

К. Г. САМОФАЛОВ, В. И. КОРНЕЙЧУК, В. П. ТАРАСЕНКО

ЦИФРОВЫЕ ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ

ТЕОРИЯ
И ПРОЕКТИРОВАНИЕ

ИЗДАНИЕ ВТОРОЕ,
ПЕРЕРАБОТАННОЕ И ДОПОЛНЕННОЕ

Допущено Министерством высшего
и среднего специального образования УССР
в качестве учебника для студентов вузов,
обучающихся по специальностям
«Электронные вычислительные машины и системы»,
«Прикладная математика»,
«Автоматизированные системы
управления»

КИЕВ
ГОЛОВНОЕ ИЗДАТЕЛЬСТВО
ИЗДАТЕЛЬСКОГО ОБЪЕДИНЕНИЯ
«ВИЩА ШКОЛА»
1983

УДК 681.3(075.8)

Цифровые электронные вычислительные машины
лов К. Г., Корнейчук В. И., Тарасенко В. П. — 2-е изд.
и доп. — Киев: Вища школа. Головное изд-во, 1983

В учебнике рассмотрены основы теории и проектирования ЭВМ, схемотехнические основы цифровой вычислительной техники, принципы построения операционных устройств цифровых ЭВМ. Особое внимание уделено оптимизации аппаратных средств цифровых ЭВМ по стоимости, стоимости и другим критериям; построению схем цифровых ЭВМ на основе больших и малых микросхем, микропроцессорных наборов, на других интегральных компонентах. Теоретические положения иллюстрированы примерами построения схем цифровых ЭВМ третьего поколения, серий ЕС, СМ и микро-ЭВМ.

Второе издание (1-е изд. — 1976 г.) дополнено разделами, посвященными проектированию операционных узлов и блокам организации структур памяти, микропрограммному управлению микропроцессорными средствами обработки информации.

Для студентов вузов, обучающихся по специальности «Электронные вычислительные машины и системы», а также для преподавателей «Вычислительная математика». Может быть полезен научным и инженерным работникам, занимающимся проектированием и применением цифровых ЭВМ.

Ил. 334. Табл. 132. Библиогр.: 62 назв.

Рецензенты:

доктора техн. наук *А. В. Плотников* (Ленинградский политехнический институт), *Л. Я. Нагорный* (Киевский институт гражданской авиации), *С. С. Забара* (Киевское ПО «Маш» им. В. И. Ленина)

Редакция литературы по кибернетике,
электронике и энергетике.

Зав. редакцией *М. С. Хойнацкий*

ПРЕДИСЛОВИЕ

Достижения в области создания и применения средств вычислительной техники в настоящее время определяют не только уровень производства и организации самых различных процессов управления, но и возможности фундаментальных исследований во всех отраслях науки. Поэтому вычислительная техника стала важнейшим фактором научно-технического прогресса и неотъемлемой составной частью производительных сил во всех технически развитых странах мира.

В решениях XXVI съезда КПСС определено, что одной из важнейших проблем, подлежащих решению в одиннадцатой пятилетке, является совершенствование вычислительной техники, ее элементной базы и математического обеспечения, средств и систем сбора, передачи и обработки информации. Кроме того, приборостроению, как отрасли промышленности, поставлена в качестве основной задача повышения технического уровня вычислительной техники и развития опережающими темпами производства быстродействующих управляющих и вычислительных комплексов, периферийного оборудования и программных средств к ним.

Учебник предназначен для студентов вузов специальности «Электронные вычислительные машины и системы» по курсу «Теория и проектирование цифровых ЭВМ». В нем рассмотрены вопросы структурно-функциональной организации аппаратных средств цифровой вычислительной техники. В соответствии с учебным планом указанной специальности содержание книги не дублирует материал, изучаемый в других дисциплинах, а рассматривает его лишь в объеме, необходимом для усвоения основного материала данного курса или для логической завершенности изложения (такой текст в книге набран преимущественно петитом). Материал представлен так, что может быть усвоен студентами специальности «Электронные вычислительные машины и системы» без использования дополнительной литературы.

В основу разделов книги, связанных с проектированием функциональных структур цифровой вычислительной техники, положен принцип оптимизации схемных и системных решений по важнейшим технико-экономическим показателям. В книге нашли отражение также результаты исследований по структурно-функциональной организации цифровых ЭВМ, полученные научной школой по вычислительной технике, сложившейся в Киевском политехническом институте.

Учебник является базовым для создания комплексного информационно-методического обеспечения по курсу «Теория и проектирование цифровых ЭВМ», включающему методические указания к курсовому и дипломному проектированию, лабораторный практикум, сборники задач для машинного и безмашинного программированного контроля знаний, каталоги типовых схемных решений, иллюстративные материалы.

Пожелания по книге просим направлять в Головное издательство издательского объединения «Вища школа» по адресу: 252054, Киев-54, ул. Гоголевская, 7.

ИНФОРМАЦИОННЫЕ ОСНОВЫ ЦИФРОВОЙ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

1.1. КРАТКИЕ ИСТОРИЧЕСКИЕ СВЕДЕНИЯ. ИНФОРМАЦИЯ. ЕДИНИЦЫ ИНФОРМАЦИИ

Создание в 40—50-х годах XX ст. цифровых электронных вычислительных машин и последующее быстрое развитие нового научно-технического направления обусловлено всем историческим процессом развития познания и производительных сил. Эта объективно-историческая обусловленность выразилась, во-первых, в значительном расширении класса задач вычислительного и логического характера в физике, астрономии, экономике, управлении производством, энергетике, космонавтике, авиации и других отраслях науки и техники, во-вторых, в возрастании сложности, громоздкости и точности самих вычислений при решении указанных задач. В то же время появление цифровых ЭВМ оказало сильное влияние на развитие вычислительной математики, поставив основной ее целью исследование математических вопросов, связанных с использованием цифровых ЭВМ.

