.36.

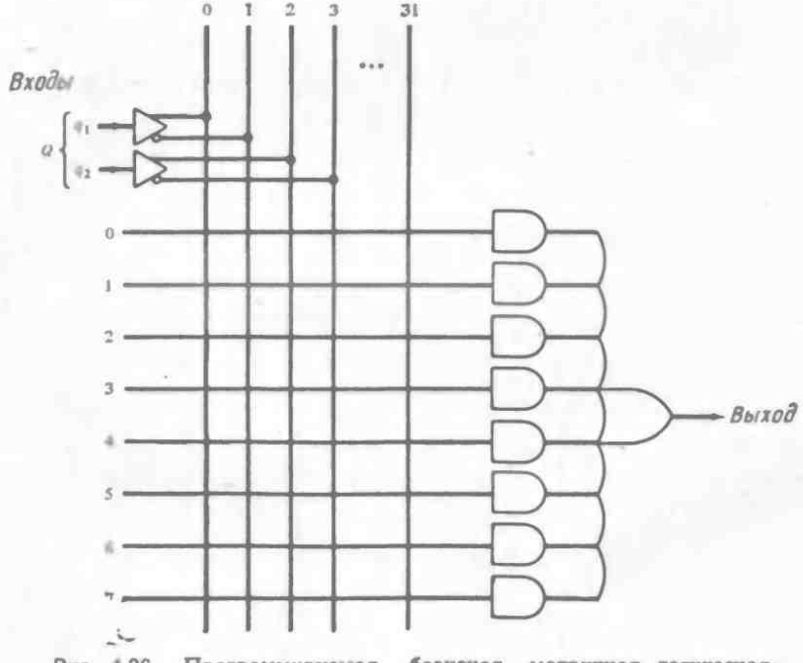


Рис. 4.36. Программируемая базисная матричная логическая ячейка.

РЕАЛИЗАЦИЯ УНИВЕРСАЛЬНОЙ МАШИНЫ СОСТОЯНИЙ

Базисная структура универсальной машины состояний состоит из комбинационных логических схем и памяти, включенной в цепи обратной связи, которые соединяют некоторые выходы с некоторыми входами машины. Структура универсальной машины состояний, реализованной с использованием программируемых логических матриц, представлена на рис. 4.37. Одной из структур, составляющих семейство логических схем с программируемыми матрицами фирмы Monolithic Memories, является элементарная универсальная машинная ячейка, схема которой изображена на рис. 4.38. Логическое значение каждого терма-произведения записывается в триггер D-типа по переднему фронту синхроимпульса; как логическое значение терма, так и его инверсное значение передаются по цепи обратной связи, что позволяет использовать их в качестве входных переменных матрицы. Все регистровые выходы ячеек, содержащихся в одном корпусе, синхронизируются сигналами, подаваемыми по общей линии. «Выход состояния» триггера D-типа буферируется элементом с тремя устойчивыми состояниями. Все эти буферы в

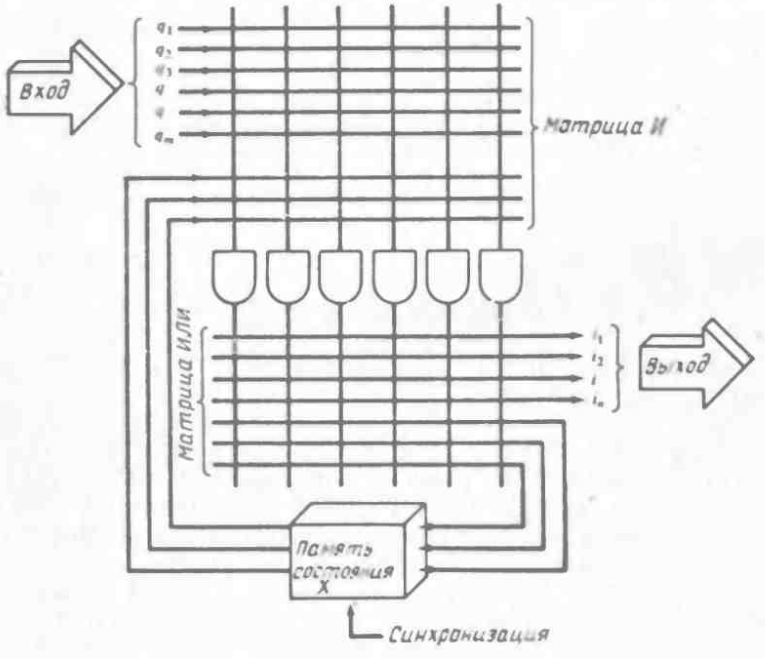


Рис. 4.37. Реализация универсальной машины состояний с помощью программируемых логических матриц.

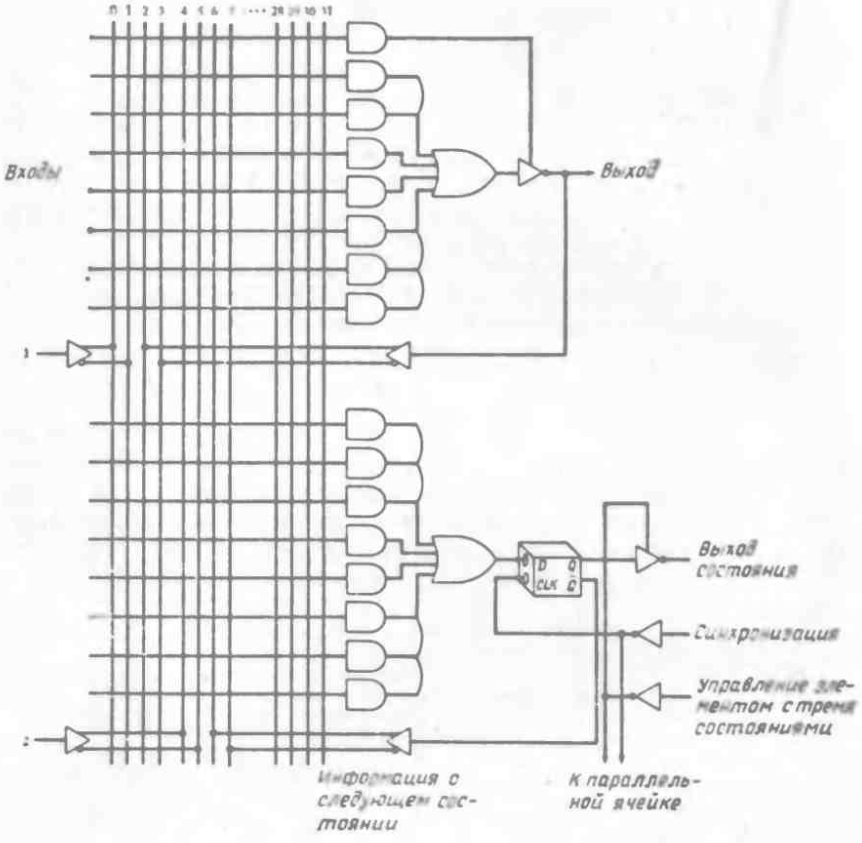


Рис. 4.38. Программируемая матричная схема элементарной универсальной машинной ячейки. (Четыре таких ячейки входят в устройство PAL16R4 фирмы Monolithic Memories.)

пределах одного корпуса управляются сигналами, поступающими по общей линии управления. Отметим, что выход комбинационной секции элементарной универсальной машины, изображенной на рис. 4.38, управляется сигналом, логическое значение которого определяется термом-произведением любых входных переменных. Такой выход не имеет общего управления, если оно не обеспечивается внутриматричными соединениями. Когда сигнал, определяемый соответствующим термом-произведением, отпирает выходной буфер, сигнал, соответствующий суммарному терму, появляется на выходе и, кроме того, подается по цепи обратной связи как входная переменная матрицы. Когда выходной буфер находится в состоянии высокого импеданса, выход схемы может служить в качестве входа матрицы.

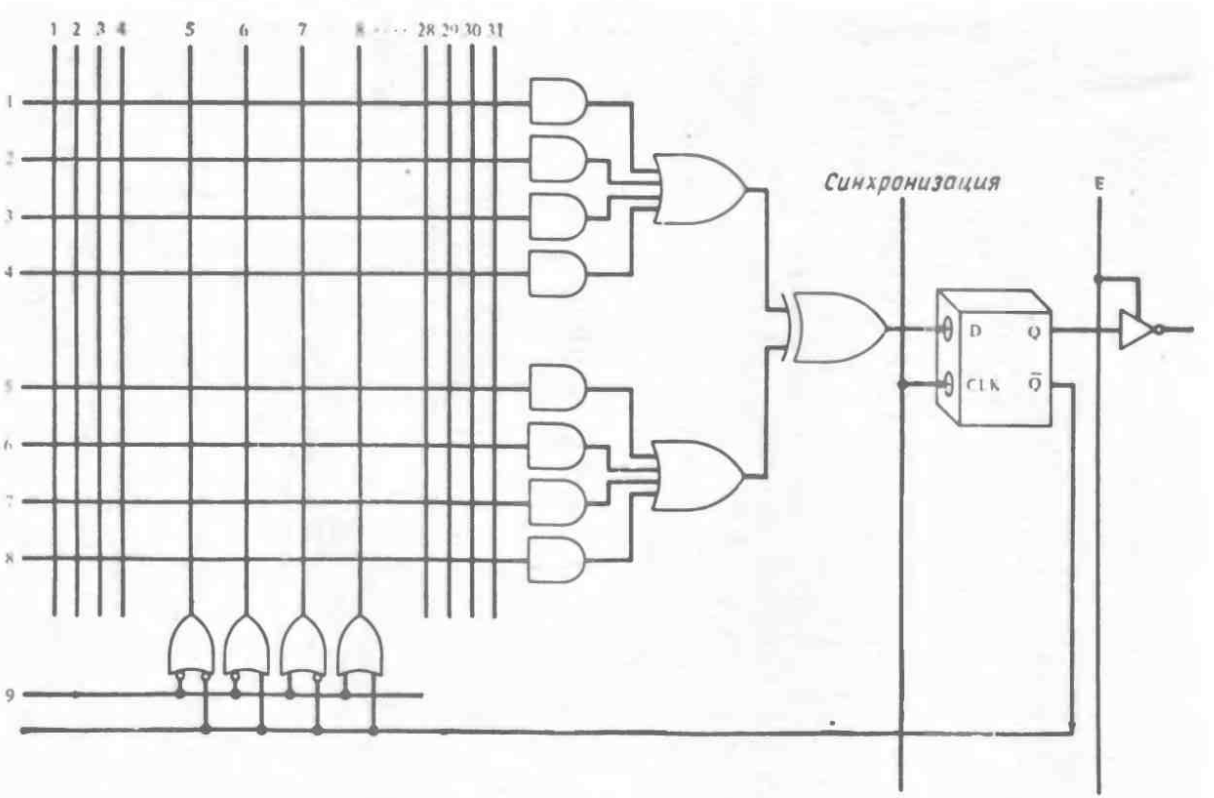


Рис. 4.39. Простая арифметическая матричная структура.

АРИФМЕТИЧЕСКИЕ МАТРИЧНЫЕ СТРУКТУРЫ

Существенное отличие булевых операций от арифметических состоит в том, что при выполнении последних формируется и распространяется сигнал переноса при сложении и осуществляется заем при вычитании. В матричных структурах, проектируемых для выполнения арифметических операций, должен предусматриваться механизм формирования и распространения сигнала переноса. Формирование сигнала переноса легко осуществляется с помощью схемы ИСКЛЮЧАЮЩЕЕ ИЛИ. Простая арифметическая ячейка приведена на рис. 4.39. Эта схема учитывает переносы, возникающие при выполнении предыдущих операций, выполняя операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над переменными, сформированными в текущей операции. Для обеспечения процесса суммирования с учетом переноса объединяется несколько подобных ячеек. В микросхеме PAL 16X4 фирмы Monolithic Memories содержится четыре арифметические матричные ячейки. Кроме того, для образования всех термов двух переменных I и Q может быть использована еще одна структура (рис. 4.40), которая соединяется с «выходом состояния» арифметической структуры.

СЕГМЕНТИРОВАННЫЕ МАТРИЧНЫЕ ЛОГИЧЕСКИЕ СТРУКТУРЫ

В ранних практических реализациях матричных схем — ПЗУ, ППЗУ и ПЛМ — были предусмотрены внутренние цепи обратной связи. Многие ПЛМ, описанные в предыдущих разделах, имеют такие внутренние цепи обратной связи. Их наличие обусловливает снижение количества выводов, которое потребовалось бы для построения комбинационных схем, имеющих более чем два уровня. Использование внутренних цепей обратной связи вместо внешних приводит к освобождению внешних выводов. Однако для образования одной цепи обратной связи, т. е. связи выхода схемы со входом, приходится затрачивать по крайней мере одну из линий термов матричной схемы, которая в типичных вариантах реализации имеет 32 такие линии. На рис. 4.41 изображены две структуры, поясняющие возможные способы организации обратной связи в матричных схемах.

Рассмотрение способа организации обратной связи с помощью внутрисхемных линий в том случае, когда линии слов, или термов, могут разрываться в требуемом месте при сегментировании схем, показывает, что к одной и той же линии слова может подаваться по цепи обратной связи несколько выходных сигналов. Впервые такая структура была реализована совместно фирмами General Electric и Monolithic Memories в 1975 г.

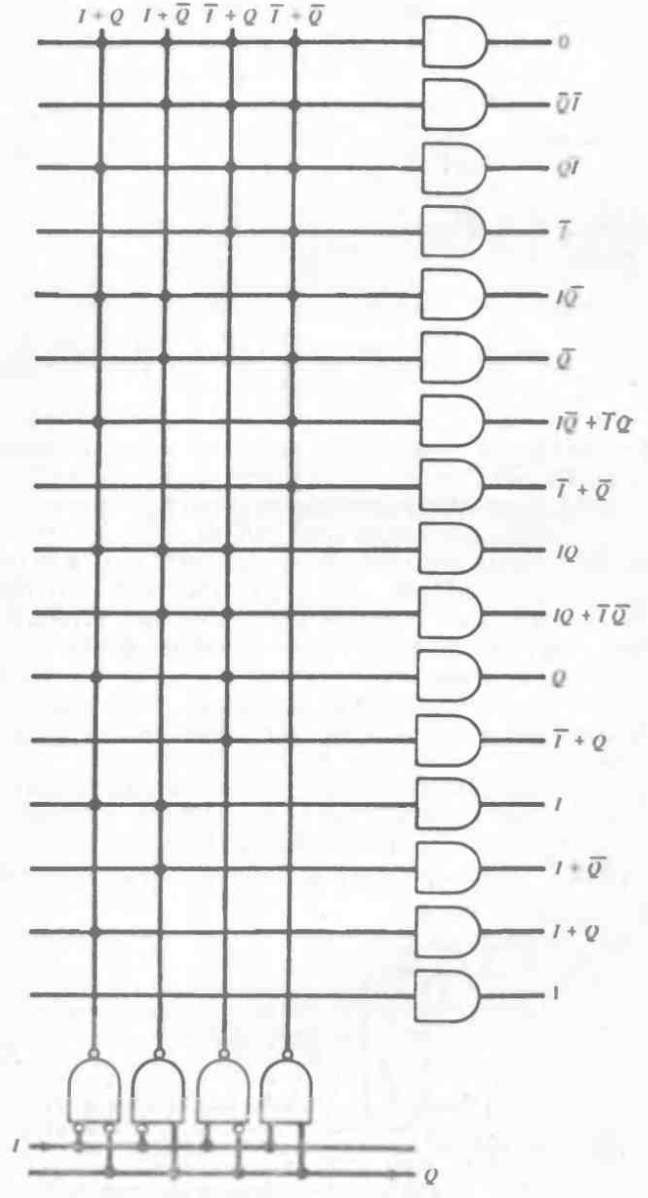


Рис. 4.40. Матричная структура для вычисления логических функций двух суммируемых переменных; используется совместно с арифметической ячейкой.

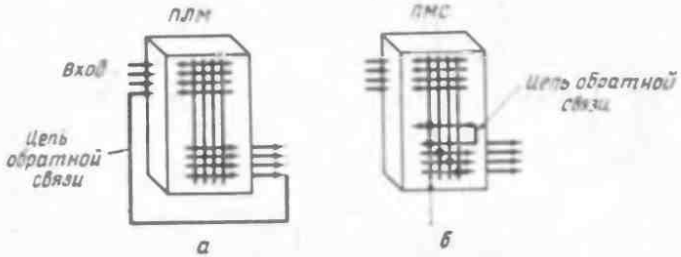


Рис. 4.41. Принципы организации обратной связи в программируемых логических матрицах (*а*) и в матричных логических схемах (*б*).

Структура базисной ячейки, являющейся сегментированной матричной логической схемой, представлена на рис. 4.42. Символические обозначения, использованные при изображении этой схемы, можно интерпретировать следующим образом: сегменты линий слов, или термов, играют роль схем И, входы которых обозначены точками; линии разрядов представляют собой либо входные проводники элементов И, либо элементы ИЛИ. Эта физическая структура по принципам действия аналогична логическим диодным матрицам, рассмотренным в предыдущих разделах. Соединения в ячейках структуры, представленной на рис. 4.44, эквивалентны соединениям, имеющимся в MS-триггере D-типа, изображенном на рис. 4.43.

Структура, представленная на рис. 4.44, была реализована, однако не был осуществлен вариант схемы с программируемым полем. Используя сегментированные логические матрицы, например подобные рассмотренным выше, легко конструировать мно-

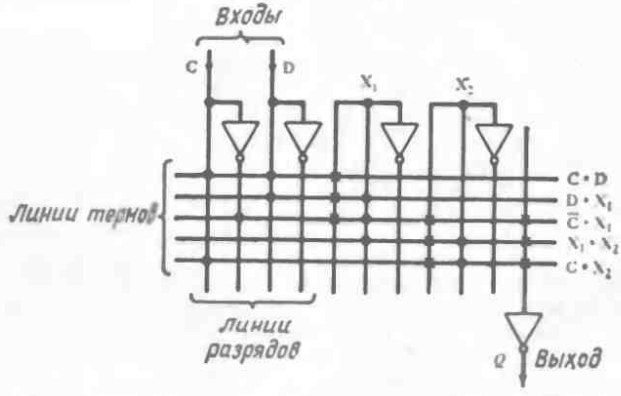


Рис. 4.42. Базисная сегментированная матричная логическая ячейка, представляющая при сделанных в ней соединениях MS-триггер D-типа.

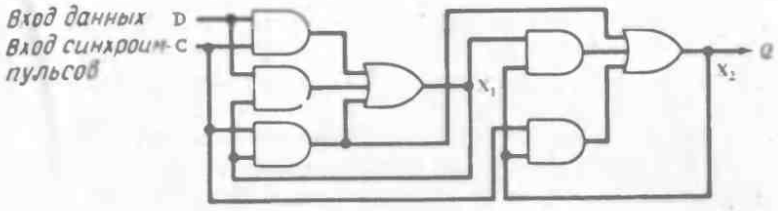


Рис. 4.43. Классическая схема MS-триггера D-типа.

гоуровневые логические схемы и строить логические сети, содержащие более 100 вентилей. Линии термов можно сегментировать между ячейками.

ЛОГИЧЕСКИЕ МАТРИЦЫ И БУЛЕВА АЛГЕБРА

Прямыми предшественниками электронных вентильных элементов и построенных на их основе цифровых электронных схем являются механические релейные схемы. Использование релейных схем в промышленности и в телефонных коммутационных системах было весьма широким, что стимулировало разработку методов инженерного проектирования, направленных на минимизацию числа элементов в системе и таким образом на уменьшение их стоимости. Замена реле электронными лампами во многих случаях позволила уменьшить время переключения, од-

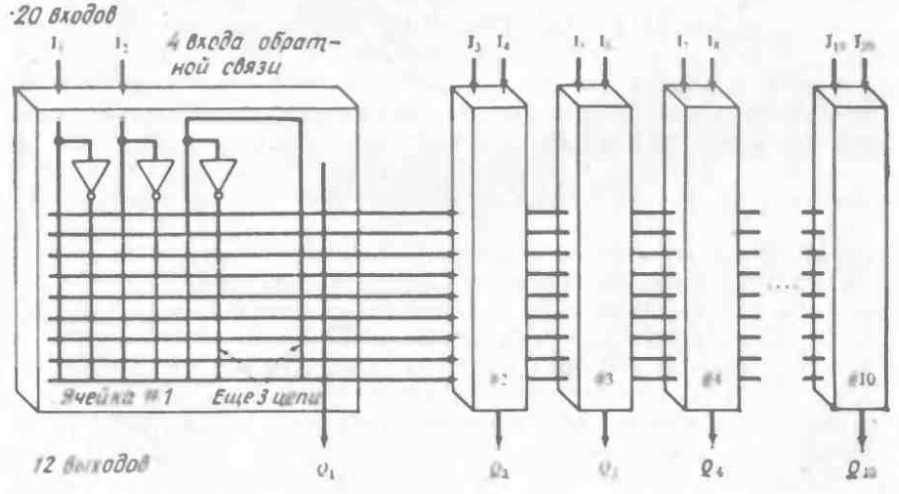


Рис. 4.44. Сегментированная логическая структура. (Воспроизводится из сборника докладов Международной конференции по интегральным схемам на твердом теле, февраль 1977.)

нако не привела к уменьшению стоимости переключающих устройств в расчете на один вентиль. Подобным же образом замена в переключательных схемах электронных ламп транзисторами вызвала изменение технологии производства, но не сказалась существенно на стоимости систем. Появление интегральных схем с малой степенью интеграции, т. е. схем, содержащих на кристалле 1—20 вентилей, столь же мало изменило экономические показатели проектируемых систем. В последнее время широкое применение интегральных схем со средней и большой степенью интеграции обусловило необходимость учета некоторых новых важных факторов экономического анализа проектов сложных цифровых систем. Первым важным фактором стало конструктивное ограничение количества внешних выводов в корпусах интегральных схем. Системы с увеличенным количеством логических элементов, но с меньшим количеством корпусов, могут иметь меньшую стоимость, чем системы с минимальным количеством элементов.

ЛОГИЧЕСКИЕ МАТРИЦЫ С ПРОГРАММИРУЕМЫМ ПОЛЕМ

Одной из ведущих фирм — пионеров в области разработки логических устройств с программируемыми матрицами — является фирма Signetics. Эта фирма выпустила ряд программируемых матриц вентилей, предназначенных для конструирования машин состояний, и логические матричные схемы с программируемым полем. Матрицы с программируемым полем отличаются от матриц с масочным программированием методом программирования: в первом случае оно выполняется посредством изменения «поля», а во втором — с помощью специальной маски. Фирма Signetics производит целое семейство интегральных схем с плавкими связями. Схема устройства 82S100, содержащего логическую матрицу с программируемым полем, изображена на рис. 4.45.

Матрица И, являющаяся матрицей первого уровня, аналогична матрицам, рассмотренным в предыдущих разделах. Введение в структуру на каждой выходной линии элемента ИСКЛЮЧАЮЩЕЕ ИЛИ обеспечивает еще одну возможность программирования схемы. Один из входов каждого элемента ИСКЛЮЧАЮЩЕЕ ИЛИ соединен плавкой вставкой с землей. Если эту вставку расплавить, то на соответствующем входе будет установлен высокий уровень напряжения. Таким образом можно программировать каждый выход схемы для того, чтобы в нормальном состоянии на выходе устанавливалось определенное логическое значение — «истинно» или «ложно». Такая возможность позволяет использовать теорему де Моргана и делает

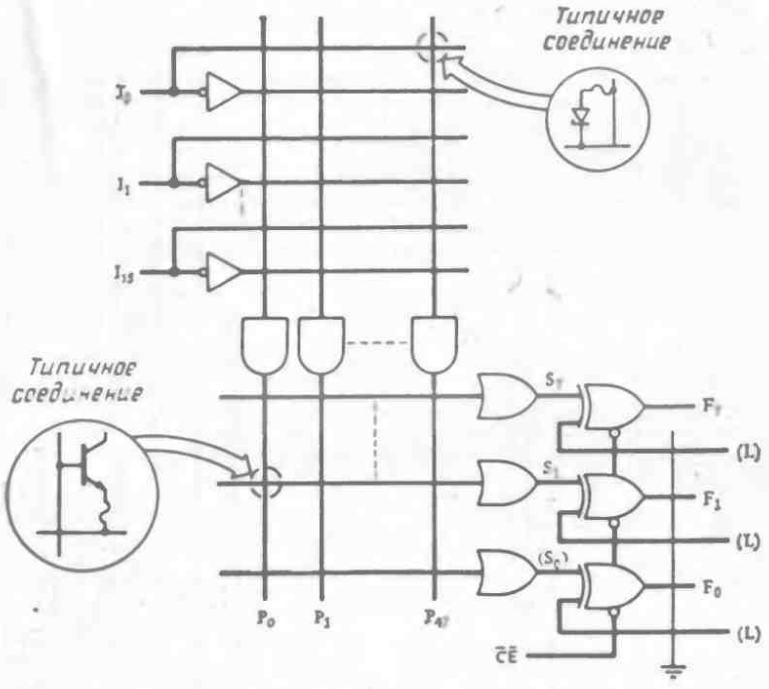


Рис. 4.45. Устройство 82S100 (16×48×8) фирмы Signetics, содержащее логическую матрицу с программируемым полем. (*С разрешения* Signetics Corp.)

Плавкие никелево-хромовые вставки первоначально существуют во всех точках пересечения линий матрицы. Фиксированное соединение обозначено черным кружком.

рассматриваемое устройство универсальным логическим элементом. Как показано на рисунке, программируемая матрица вентилей имеет 16 входов и 9 выходов.

Если к выходам матрицы ИЛИ добавить линии обратной связи, то получим логическую схему, содержащую матрицу с программируемым полем (рис. 4.46). Отметим, что в этой схеме используются такие же программируемые элементы ИСКЛЮЧАЮЩЕЕ ИЛИ, как и в описанной выше схеме.

Применение внутренних линий обратной связи имеет большое значение для устройств с ограниченным количеством внешних выводов. Схема 82S152/153 фирмы Signetics, входящая в серию логических матриц с программируемым полем, имеет такую обратную связь на программируемых выходах. Сигналы обратной связи (фактические и инверсные значения) могут использоваться на каждой из 32 вертикальных линий матрицы И, и, кроме того, они могут задаваться в любой комбинации для отпирания или запирания каждого из 10 выходов. В устройстве

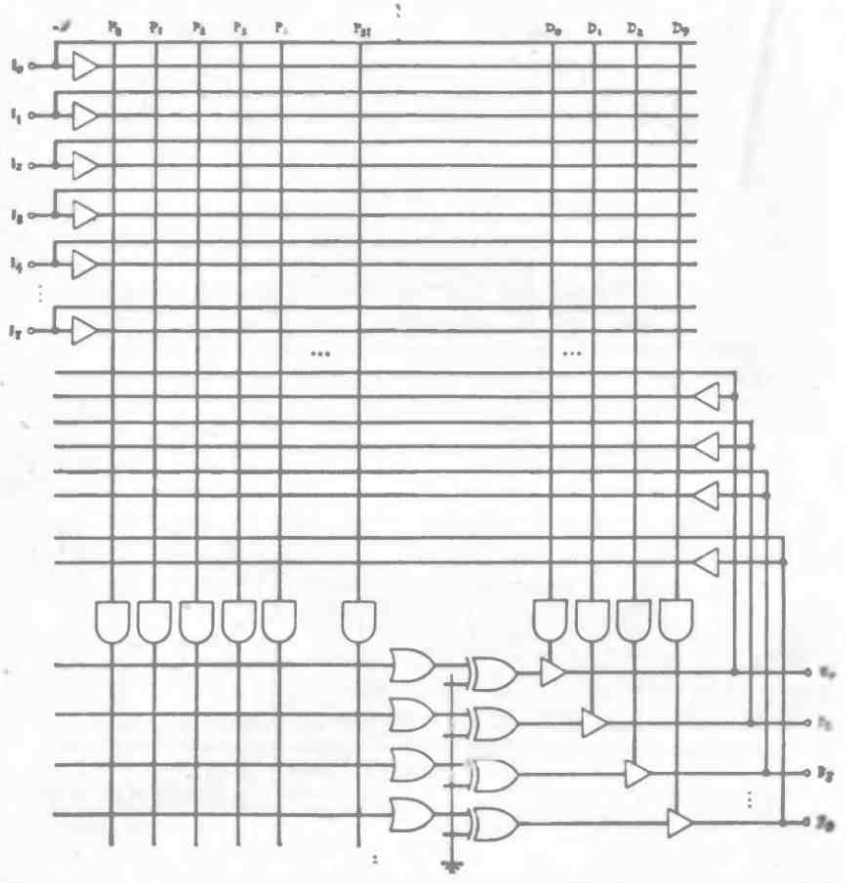


Рис. 4.46. Устройство 82S152/153 фирмы Signetics. (*С разрешения* Signetics Corp.)

82S154 для достижения максимальной гибкости используется как внутренняя обратная связь, так и внутренняя память. Эта логическая матричная схема, обладающая большими возможностями, представлена на рис. 4.47.

ОПИСАНИЕ КЛАССОВ МАШИН СРЕДСТВАМИ ПРЕДСТАВЛЕНИЯ МАТРИЧНЫХ ЛОГИЧЕСКИХ СХЕМ

Формализм, используемый нами для описания матричных логических схем, достаточно легко применить для представления машины класса 1 (рис. 4.48). Читатель, разобрав пример, дан-

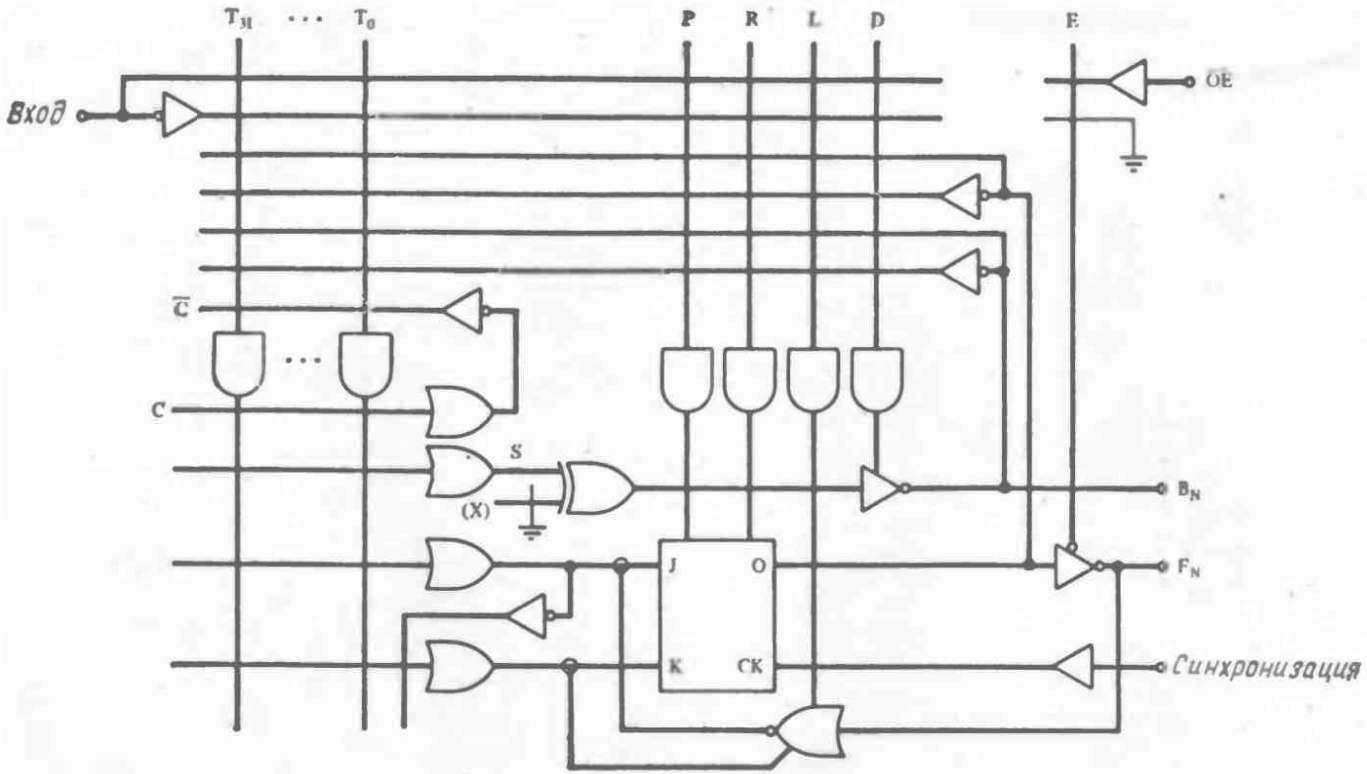


Рис. 4.47. Устройство 82S154 фирмы Signetics (*С разрешения* Signetics Corp.)

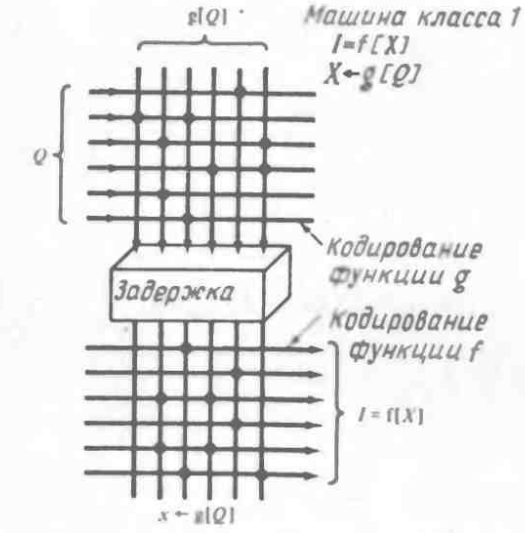


Рис. 4.48. Представление машины класса 1 матричной логической схемой.

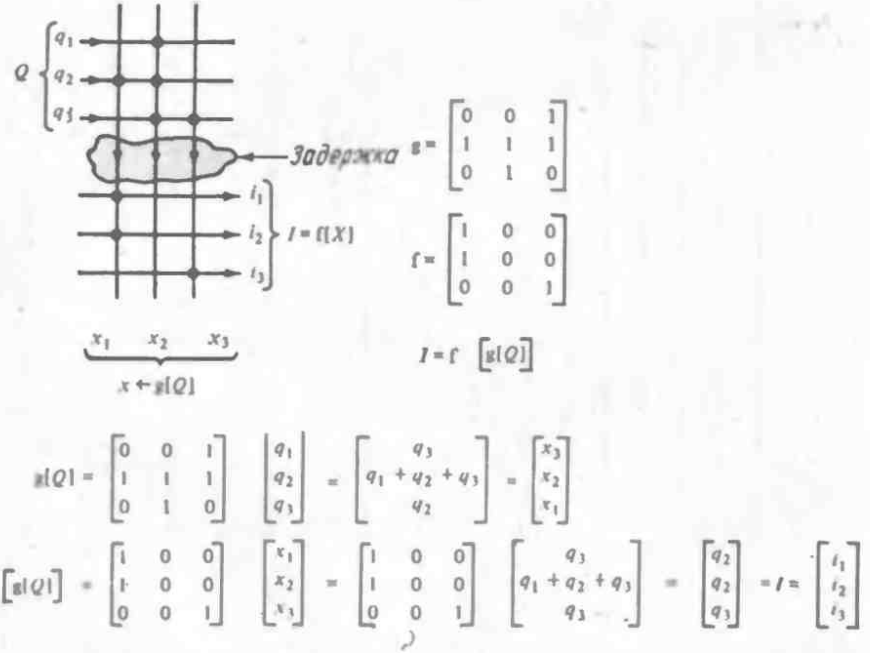


Рис. 4.49. Пример представления машины класса 1 матричной логической схемой и преобразованиями булевых матриц.

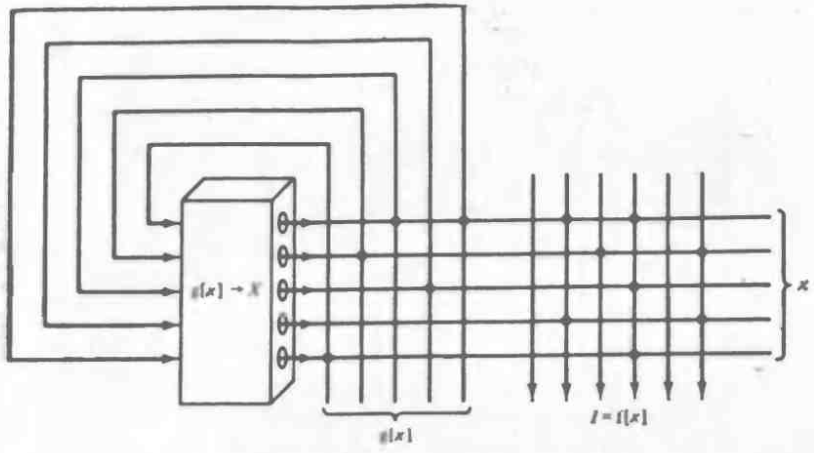


Рис. 4.50. Представление машины класса 2 матричной логической схемой.

ный на рис. 4.49, должен убедиться в том, что преобразования, выполненные с булевыми матрицами, дают правильный результат. Отметим, что в матричном представлении столбцам матрицы соответствуют входные линии, а строкам — выходные линии.

Машина класса 2 (рис. 4.50) удовлетворяет уравнениям класса 2. Читатель должен самостоятельно убедиться в том, что матричная логическая схема, изображенная на рис. 4.51, представляет собой двоичный счетчик на 5.

Представление машин классов 3 и 4 с помощью матричных логических схем дано на рис. 4.52 и рис. 4.53 соответственно.

Простая схема счетчика на 4 изображена на рис. 4.54, *а*. Изменение состояния счетчика происходит при поступлении каждого синхроимпульса; как состояния схемы, так и значения сигналов на ее выходе представляются двоичным кодом. Для того чтобы на время переходного процесса изолировать выход схемы от ее входов, использован MS-триггер.

КАНОНИЧЕСКИЕ ФОРМЫ

Естественным этапом решения сложных задач является попытка дать описание задачи в стандартной, или канонической, форме. При решении задач булевой алгебры стандартная форма представления логических выражений содержит мини-термы, или произведения, базисных переменных, инверсию этих переменных. Целесообразность такого подхода обнаруживается при рассмотрении некоторой задачи с тремя множествами x, y и z. Диаграмма Венна, изображенная на рис. 4.55, показывает, что

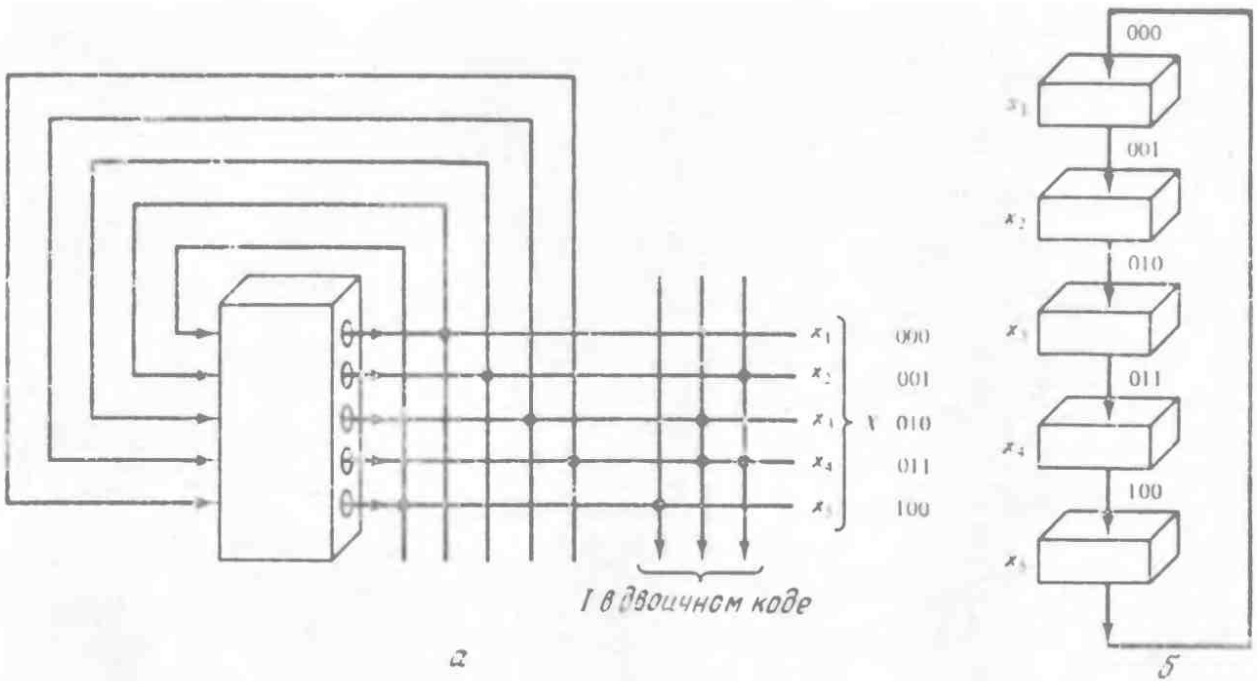


Рис. 4.51. Двоичный счетчик на 5: *а* — представление матричной логической схемой; *б* — представление алгоритмической машиной состояний.

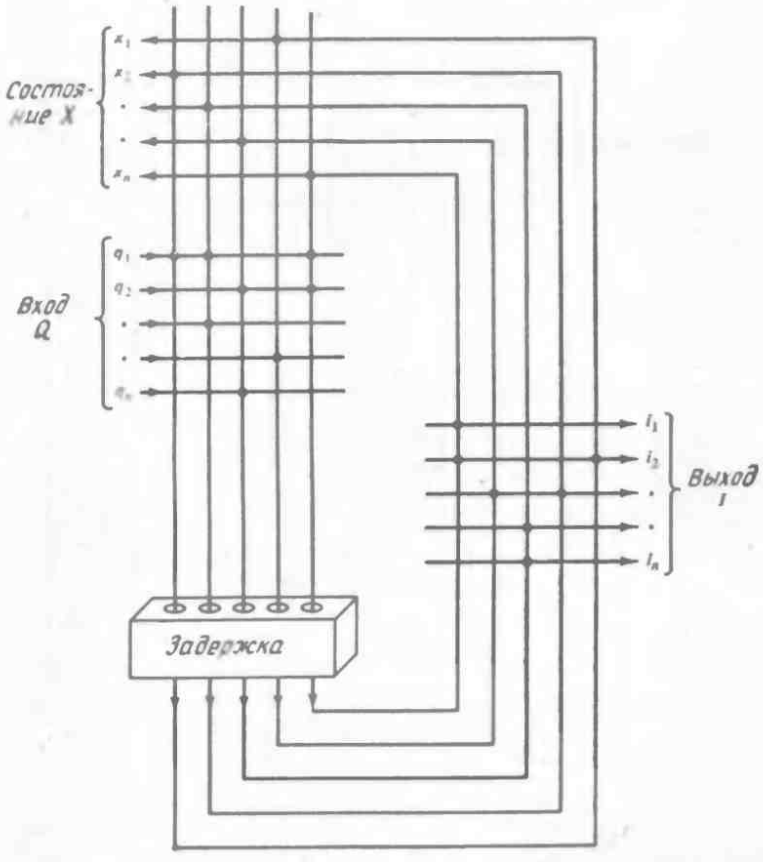


Рис. 4.52. Представление машины класса 3 матричной логической схемой.

любое множество из универсального множества данной задачи может быть выражено в виде одного из следующих произведений:

$$x\bar{y}\bar{z},\quad \bar{x}y\bar{z},\quad \bar{x}\bar{y}z,\quad \bar{x}\bar{y}\bar{z},\quad x\bar{y}z,\quad \bar{x}yz,\quad \bar{x}\bar{y}z,\quad xyz.$$

Можно убедиться в том, что любое выражение с переменными x, y и z можно представить в виде суммы произведений, или мини-термов. Каждое такое выражение на основании принципа двойственности может быть заменено эквивалентным выражением, которое будет являться произведением сумм. Целесообразность канонического представления выражений при проектировании матричных схем очевидна, поскольку двухуровневая матричная логическая структура И — ИЛИ всегда дает резуль-

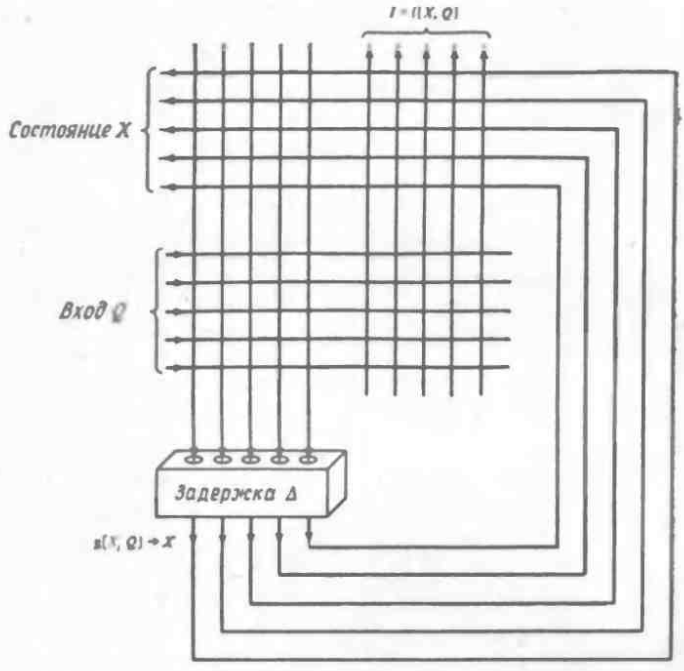


Рис. 4.53. Представление машины класса 4 матричной логической схемой.

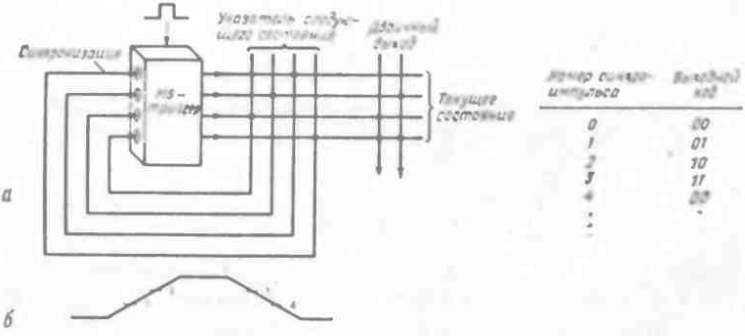


Рис. 4.54. Реализация счетчика на 4 с использованием MS-триггера (*а*); форма синхронимпульса MS-триггера (*б*).

1 — отключение ведомого триггера от ведущего; *2* — разрешение ввода данных в ведущий триггер; *3* — блокировка входов данных; *4* — передача данных из ведущего триггера в ведомый.

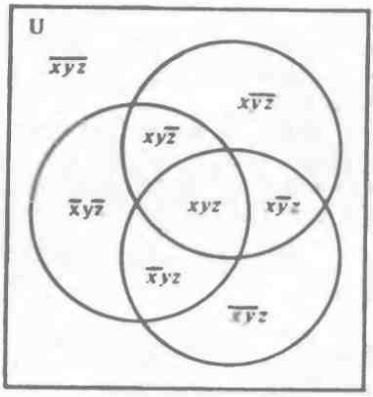


Рис. 4.55. Диаграмма Венна.

тат, который может быть выражен в виде суммы произведений. Следовательно, матричные схемы хорошо согласуются с применением методов минимизации булевых функций, представленных в виде суммы произведений.

КАРТЫ КАРНО

Любая логическая функция может быть задана таблицей истинности. Описание функции в виде полной таблицы истинности, как правило, не является ее минимальным описанием, так как информация в таблице обычно оказывается избыточной. Таблица истинности может быть преобразована в так называемую карту Карно, которая используется как вспомогательное средство для получения минимизированного представления (и, следовательно, реализации) логических функций. В данном случае элементы таблицы рассматриваются как двоичные коды, а коды, отличающиеся значением только одной переменной, определяются положением, которое представляется n-кой или вектором в n-мерном пространстве. Эти n-ки могут отображаться в двумерное пространство посредством процесса развертывания (рис. 4.56). Карта Карно может быть получена, исходя из диаграммы Венна. Так, для случая двух переменных диаграмма Венна легко отображается с сохранением существующей топологии в карту Карно. Такое преобразование представлено на рис. 4.57. Основным достоинством применения карт Карно — в отличие от использования метода упрощения булевых уравнений путем алгебраических преобразований — являются компактность и наглядность представления функций. Используя табли-

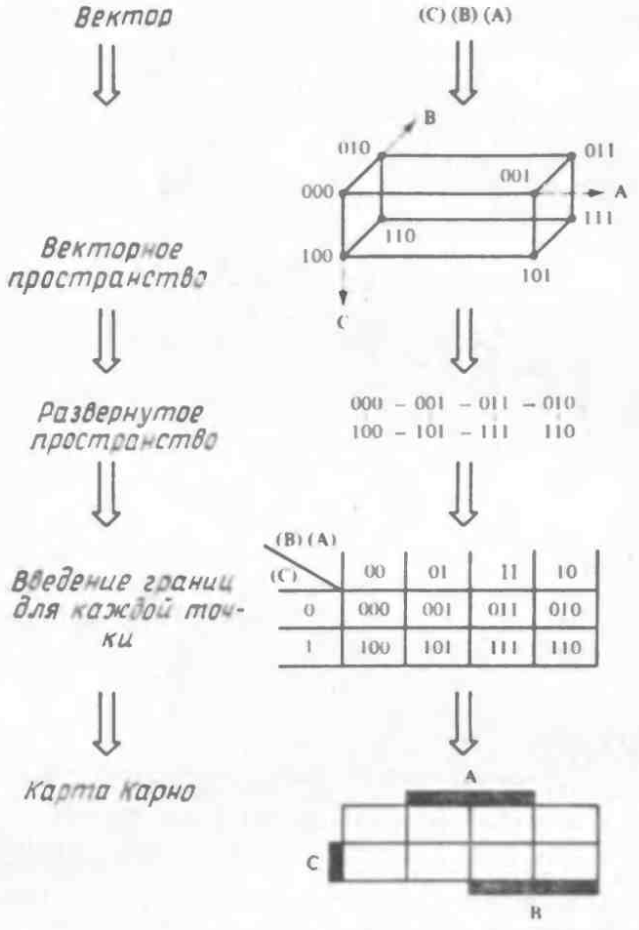


Рис. 4.56. Процесс получения карты Карно для трехэлементного вектора.

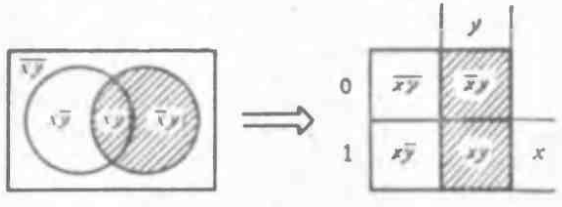


Рис. 4.57. Преобразование диаграммы Венна в карту Карно.
Карта строится так, чтобы мини-термы, для которых выполнимо поглощение, оказались смежными, например $\overline{x}y+xy=y$.

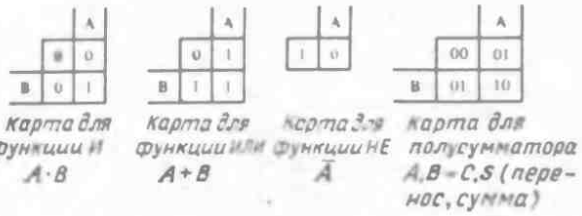


Рис. 4.58. Карты Карно для основных логических функций.

цы истинности, легко составить карты Карно для элементов И, ИЛИ, НЕ и полусумматора (рис. 4.58). Карты Карно могут применяться в методах минимизации, служащих для упрощения булевых уравнений, описывающих системы, а следовательно, и для минимизации числа элементов в реализуемых системах.

ПРОЕКТ ДЕСЯТИЧНОГО СЧЕТЧИКА НА ТРИГГЕРАХ D-ТИПА

Десятичный счетчик представляет собой машину состояний класса 2, в которой осуществляются прямые переходы из состояния в состояние. Поскольку существуют устройства с программируемыми матрицами, имеющие регистровые выходы D-типа, будем рассматривать реализацию десятичного счетчика с использованием матричных логических элементов.

Время изменения состояния логической ячейки с программируемой матрицей определяется внешним синхронизирующим сигналом, а следующее состояние триггера D-типа определяется сигналом, подаваемым на вход триггера. В табл. 4.1 представлены все состояния, в которых может находиться десятичный счетчик.

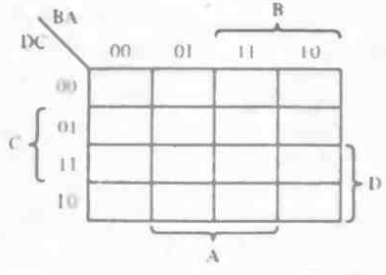


Рис. 4.59. Формат карты Карно для четырех переменных.

Таблица 4.1. Таблица состояний десятичного счетчика

Выход	Текущее состояние X				Следующее состояние g(X)			
	x_4	x_3	x_2	x_1	x_4	x_3	x_2	x_1
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	0	0	0	0

Чтобы упростить изложение материала, определим переменные, характеризующие состояние счетчика, следующим образом:

$$x_4 = D, \quad x_3 = C, \quad x_2 = B, \quad x_1 = A.$$

Для составления карты Карно будем пользоваться четырьмя переменными: A, B, C и D. Поскольку карта Карно представляет собой двумерную таблицу, сгруппируем переменные попарно так, как показано на рис. 4.59. Каждому триггеру соответствует одна карта Карно. Так как в счетчике используются четыре триггера, в данном случае потребуются четыре карты Карно. Элементы карт Карно представляют собой значения входных переменных, необходимые для конкретного триггера. Они могут быть выведены, исходя из табличного описания. Так как мы рассматриваем десятичный счетчик, то состояния счетчика 1010, 1011, 1100, 1101, 1110 и 1111 нас не интересуют. Эти состояния будем отмечать записью символа X в соответствующие клетки карт Карно. Таким образом, все четыре карты Карно будут содержать записи, представленные на рис. 4.60.

Анализируя таблицу состояний счетчика, замечаем, что первые семь переходов, соответствующих состояниям 0, 1,...,6, не приводят к появлению 1 в триггере старшего разряда, которому, согласно принятым обозначениям, соответствует переменная D. Следовательно, при указанных состояниях входной сигнал для этого триггера будет равен 0 (рис. 4.61, *б*). Для того чтобы счетчик оказался в состоянии 8, т. е. 1000, во время седьмого состояния входной сигнал триггера, соответствующего старшему разряду хранимого в счетчике числа, должен быть равен 1. Анализируя состояние счетчика по таблице состояний, замечаем, что в момент подачи сигнала 1 на вход триггера старшего разряда состояние счетчика должно быть равно 0111.

DC \ BA	00	01	11	10
00				
01				
11	X	X	X	X
10			X	X

Рис. 4.60. Карта Карно десятичного счетчика, отображающая состояния, которые не следует принимать во внимание.

Следовательно, мы должны записать 1 в клетку карты Карно, соответствующую состоянию DCBA=0111 (рис. 4.61, *в*). Аналогично, для установления счетчика в состояние 1001 на вход триггера старшего разряда должен быть подан сигнал 1. На этот раз 1 должна быть записана в клетку карты Карно, соответствующую состоянию счетчика 1000 (рис. 4.61, *г*). Наконец, если счетчик находится в состоянии 1001, то на вход триггера старшего разряда должен быть подан сигнал 0, так как состояние счетчика изменяется циклически и значение старшего

а

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0		0
11				
10				

б

DC \ BA	00	01	11	10
00				
01			1	
11				
10				

в

DC \ BA	00	01	11	10
00				
01				
11				
10	1			

г

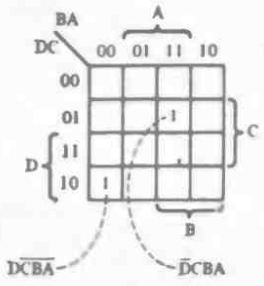
DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	1	0	X	X

д

Рис. 4.61. Вывод карты Карно для старшего разряда десятичного счетчика: *а* — не имеющие значения комбинации 1010→1111; *б* — значения для состояний 0001→0111; *в* — значение для состояния 1000; *г* — значение для состояния 1001; *д* — полная карта Карно.

Следующее состояние (X) D	C	B	A	Значение старшего разряда X
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1 ... $\bar{D}CBA$
1	0	0	0	1 ... $D\bar{C}\bar{B}\bar{A}$
1	0	0	1	0
0	0	0	0	0

а



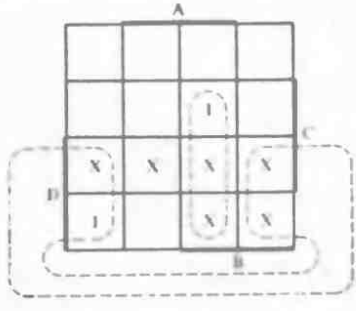
б

Рис. 4.62. Вывод логического выражения $\bar{D}CBA + D\bar{C}\bar{B}\bar{A}$ для старшего разряда десятичного счетчика: *а* — по таблице истинности; *б* — по карте Карно.

разряда становится равным 0. Полная карта Карно для триггера старшего разряда счетчика представлена на рис. 4.61, *д*.

Получив карту Карно для триггера старшего разряда счетчика, предпримем попытку минимизировать булево выражение для этого триггера. Прежде всего запишем рассматриваемое выражение в виде дизъюнкции мини-термов, т. е. в дизъюнктивной нормальной форме. В выражении должны быть представлены все термы, которые обеспечивают значение «истинно» рассматриваемой функции. Отметим, что термы могут быть получены, либо исходя из табличного описания функции, либо путем записи термов, соответствующих единичным элементам карты Карно. Оба способа получения мини-термов представлены на рис. 4.62.

Как и следовало ожидать, выражения для мини-термов оказались одинаковыми. Карта Карно используется для группи-



$\bar{D}CBA + DCBA = (\bar{D} + D)CBA = CBA$

$D\bar{C}\bar{B}\bar{A} + D\bar{C}B\bar{A} + DCB\bar{A} + DC\bar{B}\bar{A} = D\bar{A}$

Старший разряд $= D\bar{A} + CBA$

Рис. 4.63. Использование карты Карно для упрощения логического выражения, представляющего собой сумму мини-термов.

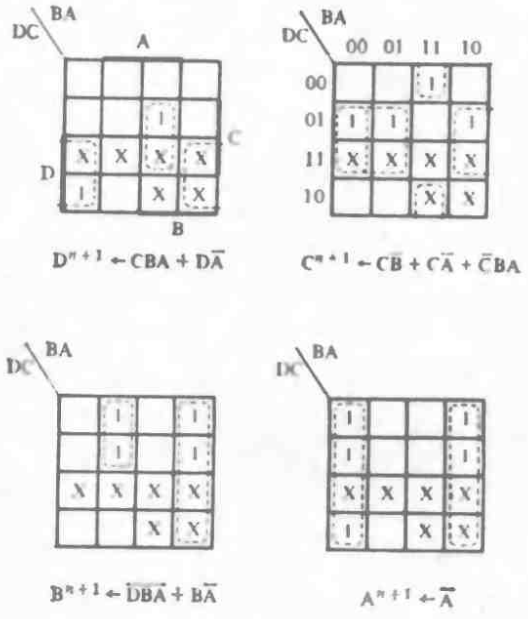


Рис. 4.64. Использование карт Карно для упрощения логических выражений, соответствующих разрядам десятичного счетчика.

рования смежных термов (т. е. термов, отличающихся значением только одной переменной), производимого с целью упрощения выражений. В процессе группирования любой неопределенный терм можно считать равным 1. На рис. 4.63 представлен один из возможных вариантов группирования термов. Здесь терм $\overline{D}CBA$ объединяется с неопределенным термом DCBA с целью получения терма CBA, независимого от D. Отметим, что последний терм можно получить путем элементарных алгебраических преобразований. Менее очевидна цель группирования терма $DC\overline{BA}$ с тремя смежными неопределенными термами. В данном случае результатом группирования и упрощения будет произведение $D\overline{A}$. Чтобы это проверить, читатель должен выполнить необходимые алгебраические преобразования.

Карты Карно для выходов каждого из четырех триггеров D-типа изображены на рис. 4.64. На этом же рисунке представлены упрощенные выражения для соответствующих выходов. Для получения этих выражений использовался описанный выше метод.

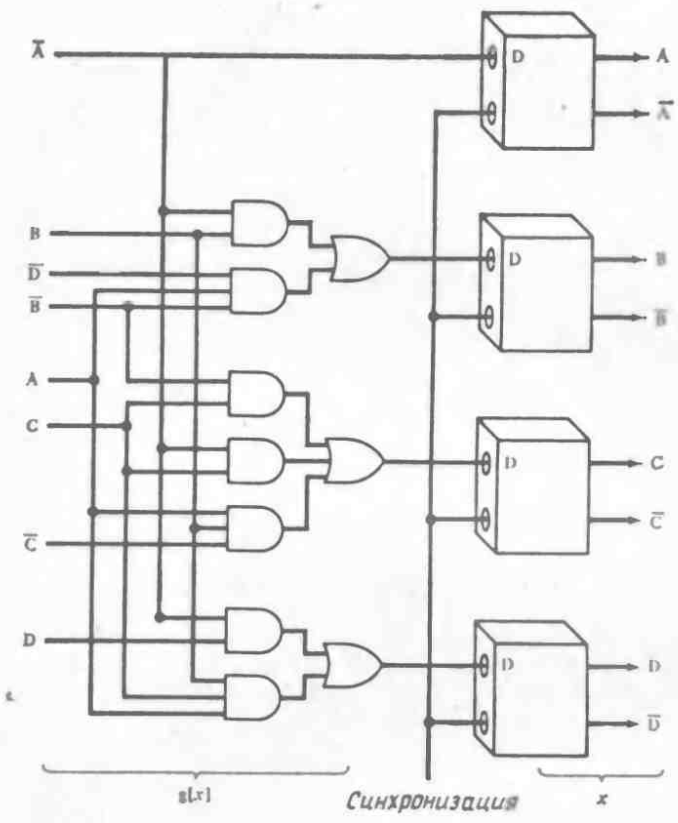


Рис. 4.65. Схема синхронного десятичного счетчика на триггерах задержки, выполненная с использованием традиционных обозначений.

Для наглядного сравнения приведем неупрощенные выражения из мини-термов и соответствующие им преобразованные выражения:

$$D^{n+1} \leftarrow \overline{D}CBA + D\overline{CBA} = CBA + D\overline{A}$$

$$C^{n+1} \leftarrow \overline{D}CB\overline{A} + \overline{D}C\overline{B}A + \overline{D}\overline{C}BA + \overline{D}CB\overline{A} = C\overline{B} + C\overline{A} + \overline{C}BA$$

$$B^{n+1} \leftarrow \overline{D}\overline{C}BA + \overline{D}\overline{C}B\overline{A} + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} = \overline{D}\overline{B}A + B\overline{A}$$

$$A^{n+1} \leftarrow \overline{D}\overline{C}\overline{B}A + \overline{D}\overline{C}B\overline{A} + \overline{D}C\overline{B}\overline{A} + \overline{D}CB\overline{A} + D\overline{CBA} = \overline{A}$$

Логическое описание системы может быть использовано для реализации различных по форме вариантов схем. Схема реализации десятичного счетчика на основе интегральных схем с

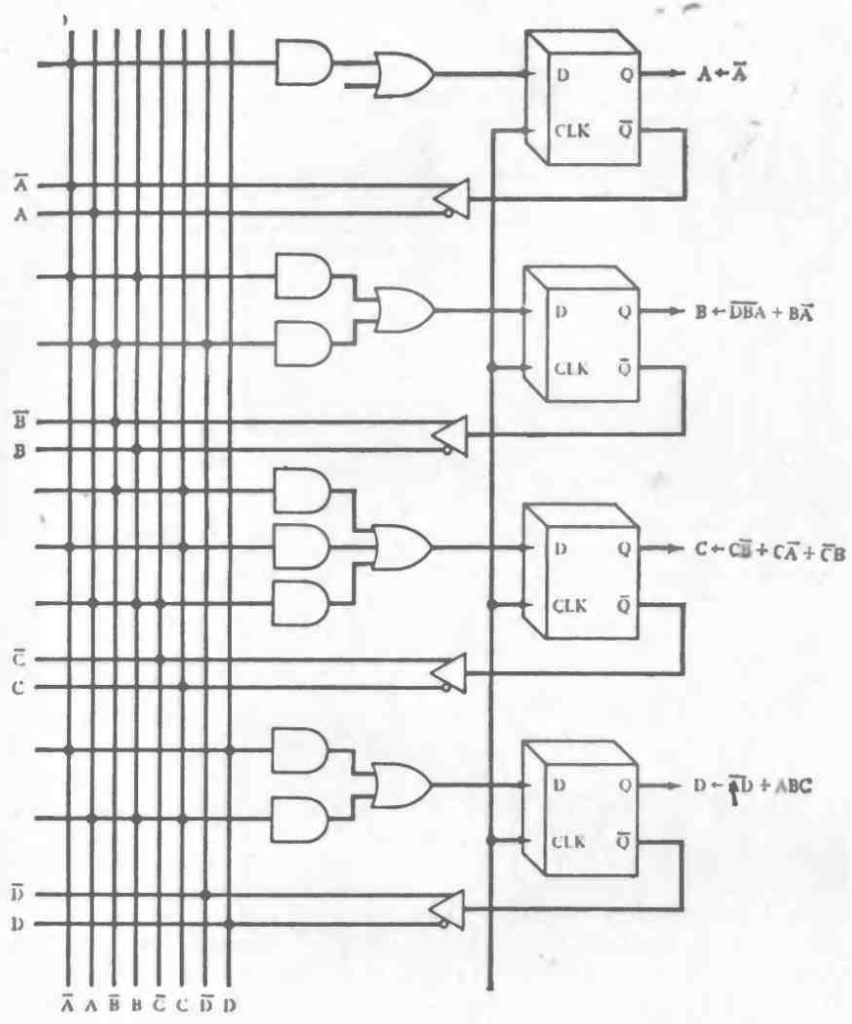


Рис. 4.66. Схема десятичного счетчика, построенного с использованием матричных логических схем.

малой степенью интеграции и с использованием набора упрощенных логических выражений приведена на рис. 4.65.

На рис. 4.66 изображена схема рассматриваемого счетчика, реализованного в виде логической схемы с программируемой матрицей. Эта схема получена путем прямого преобразования обычной (классической) логической схемы в матричную. Неиспользованные входы и элементы И для упрощения на данной схеме не изображены.

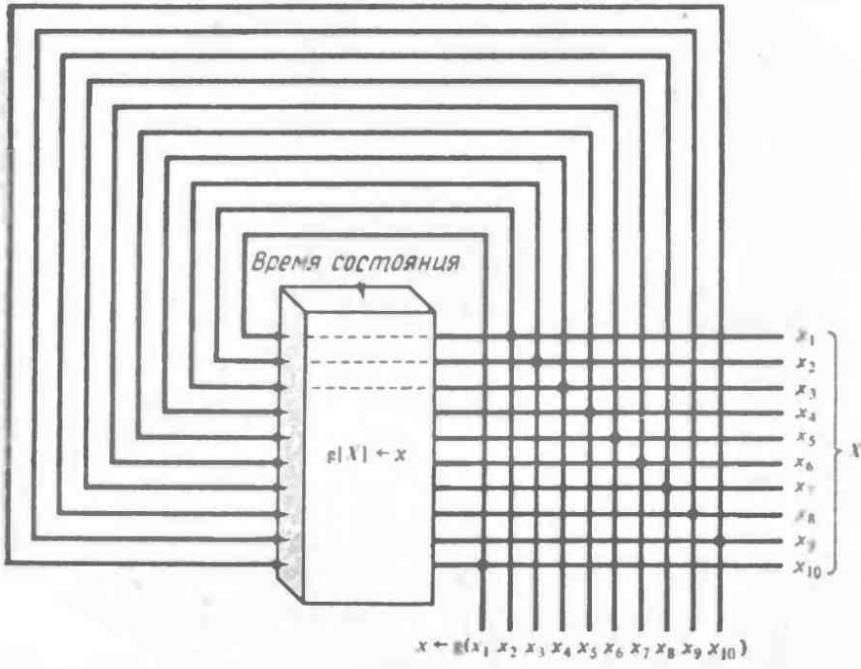


Рис. 4.67. Представление синхронного десятичного счетчика в виде машины состояний класса 2.

Для сопоставления мы также приводим схему десятичного счетчика в виде машины состояний класса 2. В схеме, представленной на рис. 4.67, используется 10 триггеров задержки, требуемых для позиционного представления вектора состояния. Различные схемные варианты реализации десятичного счетчика с использованием триггеров D-типа показаны на рис. 4.68 и 4.69. Существование триггеров различного типа (D-, RS-, T- и JK-триггеров) наводит на мысль, что в некотором смысле оптимальный проект счетчика может быть реализован с использованием триггеров одного или нескольких типов. Для сравнения на рис. 4.70 даны две схемы десятичного счетчика, базирующиеся на JK- и T-триггерах. Рассмотрим подробнее эти схемы.

На рис. 4.68 приведена схема реализации счетчика на основе приведенного набора упрощенных логических выражений с использованием программируемой логической матрицы. На схеме четко выделяется программируемая структура типа ИЛИ, а сами уравнения могут быть легко определены и записаны, исходя из схемы матрицы. Триггеры задержки могут быть либо внутрисхемными, либо внешними. На рис. 4.69 приведена схема

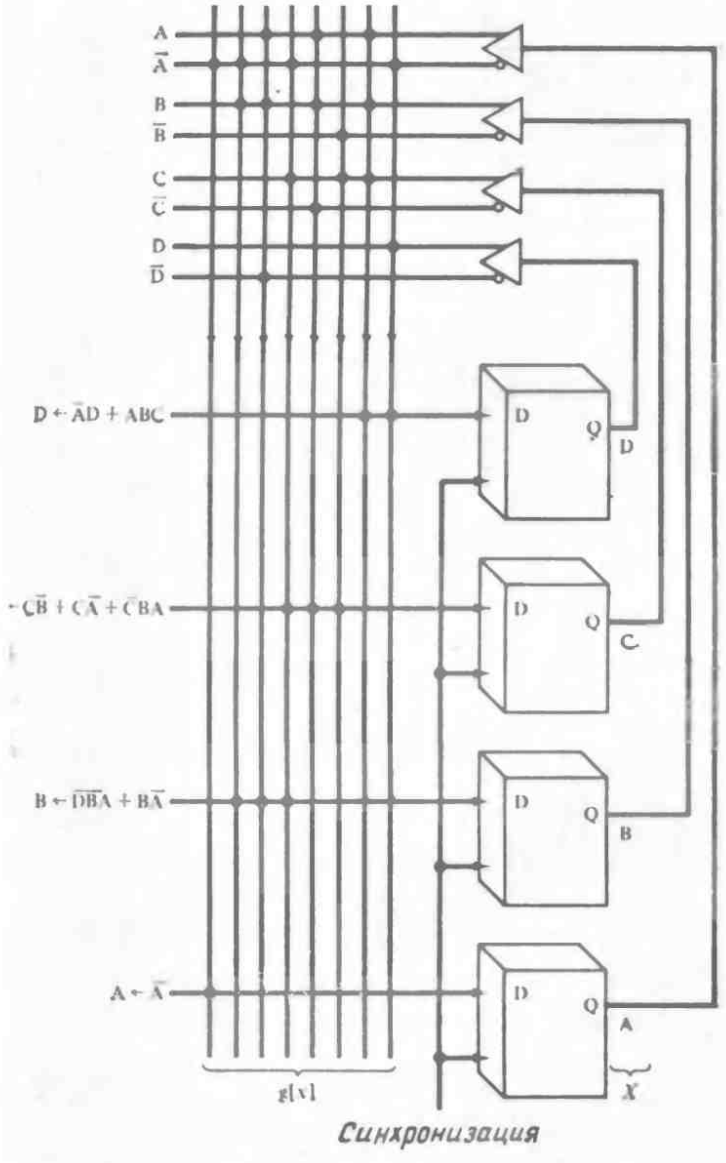


Рис. 4.68. Реализация десятичного счетчика с использованием программируемой логической матрицы и триггеров D-типа. (Схема реализована по упрощенным логическим выражениям.)

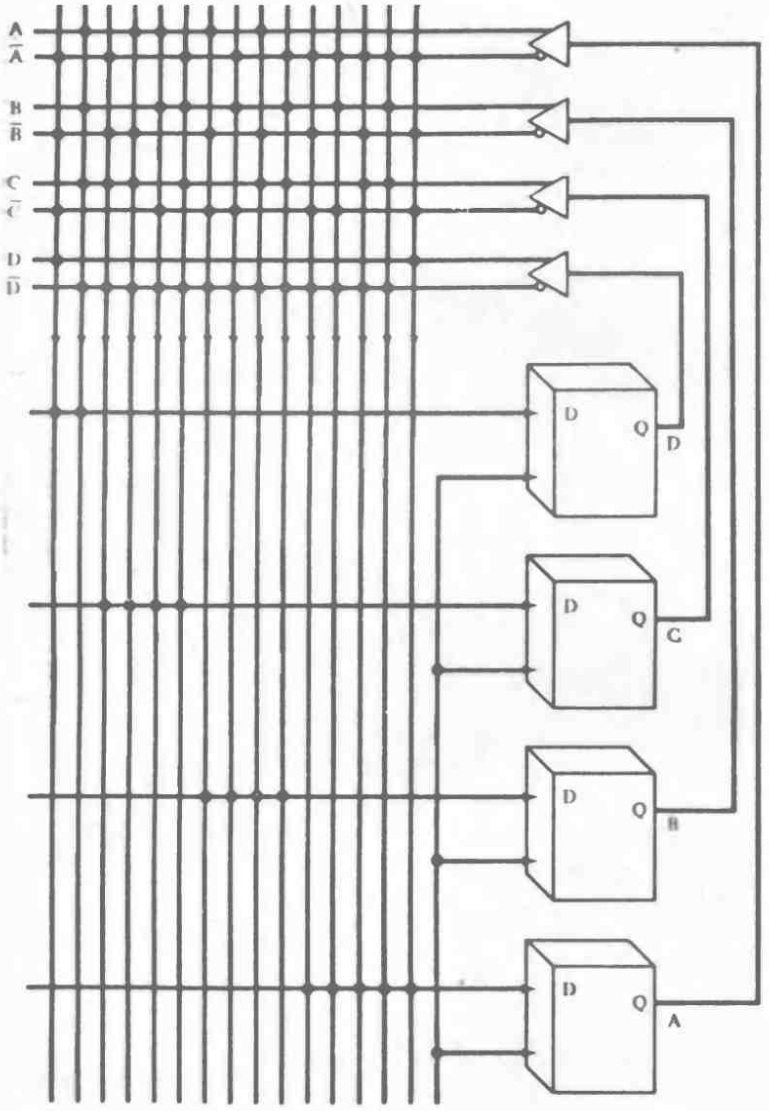


Рис. 4.69. Реализация десятичного счетчика с использованием программируемой логической матрицы и триггеров D-типа. (Схема реализована по исходным неупрощенным логическим выражениям.)