Исторические корни создания средств ЦВТ прослеживаются более-менее точно лишь в течение последних трех столетий. По-видимому, древнейшими приспособлениями для цифрового счета являлись палочки с зарубками, счетные палочки, абак, счеты в различных их вариантах. В 1642 г. Б. Паскаль сконструировал машину, выполнявшую операции сложения и вычитания, использовавшуюся при выполнении расчетов по сбору налогов во Франции. Известный математик Г. В. Лейбниц в 1694 г. усовершенствовал машину Паскаля. Машина Лейбница выполняла четыре основные арифметические операции, возведение в степень и извлечение корня. В 1820 г. Т. де Кольмар разработал арифмометр, пригодный для практического применения в коммерческих операциях. Широкое распространение получил арифмометр, созданный русским инженером В. Т. Однером в 1874 г. Эта конструкция оказалась столь удачной, что к настоящему времени разработано более 50 типов малых механических и электромеханических вычислительных машин, работающих по принципу машины Однера. В 60—90-е годы XIX ст. ряд механических вычислителей был разработан академиком П. Л. Чебышевым. В конце XIX ст. появились электромеханические счетно-перфорационные и счетно-аналитические машины, в которых были автоматизированы некоторые элементы процесса вычислений.

В 1833 г. профессор Кембриджского университета Ч. Бэббидж пришел к выводу о возможности построения машины, которая могла бы выполнять любые вычисления, заданные оператором, а не только вычисления одного специального вида, как это имело место во всех ранее предложенных проектах. Эта машина была названа Бэббиджем аналитической машиной и состояла из двух частей — «мельницы» и «склада», которые являются прототипами процессора и памяти в современных цифровых ЭВМ. «Мельница» предназначалась для производства арифметических действий над числами (арифметическое устройство), а «склад» — для хранения чисел (запоминающее устройство). По проекту Бэббиджа «склад» должен состоять из 50 000 цифровых колес. Для ввода чисел в машину и управления ходом вычислений предполагалось использовать перфокарты (соединенные в единую ленту), применявшиеся в жаккардовых машинах

при танье сложных рисунков. При жизни Ч. Бэббиджа машина не была построена.

Успехи радиотехники, телефонной связи и теории релейно-контактных схем, а также опыт эксплуатации электромеханических вычислительных устройств создали необходимые предпосылки для разработки в 40-х годах XX ст. средств ЦВТ, в которых в качестве основных структурных элементов использовались электромагнитные реле. Первая цифровая ЭВМ «Марк I» с программным управлением на электромагнитных реле построена в США 1944 г.

Через два года в США построена первая цифровая ЭВМ «ЭНИАК» (электронный цифровой интегратор и вычислитель) на электронных лампах.

В СССР первая цифровая ЭВМ «МЭСМ» (малая электронная счетная машина) построена в 1950 г. под руководством академика С. А. Лебедева в Институте электротехники АН УССР (г. Киев). Опыт ее разработки и эксплуатации послужил основой для создания в последующие годы различных по назначению и техническим возможностям отечественных цифровых ЭВМ (например, «БЭСМ», «Стрела», «Урал», «Киев», «Минск», «Раздан», «Наири», «Днепр», «М-20», «М-220», «Мир») и их серий.

Развитие ЦВТ неразрывно связано с развитием электронной техники: первые цифровые ЭВМ были ламповыми, однако уже через несколько лет достижения полупроводниковой техники позволили заменить ламповые компоненты цифровых ЭВМ полупроводниковыми, а затем начать процесс микроминиатюризации схем и элементов цифровых ЭВМ, что значительно повысило их быстродействие и надежность, уменьшило габариты, массу и потребляемую мощность. Одновременно с развитием конструктивно-технологических основ ЦВТ происходил процесс совершенствования структуры цифровых ЭВМ.

Теоретические основы ЦВТ были заложены исследованиями Дж. фон Неймана, К. Э. Шеннона, А. Тьюринга, Э. Поста, Д. Хаффмена, М. Фистера, В. И. Шестакова, М. А. Гаврилова, М. В. Келдыша, А. Н. Колмогорова, В. М. Глушкова, А. А. Маркова, А. А. Ляпунова, С. А. Лебедева, Ю. Я. Базилевского, Б. И. Рамеева, И. С. Брука, Э. В. Евреинова, а также разработками многих научно-производственных коллективов.

Современное понятие информации является одним из наиболее общих понятий науки и обозначает некоторые сведения, совокупность каких-либо данных, знаний и т. п. В вычислительной технике информация — свойство объектов и явлений (процессов) порождать многообразие состояний, которые посредством отражения передаются от одного объекта к другому и запечатлеваются в его структуре (возможно, в измененном виде). При этом под информацией понимают не сами объекты и явления, а их существенные и представительные признаки, их отражения в виде чисел, формул, описаний, чертежей, символов, образцов и других абстрактных характеристик. Сама по себе информация может быть отнесена к области абстрактных категорий, подобных, например, математическим формулам.

Передача информации (т. е. перенос ее на расстояние) предполагает наличие, как минимум, двух объектов — источника информации и приемника информации. Для передачи информации состояния источ-

ника должны быть каким-либо образом отражены во внешней среде, воздействующей, в свою очередь, на приемник. Во внешней среде информация представляется в материально-энергетической форме посредством множества состояний некоторых материальных объектов, называемых носителями информации, или сигналами. Отображение множества состояний источника во множество состояний носителя информации называют кодированием, а отображение множества состояний носителя во множество состояний приемника — декодированием. Образ состояния источника при выбранном способе кодирования называют кодом этого состояния.

Источники информации и создаваемые ими сообщения разделяют на непрерывные и дискретные. Непрерывные сообщения отображаются сигналами, представляющими собой какие-либо физические величины, изменяющиеся непрерывно и принимающие бесконечное число значений в некотором диапазоне. Такая форма представления информации используется в аналоговых моделирующих и вычислительных устройствах. Дискретные сообщения состоят из конечного множества элементов, формируемых источником информации и называемых буквами, или символами. Буквы отображаются сигналами, принимающими конечное число значений. В информационной и вычислительной технике понятие буквы включает в себя символы различных систем письменности, цифры, знаки препинания, математические знаки и др. Если элементам дискретного сообщения поставлены в соответствие цифры или некоторые их совокупности, то такое представление информации называют цифровым (числовым). Конечный набор букв образует алфавит. Число букв в алфавите определяет его объем или значность алфавита. Любую конечную последовательность букв некоторого алфавита принято называть словом. При этом число букв в слове называют его длиной.

В теории информации принято считать, что при передаче некоторого сообщения получают информацию, если состояние источника неизвестно заранее. Информация об априорно известном состоянии источника равна нулю. Таким образом, информация связана с вероятностью пребывания источника в том или ином состоянии. Пусть, в общем случае, источник может находиться в одном из k состояний $1, 2, \dots, k$, причем вероятность появления каждого состояния равна W_i ($i = 1, 2, \dots, k$). Тогда количество информации о состоянии источника

$$I = - \sum_{i=1}^k W_i \log_a W_i. \quad (1.1)$$

Если состояния источника равновероятны, то

$$I = \log_a k. \quad (1.2)$$

Основание a логарифма в выражениях (1.1) и (1.2) определяет единицу количества информации. При $a = 2$ соответствующая единица называется битом (bit — сокращение слов binary digit). Такая единица чаще всего используется в технике, что обусловлено наибольшей распространенностью алфавита с объемом, равным 2 (двухзначного алфавита), для представления дискретной информации.

Бит — такое количество информации, которое характеризует источник с двумя равновероятными состояниями.

Цифровые ЭВМ могут обрабатывать не только числовую информацию, но и информацию, заданную любыми другими символами. В современных цифровых ЭВМ символы чаще всего представляют словами из 8 бит — байтами. Количество информации в этом случае удобно измерять также в байтах. Слово длиной в один байт позволяет закодировать $2^8 = 256$ различных символов, чего вполне достаточно для всех практических ситуаций обработки символьной информации.

Количество информации, воспринимаемой цифровыми ЭВМ и хранящейся в их памяти, достигает миллиардов бит. Поэтому часто большие объемы информации выражают в специальных единицах, каждая из которых равна $2^{10} = 1024$. Такие единицы обозначаются К и М и читаются соответственно «кило» и «мега». При этом 1 мегабайт = 2^{10} килобайт = 2^{20} байт или сокращенно 1 Мбайт = 2^{10} Кбайт = 2^{20} байт. Аналогично определяются мега- и килобиты (1 Мбит = 2^{10} Кбит = 2^{20} бит).

Кроме перечисленных единиц информации пользователи цифровых ЭВМ употребляют такие единицы, как поле, массив, сегмент и другие, характеризующиеся отсутствием постоянного количественного эквивалента. При этом поле представляет собой группу бит, имеющую определенное значение. Совокупность полей, байтов или слов, объединяемых некоторым общим признаком (например, исходные данные для решения задачи), называют массивом. Сегмент — упорядоченная совокупность полей, бит, байтов, слов, массивов, сгруппированных с целью наименования.

Для характеристики скорости передачи информации используют единицу — бод, равную 1 бит/с.

1.2. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ АЛГОРИТМОВ

Обработка информации на цифровых ЭВМ представляет собой двуединный процесс изменения как формы представления, так и содержания исходной информации. Сущность этого процесса состоит в выполнении ряда операций в соответствии с некоторым алгоритмом, что в итоге приводит к получению результата или решения. Существует несколько определений понятия алгоритма, однако ни одно из них не является полным. Это понятие абстрагируется из опыта и относится к категории первоначальных понятий, таких, как множество, соответствие, натуральное число и др. Приведем некоторые наиболее распространенные определения в ЦВТ.

Алгоритм — точное, общепонятное предписание, определяющее процесс преобразования исходных данных в искомый результат.

Алгоритм — способ преобразования информации, задаваемый с помощью конечной системы правил.

Алгоритм — алфавитный оператор, задаваемый с помощью конечной системы правил. При этом алфавитным оператором (АО) называют всякое соответствие между словами входного алфавита и словами этого же или другого выходного алфавита. Понятие АО общее, к нему могут быть сведены все процессы преобразования информации.

Отличие между АО и алгоритмом в том, что АО устанавливает лишь соответствие между словами, а алгоритм указывает способ достижения такого соответствия.

Два АО считаются равными, если они определены на одних и тех же множествах входных и выходных слов и сопоставляют одному и тому же входному слову одинаковые выходные слова.

Два алгоритма считаются равными, если равны соответствующие им АО и совпадают системы правил, задающие эти алгоритмы. Два алгоритма называются эквивалентными, если совпадают их АО, но отличаются способы их задания.

Анализ элементарных операций по обработке информации, встречающихся в реальных алгоритмах, показывает, что их можно разделить на две группы: арифметические и логические. Арифметические операции выполняют непосредственное преобразование информации, а логические определяют направление процесса обработки информации. В алгоритмах арифметические и логические операции чередуются в определенной последовательности. Если выполнение алгоритма сводится к арифметическим операциям, то такой алгоритм называют численным.

Алгоритмы обладают свойствами определенности, массовости и результативности. Свойство определенности выражает тот факт, что совокупность операций, выполняемых в соответствии с некоторым алгоритмом, не допускает никакого произвола относительно их последовательности и толкования, т. е. является детерминированным процессом. Массовость алгоритма означает возможность решения с его помощью целого класса задач с изменяющимися исходными данными. Результативность алгоритма состоит в том, что искомый результат может быть получен с помощью алгоритма путем выполнения конечного числа операций при всех допустимых значениях исходных данных. Рассмотренные свойства алгоритма являются эмпирическими и их нельзя считать определением понятия алгоритма.