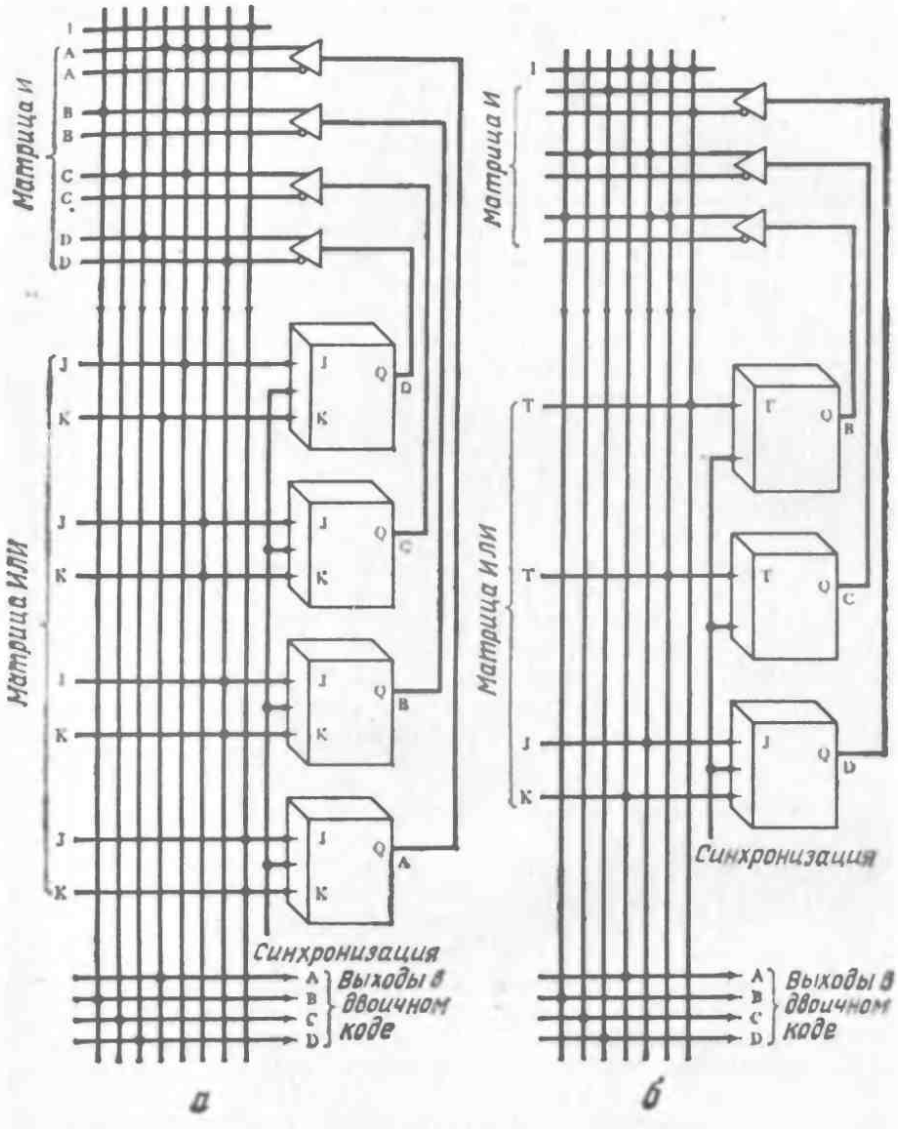


Рис. 4.70. Два варианта реализации десятичного счетчика (по упрощенным логическим выражениям): *а* — с использованием четырех JK-триггеров; *б* — с использованием одного JK-триггера и двух T-триггеров.

счетчика, выполненного с использованием программируемой логической матрицы по исходным (неупрощенным) выражениям.

В обоих вариантах построения счетчика (рис. 4.70) используется двоичное взвешенное кодирование выходных сигналов; во втором варианте применяются триггеры разного типа. [Методика проектирования этих и подобных схем (на программируемых логических матрицах) с использованием карт Карно изложена в книге Карра и Майза[1].] Схема на JK-триггерах проще, чем схема на триггерах D-типа, однако следует помнить, что для построения JK-триггеров используется больше вентилей, чем для триггеров D-типа. Следовательно, общее количество вентилей в схеме может оказаться и не минимальным. Таким образом, решение вопроса оптимальности построения счетчика зависит от того, применяются ли для его реализации внутренние триггеры D-типа (например, триггеры, входящие в устройство PAL16R4) либо внешние JK-триггеры в сочетании с программируемой логической матрицей.

Вообще формального решения задачи оптимального построения таких устройств не существует, необходим полный перебор всех возможных вариантов. А такой перебор обычно невозможен, поскольку число имеющихся в распоряжении проектировщика «строительных блоков» — схем огромно и каждый из них имеет сложную внутреннюю структуру. Принятие решения о целесообразности оптимизации конкретных проектов систем зависит от многих факторов, а сам процесс оптимизации все еще является скорее искусством, чем наукой.

КОГДА СЛЕДУЕТ ИСПОЛЬЗОВАТЬ ДАННУЮ ТЕХНОЛОГИЮ?

В течение 80-х годов выбор технологии для отдельных проектов будет сильно зависеть от объема выпускаемой продукции. Так, однокристальные ЭВМ с ППЗУ, допускающим стирание информации, выпускаются малыми сериями, а однокристальные ЭВМ, снабженные ППЗУ с масочным программированием, — большими. Высокоуровневые современные 16- и 32-разрядные процессоры не целесообразно применять в системах, объем производства которых не достигает нескольких сотен. Следующая ниже таблица показывает общий характер связи

---

[1] Carr W. N., Mize J. P., MOS/LSI Design and Application, McGraw-Hill, New York, 1972.

объема выпуска и используемой технологии; однако отдельные проекты могут не подчиняться указанным закономерностям.

Связь технологии и объема выпускаемой продукции

Вид технологии/архитектуры	Модель	Объем выпуска
Микро-ЭВМ	8048, TMS 1000 и т. д.	Несколько миллионов
16- и 32-разрядные микро-ЭВМ	Motorola 68000	100—100 000
	Zilog Z8000	100—100 000
	National 16000	100—100 000
	Intel iAPX-432	100—100 000
Схемы с разрядно-модульной организацией	AMD-29116, 2910 и т. д.	100—10 000
Матричные схемы и БИС		Десятки тысяч — миллионы

Схемы с разрядно-модульной организацией (bit-slice) позволяют проектировщику оптимизировать архитектуру специализированных устройств и достичь большей производительности, чем в случае использования стандартной архитектуры. Проект будет оправдан экономически, если требуется выпустить по крайней мере 100 таких систем, хотя, очевидно, существуют исключения из этого правила. Если необходимо произвести более чем 10 000 таких систем, то, по всей видимости, более рациональным окажется применение ИС сверхбольшой степени интеграции (СБИС). По мере развития средств автоматизации проектирования СБИС будут использоваться и в проектах, рассчитанных на меньший объем выпуска продукции. «Кремниевая технология» позволяет вести проектирование при высоком уровне стандартизации и производить схемы, период освоения которых составляет несколько недель.

Поскольку применение СБИС является естественным продолжением и развитием проектирования схем с разрядно-модульной организацией, развитые нами средства формального представления матричных схем будут кратко обсуждаться в связи с применением их при проектировании СБИС. Более подробно изучить этот вопрос читатель сможет, обратившись к книге Мида и Конвея[1].

МЕТОДОЛОГИЯ ПРОЕКТИРОВАНИЯ СБИС

Различные обозначения схемы И-НЕ приведены на рис. 4.71. Схема элемента И-НЕ в действительности в значительно большей степени соответствует физической реализации транзи-

[1] Mead, Conway, Introduction to VLSI Systems, Addison-Wesley, Lexington, MA, 1979.

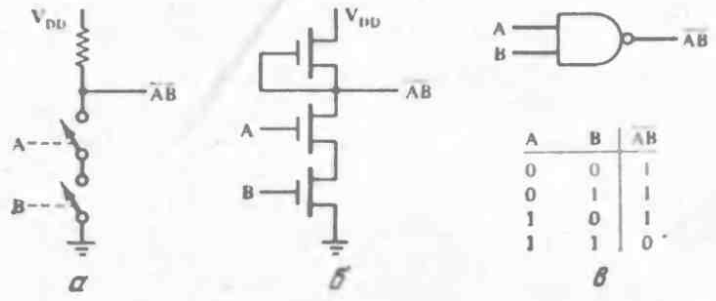


Рис. 4.71. *а* — Схема, поясняющая принцип действия вентиля И-НЕ; *б* — схема И-НЕ, выполненная на полевых транзисторах, и *в* — символическое обозначение вентиля И-НЕ и соответствующая таблица истинности.

сторов, чем это обычно отмечается в учебниках по проектированию схем. Когда аппаратура проектировалась с использованием дискретных транзисторов, указанное соответствие представляло чисто академический интерес. При проектировании СБИС физическое расположение ее элементов приобретает значительно более важное значение. Основным элементом, используемым в СБИС, является МОП-транзистор, который создается посредством формирования поликристаллического кремниевого канала, проходящего, как показано на рис. 4.72, поверх диффузионного канала и перпендикулярно к нему. Технология изготовления ИС такова, что на диффузионном уровне не существует прямого соединения между выводами транзистора, соответствующими его истоку и стоку. На одной из стадий изготовления все каналы — поликремниевый, диффузионный и металлический — обладают такой электрической проводимостью, что могут считаться проводниками. Хотя физические принципы, лежащие в основе построения и функционирования ИС, очень интересны, мы намерены показать лишь связь формальных обозначений, используемых для представления матричных логиче-

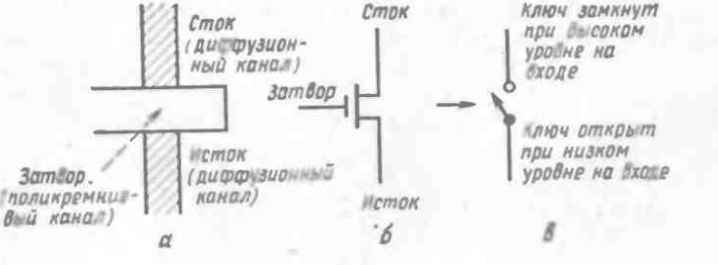


Рис. 4.72. *а* — Физическая структура МОП-транзистора; *б* — его условное обозначение и *в* — схемное представление.

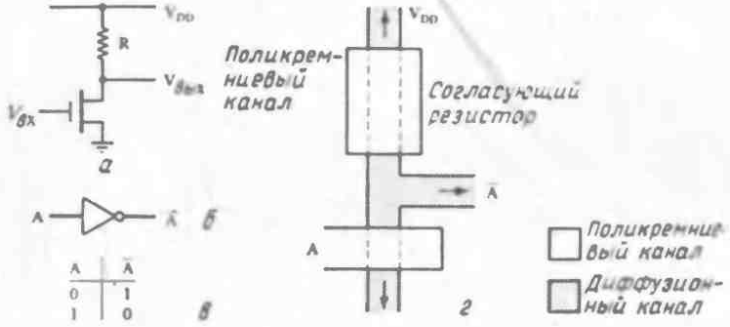


Рис. 4.73. *а* — Схема инвертора; *б* — его символическое обозначение; *в* — таблица истинности и *г* — физическая структура.

ских схем, и проектов СБИС. Читатель, желающий получить более обширные познания в данной области, может обратиться к литературе, посвященной физике твердого тела и технологии производства ИС. Тщательно отобранный и подготовленный материал по физическим основам изготовления ИС содержится в статье Кларка[1]).

Простейшим логическим устройством является инвертор. Он состоит из ключа, прерывающего и замыкающего электрическую цепь, и согласующего резистора, последовательно соединенного с ключом. Наиболее простой способ получения резистора в интегральном исполнении состоит в применении полевого МОП-транзистора, работающего в режиме, когда он обладает постоянным сопротивлением (рис. 4.73). На практике по-

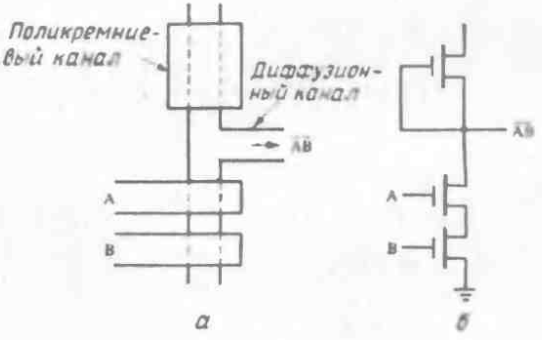


Рис. 4.74. *а* — Физическая структура и *б* — схема И-НЕ, выполненная на полевых транзисторах.

[1]) Clark W. A., From Electron Mobility to Logical Structure: A View of Integrated Circuits, *Computing Surveys*, 12, 3 (Sept. 1980).

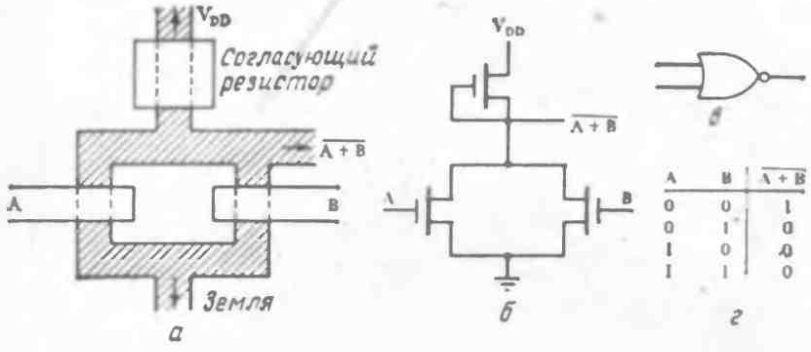


Рис. 4.75. *а* — Физическая структура, *б* — схема, *в* — символическое обозначение и *г* — таблица истинности вентиля ИЛИ-НЕ.

левые МОП-транзисторы часто используются как резисторы с переменным сопротивлением, которое определяется прикладываемым к элементу напряжением.

На рис. 4.74 и 4.75 представлены физическая структура и схемное изображение вентилей И-НЕ и ИЛИ-НЕ.

«ЛИНЕЕЧНЫЕ» ДИАГРАММЫ

Рассмотренные физические структуры изображались в виде совокупности каналов конечной ширины. При производстве СБИС получение каналов определенной ширины имеет первостепенное значение. Затрачиваются огромные усилия, чтобы добиться заданных размеров и соотношений элементов ИС. Каналы на физической схеме выглядят как «линейки». Оказалось, что и логические функции удобно выражать с использованием так называемых линеечных (или палочковых) диаграмм. В на-

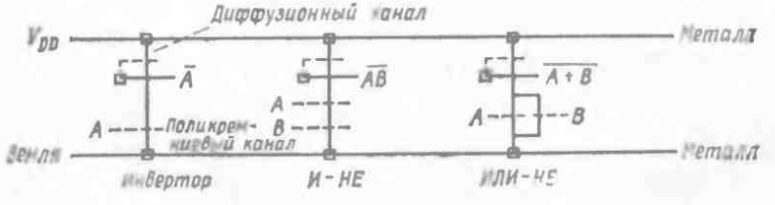


Рис. 4.76. Пример линеечной диаграммы: три диффузионных канала соединяются с двумя металлическими линиями; поликремниевые каналы пересекают диффузионные каналы, что позволяет при поступлении входных сигналов формировать выходные сигналы, представляющие собой значения логических функций входных сигналов.

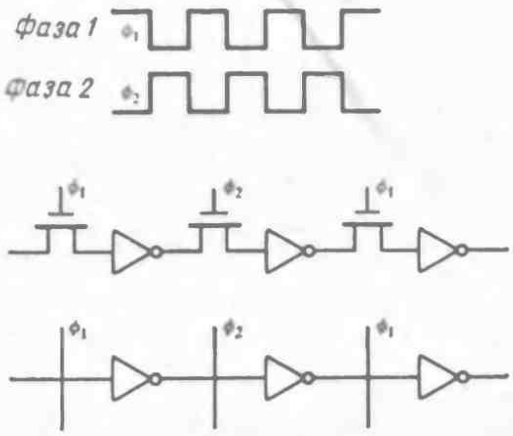


Рис. 4.77. Использование двухфазной последовательности тактовых импульсов для переноса заряда из узла одного инвертора к узлу следующего инвертора путем стробирования.

стоящее время, исходя из опыта изготовления ИС, сформулированы «λ-правила», которые устанавливают допустимые соотношения размеров элементов ИС (рис. 4.76).

СХЕМЫ СДВИГОВОГО РЕГИСТРА

В то время как при использовании ТТЛ-технологии для создания сдвигового регистра или некоторых других устройств памяти часто используются *перекрестно связанные вентили,* в СБИС основным средством хранения является заряд, накопленный в определенном узле. Для функционирования такой динамической памяти, как сдвиговый регистр, требуется двухфазная последовательность синхросигналов, которые обеспечивают

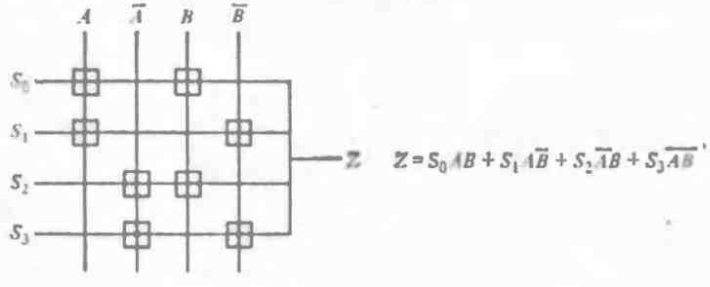


Рис. 4.78. Схема селектора, предназначенного для передачи на выход Z в зависимости от значений управляющих переменных A и B одного из входных сигналов S_i. Физически эта схема образуется пересечением диффузионных и поликремниевых каналов.

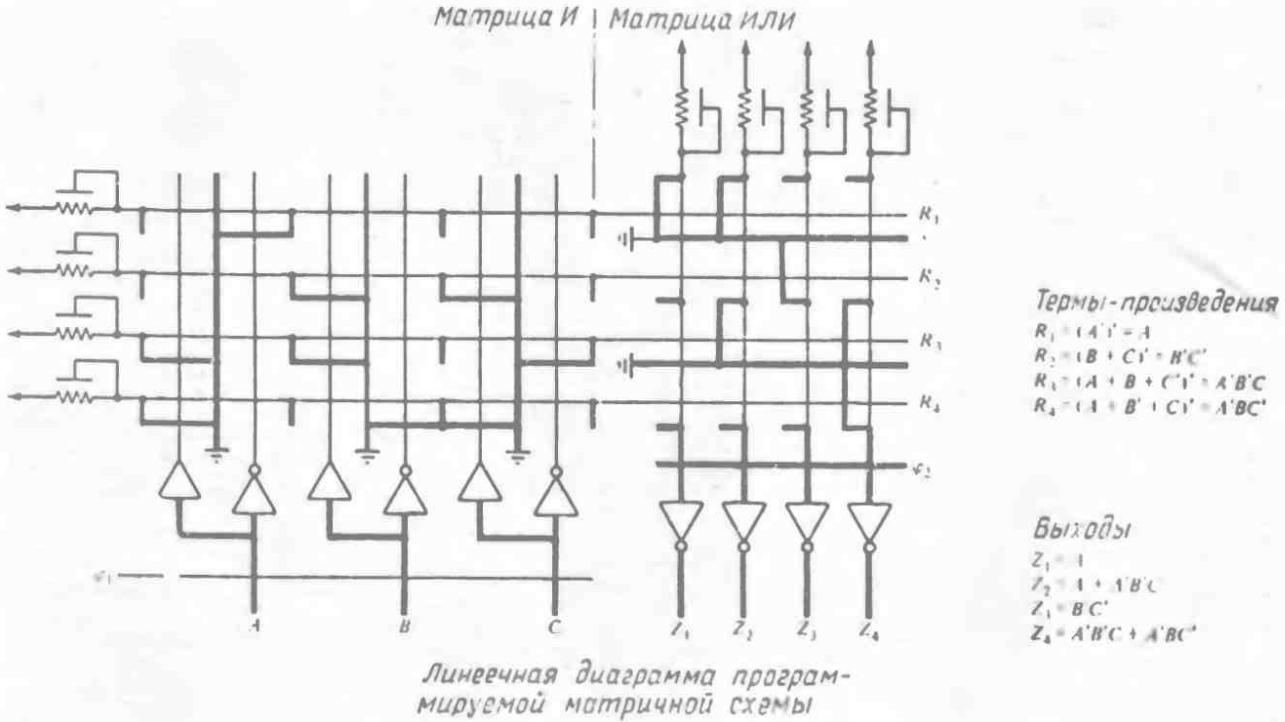


Рис. 4.79. Линеечная диаграмма программируемой матричной схемы. (Воспроизводится из книги Mead, Conway, Introduction to VLSI Systems, Addison-Wesley, Reading, 1979.)

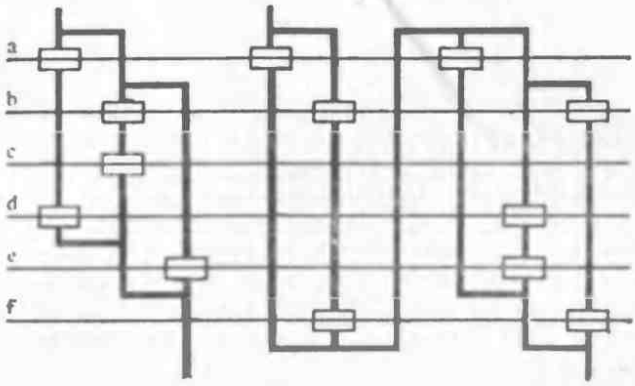


Рис. 4.80. СБИС для вычисления булевых функций F_1 и F_2. (Воспроизводится из издания IEEE Trans. on Computers, C-30, 8.)

необходимые отпирание и запирание передающих вентилей. Принцип действия такого устройства пояснен на рис. 4.77. Несколько примеров, приведенных в упомянутой выше книге Мида и Конвея, демонстрируют рациональность применения средств формального представления матричных логических схем при проектировании СБИС. Подтверждением этому являются как схема селектора (рис. 4.78), так и программируемая матричная схема, изображенная на рис. 4.79.

Как показано на рис. 4.80, логические выражения могут быть представлены так, что каждый терм-произведение реализуется посредством каскадного соединения транзисторов с применением вертикальных каналов, а сумма термов-произведений — посредством параллельного соединения вертикальных сегментов, которые представляют термы-произведения. Для более детального рассмотрения этого вопроса можно рекомендовать статью Ширакавы и др.[1].

ЗАКЛЮЧЕНИЕ

Определенные в настоящей главе понятия пространства состояний и структуры связи могут использоваться для развития формального аппарата описания матричных схем. Этот формализм имеет отношение к физической реализации диодных логических матриц и структур с плавкими связями. Машины всех

[1] Shirakawa et al., A Layout System for the Random Logic Portion of an MOS/LSI Chip, *IEEE Transactions on Computers*, C-30, 8 (Aug. 1981).

классов могут описываться с использованием формального аппарата представления матричных логических схем. Общепринятые средства формального представления логических элементов могут быть модифицированы с использованием понятий, присущих матричным логическим схемам и комбинационным логическим системам, построенным в виде двухуровневых матричных логических схем. Рассмотренные методы расширения используются с целью наращивания числа входов, выходов и вычисляемых термов в случае применения при проектировании устройств, выпускаемых промышленностью. Для описания матричных логических схем и решения задач минимизации могут быть использованы карты Карно. Описано несколько реализаций десятичного счетчика с помощью матричных логических схем разного вида.

Формальные средства описания матричных логических схем широко применяются на практике. Эти средства удобны для описания как программируемых логических матриц, так и матричных логических схем. Специальные линеечные диаграммы, используемые при проектировании СБИС, следуют из формального описания матричных схем. Проектировщик цифровых систем в 80-е годы в зависимости от цели проектирования будет использовать ИС с разрядно-модульной организацией, другие распространенные виды ИС и СБИС. Матричные средства обозначения пригодны и для различных проектов на ТТЛ-схемах с компонентами в виде БИС, и для проектов кремниевых схем на МОП-транзисторах n-типа. Важность рассмотренного формализма состоит в том, что последний позволяет отображать функции высокого уровня в ИС при полном отказе от этапа обычного логического проектирования; при этом проектируемая схема представляется с помощью линеечной диаграммы. В следующей главе средства формального описания матричных схем будут развиты далее, что позволит дать естественное введение в микропрограммирование.

МИКРОПРОГРАММИРОВАНИЕ И УСТРОЙСТВО УПРАВЛЕНИЯ ВЫПОЛНЕНИЕМ ПРОГРАММЫ

Машины состояний, описанные в предыдущих главах, представляют собой универсальные машины, предназначенные для обработки данных путем последовательного выполнения отдельных операций. Реализация таких машин классов 0, 1 и 2 обсуждалась ранее в общем виде, и приводились примеры их построения на уровне логических схем матричного типа и промышленных образцов устройств, базирующихся на программируемых матричных схемах. Хотя машины классов 3 и 4 также могут быть реализованы с помощью логических схем матричного типа, во многих случаях более простым и удобным оказывается использование устройств управления выполнением программы специального назначения. Программирование работы таких устройств обычно называют *микропрограммированием*. В данной главе рассматриваются принципы микропрограммирования и промышленные образцы устройств управления выполнением программы (УУВП). При этом применяется подход, заключающийся в расширении и модификации универсальной машины состояний, в результате чего формируются структуры специального назначения, которые далее уже реализуются физически.

АРХИТЕКТУРА И РЕАЛИЗАЦИЯ

Наиболее приемлемое определение понятия «архитектура ЭВМ» подразумевает наличие «программной модели» машины, т. е. ее представления с точки зрения программиста. Эта модель включает такие элементы, как регистры и команды передачи данных между регистрами. Архитектура в принципе не зависит от реализации системы; одна и та же архитектура может быть получена при использовании реле на электровакуумных приборах, ртутных линий задержки или рециркулирующей памяти, отдельных транзисторов, СБИС или другой техноло-

гии, позволяющей реализовать функции И-НЕ и их комбинации. Критерием выбора наиболее предпочтительной технологии часто являются экономические факторы, а не стремление к получению наилучших рабочих характеристик.

Обычно априорно наилучшая реализация для данной архитектуры не известна. В первых ЭВМ отдельные элементы соединялись между собой с помощью металлических проводников. Если по какой-нибудь причине требовалось изменять схему соединений, это осуществлялось переключением переходников на коммутационной панели. Проводники являлись носителями управляющих сигналов, благодаря чему достигался эффект передачи данных. В 1951 г. Уилкс высказал предположение, что управляющие сигналы более целесообразно хранить в специальной управляющей памяти, чем формировать их с помощью логических схем. Поскольку проектирование логических схем отличается для каждой новой архитектуры ЭВМ, причем существующая технология проектирования в значительной мере не соответствует сложности создания новых машин, предложенный Уилксом подход, основанный на микропрограммном управлении, по мере снижения стоимости управляющей памяти быстро стал доминирующим. На сегодняшний день практически все ЭВМ разрабатываются с использованием принципа микропрограммного управления. Микропрограммирование не является узкой областью, имеющей ограниченное применение. В действительности спектр систем, реализуемых на базе микропрограммного управления, включает практически все используемые в настоящее время ЭВМ: от микро-ЭВМ на кремниевом кристалле до больших универсальных вычислительных систем. Одна из проблем, возникающих при анализе реализации подобных систем, связана с сопоставлением горизонтального и вертикального микропрограммирования.

ГОРИЗОНТАЛЬНОЕ И ВЕРТИКАЛЬНОЕ МИКРОПРОГРАММИРОВАНИЕ

Чтобы объяснить различие между горизонтальным и вертикальным микропрограммированием, следует вспомнить, что обычно возможность выбора одного из множества состояний задается выражением 1 из N, в соответствии с которым каждое состояние представляется сигналом в отдельной линии, соответствующим только этому состоянию. Во многих случаях желательно кодировать состояния в сжатой форме с использованием меньшего числа линий. Простейшей схемой кодирования является двоичное кодирование, легко реализуемое с помощью простых логических схем (Кл., 1980), а следовательно, и с помощью матричных логических схем. На рис. 5.1 приведен при-

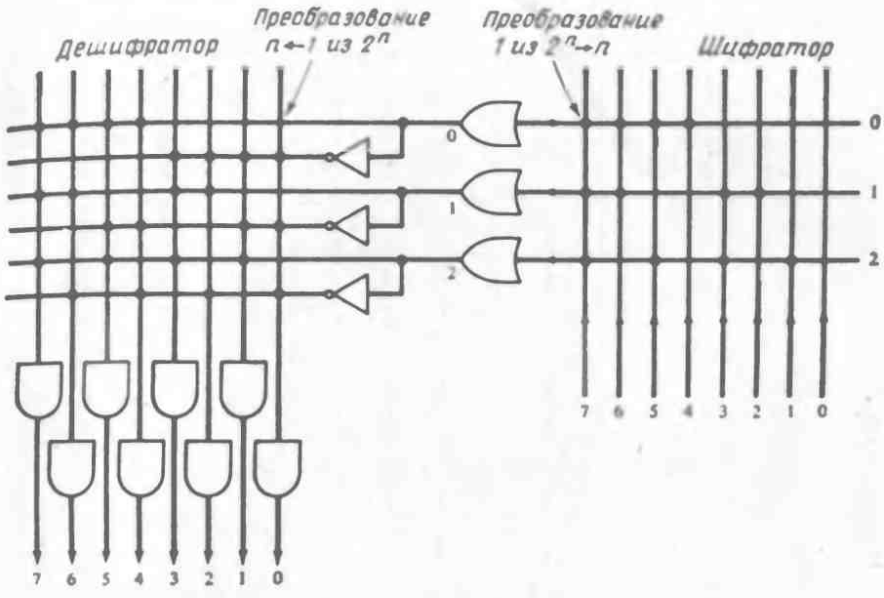


Рис. 5.1. Представление матричной логической схемы двоичного шифратора и дешифратора 1 из N.

мер, в котором представление множества состояний 1 из 8 сжимается для передачи двоичным кодом по трем линиям, а затем приводится к исходному виду. Кодирование представляет собой процесс преобразования представления информации в сжатую форму, декодирование — процесс обратного преобразования. «Сжатая», или кодированная, информация является «вертикальной» и перед использованием должна быть декодирована.

Термин «горизонтальное микропрограммирование» обычно применяется для указания на отсутствие кодирования множества управляющих сигналов 1 из N и использование каждой линии для передачи отдельного состояния. Если сигналы в определенных управляющих линиях являются взаимоисключающими, или ортогональными, в том смысле, что никогда не появляются одновременно, то состояния могут быть представлены в закодированном виде. Такое представление называют *квазигоризонтальным микропрограммированием*. Если все управляющие сигналы объединены в группы и каждый набор сигналов задается отдельным кодом, то говорят, что в системе используется *вертикальное микропрограммирование*. При горизонтальном микропрограммировании вентили трактов передачи данных управляются непосредственно сигналами, задаваемыми предназначенным для этой цели полем микрокоманды. В дру-

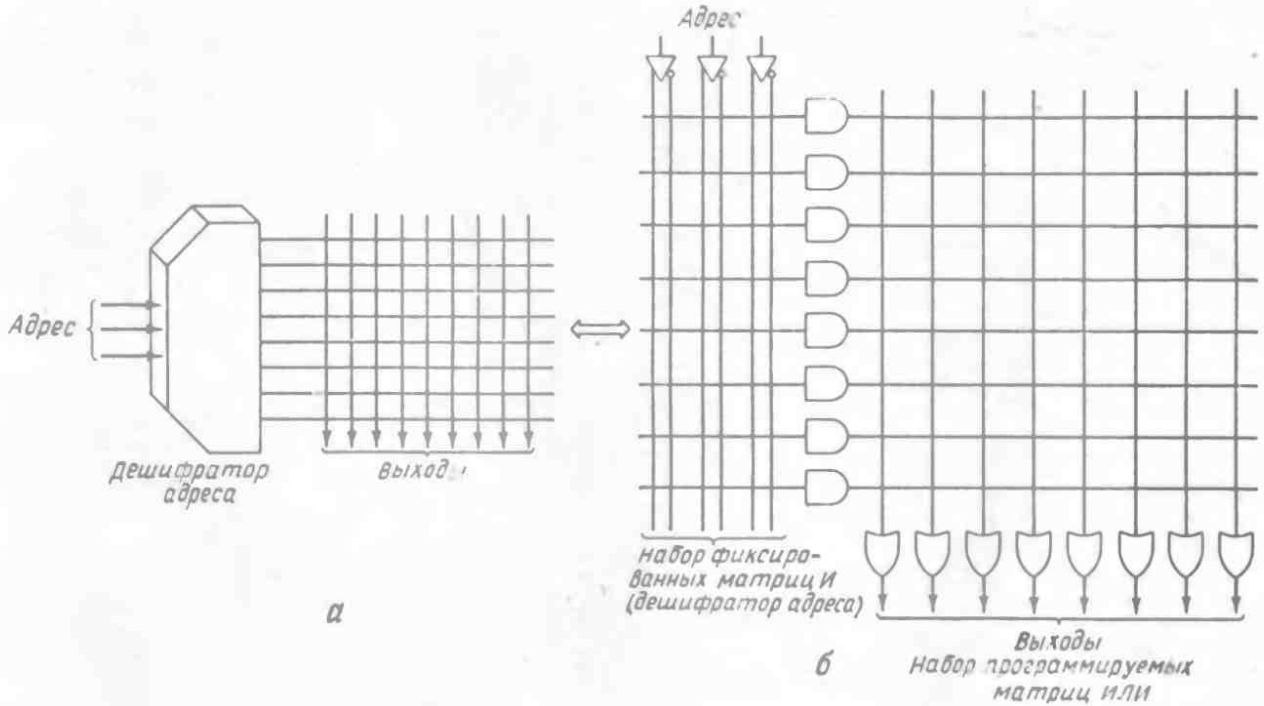


Рис. 5.2. *а* — Символическое изображение дешифратора адреса; *б* — эквивалентная матричная схема.

гих случаях между управляющими сигналами, задаваемыми микрокомандой, и управляемыми вентилями помещается дешифратор.

Функция декодирования может быть обобщена и представлена как декодирование адреса, обеспечивающее выбор в качестве выходных данных одного из N слов, записанных в памяти. Выходные линии могут рассматриваться как управляющие и использоваться для управления элементами системы. Символическое обозначение, обычно используемое для представления подобных систем, показано на рис. 5.2, *а*. Эквивалентная матричная логическая схема, включающая набор фик-

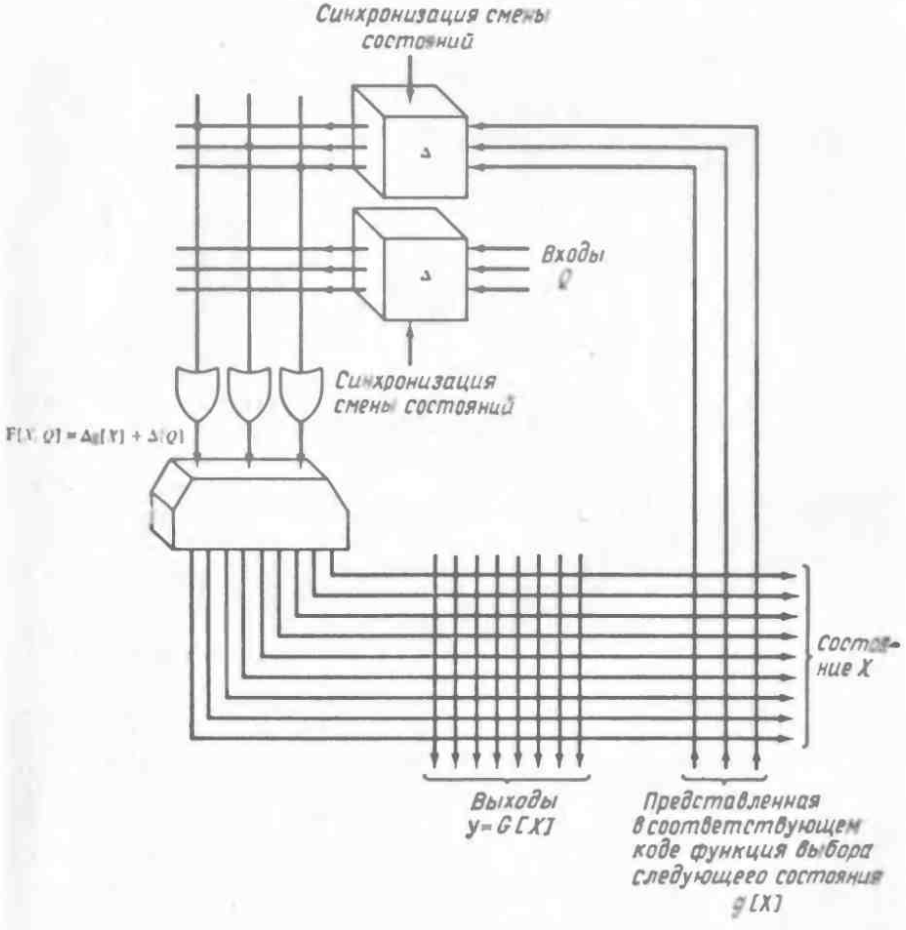


Рис. 5.3. Машина класса 3, реализующая представляемую в соответствующем коде функцию выбора следующего состояния.

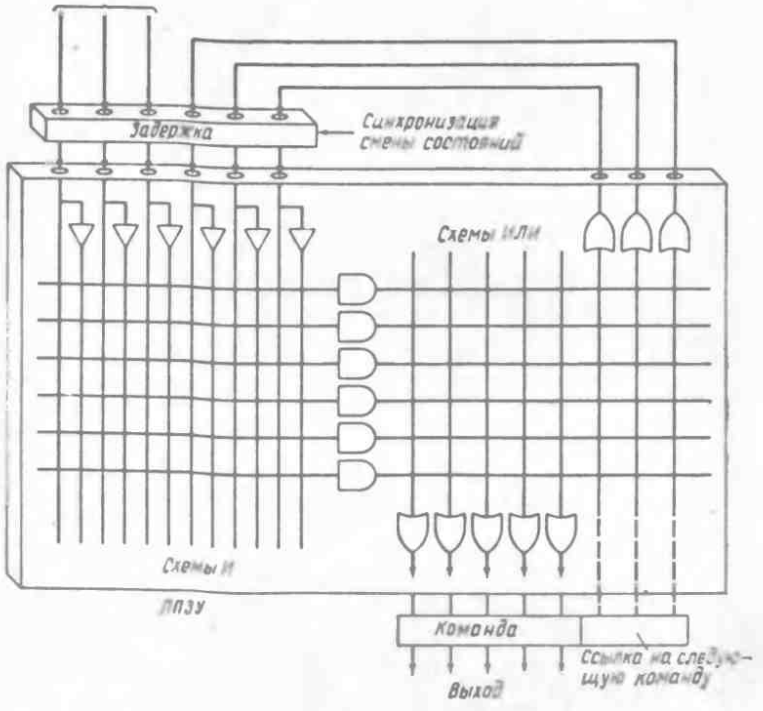


Рис. 5.4. Представление машины класса 3 в виде ППЗУ, адресуемого данными, поступающими из поля связи записанных в памяти микрокоманд.

сированных матриц И и программируемых матриц ИЛИ, приведена на рис. 5.2, *б*.

Используя символическое обозначение, представленное на рис. 5.2, можно построить машину класса 3. Основываясь на свойстве коммутативности операций задержки и декодирования, возможно произвести взаимную замену операций кодирования и задержки сигналов. В данном случае мы исключаем устройство, осуществляющее кодирование сигналов, и вместо реализации функции выбора 1 из N состояний записываем в память закодированную информацию о следующем состоянии. Машина класса 3, реализующая этот принцип, показана на рис. 5.3.

Представленная на рис. 5.4 машина класса 3, построенная на базе ППЗУ, имеет структуру, идентичную универсальной синхронной машине на программируемых матричных схемах (вследствие отключения входа Q). Дешифратор на схемах И имеет фиксированную логику, а матрицы ИЛИ программируются таким образом, что на выходе формируется определенный набор управляющих сигналов (команда), часть из которых бла-

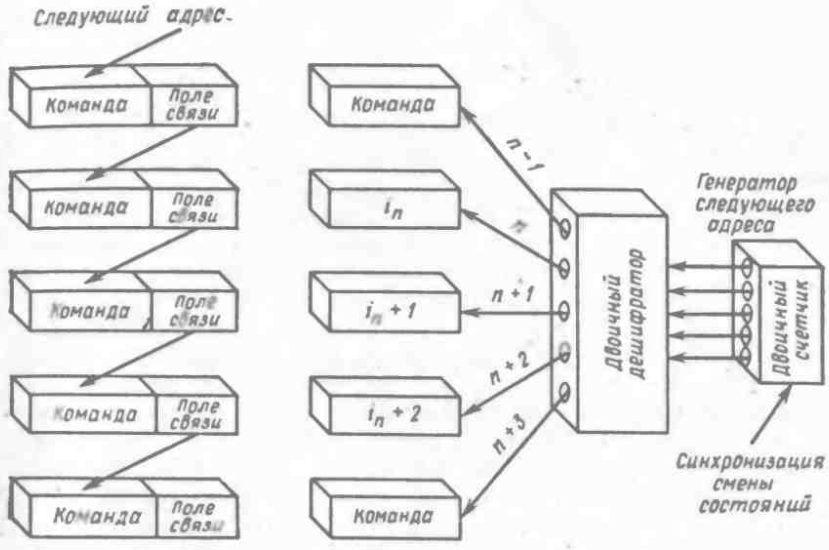


Рис. 5.5. Структура системы управления выбором следующего состояния, выполненная с использованием указателя и счетчика команд.

годаря цепи обратной связи (см. рис. 5.4) используется для определения следующего состояния. Возможны различные реализации подобной структуры, отличающиеся в основном интерпретацией входа Q. Без учета этого входа система сводится к машине класса 2, т. е. к машине, выполняющей связанный список команд. Структура управления, выполненная с использованием указателя следующего адреса и счетчика команд, показана на рис. 5.5.

В предыдущем разделе обсуждался ряд систем адресации, являющихся общими для устройств управления последовательностью выполнения программы. Появление различных систем

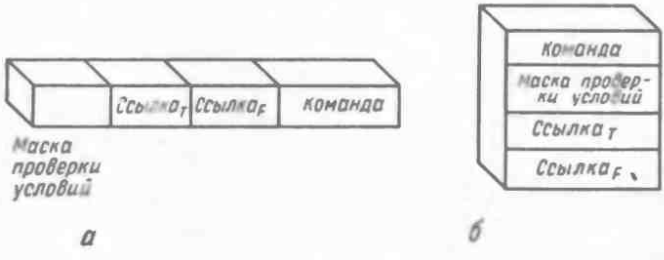


Рис. 5.6. Форматы микрокоманды: *а* — горизонтальное микропрограммирование; *б* — вертикальное микропрограммирование.

адресации обусловлено стремлением к компромиссу между обеспечением требований минимизации времени обработки и минимизации памяти (рис. 5.6). Представленные на рисунке две структуры демонстрируют различие между горизонтальным и вертикальным микропрограммированием. В первом случае различные биты микрокоманды, или управляющего слова, находящегося в управляющем ПЗУ, используются одновременно для выполнения нескольких функций. Во втором случае необходима меньшая по объему управляющая память, и те же самые функции реализуются последовательно, что требует большего числа машинных тактов для выполнения заданной операции. Клэр указывал, что управляющее ПЗУ, организованное по схеме «одно состояние автомата на один шаг алгоритма», требует минимального числа машинных тактов, которое возможно для реализации данного алгоритма, и при заданной продолжительности нахождения автомата в одном состоянии обеспечивает наибольшее быстродействие. Поскольку ПЗУ обычно имеют разрядность, кратную 4, при определении структуры микрокоманд для конкретной машины учитывается этот фактор, т. е. длина микрокоманды выбирается кратной 4. Подобным же образом количество слов в ПЗУ принимается равным некоторой степени 2.

При ограничении разрядности (количества битов в слове) ПЗУ число управляющих слов, требуемых для реализации алгоритма, возрастает, что приводит к увеличению времени обработки. Поскольку стоимость системы обычно сопоставляется с информационной емкостью (в битах) управляющей памяти, существует некоторая оптимальная величина информационной емкости (определяемая разрядностью слова, умноженной на количество слов). При этом не обеспечивается максимально возможное быстродействие или полностью соответствующий принципу горизонтального микропрограммирования набор микрокоманд, поскольку маловероятно, чтобы количество требуемых микрокоманд было достаточным для полного заполнения ПЗУ стандартных размеров, а неиспользуемые слова ПЗУ — это по существу излишние затраты.

ОРТОГОНАЛЬНАЯ СТРУКТУРА СИСТЕМЫ ВВОДА

Представленная на рис. 5.4 машина класса 3 обладает обобщенной структурой формирования следующего адреса[1)], позволяющей совместно использовать в процессе этого форми-

---

[1)] Имеется в виду адрес следующей команды (микрокоманды), подлежащей выполнению. — *Прим. перев.*

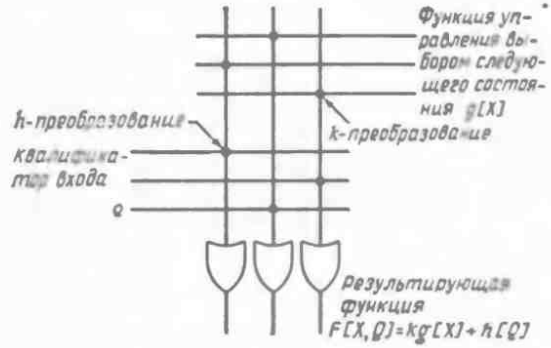


Рис. 5.7. Выбор следующего состояния.

рования квалификатор входа и функцию, определяющую следующее состояние (рис. 5.7). Если эту структуру модифицируют таким образом, что входными данными дешифратора адреса могут быть либо первый, либо второй из указанных источников (но не оба вместе), то такую структуру системы выбора следующего адреса называют *ортогональной*. На рис. 5.8 показана возможная реализация подобной ортогональной структуры и ее классический эквивалент — мультиплексор. Заметим, что эта структура требует использования специальных управляющих линий.

Выбор используемого в данный момент («открытого») канала мультиплексора осуществляется с помощью управляющих

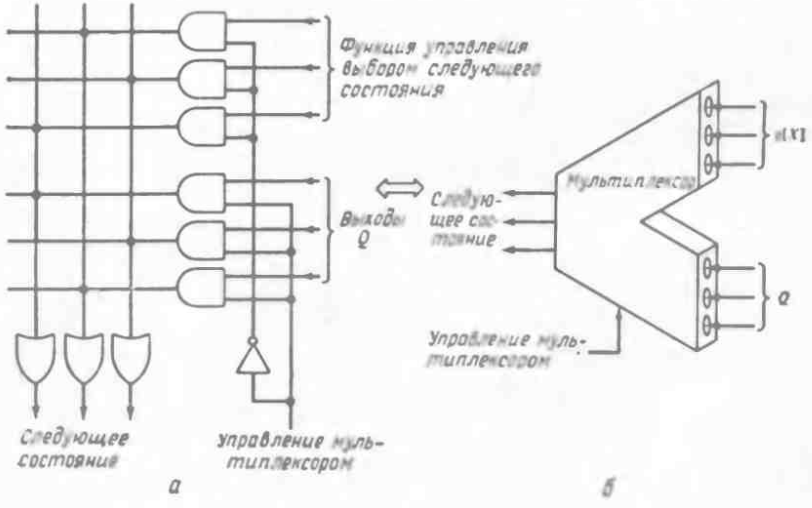


Рис. 5.8. *а* — Логическая схема и *б* — символическое обозначение ортогональной структуры выбора следующего адреса.

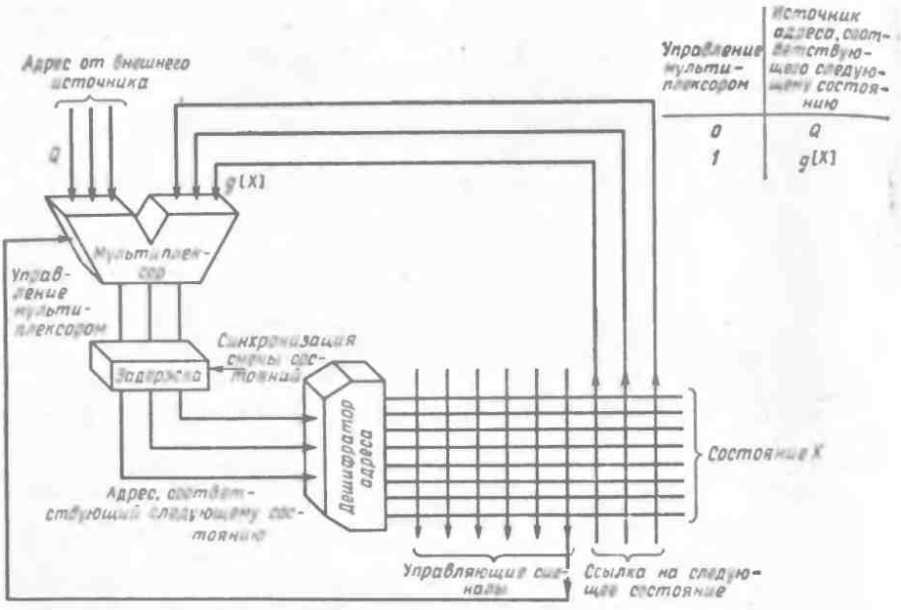


Рис. 5.9. Машина класса 3 с ортогональной структурой системы выбора следующего состояния.

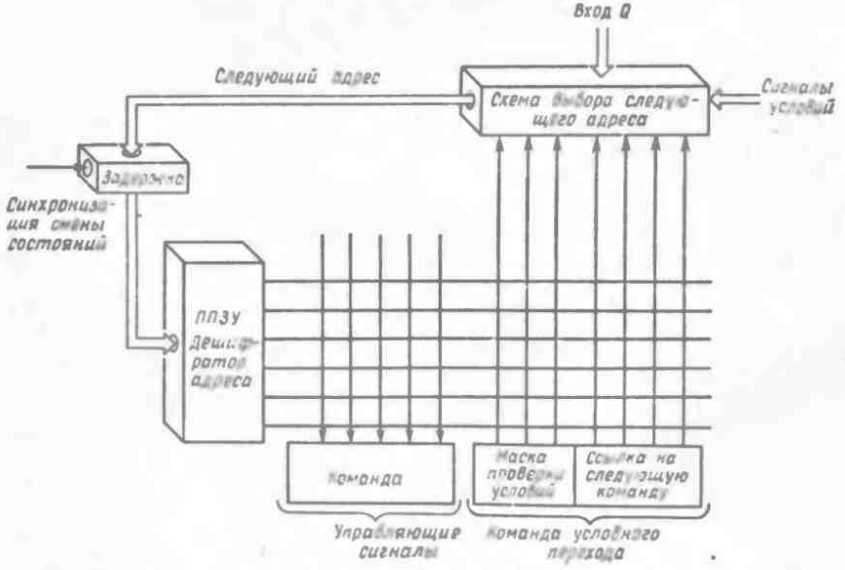


Рис. 5.10. Универсальная машина с обобщенной логической схемой выбора адреса следующей команды.

линий, причем источником управляющих сигналов, передаваемых на эти линии, может служить управляющая память. Таким образом, в системе, изображенной на рис. 5.9, в качестве следующего адреса могут выбираться либо данные, поступающие из внешнего источника, либо адрес, находящийся в поле ссылки текущей команды. Таким образом, управляющие сигналы от внешнего источника могут инициировать выполнение определенного управляющего слова (микрокоманды), которое в свою очередь откроет «внутренний канал» мультиплексора, находящегося на входе дешифратора адреса, так что последовательностью выполнения микрокоманд будет управлять находящаяся в самих микрокомандах (и хранимая в управляющей памяти) информация. По завершении выполнения последовательности микрокоманд вновь открывается «внешний канал» мультиплексора.

В управляющие линии мультиплексора (рис. 5.9) передаются сигналы, поступающие из команд, точнее из полей управляющих сигналов микрокоманд, записанных в управляющей памяти. Эти линии обеспечивают выбор одного из двух источников следующего адреса: функции g(X), формируемой благодаря обратной связи из поля микрокоманды, содержащего адрес следующей микрокоманды, и внешних данных, поступающих на входы Q. Нетрудно видеть, что выбор 1 из N может быть осуществлен путем включения мультиплексора $N\times1$ в тракт выбора следующего адреса и последующего управления работой мультиплексора с помощью управляющего поля с числом битов, равным $\log_2 N$. Это поле обычно называют *полем проверки условий;* оно показано в явном виде на рис. 5.10. Управляющее поле проверки условий может осуществлять выбор либо входов Q, либо различных входов сигналов условий, которые могут использоваться произвольным образом вместе с адресом следующей микрокоманды, хранимым в ППЗУ. Объединяя фиксатор, или регистр адреса, с устройством выбора следующего адреса, мы получаем обобщенную структуру, представленную на рис. 5.11.

Обычно внутреннюю структуру подобных логических схем не показывают, а изображают в виде единственного блока (рис. 5.12, *а*) как набор вентилей И дешифратора адреса с фиксированной логикой и памяти на программируемых матрицах ИЛИ, поскольку оба этих компонента физически размещаются в одном корпусе (рис. 5.12, *б*). Используя приведенный на рис. 5.12 способ обозначения ППЗУ, можно изображенное на рис. 5.11 УУВП представить в другом виде (рис. 5.13). Подобная форма представления лучше всего подходит для изображения таких пригодных для промышленного применения УУВП, какими являются устройства 3001 фирмы Intel. Указанные устрой-

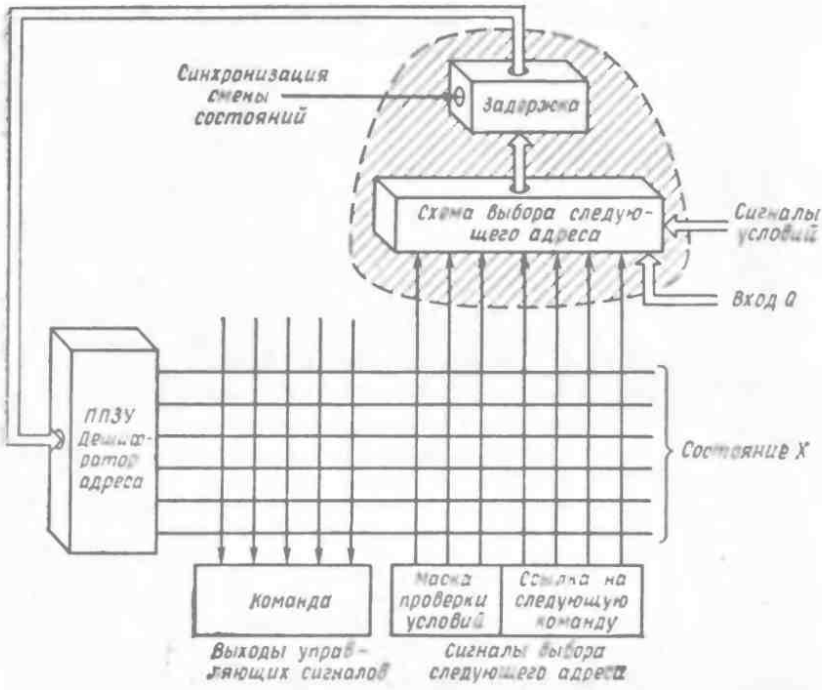


Рис. 5.11. Обобщенная подсистема выбора следующего адреса — устройство управления выполнением программы.

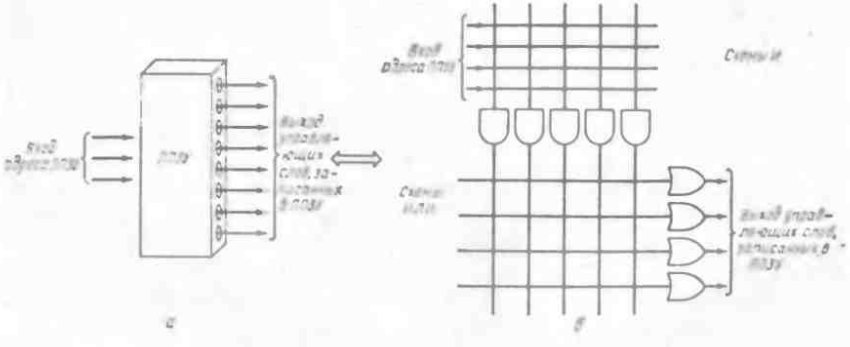


Рис. 5.12. *а* — Символическое изображение матричной логической схемы, используемой в качестве ППЗУ; *б* — стандартное изображение ППЗУ.

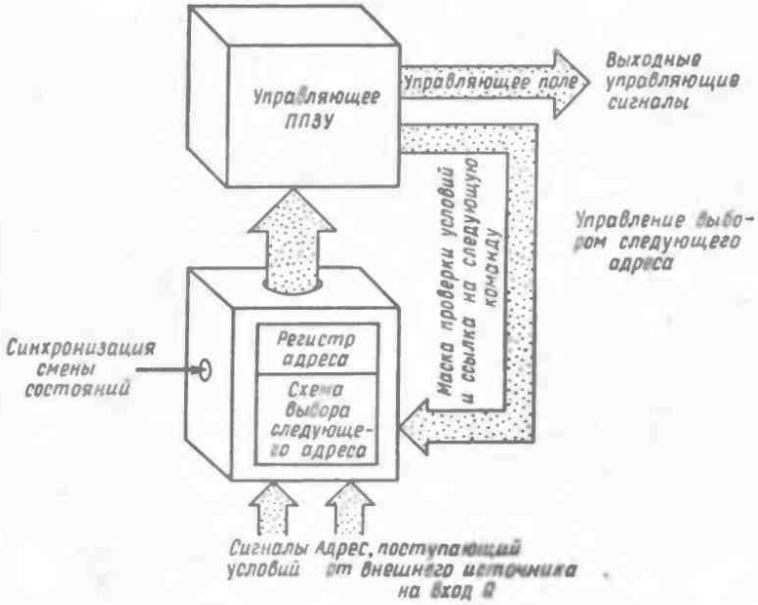


Рис. 5.13. Структура устройства управления выполнением программы 3001 фирмы Intel.

ства позволяют управлять сменой состояний машины со скоростью, равной ~10 млн. различных состояний в секунду, причем их рыночная цена составляет ~10 долл.

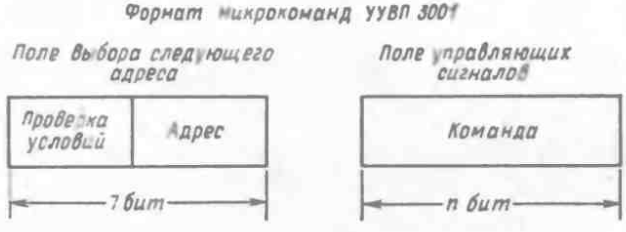
ДВУМЕРНОЕ АДРЕСНОЕ ПРОСТРАНСТВО УУВП 3001 ФИРМЫ INTEL

В системе адресации УУВП 3001 фирмы Intel использовано поле выбора следующего адреса, содержащееся в каждой микрокоманде. Для минимизации количества битов, с помощью которых задается адрес, применяется двумерная схема адресации.

Эта схема базируется на принципах страничной организации памяти, применяемой в различных мини-ЭВМ, при которой значения старших битов содержимого регистра адреса (РА) фиксируются и не меняются в течение ряда циклов, а значения младших битов, определяющих смещение адресуемой ячейки относительно начала страницы, задаются очередной микрокомандой и фиксируются только в течение данного цикла. Поскольку значения старших битов адреса не меняются, если адресуемые ячейки находятся в пределах одной и той же стра-

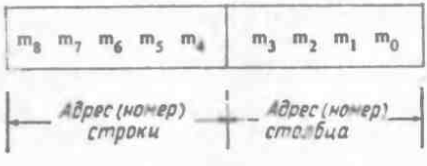
ницы, то уменьшение общего количества битов, представляющих адрес, может оказаться существенным. Хотя обычно фиксируются именно старшие биты адреса, т. е. биты, определяющие страницу, на которой находится адресуемая ячейка, отсутствуют какие-либо причины, которые не позволили бы фиксировать младшие биты адреса, а значения старших битов задавать в поле микрокоманды, содержащем информацию, определяющую следующий адрес. Рассматриваемый здесь подход становится более наглядным, если перейти к двумерному изображению адресного пространства в формате кадра и заменить термины «страница» и «слово» терминами «строка» и «столбец». В УУВП 3001 используется 4-битовая младшая часть адреса, или *адрес столбца,* и 5-битовая старшая часть адреса, или *адрес строки.*

Указатель следующего адреса, содержащийся в микрокомандах устройства 3001, может включать, как показано ниже, до 4—5 бит адреса и 2—3 бит проверки условий.



БЕЗУСЛОВНЫЙ ПЕРЕХОД В ДВУМЕРНОМ АДРЕСНОМ ПРОСТРАНСТВЕ

Устройство управления выполнением программы 3001 содержит 9-разрядный регистр адреса микропрограммы, используемый для хранения адреса текущей микрокоманды в управляющей памяти. Управляющее ПЗУ при простейшей конфигурации системы может содержать до 512 микрокоманд, каждая из которых включает 7-битовое поле выбора следующего адреса. Если полный текущий адрес задается 9-битовым кодом



и в поле выбора следующего адреса текущей микрокоманды задан безусловный переход, то либо текущий адрес строки, ли-

бо текущий адрес столбца сохраняется неизменным, а биты другой части адреса модифицируются или фиксируются посредством битов поля выбора следующего адреса микрокоманды.

Тракт передачи адреса по каналу обратной связи в УУВП 3001 и двумерное адресное пространство показаны на рис. 5.14. 16 столбцов и 32 строки образуют 512 адресуемых ячеек памяти, как этого и следовало ожидать при использовании 9-битового адреса. Каждая из ячеек содержит микрокоманду произвольной длины. Биты микрокоманды на рис. 5.14 представляются расположенными в направлении, перпендикулярном плоскости рисунка; каждая ячейка напоминает почтовый ящик, в котором хранится микрокоманда.

Команды безусловного перехода разделяются на команды перехода на новую строку и на команды перехода на новый

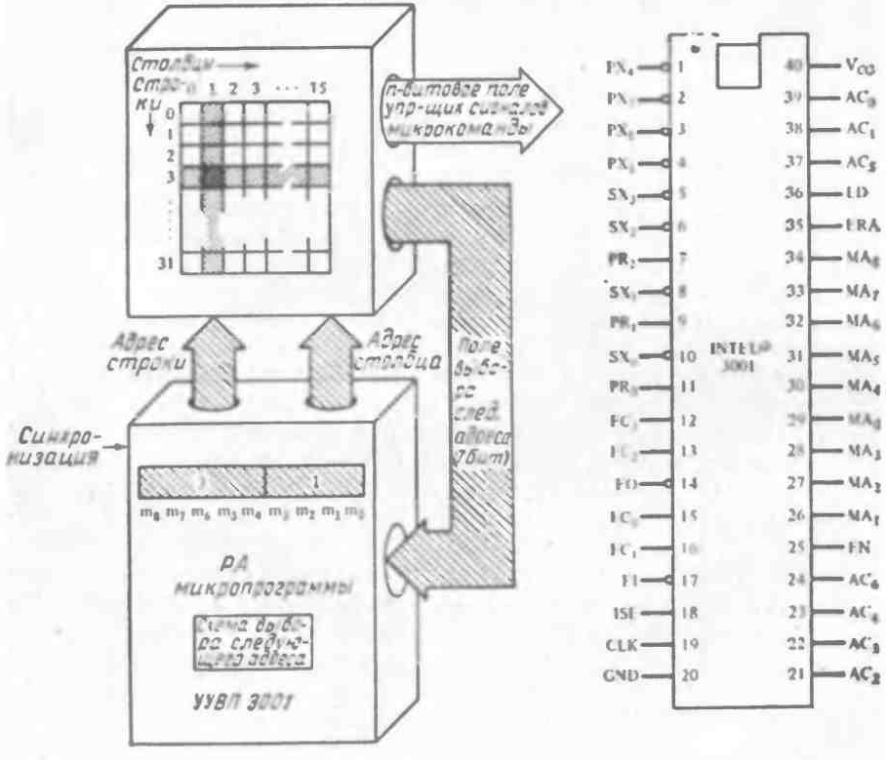
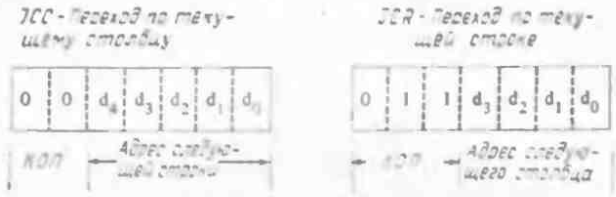


Рис. 5.14. Канал обратной связи схемы формирования следующего адреса и двумерное адресное пространство устройства 3001 фирмы Intel. (*С разрешения* Intel Corp.)

столбец. В обоих случаях поле выбора следующего адреса содержит 7 бит и имеет следующий формат:



Действие команд JCC и JCR на содержимое регистра адреса микропрограммы определяется следующими преобразованиями текущего адреса:

Из этого определения следует, что содержимое 4-битового поля адреса перехода команды JCR замещает адрес столбца текущего адреса микропрограммы, и следующий адрес микропрограммы содержит текущий адрес строки и новый адрес столбца. Подобным же образом биты поля адреса перехода команды JCC замещают младшую часть, т. е. адрес строки текущего адреса микропрограммы, тогда как биты адреса столбца текущего адреса микропрограммы сохраняются неизменными. Действие этих двух команд безусловного перехода наглядно представлено на рис. 5.15. Условные переходы в двумерном адресном пространстве описываются в гл. 6.

РАСШИРЕННАЯ СТРУКТУРА АДРЕСАЦИИ

Структура УУВП 3001 фирмы Intel была получена, исходя из обобщенной структуры машины класса 3, приведенной на рис. 5.9. В этом разделе мы рассмотрим некоторые модификации, приводящие к другой архитектуре машины с конечным

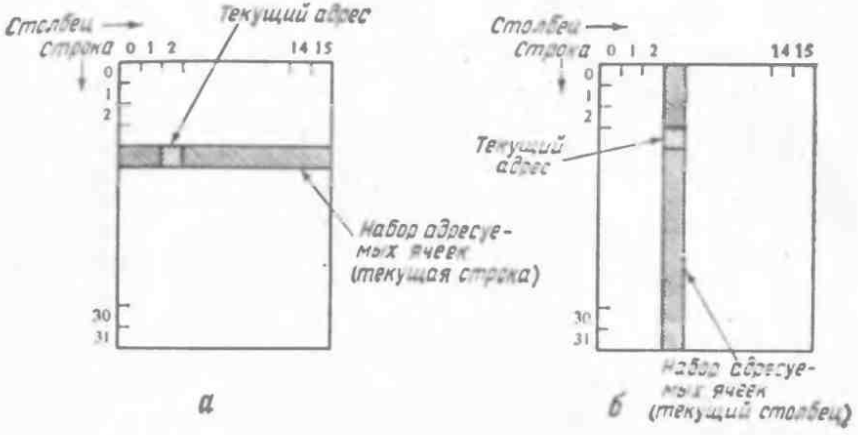


Рис. 5.15. Наборы адресуемых ячеек при выполнении команд безусловного перехода: *а* — переход при фиксации адреса строки (JCR); *б* — переход при фиксации адреса столбца (JCC).

числом состояний — архитектуре, располагающей более развитой системой выбора следующего адреса. На рис. 5.16, *а* воспроизводится структура, приведенная ранее на рис. 5.9, а на рис. 5.16, *б* представлена эквивалентная ей структура.

Устройство, обеспечивающее задержку сигналов, и регистр адреса микропрограммы помещены в новой структуре на вход мультиплексора адреса, причем на всех входах последнего используются буферы. В этой машине нормальным ходом выполнения микрокоманд является их последовательная выборка, а поле адреса перехода микрокоманды используется только для прямой (не последовательной) передачи управления по заданному адресу. При этом предполагается переменный формат микрокоманды, так что во время последовательной выборки поле адреса перехода микрокоманды может использоваться для других целей (например, для выдачи управляющих сигналов), причем формирование последовательных адресов обеспечивается с помощью счетчика или другого подобного устройства. Поскольку при последовательной выборке микрокоманд следующий адрес в общем виде равен текущему адресу плюс 1, для данной цели лучше использовать устройство приращения на 1, а не счетчик. Устройство приращения может быть реализовано на базе комбинационных логических схем и обеспечивает автоматическое формирование следующего адреса из текущего.

Выходные сигналы устройства приращения должны теперь передаваться по каналу обратной связи и использоваться в качестве потенциального источника следующего адреса (рис. 5.17).

Регистр задержки, располагаемый на выходе устройства при-

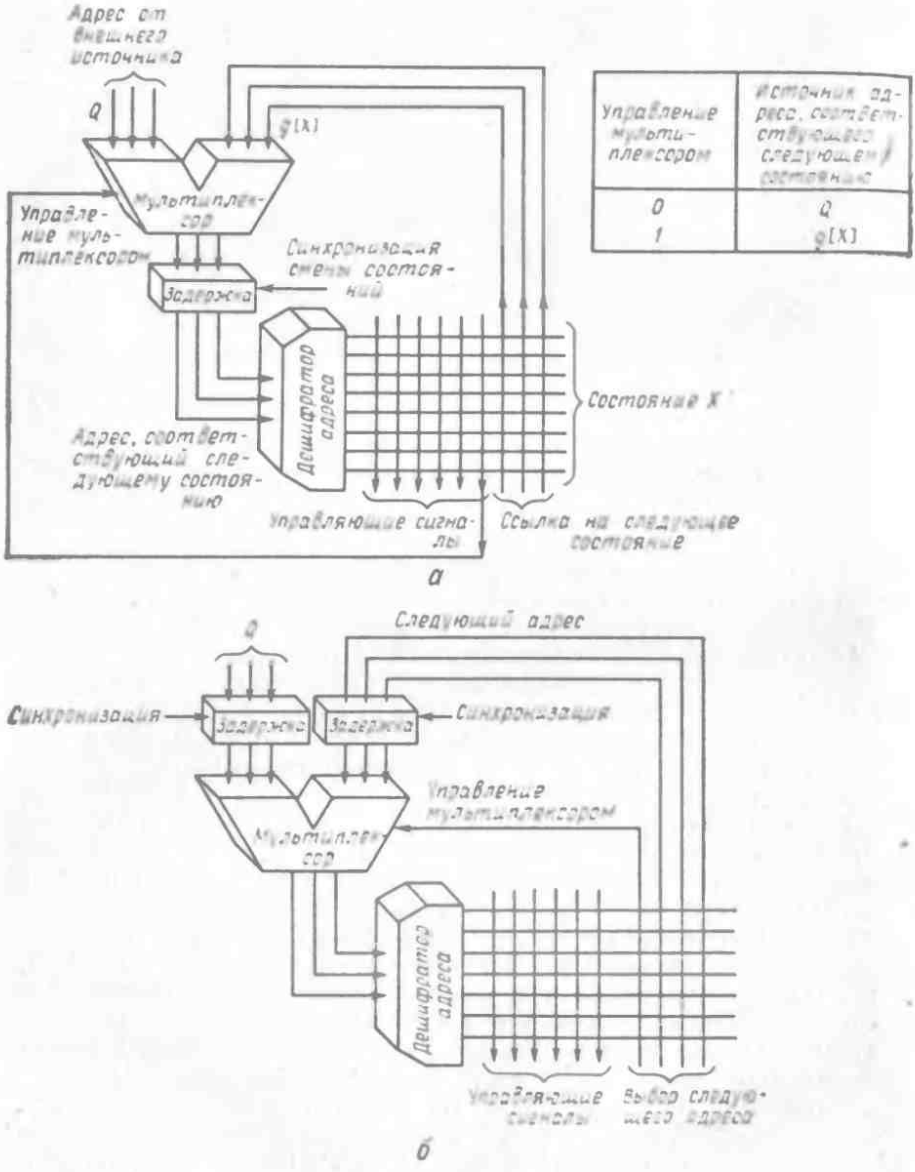


Рис. 5.16. Машина класса 3: *а* — тип 1; *б* — тип 2.

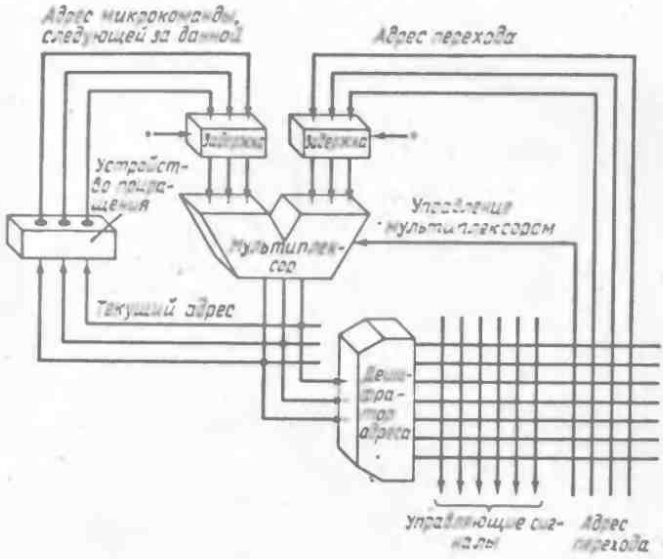


Рис. 5.17. Машина класса 3, обеспечивающая последовательную и не последовательную выборку команд.

ращения, обычно называют *счетчиком микропрограммы* или *счетчиком микрокоманд* (СМК). Система, представленная на рис. 5.17, является замкнутой и не позволяет использовать в качестве следующего адреса данные, поступающие от внешних источников. Это ограничение может быть устранено путем добавления дополнительного входного канала мультиплексора следующего адреса (рис. 5.18). Сигналы на линиях входа Q мультиплексора и линиях, через которые в мультиплексор передается содержимое поля адреса перехода микрокоманды, должны быть ортогональными, а сами линии должны управляться по схеме с тремя состояниями, так чтобы один из этих источников адреса всегда находился в состоянии высокого импеданса.

СТЕК ПОДПРОГРАММ

Последовательная выборка микрокоманд обеспечивает присвоение по умолчанию значения адресу микрокоманды, подлежащей выполнению после каждой текущей микрокоманды. Процедура вызова подпрограмм может быть реализована, если предусмотреть специальные средства для запоминания при обращении к подпрограмме предполагаемого по умолчанию адреса следующей микрокоманды. Наиболее предпочтительной

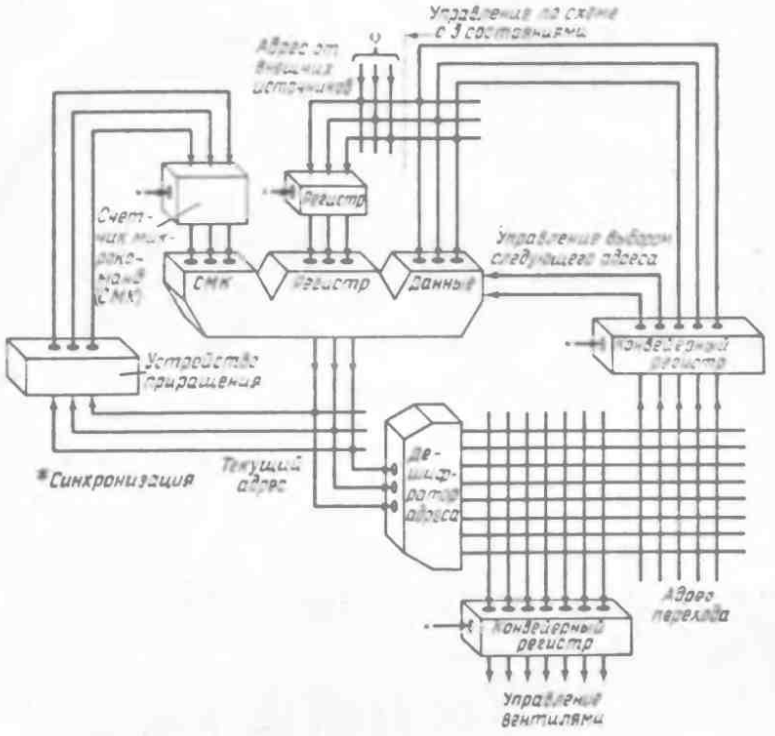


Рис. 5.18. Система адресации, в которой предусмотрено мультиплексирование содержимого счетчика микрокоманд, адреса перехода, находящегося в текущей микрокоманде, и адреса, поступающего от внешних источников.

структурой для хранения адресов возврата из подпрограмм является показанный на рис. 5.19а и 5.19б набор ячеек памяти, обслуживаемый по принципу LIFO (last-in-first-out): «поступивший последним обрабатывается первым». Эту структуру называют *стеком адресов возврата из подпрограмм*. (Более подробно этот вопрос изложен, например, в книге Кл., 1980.)

Существуют два варианта УУВП Am29xx: Am2909 (рис. 5.20, *а*) и Am2911 (рис. 5.20, *б*). Заметим, что в УУВП Am2909 ввод данных в регистр адреса осуществляется через отдельные линии, тогда как в УУВП Am2911 используется внутреннее соединение входа регистра адреса с входом D устройства[1]. Кро-

[1] Вход D в данном случае служит для передачи адреса непосредственно в мультиплексор следующего адреса без промежуточной записи этого адреса в какой-либо внутренний регистр устройства. — *Прим. перев.*

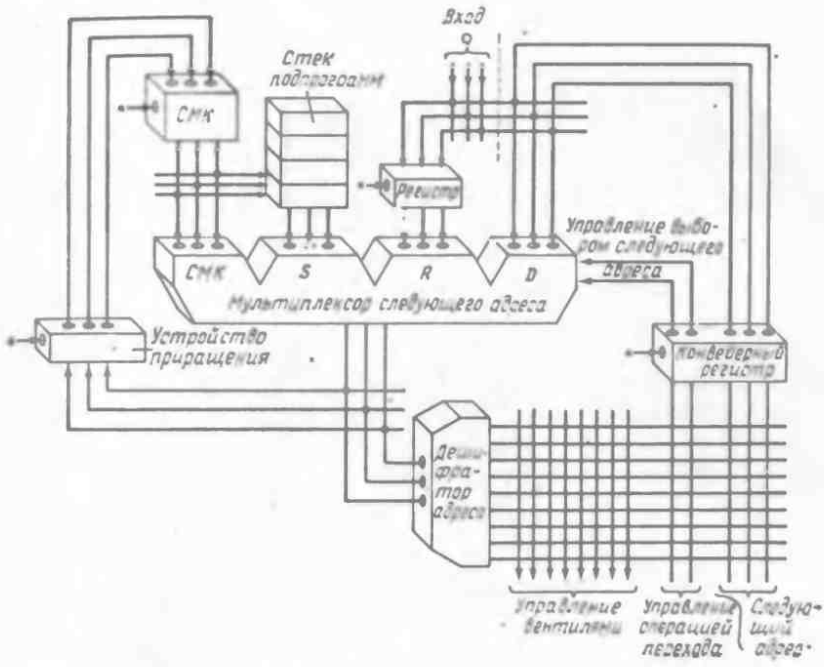


Рис. 5.19а. Архитектура системы управления выполнением последовательности микрокоманд, являющейся основой УУВП Am2909.

ме того, в УУВП Am2911 отсутствует возможность логического сложения адреса, поступающего от внешнего источника, с выходными данными мультиплексора следующего адреса. В УУВП Am2911 для ввода адресов, поступающих от внешних источников, используется вход D, поэтому желательно, чтобы управление передачей данных от этих источников производилось по схеме с тремя состояниями.

Имеется ряд линий управления (на рис. 5.20 они не показаны), обеспечивающих выбор мультиплексором следующего адреса определенного входного канала, разрешение обращения к регистру и стеку объемом в 4 слова, манипулирование указателем стека. Следует также отметить, что только один из управляемых по схеме с тремя состояниями источников следующего адреса может быть выбран в данный момент для передачи информации в устройство через вход D. Хотя необходимые управляющие сигналы могли бы выдаваться непосредственно из соответствующих полей микрокоманды, фирма Advanced Micro Devices разработала для этой цели отдельное устройство — *устройство управления выбором следующего адреса* Am29811. Данное устройство объединяет в себе реализацию

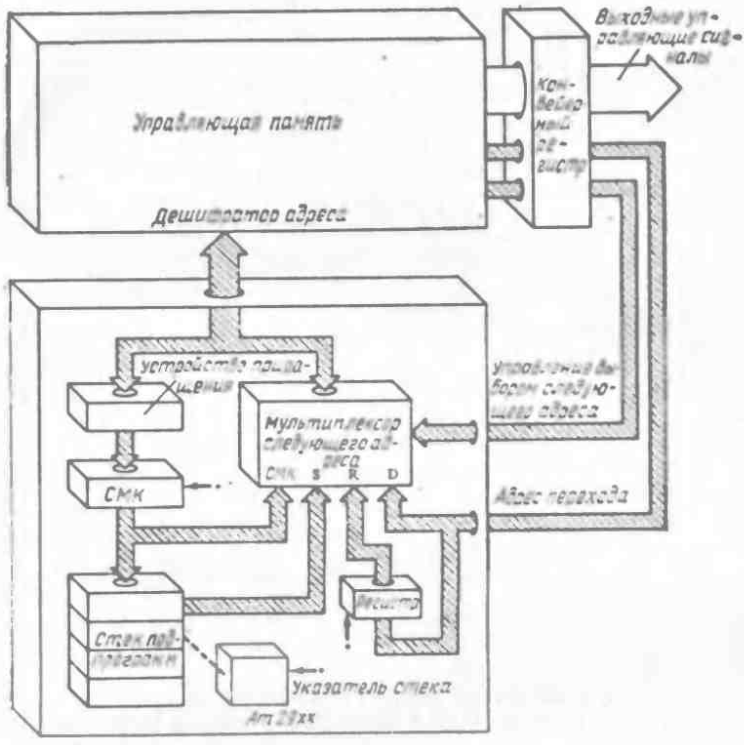


Рис. 5.19б. Архитектура системы управления выполнением последовательности микрокоманд, являющейся основой УУВП Am2911.

функций проверки сигналов условий и генерирования управляющих сигналов, необходимых для работы УУВП Am2911. Стандартная конфигурация системы, включающей устройства Am2911 и Am29811, представлена на рис. 5.21.

Благодаря наличию в микрокоманде поля адреса перехода при последовательном выполнении микрокоманд появляется возможность хранить в микропрограмме константы. Если при выполнении данной операции адрес перехода задаваться не должен, то это поле можно использовать в качестве источника данных. Обычное применение данного поля состоит в загрузке внешних регистров или счетчика. В системе, включающей БИС Am2911 и Am29811, используется внешний счетчик, в который под управлением БИС Am29811 загружается содержимое поля адреса перехода микрокоманды и сигнал переноса которого может проверяться с помощью мультиплексора сигналов условий и БИС Am29811. Выдача данных, хранимых в микропрограмме, и внешний счетчик показаны на рис. 5.21.

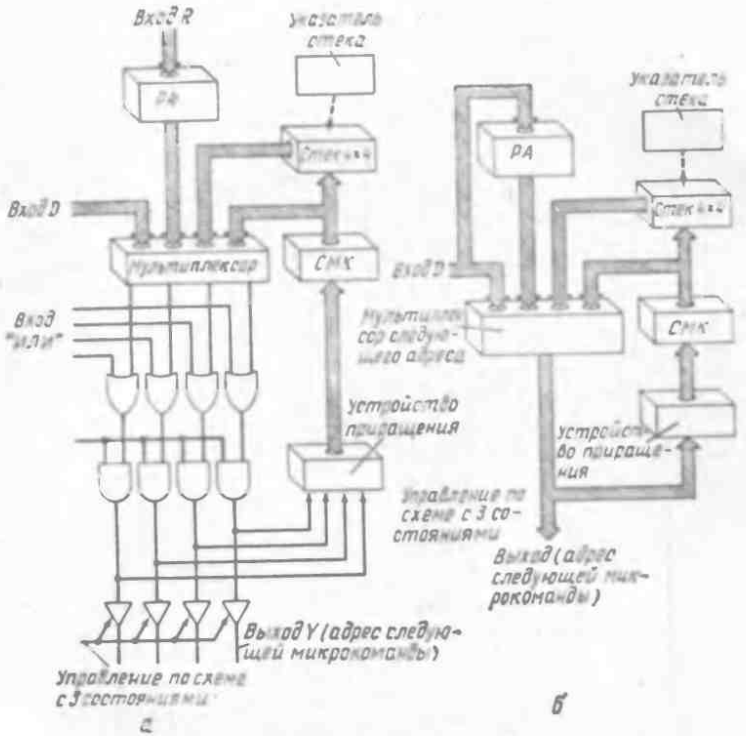


Рис. 5.20. Архитектура УУВП Am2909 (*а*) и Am2911 (*б*).

РАЗРЯДНО-МОДУЛЬНАЯ ОРГАНИЗАЦИЯ УУВП, ПОСТРОЕННОГО НА БИС Am2909

Секция УУВП Am2909 имеет разрядность, равную 4. Это означает, что разрядность всех трактов передачи данных равна 4. Следовательно, адресное пространство микропрограммы, управление которым обеспечивается с помощью единственной БИС Am2909, составляет только 16 слов, так что для большинства систем требуется применение трех БИС Am2909 с целью расширения адресного пространства до 4096 слов. Хотя в принципе возможна параллельная работа неограниченного числа БИС Am2909, в большинстве применений используются две или три БИС. Параллельная работа заключается в том, что одни и те же управляющие линии (например, линии сигналов загрузки/извлечения данных из стека, разрешения записи в регистр, управления мультиплексором и т. д.) подключаются к каждой БИС Am2909. Таким образом, каждая секция УУВП должна выполнять одну и ту же операцию и не оказывать

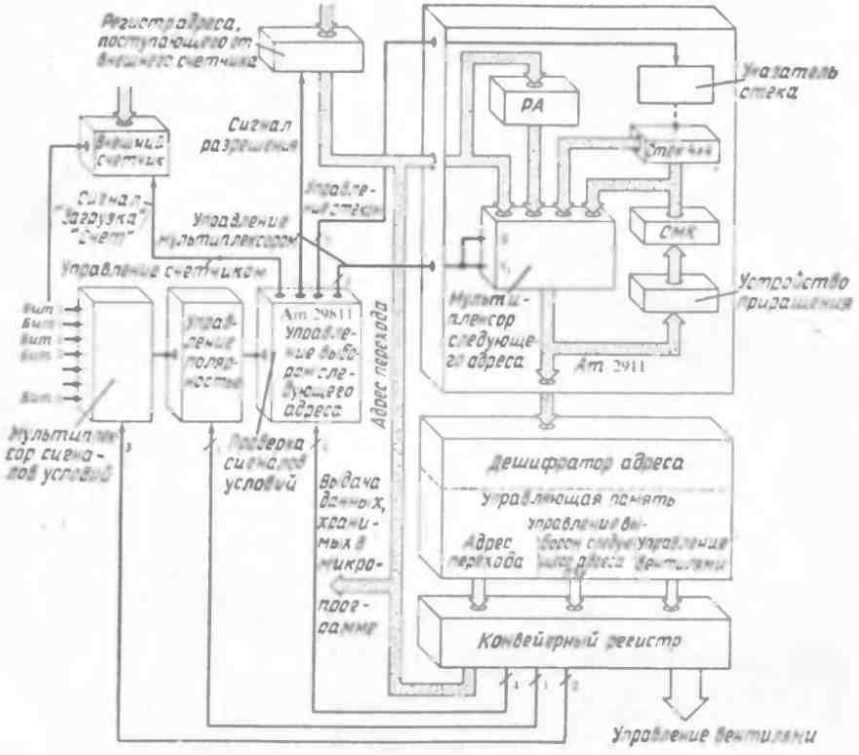


Рис. 5.21. Использование разработанного фирмой Advanced Micro Devices устройства управления выбором следующего адреса Am29811 вместе с УУВП Am2911.

влияния на работу других секций. Исключением, разумеется, является операция приращения: выходной сигнал цепи переноса секции младших по значимости разрядов адреса используется в качестве входного сигнала цепи переноса секции старших по значимости разрядов адреса.

Если предполагается, что структура, представленная на рис. 5.21, должна обеспечить разрядность УУВП, равную 12, а счетчик адреса формируется из 4-разрядных счетчиков, то количество корпусов БИС устройства управления должно быть не менее 9 без учета регистра данных, поступающих от внешнего источника, управляющей памяти и конвейерных регистров. Одной из тенденций развития технологии микроэлектронных устройств является уменьшение числа корпусов БИС за счет реализации большего числа функций отдельными БИС.

Примерно через два года после создания УУВП Am2909, реализованного в корпусе с 28 выводами, было разработано

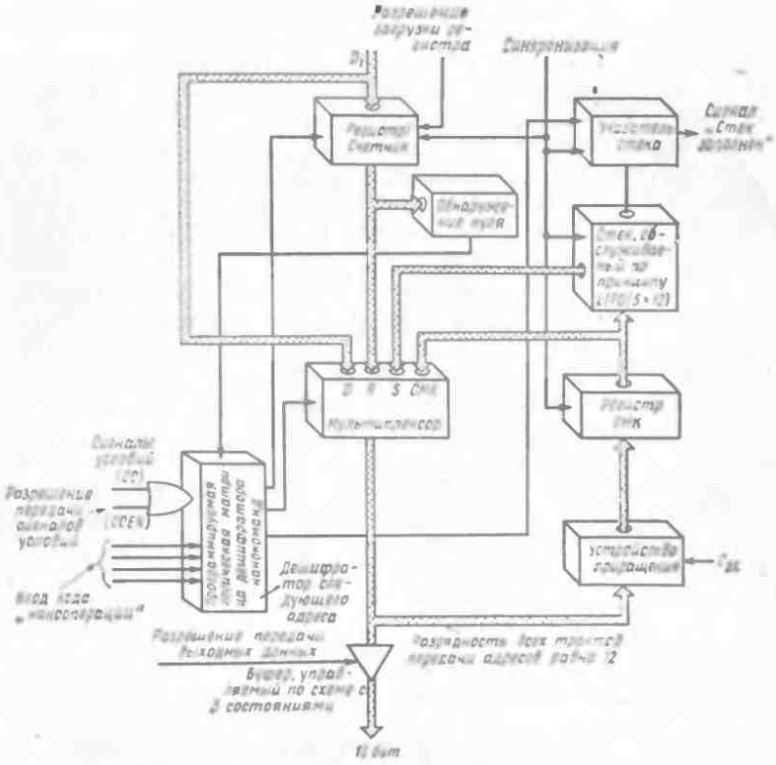


Рис. 5.22. УУВП Am2910 фирмы Advanced Micro Devices.

УУВП Am2911 в корпусе с 20 выводами, а также устройство управления выбором следующего адреса Am29811. Спустя еще два года фирма Advanced Micro Devices выпустила УУВП Am2910 (в корпусе с 40 выводами), представляющее собой эквивалент трех УУВП Am2911, устройства управления выбором следующего адреса Am29811 и 12-разрядного счетчика (рис. 5.22).

Устройство управления выполнением программы Am2910 реализует 16 различных операций, каждая из которых задается 4-битовым кодом (командой УУВП), интерпретируемым программируемой логической матрицей, вырабатывающей внутренние управляющие сигналы УУВП. Работа этой матрицы соответствует второму уровню микропрограммирования, обеспечиваемому применением полностью комбинационных декодирующих схем, и носит «горизонтальный» характер в том смысле, что производится одновременное управление всеми ресурсами. Хотя для названия этого уровня в литературе используются

различные термины, его можно рассматривать как простейшую форму «нанокодирования»[1].

Вход прямого адреса (D_I) является 12-разрядным, на него передается информация из нескольких управляемых по схеме с тремя состояниями источников. Одним из этих источников обычно является поле микрокоманды, задающее константу. Данные с этого входа загружаются в регистр/счетчик при низком уровне сигнала на выводе «Разрешение загрузки регистра». Этот регистр может использоваться в качестве регистра адреса благодаря наличию входа R 12-разрядного четырехвходового мультиплексора или в качестве счетчика на вычитание для эффективной организации циклов в микропрограмме или взаимодействия микрокоманд. В устройстве предусмотрена цепь проверки нулевого значения, сигналы в которой указывают на равенство нулю содержимого счетчика, причем тестирование этих сигналов осуществляется программируемой логической матрицей дешифратора нанокоманд УУВП. Величина N, загруженная в счетчик на вычитание, уменьшается на 1 при выполнении трех различных команд устройства, причем проверка нулевого значения может использоваться для организации выхода из цикла после N+1 выполнения последовательности микрокоманд, заканчивающихся одной из этих команд.

ЭМУЛЯЦИЯ АРХИТЕКТУРЫ ПОСРЕДСТВОМ МИКРОПРОГРАММИРОВАНИЯ

Основным следствием развития техники интегральных схем явилось изменение соотношения стоимости разработки аппаратных средств и программного обеспечения. По мере появления обладающих более широкими возможностями стандартных блоков стоимость разработки программного обеспечения стала увеличиваться по сравнению со стоимостью разработки аппаратных средств. Это обстоятельство обусловливает стремление к использованию большого объема уже существующего программного обеспечения и является основным стимулом для эмуляции путем микропрограммирования существующих машин. Такая эмуляция машины с заданной архитектурой позволяет выполнять ее набор команд и получать идентичные результаты при выполнении операции над одними и теми же данными, хотя временные характеристики процесса обработки данных для двух машин[2] обычно отличаются. Существуют следующие причины для эмуляции машины:

---

[1] Имеется в виду еще более подробное описание выполняемых операций, чем в случае микропрограммирования. — *Прим. перев.*

[2] Имеется в виду эмулируемая машина заданной архитектуры (target machine) и эмулирующая машина, в которой посредством микропрограммирования выполняется эмуляция. — *Прим. перев.*

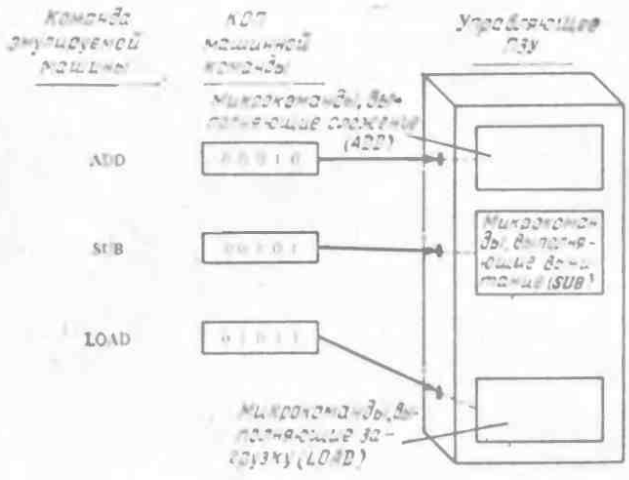


Рис. 5.23. Простейший способ организации выполнения микропрограммы, при котором коды команд исходной машины используются в качестве непосредственных адресов подпрограмм микропрограммы.

1) при эмуляции посредством микропрограммирования может использоваться существующее программное обеспечение;

2) процесс создания машины на базе эмуляции может потребовать значительно меньшего времени, чем при непосредственной аппаратной реализации архитектуры;

3) при эмуляции стоимостные затраты могут оказаться существенно меньшими, чем в случае аппаратной реализации;

4) при эмуляции возможно реализовать в одной машине несколько различных архитектур;

5) расширение исходной (эмулируемой) архитектуры может быть произведено более легким способом в системах с микропрограммным управлением.

Эмуляция существующей машины означает, что ее машинный язык будет реализован надлежащим образом в машине с микропрограммным управлением. По существу машинная команда адресует подпрограмму микропрограммы, которая выполняет требуемую операцию посредством микроопераций. С таким подходом согласуются различные способы организации доступа к микропрограмме. Простейший способ, показанный на рис. 5.23, заключается в использовании битов кода операции машинной команды непосредственно в качестве адреса соответствующей подпрограммы микропрограммы.

Очевидный недостаток такого прямого способа адресации связан с тем, что отсутствует необходимая корреляция между кодами операций машинной команды и адресами подпрограмм

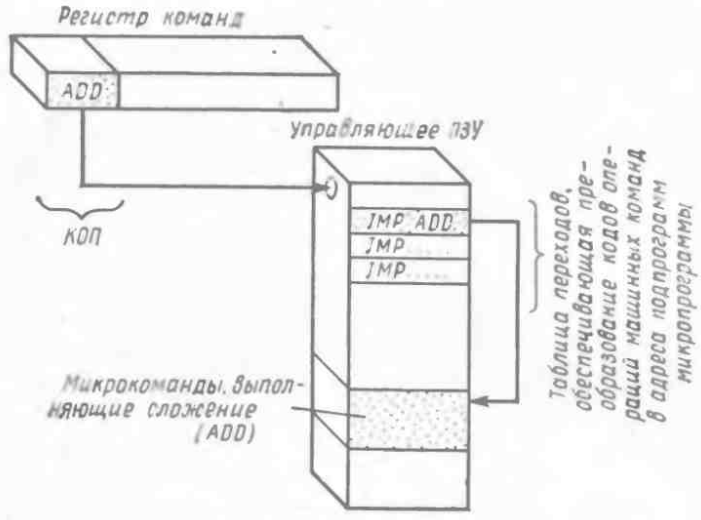


Рис. 5.24. Использование таблицы переходов в управляющей памяти с целью организации доступа к подпрограммам микропрограммы, выполняющим операции, задаваемые машинными командами.

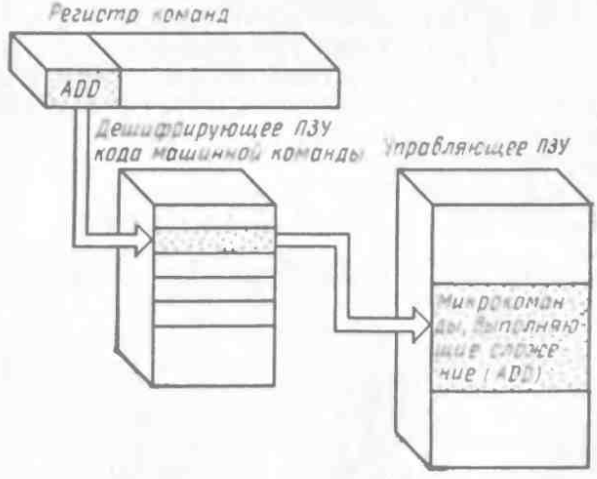


Рис. 5.25. Использование дешифрирующего ПЗУ для организации таблицы переходов, что обеспечивает большое быстродействие при эмуляции архитектуры посредством микропрограммирования.

микропрограммы, реализующими эти операции, а попытка приспособления структуры микропрограммы к таким произвольным кодам машинных команд приводит к выполнению излишних операций перехода в микропрограмме. Подобные переходы вызывают бесполезные затраты времени и усложняют отладку и внесение изменений в микропрограмму. При более рациональном подходе используют косвенную адресацию путем включения в микропрограмму таблицы переходов. В этом случае биты кода операции машинной команды адресуют определенную строку таблицы перехода, содержимое которой в свою очередь используется для организации перехода на выполнение требуемой подпрограммы микропрограммы (рис. 5.24).

Если требуется исключительно высокое быстродействие, то применение в микропрограмме таблицы переходов приводит к излишнему неоправданному снижению скорости обработки. Время цикла работы системы обычно определяется в значительно большей степени задержками в трактах передачи сигналов, чем задержками, обусловленными работой быстродействующего ПЗУ. Таким образом, использование одного полного микроцикла только для перехода на выполнение подпрограммы микропрограммы добавляет микроцикл к выполнению каждой машинной команды. Решение данной проблемы облегчается перемещением таблицы переходов из управляющего ПЗУ в отдельное ПЗУ, обычно называемое *дешифрирующим ПЗУ кода машинной команды*. Это ПЗУ транслирует код операции, поступающий из регистра машинных команд, в адрес подпрограммы микропрограммы. Такая система показана на рис. 5.25. Пример использования дешифрирующего ПЗУ рассматривается в следующем разделе.

ТЕОРИЯ СТРУКТУР УПРАВЛЕНИЯ

Прежде чем перейти к обсуждению 16 команд, интерпретируемых программируемой логической матрицей УУВП, обсудим общие принципы построения структур управления. Поскольку микропрограмма (последовательность управляющих слов) осуществляет управление работой системы, то эти принципы тесно связаны с принципами построения структуры программы. В частности, структуру программы в основном определяют операции безусловного перехода (включая приращение содержимого счетчика микрокоманд) и условного перехода. Хотя принципиально количество возможных структур управления ходом выполнения программы не ограничено, структуру любой про-

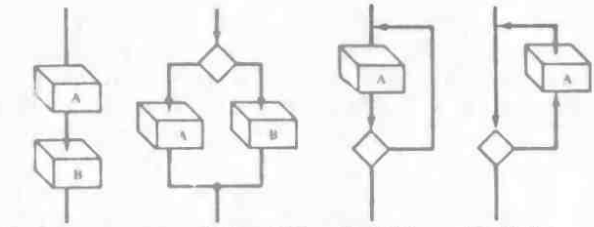
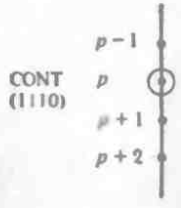


Рис. 5.26. Простейшие структуры управления: *а* — последовательная выборка; *б* — выбор альтернативы; *в* — цикл по условию; *г* — цикл по счетчику.

граммы можно свести к одной из трех простейших управляющих конструкций: последовательная выборка (*а*), ветвление (*б*), цикл (*в, г*) (рис. 5.26).

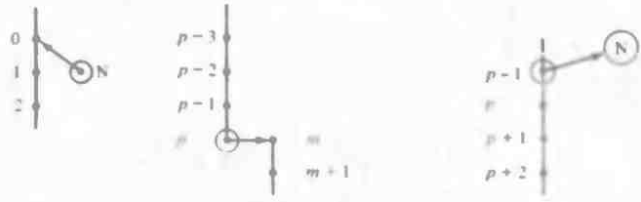
УПРАВЛЯЮЩИЕ СТРУКТУРЫ, РЕАЛИЗУЕМЫЕ УСТРОЙСТВОМ Am2910

Адрес следующей микрокоманды может быть получен с помощью любой из 16 различных команд УУВП, интерпретируемых внутренней программируемой логической матрицей устройства. Эти команды позволяют реализовать простейшие управляющие структуры и их комбинации. Например, команда ПОСЛЕДОВАТЕЛЬНАЯ ВЫБОРКА (CONT) с кодом 1110 указывает на последовательное выполнение микрокоманд микропрограммы.



Черные точки на диаграмме представляют отдельные состояния машины, или работу микропрограммы. Тремя модифика-

циями команды последовательной выборки являются следующие команды безусловного перехода:



ПЕРЕХОД ПО НУЛЕВОМУ АДРЕСУ (JZ=0000)

ПЕРЕХОД ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ ДЕШИФРИРУЮЩЕГО ПЗУ (JMAP=0010)

ЗАГРУЗКА СЧЕТЧИКА И ПОСЛЕДОВАТЕЛЬНАЯ ВЫБОРКА (LDCT=1100)

Команда JZ вызывает установку адреса следующей микрокоманды, равного нулю. Эту команду часто называют командой начальной установки (RESET); она позволяет начать работу машины с выполнения определенной микрокоманды. Команда часто используется во время пуска машины для приведения ее в требуемое состояние при включении напряжения источника питания.

Подобная же операция выполняется по команде JMAP, которая вызывает установку адреса следующей микрокоманды, равным адресу, поступающему на вход D прямого адреса УУВП Am2910, а также обеспечивает выдачу управляющего сигнала с выхода MAP того же устройства. Как упоминалось выше, на вход D могут передаваться данные из различных источников, управляемых по схеме с тремя состояниями, и выходной сигнал MAP обычно используется для разрешения передачи данных из одного из этих источников — дешифрирующего ПЗУ. Эта операция обычно выполняется в том случае, когда микропрограмма с помощью УУВП Am2910 должна интерпретировать определенные машинные команды.

Например, код машинной команды Q должен интерпретироваться подпрограммой микропрограммы, начальный адрес которой равен S. Простейшим способом реализации подобных преобразований является размещение дешифрирующего ПЗУ между источником кодов машинных команд (которым обычно является основная память ЭВМ) и входами прямого адреса Am2910 (рис. 5.27). Дешифрирующее устройство может быть построено на базе программируемого ПЗУ (ППЗУ) или программируемой логической матрицы. Другое дешифрирующее устройство, называемое дешифрирующим ППЗУ векторного адреса, имеет выходы, передача данных с которых разрешается при появлении сигнала на выводе VECT устройства Am2910,

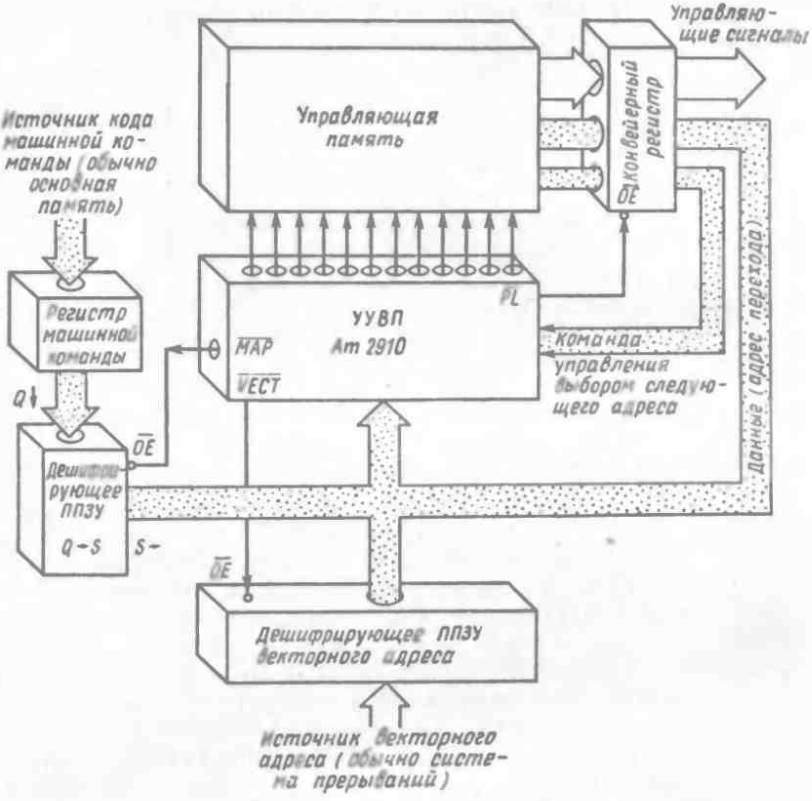


Рис. 5.27. УУВП Am2910 с дешифрирующим ППЗУ.

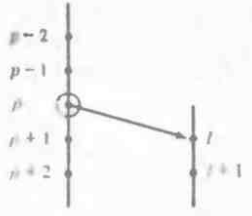
выдаваемого при выполнении соответствующей команды управления выбором следующего адреса. В то время как на вход дешифрирующего ППЗУ кодов машинных команд передаются данные более высокого (по сравнению с микрокомандами) уровня управления, вход дешифрирующего ППЗУ векторного адреса обычно связан с системой прерываний для организации управления процессами в реальном времени.

При выполнении последней команды из группы команд, осуществляющих безусловный переход, внутренний регистр/счетчик адреса загружается данными, поступающими со входной шины устройства Am2910. Эти данные обычно выбираются из поля данных конвейерного регистра и могут использоваться или в качестве адреса перехода, или в качестве величины, загружаемой в счетчик. По команде LDCT в счетчик загружается величина N, и управление передается следующей по порядку

микрокоманде. Содержимое счетчика используется при выполнении ряда команд, которые будут описаны в следующих разделах.

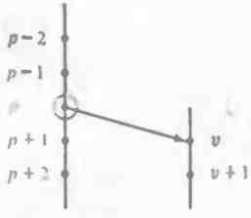
КОМАНДА УСЛОВНОГО ПЕРЕХОДА

Группа команд УУВП Am2910 проверяет сигналы на входных линиях сигналов условий CC с целью выбора источника адреса следующей микрокоманды. Примером такого типа команд условного перехода является команда, выбирающая в качестве следующего адреса содержимое поля адреса перехода конвейерного регистра, если условие «истинно», или адрес следующей по порядку микрокоманды, если условие не выполняется, т. е. «ложно». Дешифрирующая команду программируемая логическая матрица интерпретирует 4-битовый код команды и выбирает либо вход прямого адреса мультиплексора следующего адреса, либо вход мультиплексора, связанный со счетчиком микрокоманд. Если сигнал CC имеет низкий уровень, то выбирается вход прямого адреса и разрешается передача данных с выхода конвейерного регистра за счет выдачи управляющего сигнала на выводе PL (активному состоянию которого соответствует низкий уровень). Выходные управляющие сигналы MAP и VECT при этом не выдаются. Коды команд и обеспечиваемые ими последовательности выполнения микрокоманд имеют следующий вид:



УСЛОВНЫЙ ПЕРЕХОД ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ КОНВЕЙЕРНОГО РЕГИСТРА

(CJP=0011)



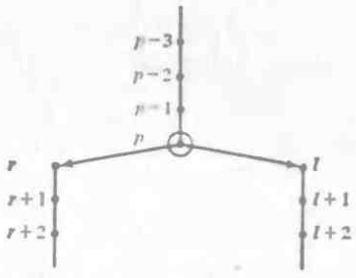
УСЛОВНЫЙ ПЕРЕХОД ПО ВЕКТОРНОМУ АДРЕСУ

(CJV=0110)

Подобная же команда используется для условного перехода по векторному адресу, задаваемому источником векторного адреса. Программируемая логическая матрица по-прежнему выбирает вход прямого адреса или вход, связанный со счетчиком микрокоманд, в зависимости от значения битов условий CC и активирует выход VECT для разрешения передачи данных в УУВП с выхода источника векторного адреса. Выходы PL и

MAP устройства при этом не активны. Диаграмма, поясняющая выполнение команды CJV, имеет такой же вид, как и диаграмма для команды CJP.

Еще одна команда условного перехода JRP позволяет выбирать в качестве источника следующего адреса либо внутренний регистр, либо конвейерный регистр.



УСЛОВНЫЙ ПЕРЕХОД ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ РЕГИСТРА / СЧЕТЧИКА ИЛИ КОНВЕЙЕРНОГО РЕГИСТРА (JPR = 0111)

В этом случае программируемая логическая матрица выбирает вход D или R мультиплексора следующего адреса, активизируя в то же время вывод PL устройства. Действие, производимое этой командой, показано в следующем разделе.

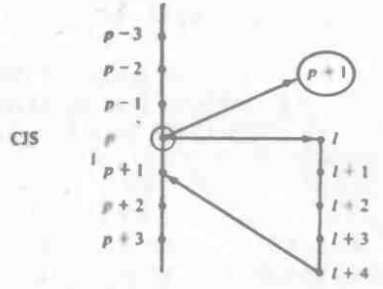
КОМАНДА УСЛОВНОГО ВЫЗОВА ПОДПРОГРАММЫ

В вычислительных машинах объем памяти, требуемой для размещения программы, во многих случаях может быть уменьшен путем замены нескольких повторяющихся последовательностей команд единственной последовательностью таких команд (подпрограммой), завершающейся командой возврата из подпрограммы. Подпрограмма выполняется путем вызова ее из основной программы.

При завершении подпрограммы управление должно быть передано обратно на выполнение следующей (за командой перехода к подпрограмме) команды в вызывающей программе. Поэтому требуются специальные средства для запоминания адреса возврата, указывающего следующую команду в вызывающей программе, подлежащую выполнению. Буфер или стек, обслуживаемый по принципу «поступивший последним обрабатывает-

ся первым», имеет структуру, идеально приспособленную для хранения адресов возврата из подпрограмм. Загрузка в стек адресов возврата из подпрограмм производится в порядке вызова и выполнения вложенных подпрограмм, тогда как извлечение этих адресов из стека осуществляется в обратном порядке. Этот же принцип может использоваться и на уровне микропрограммы, например УУВП Am2910, изображенное на рис. 5.22, включает стек. Глубина стека составляет пять слов, что обеспечивает максимальный уровень вложения подпрограмм, равный 5.

Простейшим типом команды вызова подпрограммы является команда УСЛОВНЫЙ ПЕРЕХОД К ПОДПРОГРАММЕ ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ КОНВЕЙЕРНОГО РЕГИСТРА (CJS). Команда CJS обеспечивает загрузку в стек содержимого счетчика микрокоманд при выполнении проверяемого условия, а также использование в качестве адреса следующей микрокоманды, подлежащей выполнению, содержимого поля данных конвейерного регистра (которые в данном случае являются адресом перехода). Ход выполнения микропрограммы при использовании данной команды имеет следующий вид:



УСЛОВНЫЙ ПЕРЕХОД К ПОДПРОГРАММЕ ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ КОНВЕЙЕРНОГО РЕГИСТРА (CJS = 0001)

Когда в микропрограмме встречается микрокоманда, задающая возврат из подпрограммы (в данном примере это микрокоманда, расположенная по адресу l+4), то мультиплексор выбирает стек в качестве источника следующего адреса, обеспечивая тем самым, что следующей будет выполняться микрокоман-

да, адрес которой был помещен в стек при вызове подпрограммы. По команде возврата из подпрограммы адрес возврата удаляется из стека.

УПРАВЛЕНИЕ ПРЕРЫВАНИЯМИ ПОСРЕДСТВОМ КОМАНДЫ CJS

Возможность быстрой реакции вычислительной системы на события, происходящие во внешней среде, обычно достигается с помощью системы обработки прерываний. Благодаря действию этой системы прекращается выполнение текущей последовательности команд, запоминается адрес следующей команды этой последовательности и осуществляется переход на подпрограмму обработки прерывания. Эта подпрограмма выполняет требуемое обслуживание для внешнего процесса, запросившего прерывание, а затем осуществляет возврат управления в прерванную программу. В одной из схем управления прерываниями, выпускаемой фирмой Advanced Micro Devices, используется дешифратор многоуровневых прерываний Am2914, являющийся аналогом устройства 8214 фирмы Intel (см. Кл., 1980, гл. 12).

Появление запроса на прерывание на одной из входных линий устройства Am2914 приводит к установке низкого уровня сигнала на выходе INT REQ «Запрос прерывания» (рис. 5.28). Вследствие этого отключается вход сигнала переноса счетчика микрокоманд Am2910, что вызывает требуемую задержку работы УУВП на один микроцикл, во время которого сигнал INT REQ запрещает передачу данных с выхода Y УУВП Am2910 и разрешает передачу данных из 12-разрядного буферного устройства, содержащего вектор прерываний. Этот вектор производит выбор микрокоманды, задающей команду CJS УУВП Am2910, по которой осуществляется загрузка адреса следующей микрокоманды прерываемой программы в стек. Другие поля управляющих сигналов микрокоманды, получающей управление в результате прерывания, обеспечивают запрещение передачи данных на вход Am2910 из дешифрирующего ППЗУ кодов машинных команд и разрешение передачи данных на этот вход из дешифрирующего ППЗУ векторного адреса. Векторный адрес поступает в дешифрирующее ППЗУ из устройства управления прерывания типа Am2914 или 8214, что вызывает выполнение соответствующей подпрограммы обслуживания прерываний. Подпрограмма завершается командой CRTN, которая будет описана ниже.

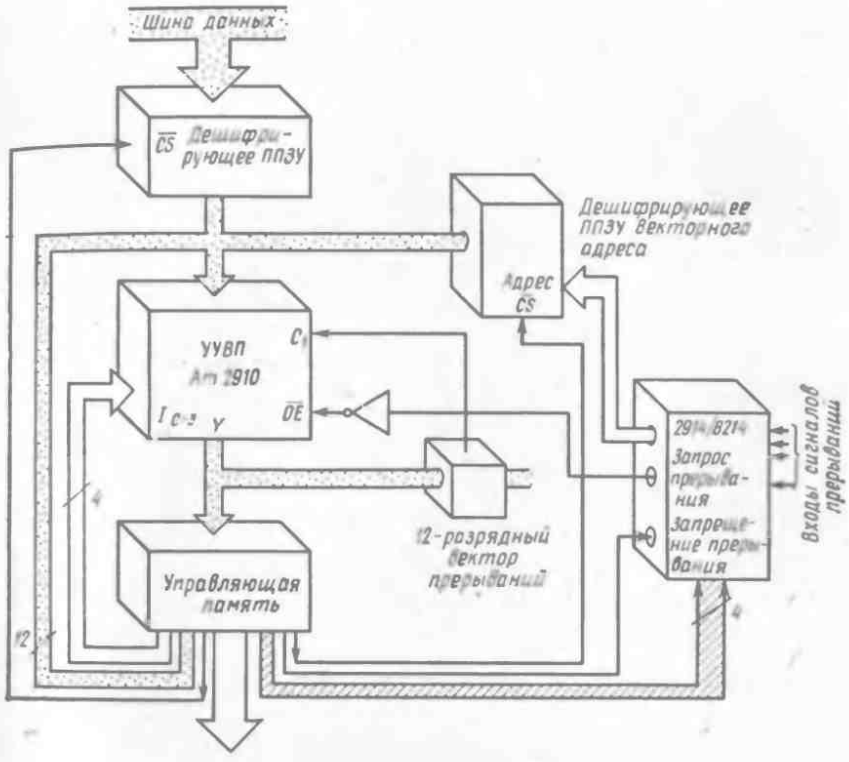
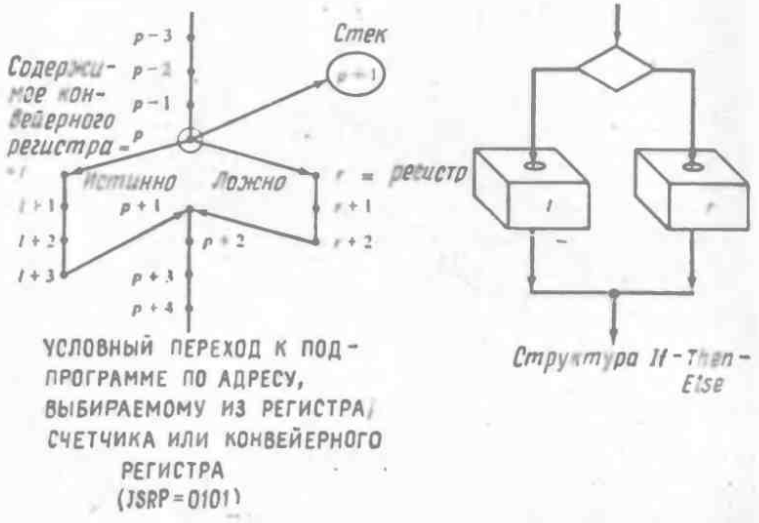


Рис. 5.28. Структура управления прерываниями, использующая устройства Am2910/Intel 8214.

Еще один вариант команды условного вызова подпрограммы позволяет использовать один из двух возможных адресов подпрограммы. При этом адрес микрокоманды, следующей по порядку за микрокомандой, задающей вызов подпрограммы, всегда запоминается в стеке, а адрес вызываемой подпрограммы может быть получен либо из внутреннего регистра/счетчика, либо из конвейерного регистра в зависимости от результата проверки условия. Если условие выполняется, то в качестве источника следующего адреса используется конвейерный регистр, в противном случае — регистр/счетчик. Разумеется, предварительно в регистр/счетчик должен быть загружен требуемый адрес с помощью команды ЗАГРУЗКА СЧЕТЧИКА И ПОСЛЕДОВАТЕЛЬНАЯ ВЫБОРКА (LDCT) или другой подобной команды. Диаграмма, поясняющая ход выполнения

программы, приведена ниже. Как нетрудно заметить, команда JSRP обеспечивает структуру типа IF-THEN-ELSE.



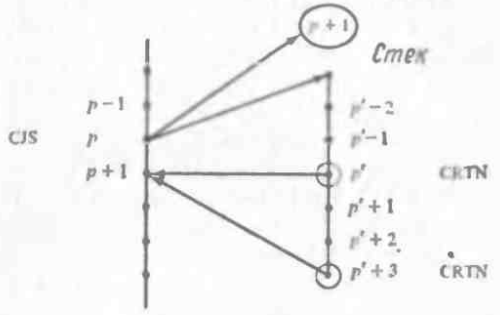
УСЛОВНЫЙ ПЕРЕХОД К ПОДПРОГРАММЕ ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ РЕГИСТРА/СЧЕТЧИКА ИЛИ КОНВЕЙЕРНОГО РЕГИСТРА (JSRP=0101)

КОМАНДА УСЛОВНОГО ВОЗВРАТА ИЗ ПОДПРОГРАММЫ

Использование команды вызова подпрограммы всегда подразумевает, что впоследствии будет выполняться команда возврата из подпрограммы. Эффективность команды возврата может быть повышена, если сделать ее условной; это означает, что операция извлечения адреса из стека адресов возврата из подпрограмм будет выполняться только тогда, когда задано условие; в противном случае управление передается следующей по порядку микрокоманде. Заметим, что команда условного возврата из подпрограммы может выполняться как безусловная путем принудительного обеспечения положительного результата проверки условия. В УУВП Am2910 это легко достигается за счет подачи высокого уровня сигнала на вход CCEN. Для разрешения проверки условий используется специальная линия CCEN; при высоком уровне сигнала на этой линии проверка условия запрещена[1]. Пример управления последовательностью

[1] Условие считается заданным независимо от значения сигналов условий CC. — *Прим. перев.*

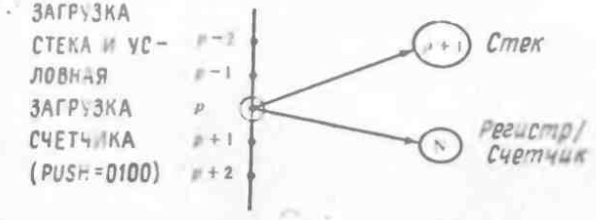
выполнения микрокоманд приводится ниже



УСЛОВНЫЙ ВОЗВРАТ ИЗ ПОДПРОГРАММЫ (CRTN=1010)

КОМАНДА РАБОТЫ СО СЧЕТЧИКОМ И СТЕКОМ

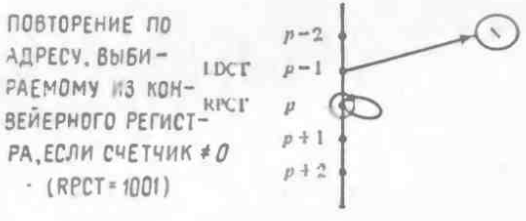
Мы уже обсуждали команду ЗАГРУЗКА СЧЕТЧИКА И ПОСЛЕДОВАТЕЛЬНАЯ ВЫБОРКА (LDCT). В некоторой степени подобная ей команда позволяет загружать в стек адрес следующей микрокоманды и переходить к ее выполнению, а также выполнять в зависимости от результатов проверки условия загрузку в счетчик содержимого поля данных конвейерного регистра. Загрузка счетчика производится, если результат проверки принимает значение «истинно». Условие может задаваться и принудительно, так что загрузка счетчика будет выполняться безусловно. Счетчик играет важную роль при организации циклов в микропрограмме (см. следующий раздел). Диаграмма, поясняющая ход выполнения микропрограммы, при использовании данной команды имеет следующий вид:



ОРГАНИЗАЦИЯ ЦИКЛОВ С ИСПОЛЬЗОВАНИЕМ СЧЕТЧИКА

В предыдущих разделах обсуждались две команды, осуществляющие загрузку внутреннего регистра/счетчика УУВП Am2910 и последовательную выборку следующей по порядку

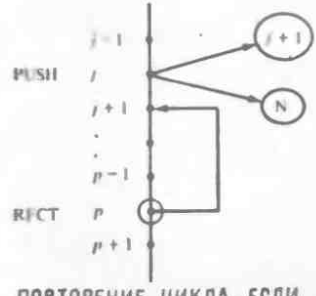
микрокоманды: ЗАГРУЗКА СЧЕТЧИКА И ПОСЛЕДОВАТЕЛЬНАЯ ВЫБОРКА (LDCT) и ЗАГРУЗКА СТЕКА И УСЛОВНАЯ ЗАГРУЗКА СЧЕТЧИКА (PUSH). Рассмотрим вначале способ, позволяющий использовать счетчик для повторения выполнения определенной команды заданное число раз. Команда LDCT загружает это число в счетчик, после чего производится последовательная выборка следующей микрокоманды. Когда встречается команда ПОВТОРЕНИЕ ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ КОНВЕЙЕРНОГО РЕГИСТРА, ЕСЛИ СЧЕТЧИК $\neq 0$ (RPCT), то проверяется содержимое счетчика. Если оно не равно нулю, производится его уменьшение на 1 (отрицательное приращение). Если счетчик не содержит нулевого значения, то вход D мультиплексора следующего адреса и конвейерный регистр используются для формирования адреса перехода. Обычно этот адрес указывает на текущую микрокоманду или на какую-либо из предшествующих микрокоманд, обеспечивая тем самым выполнение в цикле одних и тех же операций до тех пор, пока содержимое счетчика не станет равным нулю. Приводимая здесь диаграмма соответствует случаю, когда конвейерный регистр содержит адрес текущей команды. Когда содержимое счетчика станет равным нулю, будет выполняться следующая по порядку микрокоманда:



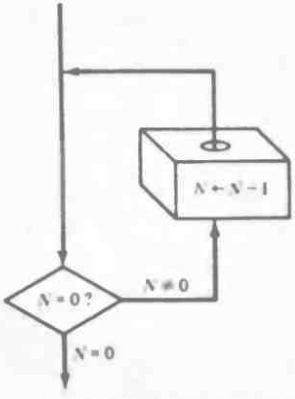
ОРГАНИЗАЦИЯ ЦИКЛА
ПО СЧЕТЧИКУ В УУВП Am2910

В модификации описанной выше команды адрес, загруженный в стек по команде PUSH, используется для организации выполнения последовательности микрокоманд в цикле заданное число раз, причем параметр цикла (число повторений) загружается в счетчик той же командой PUSH. Команда ПОВТОРЕНИЕ ЦИКЛА, ЕСЛИ СЧЕТЧИК $\neq 0$ (RFCT) проверяет сигнал условия на входе CC, и если содержимое счетчика не равно нулю, то оно уменьшается на 1, а данные, находящиеся на вершине стека, используются в качестве следующего адреса. Этим адресом является адрес микрокоманды, следующей за

микрокомандой, задающей команду PUSH. Следовательно, управление возвращается в эту точку, после чего продолжается последовательное выполнение микрокоманд. Заметим, что данные из вершины стека не удаляются, т. е. на вершине стека остается тот же самый адрес. Одна и та же последовательность микрокоманд выполняется до тех пор, пока содержимое счетчика не станет равным нулю, после чего производится извлечение адреса из вершины стека (этот адрес не используется) и управление передается микрокоманде, следующей за микрокомандой, задающей команду RFCT. Приведенная внизу слева диаграмма имеет структуру, соответствующую циклу по счетчику (Do-While), изображение которого в виде блок-схемы приведено справа.



ПОВТОРЕНИЕ ЦИКЛА, ЕСЛИ СЧЕТЧИК ≠ 0 (RFCT=1000)

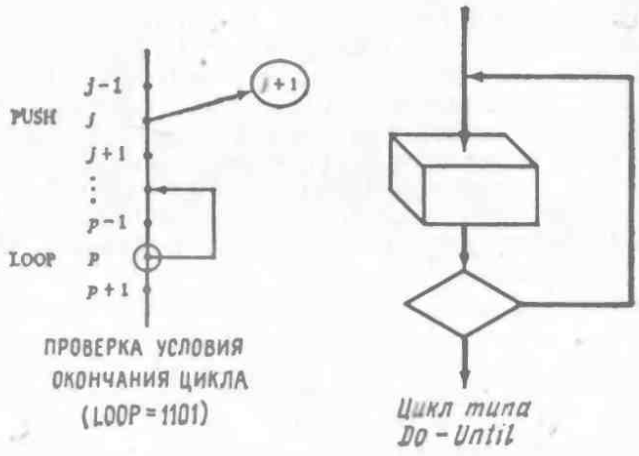


Цикл по счетчику (Do-While)

КОМАНДА ОРГАНИЗАЦИИ ЦИКЛА ТИПА DO-UNTIL

В предыдущих примерах выход из цикла осуществляется после выполнения тела цикла заданное число раз. Часто бывает желательно осуществить выход из цикла, когда происходят какие-то внешние события и(или) события, зависящие от данных. Такая возможность предоставляется командой ПРОВЕРКА УСЛОВИЯ ОКОНЧАНИЯ ЦИКЛА (LOOP), которая обеспечивает *выполнение* (Do) тела цикла до тех пор, *пока* (Until) соблюдается условие, после чего начинает выполняться микрокоманда, следующая по порядку за микрокомандой LOOP.

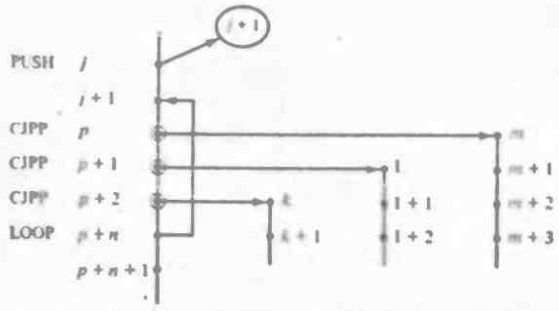
Структура управления ходом выполнения микропрограммы выглядит в этом случае следующим образом:



ОРГАНИЗАЦИЯ УСЛОВНЫХ ВЫХОДОВ ИЗ ЦИКЛА

Команда УСЛОВНЫЙ ВОЗВРАТ ИЗ ПОДПРОГРАММЫ (CRTN) позволяет осуществлять выход из подпрограммы или цикла в различных точках в зависимости от выполнения условия. После микрокоманды, задающей выход из подпрограммы или цикла, следующей будет всегда выполняться микрокоманда, адрес которой находится в стеке. Однако часто желательно осуществлять выход из цикла по разным адресам, и такая возможность предоставляется с помощью команды УСЛОВНЫЙ ПЕРЕХОД ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ КОНВЕЙЕРНОГО РЕГИСТРА, И ИЗВЛЕЧЕНИЕ ДАННЫХ ИЗ СТЕКА (CJPP). В этом случае каждая микрокоманда, задающая команду проверки выхода из цикла, содержит (в конвейерном регистре) адрес перехода. Дешифрирующая команды УУВП программируемая логическая матрица выбирает либо вход мультиплексора следующего адреса, соединенный со счетчиком микрокоманд (если условие не соблюдается), либо вход D мультиплексора (если условие соблюдается). Если условие соблюдается, разрешается передача адреса из конвейерного регистра, выбирается вход D мультиплексора следующего адреса и в стеке производится операция извлечения с целью удаления из него точки входа в цикл. Показанная ниже структура управления, обеспечивающая переходы по нескольким адресам,

представляет собой весьма эффективное средство для построения программ, которые будут рассматриваться ниже.

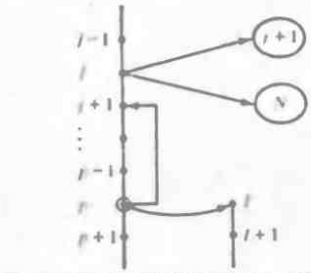


УСЛОВНЫЙ ПЕРЕХОД ПО АДРЕСУ, ВЫБИРАЕМОМУ ИЗ КОНВЕЙЕРНОГО РЕГИСТРА, И ИЗВЛЕЧЕНИЕ ДАННЫХ ИЗ СТЕКА (CJPP=1011)

КОМАНДА ПЕРЕХОДА ПО ОДНОМУ ИЗ ТРЕХ АДРЕСОВ

Последняя рассматриваемая команда позволяет во время выполнения одной микрокоманды проверить как сигналы условий, зависящие от обрабатываемых данных, так и содержимое счетчика и осуществить переход по одному из трех адресов. Перед выполнением команды ПЕРЕХОД ПО ОДНОМУ ИЗ ТРЕХ АДРЕСОВ (TWB) другая команда УУВП загружает счетчик и помещает в стек следующий адрес, устанавливая тем самым точку входа в цикл и задавая число повторений тела цикла. Команда TWB производит уменьшение на 1 содержимого счетчика и выбор входа мультиплексора следующего адреса, соединенного со стеком, для повторного выполнения тела цикла до тех пор, пока не выполняется условие и содержимое счетчика не стало равным нулю. Если содержимое счетчика становится равным нулю до того, как появится сигнал условия, то в стеке выполняется операция извлечения, в качестве источника следующего адреса выбирается счетчик микрокоманд и управление передается микрокоманде, следующей за микрокомандой, задающей команду TWB. Если при выполнении команды условие соблюдается, то осуществляется операция извлечения из стека, и следующий адрес выбирается из конвейерного регистра. Такая структура управления, приведенная ниже, оказывается исключительно эффективной для организации вы-

хода из цикла при реализации алгоритмов поиска определенных данных (просмотра таблиц).



РАССМОТРЕНИЕ ВРЕМЕННЫХ ХАРАКТЕРИСТИК: КОНВЕЙЕРНАЯ ОБРАБОТКА ДАННЫХ

Изучение вопросов микропрограммирования касалось в основном архитектуры системы (т. е. линий управления и трактов передачи данных). При обсуждении эмуляции посредством микропрограммирования определенной машины было выяснено, что «таблица переходов» может быть реализована либо в ПЗУ микропрограмм (т. е. в управляющей памяти), либо с помощью отдельного дешифрирующего ПЗУ. Хотя с точки зрения архитектуры оба варианта являются эквивалентными, система с отдельным ПЗУ, дешифрирующим коды машинных команд, обеспечивает более высокое быстродействие. Следует отметить, что разработчик системы с микропрограммным управлением должен всегда анализировать временные задержки, имеющие место в конкретной реализации системы. Быстродействие может быть иногда существенно повышено путем добавления в надлежащее место структуры системы определенного элемента. Однако обычно лишние элементы добавляют задержку передачи сигналов по трактам системы и снижают ее быстродействие.

Один из распространенных подходов к проектированию систем с высоким быстродействием предполагает использование метода, называемого *конвейерной обработкой*. Практически во всех случаях этот метод состоит в предварительной выборке информации, которая потребуется позднее, и ее сохранении в конвейерном регистре до тех пор, пока она не понадобится. В некоторых случаях используется несколько уровней конвейерной обработки. Хотя конвейерная обработка усложняет

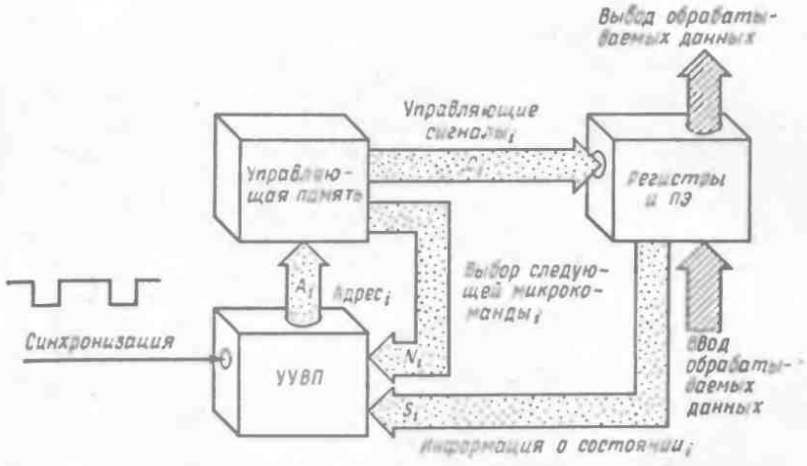
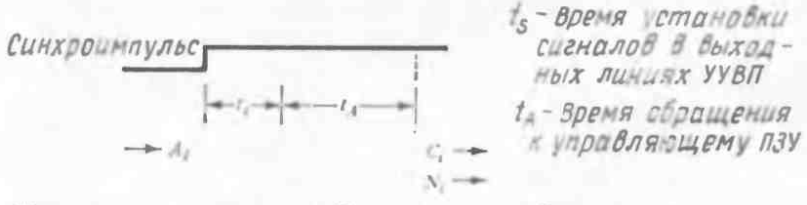


Рис. 5.29. Четыре основных канала передачи управляющих сигналов, относящиеся к i-микроциклу.

структуру процессора, ценою этого усложнения является увеличение его производительности. Начнем анализ временны́х характеристик с рассмотрения простейшей структуры. Такая структура (рис. 5.29) включает в себя управляющее ПЗУ, УУВП и ПЭ, выполняющий операции по обработке данных.

Если полагать, что адрес микропрограммы A_i фиксируется в выходном регистре УУВП при поступлении положительного фронта синхроимпульсов, то задержка передачи сигналов от момента выдачи сигналов A_i до поступления сигналов C_i и N_i состоит из времени, необходимого для установки сигналов в выходных линиях (5—20 нс), к которому добавляется время обращения к управляющему ПЗУ (20—100 нс).



Управляющие сигналы C_i начинают действовать непосредственно и вызывают выполнение требуемых операций над данными в ПЭ, или подсистеме обработки данных. Сигналы управления выбором следующей микрокоманды передаются обратно в УУВП; однако во многих случаях для полного определения адреса следующей микрокоманды A_i необходима информация о результате выполнения текущей микрооперации (так называе-

мая *информация о состоянии*). Таким образом, в то время как сигналы C_i начинают использоваться сразу же после их выдачи, действие сигналов N_i задерживается до поступления информации о состоянии S_i. Это вызывает другую задержку t_p, связанную с работой ПЭ. Подобные задержки обычно составляют от 20 до 120 нс. При наличии информации о состоянии в УУВП уровень синхроимпульса может измениться с высокого на низкий; при этом производится фиксация сигналов S_i в УУВП и инициирование процесса определения адреса следующей микрокоманды.

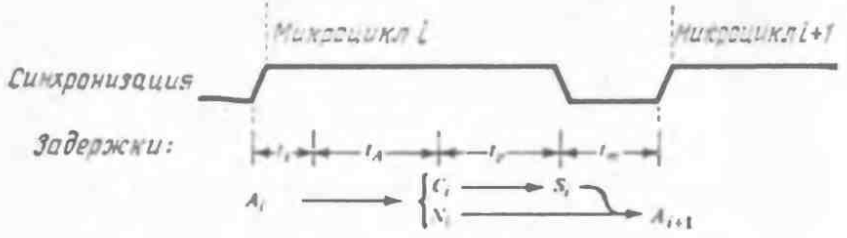


Рис. 5.30. Определение длительности микроцикла в простейшей системе.

Из приведенного выше материала следует, что при включении каких-либо дополнительных элементов (таких, как мультиплексоры) в тракт передачи информации о состоянии по каналу обратной связи (или в любой другой тракт) необходимо учитывать задержки распространения сигналов в этих элементах для определения минимально возможной длительности микроцикла (рис. 5.30). Очевидно также, что для уменьшения времени определения адреса следующей микрокоманды A_{i+1} при безусловной передаче управления (т. е. передаче управления, не зависящей от сигналов состояния S_i, характеризующих результат выполнения операции, задаваемой управляющими сигналами C_i) может использоваться схема синхронизации с переменным периодом следования синхроимпульсов. Вместо того чтобы рассматривать подобные усложнения системы, будем далее развивать простейшие концепции конвейерной обработки для повышения быстродействия системы.

Заметим, что суть обсуждаемой проблемы состоит в том, что выборка следующей микрокоманды должна задерживаться до тех пор, пока не будет завершено выполнение текущей микрокоманды. Таким образом, две самые большие по длительности задержки передачи сигналов, связанные с доступом к управляющему ПЗУ и обработкой данных, складываются. Подход, основанный на конвейерной обработке, заключается в совмеще-

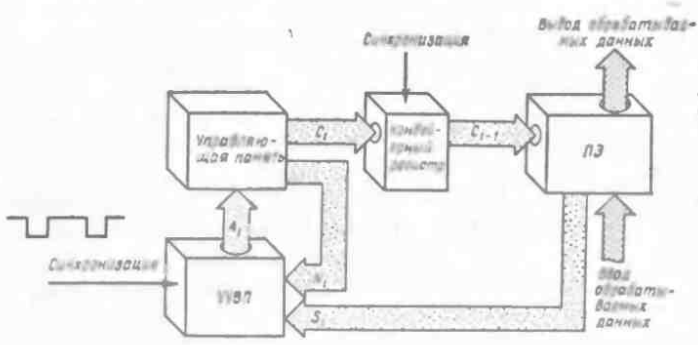
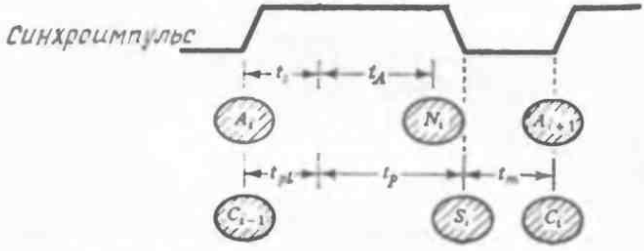


Рис. 5.31. Один из возможных вариантов размещения конвейерного регистра в структуре процессора.

нии выборки следующей микрокоманды с выполнением текущей. Одним из способов обеспечения такого совмещения является запись текущего управляющего слова (сигналов C_i) в отдельном фиксаторе, или буфере (рис. 5.31).

Анализ такой системы показывает существенные изменения ее временны́х характеристик. Основной эффект от применения конвейерного регистра (входы которого запираются при поступлении положительного фронта синхроимпульса) заключается в организации параллельной обработки данных и выборки микрокоманд.



t_s — Время установки сигналов в УУВП; t_{pl} — время установки сигналов в конвейерном регистре; t_A — задержка, вызванная обращением к управляющему ПЗУ; t_p — задержка, вызванная операцией обработки данных; t_m — задержка, вызванная работой УУВП.

Поскольку задержки, обусловленные обращением к управляющему ПЗУ и обработкой данных, являются основными, результирующая задержка передачи сигналов уменьшается, и быстродействие системы может быть увеличено примерно вдвое. Заметим, что затраты, связанные с включением в структуру процессора конвейерного регистра, являются наименьшими из общих затрат, потребовавшихся для удвоения быстродействия системы. С точки зрения проектировщика, более существенные

затраты связаны с усложнением структуры, возникающим при введении «фазового сдвига» в процессе обработки данных, т. е. сдвига на один цикл вычислительного процесса в тех местах микропрограммы, в которых производится тестирование признаков состояния. При отсутствии конвейерной обработки формирование информации о результатах выполнения операции и ее тестирование с целью определения адреса следующей микрокоманды производились в одном и том же цикле. В системе с конвейерной обработкой информация о состоянии, тестируемая в текущем цикле i, формируется при выполнении управляющего слова предыдущего цикла C_{i-1} и проверяется в текущем цикле с целью определения адреса микрокоманды, подлежащей выполнению в следующем цикле A_{i+1}. Разработчик микропрограммы должен учитывать данное обстоятельство и обеспечить компенсацию этого фазового сдвига. В связи с существенным увеличением эффективности, достигаемым за счет совмещения выборки следующей микрокоманды с выполнением текущей, в большинстве систем с микропрограммным управлением используется в настоящее время конвейерная обработка. Ряд примеров архитектуры с конвейерной обработкой на базе семейства микропроцессоров с разрядно-модульной организацией Am2900 приведен в следующем разделе.

В обобщенном виде каждый микроцикл работы машины имеет следующую структуру:

Выполнение цикла может включать операции «выборки данных». Часть цикла, связанная с выборкой микрокоманды, включает время $t_в$, необходимое для установки и стабилизации сигналов на выходе РА микропрограммы, и время доступа к управляющему ПЗУ (t_A) (рис. 5.32). Определение минимального временно́го цикла работы системы всегда базируется на анализе критических трактов передачи данных, вносящих самые большие временны́е задержки (рис. 5.32—5.35).

ДРУГИЕ ПОДХОДЫ К ПОСТРОЕНИЮ АРХИТЕКТУРЫ, ОБЕСПЕЧИВАЮЩЕЙ ВЫСОКОЕ БЫСТРОДЕЙСТВИЕ

Разработчики пытаются добиться максимального быстродействия вычислительных устройств и систем многими путями. При этом систему обычно стараются построить так, чтобы наи-

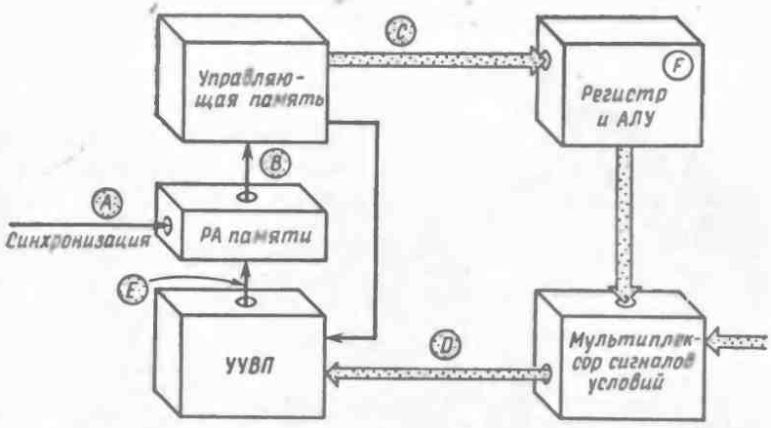


Рис. 5.32. Критические (с максимальными временны́ми задержками) тракты передачи данных процессора.

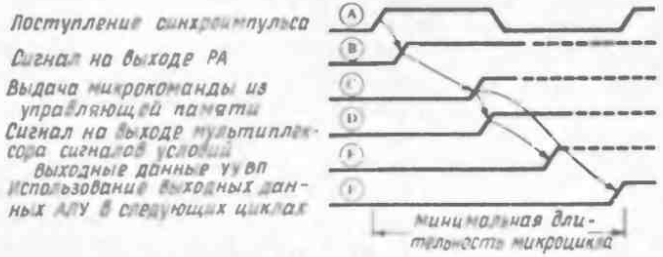


Рис. 5.33. Временна́я диаграмма работы процессора, изображенного на рис. 5.32. (Минимально возможная длительность микроцикла определяется передачей данных по тракту АЛУ.)

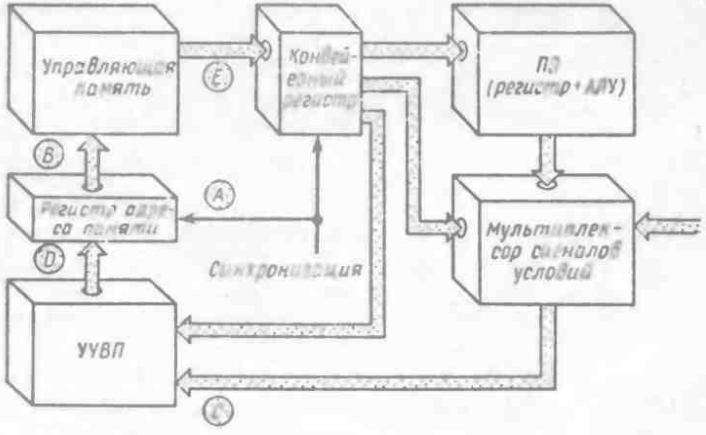


Рис. 5.34. Структура процессора с конвейерной обработкой данных.

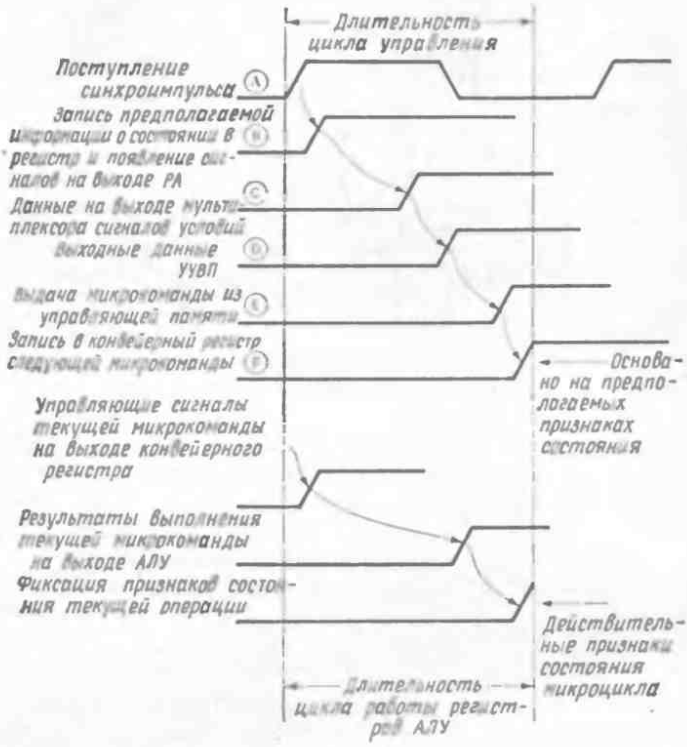


Рис. 5.35. Временна́я диаграмма работы процессора, изображенного на рис. 5.34.