В теории алгоритмов принято называть оператором последовательность величин, объединенных знаками операций, а операндами — величины, входящие в оператор. Простейшими операторами являются унарный (сингулярный) и бинарный, определяющие операции соответственно с одним и двумя операндами. Из элементарных операторов строят сложные операторы. Выполнение оператора R_j в последовательности операторов называется j -м шагом реализации алгоритма. Выбор следующего оператора часто зависит от результата выполнения предыдущего оператора. Для этого вводят специальные операторы перехода. Их использование дает возможность выполнять «ветвящиеся» и циклические (многократно повторяющиеся) алгоритмы.

Существует несколько общих способов задания алгоритмов, называемых алгоритмическими системами (АС). АС условно разделяют на алгебраические и геометрические. К первым относят АС, в которых алгоритмы рассматриваются в виде линейных текстов (рекурсивные функции, машины Тьюринга, операторные системы Ван-Хао, Ляпунова, логические схемы алгоритмов Янова). К геометрическим АС относят такие АС, где алгоритмы строятся как множества, между которыми вводятся связи, имеющие характер отображений или бинарных

отношений (представления нормальных алгоритмов Маркова в виде граф-схем, блок-схемный метод алгоритмизации и др.). Например, в операторных алгоритмах (операторных описаниях) Ляпунова буквами обозначают отдельные действия алгоритма по переработке информации — операторы и проверяемые логические условия. Логическими схемами алгоритмов называют выражения, составленные из операторов и логических условий, следующих один за другим. Последовательное выполнение нескольких операторов обозначается как их произведение, причем левый оператор действует ранее правого. В такой линейной записи алгоритма оператор отличается от логического условия

тем, что после последних ставят стрелку, направленную вверх и снабженную цифровым индексом. Если логическое условие A выполняется ($A = 1$), то осуществляется переход к оператору или логическому условию, указываемому стрелкой. Если же условие A не выполняется ($A = 0$), то осуществляется переход к оператору и логическому условию, записанному непосредственно за условием A . Например, выражение

$$AB \downarrow {}^3C \uparrow {}^1DE \uparrow {}^2 \downarrow {}^1F \downarrow {}^2G \uparrow {}^3HI \quad (1.3)$$

означает, что после выполнения операторов A и B необходимо проверить логическое условие C . Если $C = 1$, то далее надо перейти к оператору F и логическому условию G ; если же $C = 0$, то выполняют оператор D и затем проверяют условие E и т. д. Операторные алгоритмы допускают некоторые эквивалентные преобразования, что позволяет в ряде случаев отыскать наиболее рациональные алгоритмы с точки зрения удобства их реализации. Набор операторов в АС должен быть функционально полным, т. е. достаточным для записи любого алгоритма из заданного класса алгоритмов.

Иногда логическую схему алгоритма представляют в виде графа. Выражению (1.3) соответствует граф на рис. 1.1. Графическая форма операторного алгоритма очень удобна для восприятия его человеком и позволяет использовать топологические методы для эквивалентных преобразований алгоритмов.

В ЦВТ для записи алгоритмов используют, главным образом, операторные АС. С возникновением цифровых ЭВМ появились новые АС — алгоритмические языки и языки программирования (АЯ и ЯП). Непременными атрибутами любого АЯ являются алфавит, синтаксис и семантика. Алфавит АЯ — это множество символов, используемых в нем. Система правил для построения некоторых конструкций в языке составляет его синтаксис. Семантика определяет связь между синтаксически верными конструкциями АЯ и соответствующими им объектами (иначе: семантика раскрывает смысловое значение языковых конструкций). АЯ называется универсальным, если в нем может быть описан любой алгоритм.

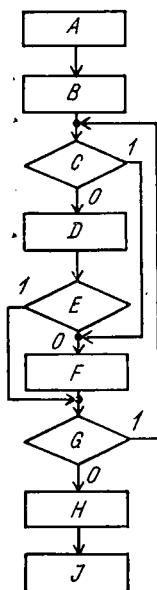


Рис. 1.1.

Необходимость разработки АЯ была вызвана тем, что записи алгоритмов на естественных языках часто оказываются весьма избыточными, неоднозначными и противоречивыми. Этим недостаткам лишены формальные языки, в которых форма каждого предложения определяет его содержание. Как правило, формальные языки строятся так, что свойства некоторого языка не могут быть описаны на этом же языке. Для этого нужен другой формальный язык, называемый метаязыком. Описываемый же язык называют языком-объектом. Иногда сам метаязык может быть описан с помощью другого метаязыка (т. е. метаметаязыка). Метаязык, служащий для описания синтаксиса языка-объекта, называют метасинтаксическим, а метаязык, описывающий семантику — метасемантическим. В некоторых случаях для этих целей используют один и тот же метаязык.

Для построения метасинтаксических языков чаще всего используют способ, получивший название нормальной формы (записи) Бэкуса (Бэкус — американский математик, один из создателей языка АЛГОЛ-60). Метасинтаксический язык в этом случае состоит из конечного числа предложений, называемых металингвистическими формулами Бэкуса. Между левой и правой частями формулы Бэкуса ставится знак $::=$, читаемый как «по определению есть» или «это есть». Левая часть формулы представляет собой наименование определяемого синтаксического понятия, заключенного в угловые скобки $\langle \rangle$. В правой части указывается последовательность символов и выражений, составляющих определяемое понятие. Здесь также используются угловые скобки и знак $|$, читаемый как «или». Остальные метасимволы могут быть выбраны произвольно разработчиком языка-объекта. Под метасимволами понимаются фразы естественного языка. Например, синтаксис языка-объекта для описания типов пассажирских самолетов, используемых на линиях Аэрофлота, с помощью нотации Бэкуса можно задать так:

$$\begin{aligned} \langle \text{тип самолета} \rangle &::= \langle \text{аббревиатура} \rangle \langle \text{число} \rangle \\ \langle \text{аббревиатура} \rangle &::= \langle \text{АН} \rangle \langle \text{ИЛ} \rangle \langle \text{ТУ} \rangle \langle \text{ЯК} \rangle \\ \langle \text{число} \rangle &::= \langle \text{цифра} \rangle \langle \text{цифра} \rangle \langle \text{цифра} \rangle \langle \text{цифра} \rangle \\ &\quad \langle \text{цифра} \rangle \langle \text{цифра} \rangle \\ \langle \text{цифра} \rangle &::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \end{aligned}$$

Из приведенного следует, что запись ИЛ 849 не противоречит синтаксису, хотя и лишена смысла с точки зрения семантики, так как таких самолетов не существует.