Если предполагаемая информация о состоянии совпадает с информацией о состоянии, полученной в результате выполнения текущей операции в АЛУ, то может сразу выполняться следующая микрооперация; в противном случае должна быть произведена выборка новой микрокоманды.

более часто встречающиеся операции выполнялись более быстро (допуская более медленное выполнение редко применяемых операций) и реализовывались на менее дорогих элементах. Типичные способы обеспечения высокого быстродействия включают в себя:

1. Многоуровневую конвейерную обработку. Если для большей части операций используется последовательная выборка микрокоманд из относительно медленнодействующей памяти, то могут быть предварительно выбраны и помещены в конвейерные регистры одновременно несколько микрокоманд.

2. Память с расслоением[1]. Вариант многоуровневой конвейерной обработки, при котором производится поочередное обращение к блокам медленнодействующей памяти, в которых ячейки памяти сгруппированы таким образом, что каждая последовательная выборка приводит к обращению к следующему независимому блоку памяти. Таким образом, операции последовательной выборки могут выполняться параллельно, так что требуемое управляющее слово в большинстве случаев будет получено из памяти до того, как потребуется его выполнение.

3. Кэш-память. В этом случае в качестве рабочей памяти используется дорогостоящая быстродействующая память, а медленнодействующая память предназначается для хранения логических блоков памяти, которые могут понадобиться в дальнейшем. При этом для проверки каждой порции информации, извлекаемой из кэш-памяти, или инициирования процесса передачи соответствующего логического блока в кэш-память применяется ассоциативная память.

4. Систему с переменной длительностью цикла. Поскольку составитель микропрограммы часто знает наиболее вероятный результат выполнения данной операции, он может предсказать, какой адрес следующей микрокоманды будет сформирован в результате проверки сигналов условий, выдаваемых по окончании текущей операции. Поэтому микрокоманда, находящаяся по этому адресу, может быть выбрана в то время, пока вычисляется результат текущей операции. Если полученный результат совпадает с предсказанным, то следующая микрокоманда уже готова к выполнению. Если нет, то длительность цикла увеличивается и выбирается нужная микрокоманда.

Основной недостаток большинства подобных схем с повышенным быстродействием заключается в увеличении сложности системы. При использовании памяти с расслоением используется отдельное устройство управления для каждого блока памяти. Многоуровневая конвейерная обработка усложняет процесс составления микропрограммы, поскольку приводит к переменной длительности цикла. Системы со свопингом («перекачкой») памяти, использующие кэш-память, обычно основываются на модели «наиболее вероятных вычислений, подлежащих выполнению в системе». Эффективность всех упомянутых подходов в существенной степени зависит от количества выполняемых операций разного типа, поскольку большое число команд перехода препятствует использованию любых средств предварительной выборки следующей микрокоманды.

Общая тенденция развития полупроводниковой технологии, заключающаяся в стремлении обеспечить высокое быстродей-

---

[1] Память, организованная из нескольких независимых (логических) блоков, допускающих одновременное к ним обращение. — *Прим. перев.*

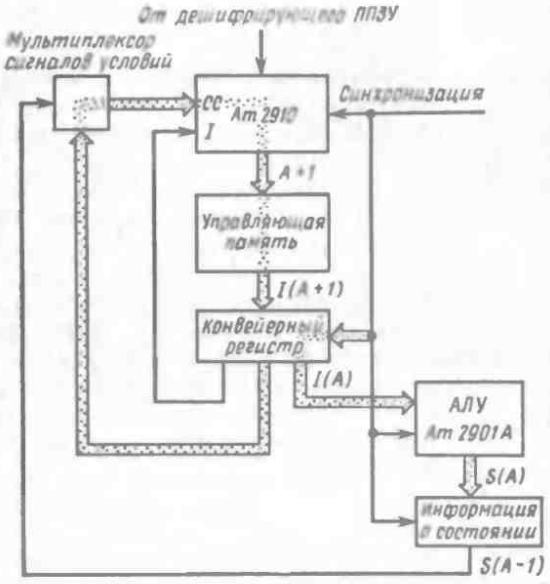


Рис. 5.36. Архитектура семейства микропроцессоров Am2900[1].

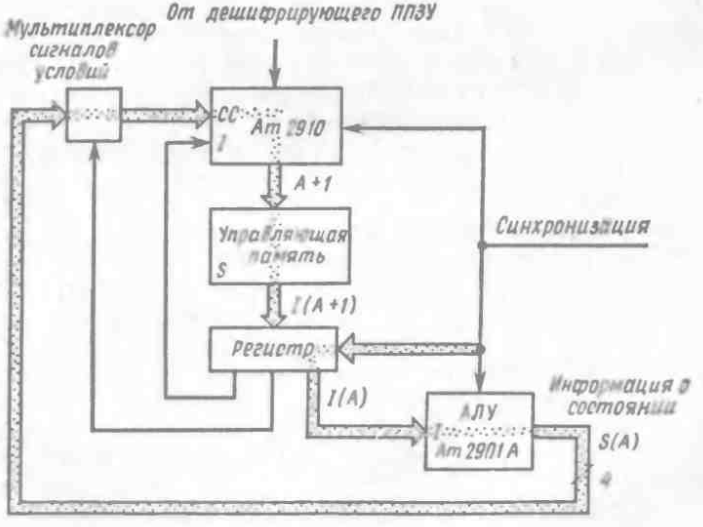


Рис. 5.37. Структура процессора с конвейерной обработкой с записью в регистр микрокоманды.

Регистр А, размещенный на выходе управляющей памяти, содержит выполняемую микрокоманду. Временные задержки, обусловленные управляющей памятью и АЛУ Am2901, складываются. Условные переходы выполняются в том же микроцикле, что и операция АЛУ, вырабатывающая соответствующие сигналы условий.

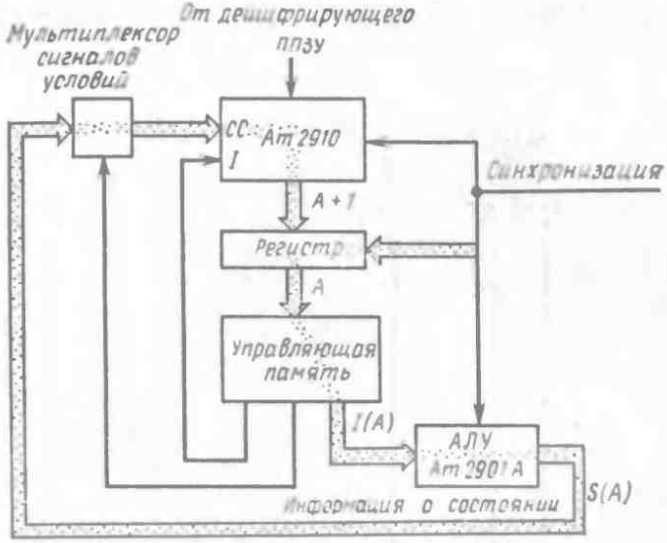


Рис. 5.38. Структура процессора с конвейерной обработкой с записью в регистр адреса микрокоманды.

Для тракта передачи данных, соответствующего максимальным временным задержкам, задержки, обусловленные управляющей памятью и АЛУ Am2901A, складываются. Данная архитектура обеспечивает то же быстродействие, что и система, основанная на предварительной выборке микрокоманды, но требует регистра для хранения только нескольких битов данных, поскольку запоминается адрес микрокоманды (10—12 бит), а не сама микрокоманда (40—60 бит).

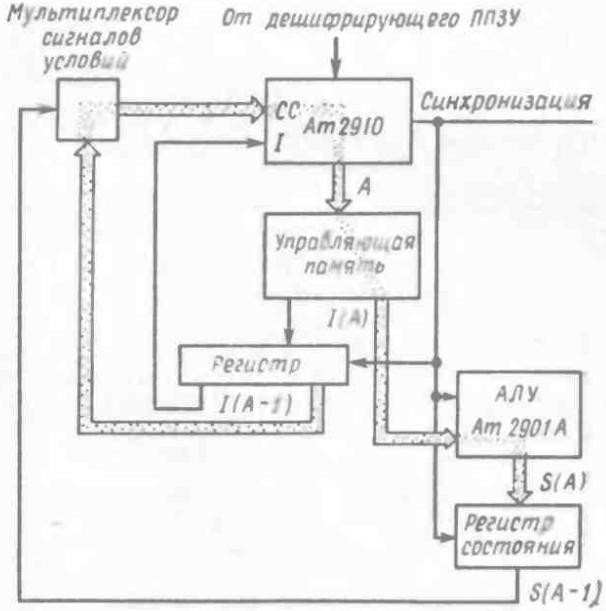


Рис. 5.39. Структура процессора с конвейерной обработкой с записью в регистр информации о результатах выполнения операции.

Для тракта передачи данных, соответствующего максимальным временным задержкам, задержки, обусловленные управляющей памятью и АЛУ Am2901A, складываются.

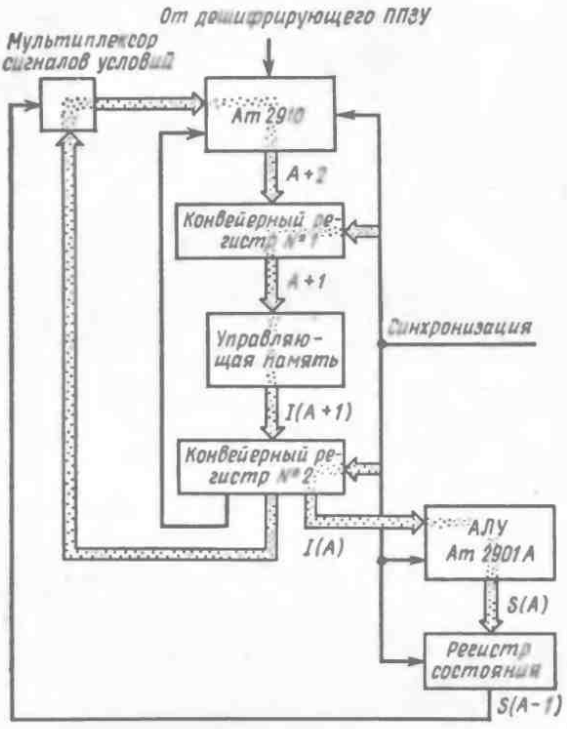


Рис. 5.40. Структура процессора с конвейерной обработкой с записью в регистры микрокоманды и информации о результатах выполнения операции.

Составление микропрограммы в этом случае связано с наибольшими трудностями, поскольку выбор следующей микрокоманды осуществляется во время выполнения двух предшествующих ей микрокоманд.

ствие при минимальных затратах, определяет выбор упрощенных подходов во всех случаях, за исключением применений, предъявляющих наиболее жесткие требования.

ТИПЫ АРХИТЕКТУРЫ ПРОЦЕССОРОВ, ИСПОЛЬЗУЮЩИХ УУВП Am2910

На рис. 5.36—5.40 изображены типы архитектуры процессоров, построенных на базе УУВП Am2910 фирмы Advanced Micro Devices (AMD). Критические тракты передачи данных, вносящие максимальные временны́е задержки, затемнены для

облегчения проведения анализа временных характеристик работы системы. Приведенные схемы заимствованы из документации фирмы AMD и приводятся с ее разрешения. Одноуровневая конвейерная обработка обеспечивает максимальное быстродействие, поскольку тракты управляющей памяти и логической матрицы ПЭ Am2901A используются параллельно, а не последовательно. Соответствующая архитектура процессоров серии Am2900, рекомендуемая фирмой AMD, показана на рис. 5.36.

ЗАКЛЮЧЕНИЕ

Рассмотренный в гл. 4 аппарат логических матричных схем использован для построения УУВП общего назначения, образующих основу всех систем с микропрограммным управлением. В первых УУВП на ТТЛ-схемах 3001 фирмы Intel требовалось указание следующего адреса для каждой выполняемой микрокоманды, тогда как в широко распространенных УУВП семейства Am2900 используется счетчик микрокоманд для определения адреса следующей микрокоманды, подлежащей выполнению. Системы с микропрограммным управлением, построенные на рассмотренных в данной главе и большинстве других семейств УУВП, имеют концептуально идентичную архитектуру.

Хотя УУВП использованы для построения ЭВМ общего назначения, появление 16- и 32-разрядных микропроцессоров, обладающих большими вычислительными возможностями, делают указанное применение УУВП все менее заслуживающим внимание. Такие микропроцессоры обладают высоким быстродействием, мощным набором команд, реализуются лишь на нескольких кристаллах и, что самое важное, располагают развитым программным обеспечением.

Таким образом, на базе БИС и СБИС могут быть реализованы ЭВМ стандартной архитектуры, для которых имеется развитое программное обеспечение, причем найдется очень немного доводов в пользу реализации ЭВМ общего назначения нестандартной архитектуры, использующих рассмотренные в данной главе УУВП. Это связано с тем, что отсутствие пригодного к использованию программного обеспечения превращает такие ЭВМ в практически никому не нужные системы.

Важным исключением из этого правила является реализация ЭВМ, ориентированных на язык программирования высокого уровня, в которых используется разрядно-модульная орга-

низация с микропрограммным управлением. Однако эта тема выходит за рамки данной книги. Для ее изучения читателю рекомендуется обратиться к монографии Чу[1].

Проектированием систем с разрядно-модульной организацией и микропрограммным управлением целесообразно заниматься при создании архитектуры специального назначения. В первую очередь это относится к системам, в которых требуется высокая скорость специальной обработки данных, превышающая возможности микропроцессоров на БИС. В последующих главах будут рассмотрены примеры построения систем, в которых используются устройства с разрядно-модульной организацией для реализации архитектуры специального назначения.

[1] High-Level Language Computer Architectures, ed. by Y. Chu, Academic Press, New York, 1975.

ДВУМЕРНАЯ ЦИФРОВАЯ ФИЛЬТРАЦИЯ

Появление быстродействующих цифровых электронных устройств явилось одной из важнейших предпосылок широкого распространения цифровой обработки двумерных изображений. Основные области применения цифровых фильтров включают обработку изображений, полученных с искусственных спутников Земли, усиление рентгеновских лучей, обработку излучения радаров и сонаров. Внедрение цифровых фильтров вызвало резкий скачок в медицинской диагностике. Эти устройства получили широкое использование как в гражданской, так и в военной авиации. В настоящее время цифровые фильтры начинают применяться для создания «органов зрения» в робототехнике. Можно ожидать и дальнейшего расширения областей их применения.

В данной главе изложен порядок проектирования двумерного фильтра, предназначенного для слежения за изображением. Возможны самые разнообразные применения этого устройства — от головок самонаведения в противоракетных системах до приборов, позволяющих избежать столкновения летательных аппаратов. Фильтр строится в предположении, что изображение воспринимается каким-либо другим устройством и требуется проследить за движением объекта в двумерном пространстве. Проектируемая система пригодна и для выполнения более общих корреляционных функций, поэтому она создается максимально универсальной и с учетом ограничений, определяемых ее назначением.

ЭТАПЫ ПРОЕКТИРОВАНИЯ

Поскольку по математической теории корреляции и цифровой фильтрации существует обширная литература, предпримем попытку использовать математический анализ для разработки оптимального проекта, если это вообще, конеч-

но, возможно. Общий подход заключается в формулировании задачи, а затем в ее анализе с тем, чтобы определить алгоритм решения. Этот алгоритм используется далее для разработки архитектуры проекта.

ФОРМУЛИРОВКА ЗАДАЧИ

Такое свойство линейных систем, как независимость от времени, в общем случае определяется утверждением, что импульсная характеристика $h(t, t_0)$ зависит только от величины интервала $(t—t_0)$. Импульсная характеристика представляет собой реакцию системы на единичное импульсное воздействие в момент t_0. Это свойство в сочетании с методами теории суперпозиции для линейных, независимых от времени систем упрощает анализ реакции таких систем на сложные входные сигналы. Сложный входной сигнал можно разложить на ряд более простых сигналов, определить реакцию системы на каждый из них, а затем, используя метод суперпозиции для этих сигналов, получить результирующий отклик системы. Функция, состоящая из суммы таких единичных импульсов, определяется выражением

$$\operatorname{comb}(x) = \sum_{n=-\infty}^{n=+\infty} \delta(x-n). \qquad (1)$$

Входной сигнал такого вида является особенно полезным для получения импульсных характеристик системы.

АНАЛИЗ ЗАДАЧИ

Существует много задач распознавания образов, которые нетрудно описать в терминах двумерного изображения, или, другими словами, с помощью двумерной матрицы данных. Будем рассматривать движущееся изображение как последовательно сменяющие друг друга образы на относительно неизменном фоне. Если необходимо следить за движением и прогнозировать траекторию движения образа, можно использовать методы покрывающей корреляции. Для облегчения восприятия материала читателем рассмотрим задачу слежения за видимым изображением. Однако этот подход может быть использован и для слежения за образом любого другого объекта, движение которого может быть описано с помощью двумерной матрицы данных.

Свойство двумерной отображающей системы, аналогичное линейной временно́й инвариантности, обычно называют *линей-*

ной пространственной инвариантностью. Система является линейной пространственно инвариантной (или линейной инвариантной при сдвиге), если ее импульсная характеристика $h(x_1, y_1; x_0, y_0)$ зависит только от расстояний $(x_1 - x_0)$ и $(y_1 - y_0)$ и описывается зависимостью

$$h(x_1, y_1) = h(x_1 - x_0, y_1 - y_0). \tag{2}$$

При движении точки в поле объектов линейной пространственно-независимой системы будет меняться лишь положение ее образа, но не форма последнего. Очевидно, что этот тип системы является наиболее предпочтительным для систем слежения за изображением. Импульсная характеристика системы в точке (x_1, y_1) пространства выходов при подаче на вход сигнала в виде δ-функции в точке (x_0, y_0) пространства входов имеет вид

$$h(x_1, y_1; x_0, y_0) = S[(x_1 - x_0, y_1 - y_0)], \tag{3}$$

где $S\{\cdot\}$ — математический оператор, описывающий систему.

Хотя функция comb определена на бесконечности, это не соответствует физической реальности; поэтому следует задаться вопросом, на что повлияет ограничение области значений входных сигналов. В соответствии с теорией Уиттекера — Шеннона для определенного класса функций может быть найдена функция, обеспечивающая на выходе системы отображение входных сигналов с требуемой точностью при условии, что интервал дискретизации не будет превышать некоторого предела. Наиболее простой вид эта теорема имеет при использовании прямоугольной решетки замеров функции g, определяемой зависимостью

$$g_s(x, y) = \mathrm{comb}(x/X)\,\mathrm{comb}(y/Y)\,g(x, y). \tag{4}$$

Здесь квантуемая функция g_s состоит из матрицы δ-функций, расположенных на интервалах величиной X по координате x и величиной Y по координате y. Перед проведением анализа таких систем рассмотрим квантующие подсистемы, обеспечивающие подачу такого типа данных на вход нашего фильтра.

ОБРАБОТКА ВИДИМОГО ИЗОБРАЖЕНИЯ

Одним из наиболее важных применений интегральных схем являются одно- и двумерные преобразователи видимого изображения, аналогичные тем, которые впервые были разработаны фирмой Reticon. Общим для всех преобразователей видимого изображения является наличие линейной или двумерной матрицы фотоэлементов, вырабатывающей токи или напряжения,

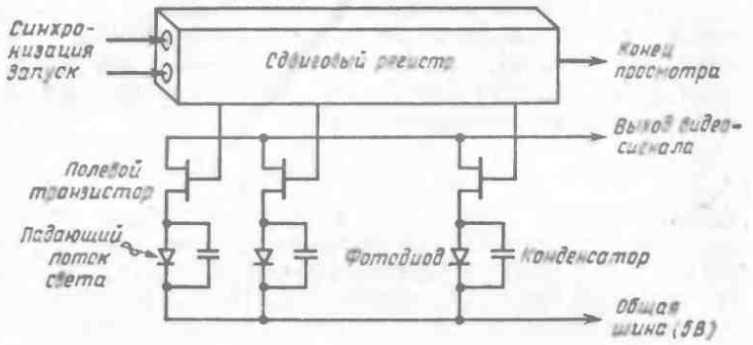


Рис. 6.1. Упрощенная схема линейного сканирующего устройства.

величина которых пропорциональна мощности излучения, падающего на элементы. Наряду с возможностью производства матриц таких фотоэлементов появилась возможность создания полупроводниковых вентилей, которые могут быть объединены в сеть, обеспечивающую последовательный доступ к каждому фотоэлементу и его кратковременное подсоединение к выходному элементу. Следовательно, могут быть разработаны микросхемы, позволяющие считывать сигнал, вырабатываемый фотоэлементом при облучении, а затем сбрасывать его в нуль. Упрощенная схема типичного линейного сканирующего устройства, разработанного фирмой Reticon, показана на рис. 6.1. В этом устройстве к каждому фотодиоду параллельно подсоединена емкость, которая заряжается до напряжения 5 В при открытом полевом транзисторе. Когда полевой транзистор закры-

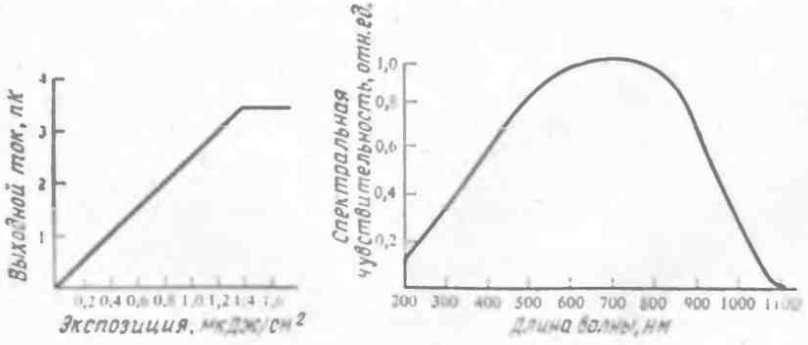


Рис. 6.2. Зависимость выходного тока от экспозиции и спектральная характеристика матрицы фотоэлементов. (*С разрешения* EG&G Reticon.)

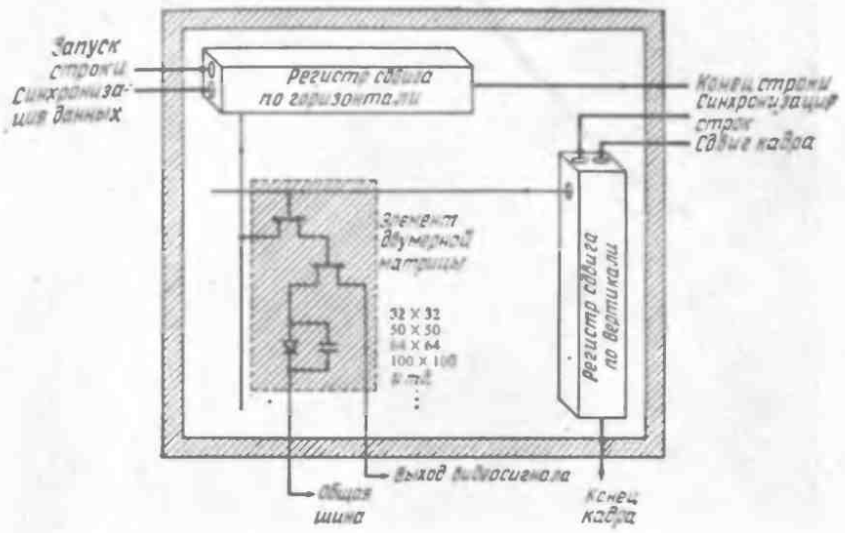


Рис. 6.3. Модифицированная схема сканирующего устройства фирмы Reticon. (*С разрешения* EG&G Reticon.)

вается, диод препятствует разряду емкости. Если на фотодиод падает свет или другое излучение, конденсатор разряжается за счет возникающего фототока. Когда полевой транзистор вновь открывается, ток заряда конденсатора пропорционален интенсивности света и времени облучения. Таким образом, для фиксированной частоты квантования ток заряда может быть использован как мера падающего на фотоэлемент излучения. Типичная зависимость выходного тока от экспозиции и характеристика спектральной чувствительности показаны на рис. 6.2. Модификация схемы позволяет создать сканирующее устройство для двумерной матрицы (рис. 6.3).

Выходные данные устройства сканирования, общий вид которого приведен на рис. 6.4, могут быть интегрированы с помощью аналоговой схемы. Полученный выходной сигнал будет иметь вид, представленный на рис. 6.5. Этот сигнал может одновременно квантоваться с помощью аналого-цифрового преобразователя, что позволит поставить в соответствие выходному сигналу каждого фотоэлемента некоторое число. Числа могут быть последовательно записаны в память; таким образом можно получить отображение распределения интенсивности на матрице. Указанный процесс можно повторить и получить второе отображение в другой области памяти. После этого достаточно провести сравнение двух записанных в памяти отображений,

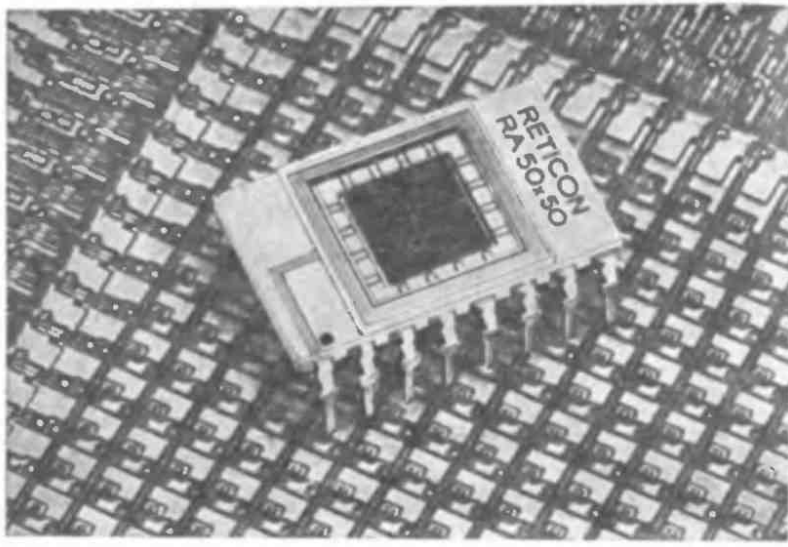


Рис. 6.4. Общий вид сканирующего устройства на фоне кремниевого кристалла. (*С разрешения* EG&G Reticon.)

чтобы выявить и проанализировать любое имеющееся в них отличие. Если разработать соответствующее фокусирующее устройство, то системы описанного здесь типа могут использоваться для выполнения спектрального анализа (одномерный вариант) или анализа изображения (двумерный вариант); последний показан на рис. 6.6.

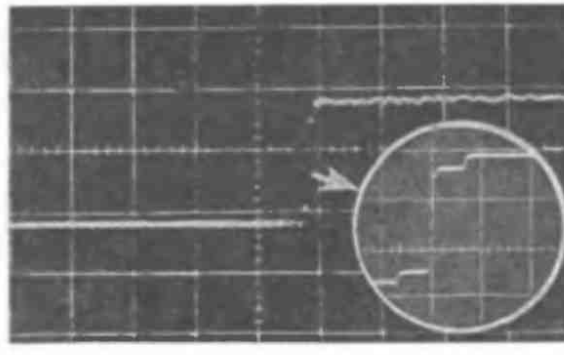


Рис. 6.5. Вид сигнала, получаемого на выходе интегрирующей схемы, на экране электронно-лучевой трубки. (*С разрешения* EG&G Reticon.)

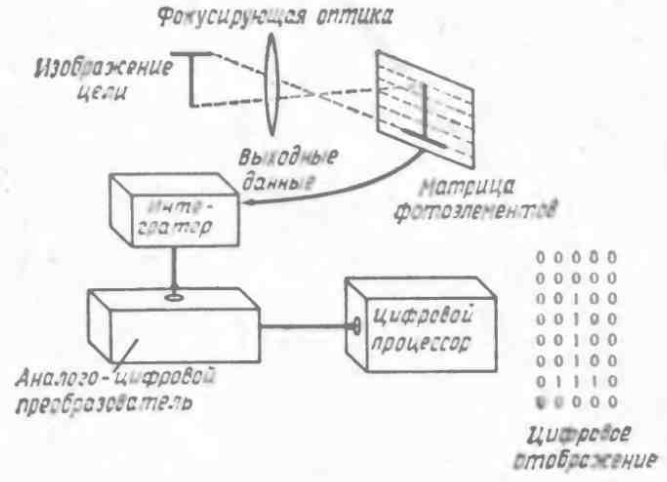


Рис. 6.6. Система обработки изображения.

ФОРМУЛИРОВКА ЗАДАЧИ

При использовании устройств с матрицами из твердотельных фотоэлементов можно получать входные матрицы с количеством замеров от 10 000 до 250 000 на кадр. В большинстве применений, особенно для слежения за движущимся объектом, обработка этой информации в реальном масштабе времени практически неосуществима. Вместо этого может быть использован сканирующий по полю «указатель», обеспечивающий отбор сигналов только из интересующей нас области (рис. 6.7). Таким образом, указатель может следовать за объектом по всему полю обзора (которое в свою очередь также может следовать за объектом).

Допустим, что указатель первоначально направлен на интересующий нас движущийся объект, как показано на рис. 6.8. Указатель направлен в точку (x_j, y_k). Затем он перемещается в точки

$$(x_{j+1}, y_k),\ (x_{j-1}, y_k),\ (x_j, y_{k+1}) \quad \text{и} \quad (x_j, y_{k-1}), \tag{5}$$

причем каждый раз производится преобразование дискретных входных сигналов. Результаты преобразования в начальной точке сравниваются с последующими и выявляются малейшие их отличия друг от друга. Эти операции выполняются для построения отображения нового положения объекта, которое становится начальным для следующего цикла сканирования. Последовательность операций повторяется, и таким образом указатель отслеживает объект по всему полю обзора. При этом,

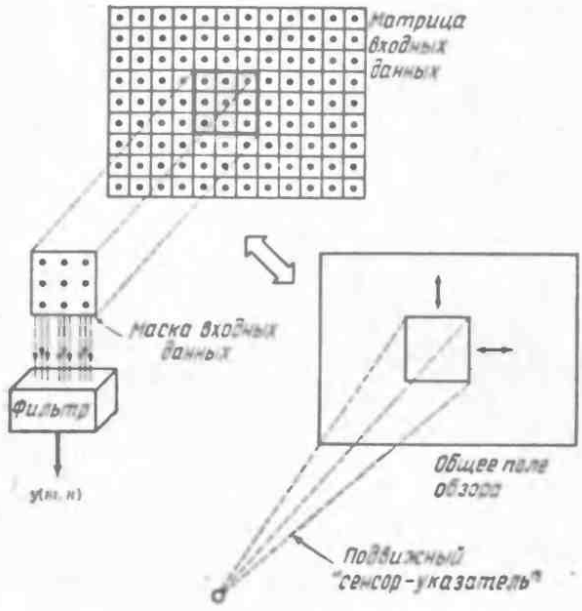


Рис. 6.7. Использование маски фильтра как подвижного сенсора-указателя.

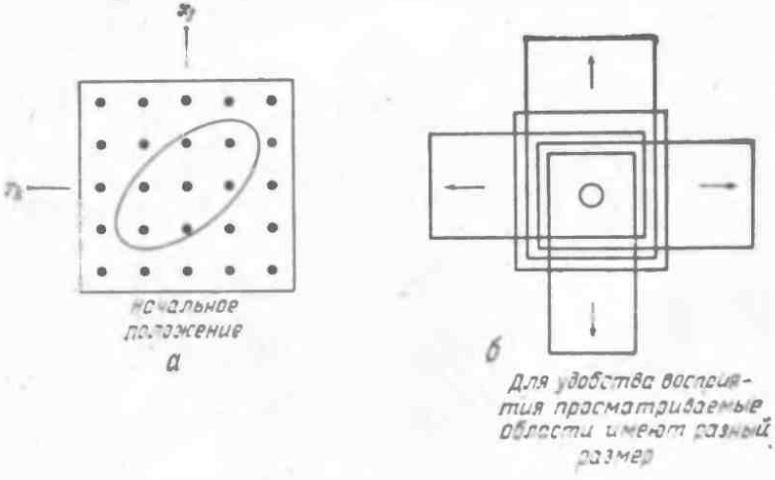


Рис. 6.8. Положения указателя: *а* — начальное; *б* — в режиме слежения.

конечно, может производиться анализ движения указателя для прогнозирования движения объекта, а получаемая информация может использоваться системой для улучшения ее характеристик.

В гл. 7 подробно рассматривается устройство генератора чисел, способного обеспечить выполнение арифметических операций с плавающей точкой, а при легко осуществляемой модификации — и тригонометрических или других операций с рядами, результатами которых являются десятичные дроби с плавающей точкой. Высокопроизводительные генераторы чисел вместе с измерительными устройствами, способными вырабатывать значительные объемы числовой информации в процессе регистрации характеристик природных явлений, могут генерировать числа в таком количестве, что их анализ становится весьма непростым делом. Это вызвало появление целого ряда методов анализа, предназначенных для распознавания образов, определенных на различных классах образов, и даже выделения классов образов, которые необходимо использовать.

Одним из таких методов является корреляционный анализ, в котором глобальная функция определена на области пересечения множеств:

$$F(t) = \int_{z=0}^{z=t} H(x(t), t)\, G(x(t), (z-t))\, dz. \tag{6}$$

До появления ЭВМ математические методы решения таких задач сводились в основном к формированию замкнутых контуров. Описания многих явлений природы формулировались с привлечением уравнений в частных производных или интегродифференциальных уравнений, решения которых зачастую имели вид сложных интегралов по контуру, а их интегрирование могло выполняться аналитически. Очень мощным инструментом анализа такого типа являются прямое и обратное преобразования Фурье для непрерывных функций

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-2\pi i f t}\, dt, \tag{7}$$

$$x(t) = \int_{-\infty}^{+\infty} X(f)\, e^{2\pi i f t}\, df, \tag{8}$$

где $x(t)$ — реальные сигналы, наблюдаемые во временно́м интервале, а $X(f)$ — их отображение в частотном интервале. Вопросам разработки методов анализа с использованием преобразований Фурье посвящена обширная литература, поэтому в данной работе рассматриваются лишь достигнутые в этой области результаты.

ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

Использование цифровых вычислительных машин привело к обработке дискретных данных и к разработке дискретных аналогов исключительно эффективных методов анализа непрерывных функций. В частности, дискретные преобразования Фурье, аналогичные (7) и (8), имеют вид

$$X(j)=\frac{1}{N}\sum_{k=0}^{N-1}x(k)\,e^{-2\pi ijk/N}, \tag{9}$$

$$x(k)=\sum_{j=0}^{N-1}X(j)\,e^{2\pi ijk/N}, \tag{10}$$

при $j=0, 1,\ldots, N-1$ и $k=0, 1, 2,\ldots, N-1$. Произведя обычную замену экспоненциального члена

$$W_N=e^{2\pi i/N}, \tag{11}$$

получим эквивалентные выражения

$$X(j)=\frac{1}{N}\sum_{k=0}^{N-1}x(k)\,W_N^{-jk}, \tag{12}$$

$$x(k)=\sum_{j=0}^{N-1}X(j)\,W_N^{jk}. \tag{13}$$

Хотя Штейглиц показал, что цифровые фильтры эквивалентны аналоговым, и дал математическое обоснование их преобразованию, процесс преобразования сопровождается определенными трудностями. Они обусловливаются граничными условиями и обычно возникают при усечении или выборе неудовлетворительной частоты квантования. Наиболее характерные из отмеченных трудностей, а также общие вопросы интерпретации дискретных преобразований Фурье приведены в работе Берглендa[1].

КОРРЕЛЯЦИЯ НА ДВУМЕРНОЙ МАТРИЦЕ ДАННЫХ

Числа, получаемые от матрицы фотодиодов, описанной в предыдущем разделе, можно рассматривать как двумерную матрицу данных. В этом случае становится возможным определить двумерную глобальную корреляционную функцию, аналогичную одномерной функции. Поскольку операция выполняется только над дискретными данными, интегрирование заменя-

---

[1] Bergland, A Guided Tour of the Fast Fourier Transform, *IEEE Spectrum*, 6, 41 — 52 (July 1969).

ется на суммирование. Такая функция может быть записана в виде

$$y(m, n) = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} x(k, l)\, h(m-k, n-l),$$

где (k, l)-е число обозначено через $x(k,l)$. Операции над данными могут рассматриваться как процесс фильтрации. В этом случае фильтр характеризуется импульсной характеристикой, определяемой как обратное преобразование Фурье над системной функцией.

ФИЛЬТРЫ С КОНЕЧНОЙ И БЕСКОНЕЧНОЙ ИМПУЛЬСНЫМИ ХАРАКТЕРИСТИКАМИ

По признаку устойчивости фильтры делятся на два класса: с *конечной импульсной характеристикой (КИХ)* и *бесконечной импульсной характеристикой (БИХ)*. В соответствии с этим определением фильтров величина выходного сигнала фильтра КИХ ограничена при любом ограниченном по величине входном сигнале, в то время как фильтры БИХ могут иметь бесконечную величину выходного сигнала при ограниченном входном. Более того, бесконечный выходной сигнал может вырабатываться даже вследствие шумов, возникающих при вычислениях. Фильтры КИХ имеют полиномиальные функции преобразования. В фильтрах БИХ в качестве функций преобразования используются рациональные функции. Фильтры БИХ разрабатываются рекурсивно с использованием разностных уравнений, в то время как для фильтров КИХ используют дискретное преобразование Фурье или эквивалентные методы преобразования.

Выходной сигнал фильтра $y(m, n)$ является требуемой корреляционной функцией. Член $h(m, n)$ представляет собой импульсную характеристику фильтра. Фильтры КИХ являются устойчивыми, в то время как фильтры БИХ могут проявлять неустойчивость, как правило, определяемую областями значений входных данных, при которых выходная функция либо не определена, либо неограниченно возрастает. Фильтры БИХ разрабатываются рекурсивным методом, и, хотя для вычислений они в некоторых случаях оказываются более эффективными, чем фильтры КИХ, в дальнейшем мы их рассматривать не будем в связи с их возможной неустойчивостью. Схема обработки двумерной матрицы данных с использованием фильтра приведена на рис. 6.9. Матрица может вводиться в фильтры в виде подмножеств, а импульсная характеристика выбирается в соответствии с каждым из подмножеств. Подмножество обычно выделяется с помощью маски, как показано на рисунке.

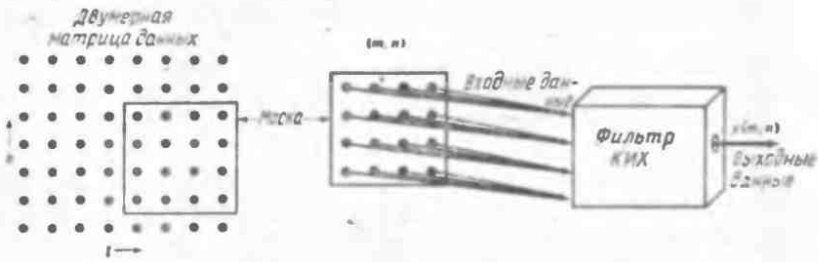


Рис. 6.9. Схема подачи выборки двумерных дискретных данных на вход фильтра КИХ.

Дискретное преобразование Фурье используется в фильтрах КИХ. Хотя операция фильтрации над двумерной матрицей данных весьма проста как концептуально, так и арифметически, количество действий умножения и сложения, которые необходимо выполнить при решении большинства реальных задач, делает прямое вычисление практически невозможным. Однако свертка импульсной характеристики с данными

$$y(m, n) = x(m, n) ** h(m, n), \tag{14}$$

позволяет применить преобразование Фурье

$$Y(k, l) = X(k, l)\, H(k, l), \tag{15}$$

где X, Y и H представляют собой двумерное дискретное преобразование Фурье для x, y и h соответственно. Если отображение данных $x(m, n)$ является матрицей размерностью $M \times N$, то все отображение может обрабатываться по подматрицам, на которые оно может быть разделено. В рассматриваемой проблеме интерес представляют именно подмножества всей матрицы, и преобразования Фурье производятся только над этими подмножествами. Двумерное дискретное преобразование Фурье выглядит следующим образом:

$$C_{xx}(k_1, k_2) = \frac{1}{N_1 N_2} \sum_{m_2=0}^{N_2-1} \sum_{m_1=0}^{N_1-1} x(m_1, m_2)\, W_1^{k_1 m_1} W_2^{k_2 m_2}, \tag{16}$$

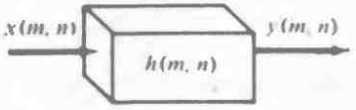
где $W_i = e^{-i2\pi/N_i}$, k_i имеет размерность пространственной частоты, m_i является координатой, а $x(m_1, m_2)$ есть матрица данных размерностью $(N_1 \times N_2)$.

$$x(m_1, m_2) = \begin{bmatrix} x(0,0) & x(0,1) \ldots x(0, N_2-1) \\ x(1,0) & x(1,1) \ldots \\ \vdots & \vdots \quad \vdots \\ x(N_1-1, 0) & \ldots x(N_1-1, N_2-1) \end{bmatrix}. \tag{17}$$

Определенное выше двумерное дискретное преобразование Фурье (ДПФ) обеспечит реализацию в устройстве требуемой в данном случае корреляционной функции.

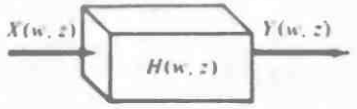
Интерпретация преобразования Фурье одинакова как для одно-, так и для двумерных сигналов: это — характеристика частотного спектра сигнала. Ниже показаны два способа представления двумерного цифрового фильтра, выполненного на БИС.

Представление в пространстве сигналов:



$$y(m, n) = x(m, n) \bullet\bullet h(m, n)$$

Представление в пространстве преобразований:



$$Y(w, z) = X(w, z) \bullet\bullet H(w, z)$$

Во многих случаях более удобно работать с сепарабельными функциями, так как свойство сепарабельности зачастую позволяет сводить сложные двумерные преобразования к более простым одномерным.

Функцию g называют *сепарабельной* в прямоугольных координатах (x, y), если

$$g(x, y) = g_X(x)\, g_Y(y). \tag{18}$$

Сепарабельная в прямоугольных координатах функция обладает тем свойством, что ее двумерное преобразование Фурье является произведением соответствующих одномерных преобразований:

$$F\{g(x, y)\} = F_X\{g_X\}\, F_Y\{g_Y\}. \tag{19}$$

Отметим, что функция comb сепарабельна в прямоугольных координатах:

$$\begin{matrix} \text{Функция} & & \text{Преобразование} \\ \operatorname{comb}(x) \operatorname{comb}(y) & = & \operatorname{comb}(f_X) \operatorname{comb}(f_Y). \end{matrix}$$

В следующем разделе будет показано, как двумерное ДПФ может быть выполнено посредством одномерных ДПФ.

РЕАЛИЗАЦИЯ ДВУМЕРНЫХ ФИЛЬТРОВ С ПОМОЩЬЮ ОДНОМЕРНЫХ

Выше одномерное дискретное преобразование Фурье было определено как

$$C_x(l) = \frac{1}{N_1} \sum_{r=0}^{N_1-1} x_l W^{rl} \quad \text{при } r = 0, 1, 2, \ldots, N-1, \tag{20}$$

где $W = e^{-2\pi i/N}$, r — коэффициенты ДПФ, x_l — l-я выборка временно́го ряда.

Анализ внутренней суммы в двумерном преобразовании Фурье

$$\frac{1}{N_1} \sum_{m_1=0}^{N_1-1} x(m_1, m_2) W_1^{k_1 m_1} =$$

$$= \frac{1}{N_1} \{x(0, m_2) + x(1, m_2) W_1^{k_1} + \cdots + x(N_1-1, m_2) W_1^{k_1 (N_1-1)}\} \tag{21}$$

показывает, что она представляет собой одномерное преобразование Фурье столбца m_2 матрицы данных $x(m_1, m_2)$, которое можно записать как

$$C_x(k_1, m_2) = \frac{1}{N_1} \sum_{m_1=0}^{N_1-1} x(m_1, m_2) W_1^{k_1 m_1}, \tag{22}$$

где m_2 — номер столбца матрицы данных, а k_1 — пространственная частота. Результатом суммирования по столбцам является функция пространственной частоты, а это и есть цель преобразований. Коэффициент $C_x(k_1, m_2)$ можно записать в виде матрицы $N_1 \times N_2$. При его подстановке в уравнение (16) получим выражение

$$C_{xx}(k_1, k_2) = \frac{1}{N_2} \sum_{m_2=0}^{N_2-1} C_x(k_1, m_2) W_2^{k_2 m_2}. \tag{23}$$

Развернутая запись уравнения (23) имеет вид

$$C_{xx}(k_1, k_2) = \frac{1}{N_2} [C_x(k_1, 0) + C_x(k_1, 1) W_2^{k_2} +$$

$$+ \cdots + C_x(k_1, N_2-1) W_2^{k_2 (N_2-1)}]. \tag{24}$$

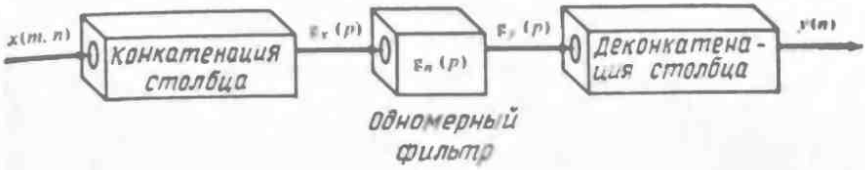


Рис. 6.10. Схема построения двумерного фильтра на базе одномерных фильтров.

Как нетрудно заметить, уравнение (24) эквивалентно сумме ДПФ строк матрицы $C_x(k, m)$. Полученное в результате выражение

$$[C_{xx}(k_1, k_2)] = \begin{bmatrix} C_{xx}(0,0) & C_{xx}(0,1)\ldots & C_{xx}(0, N_2-1) \\ C_{xx}(1,0) & C_{xx}(1,1)\ldots & \\ \vdots & & \\ C_{xx}(N_1-1,0) & \ldots\ldots & C_{xx}(N_1-1, N_2-1) \end{bmatrix} \quad (25)$$

показывает, что вычисление двумерного ДПФ эквивалентно $N \times N$ вычислениям одномерных ДПФ. Информация, содержащаяся в члене $C_{xx}(k_1, k_2)$, позволяет определять категорию каждого из $(N_1 \times N_2)$ образов точек в соответствии с пространственно-частотным содержанием образов, а именно это и нужно для обеспечения корреляционного слежения.

Более общей формой записи, определяющей порядок вычисления $C_{xx}(k_1, k_2)$, является выражение

$$[C_{xx}(k_1, k_2)] = \frac{1}{N_1 N_2} \Lambda_1 [x(m_1, m_2)] \Lambda_2, \quad (26)$$

где $x(m, n)$ является матрицей данных, а Λ_1 и Λ_2 определяются следующим образом:

$$\Lambda_1 = [\alpha_{pq}]_{N_1 \times N_1}, \quad (27)$$

где $\alpha_{pq} = W_1^{pq}$ при $p, q = 0, 1, \ldots, N_1 - 1$, и

$$\Lambda_2 = [\beta_{pq}]_{N_2 \times N_2}, \quad (28)$$

где $\beta_{pq} = W_2^{pq}$ при $p, q = 0, 1, 2, \ldots, N_2 - 1$ $(W_1 = e^{-2\pi i/N_1}$, $W_2 = e^{-2\pi i/N_2})$.

Рис. 6.10 иллюстрирует основную идею используемого нами подхода, заключающуюся в том, что одномерные (линейные пространственно-инвариантные) фильтры можно использовать для вычисления двумерных результатов. Основной целью проектирования является построение базового вычислительного устройства, осуществляющего двумерное ДПФ посредством одномерных ДПФ.

БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ (БПФ)

Знание преобразований Фурье большинством инженеров и физиков, а также эквивалентность (за исключением присущих дискретным системам граничных условий) их дискретных и непрерывных версий обусловливают широкое применение этих преобразований для решения многих задач спектрального и корреляционного анализа. Однако большой объем вычислений, связанный с ДПФ, лимитирует их применение в тех случаях, когда имеются ограничения на время вычислений и память. Кули и Тьюки рассмотрели и решили эту проблему для многих применений. Ниже изложены основные положения их подхода к решению указанной проблемы, рассмотрен специальный пример и проведен анализ связанных с этой проблемой факторов.

Алгоритм БПФ разрабатывается для вычисления выражения, которое при использовании обозначений Кули имеет вид

$$\hat{X}(j) = \sum_{k=0}^{N-1} A(k)\, W^{jk}, \tag{29}$$

где $j=0, 1,\ldots,N-1$, а $W=e^{2\pi i/N}$. В приведенных выше выражениях вместо $\hat{X}$ и A употреблялись X^* и x^*/N для прямого преобразования и x и X для обратного преобразования. Звездочка указывает на комплексное сопряжение.

Рассмотрим пример, где $N=8$ (т. е. $j=0, 1,\ldots,7$, $k=0, 1,\ldots,7$), и представим j и k в двоичной форме записи

$$j = 4j_2 + 2j_1 + j_0, \quad k = 4k_2 + 2k_1 + k_0, \tag{30}$$

где $j_i,\ k_i \in \{0, 1\}$.

Тогда уравнение (29) примет вид

$$\hat{X}(j_2, j_1, j_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} A(k_2, k_1, k_0)\, W^{(4j_2+2j_1+j_0)(4k_2+2k_1+k_0)}. \tag{31}$$

Поскольку $W^{m+n} = W^m \cdot W^n$, имеем равенство

$$W^{(4j_2+2j_1+j_0)(4k_2+2k_1+k_0)} = W^{(4j_2+2j_1+j_0)4k_2}\, W^{(4j_2+2j_1+j_0)2k_1}\, W^{(4j_2+2j_1+j_0)k_0}. \tag{32}$$

В результате дальнейших преобразований получим следующие выражения:

$$W^{(4j_2+2j_1+j_0)4k_2} = [W^{8(2j_2+j_1)k_2}]\, W^{4j_0k_2}, \tag{33}$$

$$W^{(4j_2+2j_1+j_0)2k_1} = [W^{8(j_2k_1)}]\, W^{(2j_1+j_0)2k_1}, \tag{34}$$

$$W^{(4j_2+2j_1+j_0)k_0} = W^{(4j_2+2j_1+j_0)k_0}. \tag{35}$$

Поскольку $W^8 = [e^{2\pi i/8}]^8 = e^{2\pi i} = 1$, выражения в квадратных скобках, имеющие число 8 в показателе экспоненты, можно за-

менить единицей, так как остальные показатели степени роли не играют. Таким образом, выражение (31) примет вид

$$\hat{X}(j_2, j_1, j_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} A(k_2, k_1, k_0) W^{4j_0k_2} W^{2k_1(2j_1+j_0)} W^{k_0(4j_2+2j_1+j_0)} \quad (36)$$

$$\underbrace{\qquad\qquad}_{A_1(j_0, k_1, k_0)}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{A_2(j_0, j_1, k_0)}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{A_3(j_0, j_1, j_2)}$$

Аддитивность задачи позволяет выделить следующие этапы вычислений:

$$A_1(j_0, k_1, k_0) = \sum_{k_2=0}^{1} A(k_2, k_1, k_0) W^{4k_2j_0}, \quad (37)$$

$$A_2(j_0, j_1, k_0) = \sum_{k_1=0}^{1} A_1(j_0, k_1, k_0) W^{2k_1(2j_1+j_0)}, \quad (38)$$

$$A_3(j_0, j_1, j_2) = \sum_{k_0=0}^{1} A_2(j_0, j_1, k_0) W^{k_0(4j_2+2j_1+j_0)}, \quad (39)$$

$$\hat{X}(j_2, j_1, j_0) = A_3(j_0, j_1, j_2). \quad (40)$$

Кули и Тьюки первыми получили эти рекурсивные уравнения.

В рассмотренном примере для $N=8$ требуется произвести лишь 48 операций сложения и умножения по сравнению с 64, необходимыми при прямых вычислениях в соответствии с уравнением (29). Учитывая, что умножение на $+1$ можно не выполнять — тем самым число операций БПФ сократится до 24 — и что $W^0 = -W^4$ и $W^1 = -W^5$ и т. д., число необходимых операций будет равно 12. Описанный прием может применяться при $N=2^m$, при этом число выполняемых операций уменьшится с N^2 до $(N/2)\log_2 N$ умножений и $(N/2)\log_2 N$ сложений комплексных чисел и $(N/2)\log_2 N$ вычитаний. Для $N=1024$ количество вычислений, необходимых для ДПФ, уменьшается более чем в 200 раз. В табл. 6.1 приведены данные, позволяющие сравнить количество операций умножения, которые необходимо произвести при использовании прямого метода и метода с использованием БПФ.

Таблица 6.1. Сравнение числа операций умножения при использовании прямого метода и БПФ

Функция	Выражение	Примерное число операций умножения (верхние сравнимые границы)	
		Метод прямой	Метод БПФ
Дискретное преобразование Фурье	$\sum_{k=0}^{N-1} X_k e^{-2\pi i r k/N}$ при $r=1, 2,\ldots, N-1$	N^2	$2N \log_2 N$
Фильтрация	$\sum_{k=0}^{N-1} X_k Y_{u-k'}$ при $u=0, 1,\ldots, N-1$	N^2	$3N \log_2 N$
Автокорреляционные функции	$\sum_{k=0}^{N-1-r} X_k X_{r+k'}$ при $r=0, 1,\ldots, N-1$	$\frac{N}{4}\left(\frac{N}{2}+3\right)$	$3N \log_2 N$
Двумерное преобразование Фурье (анализ траекторий)	$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{k,l} e^{-2\pi i(kq+rl/N)}$ при $r, q=0, 1,\ldots, N-1$	N^4	$4N^2 \log_2 N$
Двумерная фильтрация	$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{k,l} Y_{q-k,r-l'}$ при $q, r=1, 2,\ldots, N-1$	N^4	$3N^2 \log_2 N$

Основой вычислительного алгоритма как для прямого, так и для обратного ДПФ служит зависимость

$$A(n)=\sum_{k=0}^{N-1} a(k)\,W^{-nk} \text{ при } k=0, 1,\ldots, N-1, \tag{41}$$

где входом является $x(k)=a(k)$ (при $k=0, 1,\ldots, N-1$), а выходом — $x(n)=A(n)/N$ (при $n=0, 1,\ldots, N-1$).

Быстрое преобразование Фурье основано на исключении избыточности, возникающей из-за члена $a(k)W^{-nk}$ вследствие периодичности W с периодом N. Для любого заданного значения k при увеличении n от 0 до $N-1$ произведение проявит свой периодический характер, пройдя k периодов за весь цикл, причем даже в одном периоде может выявиться его симметричность как комплексной сопряженной величины. Эта симметрия используется во всех алгоритмах БПФ для сокращения числа необходимых операций умножения. По получаемому результату алгоритмы БПФ идентичны алгоритмам ДПФ, однако в отношении вычислений более эффективны. В общем случае

$$\text{ДПФ} \sim N^2 \quad \text{БПФ} \sim N \log N$$

Суть БПФ состоит в матричном разложении на множители. Анализ различных используемых для этого оснований проведен в работе Бергленда. Как следует из блок-схемы (рис. 6.11), непосредственная реализация ДПФ весьма проста.

Анализ выражения

$$A(n)=\sum_{k=0}^{N-1} a(k)\,W^{-nk} \text{ при } n=0, 1,\ldots, N-1 \tag{42}$$

показывает, что над данными можно произвести так называемую *децимацию*. В простейшем случае эта операция заключается в разделении четных и нечетных точек данных при соблюдении последовательности точек, в результате чего получаются последовательности длиной $(N/2)-1$. Таким образом, дальнейшие вычисления производятся над последовательностями

$$a_0, a_2, a_4, a_6, \ldots, a_{N/2} \quad \text{и} \quad a_1, a_3, a_5, a_7, \ldots, a_{(N/2)-1}. \tag{43}$$

Если принять

$$A(n) \longleftarrow 1/2\,[A(n)+A(n+N/2)\cdot W^n]$$

и

$$A(n+N/2) \longleftarrow 1/2\,[A(n)-A(n+N/2)\cdot W^n]$$

при

$$n=0, 1,\ldots, (N/2-1), \tag{44}$$

то возможно вычислить все значения $A(n)$ с помощью двух пре-

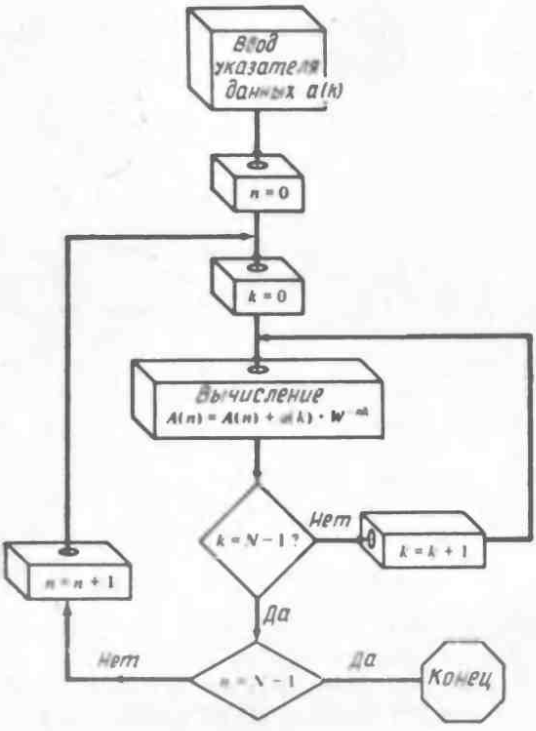
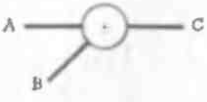


Рис. 6.11. Блок-схема выполнения ДПФ.

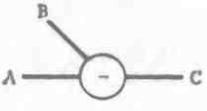
образований последовательностей точек данных длиной $N/2$ с последующей операцией взвешивания. При надлежащем выборе N операция децимации может быть повторена. Графическое описание последовательности операций складывается из следующих элементов:

$$B = A \cdot W^k \qquad (45)$$

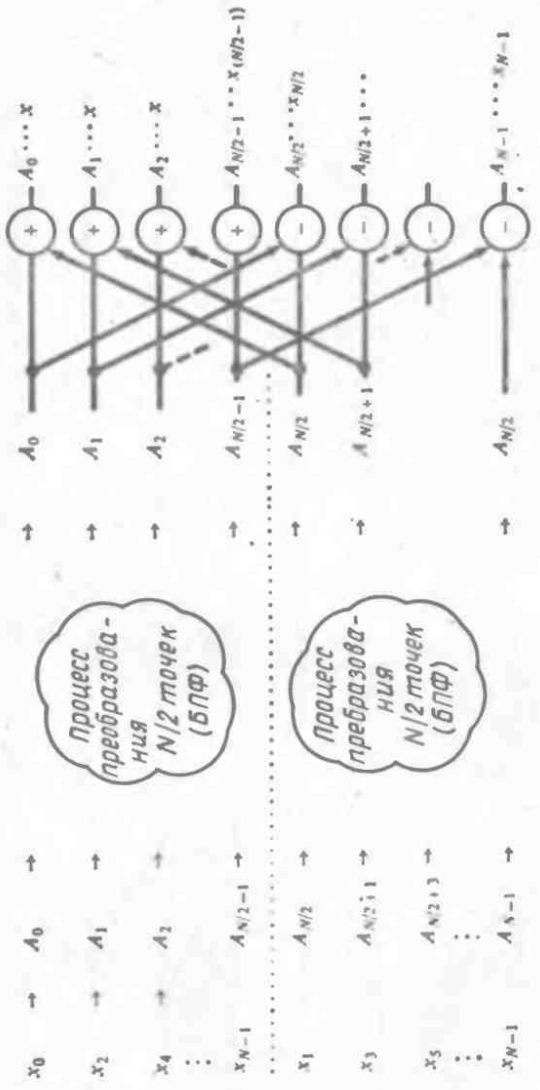
$$C = A + B \qquad (46)$$



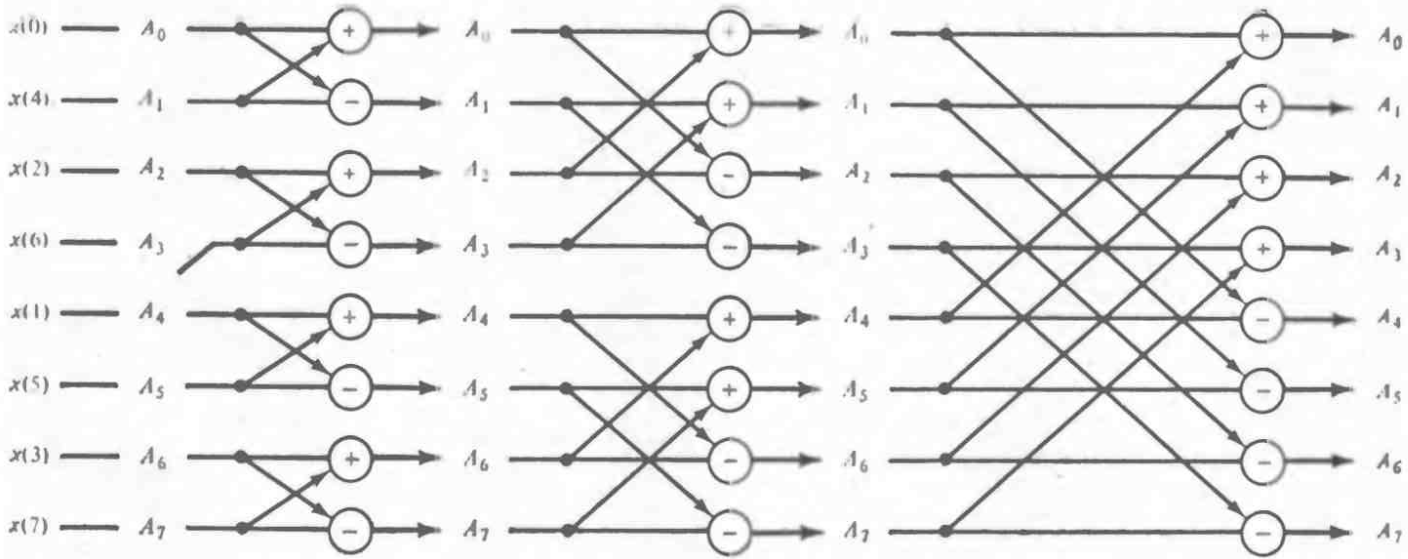
$$C = A - B \qquad (47)$$



Использование этих элементов показано на нижеследующей диаграмме, построенной в предположении, что исходные данные поделены на две последовательности (четную и нечетную):



При выборе $N=2^m$ процесс децимации входной последовательности может быть повторен m раз, так что останутся только двухточечные преобразования, в которых БПФ сводится к


x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
A0
A1
A2
A3
A4
A5
A6
A7
+
−

последовательности операций взвешивания (умножения) и сложения данных из матрицы $A(n)$.

В качестве иллюстрации на стр. 224 представлена диаграмма для случая $N=8-2$, которая состоит из трех последовательных децимаций. Далее будут обсуждены некоторые вопросы, связанные с БПФ, и рассмотрено несколько возможных вариантов его реализации

БЛОК-СХЕМА ВОСЬМИТОЧЕЧНОГО БПФ

Матричное запоминающее устройство реализует накопление с замещением, т. е. каждая область устройства используется и как регистр-источник, и как регистр-приемник. Например, при выполнении крайней слева операции (см. вышеприведенную диаграмму) содержимое A_0 складывается с содержимым A_1 и сумма запоминается в A_0. При вычитании A_0 из A_1 разность записывается в A_1. В связи с этим необходимо, чтобы блок обработки содержал регистры временного хранения. При рассмотрении требований к организации хранения данных важным является и тот факт, что на каждом уровне обработки входные данные стираются, а в область их хранения записываются результаты операции. Заслуживает внимания «перемешанная» матрица входных данных. Входная последовательность имеет вид

$$x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7,$$

в то время как начальное расположение данных в запоминающем устройстве следующее:

$$x_0 \quad x_4 \quad x_2 \quad x_6 \quad x_1 \quad x_5 \quad x_3 \quad x_7.$$

Если представить числа в двоичной форме, то легко понять, почему формируется именно такая последовательность данных:

x_0...000	*перестановка бита*	000...x_0
x_1...001	——————→	100...x_4
x_2...010		010...x_2
x_3...011		110...x_6
x_4...100		001...x_1
x_5...101		101...x_5
x_6...110		011...x_3
x_7...111		111...x_7

Именно такая «перемешанная» последовательность и позволяет производить вычисления с замещением. Быстрое преобразование Фурье можно вычислять и с использованием первона-

чальной последовательности данных, однако свойство замещения при этом теряется. Непосредственная реализация алгоритма перестановки бита будет описана ниже.

РЕКУРСИВНОЕ ВЫЧИСЛЕНИЕ БПФ

Простая последовательность этапов вычислений с использованием свойства замещения позволяет обозначать их с помощью индексов. Матрица исходных данных, хранимая в n-й

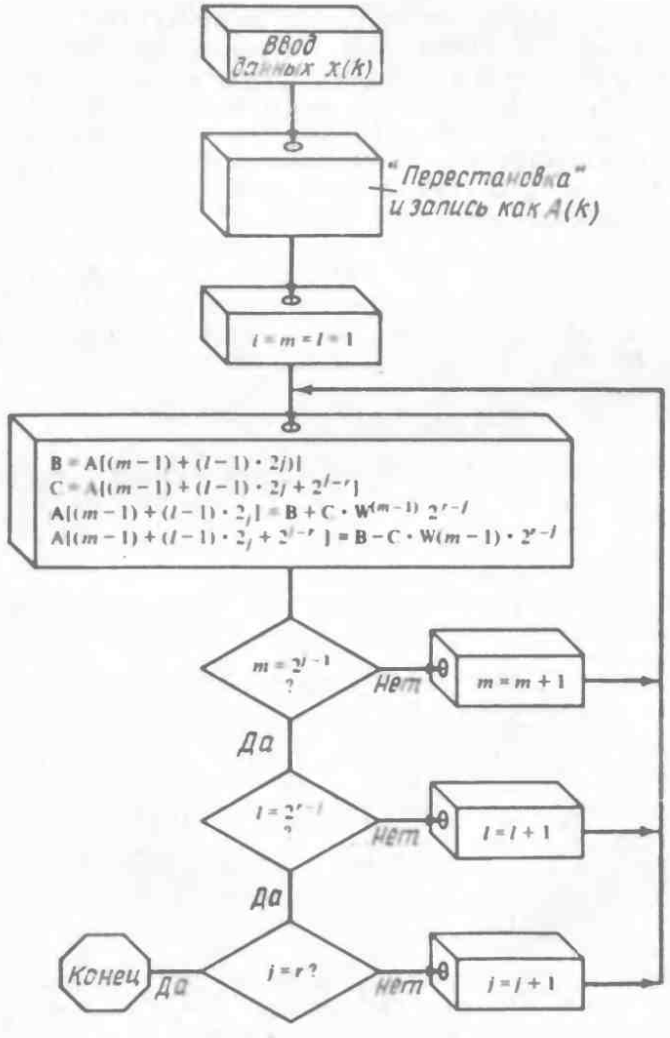


Рис. 6.12. Блок-схема рекурсивного вычисления БПФ.

области, обозначается как $A_2(n)$, а результат m-го этапа — как $A_{m+1}(n)$. Блок-схема вычислений построена на следующих рекурсивных зависимостях:

$$A_{j+1}(n) \longleftarrow [A_j^r(n) + A_j(n+N/2)\cdot W^n], \quad (48)$$

$$A_{j+1}(n+N/2) \longleftarrow [A_j(n) - A_j(n+N/2)\cdot W^n]. \quad (49)$$

В более наглядной форме вышесказанное представлено на блок-схеме (рис. 6.12). Вычисления включают суммирование по трем параметрам: i, l, m.

РЕАЛИЗАЦИЯ БПФ

Используя блок-схему (рис. 6.12), легко представить себе набор основных операций, выполняемых при рекурсивном вычислении БПФ:

1) перестановка бита;
2) подсчет числа циклов;
3) умножение и сложение;
4) вычисление sin/cos.

Рассмотрим каждую из этих операций с точки зрения возможности ее использования в системе слежения. Обсудим также достоинства и недостатки каждого из способов реализации этих операций, отметив «наилучшее» решение.

1. Перестановка бита. Реорганизация входных данных в области хранения матриц с переставленными битами может быть осуществлена как программным, так и аппаратным путем. При аппаратной реализации эта операция выполняется быстрее, чем при программном способе — за 1 микроцикл. На рис. 6.13 показано, как можно реализовать эту операцию с использованием процессорных элементов 3002 фирмы Intel. Аналогичную схему можно получить, применив набор регистров фирмы AMD.

Однако при этом потребуется ввести буфер с тремя устойчивыми состояниями в цепь обратной связи (в схеме на ПЭ 3002 это также необходимо сделать, если для этого нельзя использовать входной порт). В рассматриваемой схеме адрес матрицы хранится в одном из регистров внутреннего набора регистров. В соответствующие моменты времени он увеличивается и загружается в аккумулятор, из которого выдается на шину вывода данных. Затем адрес матрицы подается во входной порт. Три младших бита (в случае восьмиточечной матрицы, см. рис. 6.13) переставляются с использованием соответствующей распайки проводов, в то время как порядок старших битов остается прежним. Входное значение загружается в аккумулятор и копируется в регистре адреса памяти. Этот адрес используется для доступа к нужным данным матрицы.

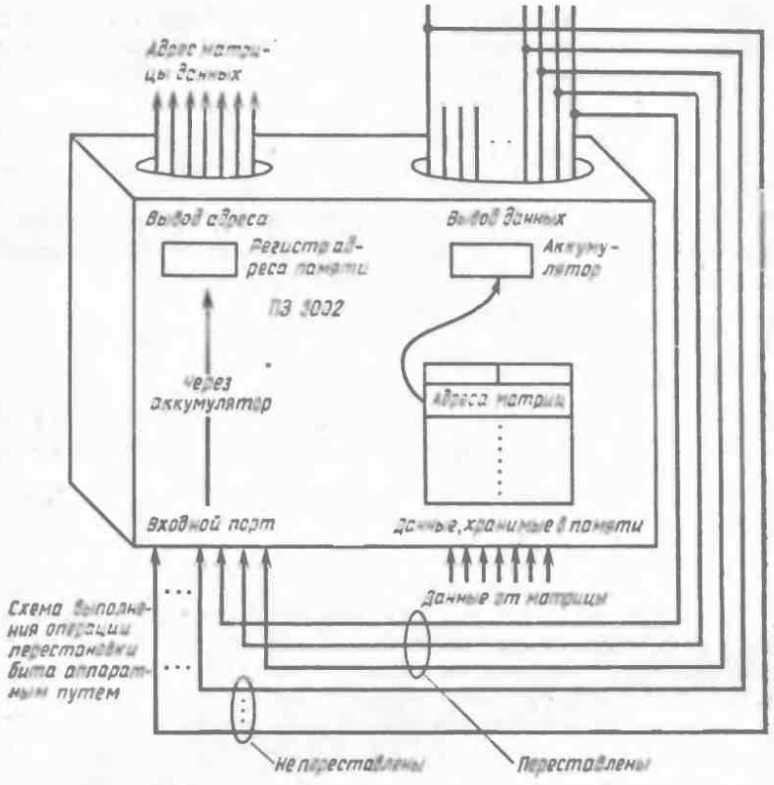


Рис. 6.13. Аппаратная реализация операции перестановки бита.

2. Подсчет числа циклов. Второй операцией, необходимой для реализации рекурсивного алгоритма вычисления БПФ, является подсчет числа циклов и управление ими. Для управления тремя вложенными друг в друга циклами необходимы три числа. Эти числа могут храниться в регистре (внутреннем или внешнем) и увеличиваться посредством их сложения со значениями аппаратных счетчиков. Уменьшение и контроль значений последних производятся параллельно с выполнением других операций, что позволяет избежать дополнительных затрат времени, за исключением операций по начальной установке счетчиков, выполняющихся при любом способе их реализации. Решающим доводом для скорейшего внедрения счетчиков является их дешевизна и компактность. Схема устройства, построенного на основе счетчиков циклов, показана на рис. 6.14. Она может быть реализована как на устройствах серии 3000, так и на устройствах серии 2900 (или на основе их комбинации).

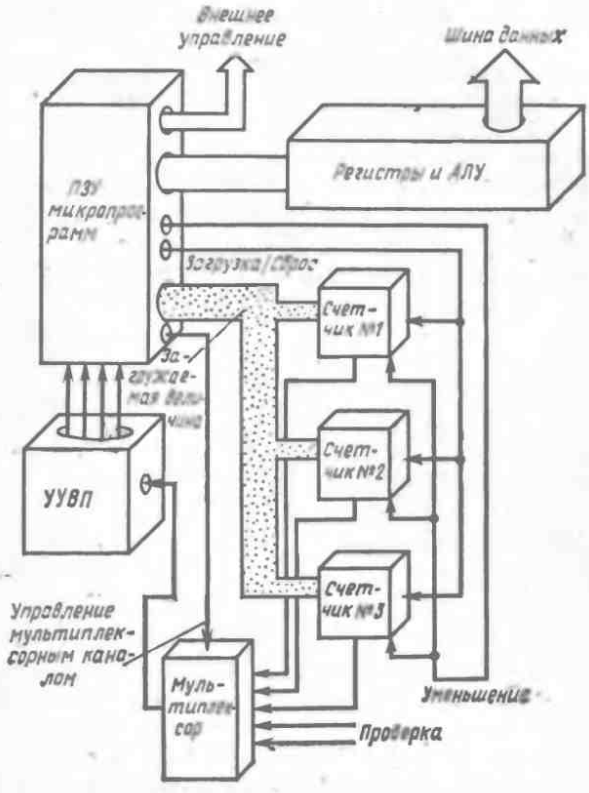


Рис. 6.14. Аппаратная реализация счетчиков циклов.

В аппаратно реализованных счетчиках с помощью микропрограмм осуществляются установка начального значения счетчика, его однократное уменьшение в каждом цикле, проверка условия выхода из цикла, выход из него или возврат в точку входа цикла.

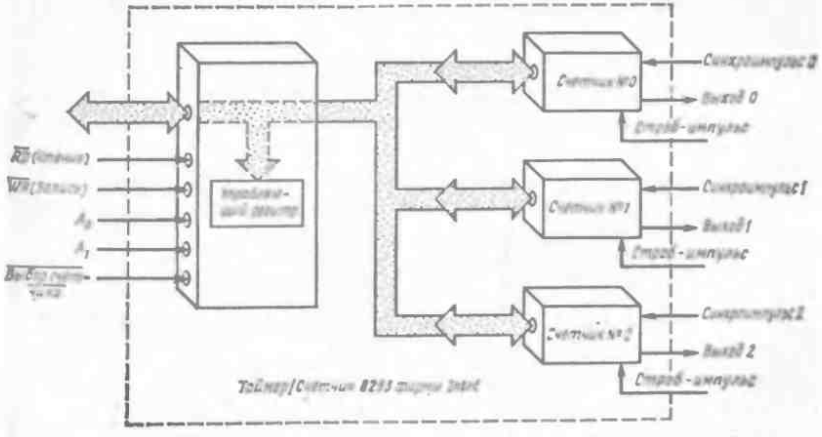
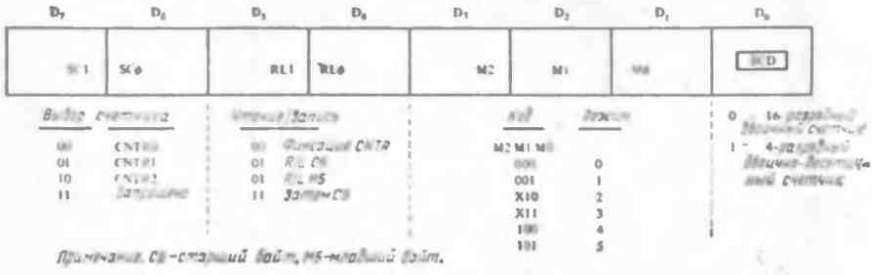


Рис. 6.15. Пример использования устройства 8253 фирмы Intel для построения счетчиков циклов.

Если частота цикла менее 2 МГц, то для организации счетчика можно использовать БИС 8253 фирмы Intel (рис. 6.15). Эта схема включает три 16-разрядных счетчика, каждый из которых может быть загружен некоторым начальным значением и имеет независимо управляемую входную шину синхроимпульсов. Кроме того, каждый счетчик может функционировать в одном из пяти режимов, выбор которого осуществляется путем загрузки в регистр управления соответствующего управляющего слова, имеющего следующий формат:



Коды, определяющие номер счетчика и двоично-десятичный или двоичный режим его работы, не требуют пояснений. Коды «Чтение/Запись» используются для указания, какой байт данного 16-разрядного счетчика должен быть считан или записан в соответствии с импульсом «Чтение» или «Запись». Код 00 используется для асинхронного сохранения значения счетчика без его остановки. Каждый из счетчиков может работать в следующих режимах:

Режим 0 «Прерывание по окончании счета». Уровень выходного сигнала становится высоким после отсчитывания заданного числа синхроимпульсов.

Режим 1 «Программируемый импульс». Уровень выходного сигнала становится низким по переднему фронту входного сигнала от вентиля и высоким после отсчитывания заданного числа синхроимпульсов.

Режим 2 «Генератор частоты». Производится деление частоты входных синхроимпульсов на N, где N — число, предварительно загруженное в счетчик.

Режим 3 «Генератор прямоугольных импульсов». Аналогичен режиму 2, за тем исключением, что уровень выходного сигнала поддерживается высоким в течение $(N—1)/2$ такта, а затем в течение $(N—1)/2$ такта — низким.

Режим 4 «Программно запускаемый строб». Уровень выходного сигнала остается высоким до окончания отсчитывания заданного числа синхроимпульсов, после чего в течение одного такта поддерживается низким.

Режим 5 «Аппаратно запускаемый строб». Счетчик начинает производить отсчет по переднему фронту входного сигнала от вентиля, после отсчитывания заданного числа синхроимпульсов уровень выходного сигнала поддерживается низким в течение одного такта.

Временные диаграммы для различных режимов функционирования БИС 8253 приведены на рис. 6.16. Можно использовать и другую схему устройства, производящего подсчет числа циклов (рис. 6.17). Разработан набор микрокоманд, позволяющий осуществлять установку нужного режима функционирования устройства 8253 занесением соответствующего управляющего слова в выходное поле управляющего постоянного запоминающего устройства и стробированием импульса «Запись». Эти операции повторяются для загрузки начального значения счета в каждый из счетчиков. В каждом из циклов по одному

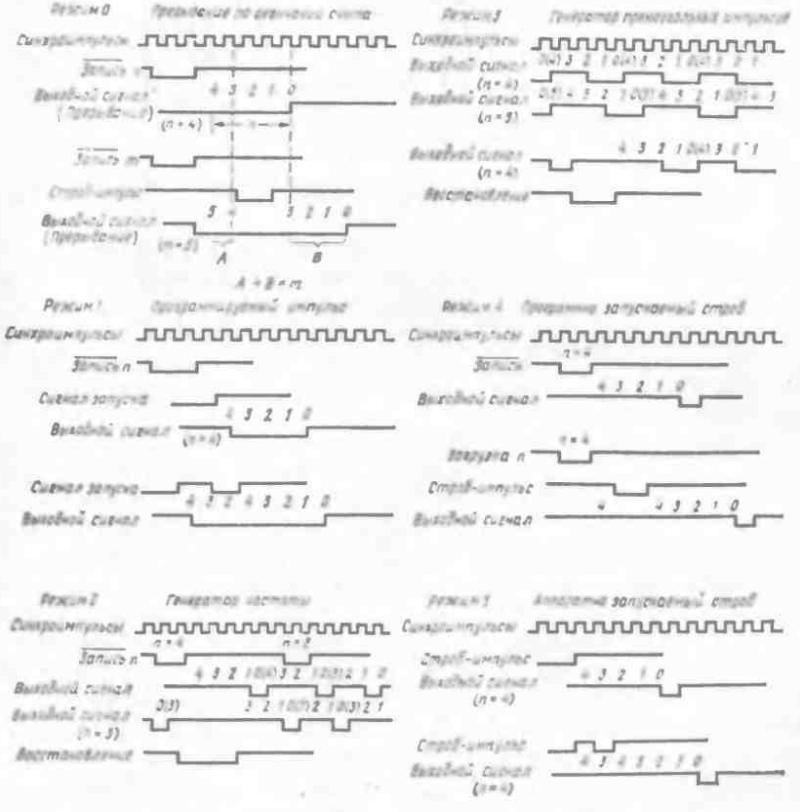


Рис. 6.16. Временные диаграммы для таймеров/счетчиков БИС 8253. (*С разрешения* Intel Corp.)

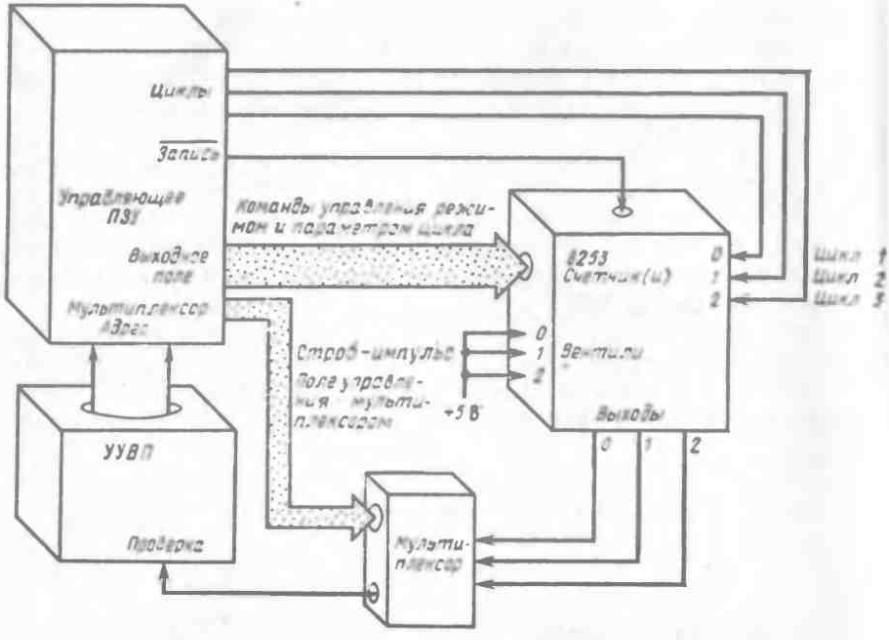


Рис. 6.17. Вариант схемы построения счетчика циклов.

разу активируется цепь соответствующего цикла, и поле управления мультиплексором выбирает мультиплексный канал для проверки соответствующих выходных цепей. Поскольку цепи циклов управляются микропрограммно и микросхема 8253 функционирует в режиме 4, на вентильные цепи подается напряжение +5 В, таким образом постоянно поддерживая счетчик в незапертом состоянии.

3. Реализация схемы умножения. Необходимые операции умножения могут быть реализованы как аппаратным, так и программным способом. Программная реализация сводится к простому использованию внутренних регистров и микропрограммы, осуществляющей сдвиг множителя и сдвиг или сложение — сдвиг частичного произведения, полученного по результатам проверки. При аппаратном способе в полной мере используются устройства параллельной обработки, такие, как MPY16 фирмы TRW, которые выполняют перемножение 16-разрядных чисел за ~100 нс. Общая схема применения внешних схем умножения для выполнения БПФ показана на рис. 6.18. В этом случае данные хранятся во внутренних регистрах и выдаются во внешний блок умножения. При программном способе реализации блок умножения непосредственно сопрягается с областью хранения матриц. При таком построении устройства

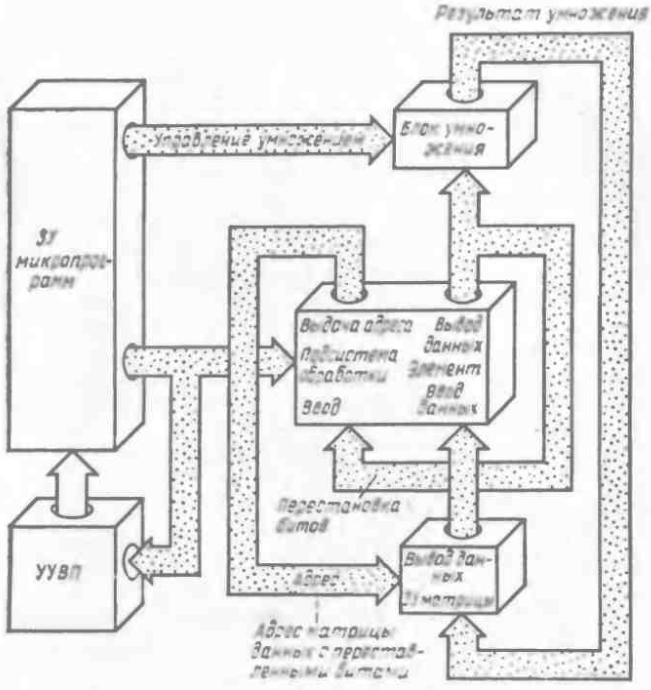


Рис. 6.18. Простейшая схема реализации операции умножения для БПФ.

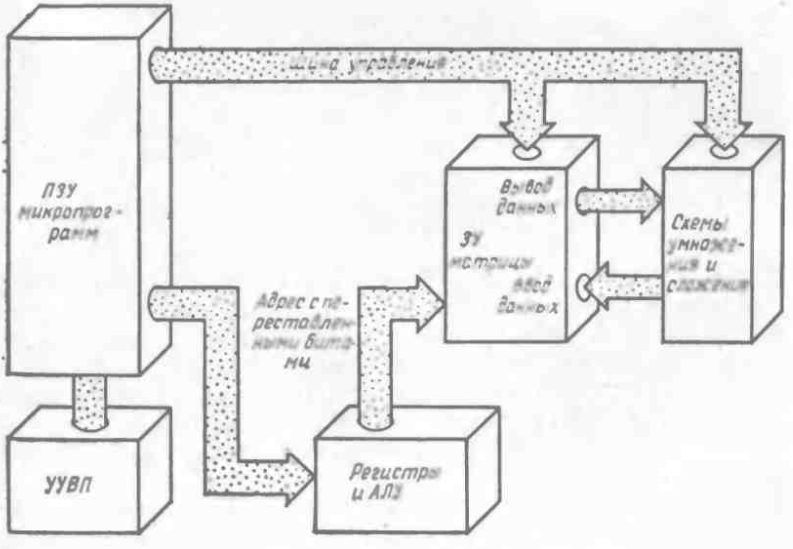


Рис. 6.19. Схема реализации операции умножения для БПФ, обеспечивающая наибольшее быстродействие.

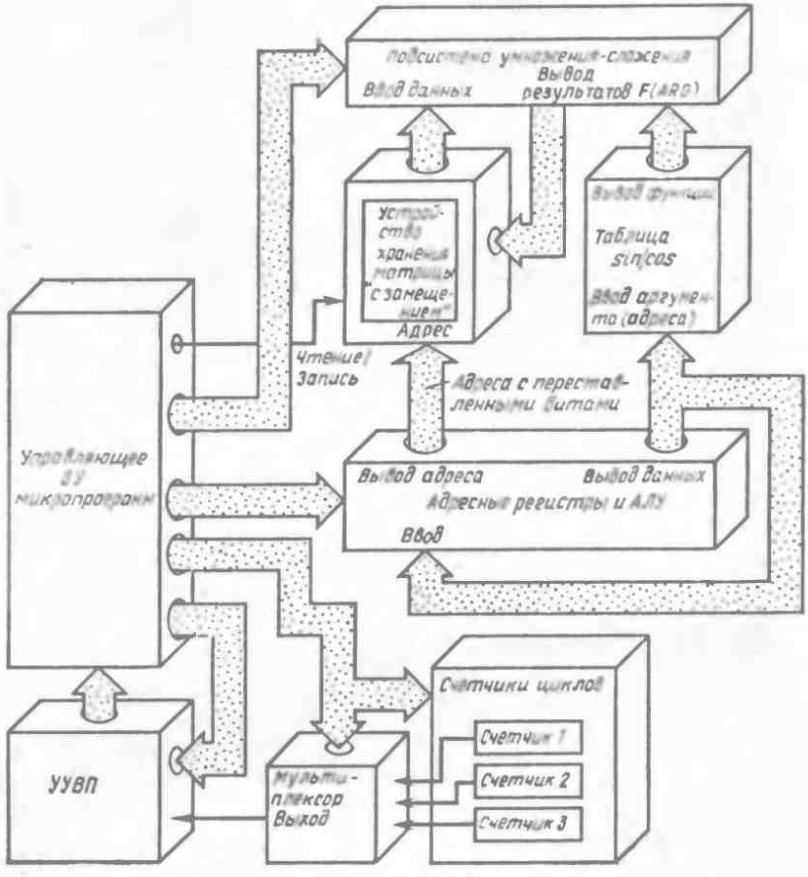


Рис. 6.20. Схема реализации вычисления функций cos/sin для БПФ.

механизм адресации матриц последовательно выдает необходимые данные в схему умножения — сложения, а затем записывает результат в область хранения матриц «с замещением». Этот вариант реализации схемы умножения (рис. 6.19) обеспечивает наиболее быстрое выполнение операции при условии использования быстродействующей памяти произвольного доступа, имеющей такую же высокую тактовую частоту, как и внутренние регистры.

4. Тригонометрические преобразования. Четвертой операцией, выполняемой при БПФ, является вычисление выражений типа

$$ei\,x = \cos x + i \sin x.$$

Эта операция может быть реализована алгоритмическим либо табличным способом. При алгоритмическом способе производится непосредственное вычисление sin или cos от требуемого аргумента, обычно при использовании разложения в ряд. Табличный метод заключается в хранении результатов предварительного вычисления всех требуемых функций в справочной таблице. Таким образом обеспечивается доступ к этим данным за один такт. Поскольку при вычислении путем разложения в ряд производится большое количество операций умножения и сложения, предпочтительнее использовать табличный поиск, а наличие быстродействующих, обладающих большой емкостью ПЗУ делает его реально осуществимым. Схема устройства, реализующего такой подход, приведена на рис. 6.20.

ЗАКЛЮЧЕНИЕ

Данная глава была посвящена разработке проекта, основанного на математическом анализе проблемы. Задача двумерного динамического слежения за изображением представляет собой экстремальный случай анализа, основанного на вычислениях. Аналитические подходы к решению этой задачи хотя и являются весьма сложными, хорошо описаны в литературе, так что разобраться в алгоритме ее решения, принятом в данной главе, не составит большого труда. Как следует из данной главы, возникает необходимость в переводе алгоритма в соответствующую архитектуру устройства. Первым шагом на этом пути является представление алгоритма в виде блок-схемы и анализ процесса вычислений с этой точки зрения. Такого анализа в сочетании со знанием разработчика о том, какие элементарные цифровые устройства ему доступны, достаточно для построения первоначального варианта архитектуры системы. Далее используется метод поэтапного уточнения разработанной схемы. На каждом шаге имеющаяся архитектура все более детализируется до тех пор, пока выявившиеся противоречия или неэффективность решения не заставят проектировщика пересмотреть архитектуру, после чего процесс повторяется на новом уровне. Таким образом, может быть либо разработан весьма эффективный проект, либо проектировщик приобретает опыт, достаточный для того, чтобы начать разработку с нового, более подходящего варианта архитектуры.

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ С ПЛАВАЮЩЕЙ ТОЧКОЙ С МИКРОПРОГРАММНЫМ УПРАВЛЕНИЕМ

В большинстве работ, посвященных микропрограммированию, приводится пример архитектуры ЭВМ, в которую включены устройства с микропрограммным управлением. Эти работы, как правило, ограничиваются рассмотрением простых классических архитектур. В известной степени они вводят читателя в заблуждение, поскольку устройства с микропрограммным управлением и разрядно-модульной организацией применяются главным образом в вычислительных системах, обладающих очень сложной архитектурой. В данной главе рассматривается построение вычислительной системы с плавающей точкой. Выбор этой системы обусловлен тем, что она находит очень широкое применение, а ее сложность позволяет представить весь необходимый материал в пределах одной главы. Данная задача, кроме того, дает возможность продемонстрировать пути достижения высокой степени параллелизма вычислений системы. Соответствующие характеристики аппаратных средств обсуждаются в ходе изложения.

СПОСОБЫ РЕАЛИЗАЦИИ ОПЕРАЦИЙ С ПЛАВАЮЩЕЙ ТОЧКОЙ

У разработчика вычислительной системы с плавающей точкой существует много возможностей для построения системы. Если оставить пока открытым вопрос выбора системы счисления, то приводимые ниже возможные варианты весьма многочисленны.

1. Программный способ. Написание программы вычислений с плавающей точкой является, пожалуй, наиболее простым способом. Как правило, она разрабатывается на языке ассемблера данной ЭВМ. В качестве вариантов при таком подходе можно

указать написание программ на языке высокого уровня или на микрокоде, если система предоставляет такую возможность. Последний вариант по сравнению с первым обеспечивает более высокую скорость выполнения программ. Типичное время выполнения программы, реализующей операцию с плавающей точкой, составляет от микро- до миллисекунд в зависимости от системы команд и быстродействия процессора.

2. Сложные интегральные схемы. В настоящее время на рынке имеется большое количество кристаллов — сверхсложных интегральных схем, предназначенных для вычисления математических функций. Эти схемы первоначально были разработаны для использования в портативных калькуляторах для проведения научных или экономических расчетов. Диапазон реализуемых ими математических функций очень широк и включает преобразование координат, возведение в степень, все прямые и обратные тригонометрические функции, в том числе гиперболические и др. Эти интегральные схемы обычно выполняются на базе *p*-МОП-технологии, что требует использования нестандартных напряжений и применения комплементарных МОП-схем для осуществления интерфейса ТТЛ- и *p*-МОП-схем. Внутренняя организация интегральных схем последовательная, поэтому они обладают относительно низким быстродействием. Обычный диапазон времени вычислений функции составляет 10—100 мс.

3. Интегральные схемы с высоким быстродействием. Второе поколение описанных выше интегральных схем было разработано применительно не к калькуляторам, а к микропроцессорам. Эти схемы, например, такие, как AM9511 фирмы AMD, выполняются по *n*-МОП-технологии, а потому обладают высоким быстродействием и совместимы с ТТЛ-схемами. Хотя им и не свойственны экзотические возможности рассмотренных выше интегральных схем для калькуляторов, достоинством таких схем является возможность прямого сопряжения с портами ввода-вывода микропроцессора и предоставление эффективных средств для повышения вычислительной мощности процессоров типа 8080. Время выполнения операций составляет сотни микросекунд.

4. Сопроцессоры. Представителем третьего поколения вычислительных интегральных схем является сопроцессор 8087 фирмы Intel, функционирующий параллельно с ведущим процессором. Сопроцессор осуществляет перехват кодов потока команд к ведущему процессору и выполняет необходимые операции над данными независимо от него (с соблюдением тем не менее определенной степени согласованности действий).

5. Однокристальные микро-ЭВМ. Вариант использования однокристальных микро-ЭВМ, располагающих сотнями байтов оперативной памяти и несколькими тысячами команд (при на-

пряжении питания 5 В) представляет большой интерес в тех случаях, когда требуется получить вычислительную систему с нестандартными свойствами. При периоде тактовых импульсов, равном 100 нс, эти устройства имеют относительно высокое быстродействие.

6. Устройства с микропрограммным управлением и разрядно-модульной организацией. Эти устройства позволяют получить не только те преимущества, которыми обладают однокристальные микро-ЭВМ, но и более высокую степень параллелизма вычислений.

7. Устройства обработки сигналов. Устройства обработки двоичных сигналов (уменьшенного динамического диапазона) с плавающей точкой предназначены для выполнения с высокой скоростью операций над числами, получаемыми из аналоговых данных, которые генерируются такими устройствами, как радары, медицинские сканирующие приборы и т. п.

СИСТЕМЫ СЧИСЛЕНИЯ И ФУНКЦИИ

Подавляющее большинство вычислений с плавающей точкой выполняется в двоичной системе счисления, т. е. системе счисления по основанию 2. Это связано с наличием простых алгоритмов умножения по основанию 2, состоящих из операций сдвига и сложения, а также с немаловажной экономией памяти, получаемой в результате представления чисел в более компактной двоичной форме, нежели в двоично-десятичном коде (BCD) или в коде ASCII. К недостаткам представления чисел с плавающей точкой в двоичной форме следует отнести неестественность использования данной системы счисления людьми, поскольку они привыкли к десятеричной системе, а также необходимость преобразования двоичного кода в код ASCII для вывода данных на дисплей и печатающие устройства.

Маловероятно, что двоичное представление чисел с плавающей точкой будет когда-либо желательным для пользователя. В то же время удвоение количества информации, хранимой в единице объема запоминающих устройств, каждые несколько лет постоянно снижает значение второго из указанных достоинств этого формата. Поэтому становится очевидным, что использование кода ASCII как для внутреннего представления чисел, так и для ввода-вывода будет все более расширяться. По этим причинам для описываемого в данной главе проекта системы с плавающей точкой было выбрано представление чисел в коде ASCII.

МЕТОД ДОСТУПА К ДАННЫМ

При представлении чисел в коде ASCII более эффективным является не параллельный метод доступа к данным, а последовательный — по байтам. Для достижения большей универсальности система будет разрабатываться для асинхронного интерфейса с ведущим процессором.

СИНТАКТИЧЕСКИЙ АНАЛИЗ В ОПЕРАЦИЯХ С ПЛАВАЮЩЕЙ ТОЧКОЙ

Выражение, подаваемое на вход системы, выполняющей операции с плавающей точкой, состоит из операндов и операторов. Расположение элементов этого выражения может быть самым различным. Для простоты положим, что допустимой является следующая последовательность его элементов:

ОПЕРАНД1 ОПЕРАТОР ОПЕРАНД2

Числовой результат должен выдаваться в одном из стандартных форматов, по выбору пользователя. Код, в котором представлено число, был определен. Операнд должен удовлетворять единственному требованию — содержать по крайней мере один разряд. Операторы изображают с помощью символов +, —, * и / кода ASCII. Учитывая, что два операнда в одном из допустимых форматов и четыре оператора могут образовывать различные комбинации, необходимо производить анализ входной строки. Такой анализ обычно называют *синтаксическим.*

Ниже будет показана реализация операций сложения, умножения и вычитания и обсуждены некоторые вопросы, связанные с операциями деления и сравнения. Представление чисел с плавающей точкой должно соответствовать следующему формату:

$$\pm n.mE \pm pq$$

Примеры правильно представленных чисел:

3 3. —3.159 +3.024 —3.79E—24

На окончание числа указывает символ пробела (20H) в коде ASCII или допустимого оператора: +, * и т. д.

ПРОЦЕДУРА СИНТАКСИЧЕСКОГО АНАЛИЗА

Синтаксический анализ входной строки включает следующие шаги:

1. Установку начального состояния.

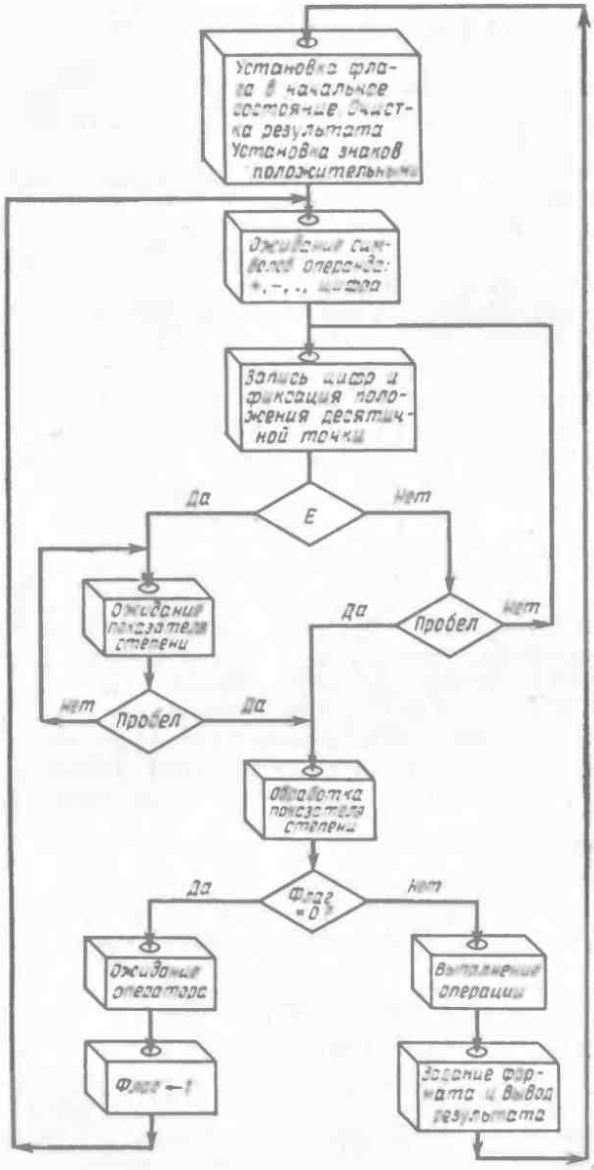


Рис. 7.1. Блок-схема алгоритма синтаксического анализа для вычислений с плавающей точкой.

2. Ожидание символа первого операнда.
3. Запись цифр и фиксацию местоположения точки.
4. Обработку показателя степени, если он имеется.
5. Ожидание символа оператора.
6. Ожидание символа второго операнда.
7. Запись цифр и фиксацию местоположения точки.
8. Обработку показателя степени, если он имеется.
9. Выполнение операции.
10. Представление результата в нужном формате и его вывод.

Блок-схема этой процедуры представлена на рис. 7.1. Необходимо отметить, что флаг первоначально устанавливается в 0 для индикации первого операнда и в 1 после выявления оператора для указания перехода ко второму операнду. Значение этого флага проверяется для определения наличия обоих операндов и возможности начала вычислений.

ПРЕДСТАВЛЕНИЕ ПОДСИСТЕМЫ С ПЛАВАЮЩЕЙ ТОЧКОЙ В ВИДЕ МАШИНЫ СОСТОЯНИЙ

Функционирование подсистемы, выполняющей операции с плавающей точкой, может быть легко описано с привлечением понятий машины состояний (рис. 7.2). Изучение представленной на рисунке машины состояний показывает, что среди прочих она имеет несколько состояний «Ожидание», когда система ждет получения следующей порции информации, необходимой для перевода ее в другое состояние. Строка операндов и оператор в коде ASCII вводятся асинхронно побайтно, а следовательно, необходимы средства, обеспечивающие прием этих байтов (т. е. средства синхронизации подсистемы с плавающей точкой и ведущей машины). Ниже рассматривается способ реализации синхронизирующей машины состояний с микропрограммным управлением, которая может быть использована для создания соответствующих состояний ожидания. Допустим, что ведущая машина генерирует синхронизирующий сигнал f, инициирующий переход вспомогательной системы в другое состояние. Вид сигнала и процесс перехода вспомогательной системы из одного состояния в другое показаны на рис. 7.3.

Устройство управления выполнением программы 3001 (рис. 7.4) имеет сигнальную линию ввода, которая запирается перед окончанием каждого микроцикла. Для определения следующего состояния машины передаваемые в данный момент по шине данные могут быть проверены с использованием одной из функций управления адресом, реализованных в устройстве 3001.

С помощью устройства 3001 синхронизация осуществляется следующим образом. При поступлении переднего фронта син-

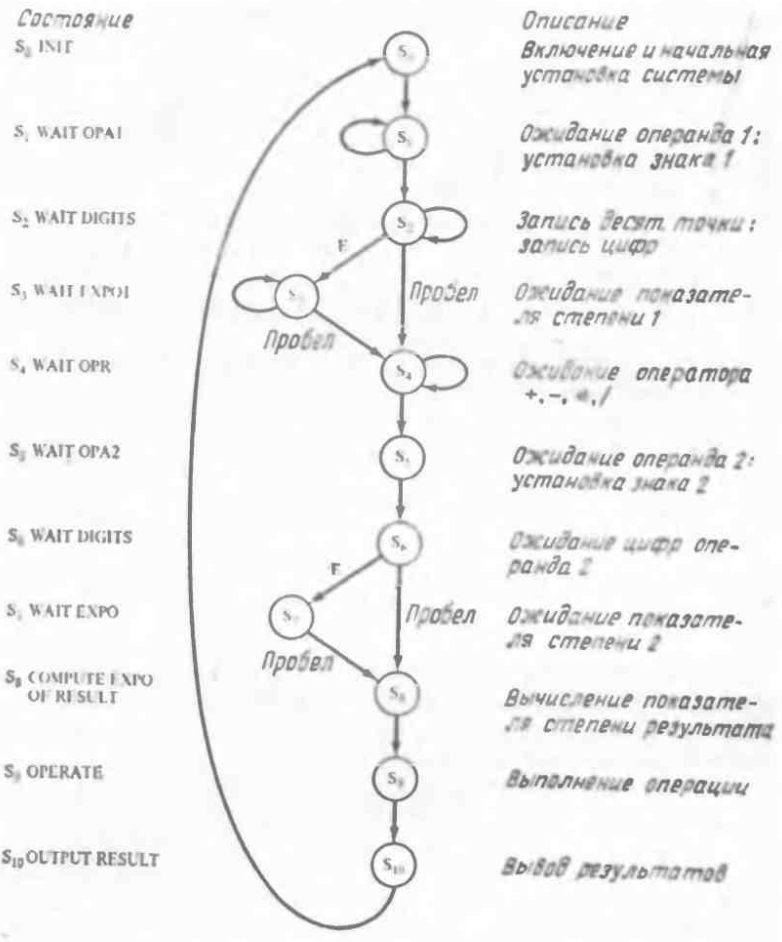


Рис. 7.2. Представление подсистемы, выполняющей операции с плавающей точкой, в виде машины состояний.

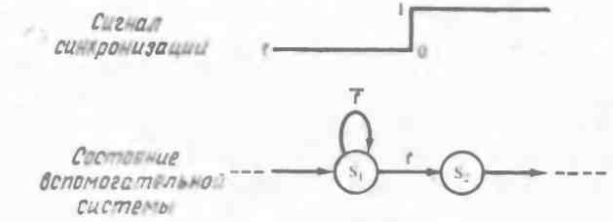


Рис. 7.3. Сигнал синхронизации и синхронизирующая машина состояний.

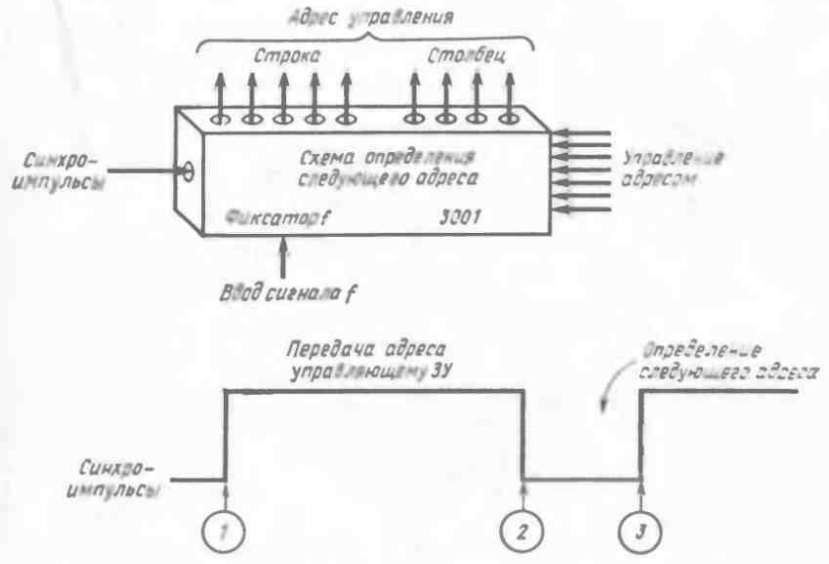


Рис. 7.4. Ввод флага в устройство 3001 и временные соотношения.

хроимпульса производится загрузка адреса управления в регистр адреса микропрограммы устройства 3001. По этому адресу из управляющей памяти выбирается управляющее слово и производится соответствующая операция. Адрес следующего управляющего слова определяется функцией управления адресом, реализованной в виде обратной связи (рис. 7.5), и может также зависеть от входов сигналов условий, таких, как вход f.

Длительность синхроимпульса определяется величиной внешних задержек в управляемой подсистеме. Должен быть произведен анализ величин этих задержек в каждой конкретной ситуации и выбран наихудший случай. Синхронизация, построенная в расчете на худший случай, обеспечит разрешение выполняемых в управляемой подсистеме действий к моменту времени 2 (рис. 7.4). В этот момент поступает задний фронт импульса синхронизации и сигнал на входе f сохраняется в фиксаторе f устройства 3001. Устройство 3001 производит выбор следующего адреса управления с помощью функции управления адресом и сигналом условий. Продолжительность этого периода определения следующего адреса составляет $\sim$30 нс, после чего происходит переключение уровня синхросигнала с низкого на высокий. Это переключение приводит к загрузке адреса управления в регистр адреса микропрограммы (момент времени 3), и начинается новый цикл.

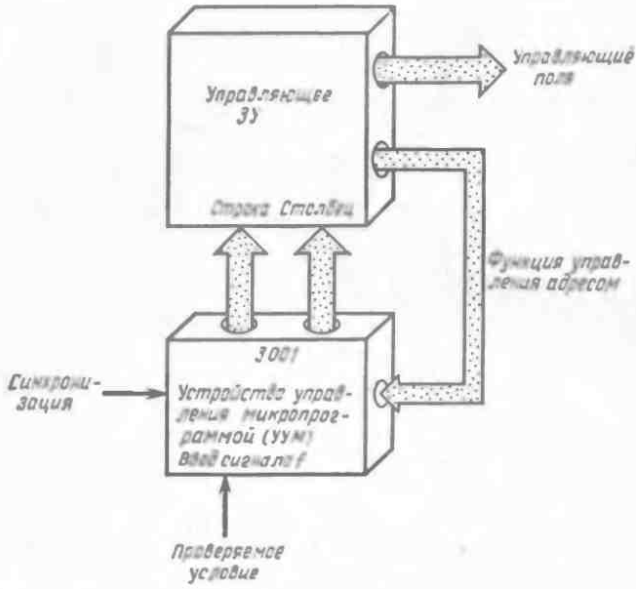


Рис. 7.5. Ввод синхроимпульсов и флагов в УУВП.

УСТРОЙСТВО УПРАВЛЕНИЯ ВЫПОЛНЕНИЕМ ПРОГРАММЫ 3001

ПРОВЕРКА ФИКСАТОРА

В гл. 5 указывалось, что в УУВП 3001, задающем последовательность управления микропрограммой, используется структура адресации типа Мили, при которой каждое управляющее слово содержит поле следующего адреса. Наличие такого поля исключает необходимость в счетчике микрокоманд и, следовательно, меняет последовательный характер адресного пространства.

Из 9 бит адреса управления 5 бит интерпретируются как адрес строки, а 4 бит — как адрес столбца в двумерном адресном пространстве, являющемся отличительной особенностью устройства 3001. Каждая из функций управления адресом задается как соответствующее преобразование 9-битового адреса. Функция управления, осуществляющая проверку фиксатора f для

преобразования следующего адреса по условию, представляется следующим образом:

JFL

Переход по состоянию фиксатора f

КОП Смещение

1	0	0	d_3	d_2	d_1	d_0

	Строка					Столбец			
Текущий адрес:	m_8	m_7	m_6	m_5	m_4	m_3	m_2	m_1	m_0
Следующий адрес:	m_8	d_3	d_2	d_1	d_0	m_3	0	1	(f)

Из определения функции следует, что значение старшего бита в адресе строки сохраняется, таким образом 32-строчное пространство разделяется на два подпространства: 0xxxx и 1xxxx, каждое из которых содержит 16 строк. При этом следующий адрес, формируемый функцией управления адресом JFL, будет всегда находиться в том же подпространстве строк, что и текущее управляющее слово.

Функция управления JCC (переход в пределах столбца) позволяет осуществлять управление во всем 32-строчном пространстве. Смещение, содержащееся в команде JFL, замещает четыре младших бита текущего адреса микропрограммы, поэтому в качестве следующего адреса строки выбирается одна из 16 строк текущего подпространства. Подпространства строк показаны на рис. 7.6, *а*.

Воздействие функции JFL на пространство столбцов такое же, как и на пространство строк. Вновь значение старшего бита адреса столбца сохраняется неизменным, что приводит к разделению пространства столбцов на два подпространства (рис. 7.6, *б*). Младшие биты адреса столбца обрабатываются по-раз-

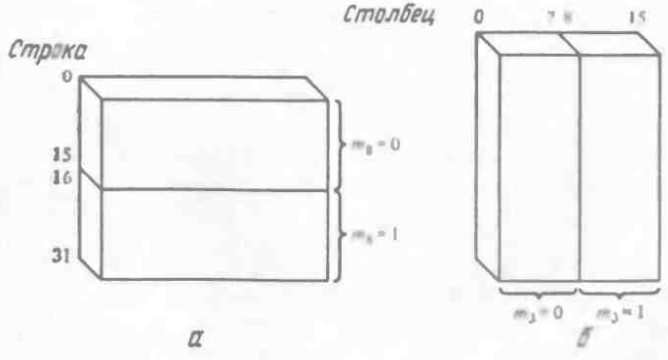


Рис. 7.6. Подпространства строк (*а*) и столбцов (*б*), соответствующие функции управления адресом JFL устройства 3001.

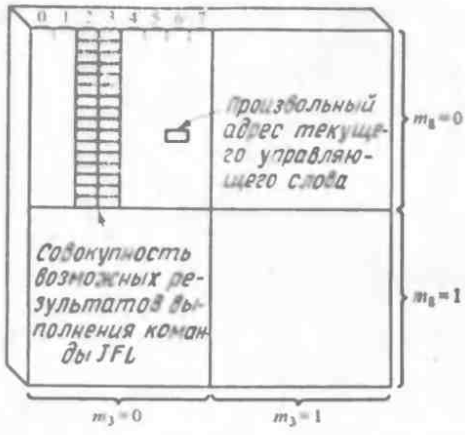


Рис. 7.7. Совокупность адресов, получаемая в результате обработки адреса 0xxxx0xxx по команде JFL.

ному. Значение двух следующих по старшинству битов устанавливается равным 01, следовательно, следующий адрес столбца определяется выражением $m_3$01x. Допустим, что $m_3=0$. Адрес столбца становится равным 001x (т. е. 0010 или 0011). Значение младшего бита определяется уровнем выходного сигнала фиксатора f в тот момент, когда синхронизатор переключает уровень сигнала с высокого на низкий. Таким образом, при $m_3=0$ следующий адрес столбца равен 2 (0010), если уровень сигнала на входе f низкий, и 3 (0011), если высокий. Очевидно, что при $m_3=1$ адресом столбца будет число 1010 или 1011 соответственно.

При рассмотрении функции JFL необходимо обратить внимание на следующее. Во-первых, следует отметить, что подпространства строк и столбцов задаются таким образом, что следующий адрес столбца является функцией входного сигнала, подаваемого на фиксатор f, в то время как следующий адрес строки определяется смещением $d_3d_2d_1d_0$, которое обычно назначается программистом. В дальнейшем будет показано, что подобная передача управления характерна для устройства 3001, задающего последовательность управления: пространство столбцов находится под управлением внешних условий, а пространство строк — под контролем программиста. Необходимо, однако, отметить, что значение битов смещения предпочтительнее задавать посредством внешних схем, а не с помощью обратной связи от поля управления адресом.

Вторым важным моментом, касающимся не только команды JFL, но и большинства других команд устройства 3001, является тот факт, что один или два старших бита в адресах

как строк, так и столбцов могут сохранять свои значения неизменными. Таким образом, эти пространства делятся на подпространства. Это дает возможность программисту, отчетливо представляя себе последствия использования команды в одном из этих подпространств, получать результаты ее выполнения в других подпространствах простым наложением. Наличие такой возможности вытекает из рассмотрения рис. 7.7, на котором показана область значений результатов выполнения команды JFL.

ФЛАГИ C И Z

В предыдущем разделе подробно описывалось использование фиксатора f для проверки текущих условий. Помимо этого фиксатора в устройстве имеются два внутренних фиксатора C и Z, используемых для хранения значения вводимых условий. Поскольку синхроимпульс подается на фиксатор f в каждом цикле, он всегда содержит текущую информацию. На фиксаторы C и Z синхроимпульсы подаются только по команде, так что они могут хранить информацию в течение произвольного чис-

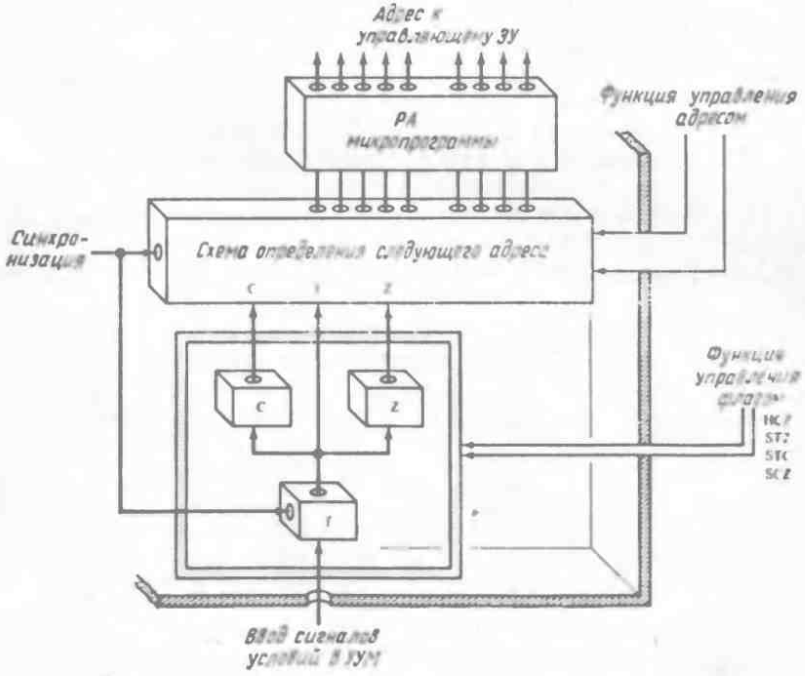
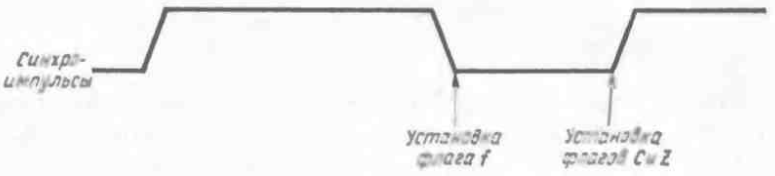


Рис. 7.8. Схема использования флагов в устройстве 3001, задающем последовательность выполнения программы.

ла циклов синхронизации. Назначение этих фиксаторов (флагов) связано с тем, что они, как правило, используются для проверки информации об арифметическом переносе (Carry) и нулевом результате (Zero). Схема с использованием этих фиксаторов показана на рис. 7.8.

Диаграмма синхронизации для устройства 3001 имеет следующий вид:



Из этой диаграммы следует, что флаги C и Z устанавливаются по окончании текущего цикла. Таким образом, флаги C и Z никогда не будут содержать текущую информацию. Их содержимое всегда будет представлять собой состояние линий ввода условий в предыдущем такте или ранее. Как оба, так и каждый из флагов C и Z могут быть установлены в 1 (или в 0) в соответствии с управляющим полем управляющего слова. Соответствующие команды имеют следующие мнемонические обозначения:

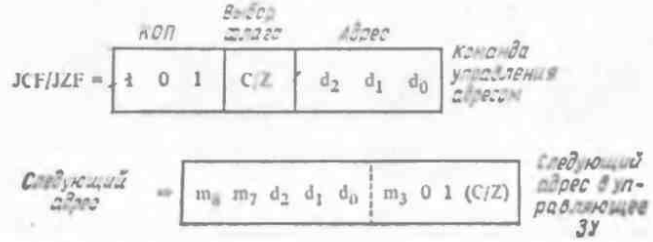
HCZ сохранение значений флагов C и Z без изменений;

STC установка в 1 флага C в соответствии с текущим вводом;

STZ установка в 1 флага Z в соответствии с текущим вводом;

SCZ установка флагов C и Z в соответствии с текущим вводом.

Значение фиксатора f устанавливается в каждом синхронизирующем цикле и поэтому не имеет управляющих линий, связанных с этой операцией. Две команды управления адресом JZF и JCF, осуществляющие проверку флагов C и Z, определяются следующим образом:



В команде управления адресом бит выбора флага используется для указания того, значение какого флага — C или Z — должно быть проверено. Значение флага обозначается соответственно как (C/Z). Читателю рекомендуется оценить размеры допустимого пространства переходов, связанного с пространствами адресов строк и столбцов. Сравнение с командой управления адресом JFL показывает, что все команды ветвления на два направления (JFL, JCF и JZF) реализуют переход на столбец 2, если результатом проверки является значение «ложно», и на столбец 3, если значение «истинно» (или 10 и 11 при $m_3=1$). Диапазон адресов строк, в котором возможен переход, для команд JCZ и JZF уменьшен. Однако выполнение этих команд по существу ничем не отличается от выполнения других команд с учетом ограничений, накладываемых на пространство строк. Поэтому совокупность адресов перехода для команд JZF, JCF и JFL аналогична приведенной на рис. 7.7.

СИНХРОНИЗАЦИЯ С ИСПОЛЬЗОВАНИЕМ ФИКСАТОРА f

Команда JFL, рассмотренная в предыдущем разделе, обеспечивает структуру управления, необходимую для синхронизации устройства, задающего последовательность выполнения команд, внешним событием. Для иллюстрации происходящего процесса изменим рис. 7.3, добавив к нему схемы адресов, применяемые в устройстве 3001 (рис. 7.9).

Команда JFL (S_1, S_2), размещенная в строке m, столбце 2, получает смещение $d_3d_2d_1d_0=m$ в текущем подпространстве строк, поэтому управление всегда возвращается в строку m, столбец 2 или 3. При низком уровне сигнала f управление возвращается в строку m, столбец 2. Как только уровень сигнала f

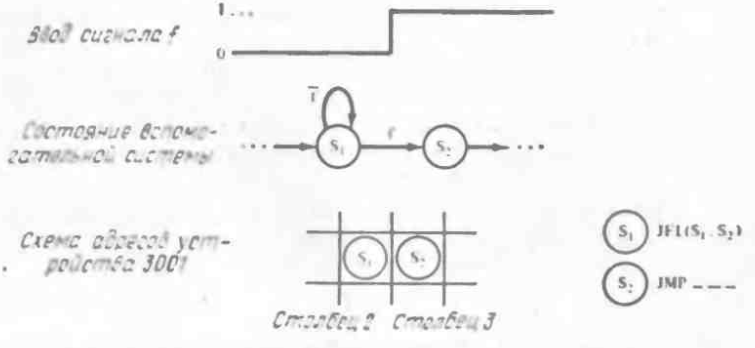


Рис. 7.9. Механизм синхронизации на базе устройства 3001.

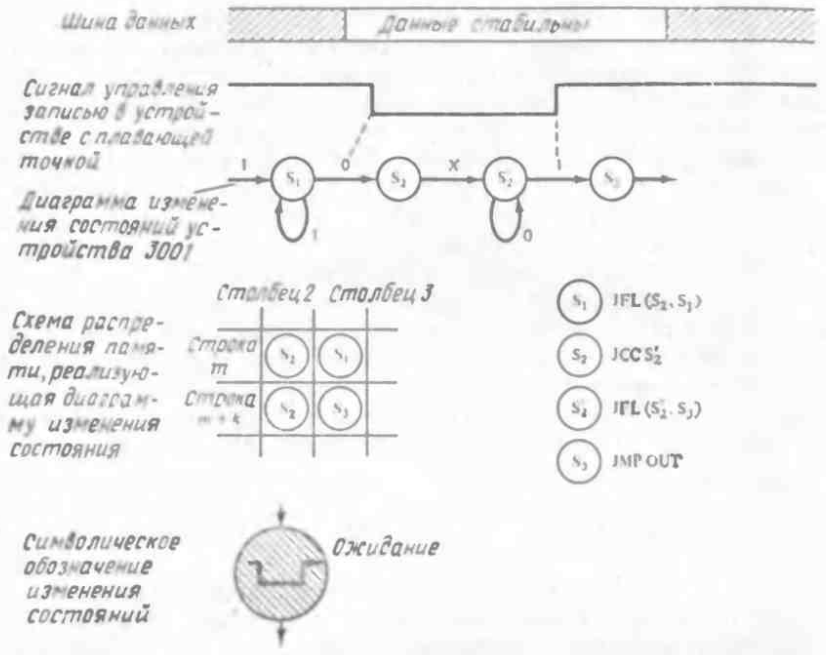


Рис. 7.10. Схемы изменения состояния при использовании синхронизирующей машиной состояний сигнала «Запись».

становится высоким, в следующем такте происходит переход к строке m, столбец 3. Таким образом осуществляется переход из состояния S_1 в состояние S_2.

ПРОЦЕДУРА ПЕРЕДАЧИ ДАННЫХ

Механизм синхронизации, описанный в предыдущих разделах, рассматривался как средство реализации асинхронного побайтного последовательного метода доступа. В этом разделе рассматривается случай, при котором ведущий процессор стробирует каждый байт данных, а также метод установления связи, при котором подтверждаются как передача, так и прием сигнала.

Необходимое условие согласования во времени для синхронизации на микропрограммном уровне определяется неравенством

$$t_d > t_c,$$

где t_d — минимальное время между двумя последовательными изменениями входного сигнала, а t_c — период синхроимпульсов

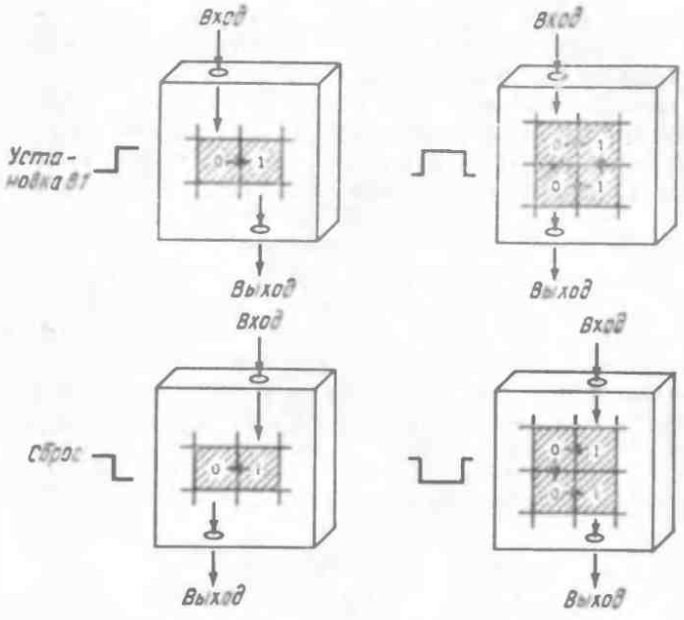


Рис. 7.11. Схемы реализации R—S- и J—K-триггеров с помощью команды JFL.

микроцикла. Согласно используемому методу, имеет место условие

$$t_d > 2t_c.$$

Рассмотрим вначале случай, когда ведущий микропроцессор передает байт данных системе с плавающей точкой посредством подачи отрицательного сигнала на управляющую линию «Запись» (рис. 7.10). Изучение временно́й диаграммы показывает, что команда JFL (0,1) позволяет эффективно реализовать лю-

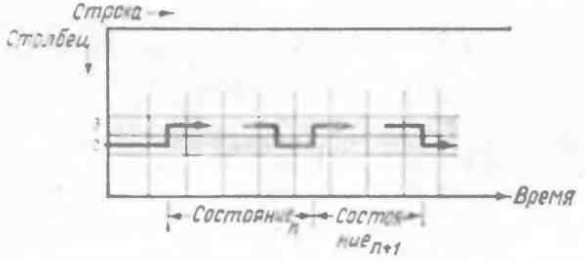


Рис. 7.12. Схема строк и столбцов, повернутая на 90°, дает наглядное представление о наличии связи между временными соотношениями и синхронизатором, использующим команду JFL.

бой желаемый метод синхронизации. Примеры такой реализации приведены на рис. 7.11. Здесь же показано прохождение управляющих сигналов через синхронизатор.

Хотя, как правило, в данной работе и выполняется соглашение фирмы Intel о расположении строк и столбцов, их расположение на схеме рис. 7.12 изменено на противоположное для облегчения понимания читателем излагаемого материала.

ВЕТВЛЕНИЕ НА n НАПРАВЛЕНИЙ

Способность осуществлять ветвление позволяет устройству 3001 выполнять параллельно многие проверки, которые обычно осуществляются последовательно. Выше рассматривалось бинарное ветвление, или ветвление на два направления. Было показано, что текущее условие может быть проверено с помощью команд JZF и JCF. Рассмотрим теперь ветвление на n направлений при n больше 2.

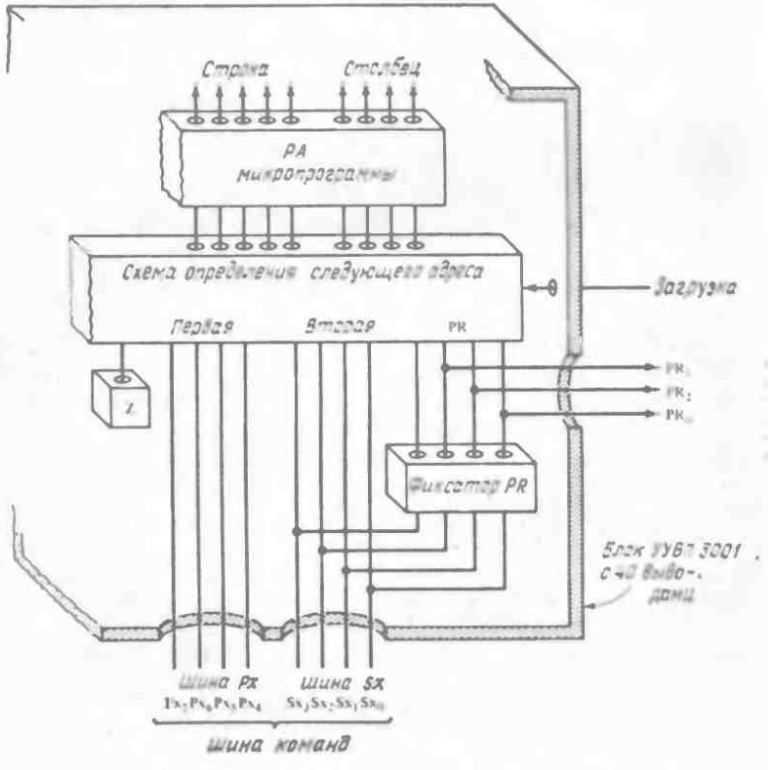


Рис. 7.13. Шина команд устройства 3001, задающего последовательность выполнения микропрограммы.

Для записи условия ветвления в предельном случае может использоваться 8 бит, что позволяет производить переход на 256 направлений. Эти биты помещаются на шину команд, состоящую из 8 линий и являющуюся входной для логической схемы определения следующего адреса. Ветвление на 256 направлений осуществляется не посредством команды, как в других случаях, а путем активации линии «Загрузка», доступной извне через один из 40 выводов блока. Когда на этот вывод подается напряжение, данные с шины команд синхронно загружаются в восемь младших разрядов регистра адреса микропрограммы, осуществляя тем самым выбор одного из 256 управляющих слов. Линия «Загрузка», конечно, может управляться и с помощью специального поля управляющего слова. Ветвление на 256 направлений обычно используется при эмуляции другой ЭВМ с 8-битовыми кодами команд. При таких применениях устройство отображения программируемой логической матрицы или постоянного запоминающего устройства обычно преобразует коды операций в соответствии с подходящей организацией памяти.

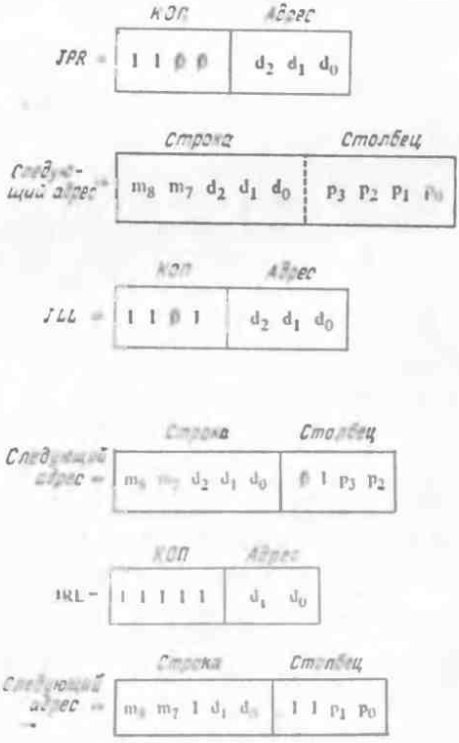
ШИНЫ КОМАНД

Хотя 8-разрядная шина команд недоступна для команды управления адресом, она может быть организована как две 4-разрядные шины. В этом случае обе шины становятся доступными для таких команд. К первой шине имеется прямой доступ, а ко второй — косвенный через 4-разрядный фиксатор, называемый *программным* или *фиксатором PR* (рис. 7.13). Фиксатор PR сохраняет данные, передаваемые по второй шине, во время проведения проверки первой шины. Все четыре вывода фиксатора PR также могут подвергаться проверке; кроме того, три из них имеют отдельные контактные штеккеры. Команды, используемые для проверки первой и второй шин здесь не рассматриваются.

Первая шина проверяется с помощью команды JPX (Переход по состоянию первой шины). По этой команде определение адреса производится следующим образом:

Изменение адреса строки по этой команде выполняется аналогично изменению адреса по другим командам условного перехода. Однако при этом используются лишь два разряда из адресного пространства, предоставляемого пользователю. Результат выполнения команды JPX над адресом столбца определяется довольно просто. Текущие данные, находящиеся на первой шине Px_7—Px_4, помещаются в разряды столбца регистра адреса микропрограммы. Эти разряды непосредственно определяют адрес одного из 16 столбцов, позволяя тем самым осуществлять ветвление на 16 направлений. По команде JPX производится также подача синхроимпульса на фиксатор PR, что приводит к сохранению данных, находящихся на второй шине. Эти данные могут быть проверены с помощью нескольких команд перехода, которые описываются ниже.

Ниже приведены форматы и результаты выполнения трех команд, осуществляющих проверку данных, сохраняемых в фиксаторе PR.



Команда JPR использует все четыре разряда, сохраняемые в фиксаторе PR, для выполнения ветвления на 16 направлений, в

то время как команда JLL осуществляет проверку только двух левых разрядов фиксатора p_3 и p_2, а команда JRL — только двух правых разрядов p_1 и p_0. Вследствие этого две последние команды производят ветвление на четыре направления, выбирая в качестве следующего адреса один из четырех столбцов. Команда JLL позволяет выполнять передачу управления одному из четырех столбцов с номерами 4—7, а команды JRL — одному из столбцов с номерами 12—15.

Все проверки фиксатора PR являются неразрушающими в том смысле, что они не изменяют содержимого фиксатора PR, за исключением команды JPX. Введение этих команд управления адресом, являющихся вполне удовлетворительным средством организации ветвления, дает возможность успешно завершить проектирование системы с плавающей точкой.

ДЕКОДИРОВАНИЕ СИМВОЛОВ КОДА ASCII

Представление чисел в коде ASCII было выбрано для системы с плавающей точкой в связи с тем, что этот код позволяет осуществлять вывод информации на дисплей или печатающее устройство в виде, удобном для человека, без дополнительных

Таблица 7.1. Символы кода ASCII, используемые в вычислительной системе с плавающей точкой, и их значение

Символ ASCII	Шестнадцатеричное представление	Значение символа
Пробел	20	Ограничитель
+	2B	Положительный знак или оператор сложения
—	2D	Отрицательный знак или оператор вычитания
*	2A	Оператор умножения
/	2F	Оператор деления
.	2E	Десятичная точка
$	24	Оператор извлечения квадратного корня
E	45	Префикс показателя степени
0	30	Цифра 0
1	31	Цифра 1
.	.	.
.	.	.
.	.	.
8	38	Цифра 8
9	39	Цифра 9

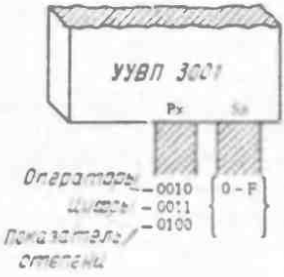


Рис. 7.14. Использование шины команд устройства 3001 для распознавания символов кода ASCII.

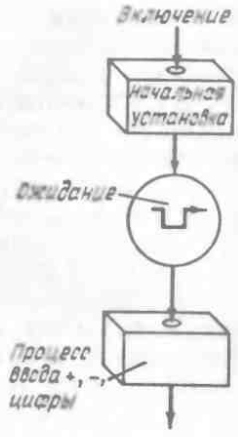


Рис. 7.15. Первые шаги синтаксического анализа для вычислений с плавающей точкой.

преобразований. В результате такого выбора встала проблема разработки системы, производящей обработку информации в коде ASCII. Единственная спецификация, которая была нами определена, относится к синтаксическому анализу входной строки символов в коде ASCII. Синтаксический анализ является одной из форм декодирования, поэтому ниже изучаются символы кода с целью поиска методов эффективной реализации декодирования (табл. 7.1).

Необходимо иметь в виду, что символы кода ASCII понятны лишь человеку. Что касается процессора с плавающей точкой, то он воспринимает наборы нулей и единиц, а не числа в шестнадцатеричном коде. Анализ шестнадцатеричного кода в табл. 7.1 показывает, что значение старшего шестнадцатеричного разряда является разным для операторов и для чисел. Рассматривая такое отличие в символах в качестве ключа для осуществления декодирования, изучим возможность использования шины команд устройства 3001 для распознавания символов кода ASCII (рис. 7.14). Нетрудно заметить, что по команде JPX будет происходить передача управления столбцу 2, если символ является оператором, столбцу 3, если символ является цифрой, и столбцу 4, если обработке подвергнут префикс «E» показателя степени. Таким образом, в результате выполнения команды JPX будет произведено эффективное декодирование каждого символа ASCII, помещенного на шину команд ведущим процессором. Использование этого механизма позволит завершить рассматриваемый проект процессора с плавающей точкой.

ИСПОЛЬЗОВАНИЕ ВЕТВЛЕНИЯ НА n НАПРАВЛЕНИЙ ДЛЯ СИНТАКСИЧЕСКОГО АНАЛИЗА

Строка символов, переданная от ведущего процессора процессору с плавающей точкой, подвергается синтаксическому анализу в соответствии с алгоритмом, блок-схема которого представлена на рис. 7.1. Несколько первых шагов этого процесса изображены на рис. 7.15.

Начальная установка системы происходит при подаче на нее напряжения питания, после чего система находится в ожидании строба «Запись» от ведущего процессора, указывающего на то, что первый байт подан на вход системы. В соответствии с предыдущими рассуждениями этим входом должна быть 8-разрядная шина команд устройства 3001. В этом случае выход из состояния «Ожидание» производится при выполнении команды JPX, заключающемся в выборе одного из 16 направлений, обеспечивающих декодирование символа кода ASCII.

Если набор допустимых символов ограничивается символами, представленными в табл. 7.1, то допустимыми областями передачи управления по команде JPX являются столбцы 2—4. Поскольку первым символом входной строки не может быть символ «Ё», переход к столбцу 4 должен расцениваться как наличие ошибки. Если по шине Px передается число 3, то входной символ является цифрой в коде ASCII и его следует сохранить. Таким образом, передача управления столбцу 3 приведет к инициированию процедуры записи цифры в память. Наконец, переход к столбцу 2 потребует дальнейшего декодирования, т. е. определения того, каким конкретно оператором является данный символ. Напомним, что по команде JPX данные с первой шины загружаются в регистр адреса столбца, а данные со второй шины помещаются в фиксатор PR. Следовательно, определение того, каким именно является данный оператор, может быть выполнено лишь в результате анализа содержимого фиксатора PR. Эта операция осуществляется с помощью команды JPR, по которой содержимое фиксатора PR загружается в регистр адреса столбца, обеспечивая тем самым переход к соответствующему столбцу для каждого из допустимых операторов. Напомним также, что строка, на которую производится переход, управляется программистом. Представим теперь алгоритм синтаксического анализа (рис. 7.1) несколько в другом виде (рис. 7.16).

Как следует из рис. 7.16, первый символ, получаемый процессором с плавающей точкой, декодируется с помощью команды JPX, осуществляющей ветвление на 16 направлений, в результате чего обеспечивается передача управления столбцам 2—4 выбранной строки. Отметим, что на данном этапе передача управления любым другим столбцам этой строки расценивается как ошибочная ситуация. При передаче управления столбцам 3

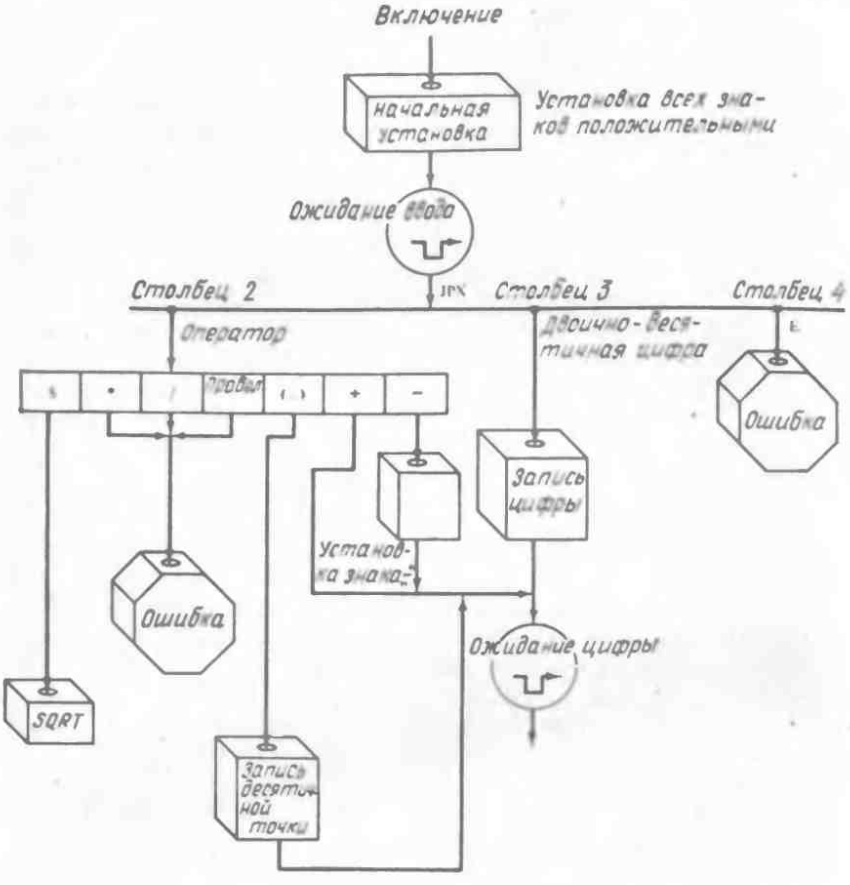


Рис. 7.16. Начальная часть блок-схемы алгоритма синтаксического анализа для вычислений с плавающей точкой.

или 4 начинается выполнение соответствующих процедур. Если же управление передано столбцу 2, то требуется дальнейшее декодирование, выполняемое посредством команды JPR, осуществляющей ветвление на 16 направлений. Четыре младших бита оператора в коде ASCII, передаваемого по шине команд, используются для адресации столбца, содержащего соответствующую обрабатывающую программу. Поскольку адресация строк осуществляется программистом, читателю рекомендуется еще раз обратиться к рассмотрению команды JPR, чтобы определить пределы изменения адреса строки.

Из рис. 7.16 следует, что передача управления столбцам, соответствующим символам *, / и «Пробел» кода ASCII, приводит к ошибочной ситуации, поскольку недопустимо их появле-

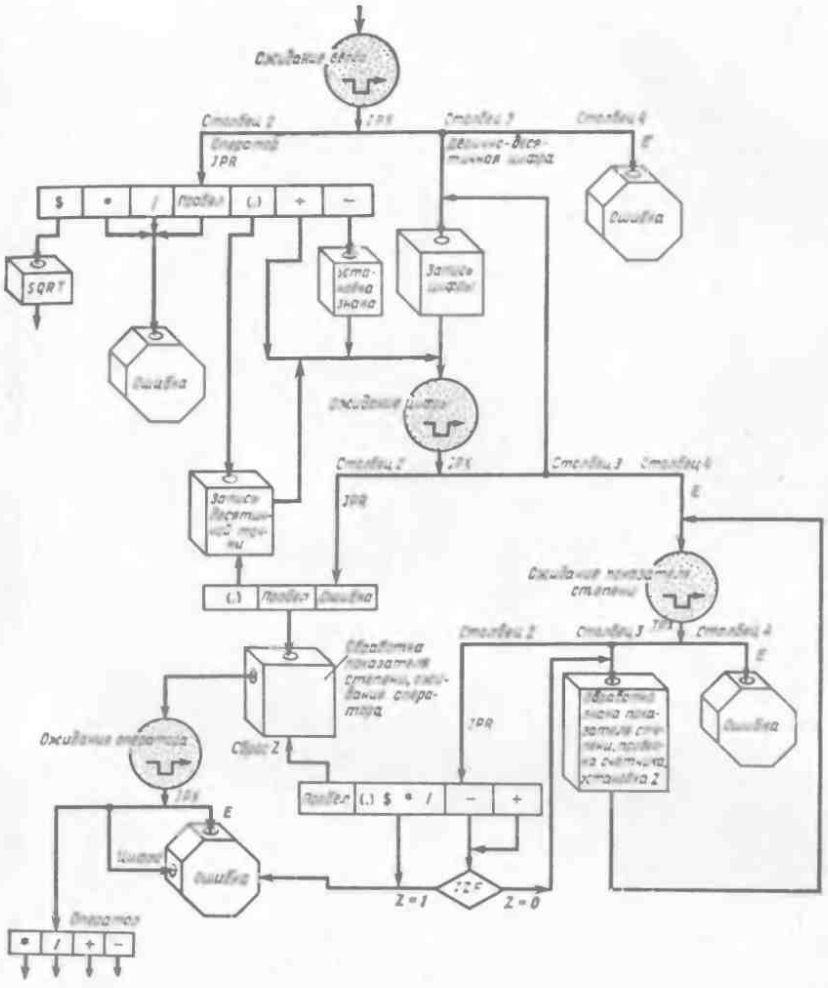


Рис. 7.17. Блок-схема алгоритма синтаксического анализа (обладающего высокой степенью параллельности) первого вводимого в процессор с плавающей точкой операнда.

ние в качестве первых символов строки. Столбец, которому передается управление при приеме символа «+», обеспечивает немедленный переход в состояние ожидания следующего символа, так как первоначально все знаки устанавливаются положительными. Если принятым символом является «—», знак первого операнда устанавливается отрицательным, и система переводится в состояние ожидания следующего символа. При прие-

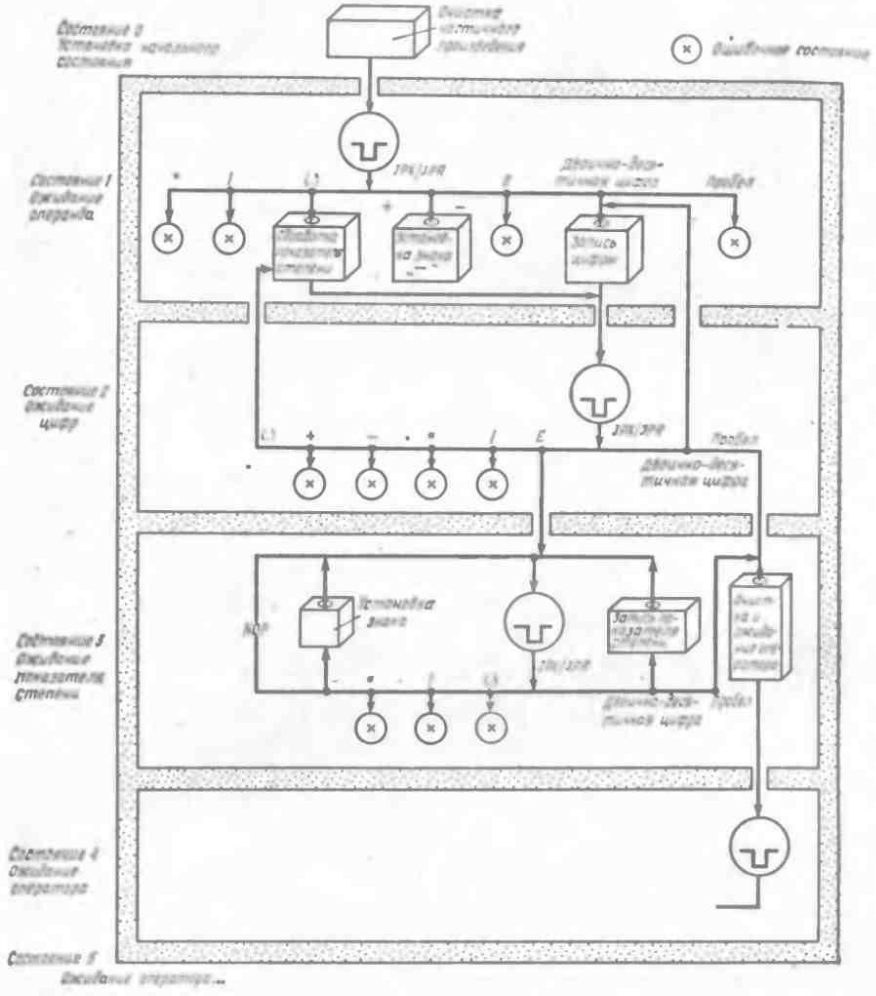


Рис. 7.18. Соответствие шагов алгоритма синтаксического анализа различным состояниям машины состояний.