ЯП являются также формальными языками и служат для связи человека с цифровыми ЭВМ. Возникновение и развитие ЯП связано с развитием ЦВТ и расширением сферы применения ее средств. Первыми ЯП были внутренние машинные языки, использующие цифровой алфавит. Операторы машинного языка называются командами. Команды выполняются схемами цифровых ЭВМ, что приводит в итоге к реализации алгоритма. Записи алгоритмов на машинном языке очень детализированы и неудобны для чтения человека. В дальнейшем были разработаны укрупненные и алгебраизированные ЯП, позволяющие формализовать перевод записей с ЯП на машинный язык. Алфавит

и синтаксис таких языков близки к естественному языку. Вместе с тем, синтаксические и семантические правила в них строги и категоричны, что обеспечивает однозначность перевода алгоритма на машинный язык. Существующие ЯП делят на три класса: машинно-ориентированные, процедурно-ориентированные и проблемно-ориентированные.

К машинно-ориентированным ЯП относятся языки, использование которых требует знания принципов и особенностей функционирования конкретных цифровых ЭВМ. В этом классе ЯП выделяют два подкласса: мнемокоды (языки ассемблера) и автокоды. Мнемокоды используют мнемонические обозначения (обычно буквенные или десятичные) операций и операндов. Структура операторов мнемокода аналогична структуре команд машинного языка. Введение мнемоники в некоторой степени упрощает процесс записи алгоритмов по сравнению с машинными языками. Автокоды отличаются от мнемокодов тем, что позволяют уже однажды сделанную запись алгоритма в дальнейшем использовать как отдельный оператор. Общими недостатками машинно-ориентированных языков являются громоздкость записей алгоритмов и различие в записях одних и тех же алгоритмов для различных цифровых ЭВМ.

Процедурно-ориентированные языки (АЛГОЛ, ФОРТРАН, КОБОЛ, и др.) в подавляющем большинстве не зависят от цифровой ЭВМ, на которой реализуются алгоритмы. Эти языки допускают построение различных форм операторов, благодаря чему записи правил вычислений (процедур) с помощью таких языков приближаются к записям на языке математики.

Проблемно-ориентированные языки предназначены для записи алгоритмов решения некоторого довольно узкого круга задач и используют средства, близкие к терминологии пользователя. Строятся эти языки так, чтобы максимально упростить запись алгоритмов. Примерами таких языков являются ЛЯПАС, СИМУЛА, СЛЕНГ.

Алгоритм, записанный на каком-либо машинном языке, допустимом для данной цифровой ЭВМ, называют программой. Цифровая ЭВМ может выполнять программы, записанные только на ее машинном языке. Поэтому все алгоритмы, записанные на других языках, перед их реализацией должны быть переведены на машинный язык. Такой перевод может осуществляться непосредственно программистом или же автоматически с помощью специальной программы — транслятора (программирующей программы, компилирующей программы).

1.3. ПРИНЦИПЫ ПОСТРОЕНИЯ И КЛАССИФИКАЦИЯ ЦИФРОВЫХ ЭВМ

В настоящее время электронными вычислительными машинами принято называть комплексы технических средств, которые в соответствии с некоторой программой реализуют математические операции по обработке информации, представленной в цифровой форме. Главным отличительным признаком цифровых ЭВМ является то, что в них автоматизирован процесс вычислений за счет использования принципа программного управления (основные идеи программного управления

изложены английским математиком Ч. Бэббиджем в 1834 г.) и принципа хранимой в памяти цифровых ЭВМ программы вычислений (сформулирован в 1946 г. американским ученым Дж. фон Нейманом и независимо от него в 1950 г. советским ученым академиком С. А. Лебедевым).

Решение задачи на цифровой ЭВМ сводится к выполнению арифметических и логических операций над исходными данными и промежуточными результатами. При этом исходные данные и промежуточные результаты обычно представляются в виде совокупности цифр в некоторой позиционной системе счисления, которые, в свою очередь, отображаются чаще всего электрическими сигналами с определенным числом уровней квантования. Перед решением задачи на цифровой ЭВМ алгоритм решения записывается в виде последовательности тех простейших операций, выполнение которых предусмотрено конструкцией машины, т. е. составляется программа решения. Вид программы не зависит от конкретных значений чисел, представляющих обрабатываемую информацию. Поэтому одна и та же программа пригодна для решения множества задач по заданному алгоритму.

Сущность принципа программного управления состоит в том, что процесс обработки информации осуществляется на основе информации, заданной для управления этим процессом. В гипотетической машине Ч. Бэббиджа и в первой реальной цифровой ЭВМ с программным управлением «Марк I» программа заносилась на перфоленгу. Машина последовательно считывала с перфоленты и расшифровывала информацию об управлении процессом вычислений.

Любая программа состоит из отдельных команд, каждая из которых определяет действия машины по выполнению какой-либо одной операции. В соответствии с принципом хранимой в памяти машины программы вычислений, команды кодируются цифрами и хранятся в запоминающем устройстве (ЗУ) цифровой ЭВМ наряду с обрабатываемой информацией. При этом в командах указывается код подлежащей выполнению операции и номера (адреса) ячеек ЗУ, в которых хранятся используемые в операции числа и ее результат. Хранение программы в ЗУ машины обеспечивает возможность многократного оперативного использования одних и тех же команд и выполнения операций над самими командами.

Все вычислительные операции в цифровых ЭВМ выполняются либо с помощью схем, специально предназначенных для реализации этих операций — так называемых аппаратных средств, либо программным способом, т. е. путем разложения заданной операции на ряд простейших, реализуемых аппаратными средствами. Набор операций, реализуемых в цифровых ЭВМ аппаратными средствами, называют системой команд. Вся же совокупность операций, реализуемых как аппаратными, так и программными способами, принято называть операционными ресурсами цифровых ЭВМ. При разработке цифровой ЭВМ стремятся получить такое сочетание аппаратных и программных средств, при котором обеспечивается наибольшая эффективность машины в целом.