ме символа «.» фиксируется позиция десятичной точки, после чего ожидается появление следующего символа. Продолжение синтаксического анализа первого операнда показано на рис. 7.17.

Показанный на рис. 7.17 алгоритм синтаксического анализа может быть представлен в виде, более наглядно отображающем возможность его формализации как машины состояний (рис. 7.18). В обоих случаях требуется, чтобы число с плавающей точкой оканчивалось символом пробела.

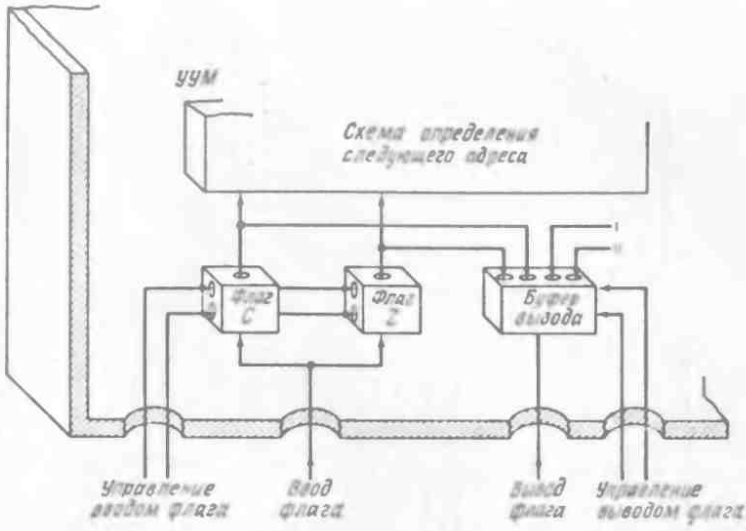
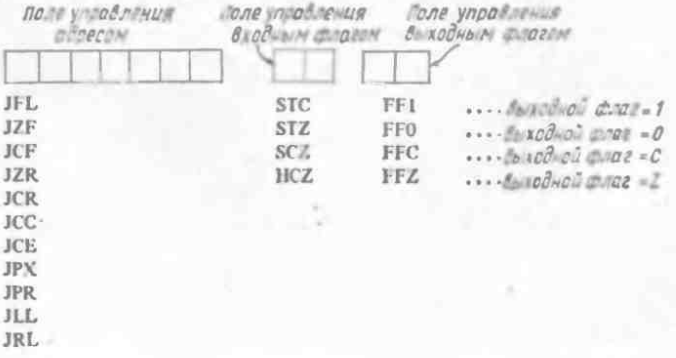


Рис. 7.19. Схема управления флагами в УУВП 3001.

ПОЛЯ УПРАВЛЕНИЯ ФЛАГАМИ

Рассмотрим еще несколько вопросов, касающихся УУВП, в частности управления выходным флагом. Этот флаг обеспечивает возможность доступа внешних устройств к содержимому флагов C и Z. Он также может быть установлен или сброшен. Мультиплексору 4×1, показанному на рис. 7.19, требуются две управляющие линии для выбора одного из четырех входных состояний: C, Z, 1, 0. Две выходные и две входные управляющие линии флагов могут рассматриваться как специальные управляющие поля, связанные с управляющим словом устройства 3001.



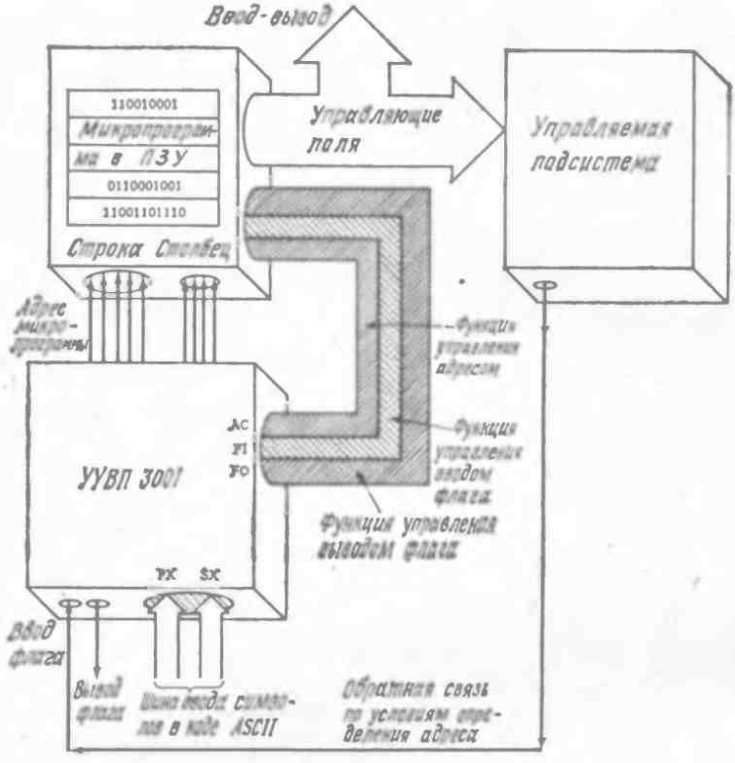


Рис. 7.20. Схема процессора с плавающей точкой.

Функция управления адресом, которая по линии обратной связи передается из ПЗУ в УУВП, используется для определения адреса следующего управляющего слова. Следующий адрес иногда зависит от значений флагов C и Z или содержимого фиксатора f, поэтому разряды управления вводом флагов могут рассматриваться как некоторое расширение функции управления адресом и по линии обратной связи также передаваться от ПЗУ в УУВП. Сигналы управления выводом флагов также могут быть переданы по линии обратной связи из соответствующего управляющего поля (рис. 7.20). Функция управления адресом позволяет разработчику указывать адрес следующего состояния вне зависимости от каких-либо условий или осуществлять управление потоком адресов посредством фиксатора f в соответствии с внешними условиями. Фиксатор PR и флаги C и Z также дают возможность управления переводом из одного состояния в другое в соответствии с предварительно накопленной информацией (рис. 7.21).

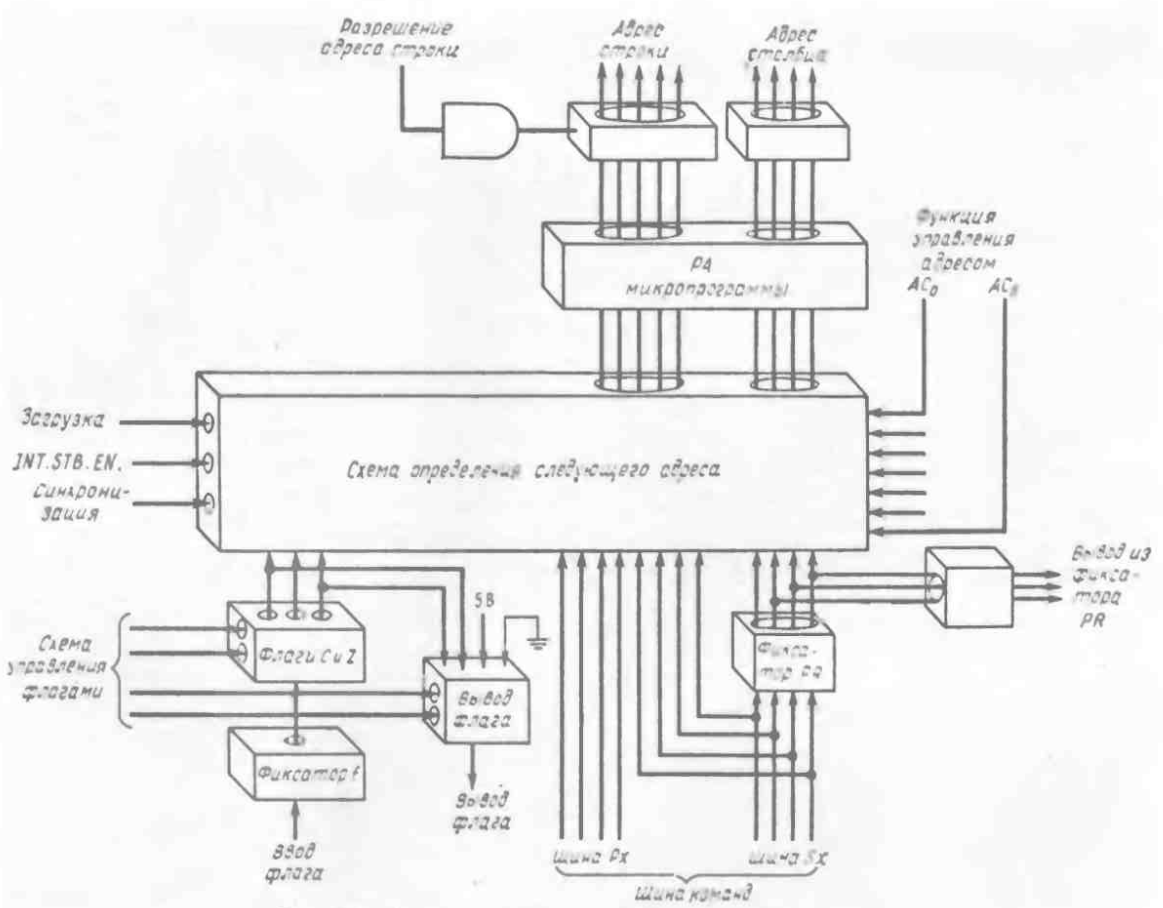


Рис. 7.21. Структурная схема устройства 3001.

РАЗРАБОТКА АРХИТЕКТУРЫ ПРОЦЕССОРА С ПЛАВАЮЩЕЙ ТОЧКОЙ

Существует по крайней мере два подхода к решению любой вычислительной задачи. Процедурно-ориентированный подход заключается в хранении алгоритма выполнения соответствующих вычислений и в реализации этой процедуры всякий раз, когда требуется получить искомый результат. Такой подход обычно приводит к экономии ресурсов памяти, но при этом увеличивается время решения задачи. Второй подход состоит в организации структур данных, содержащих результаты соответствующей обработки всех входных данных. В этом случае задача сводится к просмотру таблиц, а входные данные используются для вычисления адреса в этой таблице. При таком подходе, как правило, время выполнения программы сокращается за счет увеличения затрат памяти, связанных с хранением таблиц. В проектируемом нами процессоре с плавающей точкой выполнение основных математических действий над числами — сложения, вычитания, умножения и деления — будет осуществляться с использованием подхода организации структур данных, а операции с десятичной точкой и определение знака числа будут реализованы в виде процедур.

Аргументы хранятся локализованно в специальном запоминающем устройстве (рис. 7.22). Устройство управления также содержит локальную память. Это устройство обеспечивает выполнение процедур, определение последовательности выборки данных и формирование значений адресов, передаваемых в ЗУ. Детальное описание каждой из подсистем приводится ниже.

ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО

Запоминающее устройство для хранения аргументов выполняется на основе быстродействующей полупроводниковой памяти. Малые размеры этого устройства делают целесообразным использование памяти статического, а не динамического типа, так как несколько бо́льшая стоимость статической памяти вполне компенсируется простотой и легкостью установления интерфейса. После определения типа памяти целесообразно перейти к выбору внутреннего представления чисел. Введение указателей положения десятичной точки и хранение чисел в нормализованном формате делают возможным использование 4-разрядного местного ЗУ (рис. 7.23). Операнды вводятся и хранятся в буферах, обозначенных ARG1 и ARG2. В первых 14 ячейках ЗУ хранятся двоично-десятичные цифры, представляющие цифры в коде ASCII. Последние два символа отображают значение показателя степени. Положение десятичной точки записывается в

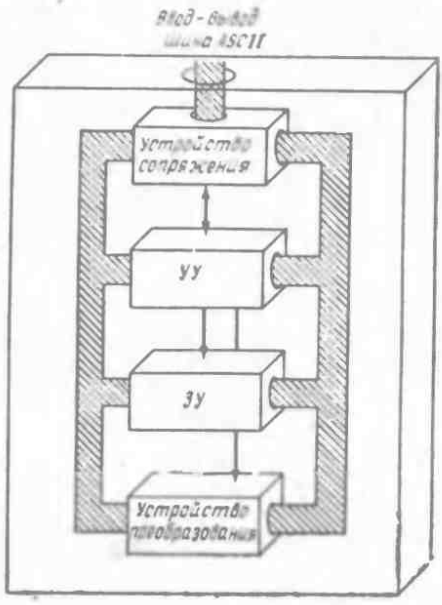


Рис. 7.22. Архитектура процессора с плавающей точкой.

ячейку *а* для числа ARG2 и в ячейку *б* для числа ARG1. Сама десятичная точка не хранится. Реализация этой подсистемы будет описана ниже.

УСТРОЙСТВО ПРЕОБРАЗОВАНИЯ

На вход устройства преобразования (рис. 7.24) подаются две двоично-десятичные цифры и код выбора функции. Выход устройства интерпретируется как функция кода выбора функции. Если выбирается функция сложения, то выход рассматривается как цифра суммы и цифра переноса. Если выбирается функция умножения, то выход интерпретируется как цифра произведения и цифра переноса. Аналогично в результате вычитания одной входной цифры из другой получаются цифра разности и цифра заема. Для простоты деление будет выполняться посредством невосстанавливаемого вычитания.

Основу показанного на рис. 7.24 устройства преобразования составляют восемь блоков быстродействующего ПЗУ объемом 1К. Десять адресных линий используются следующим образом. Две линии старших разрядов определяют выбор функции — умножение, деление, сложение или вычитание. Восемь линий младших адресных разрядов используются для ввода данных: одна

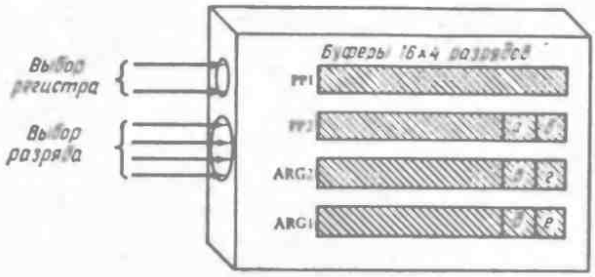


Рис. 7.23. Запоминающее устройство процессора с плавающей точкой: *а* — положение десятичной точки (DPT2) для ARG2; *б* — положение десятичной точки (DPT1) для ARG1; *в* — ARG2.EXP2; *г* — ARG2.EXP1; *д* — ARG1.EXP2; *е* — ARG1.EXP1.

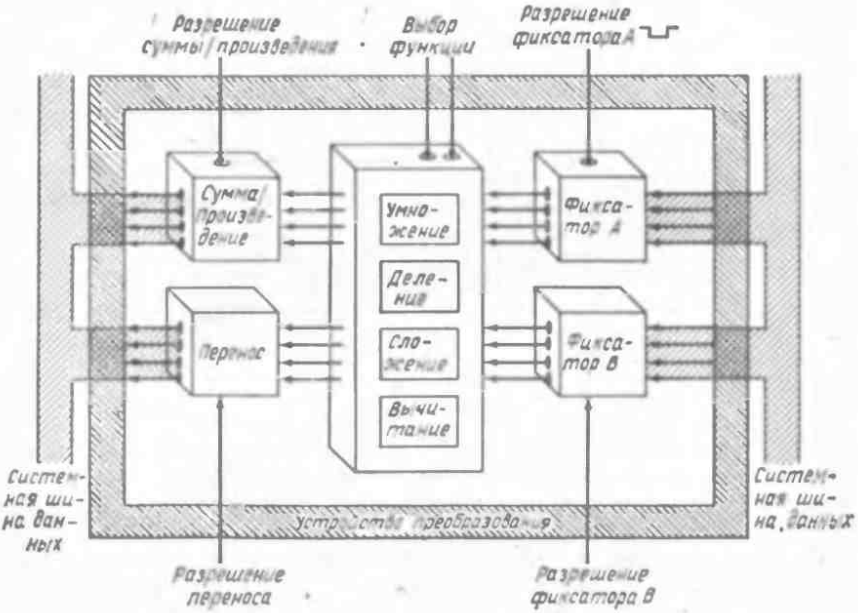


Рис. 7.24. Устройство преобразования для процессора с плавающей точкой.

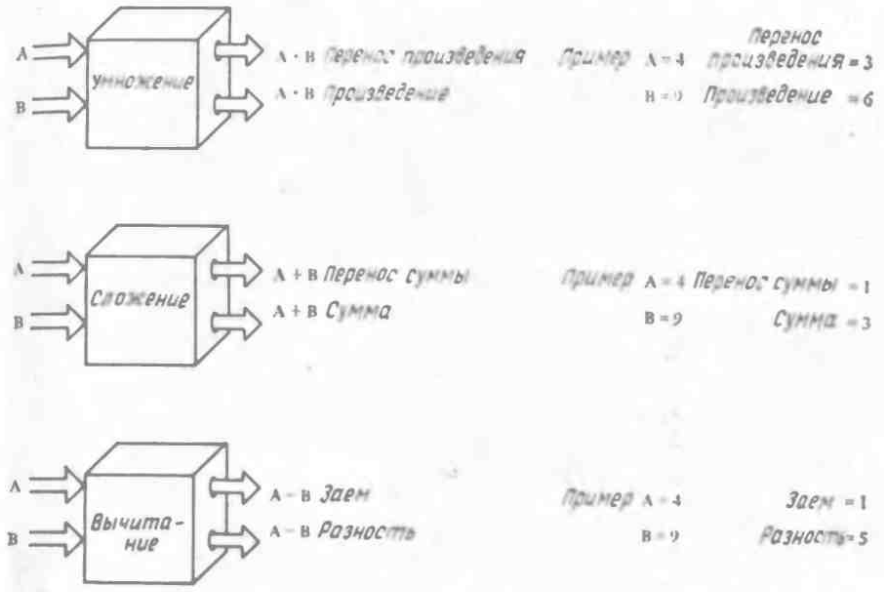


Рис. 7.25. Примеры использования устройства суммирования/умножения (Σ-П-ПЗУ).

двоично-десятичная цифра записывается в фиксатор А, другая — в фиксатор В. Восемь линий выхода из ПЗУ предназначены для 4-разрядной суммы/произведения (или разности) и 4-разрядного переноса (или заема), как показано на рис. 7.25. Эти выходные данные подаются в буферы с тремя состояниями и помещаются на системную шину данных с помощью линий разрешения суммы/произведения или переноса. Управление линиями разрешения осуществляет управляющее ПЗУ.

РЕАЛИЗАЦИЯ ЗАПОМИНАЮЩЕГО УСТРОЙСТВА

Исходная система содержит 64 полубайт местного ЗУ. В ЗУ этого типа легко встраивается кристалл памяти статического типа с произвольным доступом, аналогичный кристаллу ОЗУ 7489, показанному на рис. 7.26. Каждое из устройств, имеющих организацию 16×4 бит, представляет собой один регистр. Устройство 7489 обеспечивает инвертированный вывод и используется с буфером с тремя состояниями. Кристаллом, содержащим все местные регистры, является устройство 74S207 фирмы Texas Instruments емкостью 256×4 бит с временем цикла

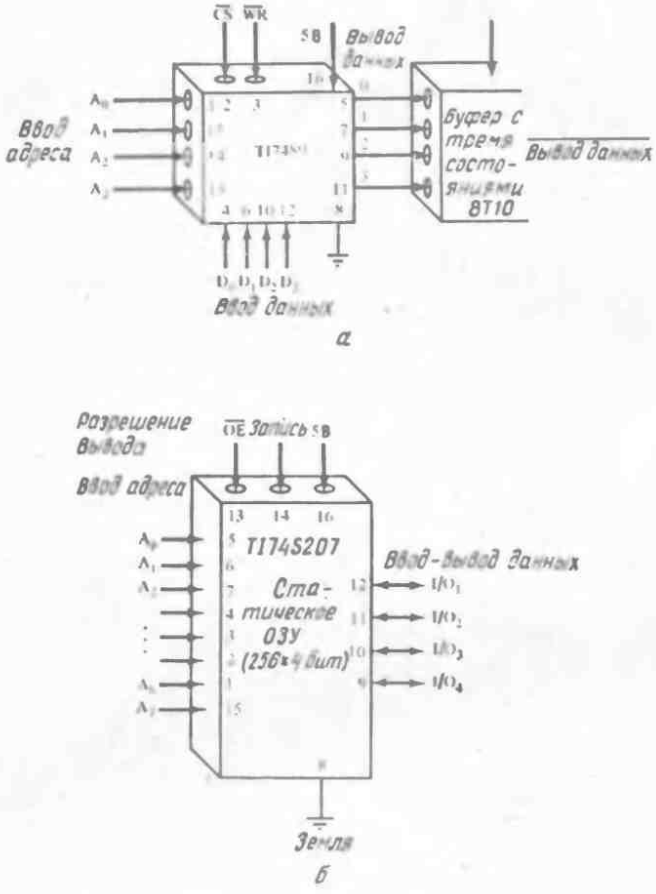


Рис. 7.26. Примеры реализации местных ЗУ: *а* — ОЗУ 7489; *б* — ОЗУ 74S207.

40 нс и выходами с тремя состояниями. Местное ЗУ управляется сигналами, передаваемыми по следующим линиям.

1. Выбор регистра. Эти два бита определяют выбор регистра, который будет использован в текущей операции. Физически это могут быть старшие адресные биты устройства 74S207 или сигналы, используемые для управления выбором кристалла 7489.

2. Выбор цифры. Эти четыре бита определяют выбор цифры, над которой в текущем цикле будет производиться операция. Их значение задается устройством управления.

ОПЕРАЦИИ ОБРАБОТКИ ПОКАЗАТЕЛЕЙ СТЕПЕНИ

При использовании местного ЗУ 7489, показанного на рис. 7.26, и регистров, приведенных на рис. 7.23, описанные выше линии выбора регистра и цифры могут быть использованы для доступа к показателям степени. Приведенная здесь процедура иллюстрирует способ определения показателя степени для ARG1 в соответствии с положением десятичной точки, т. е. нормализации операнда ARG1. Процедура описывает порядок обработки при положительных показателях степени. Для отрицательных показателей используется блок вычитания суммирующего ПЗУ (Σ-ПЗУ).

Процедура нормализации операнда ARG1 состоит в следующем:

1. Считать DPT1 и передать в блок Σ-ПЗУ (если 0, то переход к обработке следующего показателя степени).
2. Считать ARG1.EXP1 и передать в блок Σ-ПЗУ.
3. Записать сумму, полученную в результате выполнения шагов 1 и 2, в область ARG1.EXP1.
4. Проверить значение переноса. Если оно равно 0, то переход к следующему показателю степени; в противном случае переход к выполнению следующего шага.
5. Считать ARG1.EXP2 и сложить со значением переноса (который должен быть зафиксирован на входе).
6. Записать сумму в область ARG1.EXP2.
7. Проверить значение переноса (если это значение не равно 0, прервать обработку либо произвести выдачу трехзначного показателя степени без младшего знака).
8. Переход к обработке показателя степени для ARG2.

Приведенный выше пример является типичным для всех алгоритмов операции сложения, использующих Σ-ПЗУ. В следующем разделе описан алгоритм умножения с использованием перемножающего ПЗУ (П-ПЗУ), представляющего собой часть общего устройства Σ-П-ПЗУ.

АЛГОРИТМ УМНОЖЕНИЯ

Ниже описывается алгоритм вычисления (i+j)-й цифры частичного произведения, получаемой при умножении i-й цифры первого из сомножителей на j-ю цифру второго.

1. Передать $a_j \cdot b_i$ на вход П-ПЗУ, где a_j—j-я цифра множителя, а b_i—i-я цифра множимого.
2. Принять с выхода П-ПЗУ $a_j \cdot b_i$ = Произведение + Перенос.
3. Вычислить выражение

Сумма (i+j) + Перенос суммы = Предыдущая сумма (i+j) + + Произведение + Перенос предыдущей суммы + Перенос предыдущего произведения.

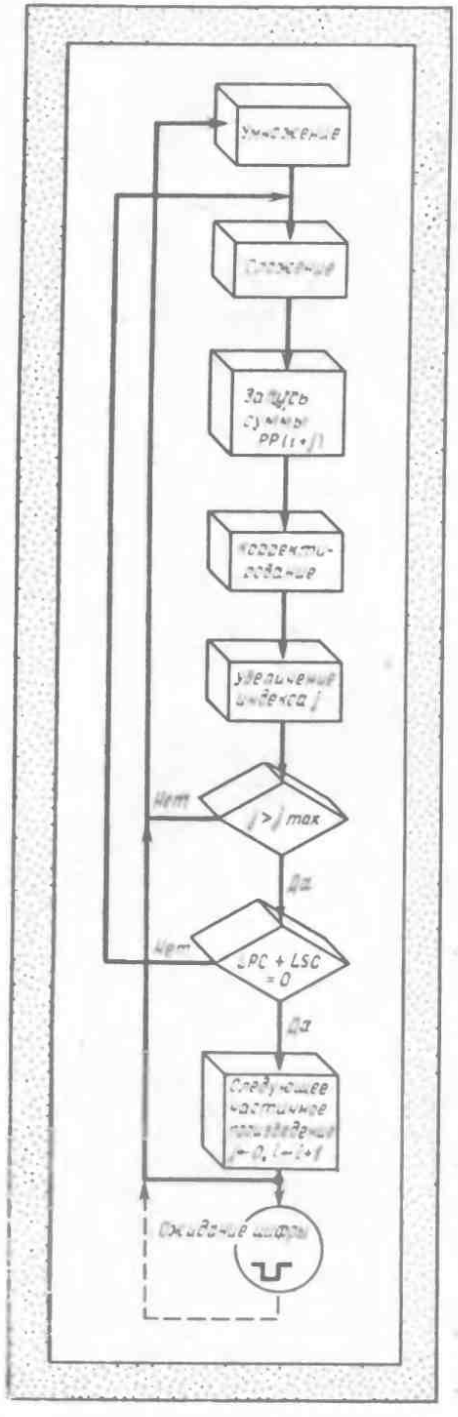


Рис. 7.27. Блок-схема алгоритма умножения.

4. Записать сумму (i+j) в ячейку (i+j) частичного произведения ОЗУ.

5. Перенос предыдущего произведения ← Перенос произведения.

6. Перенос предыдущей суммы←Перенос суммы.

7. Увеличить индекс j множителя и сравнить с j max: переход к 1, если $j \leqslant j\,max$.

8. Проверить условие

Перенос предыдущего произведения + Перенос предыдущей суммы=0.

Если условие не соблюдается, то переход к п. 3, иначе выполнение следующего шага.

9. Увеличить индекс i множимого, установить начальное значение индекса множителя, равное 0.

10. Переход к п 1.

Блок-схема этой процедуры представлена на рис. 7.27. Введение состояния «Ожидание цифры» для нормализованных входных данных позволяет производить умножение сразу же при поступлении в процессор очередной цифры множителя. Для того чтобы понять действие алгоритма, читателю рекомендуется проверить его на нескольких примерах с числами, состоящими из трех или четырех цифр.

Прежде чем приступить к проектированию управляющей части процессора с плавающей точкой, рассмотрим последовательность опе-

раций, необходимых для управления входными и выходными данными устройства преобразования, показанного на рис. 7.24.

1. Подать a_J и b_I в П-ПЗУ на входы Па и Пb.
2. П-ПЗУ⟶Произведение+Перенос произведения; загрузить «Перенос произведения» в регистр временного хранения.
3. Подать значение произведения в Σ-ПЗУ на вход Σa.
4. Подать величину «Предыдущая сумма=(I+J)» в Σ-ПЗУ на вход Σb.
5. Зафиксировать значение временной суммы и временного переноса, полученные из Σ-ПЗУ.
6. Прибавить значение временного переноса к содержимому регистра временного хранения. (В регистре временного хранения теперь находится «Перенос произведения+Временный перенос».)
7. Подать значение «Временная сумма=Произведение+ +Предыдущая сумма» на вход Σa.
8. Подать предыдущий перенос на вход Σb.
9. Записать сумму, полученную из Σ-ПЗУ, в ячейку I+J ОЗУ.
10. Сложить перенос суммы, полученный из Σ-ПЗУ, с содержимым регистра временного хранения.
11. Записать результирующее значение переноса в область хранения предыдущего переноса.

Из последовательности приведенных выше операций следует, что для хранения промежуточных результатов и таких величин, как «Предыдущий перенос», требуется дополнительная память. Хотя местное ЗУ и может быть расширено до необходимого объема, наиболее предпочтительным является включение оперативной памяти непосредственно в управляющий элемент, в том числе памяти, необходимой для хранения индексов.

УСТРОЙСТВО УПРАВЛЕНИЯ

До того как приступить к конкретному проектированию устройства управления, необходимо четко представить себе все действия по управлению, которые необходимы для обеспечения нормальной работы процессора. К этим действиям следует отнести

- Разрешение/Запрет выдачи данных на системные шины.
- Фиксацию данных в регистрах временного хранения.
- Выбор регистра запоминающего устройства.
- Выбор цифры в регистре.
- Декодирование и синтаксический анализ входной строки символов в коде ASCII.
- Синхронизацию по стробам записи от ведущего процессора.
- Генерирование сигналов подтверждения и готовности ведущему процессору.
- Выдачу данных в коде ASCII ведущему процессору.

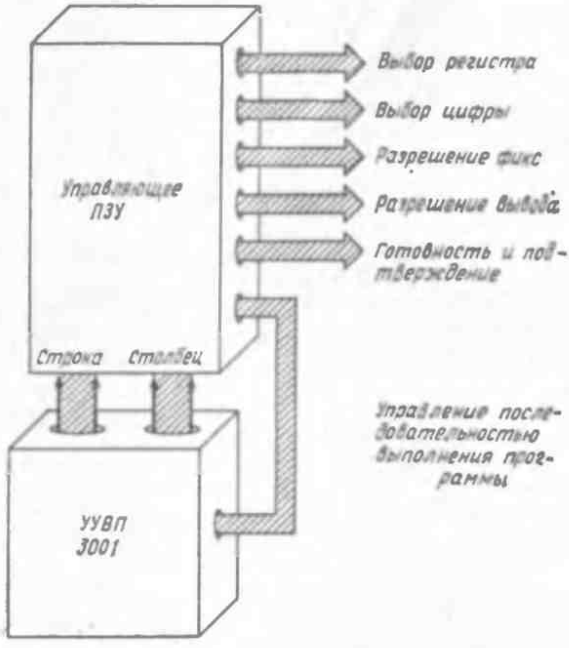


Рис. 7.28. Первый уровень рассмотрения устройства управления.

Порядок синхронизации и синтаксический анализ входной строки символов в коде ASCII описывались в предыдущих разделах. Генерирование сигналов готовности и подтверждения ведущему процессору и выдача данных в коде ASCII реализуются с помощью процедур. Выбор регистров и цифр, разрешение выдачи данных на системные шины и фиксация данных, полученных с шины, будут производиться посредством управляющих полей в управляющих словах (рис. 7.28).

Поля управления микропрограммой имеют следующий вид:

Поле выбора регистра	Поле выбора цифры	Поле разрешения работы фиксатора	Поле разрешения вывода	Поле готовности и подтверждения	Поле управления мультиплексированием	Поле управления последовательностью микрокоманд 3001

Анализ алгоритмов умножения и сложения показывает, что в поле выбора цифры хранится переменный индекс, и это поле, строго говоря, не принадлежит управляющему ПЗУ. Более удобной была бы его реализация с помощью заранее устанавливаемого регистра с автоматическим увеличением значения. Однако хотя такие регистры и выпускаются, мы будем использовать

устройство более общего назначения — процессорный элемент 3002 фирмы Intel, характеристики которого идеально подходят для этого применения. Ниже это устройство рассматривается более подробно.

РЕАЛИЗАЦИЯ УСТРОЙСТВА УПРАВЛЕНИЯ

Как указывалось в предыдущем разделе, устройство 3002 обеспечивает изменение и хранение переменных индексов, а также передачу сигналов подтверждения и готовности ведущему процессору. Поскольку максимальное количество цифр в числах, обрабатываемых процессором, равно 14, достаточно иметь 4-битовый индекс для адресации любой цифры в данном регистре запоминающего устройства. Значение индекса может сохраняться в подсистеме разрядно-модульной организации, состоящей из двух параллельно соединенных блоков 3002. Индексы будут храниться в наборе регистров. Устройство 3002 будет также использоваться для сохранения знаков операндов и показателей степеней, а при необходимости и информации о промежуточных переносах. Исходное распределение регистров может быть следующим:

R0 Индекс I
R1 Индекс J
R2 Перенос произведения
R3 Перенос предыдущего произведения
R4 Перенос суммы
R5 Перенос предыдущей суммы
R6 Знаки
R7 I+J

Напомним, что значение предыдущей суммы будет храниться в области PP(i+j) частичного произведения ОЗУ, протяженность которой не более двух 16-разрядных регистров. Отметим также, что декодирование данных в коде ASCII выполняется с помощью команды JPX в УУВП 3001, а положение десятичной точки, E и пробелы в коде ASCII являются ограничителями, которые в формате ASCII внутреннему хранению не подлежат. Поэтому, несмотря на количество типов данных в коде ASCII, вводимых и выводимых из процессора с плавающей точкой, 4-разрядные внутренние шины данных отвечают предъявляемым требованиям. Вернемся теперь к рассмотрению устройства управления.

Как показано на рис. 7.29, адресная шина, ведущая к ОЗУ и обеспечивающая выбор цифры в данном регистре, получает данные с аккумулятора подсистемы, построенной на процессорных элементах 3002. Системная шина данных является входной по отношению к входному порту устройства 3002 и управляется выходным портом этого устройства, обычно называемым выво-

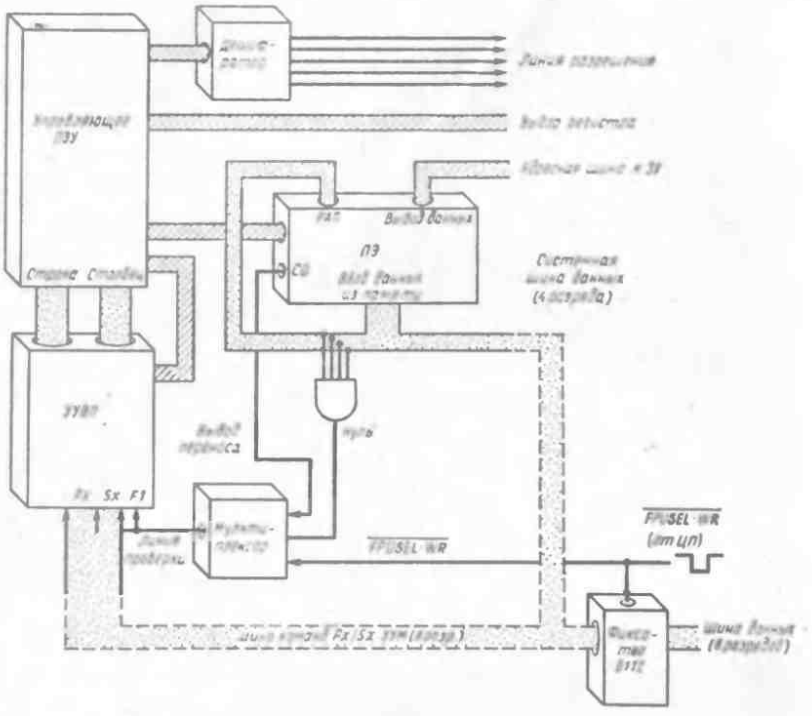


Рис. 7.29. Второй уровень рассмотрения устройства управления.

дом регистра адреса памяти, работа которого производится под управлением ПЗУ. Необходимо отметить, что передаваемые по системной шине данные проверяются на нуль с помощью схемы ИЛИ и что линия ввода значения флага в УУВП 3001 связана с мультиплексором, позволяющим осуществлять проверку линии переноса устройства 3002, проверку на нуль 4-разрядной системной шины данных и проверку строба записи от ведущего процессора.

АРХИТЕКТУРА ПРОЦЕССОРА С ПЛАВАЮЩЕЙ ТОЧКОЙ

При проектировании процессора с плавающей точкой были пройдены следующие этапы:

1. Оценка возможных способов разработки процессора и выбор разрядно-модульной организации.
2. Выбор десятичной системы счисления и формата ASCII для представления чисел с плавающей точкой.
3. Выбор последовательного побайтного метода доступа и разработка алгоритма синтаксического анализа.

4. Представление алгоритма синтаксического анализа в виде модели машины состояний с высокой степенью адекватности при использовании УУВП 3001.

5. Проектирование архитектуры процессора на базе устройства преобразования, использующего табличный метод вычислений, и местного запоминающего устройства для двоично-десятичных чисел.

6. Выбор способа реализации местного запоминающего устройства.

7. Разработка на основе ПЗУ устройства преобразования, осуществляющего операции сложения и умножения ($\Sigma - \Pi$).

8. Разработка алгоритма умножения, основанного на применении устройства Σ—П-преобразования.

9. Определение управляющих полей микрокоманды в первом приближении.

10. Проведение более глубокого анализа алгоритма умножения и выявление необходимости в полях, хранящих переменную информацию.

11. Совершенствование устройства управления для обеспечения возможности обработки полей, хранящих переменную информацию, и включение в него оперативной памяти.

Архитектура системы после выполнения перечисленных этапов представлена на рис. 7.30.

ИНТЕРФЕЙС С ВЕДУЩИМ ПРОЦЕССОРОМ

Спроектированный интерфейс с ведущим процессором является самым обычным. Рассмотрим пример, иллюстрирующий интерфейс с процессором типа 8080. Управляющий процессор 8080, показанный на рис. 7.31, передает данные в коде ASCII по 8-разрядной шине данных, а адрес процессора с плавающей точкой — по адресной шине. Если выбрана структура адресации по аналогии с обращением к памяти, то для чтения данных из процессора и записи в него данных служат линии MEMR и MEMWR. При использовании обычных команд ввода-вывода эти операции выполняются с помощью сигналов IOR и IOW. Эти сигналы вырабатываются устройством 8224 или аналогичной схемой. Интерфейс процессора, показанный в правой части рисунка, допускает это и является достаточно общим для того, чтобы его использовать при работе почти с любым микропроцессором или мини-ЭВМ.

ПРОЦЕДУРА ОБРАБОТКИ ПОКАЗАТЕЛЯ СТЕПЕНИ

Знаки показателей степени первоначально устанавливаются положительными и изменяются лишь в том случае, если после приема символа «Е» встречается отрицательный знак. Описанная ниже процедура для простоты производит обработку только

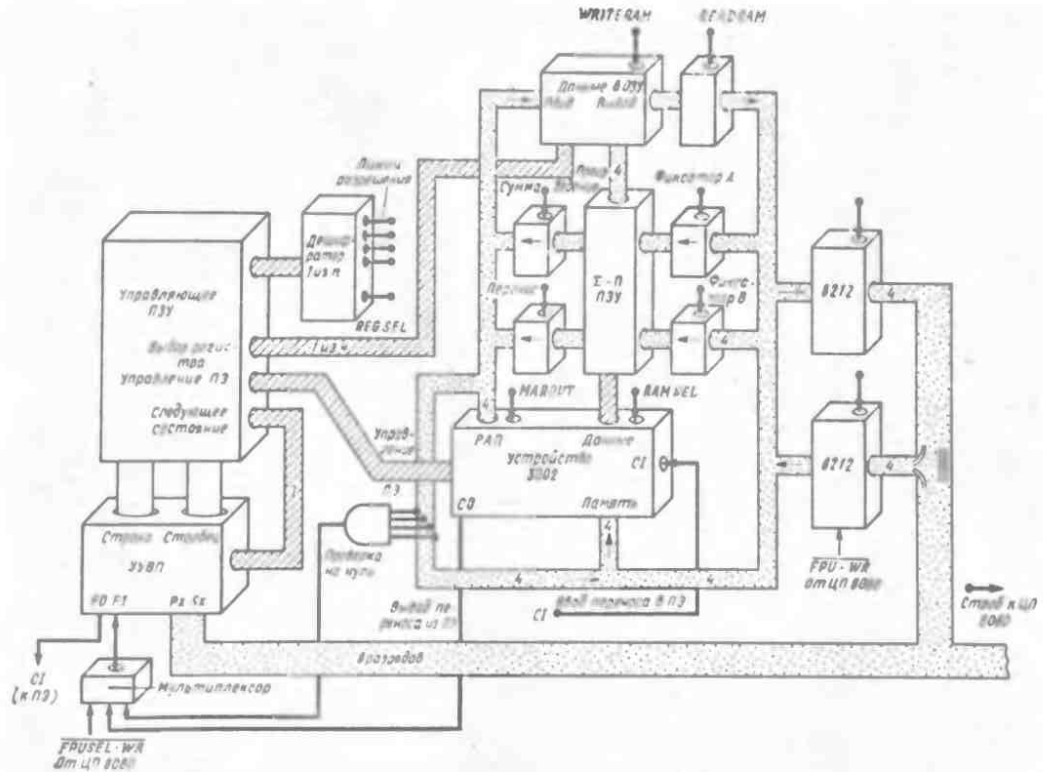


Рис. 7.30. Структурная схема процессора с плавающей точкой.

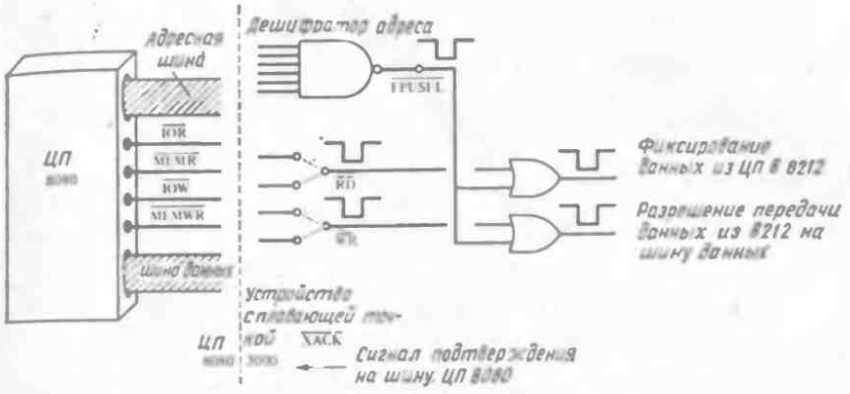


Рис. 7.31. Пример интерфейса с ведущим процессором типа 8080.

положительных показателей. Напомним, что первоначальный формат записи, показанный на рис. 7.23, был определен до того, как было принято решение об использовании ПЭ 3002. После разработки алгоритма синтаксического анализа становится ясно, что положение десятичной точки может быть определено гораздо проще, если указатели ее положения размещаются в двух регистрах из имеющихся в устройстве 3002, например в регистрах R8 и R9. Таким образом, новый формат записи для процессора с плавающей точкой становится таким, как показан на рис. 7.32.

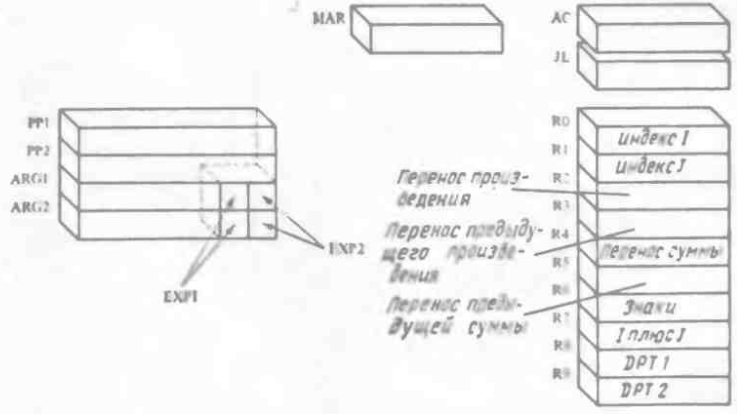


Рис. 7.32. Внутренний формат записи в системе с плавающей точкой.

Используя новый формат записи, можно определить последовательность команд управления, необходимую для нормализации числа, заданного в экспоненциальной форме:

$$XXX.XXXEY \longrightarrow .XXXXEZ \quad \text{где } Z = Y + 3$$

или, в общем случае,

$$X\cdots X.XXXXEY \longrightarrow .X\cdots XXXXEZ \quad \text{где } Z = Y + n$$

Если допустить, что аргументы хранятся в памяти и что в регистрах DPT1 и DPT2 содержится местоположение десятичных точек для аргументов 1 и 2, то процедура нормализации ARG1 будет аналогична процедуре, описанной в разделе «Операции обработки показателей степени». Ниже будет рассмотрен каждый из составляющих ее шагов и определена соответствующая последовательность команд управления.

Таблица 7.2. **Функции, реализуемые в ПЭ при передаче по шине К либо нулей, либо единиц во всех разрядах**

Мнемоническое обозначение	Микрофункция (шина K=00)	Мнемоническое обозначение	Микрофункция (шина K=11)
ILR ACM SRA	$R_n + CI \rightarrow R_n$ AC $M + CI \rightarrow AT$ $AT_L \rightarrow RO$ $AT_H \rightarrow AT_L$ $LI \rightarrow AT_H$	ALR AMA –	$AC + R_n + CI \rightarrow R_n$ AC $M + AC + CI \rightarrow AT$
LMI LMM CIA	$R_n \rightarrow MAR$ $R_n + CI \rightarrow R_n$ $M \rightarrow MAR$ $M + CI \rightarrow AT$ $AT + CI \rightarrow AT$	DSM LDM DCA	$11 \rightarrow MAR$ $R_n - 1 + CI \rightarrow R_n$ $11 \rightarrow MAR$ $M - 1 + CI \rightarrow AT$ $AT - 1 + CI \rightarrow AT$
CSR CSA –	$CI - 1 \rightarrow R_n$ (См. прим. 1) $CI - 1 \rightarrow AT$ (См. CSA)	SDR SDA LDI	$AC - 1 + CI \rightarrow R_n$ $AC - 1 + CI \rightarrow AT$ (См. прим. 1) $I - 1 + CI \rightarrow AT$
INR INA	$R_n + CI \rightarrow R_n$ (См. ACM) $AT + CI \rightarrow AT$	ADR – AIA	$AC + R_n + CI \rightarrow R_n$ (См. AMA) $I + AT + CI \rightarrow AT$
CLR CLA –	$CI \rightarrow CO$ $0 \rightarrow R_n$ $CI \rightarrow CO$ $0 \rightarrow AT$ (См. CLA)	ANR ANM ANI	$CI \vee (R_n \wedge AC) \rightarrow CO$ $R_n \wedge AC \rightarrow Rn$ $CI \vee (M \wedge AC) \rightarrow CO$ $M \wedge AC \rightarrow AT$ $CI \vee (AT \wedge I) \rightarrow CO$ $AT \wedge I \rightarrow AT$
– – –	(См. CLR) (См. CLA) (См. CLA)	TZR LTM TZA	$CI \vee R_n \rightarrow CO$ $R_n \rightarrow R_n$ $CI \vee M \rightarrow CO$ $M \rightarrow AT$ $CI \vee AT \rightarrow CO$ $AT \rightarrow AT$
NOP LMF –	$CI \rightarrow CO$ $R_n \rightarrow R_n$ $CI \rightarrow CO$ $M \rightarrow AT$ (См. NOP)	ORR ORM ORI	$CI \vee AC \rightarrow CO$ $R_n \vee AC \rightarrow R_n$ $CI \vee AC \rightarrow CO$ $M \vee AC \rightarrow AT$ $CI \vee I \rightarrow CO$ $I \vee AT \rightarrow AT$
CMR LCM CMA	$CI \rightarrow CO$ $\overline{R_n} \rightarrow R_n$ $CI \rightarrow CO$ $\overline{M} \rightarrow AT$ $CI \rightarrow CO$ $\overline{AT} \rightarrow AT$	XNR XNM XNI	$CI \vee (R_n \wedge AC) \rightarrow CO$ $R_n \overline{\vee} AC \rightarrow R_n$ $CI \vee (M \wedge AC) \rightarrow CO$ $M \overline{\vee} AC \rightarrow AT$ $CI \vee (AT \wedge I) \rightarrow CO$ $I \overline{\vee} AT \rightarrow AT$

Примечания.

1. В дополнительном коде в двоичной системе счисления вместо вычитания 000...01 производится прибавление 111...11.

2. В микрофункциях группы R подгруппы 1 в качестве R_n могут использоваться T и AC и как регистры-источники, и как регистры-приемники.

3. Стандартные выходные значения арифметического переноса формируются с помощью команд группы F подгрупп 0—3.

Символ	Значение
I, K, M	Данные на шинах I, K и M соответственно
CI, LI	Данные на входе переноса и левом входе соответственно
CO, RO	Данные на выходе переноса и правом выходе соответственно
R_n	Содержимое регистра n, включая T и AC (группа R, подгруппа 1)
AC	Содержимое аккумулятора
AT	Содержимое AC или T (указывается в команде)
MAR	Содержимое регистра адреса памяти
L, H	Индекс, обозначающий младший или старший разряд соответственно
+	Сложение в дополнительном коде в двоичной системе счисления
—	Вычитание в дополнительном коде в двоичной системе счисления
$\vee$	Логическое И
$\wedge$	Логическое ИЛИ
$\overline{\otimes}$	ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ
→	Перенос

ПОСЛЕДОВАТЕЛЬНОСТЬ КОМАНД УПРАВЛЕНИЯ ДЛЯ НОРМАЛИЗАЦИИ ARG1

Первым должен быть выполнен следующий шаг:

1. Считать DPT1 и передать в блок Σ-ПЗУ (если 0, то переход к обработке следующего показателя степени).

Анализ формата записи и архитектуры системы (рис. 7.30) показывает, что содержимое регистра R8 устройства 3002 должно быть помещено на системную шину данных и сохранено в фиксаторе A или B. Из табл. 7.2 следует, что микрокоманда LMI, получаемая из булевых уравнений

$$LMI(R_n) = K \vee R_n \longrightarrow MAR \quad R_n + K + C_{in} \longrightarrow R_n$$

при K=0 обеспечивает выполнение необходимой операции:

$$LMI(R_n) = R_n \longrightarrow MAR \qquad R_n + C_{in} \longrightarrow R_n$$

Здесь необходимо остановиться на некоторых аспектах рас-

Таблица 7.3. Операции, выполняемые ПЭ

Группа F	Группа R	Микрофункция
0	I	$R_n + (AC \wedge K) + CI \rightarrow R_n$, AC
	II	$M + (AC \wedge K) + CI \rightarrow AT$
	III	$AT_L \wedge (\overline{I_L} \wedge \overline{K_L}) \rightarrow RO \quad LI \vee (I_H \wedge K_H) \wedge AT_H] \rightarrow AT_H$ $[AT_L \wedge (\overline{I_L} \wedge \overline{K_L})] \vee [AT_H \wedge (I_H \wedge K_H)] \rightarrow AT_L$
1	I	$K \vee R_n \rightarrow MAR \qquad R_n + CL + K \rightarrow R_n$
	II	$K \vee M \rightarrow MAR \qquad M + CI + K \rightarrow AT$
	III	$(\overline{AT} \vee K) + (AT \quad K) + CI \rightarrow AT$
2	I	$(AC \wedge K) - 1 + CI \rightarrow R_n$ *(См. прим.)*
	II	$(AC \wedge K) - 1 + CI \rightarrow AT$
	III	$(I \wedge K) - 1 + CI \rightarrow AT$
3	I	$R_n + (AC \wedge K) + CI \rightarrow R_n$
	II	$M + (AC \wedge K) + CI \rightarrow AT$
	III	$AT + (I \wedge K) + CI \rightarrow AT$
4	I	$CI \vee (R_n \quad AC \quad K) \rightarrow CO \qquad R_n \wedge (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (M \quad AC \quad K) \rightarrow CO \qquad M \wedge (AC \wedge K) \rightarrow AT$
	III	$CI \vee (AT \quad I \quad K) \rightarrow CO \qquad AT \wedge (I \wedge K) \rightarrow AT$
5	I	$CI \vee (R_n \wedge K) \rightarrow CO \qquad K \wedge R_n \rightarrow R_n$
	II	$CI \vee (M \wedge K) \rightarrow CO \qquad K \wedge M \rightarrow AT$
	III	$CI \vee (AT \wedge K) \rightarrow CO \qquad K \wedge AT \rightarrow AT$
6	I	$CI \vee (AC \wedge K) \rightarrow CO \qquad R_n \vee (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (AC \wedge K) \rightarrow CO \qquad M \vee (AC \wedge K) \rightarrow AT$
	III	$CI \vee (I \wedge K) \rightarrow CO \qquad AT \vee (I \wedge K) \rightarrow AT$
7	I	$CI \vee (R_n \wedge AC \wedge K) \rightarrow CO \qquad R_n \overline{+} (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (M \wedge AC \wedge K) \rightarrow CO \qquad M \overline{+} (AC \wedge K) \rightarrow AT$
	III	$CI \vee (AT \wedge I \wedge K) \rightarrow CO \qquad AT \overline{+} (I \wedge K) \rightarrow AT$

Примечание. В дополнительном коде в двоичной системе счисления вместо вычитания 000...01 производится прибавление 111...11. С *разрешения* Intel Corp., © 1981.

сматриваемого проекта, в частности на зависимости результата операции от состояния линии ввода переноса C_{In}. Анализ табл. 7.3 показывает, что большинству операций свойственна такая зависимость, т. е. состояние линии определяет необходимость увеличения или сохранения результата операции. Состояние линии ввода переноса матрицы устройства 3002 обычно находится под управлением микропрограммы. Это может быть осуществлено с помощью специального управляющего бита при вводе переноса. Однако обычно для этого используется вывод значения флага устройства 3001 (рис. 7.33). С помощью значений, предварительно занесенных в флаги C и Z, оказывается возможным влиять на состояние линии переноса, а также производить установку и сброс передаваемых по ней значений. Отсюда следует, что параллельно с управлением процессорным элементом осуществляется управление линией ввода переноса

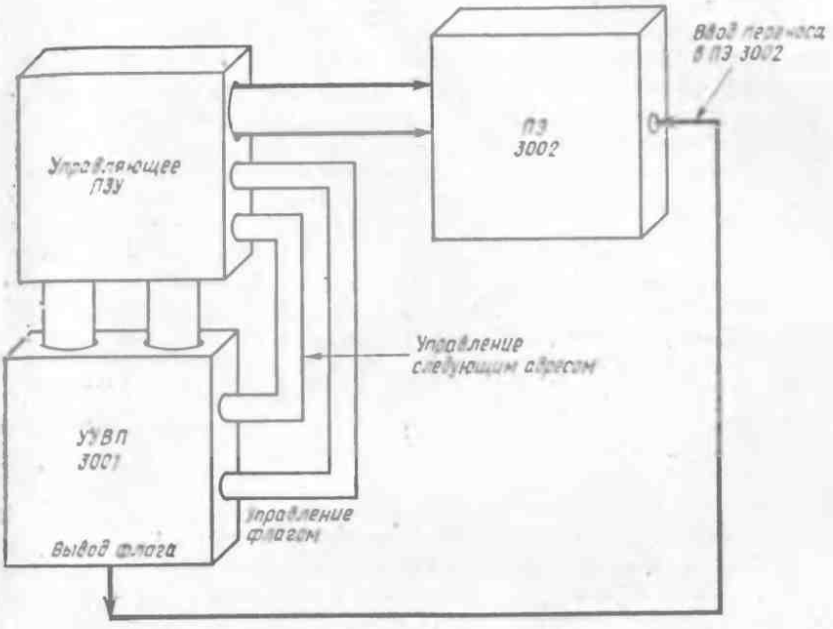


Рис. 7.33. Общая схема использования устройства 3002.

посредством управляющего поля вывода флага. При этом три управляющих поля будут иметь следующие значения:

Шина К	Управление регистровым АЛУ	Управление выводом флага	
0	LMI(R8)	FF1	---

Кроме того, необходимо сохранить данные, выходящие из регистра адреса памяти (РАП) устройства 3002, в фиксаторе А. Это выполняется путем подачи соответствующих данных из РАП на системную шину данных и активацией шины записи фиксатора А, что вызывает необходимость введения дополнительных управляющих полей:

Шина К	Управление регистровым АЛУ	Управление выводом флага	Разрешение вывода	Разрешение работы фиксатора	
0	LMI(R8)	FF1	MAROUT	LATCHA	---

Наконец необходимо проверить на нуль положение десятичной точки. Для этого мультиплексор в линии ввода флага устанавливается в состояние «Проверка на нуль» и производится проверка фиксатора f, в результате которой выполняется следующий шаг или вводится следующий показатель степени. Это обеспечивается двумя управляющими полями.

Шина К	Управление регистром АЛУ	Управление выводом флага	Разрешение вывода	Разрешение работы фиксатора	Канал мультиплексора	Функция управления адресом	
0	LMUX(1)	FF1	MAROUT	LATCHA	ZEROTEST	JF1 (NEXT NEXT EXPO STEP)	***

В результате выполнения операции положение десятичной точки в ARG1 фиксируется на одном из вводов в Σ-ПЗУ. Затем младший разряд аргумента 1 помещается на другой ввод в Σ-ПЗУ. Это значение хранится в крайнем справа разряде регистра ARG1 — разряде 15 или 0FH. Поэтому необходимо определить адрес этого разряда, считать показатель степени из ОЗУ и сохранить его в другом фиксаторе блока Σ-ПЗУ — фиксаторе B.

Эти операции выполняются на втором шаге процедуры обработки показателя степени:

2. Считать ARG1.EXP1 и передать в блок Σ-ПЗУ.

Для написания микропрограммы этого шага необходимо отметить, что адрес разряда указывается ОЗУ путем вывода данных из буферного аккумулятора устройства 3002. Поэтому необходимо сначала загрузить в аккумулятор число 0FH и одновременно подать сигнал разрешения вывода данных. Значение 0FH может быть занесено в аккумулятор с помощью микрокоманды CSA («Сброс или установка в 1 аккумулятора»). Из табл. 7.2 следует, что микрокоманда CSA имеет нулевую шину К и определяется как C_{in}—1⟶AT

Когда по линии ввода переноса передается нуль (посредством передачи числа FF0 по управляющим линиям вывода флагов), значение — 1 помещается либо в аккумулятор A, либо во временный аккумулятор T. Поскольку — 1 в четырехразрядном двоичном представлении имеет значение 1111, для получения в аккумуляторе значения 0FH необходимо записать команду CSA(AC) FF0.

Таким образом, второе управляющее слово имеет следующие управляющие поля:

Шина К	Управление регистровым АЛУ	Управление выводом Флага	Разрешение вывода	Разрешение работы фиксатора	Канал мультиплексора	Управление адресом	Управление ОЗУ	Выбор регистра
0	CSA(AC)	FF0	RAMSEL	LATCHB	----	JMP---	READRAM	ARG1	---

В результате выполнения этого шага значение DPT1 помещается в фиксатор A, а младшая цифра ARG1.EXP1 — в фиксатор B. Сумма этих чисел и значение возможного переноса появляются на выходе блока Σ-ПЗУ. Эти данные обрабатываются в соответствии с шагом 3.

3. Записать сумму ARG1.EXP1 и DPT1 в область ARG1.EXP1.

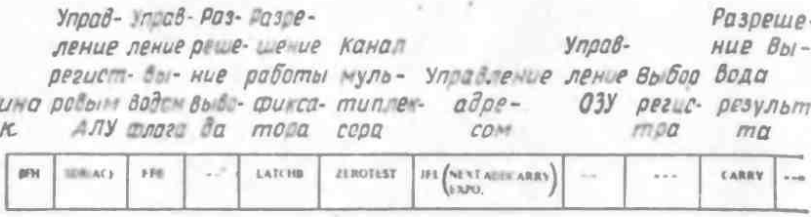
Этот шаг легко выполнить выбором соответствующего регистра ОЗУ и разряда, помещением суммы на системную шину данных и записью в ОЗУ. Поскольку в аккумуляторе уже содержится нужный адрес ОЗУ, для устройства 3002 указывается оператор NOP (пустой оператор). Таким образом, управляющее слово на шаге 3 имеет вид

Шина К	Управление регистровым АЛУ	Управление выводом Флага	Разрешение вывода	Разрешение работы фиксатора	Канал мультиплексора	Управление адресом	Управление ОЗУ	Выбор регистра	Разрешение вывода результата
0	NOP	...	RAMSEL	.....	.....	JMP...	WRITERAM	ARG1	SUM	---

Следующим шагом процедуры является

4. Проверить значение переноса. Если оно равно 0, то переходим к следующему показателю степени.

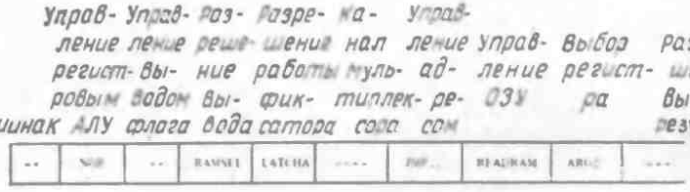
Для проверки значения переноса подается сигнал разрешения вывода переноса из блока Σ-ПЗУ и производится выбор канала ZEROTEST (проверка на нуль) мультиплексора ввода сигнала f. Если значение переноса равно нулю, то обрабатывается следующий показатель степени; в противном случае перенос прибавляется к ARG1.EXP2. (Здесь сделано допущение, что число, определяющее положение десятичной точки, $\leqslant 9$.) При выполнении проверки на нуль производится уменьшение значения аккумулятора с 0FH до 0EH, так что он начинает указывать адрес ARG1.EXP2 на тот случай, если он понадобится.



5. Считать ARG1.EXP2 и сложить со значением переноса.

Теперь перенос зафиксирован на входе в блок Σ-ПЗУ, а адрес области ARG1.EXP2 находится в аккумуляторе устройства 3002. Остается лишь выбрать соответствующий регистр, считать ARG1.EXP2, сложить со значением переноса и записать результат в область ARG1.EXP2. Это может быть выполнено с помощью управляющего слова.

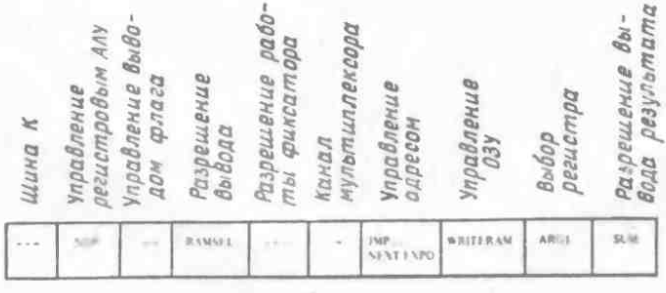
```
ADDCARRY1:
```



Значение области ARG1.EXP2 помещается на вход блока Σ-ПЗУ, далее выполняется следующий шаг.

6. Записать сумму в область ARG1.EXP2.

```
STORECARRY:
```



После выполнения этого шага управление передается NEXTEXPO (обработка следующего показателя степени), и описанная процедура повторяется для ARG2.

МИКРОПРОГРАММА ОПЕРАЦИИ УМНОЖЕНИЯ

В предыдущих разделах была описана процедура обработки показателей степени, а также указан способ выполнения сложения и вычитания мантисс. В этом разделе рассматривается обработка разряда b_i множителя и разряда a_j множимого из ОЗУ, передача их в П-ПЗУ и Σ-ПЗУ и обработка произведения и значения переноса произведения в соответствии с алгоритмом

Сумма + Перенос суммы = Произведение + Предыдущая сумма + + Перенос предыдущего произведения + + Перенос предыдущей суммы

Сумма хранится в ячейке I+J в 32-разрядном регистре частичного произведения, замещая величину предыдущей суммы, а перенос произведения и перенос суммы формируют предыдущий перенос, который запоминается и модифицируется в регистре временного хранения устройства 3002.

Как правило, при написании микропрограмм указываются только активные поля. Обычно имеется программа, которая осуществляет перевод мнемонической записи в набор двоичных разрядов для генерирования управляющих слов, которые подлежат хранению в управляющем ЗУ. Эта программа, называемая микроассемблером, обычно заносит набор, определенный по умолчанию, в поля, не указанные в данном слове. С учетом сказанного первое управляющее слово процедуры умножения может быть записано в виде

ПРОГРАММА ОБРАБОТКИ ПОКАЗАТЕЛЕЙ СТЕПЕНИ

Управляющая программа для обработки показателя степени аргумента 1 может иметь следующий вид:

EXPO1:	LMI(R8)	K=0	FF1	MAROUT	LATCHA	ZEROTEST	JFL(NEXTEXPO NEXTSTEP)	
NEXTSTEP:	CSA(AC)	K=0	FF0	RAMSEL	LATCHB	READRAM	ARG1	JCR(NXXT)
NXXT:	NOP	-	-	RAMSEL	SUM	WRITERAM	ARG1	JCR(NXXXT)
NXXXT:	SDR(AC)	K=0FH	FF0	+ 1	LATCHB	ZEROTEST	CARRY	JFL(NXTX1 ADDCARRY)
ADDCARRY:	NOP	-	-	RAMSEL	LATCHA	READRAM	ARG1	JCC(NYYT)
NYYT:	NOP	-	-	RAMSEL	SUM	WRITERAM	ARG1	JCR(NXTX2)
NEXTEXPO:	-	-	-	-	-	-	-	JCC(NXTX2)

Из рис. 7.34 следует, что первая команда обусловливает размещение NEXTEXPO в столбце 2, строка m, в то время как команда NXXXT определяет ее размещения в столбце 2, строка n, а команда NYYT — в той же строке, что и сама NYYT. Хотя обычно имеется возможность выбрать такое размещение

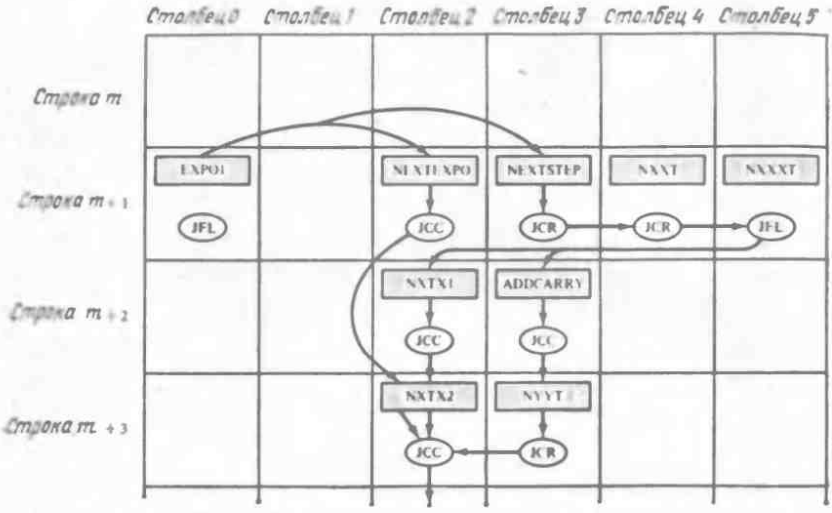
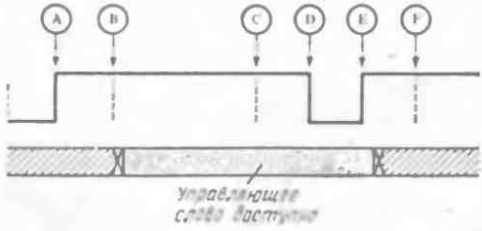


Рис. 7.34. Диаграмма программы обработки показателя степени в двумерном пространстве.

команды NYYT, чтобы позволить прямую передачу управления NEXTEXPO, нельзя передать управление одной и той же ячейке с помощью двух различных ветвей команды JFL. Поэтому необходимо дублировать значения NEXTEXPO для каждой команды JFL, что и сделано с помощью NXTX1 и NXTX2 (рис. 7.34).

СИНХРОНИЗАЦИЯ СИСТЕМЫ

В проектируемой системе используется однофазный генератор тактовых импульсов на ТТЛ-схеме. Ниже показана форма сигнала и описаны соответствующие части каждого микроцикла.



A — Начало микроцикла. Адрес строки — столбца находится в регистре адреса микропрограммы устройства 3001 и передается в управляющее ПЗУ. Соответствующее управляющее слово считывается из ПЗУ.

B — Управляющее слово появляется на выходе управляющего ПЗУ. Задержка от A до B определяется временем работы

дешифратора адреса и конечным временем установления сигнала и может находиться в диапазоне 30—100 нс и более. Именно с этого момента начинается выполнение операции в устройстве 3002 и (или) других управляемых подсистемах.

C — К этому моменту результаты операции, которые будут проверяться с помощью линии ввода флага, должны стать устойчивыми. Интервал времени B — C определяется временем работы устройства 3002, а также временем распространения переноса между всеми разрядно-модульными элементами. Если для других управляемых схем требуется большее время работы, то интервал B — C будет определяться этим временем. Во всяком случае, этот период должен быть достаточно длительным, чтобы появление результата в момент C опережало момент D на необходимое время установки C — D. Обычно он составляет величину порядка 15 нс.

D — По заднему фронту синхроимпульса происходит загрузка результата операции в регистр результата устройства 3002. В тот же момент начинается вычисление следующего адреса в устройстве 3002 в соответствии с командой управления адресом, считываемой из ПЗУ, и всеми значениями условий, необходимых для функции управления адресом. Кроме того, снижение уровня сигнала может быть использовано для генерирования управляющих сигналов типа WRITERAM. Минимальная продолжительность интервала D — E составляет $\sim$30 нс.

E — По переднему фронту синхроимпульса начинается следующий микроцикл, состоящий из описанных выше этапов.

F — Эта точка указывает момент времени, когда управляющее слово предыдущего цикла замещается управляющим словом текущего цикла.

С учетом сказанного рассмотрим первое управляющее слово:

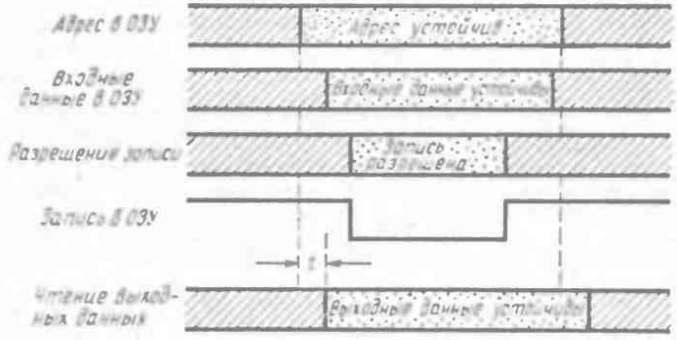
MPY: ILR (IINDEX) RAMSEL READRAM LATCHB ARG1

Целью этой операции является считывание разряда b_i из регистра ARG1 ОЗУ и передача его в Σ-ПЗУ через фиксатор B. Микрокоманда ILR по условию увеличивает и загружает аккумулятор из регистра-источника в соответствии с определением операции, приведенным в табл. 7.2.

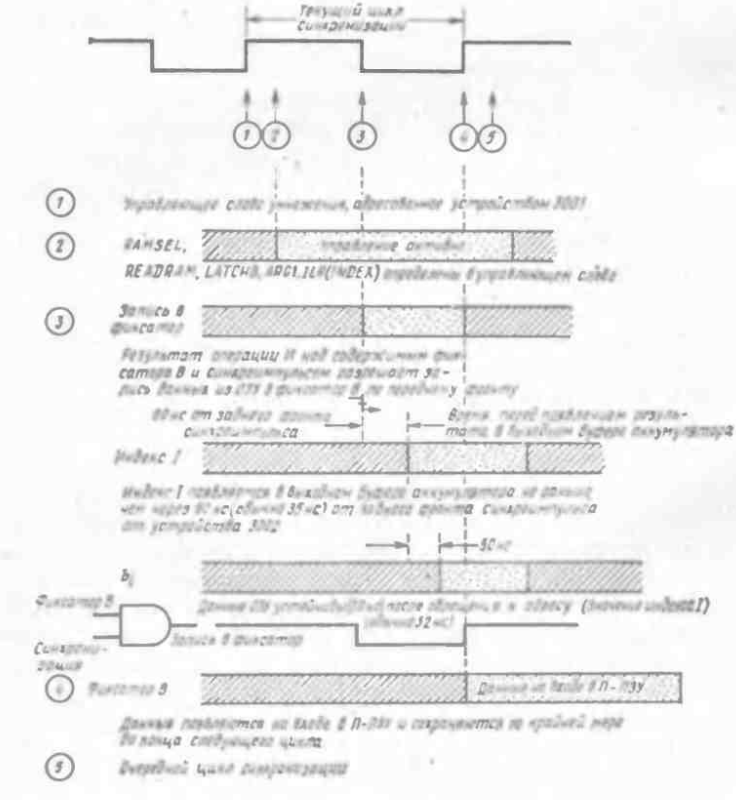
$$\text{ILR}\,(R_n) \quad (K = 0) \quad R_n + C_{In} \longrightarrow R_n, AC$$

Таким образом, значение переноса с линии ввода складывается с содержимым регистра n, а результат считывается в аккумулятор и посылается обратно в регистр R_n. Следовательно, эта команда является средством либо загрузки аккумулятора адресом разряда, либо увеличения индекса в момент помещения его на адресную шину ОЗУ. Как указывалось выше, управление линией ввода переноса осуществляется с помощью линии вывода флага. Таким образом, значение поля вывода флага FF0 должно быть прямо указано или приниматься по умолчанию.

Анализ временных соотношений ОЗУ проведем на примере запоминающего устройства произвольного доступа SN7489 (16×4 бит) фирмы Texas Instruments, временна́я диаграмма работы которого приведена ниже:



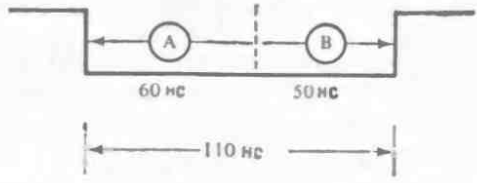
Минимальная длительность импульса записи, поступающего в ЗУ 7489, должна составлять 40 нс. Генератор тактовых импульсов системы должен удовлетворять этому требованию.



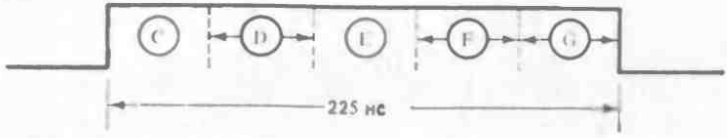
Анализ приведенных выше временны́х соотношений показывает, что минимальная длительность синхроимпульса определяется следующими параметрами:

A — Временем от заднего фронта синхроимпульса до появления содержимого аккумулятора (адреса в ОЗУ) на выходе — 60 нс.