Цифровые ЭВМ с программным управлением разделяют на специализированные и алгоритмически универсальные. Специализирован-

ные цифровые ЭВМ (цифровые ЭВМ с жесткой программой) предназначены для выполнения одного или нескольких вполне определенных вычислительных алгоритмов. Программа в специализированных цифровых ЭВМ является постоянно заложенной тем или иным способом в конструкцию машины. Эти цифровые ЭВМ строятся для разовых применений (цифровые ЭВМ на луноходе), а также для таких применений, где доминирующую роль играет одно из следующих качеств: простота технической реализации, высокая надежность и быстродействие, низкая стоимость, малые габариты и масса (например, самолетные средства ЦВТ). К специализированным относятся и так называемые неалгоритмические цифровые ЭВМ, не реализующие принцип программного управления. В таких цифровых ЭВМ используется цифровое представление информации, а состав вычислительных блоков и связи между ними определяются математическим описанием процесса обработки информации.

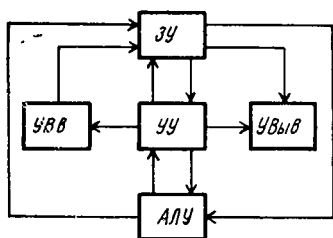


Рис. 1.2.

Алгоритмически универсальные (или просто универсальные) цифровые ЭВМ способны, в принципе, реализовать любой алгоритм обработки информации. Приведенная классификация цифровых ЭВМ в качестве классификационного признака использует способы управления вычислительным процессом и является наиболее общей.

В состав универсальной цифровой ЭВМ (рис. 1.2) входят ЗУ, арифметикологическое устройство (АЛУ), устройство управления (УУ) и устройства ввода (УВВ) и вывода (УВыв) информации. ЗУ служит для хранения исходных данных, промежуточных и конечных результатов, а также программ обработки информации. Емкостью памяти цифровой ЭВМ (емкостью ЗУ машины в целом) называют максимальное количество информации, которое может быть записано в ЗУ. Емкость памяти является одной из основных характеристик не только собственно ЗУ, но и всей машины. Обычно память цифровой ЭВМ разделяют на две части — оперативное ЗУ (ОЗУ) и внешнее ЗУ (ВЗУ). ОЗУ характеризуется более высоким быстродействием по сравнению с ВЗУ, но емкость его значительно меньше емкости ВЗУ.

В АЛУ выполняются все операции по обработке поступающей из ЗУ информации. Число операций, выполняемых АЛУ в единицу времени, определяет производительность АЛУ и цифровой ЭВМ в целом. В настоящее время принято оценивать производительность цифровой ЭВМ по способу Гибсона. Сущность этого способа заключается в определении весов отдельных операций γ_i на основе частоты их появления в программах и дальнейшем подсчете среднего числа операций в единицу времени по формуле

$$\rho = \frac{\sum_{i=1}^n \gamma_i}{\sum_{i=1}^n \gamma_i t_i},$$

где ρ — производительность цифровой ЭВМ; t_i — время выполнения i -й операции; n — число операций, выполняемых машиной. Набор весов γ_i , характеризующий некоторый класс задач, решаемых на цифровой ЭВМ, называют смесью Гибсона. В зависимости от значения ρ ЭВМ разделяют на большие, средние и малые. Большие цифровые ЭВМ имеют производительность не менее 500 000 операций в секунду. Производительность средних цифровых ЭВМ составляет от 100 000 до 500 000 операций в секунду. Малые цифровые ЭВМ выполняют менее 100 000 операций в секунду.

Подкласс малых цифровых ЭВМ составляют появившиеся в последние годы так называемые мини-ЭВМ и микро-ЭВМ — универсальные машины, характеризующиеся относительно узким диапазоном обрабатываемых чисел, небольшой емкостью ЗУ, малыми габаритно-весовыми показателями, невысокой стоимостью. УУ обеспечивает выборку обрабатываемой информации и команд программы из ЗУ, расшифровку команд и формирование необходимых последовательностей управляющих сигналов. Эти сигналы инициируют работу отдельных устройств цифровых ЭВМ, что приводит к выполнению действий, предписываемых командой. Так как все процессы по обработке информации сосредоточены, в основном, в АЛУ и УУ, то эти два устройства в совокупности часто называют процессором цифровой ЭВМ. УВв предназначено для преобразования исходной информации в форму, принятую в машине. УВыв — преобразует полученную в результате вычислений информацию из машинной формы в форму, удобную для восприятия ее человеком, исполнительными механизмами и т. п. В совокупности устройства ввода и вывода информации иногда называют внешними или периферийными устройствами.

Существует еще один класс электронных цифровых ЭВМ — с ручным управлением. К таким цифровым ЭВМ относятся настольные клавишные вычислительные машины типа электронных арифмометров, которые предназначены для решения простейших задач с небольшим объемом вычислений.

Цифровые ЭВМ классифицируют также по их назначению: для научных и технических расчетов, для обработки данных, для инженерных расчетов, для управления производственными и технологическими процессами. В некоторых случаях цифровые ЭВМ объединяют в классы по их технико-экономическим характеристикам (емкости ЗУ, стоимости и т. п.), структуре команд (одноадресные, двухадресные и т. д.), способу организации вычислительного процесса (однопрограммные и многопрограммные) и другим признакам.

Общепринятой также является классификация цифровых ЭВМ по конструктивно-технологическим особенностям их реализации, согласно которой созданные до настоящего времени машины разделяют на несколько поколений. К первому поколению (1950—1958 гг.) относят цифровые ЭВМ, построенные на электронных лампах с использованием дискретных радиодеталей и методов навесного монтажа. В машинах второго поколения (1959—1967 гг.) транзисторы полностью заменили электронные лампы. Схемы цифровых ЭВМ второго поколения изготавливались методами печатного монтажа. Производимые в настоящее время цифровые ЭВМ являются машинами третьего поко-

ления. В этих машинах большинство транзисторов и дискретных радиодеталей заменено интегральными микросхемами, а все соединения между ними выполняются методами многослойного печатного монтажа.

Смена поколений цифровых ЭВМ происходит каждые восемь—десять лет. Сейчас ведется интенсивная разработка принципов построения и исследование особенностей конструктивно-технологической реализации машин четвертого и пятого поколений на интегральных схемах с высокой степенью интеграции. Применение прогрессивных технологических методов изготовления схем цифровых ЭВМ позволяет не только значительно улучшить их качество, но и реализовать схемы с новыми возможностями. Это обуславливает, в свою очередь, существенные отличия в составе аппаратных средств машин каждого нового поколения по сравнению с предыдущим. В качестве примера можно указать для машин первых четырех поколений тенденцию роста числа операций, реализуемых аппаратными средствами. Цифровые ЭВМ пятого поколения, по-видимому, будут представлять собой так называемые рекурсивные машины, характерные принципиальным отсутствием ограничений на число используемых элементов, программной перестройкой обрабатываемых блоков (любая группа элементов может образовывать независимый вычислитель) и памяти в соответствии со структурой данных и программ решаемых задач, параллельным выполнением всех программ и их элементов, для которых имеются в наличии исходные данные, языком программирования, приближающимся к обычному языку и пр.

Приведенная периодизация развития цифровых ЭВМ отражает не только изменения в технологии изготовления и составе аппаратных средств, но и глубокие отличия в составе программных средств и архитектуре цифровых ЭВМ в целом. При этом под архитектурой цифровых ЭВМ понимают весь комплекс программных и аппаратных средств, а также организацию их взаимодействия в вычислительном процессе. Термин же «структура цифровых ЭВМ» обычно используют для обозначения состава, внутренней организации и функций аппаратных средств.

Программы, составленные для решения некоторых задач, называют рабочими программами. Наряду с ними в памяти цифровых ЭВМ хранятся служебные программы, предназначенные для расширения операционных ресурсов, обеспечения требуемых режимов работы машины, автоматизации перевода программы с языка программирования на машинный язык и др. Совокупность служебных программ получила название математического (программного) обеспечения. Цифровые ЭВМ и их математическое обеспечение образуют вычислительную систему.

Число радиокомпонентов в современных вычислительных системах достигает десятков миллионов штук, а число команд в программах — сотен тысяч. Поэтому для описания структуры и функционирования вычислительных систем в целом практически невозможно использовать ни один из способов, позволяющих описать состав, связи, физические и информационные процессы для некоторого небольшого множества радиокомпонентов. Однако, как аппаратные средства

цифровых ЭВМ, так и программные характеризуются иерархическим принципом построения, т. е. наличием определенных совокупностей максимальных неделимых единиц (элементов), которые, будучи объединены в подсистемы некоторого уровня, могут рассматриваться в качестве элементов в подсистемах более высокого уровня и т. д. Это дает возможность использовать иерархический принцип описания, согласно которому каждому уровню подсистем соответствует определенная степень детализации, достаточная для описания структуры с точностью до элементов данного уровня.

В ЦВТ самым низким иерархическим уровнем является уровень электрических схем, где в качестве элементов рассматриваются радиокомпоненты. Степень детализации здесь такова, что позволяет анализировать зависимости между электрическими параметрами входов и выходов схем методами теории электрических цепей. Этот уровень не используют для описания процессов обработки информации.

На уровне логических схем абстрагируются от конкретных свойств физических явлений, положенных в основу построения электрических схем. Наименьшими структурными единицами здесь считаются логические и запоминающие элементы. Процессы обработки информации на этом уровне детализируются до операций над отдельными буквами алфавитов путем использования теории переключательных функций и теории конечных автоматов.

Следующий иерархический уровень — уровень операционных схем — предполагает детализацию информационных процессов в средствах ЦВТ до некоторых элементарных операций (микроопераций) над отдельными словами. В качестве элементов на этом уровне используют узлы (операционные схемы), построенные из логических и запоминающих элементов. Каждый узел выполняет одну или несколько микроопераций. Для описания алгоритмов обработки информации на этом уровне используют простейшие формальные языки типа операторных описаний.

Уровень структурных схем позволяет описывать состав и взаимодействие оборудования цифровых ЭВМ с детализацией до отдельных операций из системы команд, выполняемых как последовательность микроопераций. На этом уровне в качестве элементов рассматривают операционные блоки, объединяющие несколько узлов. Средством описания информационных процессов здесь служат машинно-ориентированные языки.

Программный уровень является самым высоким иерархическим уровнем описания процессов обработки информации в цифровых ЭВМ, детализованным до команд, составляющих операционные ресурсы машины, или до отдельных программ.

1.4. СИСТЕМЫ СЧИСЛЕНИЯ И СПОСОБЫ ПЕРЕВОДА ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

Системой счисления (счислением, нумерацией) называют систему приемов и правил, позволяющих установить взаимно однозначное соответствие между любым числом и его представлением в виде совокупности конечного числа символов. Множество символов, использу-

емых для такого представления, называют цифрами. Каждой цифре соответствует определенное количество, выражаемое этой цифрой и называемое численным значением или количественным эквивалентом данной цифры.

Различают непозиционные и позиционные системы счисления. В непозиционных системах имеет место однозначное соответствие между цифрами и их количественными эквивалентами, а любое число определяется как некоторая функция от количественных эквивалентов совокупности цифр, представляющих это число. Если в качестве этой функции используется функция сложения, то систему называют аддитивной, если же используется функция умножения, систему называют мультипликативной. Примерами непозиционных аддитивных систем счисления являются римская система и единичная (унитарная) система.

Однако непозиционные системы не получили сколько-либо значительного распространения в ЦВТ, так как они характеризуются очень сложными и громоздкими алгоритмами представления чисел и выполнения арифметических операций.