B — Временем от момента выдачи адреса в ОЗУ до момента выдачи данных из ОЗУ — 50 нс.



Максимальная длительность синхроимпульса определяется следующими параметрами:



C — Временем от переднего фронта синхроимпульса до установления устойчивого состояния адреса микропрограммы в устройстве 3001—45 нс.

D — Временем доступа к ПЗУ — от ввода адреса до получения данных — 50 нс.

E — Временем подготовки к работе устройства 3002—65 нс.

F — Временем распространения переноса (два этапа) — 50 нс.

G — Временем формирования переноса — 15 нс.

Итак, уровень синхросигнала должен оставаться высоким по крайней мере 225 нс. Тогда общее время цикла синхронизации для данной системы составит 335 нс. Следовательно, новое управляющее слово будет появляться каждые 335 нс. Скорость выполнения микрокоманд, составляющая ~3 млн. микрокоманд, может быть увеличена с помощью так называемой конвейерной обработки, при которой выборка следующей команды совмещается с выполнением текущей. Второе управляющее слово во многом похоже на первое, так как оно используется для передачи разряда множимого в П-ПЗУ. Первая и вторая команды записываются следующим образом:

```
MPYBI:  ILR (IINDEX)  RAMSEL  ARGI  READRAM
                LATCHB  JMP (MPYAJ)
MPYAJ:  ILR (JINDEX)  RAMSEL  ARGI  READRAM
                LATCHB  JMP (SAVPCY)
```

Следующая команда обеспечивает сохранение переноса произведения в аккумуляторе временного хранения устройства 3002 (PCARRY$\leqslant$8). Это действие осуществляется путем помещения переноса на системную шину и ввода в устройство 3002 через порт ввода памяти. Отметим, что для управления линией C_{in} по умолчанию принимается значение FF0.

SAVPCY: LTM (T) CARRYENABLE JMP (SETUPRODUCT)

На следующем шаге произведение $a_j \cdot b_i$ перемещается во входной фиксатор А для использования в П-ПЗУ. Кроме того, определяется адрес предыдущей суммы в ячейке i+j частичного произведения. Вывод произведения обеспечивается следующим образом:

SETUPRODUCT: ILR (IPLUSJ) PRODUCTENABLE ⟶ LATCHA

Затем значение OLDSUM считывается из ячейки частичного произведения, адрес которой выдает аккумулятор устройства 3002, и помещается на ввод В П-ПЗУ с помощью команды

RAMSEL READRAM LATCHB

Выходные данные из П-ПЗУ теперь представляют собой сумму Произведение+Предыдущая сумма. Значение переноса суммы на выходе П-ПЗУ будет складываться с содержимым аккумулятора временного хранения. В последнем находится перенос произведения, полученный от умножения a_j на b_i. Операция сложения выполняется по команде

ACM (T) CARRYENABLE

При этом значение переноса помещается на системную шину, а затем из порта данных памяти оно прибавляется к содержимому регистра Т. Заметим, что при сложении значения переноса суммы ($\leqslant$1) с предварительно загруженным в регистр Т переносом произведения ($\leqslant$8) обеспечивается получение суммы в двоично-десятичной системе. Далее величина Произведение+ +Предыдущая сумма передается на ввод А П-ПЗУ, а значение предыдущего переноса поступает в РАП устройства 3002 (но без разрешения на выдачу) с помощью команды

LMI (LASTCARRY) PRODUCTENABLE ⟶ LATCHA

После фиксирования суммы на вводе А отпирается РАП устройства 3002, в результате чего предыдущий перенос помещается на системную шину и фиксируется на вводе В П-ПЗУ. В то же время адрес ячейки памяти в области хранения частичного произведения вновь загружается в аккумулятор.

ILR (PLUSJ) MAROUT ⟶ LATCHB

В результате получена сумма значения предыдущего переноса с выражением Произведение+Предыдущая сумма. Теперь

должна быть выполнена запись результата в область частичного произведения, модификация значения предыдущего переноса, установка индексов. Новая сумма записывается в область частичного произведения по команде

RAMSEL PRODUCTENABLE ⟶ WRITERAM

Перенос, полученный при последней операции, подается на ввод Па, а содержимое регистра Т в устройстве 3002 поступает в РАП по команде

LMI(T) CARRYENABLE ⟶ LATCHA

Затем временный перенос из регистра адреса памяти подается на шину и фиксируется на вводе Пb по команде

MAROUT ⟶ LATCHB

Таким образом, произведено сложение переноса суммы с временным переносом. Их сумма теперь выводится на системную шину и загружается в аккумулятор устройства 3002:

LTM(AC) PRODUCTENABLE

Этот общий перенос и является числом, представляющим предыдущий перенос в операции. Следовательно, он должен быть записан в область хранения предыдущего переноса в устройстве 3002:

SDR(LASTCARRY) FFI

Ниже приводится общий алгоритм процедуры умножения и соответствующая ему микропрограмма:

1. Передать a_j и b_i на выводы Па и Пb П-ПЗУ.
2. Выдача П-ПЗУ Произведение+Перенос, запись переноса произведения в регистр Т.
3. Передать произведение на ввод Па П-ПЗУ.
4. Передать предыдущую сумму (=IPLUSJ) на ввод Пb П-ПЗУ.
5. Зафиксировать значения временной суммы и временного переноса, полученные от П-ПЗУ.
6. Прибавить временный перенос к регистру Т, содержащему теперь Перенос произведения ($\leqslant 8$)+Временный перенос ($\leqslant 1$).
7. Передать временную сумму (=Произведение+Предыдущая сумма) на ввод Па.
8. Переслать значение предыдущего переноса из устройства 3002 на ввод Пb.
9. Записать сумму, полученную из П-ПЗУ, в ячейку IPLUSJ в ОЗУ.
10. Сложить перенос суммы из П-ПЗУ с содержимым регистра Т.

11. Записать результирующее значение переноса в область хранения предыдущего переноса в устройстве 3002.

```
MPYBI:      ILR(IINDEX)   RAMSEL      ARG1   READRAM  LATCHB   JMP(MPYAJ)
MPYAJ:      ILR(JINDEX)   RAMSEL      ARG2   READRAM  LATCHA   JMP(SAVPCY)
SAVPCY:     LTM(T)        CARRYENABLE
SETUPRODUCT: ILR(IPLUSJ)  PRODUCTENABLE  •                LATCHA
             -----        RAMSEL            READRAM       LATCHB
             ACM(T)       CARRYENABLE
             LMR(LASTCARRY) PRODUCTENABLE •                LATCHA
             ILR(IPLUSJ)  MAROUT            •              LATCHB
             -----        RAMSEL    PRODUCTENABLE  •       WRITERAM
             LMI(T)       CARRYENABLE       •              LATCHA
             LTM(AC)      PRODUCTENABLE     •
             SDR(LASTCARRY) FF1
```

Таким образом, разработана микропрограмма повторяющейся части цикла умножения, показанного на рис. 7.35.

Из рис. 7.35 видно, что следующий шаг, который необходимо выполнить, заключается в увеличении и проверке индекса J. Увеличение производится командой

```
ADJUSTINDEX:  INR (JINDEX)   FF1   JMP (TESTJINDEX)
```

Значение индекса J сравнивается с его максимальным значением следующим образом. К индексу J прибавляется число 4, и проверяется линия вывода переноса. Если $J \leqslant 11$, то перенос отсутствует и выполняется следующая операция умножения, в противном случае значение предыдущего переноса должно быть прибавлено к новой сумме, индекс J установлен в нуль, а индекс I увеличен и проверен. Проверка индекса J может производиться следующим образом. Когда множимое загружается в ОЗУ, производится подсчет числа разрядов множимого. Затем это число переводится в отрицательное и записывается как JMAXNEG. Каждый раз после увеличения индекса J производится прибавление JMAXNEG к новому значению индекса J и проверяется линия вывода переноса устройства 3002 на наличие заема. Если последний имел место, то осуществляется возврат в начало цикла для вычисления следующего члена произведения $a_{J+1} \cdot b_I$; в противном случае предыдущий перенос проверяется на нуль.

Основная трудность, связанная с этим подходом, заключается в изменении значения JINDEX при проверке. Эта трудность может быть преодолена, если учесть, что изменение уровня синхросигнала с высокого на низкий используется для фиксации результатов текущей операции в устройстве 3002 в регистре-получателе. Если такого изменения не происходит, эти же ре-

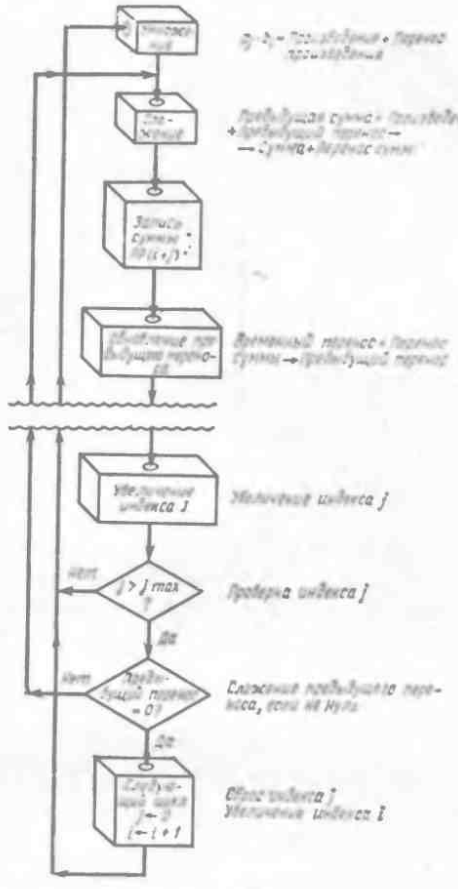


Рис. 7.35. Блок-схема алгоритма умножения с плавающей точкой.

зультаты, в том числе перенос, появляются на выходе АЛУ, однако они не будут сохранены. Поскольку в данном случае необходимо проверять только линию вывода переноса, можно останавливать генератор тактовых импульсов устройства 3002, запирая его с помощью линии управления на один цикл ($\overline{\text{INH}}$), как показано на рис. 7.36. Нормальное функционирование устройства 3002 восстанавливается в следующем цикле.

Если индекс J достиг своего максимума, необходимо проверить значение предыдущего переноса. Если оно не равно нулю, то производится его сложение с частичным произведением посредством передачи управления в ту область цикла, где выполняется сложение. В противном случае значение индекса J множимого устанавливается равным нулю, а индекс I множителя

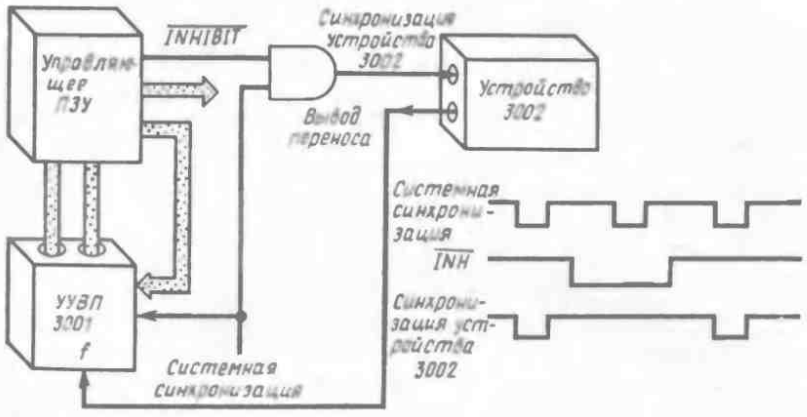


Рис. 7.36. Использование поля «Запрет тактового импульса».

увеличивается и проверяется. Проверка значения предыдущего переноса может выполняться внутренними средствами устройства 3002 с помощью команды проверки нулевого регистра, определяемой как

$$TZR\,(R_n) \quad (K = 1 \ldots 1) \quad C_{in} \vee R_n \longrightarrow C_0 \quad R_{n\cdot} \longrightarrow R_n$$

где способность устройства 3002 выполнять операцию ИЛИ над словом применяется по отношению к линии вывода переноса. Таким образом,

```
                                        (управление мультиплексором)
TESTLASTCARRY: TZR (LASTCARRY)  TESTCARRY JFL (NEXTI, NEWSUM)
NEXTI:         INR (INDEX)      FFI      JMP (TESTIINDEX)
```

Как и при проверке индекса J, необходимо отыскать значение (IMAX) и прибавить его к новому значению индекса I, не разрушая последнее. Если заема не происходит, работа цикла заканчивается; в противном случае JINDEX устанавливается в нуль и цикл продолжается для вычисления произведения $a_0 \cdot b_{I+1}$.

Необходимо подчеркнуть, что структура регистрового ЗУ вновь изменилась. Вместо регистров переноса произведения, переноса предыдущего произведения, переноса суммы и переноса предыдущей суммы используется регистр T для текущей суммы и новый регистр предыдущего переноса. Из трех доступных при такой структуре регистров два требуются для хранения величин JMAXNEG и IMAXNEG. Таким образом, остается еще один сво-

бодный регистр. Для завершения программы цикла запишем

```
TESTINDEX:  ILR (IINDEX)  FFI  JMP (TSTII)
TSTII:  ADR (IMAXNEG)  INH  TESTCARRY  JFL (LOOPEXIT, LOOPBACK)
```

По первой команде производится увеличение индекса I и его загрузка в аккумулятор. Вторая команда определяется следующим образом:

$$\mathrm{ADR}(R_n) \quad (K = 1 \ldots I) \quad AC + R_n + C_{In} \longrightarrow R_n$$

т. е. содержимое регистра складывается с содержимым аккумулятора и результат записывается обратно в регистр. Другая микрокоманда $\mathrm{ALR}(R_n)$ оставляет копию результата в аккумуляторе.

Как указывалось ранее, изменение значения IMAXNEG нежелательно. Поэтому с помощью линии управления INH осуществляется запирание генератора тактовых импульсов устройства 3002 на один микроцикл. При этом определяется разряд заема, который далее проверяется с помощью команды JFL переключением мультиплексора на канал TESTCARRY. Если происходит заем, то CY=1, увеличенное значение аккумулятора переписывается в ячейку индекса I и цикл продолжается; в противном случае осуществляется выход из цикла.

```
LOOPBACK:  SDR (IINDEX)  FFI  JMP (CLRJ)
CLRJ:  CLR (JINDEX)  JZR (MPYBI)
LOOPEXIT: переход к процедуре формирования выдачи результатов
```

ЗАКЛЮЧЕНИЕ

В данной главе наиболее подробным образом по сравнению с другими главами изложена разработка проекта. Если в других главах проекты приводились для иллюстрации отдельных концепций или принципов, то здесь показан реальный процесс проектирования. Цель проектирования — разработка процессора с десятичной плавающей точкой для выполнения арифметических вычислений. Первый шаг заключается в рассмотрении возможных путей построения устройства. Выбираются основание системы счисления и вид представления чисел, обсуждаются операции синтаксического анализа входных строк. Строятся диаграммы изменения состояния и разрабатывается метод доступа к данным. На основании анализа диаграмм состояний выбирается устройство управления выполнением программы, обеспечивающее возможность реализации этих диаграмм. Описан новый подход к декодированию символов кода ASCII и разработаны основывающиеся на нем диаграммы изменения состоя-

ния. После этого в соответствии со схемой декодирования спроектировано устройство управления выполнением программы.

После определения потока команд управления необходимо разработать процедуры обработки данных. В данном случае была выбрана табличная схема обработки и определены соответствующие ей основные элементы архитектуры системы. Затем был построен алгоритм табличного умножения и разработана блок-схема этой процедуры. С использованием упомянутой блок-схемы были определены пути передачи данных и линии управления и выбраны элементы обработки данных. Заключительная часть проектирования состоит из разработки микропрограммы для управляющей памяти, реализующей алгоритм при выбранной архитектуре, и исследования временны́х соотношений. Хотя порядок проектирования в главе изложен как последовательный, в действительности, конечно, многие этапы выполняются параллельно.

СИСТЕМА КОДИРОВАНИЯ И ДЕКОДИРОВАНИЯ ДВУМЕРНЫХ ИЗОБРАЖЕНИЙ

Предмет теории кодирования является слишком обширным и не может быть изложен в рамках одной главы книги. Но мы будем рассматривать сугубо специальный вопрос приложения теории кодирования, важность которого все возрастает, а именно кодирование и декодирование двумерных изображений в системе факсимильной связи. Визуальная информация, как правило, передается с использованием обычных носителей и каналов связи, и, следовательно, затраты на ее передачу пропорциональны количеству передаваемых информационных битов. В связи с этим желательно по возможности уменьшать избыточность передаваемой информации. Хотя избыточность часто используется в качестве средства сохранения целостности данных, природе передаваемых документов, как известно, практически присуще несколько видов избыточности, в частности контекстуальная избыточность, и таким образом желательно кодировать информацию об изображении по возможности эффективно. В настоящей главе обсуждается проект системы кодирования, реализованной с применением схем с разрядно-модульной организацией.

ПЕРЕДАЧА ДВУМЕРНЫХ ИЗОБРАЖЕНИЙ

В то время как в фотографии применяются параллельная передача и параллельно записывающие механизмы, т. е. фотопленка, электронная передача изображений обычно производится последовательным способом, как в известных системах передачи телевизионного изображения. В таких системах двумерное изображение подвергается декомпозиции в упорядоченную последовательность одномерных образов, при этом ширина электронного сканирующего луча считается равной нулю. Затем эти

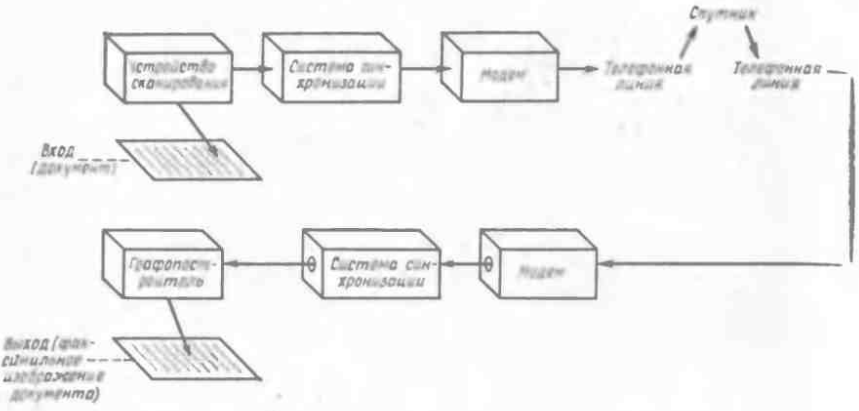


Рис. 8.1. Типичная система передачи документов.

одномерные образы передаются последовательно, а при приеме производится реконструкция (восстановление) двумерных изображений. В то время как нормальное телевизионное изображение имеет много уровней тона, для представления типичного делового документа достаточно использовать только два уровня тона — черный и белый. В дополнение к двухуровневой градации тона документ содержит значительный объем избыточной визуальной информации. Таким образом, использование методов сжатия данных для сокращения визуальной избыточности и применение процедур восстановления изображения на приемном конце весьма целесообразны при передаче визуального образа документа.

Минимальный состав системы передачи изображения включает устройство сканирования, графопостроитель и подсистему связи. В подсистему связи входят подсистема синхронизации и информационный канал. В типичном информационном канале имеется *модем*, т. е. модулятор-демодулятор, который для обеспечения передачи по телефонным каналам преобразует сигнал по методу двоичного тонового кодирования. Схема такой системы представлена на рис. 8.1.

ПРОТОКОЛ ПЕРЕДАЧИ ИЗОБРАЖЕНИЯ

Рассмотрим процедуру автоматической передачи изображения, типичную для большинства синхронных систем. Сначала производится подача вызова, а затем вызванный номер проверяется посредством подачи от удаленного абонента сигнала под-

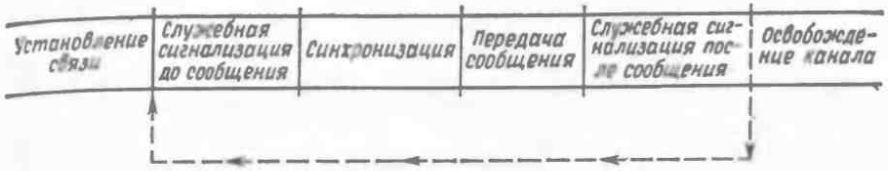


Рис. 8.2. Протокол передачи изображения.

тверждения установления связи частотой 1500 Гц. Информация о скорости приема задается с помощью последовательности сигналов 1100 Гц, подаваемых в течение определенных интервалов времени. После установления канала передатчик передает сигнал синхронизации, используемый для выравнивания границ. Затем информация документа передается таким образом, что сканирующее устройство и графопостроитель синхронно переходят к следующей строке документа. Признаком конца документа служит сигнал частотой 1100 Гц и длительностью 3 с. Затем линия автоматически разъединяется, и машина оказывается готовой к следующему вызову. Такая процедура передачи используется в устройствах фирмы Хегох, однако большинство схем синхронной передачи изображения имеет аналогичный протокол передачи. Типичный протокол такой передачи представлен на рис. 8.2.

Простая схема синхронной передачи не рассчитана на использование сжатия данных. В этой схеме нет взаимно однозначного соответствия между элементами изображения и передаваемыми битами. По этой причине система должна быть асинхронной и иметь буферы для хранения входных и выходных закодированных данных. На рис. 8.3 изображена схема, дающая общее представление об архитектуре такой системы. Универсальное устройство сжатия — восстановления данных, предназначенное для обеспечения асинхронной работы, является основной частью системы передачи изображения.

Универсальная система сжатия — восстановления (УССВ) управляется устройством управления (УУ), которое, в частности, выполняет следующие функции:

начальную установку УССВ посредством подачи сигнала RESET (по этому сигналу в УССВ производится сброс внутренних регистров и переход в состояние ожидания следующей команды);

передачу в УССВ команд, определяющих алгоритм, который должен быть использован при работе УССВ;

передачу в УССВ параметров, которые определяют режимы его работы, описанные ниже.

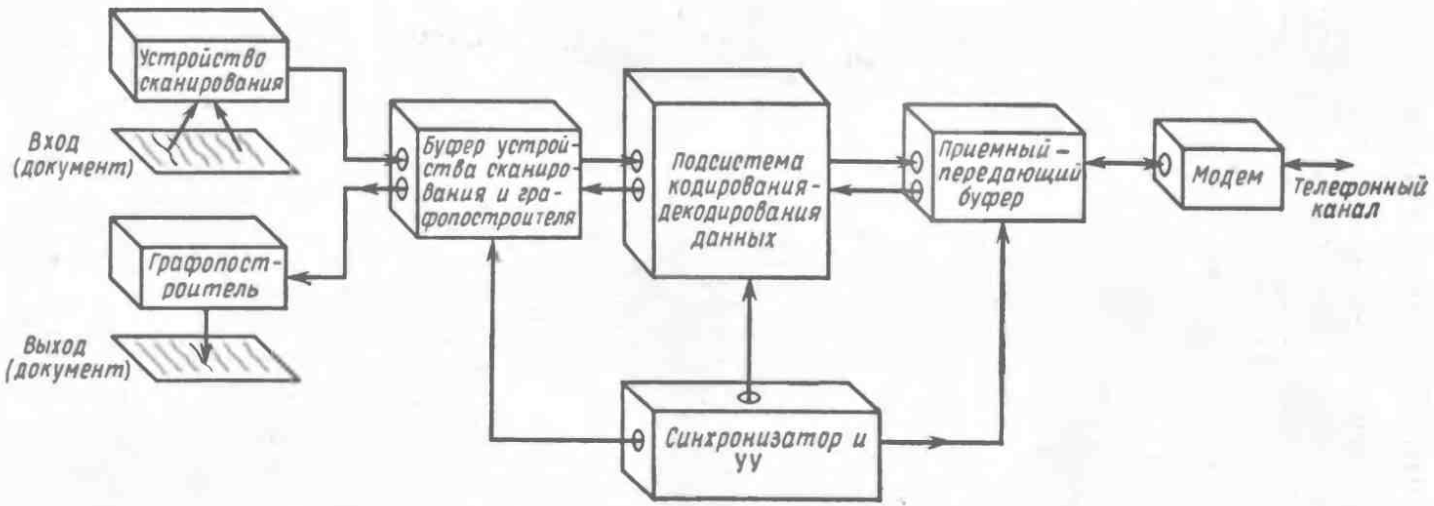


Рис. 8.3. Схема системы сжатия данных и их последующего восстановления.

АРХИТЕКТУРА УССВ

Архитектуру УССВ определяют следующие критерии:

1. Стоимость реализации системы, которая должна быть возможно более низкой.

2. Гибкость системы, благодаря которой возможности системы могут быть впоследствии расширены. Это достигается посредством введения новых алгоритмов, а также изменением скорости передачи или других желаемых параметров.

Гибкость системы может быть достигнута путем использования принципа модульного построения системы. Уменьшение стоимости реализации достигается при использовании современной технологии, которая обеспечивает системе бóльшие функциональные возможности на единицу стоимости по сравнению с более ранними технологиями. Следовательно, возможно спроектировать систему, удовлетворяющую как первому, так и второму критериям. Для реализации системы будут использоваться однокристальные микро-ЭВМ, УУВП, управляющие ПЗУ на БИС, процессорные элементы на БИС. Разбиение системы на подсистемы должно основываться на следующих критериях:

1. Необходимая начальная установка состояния подсистем должна производиться под управлением однокристальной микро-ЭВМ.

2. Кодирование и декодирование данных должно выполняться высокоскоростными компонентами системы с применением микропрограммного управления.

Таким образом, система будет подразделяться на две основные подсистемы: подсистему связи и управления (ПСУ), связанную в системе с устройством управления, и подсистему кодирования-декодирования (ПКД), которая соединена с входным и выходным буферами. Эти две подсистемы представлены на рис. 8.4.

Для реализации ПСУ можно воспользоваться однокристальной микро-ЭВМ. С этой целью может быть использована любая микро-ЭВМ общего назначения, предназначенная для работы в режиме контроллера, например микро-ЭВМ 3870 фирмы Mostek, 8048 фирмы Intel, Z8 фирмы Zilog и др. Не отдавая в нашем проекте предпочтения ни одной из указанных микро-ЭВМ, будем основывать свой выбор, если другие соображения не окажутся существенными или решающими, на общих экономических факторах. Исходя из стоимости, приемлемости архитектуры и доступности, выберем микро-ЭВМ 8748 фирмы Intel, имеющую ППЗУ с возможностью стирания информации, которая аналогична микро-ЭВМ 8048. Подсистема сжатия — восстановления будет реализована с использованием устройств с разрядно-модульной организацией и применением микропрограммирования, выполненных на основе серий 3000 фирмы Intel, 2900 фирмы

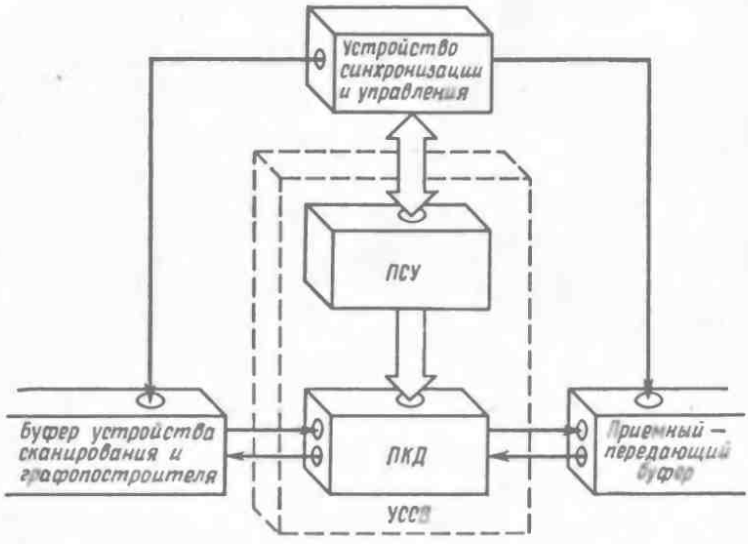


Рис. 8.4. Универсальная система сжатия данных и их последующего восстановления, состоящая из подсистемы связи и управления и подсистемы кодирования-декодирования.

AMD, 74480 фирмы Texas Instruments. Выбор семейства схем с разрядно-модульной организацией на этом этапе преждевременен.

ОРГАНИЗАЦИЯ ИНТЕРФЕЙСА ПОДСИСТЕМ

Интерфейс УССВ и устройства синхронизации и управления реализуется с помощью однокристального процессора. Устройство кодирования-декодирования проектируется в предположении, что оно будет настраиваться на одну из нескольких процедур кодирования. Устройство управления и устройство синхронизации и управления выбирают процедуру кодирования, которая должна быть использована, и информируют об этом ПСУ. Затем ПСУ передает соответствующие параметры в УССВ, реализованную с использованием схем с разрядно-модульной организацией, и выполняет установку начального состояния этой системы. Схема потоков управления представлена на рис. 8.5.

ПАРАМЕТРЫ УПРАВЛЕНИЯ

Универсальная система сжатия-восстановления информации переводится в определенный режим работы с помощью команд, поступающих из УУ. В системе может быть выбрана одна из следующих трех процедур кодирования:

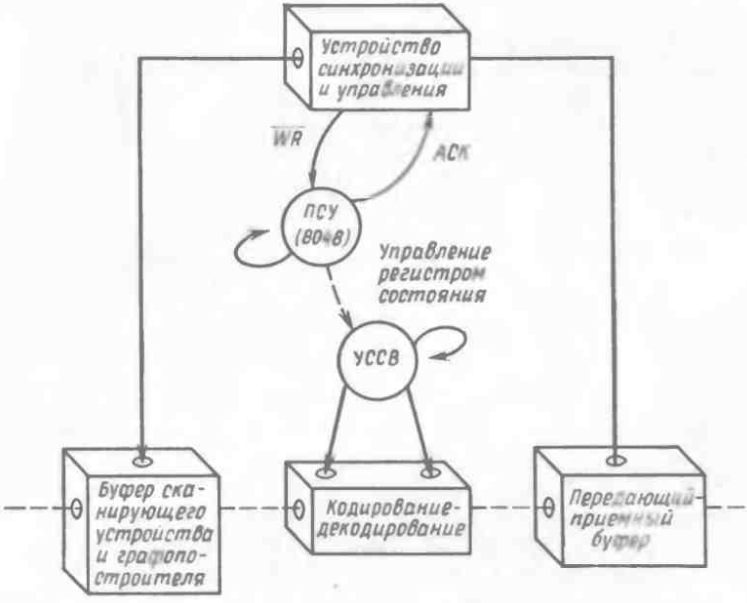


Рис. 8.5. Схема потоков управляющих сигналов, поступающих из синхронизатора в устройство кодирования-декодирования.

1. Процедура адаптивного кодирования DFC-20, используемая фирмой Ricoh Ltd.
2. Расширенная процедура DFC-20.
3. Международная стандартная процедура кодирования, принятая МККТТ.

После установления определенного режима работы УССВ УУ может передать следующие параметры:

1. признак синхронизации (наличие или отсутствие синхронизации строк);
2. число элементов изображения на линию;
3. максимальную длину блока (размер приемо-передающего буфера);
4. длину заголовка;
5. длину кода.

ОПИСАНИЕ ИНТЕРФЕЙСА УУ И ПСУ

Все команды и параметры передаются между УУ и однокристальной микро-ЭВМ, играющей роль ПСУ. Параметры используются системой во время ее функционирования. Следовательно, они должны быть переданы из блока управления ПСУ в рабочие регистры УССВ, которые содержатся в блоке регистров

арифметическо-логического устройства. Общая схема системы представлена на рис. 8.6.

Подсистема связи и управления связывает устройство управления и кодирующее-декодирующее устройство УССВ. Как показано на рис. 8.6, все команды, параметры и информация о состоянии передаются по шине данных и интерпретируются ПСУ согласно сигналам на адресных линиях. На рис. 8.7 представлен адресный интерфейс УУ/ПСУ. Передача данных синхронизируется посредством основных синхросигналов, поступающих из УУ, и вспомогательных синхросигналов, поступающих из ПСУ. При поступлении основного синхроимпульса происходит прерывание ПСУ 8748 в случае, если УУ имеет данные или команды, предназначенные для передачи в УССВ. Для указания того, что производится передача из УУ в УССВ, на линии управления вводом-выводом поддерживается высокий уровень напряжения. Когда по сигналу основной последовательности синхроимпульсов происходит прерывание работы ПСУ 8748, эта подсистема проверяет состояние линии управления вводом-выводом. Если на этой линии обнаруживается высокий уровень напряжения, то система, чтобы получить описание данных, производит ввод адреса по адресной шине. Затем ПСУ вводит данные с шины данных, если это нужно, и вырабатывает сигнал вспомогательной синхронизации, с помощью которого указывается, что данные зафиксированы в ПСУ. Далее, как показано на рис. 8.8, перестает действовать основной синхросигнал УУ. Когда этот факт обнаруживается в ПСУ, она разблокирует си-

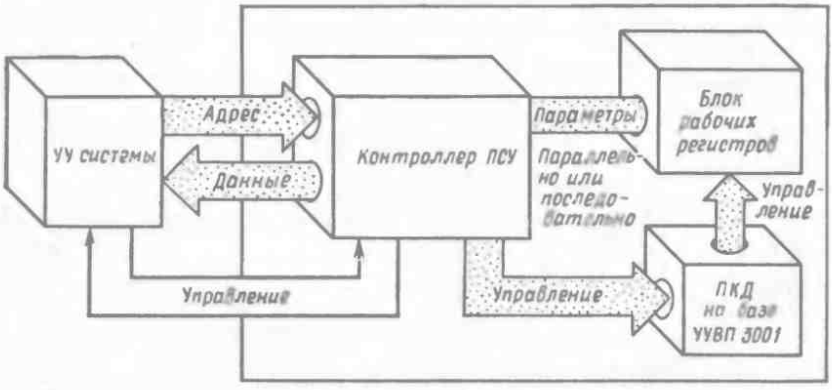


Рис. 8.6. Схема взаимодействия подсистемы связи и управления, реализованной в виде однокристальной микро-ЭВМ, устройства управления и подсистемы кодирования-декодирования. (ПСУ принимает команды и параметры из УУ, выбирает схему кодирования-декодирования, которая должна быть использована, и инициализирует подсистему кодирования-декодирования в УССВ.)

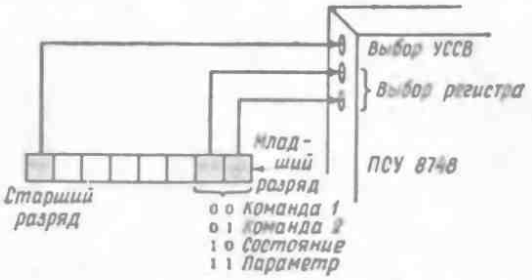


Рис. 8.7. Адресный интерфейс УУ и ПСУ.

стему прерываний; после этого перестает действовать вспомогательный синхросигнал.

Из блока синхронизации и управления УУ в ПСУ могут быть посланы команды следующего типа:

1. Выбрать процедуру кодирования.
2. Принять параметры.
3. Выдать состояние.
4. Начать кодирование.
5. Начать декодирование.
6. Произвести сброс сигналов условия ошибки и центрирование.

Процедуры кодирования были упомянуты выше, более подробно они будут описаны в следующем разделе. Команды начала кодирования или декодирования вырабатываются УУ, который определяет, находится ли система в состоянии приема или передачи. Команды подаются в ПСУ, которая подготавливает высокоскоростную модульную систему кодирования-декодирования, входящую в УССВ. На рис. 8.9 представлена схема, отображающая действия, выполняемые ПСУ. Эти действия состоят в декодировании описателя данных на адресной шине и соответствующей обработке данных.

Представление состояния и параметры зависят от выбираемой процедуры кодирования. Прежде чем приступить к подробному изложению проекта УССВ, обсудим процедуры кодирования.

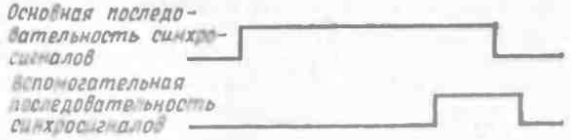


Рис. 8.8. Временная диаграмма основных и вспомогательных синхросигналов.

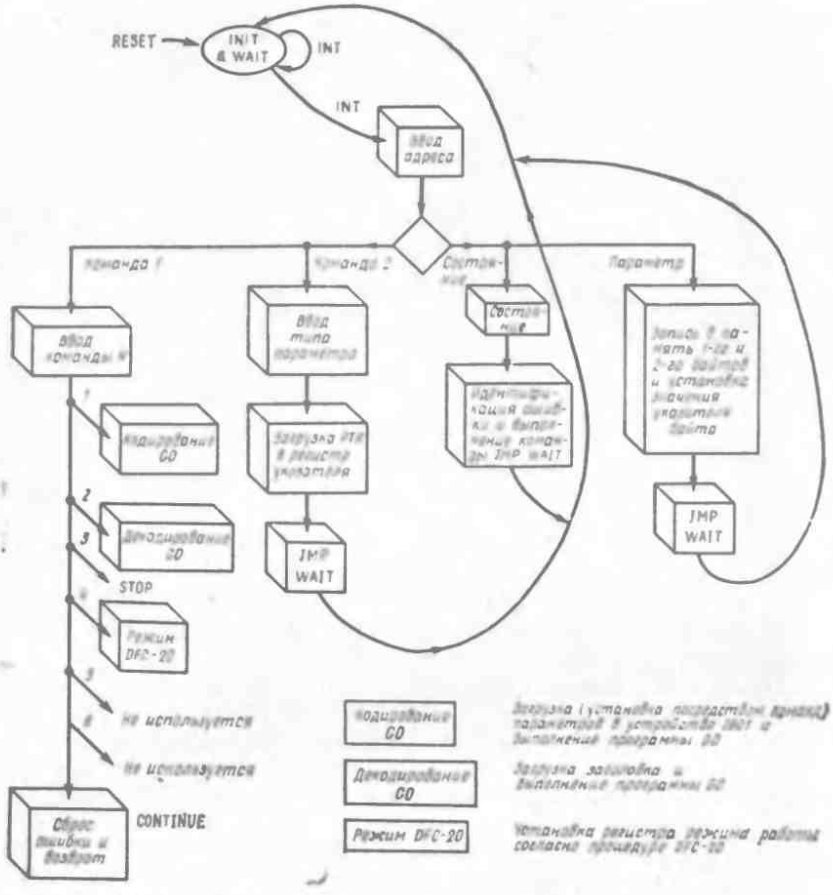


Рис. 8.9. Схема потоков управления в подсистеме связи и управления.

ПРОЦЕДУРЫ КОДИРОВАНИЯ В СИСТЕМЕ ПЕРЕДАЧИ ИЗОБРАЖЕНИЯ

В системах передачи документов обычно используют двоичную шкалу цветового тона, т. е. все изображения передаются с использованием только двух градаций — черного и белого цветов. Такой принцип хорошо подходит для цифровой передачи с помощью ТТЛ-устройств, обеспечивающих последовательную передачу потока нулей и единиц. Хотя возможна передача всего документа в виде потока нулей и единиц, однако тот факт, что большинство документов содержит очень большой процент бе-

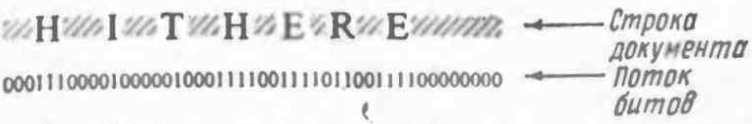


Рис. 8.10. Пример строки документа, которая сканируется и преобразуется в строку из единиц и нулей.

лого поля и соответственно малый процент черного поля наводит на мысль, что желательно применение некоторого способа сокращения избыточности путем сжатия информации при передаче и восстановления изображения на приемном конце. При стандартном механизме сканирования квантуются горизонтальные линии документа. Эти линии последовательно передаются по телефонным каналам как последовательности элементов изображения. На рис. 8.10 приведен пример части документа и соответствующей последовательности передаваемых элементов изображения.

Очевидно, что если нули используются для кодирования белого цвета, а единицы — для кодирования черного цвета, то будет передаваться много длинных строк из нулей. Первый подход к сокращению количества передаваемых нулей состоит в подсчете количества подряд идущих нулей и передаче этого числа вместо последовательности нулей. Конечно, поскольку для реализации такого метода потребуется использовать счетчик и некоторые другие схемы, то, вероятно, окажется целесообразным и черные элементы изображения передавать аналогичным образом, т. е. целесообразно подсчитывать и единицы, и передавать только их количество. На приемном конце выполняется обратная операция, при этом принятое количество нулей или единиц будет просто использоваться для определения того, сколько белых и черных элементов изображения должно быть репродуцировано на факсимильном изображении. Эта простая процедура кодирования кратко может быть определена следующим образом.

Для каждой непрерывной последовательности черных или белых элементов изображения формируется кодовое слово. Это слово определяет длину серии. Такая процедура кодирования называется *процедурой кодирования длины серии.*

Существует много вариантов рассматриваемой процедуры кодирования. Поскольку литеры шрифтов, используемых для большинства документов, имеют вертикальные линии равной ширины, длины серий элементов изображения, представляющих тонкий слой, поперечный вертикальным линиям литер, обычно являются равными или незначительно отличающимися друг от друга. Кроме того, тот факт, что расстояние между литерами также примерно одинаковое, приводит к появлению серий белых

элементов изображения, состоящих из одинакового количества элементов. Таким образом, получив статистические данные по некоторому количеству документов, можно определить наиболее часто встречающиеся длины серий и поставить им в соответствие наиболее короткие коды. На этом приеме основана процедура кодирования MKKTT, которая является одной из процедур, реализованных в нашей системе.

Если шумы в передающей системе приводят к потере одного или нескольких кодовых слов, то все последующие слова будут сдвинуты на длину элементов изображения, описываемых утраченными словами. Для того чтобы уменьшить влияние шумов, документ обычно передают строка за строкой с построчной синхронизацией, благодаря которой каждая новая строка начинается с определенного места на документе — обычно с его левой границы. Для этой цели пригодны твердотельные одномерные сканирующие устройства. Широко применяемые сканирующие устройства обеспечивают получение 1728 элементов изображения на одной прямой линии. Использование сканирующего устройства такого типа позволяет получить разрешение по горизонтали, равное 79 точкам на сантиметр для типичного документа размером 21,6×28,0 см. Разрешение по вертикали составляет либо 26, либо 39 линий на сантиметр, управление при этом осуществляется устройством управления. Разрешение по горизонтали зависит от характеристик сканирующего устройства и оптических элементов системы. Конструктивно сканирующее устройство напоминает матрицу, описанную в гл. 6, за исключением того, что в сканирующем устройстве используется только одна линия фотоэлементов. Такие матрицы производятся, например, фирмами Reticon и Fairchild, а также фирмами — изготовителями полупроводниковых приборов. Передача линии, проходящей через всю страницу, обеспечивает помехозащищенность, проявляющуюся в почти полной невосприимчивости к пакетам ошибок. Такая помехозащищенность объясняется тем, что сдвиг только одной линии из нескольких линий, необходимых для передачи строки символов, зрительно корректируется человеком. По этой причине системы факсимильной передачи могут функционировать при значительно большем уровне шумов, чем большинство систем передачи данных.

ФОРМАТ БЛОКА ДАННЫХ

В устройстве сжатия и восстановления данных могут быть реализованы два метода синхронизации. При использовании процедуры кодирования DFC-20 перед блоком данных передается заголовок, содержащий управляющую информацию. В случае расширенной процедуры DFC-20 обеспечивается синхрони-

Таблица 8.1. Формат кадра при использовании процедуры кодирования DFC-20

Код синхронизации	24 бит	
Состояние AECU	7 бит	
Число битов данных	10 бит	Заголовок
Позиция элемента	12 бит	
Длина серии черных элементов	3 бит	
Длина серии белых элементов	3 бит	
Данные режима запуска	2 бит	
Закодированные данные	512 бит	
Код проверки	12 бит	

зация линий и информация заголовка не используется. Для нашего проекта воспользуемся процедурой кодирования DFC-20. Формат кадра, передаваемого и принимаемого устройством сжатия — восстановления данных при использовании процедуры кодирования DFC-20, представлен в табл. 8.1. Устройство управления оперирует кодами синхронизации, контроля ошибок и проверки, определяемыми соответствующим полиномом. Устройство сжатия — восстановления данных производит кодирование и преобразование потока единиц и нулей в блок данных, который состоит из чередующихся серий нулей и единиц разной длины. В кадре, соответствующем процедуре кодирования DFC-20, имеется также заголовок, который содержит контекстную информацию. Поля, содержащиеся в заголовке, представлены в табл. 8.2.

В проектируемой нами системе факсимильной передачи информация заголовка принимается УУ и подается в подсистему связи и управления. Эта подсистема использует полученную информацию для настройки подсистемы кодирования-декодирования на выполнение операции кодирования или декодирова-

Таблица 8.2. Поля заголовка кадра при использовании процедуры DFC-20

Имя поля	Длина поля, бит	Содержимое поля
Заголовок	10	Число битов данных
	12	Позиция элемента в текущей строке
	3	Длина серии черных элементов
	3	Длина серии белых элементов
	2	Данные режима запуска (описываются ниже)
Данные	512	Закодированные данные

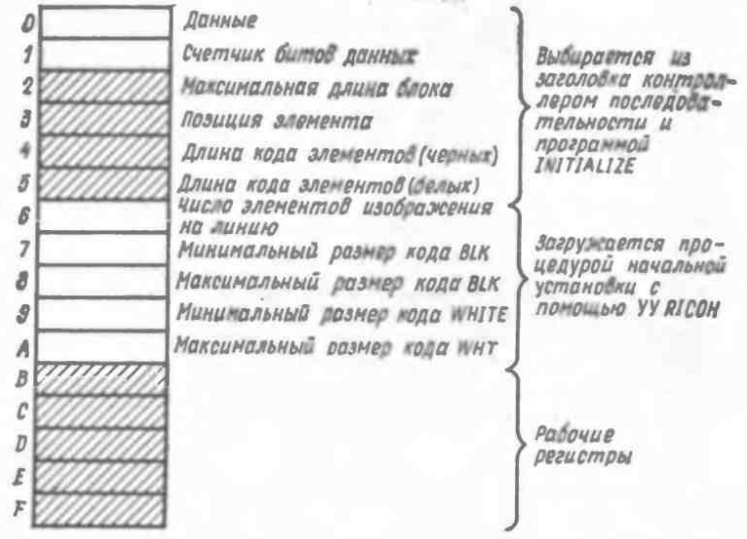


Рис. 8.11. Распределение регистров в кодирующем-декодирующем устройстве универсальной системы сжатия — восстановления.

ния. Если используется формат кадра с заголовком, то проектируемое устройство сжатия — восстановления данных должно выполнять преобразование потока нулей и единиц в длины серий или преобразовывать длины серий в последовательности нулей и единиц при восстановлении изображения. Данные, соответствующие сериям черных и белых элементов, перемежаются и, следовательно, должны храниться в различных регистрах. Для функционирования устройства сжатия — восстановления данных, помимо информации о длине серий, требуется указывать информацию о минимальной и максимальной длинах серий, которые могут встретиться в процессе работы проектируемой системы. Следовательно, необходимо предусмотреть регистры для хранения и этой информации. Вариант распределения регистров устройства сжатия — восстановления данных представлен на рис. 8.11.

В соответствии с рисунком требуется 11 регистров; кроме того, необходимо некоторое количество рабочих регистров. Регистровое АЛУ 3002 фирмы Intel располагает только 10 регистрами. Что касается процессорного элемента 2901, он содержит 16 регистров, которых достаточно для хранения данных в устройстве сжатия — восстановления. На рис. 8.12 изображена схема процессорного элемента 2901. Прежде чем устройство сжатия — восстановления приступит к выполнению кодирования или декодирования данных, подсистема связи и управления по-

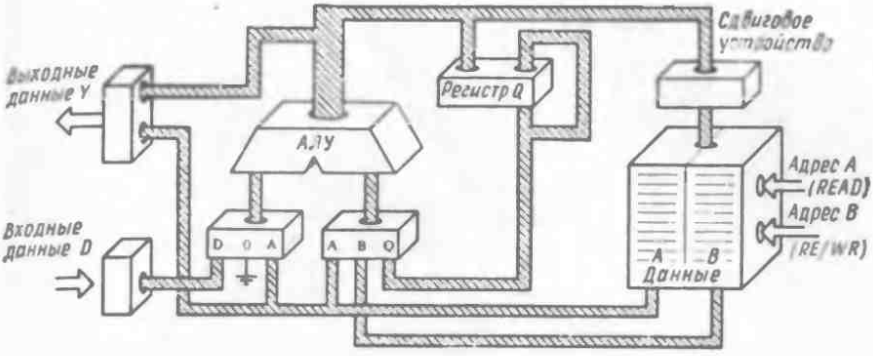


Рис. 8.12. Устройство 2901, используемое в универсальной системе сжатия — восстановления.

лучит из устройства управления заголовок кадра и произведет загрузку определенного регистра устройства сжатия — восстановления данных. На рис. 8.13 представлена блок-схема, отображающая порядок выполнения этой операции.

УПРАВЛЯЮЩИЕ ПОЛЯ УСТРОЙСТВА 2901

Основные функции управления устройства 2901 задаются 17-разрядным управляющим словом. Это слово имеет поля, соответствующие определенным функциям управления устройства 2901; функции управления работой устройства 2901 и соответствующие им управляющие слова описаны в гл. 2. Формат управляющего слова процессорного элемента 2901 имеет следующий вид:

Указание источника операндов АЛУ	Указание функции АЛУ	Указание адресата	Выбор регистра канала A	Выбор регистра канала B	----

Кроме основных управляющих полей, приведенных на рисунке, в устройстве 2901 используется несколько второстепенных управляющих сигналов. Эти сигналы также описаны в гл. 2. Напомним, что они действуют на следующих линиях:

1. Линия F. По состоянию этой линии определяется, был ли равен нулю (F=0) результат текущей операции АЛУ.

2. Линии Q_0 и Q_3 соответствуют значениям младшего и старшего по значимости разрядов регистра Q. Они доступны во время выполнения операции сдвига содержимого регистра Q.

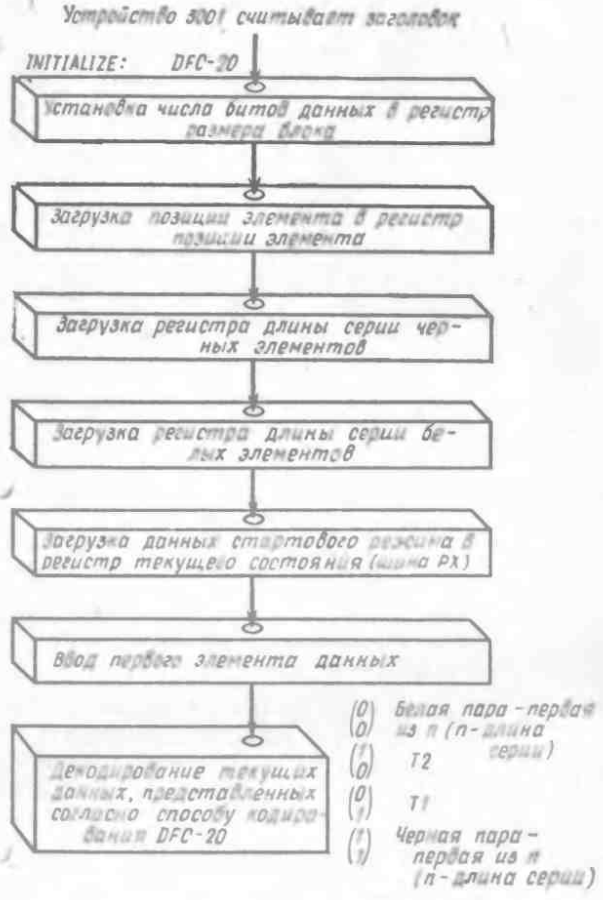


Рис. 8.13. Блок-схема, отображающая процесс загрузки регистров универсальной системы сжатия — восстановления.

3. Линии $СОЗУ_0$ и $СОЗУ_3$ соответствуют значениям младшего и старшего по значимости разрядов регистра сдвига СОЗУ и доступны во время выполнения операции сдвига в регистре сдвига СОЗУ.

ОПИСАНИЕ ИНТЕРФЕЙСА УСТРОЙСТВА СЖАТИЯ-ВОССТАНОВЛЕНИЯ ДАННЫХ И БУФЕРА

В операции передачи изображения всегда участвуют два устройства сжатия — восстановления данных: одно из них является источником, а другое — приемником данных. Одно уст-

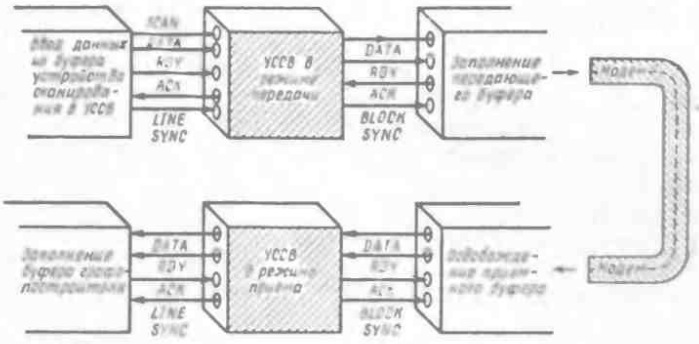


Рис. 8.14. Интерфейс УССВ и буферов.

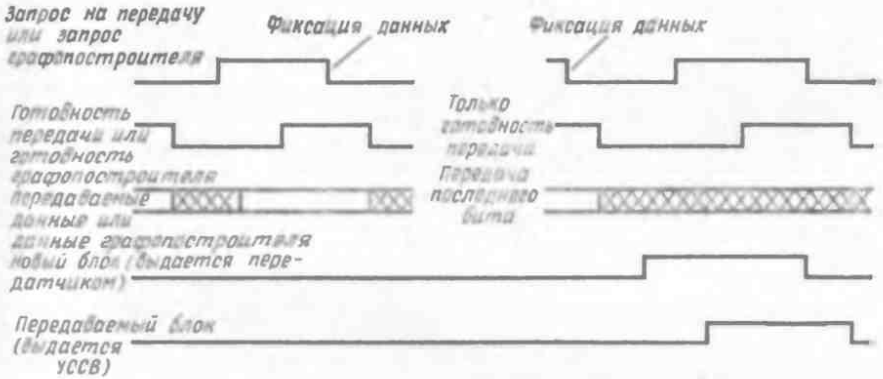


Рис. 8.15. Временные диаграммы управляющих сигналов и пересылки данных из УССВ в буфер передатчика или графопостроителя.

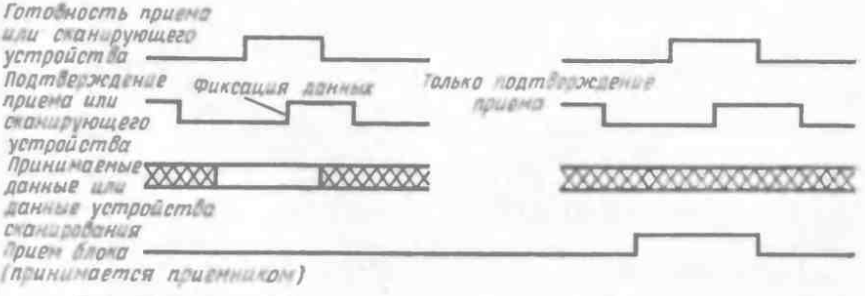


Рис. 8.16. Временные диаграммы управляющих сигналов и пересылки данных из буфера приемника и буфера сканирующего устройства в УССВ.

ройство читает данные из буфера сканирующего устройства, производит сжатие данных согласно выбранному алгоритму и выдает данные в выходной буфер. Приемное устройство считывает данные из приемного буфера, восстанавливает первоначальные данные и выдает данные в буфер графопостроителя. Отметим, что буферы устройства сканирования и графопостроителя идентичны. Они управляются согласно режиму работы устройством управления. То же самое можно сказать относительно передающего и приемного буферов. Устройство управления, кроме того, устанавливает канал, передает данные между приемным и передающим буферами и модемом, вставляет биты синхронизации в передаваемый кадр и проверяет их, обрабатывает состояние ошибки управления и выполняет проверку закодированных данных с использованием полиномов. Из этого описания следует, что устройство сжатия — восстановления непосредственно с модемом не связано. Как показано на рис. 8.14, это устройство связано с буферами сканирующего устройства и графопостроителя передающим и приемным буферами. Временны́е диаграммы рассматриваемых процессов передачи данных представлены на рис. 8.15 и 8.16.

ПРОВЕРКА НАЛИЧИЯ СИНХРОНИЗАЦИИ СТРОК СКАНИРУЮЩЕГО УСТРОЙСТВА

Как показано на рис. 8.17, устройство сжатия — восстановления выполняет подсчет пар элементов. Когда количество пар достигнет 1728, должно проверяться наличие синхронизации строк устройства сканирования. На рис. 8.18 показаны временна́я диаграмма синхронизации строк сканирующего устройства и схема формирования сигнала синхронизации графопостроителя. Устройство сжатия — восстановления находится в состоянии ожидания либо до появления сигнала синхронизации строк сканирующего устройства, либо до тех пор, пока не возникнет сигнал прерывания. Если синхронизация строк обнаруживается, устройство сжатия — восстановления данных производит очистку счетчика позиции элемента и выполняет обработку следующих данных.

СИНХРОНИЗАЦИЯ СИГНАЛОВ СДВИГА И СОСТОЯНИЯ

Признак нулевого результата АЛУ устройства 2901 может проверяться в течение неконвейерного цикла посредством УУВП. Временна́я диаграмма процессорного элемента 2901 в сочетании с действиями, осуществляемыми УУВП для команды

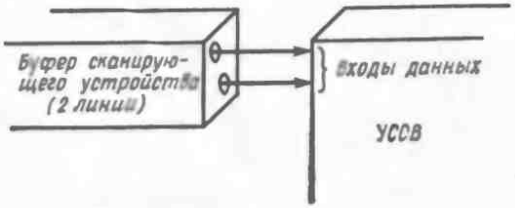


Рис. 8.17. Схема соединения буфера сканирующего устройства и УССВ.

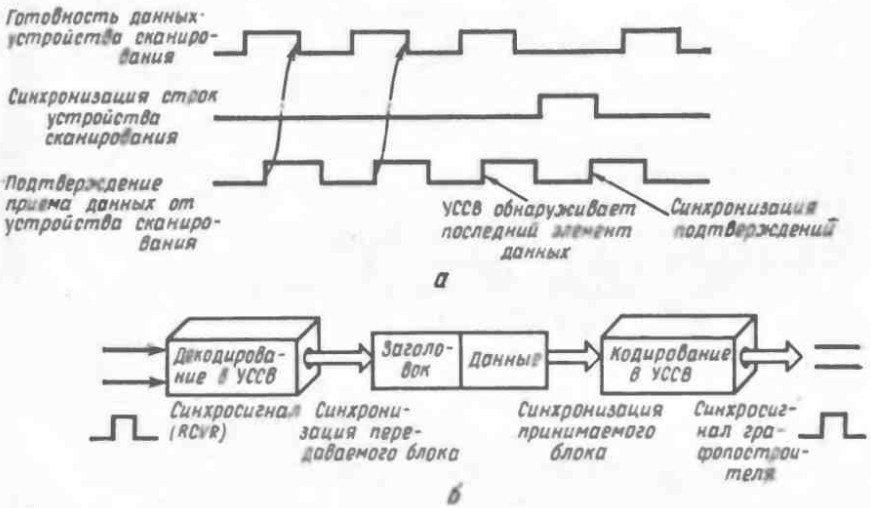


Рис. 8.18. *а* — Временна́я диаграмма синхронизации строк сканирующего устройства; *б* — схема формирования сигнала синхронизации графопостроителя.

JFL, показана на рис. 8.19. Данные вводятся в регистр Q по переднему фронту синхроимпульса. Если выполняется сдвиг, то уровень на соответствующем выходе должен стабилизироваться, прежде чем с него будет снят сигнал. Период стабильности отмечен на рисунке заштрихованной зоной. Как только сигнал на выходе стабилизируется, он может быть проверен контроллером или просто фиксироваться устройством 2901. На временно́й диаграмме, изображенной на рис. 8.19, показано, что данные из процессорного элемента 2901 выдаются из порта СОЗУ$_0$ по возрастающему фронту синхроимпульса. Сигнал управления записью выдается управляющим ПЗУ и деблокирует фиксатор сдвига, в качестве которого используется триггер типа D. В течение текущего периода синхроимпульсов сдвигаемые данные стабилизируются и стабилизируется уровень на линии записи.

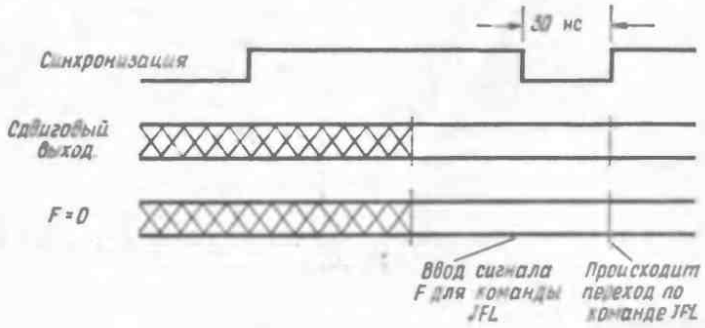


Рис. 8.19. Временна́я диаграмма сигналов на входе F устройства 2901 при выполнении команды JFL устройства 3001.

Как показано на рис. 8.20, по заднему фронту синхроимпульса и при наличии сигнала на линии записи (WRITE) поступает сигнал на вход «Синхронизация» триггера типа D и осуществляется запись в него данных с выхода СОЗУ$_0$. Теперь выход триггера типа D становится доступным для передающего буфера или буфера графопостроителя. Такой тип схемы обеспечивает последовательный интерфейс при передаче данных кодирующего-декодирующего устройства системы сжатия — восстановления

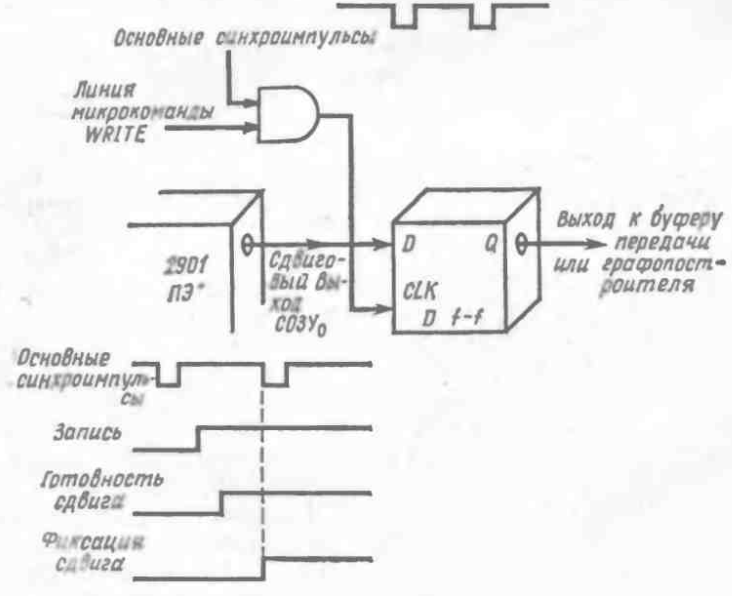


Рис. 8.20. Схема соединения выхода СОЗУ$_0$ устройства 2901 и D-триггера и временна́я диаграмма управляющих сигналов.

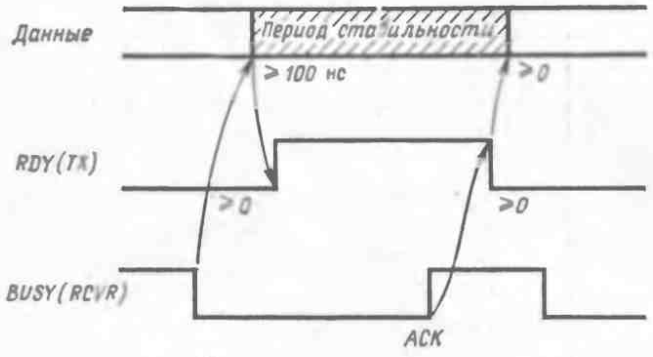


Рис. 8.21. Временна́я диаграмма передачи данных в асинхронном режиме.

данных и передающего буфера или буфера графопостроителя системы.

Выход сдвигового устройства остается активным до тех пор, пока не поступит следующая команда записи из управляющего ПЗУ. Команда SHIFT (СДВИГ) используется для передачи последовательного потока битов к передающему буферу или буферу графопостроителя системы Ricoh. Соответствующие данные устанавливаются и сдвигаются так, как было описано выше. Счетчик циклов организуется в регистре устройства 2901, его содержимое уменьшается и проверяется с помощью микрокоманды. Микрокоманда устанавливает стробирующий сигнал на линии DATA RDY («Данные готовы») с задержкой на один микроцикл и выдает правый бит кода для представления его на выходе СОЗУ$_0$ или Q_0. Временна́я диаграмма асинхронной передачи данных представлена на рис. 8.21.

Стробирующий сигнал RDY («Готово») нарастает после обнаружения приемником сигнала BUSY («Занято»). Выполняется сдвиг, содержимое счетчика сдвига уменьшается на 1 и если оно равно 0, то производится выход из цикла. Если же содержимое счетчика не равно 0, то как показано на рис. 8.22, производится переход к следующему циклу. Стандартная программа установления связи синхронизирует кодирующее-декодирующее устройство для асинхронного канала связи с использованием буферов, и, кроме того, в ней используется двоичный счетчик данных, требуемый для организации поблочной передачи.

Стандартная программа установления связи реализуется микропрограммным путем. Сигналы в устройство сжатия — восстановления данных вводятся на вход мультиплексора, выход которого проверяется посредством УУВП. Сигналы из УССВ представляют собой сигналы управления из управляющего ПЗУ. Поля микрокоманд, которые служат для обеспечения интерфей-

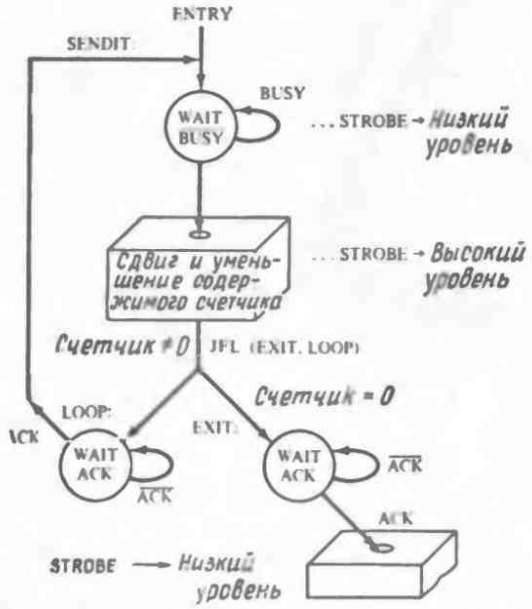


Рис. 8.22. Схема стандартной программы синхронизации кодирующего-декодирующего устройства.

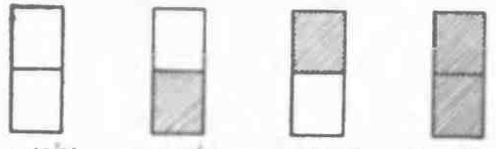
са УССВ и буферов данных, содержат поле выбора канала мультиплексора и сигналы индивидуальной готовности и подтверждения, сгруппированные в поле управления буфером (рис. 8.23).

АЛГОРИТМ DFC-20 ФИРМЫ RICOH

Существуют многочисленные подходы к оптимизации схем кодирования. В некоторых из них для получения более эффективных алгоритмов используют статистические закономерности, выявленные в документах определенного типа. Такой подход базируется на том факте, что линейные изображения и печатный текст в соответствии с данным алгоритмом можно эффективно различать. Алгоритм фирмы Ricoh составлен на основе патента № 3916095, выданного Веберу и др. В нем используется адаптивное кодирование длин серий, при котором элементы данных формируются из элементов изображения, принадлежащих двум последовательным линиям. Следовательно, элементы изображения расположены как бы в вертикальных двухэлементных колонках. В верхней части колонки находится элемент изображения из первой линии, а в нижней части колонки располагается элемент изображения из следующей линии. Кодирование осно-

Рис. 8.23. Управляющие биты микропрограммы.

вывается на двухэлементных колонках данных. Каждая колонка может находиться в одном из четырех состояний, представленных на следующем рисунке:



Терминология, используемая для описания колонок, основана на том, что колонки рассматриваются как пары. Первая колонка называется текущей колонкой — PC (present column), в то время как вторая колонка называется следующей колонкой — NC (next column). Элементы данных в каждой колонке обозначаются в соответствии с линиями, которым принадлежат эти элементы, т. е. элемент изображения из первой линии снабжается индексом 1, а элемент изображения из второй линии — индексом 2. Такая схема обозначений представлена в следующей таблице:

Текущая колонка	Следующая колонка	Линия
PC_1	NC_1	Первая
PC_2	NC_2	Вторая

Ниже термин «элемент изображения» будет использоваться для обозначения некоторой двухэлементной колонки, составленной из элементов изображения, принадлежащих первой и второй линиям. Термин «длина серии» будет использоваться для обозначения длины непрерывной серии либо белых элементов изображения $S_0=[0/0]$, либо черных элементов изображения $S_3=[1/1]$. Понятие «серии» неприложимо к переходным, или комбинированным, элементам изображения $S_1=[0/1]$ и $S_2=[1/0]$. Одинаковые комбинированные элементы изображения, конечно, могут следовать друг за другом, но такие последовательности мы не будем называть сериями. Они будут толковаться так, как описано ниже.

Кодовый терм, соответствующий серии определенной длины, будет содержать первое двоичное кодовое слово, достаточное для представления серии не более чем из 2^n элементов изображения; если общее число элементов изображения в серии превышает 2^n, то терм будет содержать второе кодовое слово длины $n+1$, способное представлять 2^{n+1} дополнительных элементов изображения и т. д. вплоть до максимального допустимого размера кодового слова. Кодовое слово, содержащее в себе только единицы, является условием «продолжения», и, таким образом, другое кодовое слово должно последовать, даже если «заполненное» кодовое слово точно соответствует числу элементов изображения в серии. Обычно следующее кодовое слово должно быть на 1 бит длиннее, чем «заполненное» кодовое слово. Такая процедура подбора длины кодового слова продолжается до тех пор, пока не завершится серия.

Этот метод является адаптивным в том смысле, что размер первого кодового слова каждого очередного кодового терма оп-

ределяется по размеру последнего кодового терма, представляющего длину серии одних и тех же состояний данных. Черные и ортогональные им белые серии оказываются некоррелированными. Размер кодового слова может быть либо увеличен, либо уменьшен.

Сказанное выше поясняет следующий пример. Предположим, что для данной серии черных элементов изображения требуется использовать кодовый терм, включающий в себя 2-битовое кодовое слово (n), 3-битовое кодовое слово (n+1) и 4-битовое кодовое слово (n+2). Следующую серию черных элементов изображения желательно было бы начать с 4-битового кодового слова. Если серия так коротка, что 4-битовое кодовое слово заполняется менее чем на некоторый установленный процент ее протяженности, например менее чем на 25%, то следующая серия черных элементов будет начинаться с 3-битового кодового слова. В противном случае она начиналась бы с 4-битового кодового слова.

Каждый раз, когда происходит переход либо от белого цвета к черному, либо от черного цвета к белому, либо одноцветная пара переходит в смешанную пару, т. е. происходит переход в состояние S_1 или S_2, будет передаваться специальный префикс, или P-бит. Когда же имеет место переход из одного переходного состояния к подобному переходному состоянию (S_1 в S_1 или S_2 в S_2), производится вывод бита PC. При переходе из переходного состояния в белое или черное состояние формируются биты PC_2, PC_1, NC_1 и NC_2. Данные будут выводиться только тогда, когда происходит переход между текущей и следующей парами.

ПОСЛЕДОВАТЕЛЬНОСТЬ ПЕРЕХОДОВ СОСТОЯНИЙ ДЛЯ ПРОЦЕДУРЫ КОДИРОВАНИЯ DFC-20

Последовательность переходов состояний определяет процедуру кодирования DFC-20 для любой возможной комбинации входных данных. Эта процедура позволяет выполнять однозначное кодирование для произвольного потока входных данных, а также однозначное декодирование каждой принимаемой кодовой последовательности. Поэтому диаграмма переходов состояний является идеальной отправной точкой для проектирования цифровых схем, используемых для реализации механизма кодирования-декодирования в универсальном устройстве сжатия — восстановления данных. Диаграмма переходов состояний представлена на рис. 8.24.

Если входной поток данных к кодирующему устройству находится в состоянии S_1, то на его выходе формируется поток единиц. Появление первого нуля рассматривается как бит окончания серии, и следующие два бита теперь идентифицируют

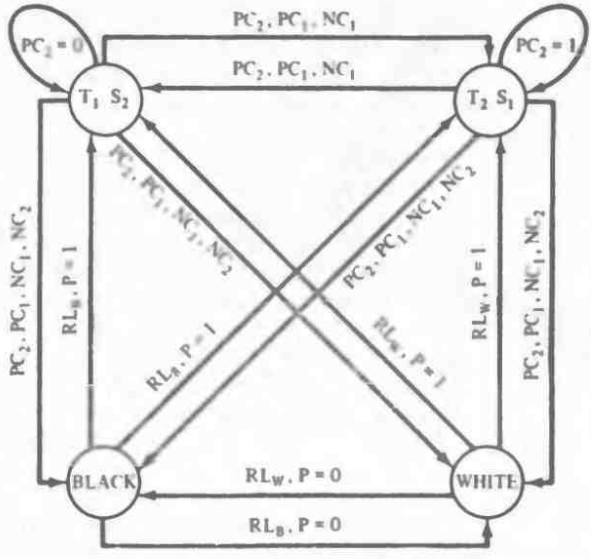


Рис. 8.24. Диаграмма переходов состояний, соответствующая процедуре кодирования DFC-20. (Патент США № 3916095 от 28 октября 1975 г.)

следующую колонку как состояние $S_0[0/0]$, $S_3[1/1]$ или $S_2[1/0]$.

Каждая серия белых или черных элементов завершается префиксом, указывающим следующее состояние колонки. В данной таблице приведены префиксы, соответствующие различным состояниям следующей колонки:

Цвет данной серии элементов	Тип следующей колонки	Префикс
Белый	Черная пара	1
Белый	Переходная пара	0
Черный	Белая пара	0
Черный	Переходная пара	1

Индикаторный бит I используется для указания того, какое из переходных состояний колонки имеет место:

$$I = 1 \text{ для } [0/1] \quad I = 0 \text{ для } [1/0]$$

В алгоритме декодирования DFC-20 используется следующая схема. Пусть X_i является одним из четырех возможных состояний пары элементов изображения, т. е. $X_{i+1} \in \{S_i\}$. Выход кодирующего устройства полностью определяется текущим состоянием (последняя введенная в систему пара элементов) и

текущим входным воздействием. Соответствующее отношение может быть представлено следующим образом:

X_I	X_J	=	X_i	Y_j
Текущие входные данные	Текущее состояние		Следующее состояние	Выход

Комбинации из входных пар, которые имеют четыре возможных состояния, и четырех текущих возможных состояний приводят к 16 допустимым комбинациям:

$$\begin{array}{cccc} X_1X_1 & X_1X_2 & X_1X_3 & X_1X_4 \\ X_2X_1 & X_2X_2 & X_2X_3 & X_2X_4 \\ X_3X_1 & X_3X_2 & X_3X_3 & X_3X_4 \\ X_4X_1 & X_4X_2 & X_4X_3 & X_4X_4 \end{array}$$

Каждый из 16 термов-произведений вида X_IX_J обусловливает выполнение определенных действий УССВ. Следовательно, наша система должна выполнять некоторый переход по одному из 16 возможных направлений (адресов), соответствующему терму X_IX_J. Отметим, что следующее состояние в следующем цикле становится текущим состоянием. Таким образом, отношение переходов состояний может быть выражено иначе, с явным указанием этой обратной связи. В матричном формате это можно представить так, как показано на рис. 8.25.

До сих пор мы вели разработку с применением принципа нисходящего проектирования систем, исходя при этом из функционального описания процесса и описания интерфейсов. Этот метод очень хорошо подходит для проектирования чисто программного обеспечения. При проектировании микропрограммных систем желательно достичь явного определения функций системы и последующей проверки различных компонентов системы, что позволит выявить преимущества определенных решений. В этом случае методология по принципу восходящего проектирования часто дает более эффективное решение по сравнению с методологией нисходящего проектирования.

Использование 4-разрядного УУВП AMD 2909 совместимо с определенным выше отношением переходов состояний. Однако 4-разрядный адрес для управляющего ПЗУ явно недостаточен, и потребуется применить либо два, либо три 4-разрядных модуля. Устройство управления выполнением программы 3001 фирмы Intel имеет 9-разрядный регистр адреса микропрограммы. Это устройство позволяет выполнять прямое ветвление по 16 направлениям и переход к микропрограмме по 4-битовому коду, подаваемому на 4-разрядную шину Рх. (В дальнейшем изложении обозначение строк и столбцов будет противоположным по отношению к обозначению, использованному в предыдущих гла-

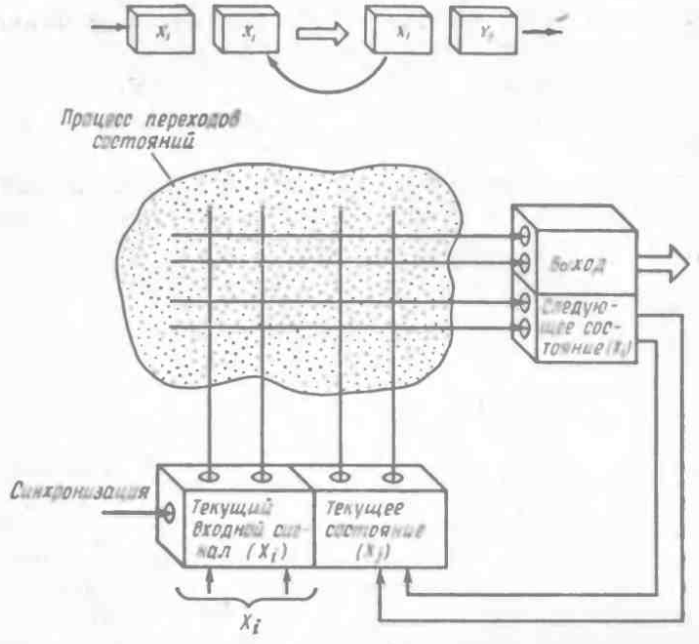


Рис. 8.25. Схема матричной реализации основного уравнения состояний.

вах.) Данные с шины Px декодируются при выполнении команды JPX, как показано ниже.

JPX	1	1	1	1	0	d_1	d_0
		Столбец				Строка	
Текущий адрес	m_8	m_7	m_6	m_5	m_4	m_3	m_2 m_1 m_0
Следующий адрес	m_8	m_7	m_6	d_1	d_0	x_7	x_6 x_5 x_4
		Столбец				Строка	

На рис. 8.26 показаны пути передачи управляющих сигналов при декодировании 16 состояний с помощью УУВП 3001. Отметим, что на выходе Y_1 получается последовательный поток данных, а не поток пар элементов. Все управление осуществляется микропрограммным путем. Табл. 8.3 представляет собой таблицу истинности для диаграммы переходов, реализуемой с

Таблица 8.3. Таблица истинности для диаграммы переходов в процессе декодирования с использованием устройства 3001[1–5]

Адрес перехода по команде JPX	Данные на первичной шине: Текущее состояние S_j	Данные на первичной шине: Входное состояние S_I	Следующее состояние или начальная серия	Правила формирования выходной последовательности	Выходной код Y	PC_1	PC_2	NC_1	NC_2
0	S_0	S_0	Белая серия	$№_W \rightarrow №_W + 1$	Пусто	0	0	0	0
1	S_0	S_1	S_1	RL_W, P=1	RL_W1	0	0	0	1
2	S_0	S_2	S_2	RL_W, P=1	RL_W1	0	0	1	0
3	S_0	S_3	Черная серия	RL_W, P=0	RL_W0	0	0	1	1
4	S_1	S_0	Белая серия	PC_2, PC_1, NC_1, NC_2	1000	0	1	0	0
5	S_1	S_1	S_1-серия	PC_2=1	..01...	0	1	0	1
6	S_1	S_2	S_2	PC_2, PC_1, NC_1	101	0	1	1	0
7	S_1	S_3	Черная серия	PC_2, PC_1, NC_1, NC_2	1011	0	1	1	1
8	S_2	S_0	Белая серия	PC_2, PC_1, NC_1, NC_2	0100	1	0	0	0
9	S_2	S_1	S_1	PC_2, PC_1, NC_1	010	1	0	0	1
A	S_2	S_2	S_2-серия	PC_2=0	..10	1	0	1	0
B	S_2	S_3	Черная серия	PC_2, PC_1, NC_1, NC_2	0111	1	0	1	1
C	S_3	S_0	Белая серия	RL_B, P=0	RL_B0	1	1	0	0
D	S_3	S_1	S_1	RL_B, P=1	RL_B1	1	1	0	1
E	S_3	S_2	S_2	RL_B, P=1	RL_B1	1	1	1	0
F	S_3	S_3	Черная серия	$№_B \rightarrow №_B + 1$	Пусто	1	1	1	1

[1]) P — префиксный бит.
[2]) 10 — завершает серию S_1: .....0000010←последняя пара S_1.
[3]) 01 — завершает серию S_2: ...1111111101←последняя пара S_2.
[4]) W — обозначение белой серии.
[5]) B — обозначение черной серии.

помощью УУВП 3001 посредством подачи по цепи обратной связи текущего состояния на шину Px. В этой таблице указаны действия, которые должны производиться, когда по команде JPX осуществляет переход к одной из 16 строк. Отметим, что этот переход фактически является переходом из одного состояния в другое, основанным на входном и текущем состояниях.

СДВИГ ЗАКОДИРОВАННЫХ ДАННЫХ ПО АЛГОРИТМУ КОДИРОВАНИЯ DFC-20

Если в алгоритме кодирования DFC-20 по команде JPX производится переход к строке с номером 1, 2, 3 или C, D, E, то осуществляется передача длины серии и префиксного бита. Указанные действия выполняются для каждой строки согласно определенному правилу. В этих случаях код снабжается счетчиками серий: для белых элементов используется регистр B, для черных — регистр C. Указанные регистры являются рабочими. Размер кода текущей серии содержится в регистрах размера кода (для черных серий это регистр 4, для белых — регистр 5). Таким образом, производится сдвиг содержимого счетчика се-

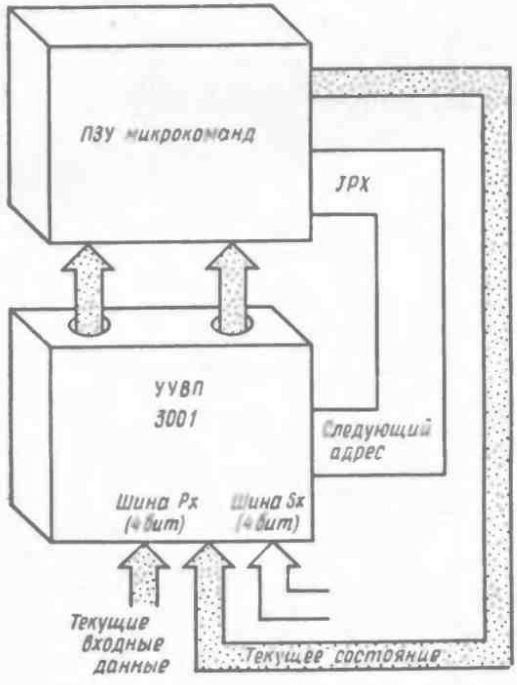


Рис. 8.26. Схема взаимодействия УУВП 3001 и ПЗУ микрокоманд.

рий, а величина копии содержимого регистра размера кода при этом уменьшается. Алгоритм циклически выполняется до тех пор, пока копия содержимого регистра размера кода не станет равной нулю. Затем выполняются служебные действия, в частности проверяется размер кодового слова и производится его выравнивание согласно следующим правилам:

1. Если общее число элементов изображения превышает размер кодового слова, то должно быть выдано дополнительное кодовое слово длины M+1, и т. д. до тех пор, пока не будет достигнуто кодовое слово максимального размера. Если кодовое слово содержит во всех разрядах единицы, то вырабатывается условие продолжения определения еще одного кодового слова.

2. Если кодовое слово не полностью заполнено единицами, то условие продолжения сбрасывается и выполняется тест, заключающийся в проверке заполнения слова на 25%. Если текущая серия меньше, чем четверть размера слова, то размер кодового слова уменьшается, в противном случае он остается без изменений.

КОДЫ ПЕРЕХОДОВ В АЛГОРИТМЕ КОДИРОВАНИЯ DFC-20

Длины серий передаются так, как было описано выше, т. е. длина серии сдвигается вправо, а копия содержимого регистра размера слова используется в качестве счетчика циклов. Когда текущее состояние является переходным, т. е. определяемым элементами [0/1] или [1/0], по алгоритму кодирования DFC-20 предусматривается переход по команде JPX к соответствующей строке из диапазона (4—B). При выборе этих строк производится передача определенной, соответствующей выбранной строке комбинации битов, т. е. выполняется особый переход вида $S_i \longrightarrow S_{i+1}$.

Длина требуемых битовых комбинаций не превышает 4 бит, и, следовательно, они могут загружаться из микрокоманды как 4 бит кода и 4 бит счетчика циклов. В нижеследующей таблице представлены номера строк, коды и число битов, задающих количество сдвигов:

Строка	Код	Число битов
4	1000	4
5	0	1
6	101	3
7	1011	4
8	0100	4
9	010	3
A	1	1
B	0111	4

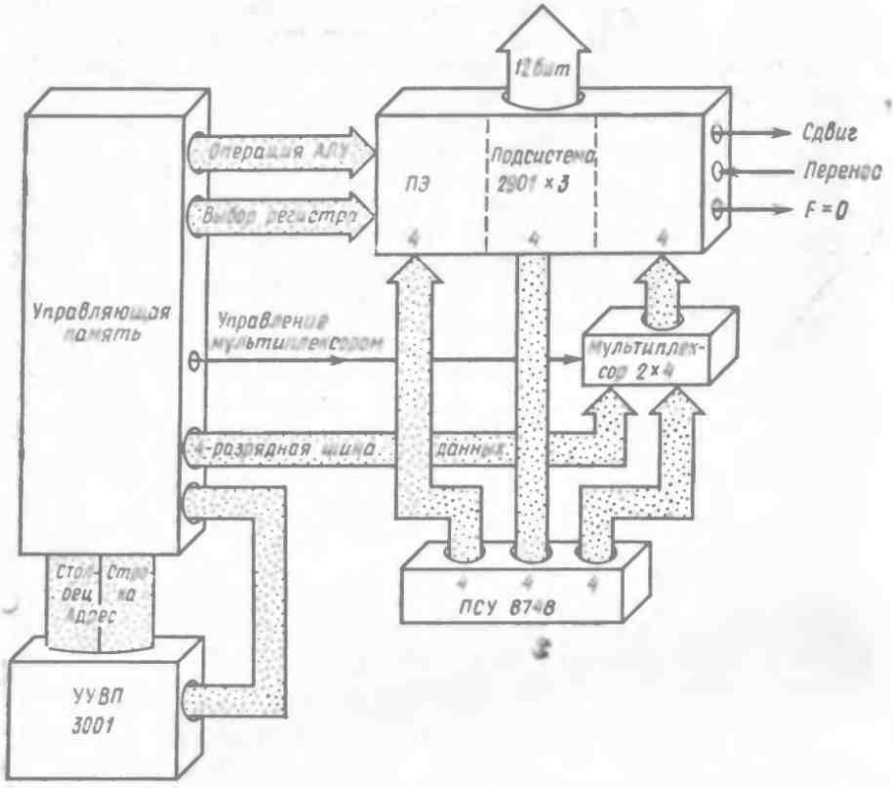


Рис. 8.27. Схема загрузки констант в системе с разрядно-модульной организацией.

ПРОЦЕДУРА ЗАДАНИЯ КОНСТАНТ ПЭ

Процессорный элемент универсальной системы сжатия — восстановления данных должен хранить 12-битовые слова и оперировать ими (должна обеспечиваться возможность увеличения разрядности), и, следовательно, должен состоять из трех модулей 2901, работающих параллельно. Регистры этих устройств должны загружаться данными. На рис. 8.27 представлена система передачи данных при загрузке констант в устройство с разрядно-модульной организацией.

РЕЖИМЫ РАБОТЫ УСТРОЙСТВА СЖАТИЯ — ВОССТАНОВЛЕНИЯ ДАННЫХ

После того как по соответствующим командам будет выполнена начальная установка УССВ и переданы необходимые параметры из устройства управления, УССВ вводится в режим ко-

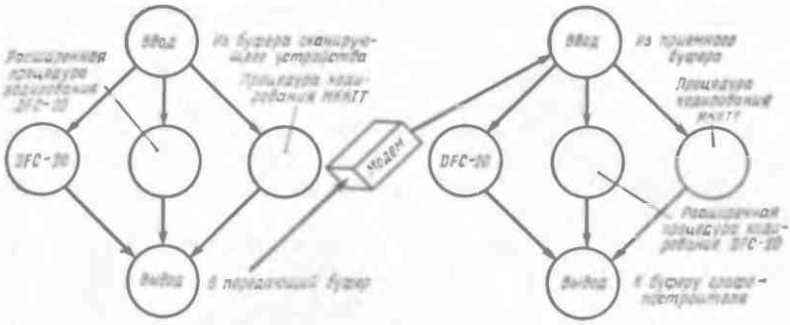


Рис. 8.28. Режимы работы УССВ.

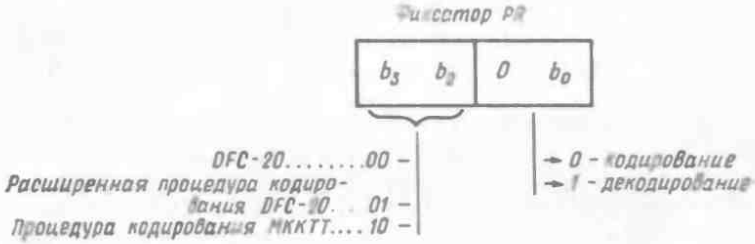


Рис. 8.29. Формат данных для фиксатора PR устройства 3001 фирмы Intel.

Рис. 8.30. Варианты перехода при выполнении команды JPR.

дирования или декодирования и, кроме того, выбирается одна из процедур кодирования: DFC-20, расширенная процедура DFC-20 или процедура МККТТ. В каждом основном режиме (рис. 8.28) производится ввод данных из соответствующего буфера, выполнение выбранного алгоритма кодирования и вывод данных в соответствующий буфер.

Формат данных, передаваемых по шине Sx и временно запоминаемых в фиксаторе PR устройства 3001, представлен на рис. 8.29. Очевидно, что этот формат удобен для выбора процедуры кодирования и режима работы. Информация на вторичной шине Sx устройства 3001 воспринимается фиксатором PR, когда выполняется команда JPX. Данные рабочего режима должны подаваться на шину Sx посредством подсистемы связи и управления во время процесса начальной установки. Команда JPR выполняется во время установки начального состояния устройства кодирования-декодирования и задает переход по одному из 16 адресов, чтобы выбрать соответствующий режим (рис. 8.30).

Схема управляющего ПЗУ рассчитана на использование команд перехода по одному из 16 адресов, что позволяет задавать как информацию режима работы, устанавливаемую посредством системы связи и управления на шине Sx, так и операцию декодирования DFC-20, определенную посредством ввода данных следующего состояния и подачей текущего состояния на шину Px по цепи обратной связи (см. рис. 8.26). Объединенная схема управления процессом кодирования-декодирования представлена на рис. 8.31. Отметим, что по команде перехода JZR из нескольких состояний системы кодирования может быть выполнен переход к программе вывода.

Данные, подаваемые посредством подсистемы связи и управления, представляют собой 12-битовое слово, загружаемое параллельно. Когда линии для передачи этого слова не используются, на них устанавливается низкий уровень напряжения. Тем самым микрокомандам управления дается возможность загружать 4-битовые данные в младшую по значимости секцию устройства путем мультиплексирования младших четырех битов и загрузки нулей в восемь старших разрядов. Поля микрокоманд, соответствующие загрузке 4-битовых констант для строк с номера 4 по номер B, расположены следующим образом:

Приемник

Функция АЛУ	Адрес СОЗУ (B_I)	Поле константы	Управление мультиплексором	Следующий адрес

После загрузки кода для i-й строки ($4 \leqslant i \leqslant B$) по микрокоманде передается управление программе SENDIT, которая выполняет вычитание 1 из содержимого регистра 1 до тех пор,

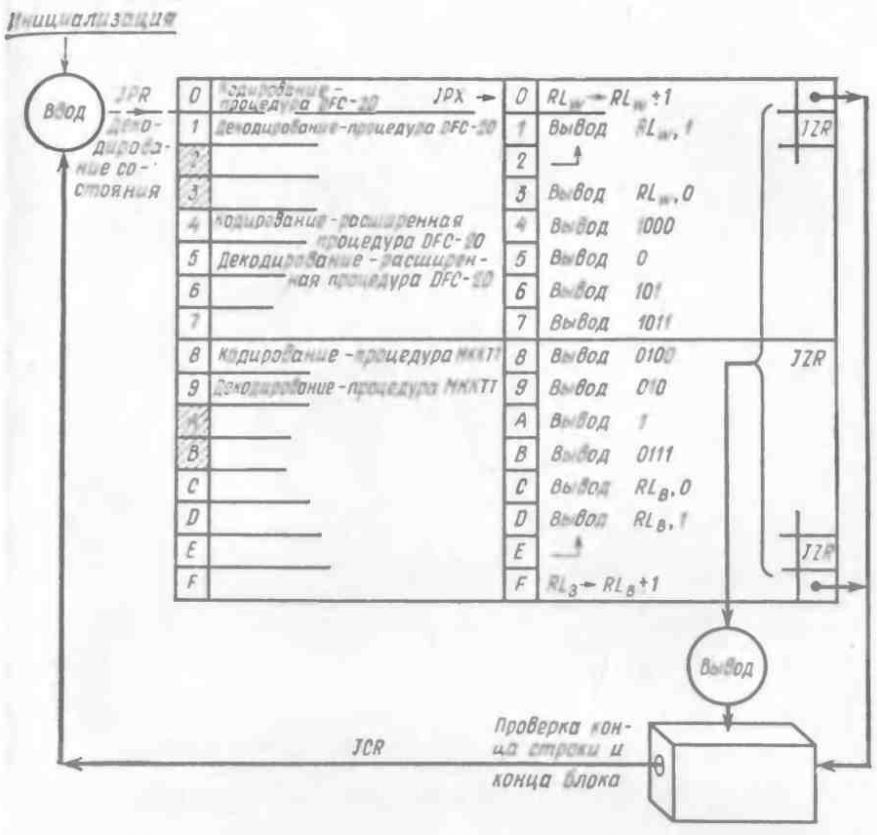


Рис. 8.31. Схема управления выбором режима функционирования процедуры кодирования.

При выполнении команды перехода по одному из 16 адресов декодируется информация, помещаемая на первичные (Px) и вторичные (Sx) входы УУВП 3001.

пока оно не станет равным нулю; одновременно в каждом цикле содержимое нулевого регистра сдвигается вправо. Подобная программа используется для того, чтобы загружать 12-битовое слово данных из подсистемы связи и управления в устройство 2901. В этом случае устройство 3001 и устройство 8748, входящее в подсистему связи и управления, выполняют операции установления соединения, так как эти два устройства работают асинхронно.

На рис. 8.32 представлена структура устройства сжатия — восстановления данных, соответствующая нашему проекту. Здесь показаны основные компоненты устройства сжатия — восстановления и линии передачи данных.

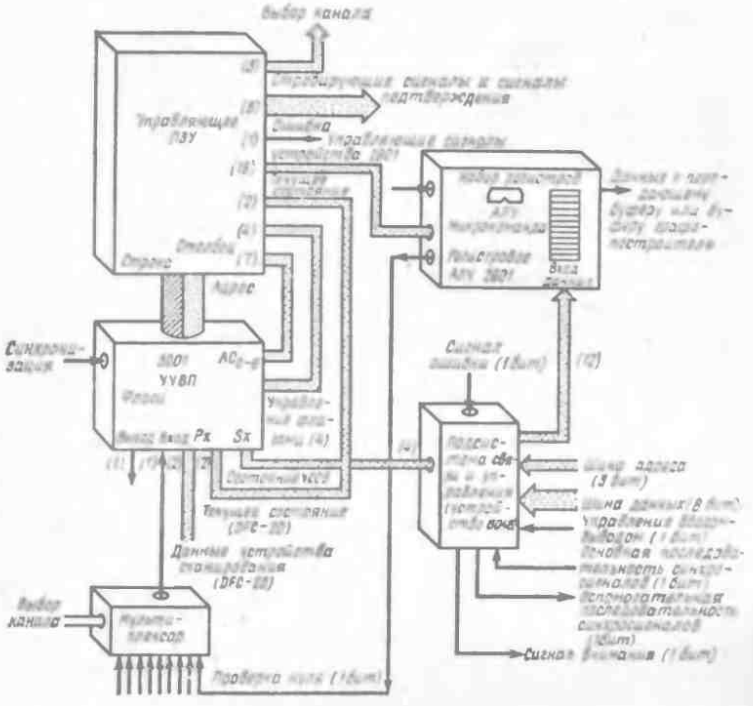


Рис. 8.32. Структура универсальной системы сжатия — восстановления.

Процессор 8048 подсистемы связи и управления имеет командный декодер и регистры состояния. Это устройство связывается с машинным блоком управления Ricoh, загружает регистры процессорного элемента параметрами и устанавливает состояние устройства 3001.

ВВОД ДАННЫХ ПРИ ИСПОЛЬЗОВАНИИ АЛГОРИТМА КОДИРОВАНИЯ DFC-20

Блок-схема алгоритма кодирования DFC-20 представлена на рис. 8.33. Данные выбираются из буфера сканирующего устройства, содержимое счетчика позиции элемента уменьшается на 1 и выполняется проверка достижения конца линии. Если конец линии не достигнут, то по команде JPX, осуществляющей передачу управления по одному из 16 адресов, производится переход к соответствующему алгоритму кодирования. В противном случае система ожидает установления синхронизации линий сканирующего устройства. Поля соответствующих микрокоманд представлены в табл. 8.4.

Таблица 8.4. **Поля микрокоманд для алгоритма, представленного на рис. 8.33**

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
SCANRDY?:	Ожидание сигнала готовности данных устройства сканирования	NOP	×	×	×	JFL(SCANRDY?, RDY)	RESET SCANACK MUX=SCANRDY?
RDY:	Обратный счет позиций элемента; данные подтверждения	DECR			Счетчик позиции элемента	JFL(GETDATA, EOLINE)	MUX=ZEROTEST; SET SCANACK
EOLINE:	Конец линии. Ожидание синхронизации строк	NOP	×	×	×	JFL(EOLINE, RESETCNTR)	MUX= =SCANLINESYNC RESET SCANACK
RESETCNTR:	Сброс счетчика позиции элемента	COPY	Позиция элемента	×	Копия позиции элемента	JMP(SCANRDY?)	SET SCANACK
GETDATA:	См. блок-схему алгоритма GETDATA	NOP	×	×	×	JPX (строка)	RESET SCANACK

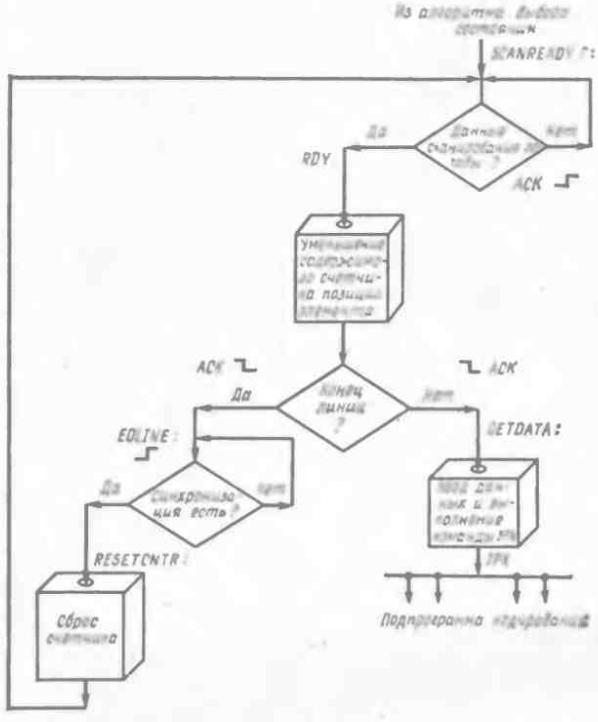


Рис. 8.33. Блок-схема алгоритма кодирования DFC-20.

ПОЛУЧЕНИЕ ДАННЫХ СКАНИРОВАНИЯ ПРИ ИСПОЛЬЗОВАНИИ ПРОЦЕДУРЫ КОДИРОВАНИЯ DFC-20

Программа GETDATA запускается, когда подается сигнал SCAN DATA RDY. Текущее состояние доступно на двух линиях шины Px устройства 3001, а входные данные устанавливаются

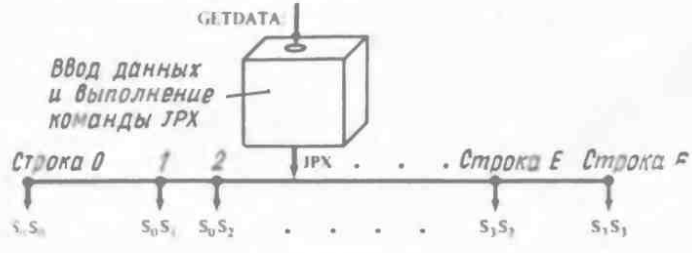


Рис. 8.34. Схема перехода по одному из 16 адресов к микропрограмме кодирования.

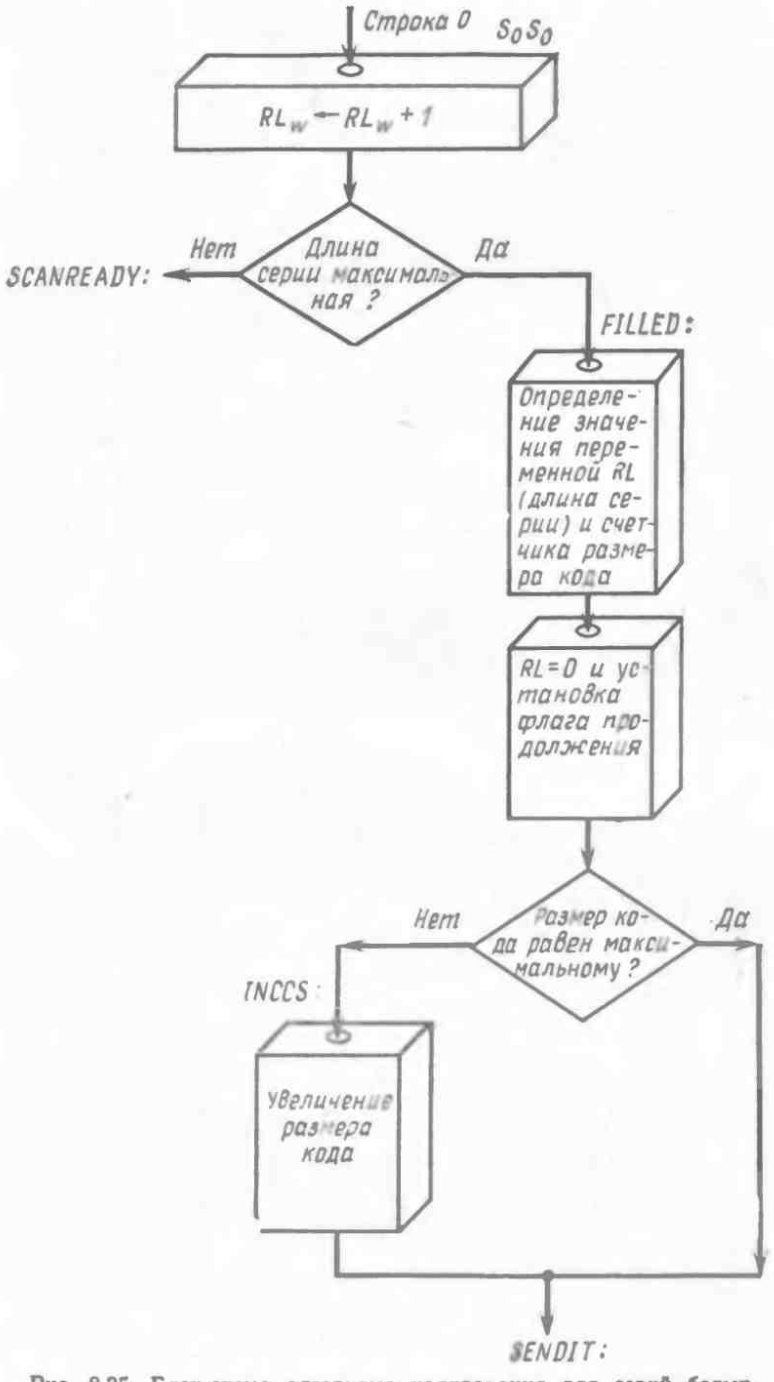


Рис. 8.35. Блок-схема алгоритма кодирования для серий белых и черных элементов.

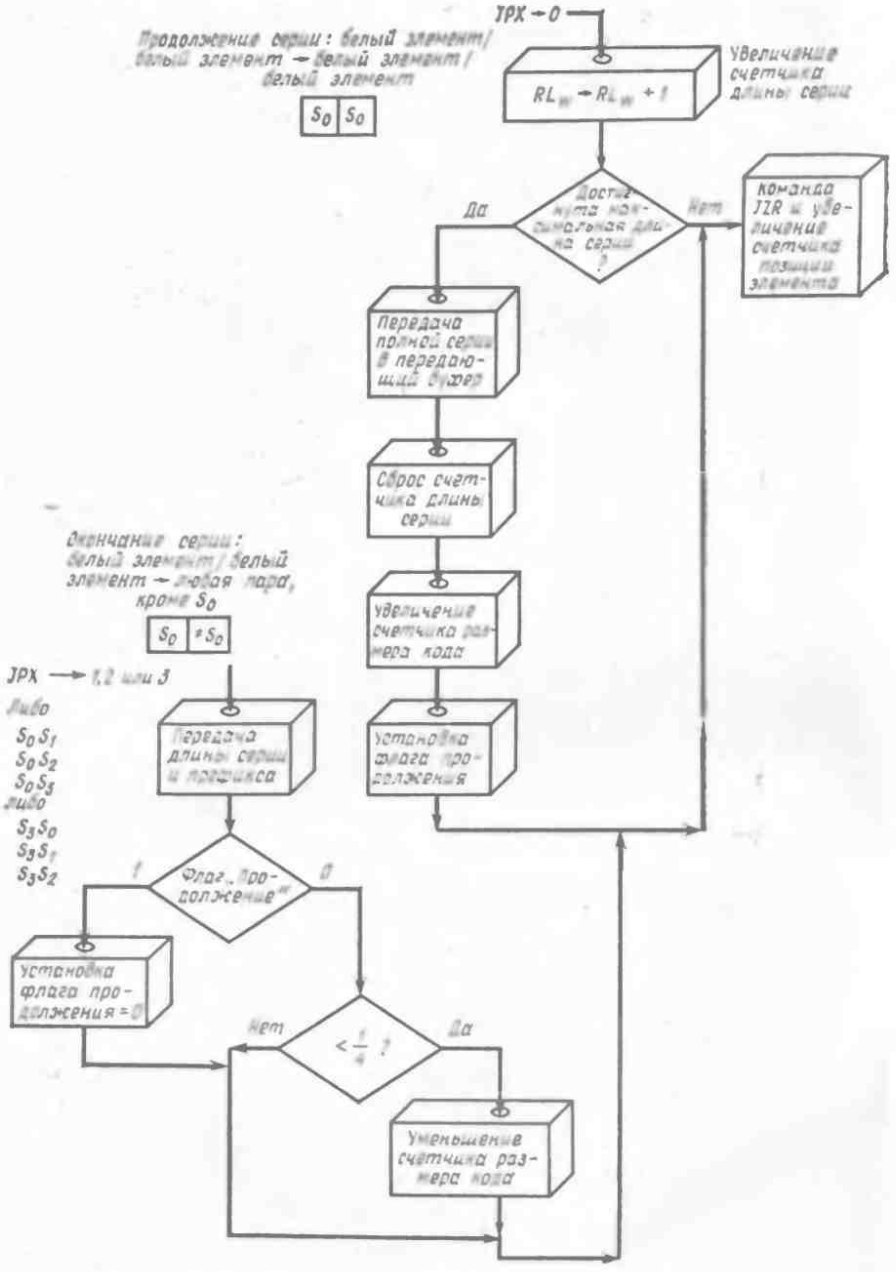


Рис. 8.36. Блок-схема алгоритма обработки серий для случаев их продолжения и окончания.

Таблица 8.5. Поля микрокоманд для алгоритма, представленного на рис. 8.34

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
ROW 0 (F)	Установка текущего состояния и увеличение счетчика длины серии белых (черных) элементов	INC	RLw	0	RLw	↓	SET (STATE=00) (и все остальные в этой подпрограмме)
	Проверка переменной RL на достижение максимального значения RL	COMPARE	WHT MAX	RLw	×	JFL(FILLED, SCANREADY)	MUX=ZEROTEST
FILLED:	Запись единиц в регистр 0	LOAD	INPUT		DATA	↓	CONSTANT=0FFH
	Копирование размера кода в регистр 1	COPY		WHT CODE SIZE	REG1	↓	
	Установка нулевого значения длины серии белых элементов	COPY	0	0	WHT RUN LENGTH	↓	SET CONTINUE FLAG
	Проверка: размер кода = =MAX?	COMPARE	CODE SIZE	WHT MAX CODE	×	JFL(OK, INCCS)	MUX=ZEROTEST
OK:	Размер кода=MAX	NOP	×	×	×	JMP(SENDIT)	
INCCS:	Увеличение размера кода	INC	WHT CODE SIZE	0	WHT CODE SIZE	JMP(SENDIT)	

на двух других линиях шины Px. Команда JPX выполняется тогда, когда, как уже описывалось, декодируются биты текущего состояния и текущих входных данных. При этом по микрокоманде, как показано на рис. 8.34, происходит переход к соответствующей строке управляющей схемы. Переход на определенную строку приводит к установлению текущего состояния (2 бит) и последующей передаче кода. Основные операции кодирования показаны раздельно на рис. 8.35 и 8.36. Если анализ данных показывает, что продолжается серия белых элементов (или серия черных элементов), то по команде JPX производится переход к строке 0 (или 0FH) и начинается операция кодирования, блок-схема алгоритма которой представлен на рис. 8.35. Микропрограмма для этого алгоритма представлена в табл. 8.5. Условие продолжения серии белых или черных элементов отображается более подробно с помощью блок-схемы, представленной на рис. 8.36. Точка входа, изображенная на схеме слева, соответствует случаю окончания серии.

ДЕЙСТВИЯ В СООТВЕТСТВИИ С АЛГОРИТМОМ КОДИРОВАНИЯ DFC-20 ПРИ ПЕРЕХОДЕ К СТРОКАМ 1—3

По завершении серии белых элементов происходит переход к строке 1, 2 или 3; устанавливается новое текущее состояние и передается соответствующий код. Этот код будет состоять из длины серии белых элементов, определенной для строк 1, 2 или 3, а механизм установки и передачи останется неизменным. Длина серии белых элементов загружается в регистр 0. В счетчик битов (регистр 1) помещается размер кода, затем производится сдвиг кода с использованием при этом счетчика битов в качестве счетчика циклов. Общая блок-схема алгоритма кодирования для строк 1, 2 или 3 представлена на рис. 8.37. Поля микрокоманд представлены в табл. 8.6.

ДЕЙСТВИЯ В СООТВЕТСТВИИ С АЛГОРИТМОМ КОДИРОВАНИЯ DFC-20 ПРИ ПЕРЕХОДЕ К СТРОКАМ 4 — B

Когда по команде JPX происходит переход к одной из строк с 4-й по B, то выполняется передача фиксированного кодового слова, которое содержится в микрокоманде и загружается в регистр данных 0 (рис. 8.38). Длина кода также хранится в микрокоманде и загружается в регистр 1. Соответственно каждой

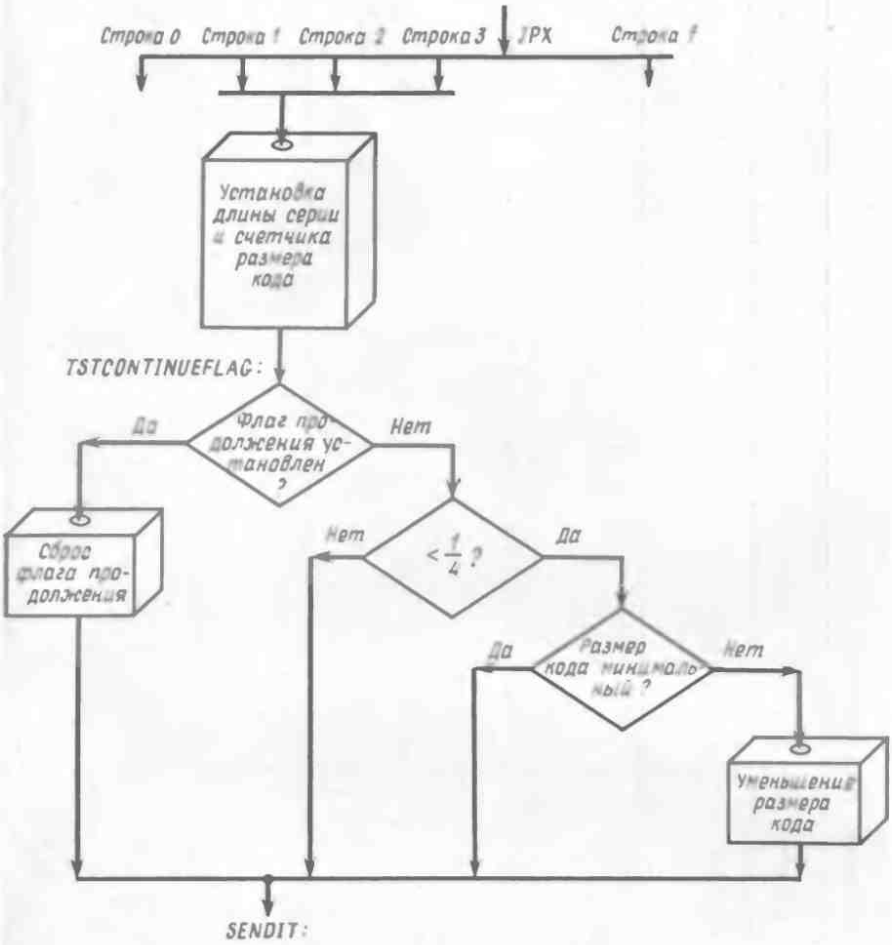


Рис. 8.37. Блок-схема алгоритма кодирования для строк 1, 2 и 3.

строке устанавливается определенный код, и затем выполняется переход к программе передачи.

Программа SENDIT последовательно передает код к передатчику. Число битов, передаваемых в кодовом слове, переменно, и, следовательно, при передаче кодового слова должна использоваться служебная информация (рис. 8.39). Закодированные данные загружаются в регистр 0, а число битов помещается в регистр 1.

Код длины серии формируется в регистре 0, а содержимое регистра размера кода копируется в регистре 1 для серии как белых, так и черных элементов. Для передаваемых элементов

Таблица 8.6. Поля микрокоманд для алгоритма, представленного на рис. 8.36

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
Строка 1, 2, 3	Запись в регистр 0 длины серии белых элементов (RL_w)	COPY	0	RL_w	Регистр 0	↓	Установка текущего состояния=01, или 10, или 11
	Копирование размера кода серии белых элементов в регистр 1	COPY	0	WHT CODE SIZE	Регистр 1	↓	
TESTCONTINUEFLAG:	Проверка флага продолжения сбрасывается автоматически. Если флаг установлен, то переход к программе SENDIT, иначе выполняется следующая команда	DECR	0	CONTFLAG	Регистр D	JFL(SENDIT, TESTMRL)	Установка C=0 в конце текущего цикла
TESTMRL:	Проверка 25%-ного заполнения	?				JFL(SENDIT, TESTSIZE)	
TESTSIZE:	Проверка возможности уменьшения размера кода	COMPARE	WHT CODE SIZE (Рег. 5)	WHT MIN SIZE (Рег. 9)	×	JFL(SENDIT, DECRSIZE)	
DECRSIZE:	Уменьшение размера кода серии белых элементов	DECR	WHT CODE (Рег. 5)	×	WHT CODE (Рег. 5)	JMP(SENDIT)	

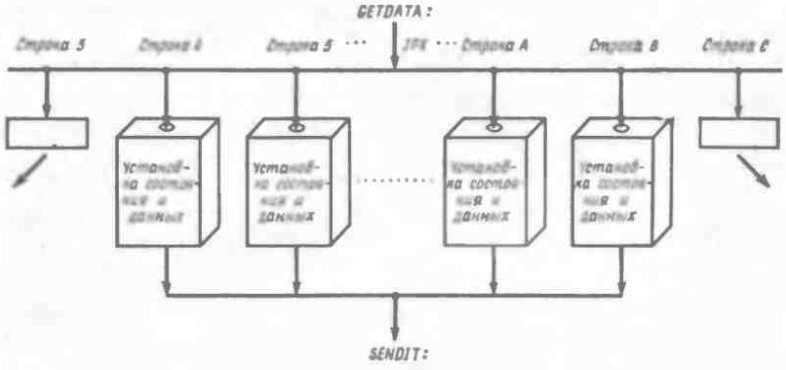


Рис. 8.38. Дерево кодирования для вариантов перехода к строкам 4 — B.

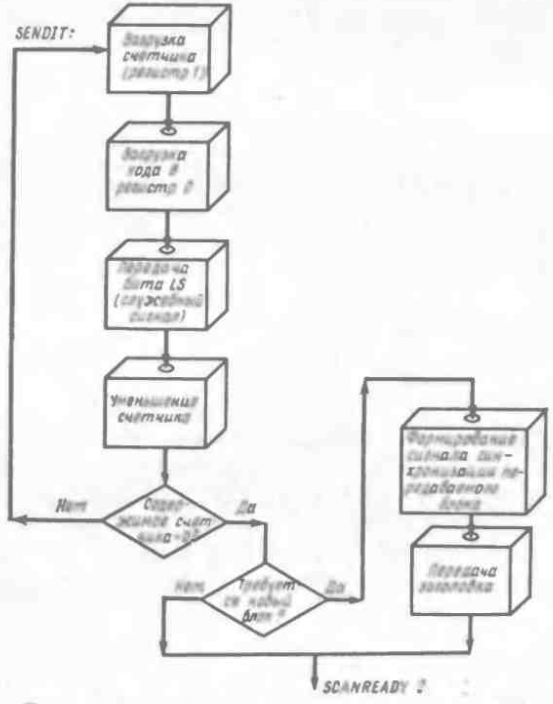


Рис. 8.39. Блок-схема программы синхронизации кодирующего-декодирующего устройства.

кодовые слова и размеры кода запоминаются в микрокоманде для каждого перехода (строки 4—8). Микрокоманды программы SENDIT имеют следующий вид:

```
SENDIT:   STROBELOW  ACKMUX                        JFL(BUSY, SENDIT)
BUSY:     STROBEHI   ZMUX    SHIFT DECCOUNTER      JFL(LOOP, EXIT)
LOOP:                                              JMP(WAITACK)
WAITACK:             ACKMUX                        JFL(WAITACK, GOTACK)
GOTACK:                                            JMP(SENDIT)
EXIT:                ACKMUX                        JFL(WAITEXIT, GETOUT)
WAITEXIT:            ACKMUX                        JFL(WAITEXIT, GETOUT)
GETOUT:
```

Блок-схема программы SENDIT изображена на рис. 8.40.

Схема управляющего ПЗУ для программы SENDIT представлена на рис. 8.41. Точкой входа в микропрограмму является пересечение строки 3 и колонки 0. По команде JFL (строка 2, колонка 0) выполняется переход на метку LOOP или EXIT (строки 2 и 3 соответственно). Затем проверяется состояние линии подтверждения (указывается посредством поля «управление мультиплексором»), и выполняется циклическое возвращение назад к строке 3 до тех пор, пока на этой линии не установится низкий уровень. Затем управление передается к строке 2, и на линии STROBE устанавливается высокий уровень напряжения; содержимое регистра сдвигается и уменьшается значение счетчика битов. Мультиплексор настраивается на проверку линии нуля, выходящей из устройства 2901; результат этой проверки используется для перехода на метку LOOP или EXIT. Из поля LOOP производится переход к полю WAITACK, где проверяется состояние линии подтверждения и сохраняется состояние ожидания до тех пор, пока на ней поддерживается высокий уровень напряжения. Затем управление передается на поле GOTACK. Если достигается точка EXIT, то также проверяется состояние линии подтверждения и производится переход к полю GETOUT, когда на этой линии устанавливается высокий уровень.

После передачи закодированных данных устройство сжатия — восстановления данных проверяет наличие запроса на новый блок. При отсутствии запроса система выполняет новый цикл, т. е. переходит к вводу данных. Если же запрос обнаружен, то система формирует сигнал TRANSMIT BLOCK SYNC и передает заголовок. В процедуре кодирования DFC-20 синхронизация строк не применяется, в ней используется следующая информация заголовка, устанавливаемая в регистрах устройства 2901:

позиция элемента (12 бит);
длина серии черных элементов (3 бит);
длина серии белых элементов (3 бит);
данные режима запуска (2 бит).

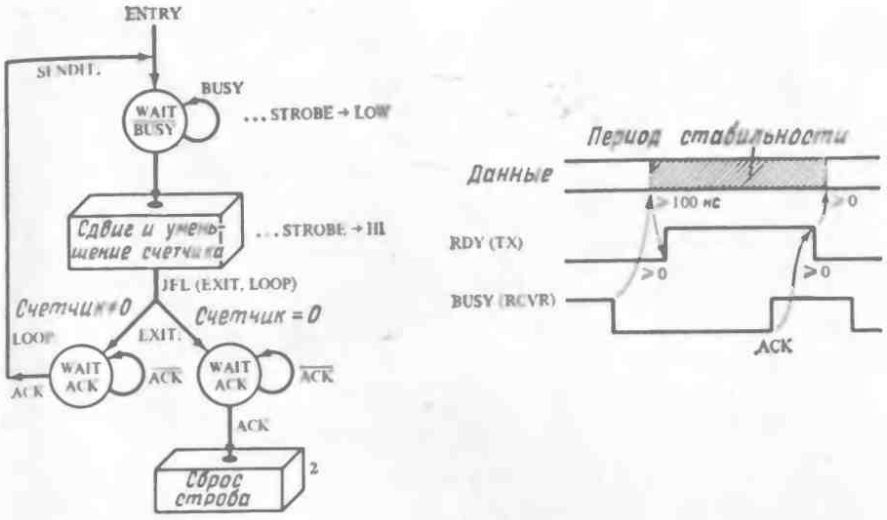


Рис. 8.40. Блок-схема алгоритма микропрограммы SENDIT.

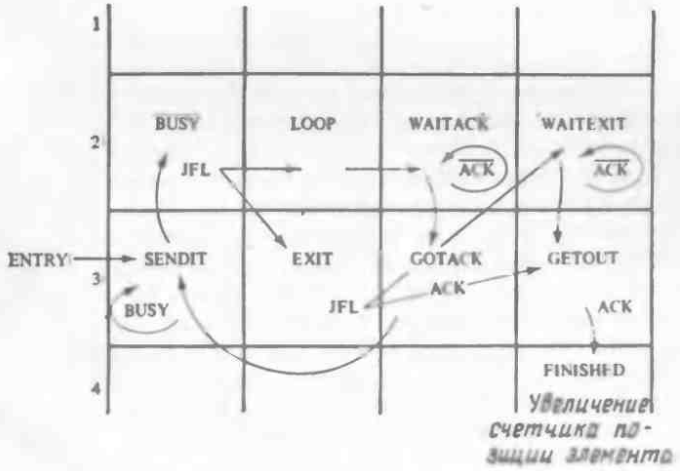


Рис. 8.41. Схема управляющего ПЗУ для УУВП 3001.

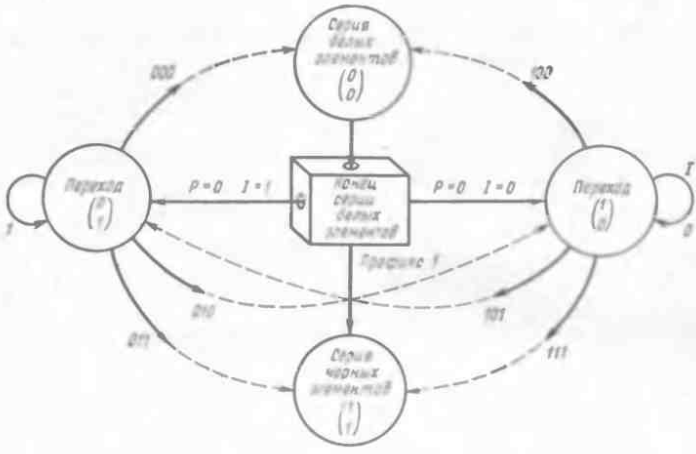


Рис. 8.42. Диаграмма переходов состояний для схемы кодирования УССВ.

Позиция элемента определяет положение начальной точки в текущей линии. Данные стартового режима (или два начальных элемента изображения) используются для индикации того, закончился предыдущий блок серией белых или черных элементов. Если начальным элементом является черный [1/1] или белый [0/0] элемент, то осуществляется ввод двух первых битов данных, которые используются для установления текущего состояния. Счетчик размера кода используется при вводе длины серии. Если превышается максимальная длина кода, то размер кода (если максимальный размер не достигнут) увеличивается. Затем производится вывод черного или белого элемента, и значение счетчика RL уменьшается до нуля. Если серия была не полной, то осуществляется вывод соответствующего числа элементов изображения, как описано выше, и проверяется значение префиксного бита. Если префиксный бит равен нулю, то формируется дополнительная серия, в противном случае происходит переход к следующему состоянию и вводятся следующие два бита с целью определения конкретного состояния.

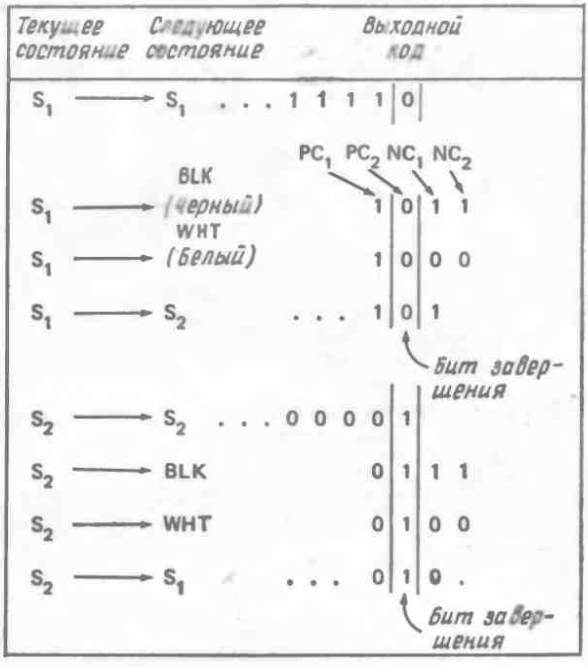
Диаграмма переходов состояний в несколько измененном виде изображена на рис. 8.42. В данном случае используются следующие соотношения:

$$S_1 = [0/1] = [PC_1/PC_2]$$
$$S_2 = [1/0] = [PC_1/PC_2]$$

переходные элементы

Переходы состояний для этих переходных элементов даны в табл. 8.7.

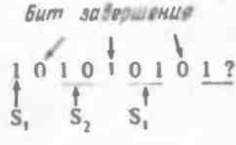
Таблица 8.7. **Переходы состояний**



Согласно правилу формирования выходной последовательности битов, будет получена следующая цепочка:

$$S_1 \longrightarrow S_2 \longrightarrow S_1 \longrightarrow S_2$$
$$(101) \quad (010) \quad (101)$$

Эта кодовая последовательность должна быть декодирована следующим образом:



Основные переходы состояний при декодировании представлены на рис. 8.43. Когда текущий вводимый элемент изображения является элементом $S_1 = [0/1]$, существуют две возможности для определения следующего состояния (S_1 и не S_1). Если имеет место последовательность состояний S_1, то будет закодирована последовательность единиц. Появление нуля в закодированных данных рассматривается как бит окончания, а сле-

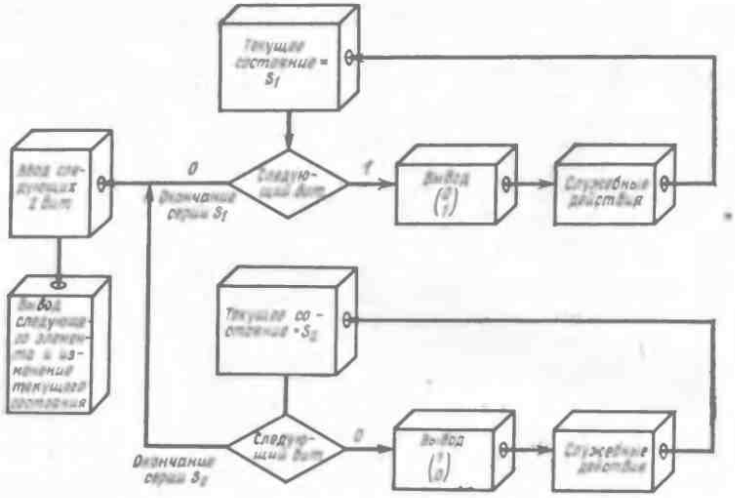


Рис. 8.43. Блок-схема алгоритма декодирования переходов состояний.

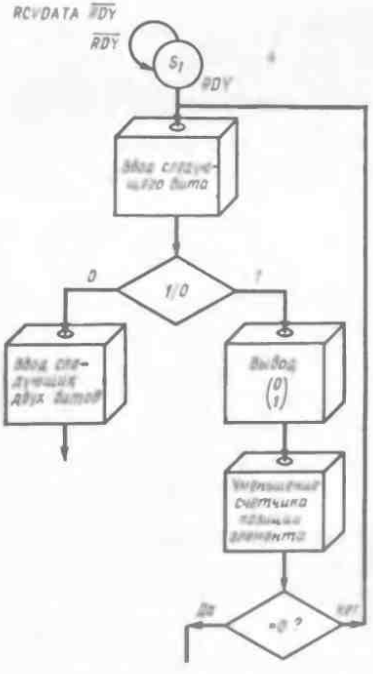


Рис. 8.44. Блок-схема процедуры обработки состояния S_1.

Таблица 8.8. Поля микрокоманд для алгоритма, представленного на рис. 8.43

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
S1:	Ожидание сигнала RCVDATARDY Сброс сигналов PLOTTERDATARDY и ACK	NOP	×	×	×	JFL(S1, S1+1)	MUX=RCVDATARDY RESET PLOTDATARDY &ACK
S1+1:	Проверка бита данных посредством флага F	NOP	×	×	×	JFL(POINTA, SENDS1)	MUX=DATA
SENDS1:	Пересылка пары S1 и ожидание сигнала ACK	NOP	×	×	×	JFL(SENDS1, DECELPC)	DATAOUTPAIR= $\binom{0}{1}$ MUX=PLOTDATAACK SET PLOTDATARDY
DECELPC:	Уменьшение счетчика позиции элемента и условный переход	DECR	ELPOSCNT	×	ELPOSCNT	JFL(S1, ↓)	MUX=ZEROTEST

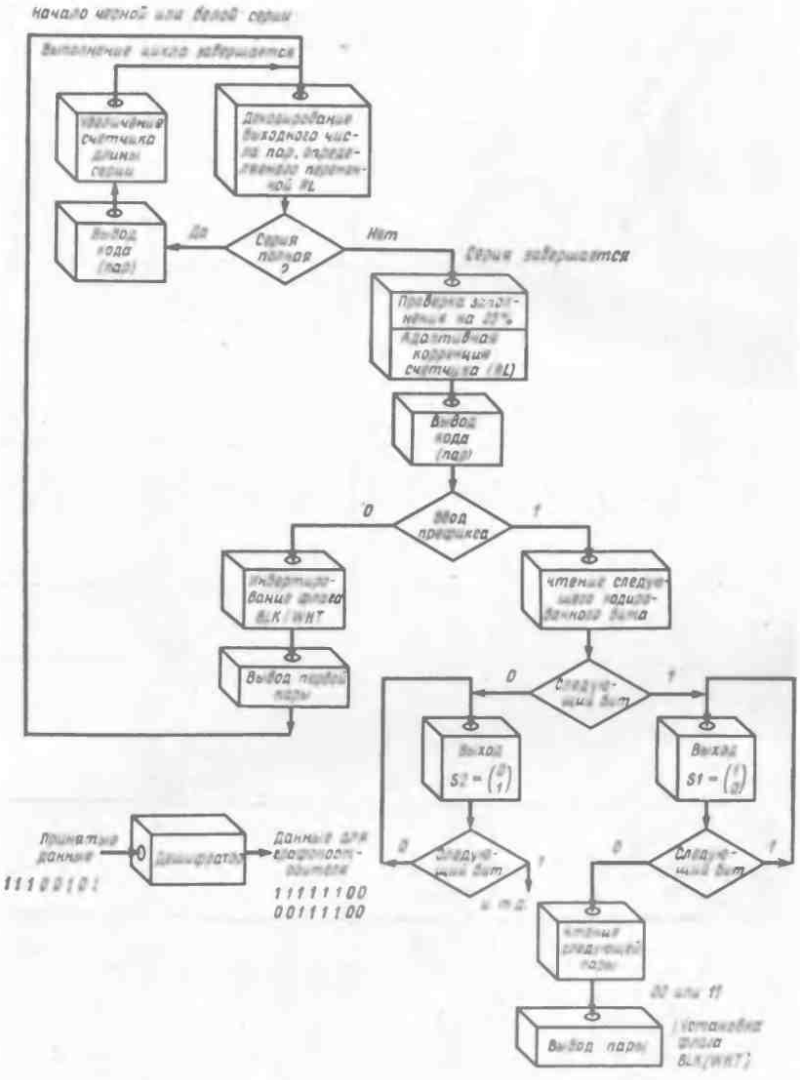


Рис. 8.45. Блок-схема алгоритма декодирования DFC-20.

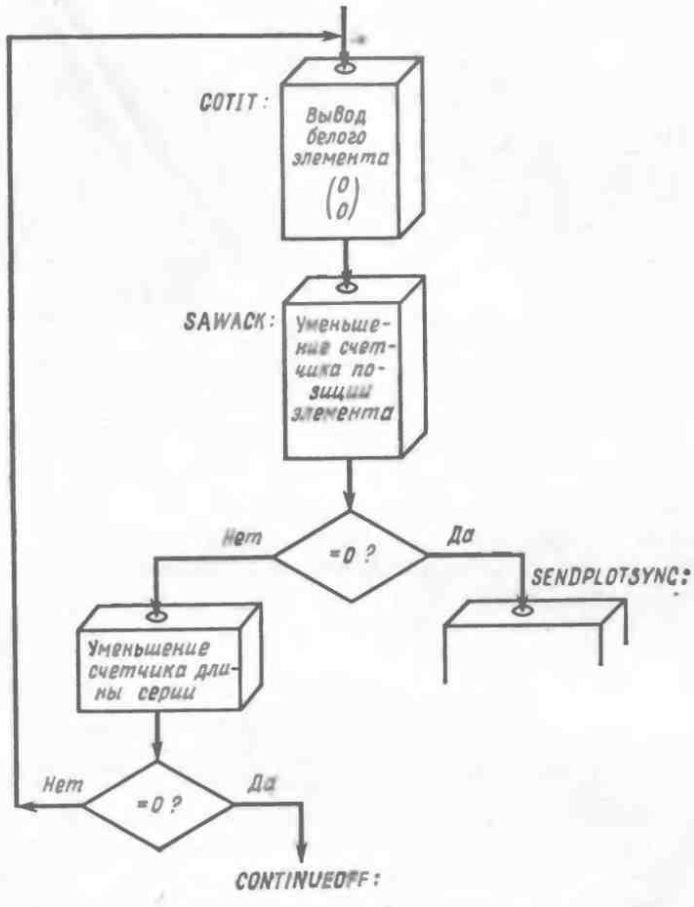


Рис. 8.46. Блок-схема алгоритма использования счетчика длины серии для вывода пар элементов данных.

дующие два бита кодированных данных определяют следующее состояние. Блок-схема этой процедуры представлена на рис. 8.44.

Таблица 8.8 содержит поля соответствующих микрокоманд. Микропрограмме, представленной в табл. 8.8, для состояния S_1 соответствует следующая диаграмма ее выполнения:

ВЫВОДЫ ПО ПРОЦЕДУРЕ ДЕКОДИРОВАНИЯ DFC-20

Процесс декодирования, обобщенно представленный в табл. 8.9, начинается тогда, когда устройство сжатия — восстановления данных переводится в режим декодирования по алгоритму DFC-20 посредством информации макросостояния, подаваемой на шину Sx устройства 3001 подсистемой связи и управления. Эта подсистема указывает устройству 3001, что информация заголовка готова путем установки высокого уровня на линии RDY. Затем подсистема связи и управления ожидает сигнал подтверждения ACK. Устройство 3001 воспринимает данные заголовка и загружает регистры устройства 2901. Длины серий белых и черных элементов вводятся в устройство 2901, как описано ниже. Размер кода копируется в счетчик битов (регистр 1) и используется как счетчик циклов при вводе битов данных. Данные анализируются для определения того, заполнена серия или нет. Блок-схема этой процедуры представлена на рис. 8.45.

После загрузки длины серии в регистр 0 последний используется для вывода пар белых (или черных) элементов. Счетчик позиции элемента проверяется для установления условий «Конец линии» (рис. 8.46).

Микропрограмма для вывода пар данных представлена в табл. 8.10. Когда содержимое счетчика становится равным 0, как показано на рис. 8.47, проверяется условие продолжения и выполняется проверка размера кода.

ИЗМЕНЕНИЕ РАЗМЕРА КОДА

Как следует из микропрограммы, представленной в табл. 8.11, с целью определения того, была ли полной предыдущая серия, проверяется флаг продолжения. Если предыдущая серия была полной, то производится проверка размера кода для

Таблица 8.9. Процесс декодирования по алгоритму DFC-20

Текущее состояние	Описание
WHT	При завершении выдается длина серии; префикс, следующий за длиной серии, определяет следующий элемент — черный или переходный
S2	Последовательность единиц; бит завершения =0; следующие два бита определяют следующее состояние (колонку)
S1	Последовательность нулей; бит завершения =1; следующие два бита определяют следующее состояние (колонку)
BLK	При завершении выдается длина серии; префикс, следующий за длиной серии, определяет следующий элемент — белый или переходный

Таблица 8.10. Поля микрокоманд для алгоритма, представленного на рис. 8.45

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
GOTRL:	Вывод $\binom{0}{0}$ и установка сигнала RDY	NOP	×	×	×	JMP (↓)	Установка сигнала DATARDY. Установка $\binom{0}{0}$
LOOP:	Ожидание сигнала ACK для графопостроителя	NOP	×	×	×	JFL(LOOP, SAWACK)	MUX=PLOTACK
SAWACK:	Уменьшение счетчика позиции элемента	DECR	ELPOSCNT	×	ELPOSCNT	JFL(MORE?, SNDPLOT SYNC)	MUX=ZEROTEST Установка DATARDY
MORE?:	Передавать еще пары белых элементов?	DECR	DATAREG (Регистр 0)	×	DATAREG	JFL(GOTRL., TSTCONT FLAG)	MUX=ZEROTEST
TSTCONT INFLAG:	Проверка флага продолжения	NOP	×	×	×	JFL(CONTINON, CONTINOFF)	⧖

Таблица 8.11. Поля микрокоманд для алгоритма, представленного на рис. 8.47

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
CONTINUE ON:	Проверка размера кода на максимальное значение	COMPARE	WHT CODE SIZE	MAX WHT SIZE	×	JFL(INCCS, S0)	MUX=ZEROTEST
INCCS:	Увеличение размера кода (WHT)	INCR	WHT CODE SIZE	×	WHT CODE SIZE	JMP(S0)	
CONTINUE OFF:	Проверка MRL< <25%	DECR	CODE SIZE (WHT)	×	WHT CODE SIZE COPY (Per. 1)	JMP (↓)	
		DECR	CODE SIZE COPY (Per. 4)	×	WHT CODE SIZE COPY (Per. 1)	JMP (↓)	
DOMRL:	Сдвиг всех битов, кроме двух старших	SHIFTRT	DATA REG (Per. 0)	×	×	JMP (↓)	

	Проверка счетчика сдвигов	DECR	CODE SIZE COPY (Per. 1)	×	CODE SIZE COPY (Per. 1)	JFL(DOMRL, LOOKATEM)	MUX=ZEROTEST
LOOKATEM:	Анализ двух старших битов	AND	DATABLE	DATA REG(0)	×	JEL(GETBIT, DCCS)	DATABUS=03H MUX=ZEROTEST
TESTMIN:	Меньше 25%; можно ли уменьшить размер?	COMPARE	CODE SIZE (WHT)	MIN (WHT) CODE SIZE	×	JFL(GETPBIT, DCCS)	MUX=ZEROTEST
DCCS:	Уменьшение размера кода (WHT)	DECR	WHT CODE SIZE	×	WHT CODE SIZE	JMP(GETPBIT)	
GETPBIT:	Ожидание сигнала готовности (RDY) данных	NOP	×	×	×	JFL(GETPBIT, PBITRDY)	MUX=RCVDATA Установка сигнала RDY
PBITRDY:	Установка сигнала ACK и проверка состояния PBIT	NOP	×	×	×	JFL(S3, POINTA)	Установка сигнала ACK MUX=RCVDATA

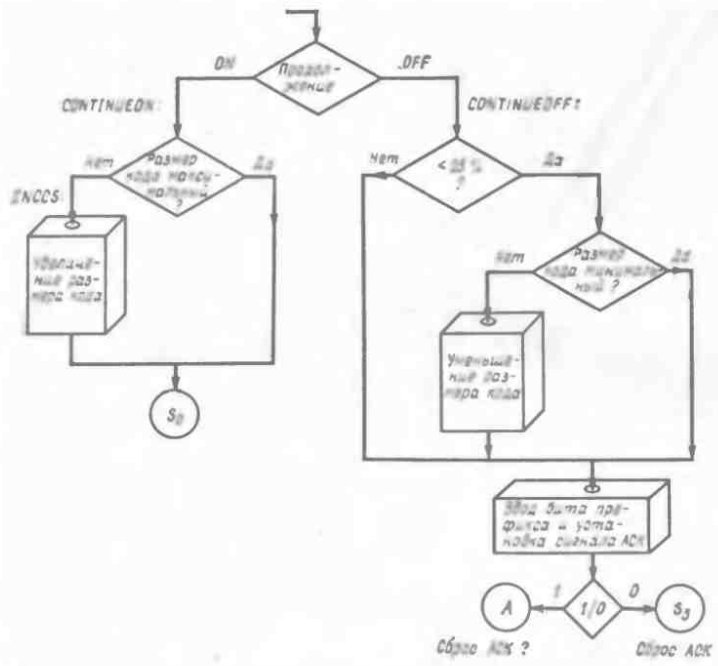


Рис. 8.47. Блок-схема алгоритма проверки условия продолжения (CONTINUE).

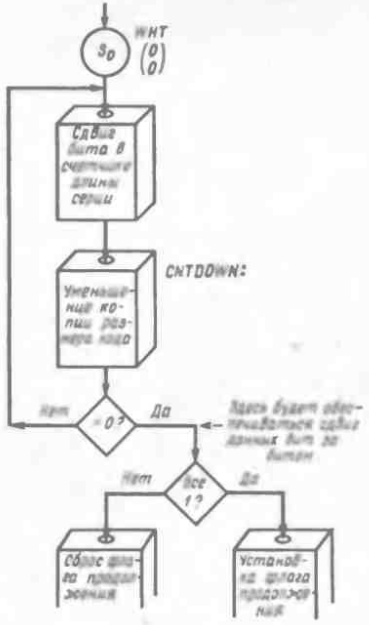


Рис. 8.48. Блок-схема алгоритма проверки размера кода.

Таблица 8.12. Поля микрокоманд для алгоритма, представленного на рис. 8.48

Метка	Описание операции	Операция АЛУ	Источник 1	Источник 2	Адресат	Команда перехода	Управляющее поле
S0:	Ожидание сигнала DATARDY	NOP	×	×	×	JFL (S0, S0+1)	Сброс сигнала ACK Установка флага C
S0+1:	Передача 1 бит кода в регистр 0	SHIFT	DATA REG RAM_0	×	DATAREG	JFL(SETCONTFLAG, CNTDOWN)	MUX= =RCVDATALINE
SET CONTFLAG:		NOP	×	×	×	JMP CNTDOWN	MUX=DATALINE Установка флага C
CNTDOWN:	Установка флага C=0 для указания неполного заполнения серии	DECR	CODE SIZE COPY	×	CODE SIZE COPY (Рег. 1)	JFL(S0, GOTRL)	MUX=TESTZERO
GOTRL:	Вывод элемента $WHT\binom{0}{0}$ и установка сигнала RDY	NOP	×	×	×	JMP ( )	Установка сигнала DATARDY

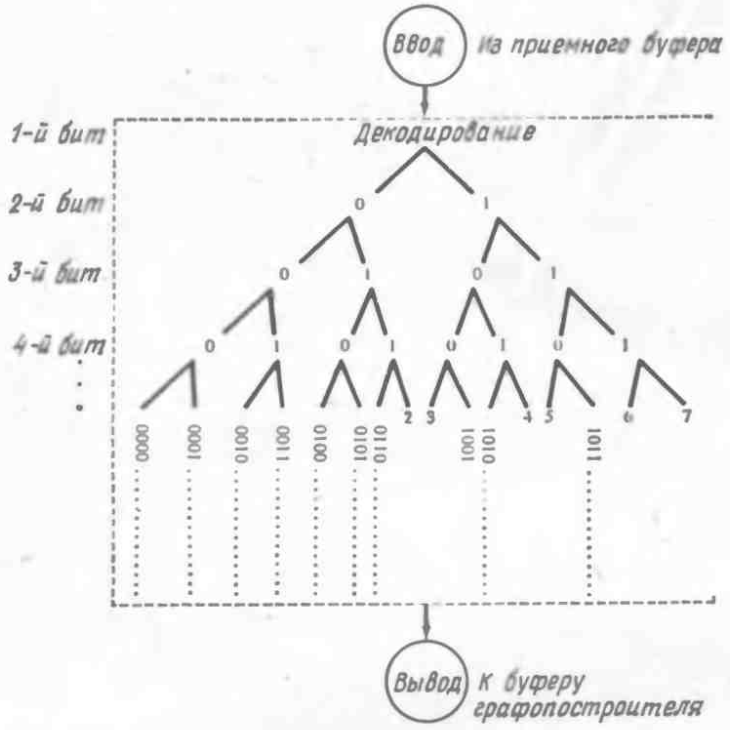


Рис. 8.49. Представление алгоритма декодирования кода МККТТ.

определения того, увеличился размер кода или нет (рис. 8.48). Если размер кода не увеличился, то выполняется тест на 25%-ное заполнение и затем размер кода, если это целесообразно, проверяется еще раз. Затем вводится префиксный бит и делается соответствующий переход. Микропрограмма для тестирования размера кода представлена в табл. 8.12.

ОБЕСПЕЧЕНИЕ СИНХРОНИЗАЦИИ ПРИ ИСПОЛЬЗОВАНИИ ПРОЦЕДУРЫ КОДИРОВАНИЯ-ДЕКОДИРОВАНИЯ МККТТ

Алгоритм МККТТ был запрограммирован для однокристальной микро-ЭВМ с целью практической проверки реализации синхронизации. Был запрограммирован только алгоритм декодирования, так как кодирование реализуется с использованием табличного поиска и проблемы обеспечения синхронизации не возникает.

При скорости передачи 19,2 Кбит/с для кодирования данных, представленных в коде Хаффмена, требуется ~52 мкс/бит. Программа декодирования для микропроцессора 8048 будет затрачивать 35 мкс/бит и, таким образом, она сможет обеспечить требуемую скорость передачи. После декодирования данных микропроцессор 8048 может загрузить декодированные данные в подсистему 3001/2901 для последующего их вывода. Таким образом, входной поток, закодированный по алгоритму МККТТ, и выходной поток декодированных данных могут обрабатываться одновременно. Алгоритм процесса декодирования данных с использованием алгоритма МККТТ изображен на рис. 8.49.

ОГЛАВЛЕНИЕ