Систему счисления называют позиционной, если одна и та же цифра может соответствовать различным количественным эквивалентам в зависимости от номера местоположения (разряда) этой цифры в совокупности цифр, представляющих заданное число. Позиционные системы разделяют на однородные и смешанные. Во всех разрядах числа, представленного в однородной системе, используются цифры из одного и того же множества. Например, в обычной десятичной системе во всех разрядах любого числа используются цифры из множества $\{0, 1, \dots, 9\}$, в двоичной системе — цифры из множества $\{0, 1\}$ и т. п. В смешанных системах множества цифр различны для разных разрядов. Примерами смешанных систем являются система для измерения углов и дуг (в разрядах минут и секунд могут быть использованы 60 различных цифр, в разряде градусов — 360 различных цифр), система измерения времени, например, в тысячелетиях, столетиях, годах, месяцах, неделях, сутках, часах, минутах, секундах, десятых, сотых долях секунд.

Когда в позиционной системе для каждой цифры имеется отдельный символ, ее называют системой с непосредственным представлением цифр. Существуют также позиционные системы с кодированным представлением цифр. В таких системах количество символов меньше, чем количество цифр, а каждая цифра кодируется определенной комбинацией нескольких символов, представляющих собой, как правило, цифры другой системы счисления. Например, в смешанной системе измерения дуг и углов каждая цифра разряда градусов кодируется тремя десятичными символами, а в разрядах минут и секунд — двумя десятичными символами.

Преимущественное распространение в ЦВТ получили позиционные однородные системы счисления. В такой системе с непосредственным представлением цифр любое число X выражается в виде

$$X = k^n x_n + k^{n-1} x_{n-1} + \dots + k^1 x_1 + k^0 x_0 + \\ + k^{-1} x_{-1} + \dots + k^{-m} x_{-m} = \sum_{i=-m}^n k^i x_i, \quad (1.4)$$

где k — основание системы счисления, т. е. количество цифр, используемых в данной системе ($k = 2, 3, \dots$); x_i — цифры i -го разряда представления числа в системе с основанием k . Величину k^i принято называть весом i -го разряда. Поскольку значение k известно заранее, то выражение (1.4) записывает в более простой форме

$$X = x_n x_{n-1} \dots x_1 x_0, \quad x_{-1} \dots x_{-m}. \quad (1.5)$$

В выражении (1.5) запятая отделяет целую часть числа ($n + 1$ разрядов) от дробной (m разрядов), а вес i -го разряда в k раз больше веса $i - 1$ -го разряда. Такую систему счисления называют системой с естественным порядком весов. Существуют системы с искусственным порядком весов, для которых указанное соотношение весов соседних разрядов не является обязательным. Известны, например, системы с искусственным порядком весов, в которых целое положительное число X выражается так:

$$X = \sum_{i=0}^{j-1} k^i x_i + \sum_{i=j}^n k^{i-j} (k^j - 1) x_i, \quad (j = 1, 2, \dots, n).$$

Позиционные однородные системы с естественным порядком весов разделяют также на системы с натуральными, отрицательными, дробными и комплексными основаниями. Представление числа в какой-либо системе счисления называют кодом.

Системы счисления с натуральным основанием, в которых имеет место взаимно однозначное соответствие между числом и его кодом конечной длины, получаемым за конечное число шагов, называют каноническими. В канонических системах счисления при записи чисел в каждом разряде может быть использована одна из k различных цифр, включая цифру 0. Позиционные системы счисления с естественным порядком весов, в которых количество различных допустимых цифр превышает основание k , называют избыточными. Избыточные системы используют в ЦВТ с целью повышения надежности обработки информации и скорости выполнения арифметических операций. В таких системах одному числу может соответствовать несколько кодов конечной длины, но одному коду соответствует одно число. Если количество различных цифр в избыточной системе равно $k + 2$ и при этом $k = 2l$, $x_i \in \{-l, -l + 1, \dots, 0, \dots, l, l + 1\}$, или $k = 2l + 1$, $x_i \in \{-l - 1, -l, \dots, 0, \dots, l, l + 1\}$, то такую систему называют квазиканонической. Избыточную систему, в которой количество допустимых цифр в каждом разряде составляет $k + 1$ и при этом $k = 2l$, $x_i \in \{-l, -l + 1, \dots, 0, l - 1, l\}$ или $k = 2l + 1$, $x_i \in \{-l, -l + 1, \dots, l, l + 1\}$, называют модифицированной квазиканонической.

Определим максимальное число $X_{\max}$, которое может быть записано в форме (1.4). Для этого в (1.4) вместо каждого x_i подставим максимальную цифру, равную $k - 1$,

$$\begin{aligned} X_{\max} &= \sum_{i=-m}^n k^i (k - 1) = (k - 1) \sum_{i=-m}^n k^i = (k - 1) \frac{k^{n+1} - k^{-m}}{k - 1} = \\ &= k^{n+1} - k^{-m}. \end{aligned}$$

Минимальное положительное отличающееся от нуля число, представимое согласно (1.4), равно

$$X_{\min} = k^{-m}.$$

Очевидно, что количество различных чисел, которое можно представить в форме (1.4), определяется как

$$M = \frac{X_{\max}}{X_{\min}} + 1 = \frac{k^{n+1} - k^{-m} + k^{-m}}{k^{-m}} = k^{m+n+1}.$$

Отсюда следует, что для представления N различных чисел в канонической системе с основанием k необходимо не менее

$$r = \lceil \log_k N \rceil$$

разрядов. Здесь и далее запись $\lceil A \rceil$ означает округление в сторону ближайшего большего целого числа, если A — дробное, т. е. $\lceil A \rceil = \text{int } A + 1$.

Как в системах счисления с естественным порядком весов, так и в системах с искусственным порядком весов каждый разряд в записи числа имеет свой постоянный вес. Поэтому такие системы называют взвешенными или весомазначными. Возможны также системы, в которых каждая отдельная цифра не имеет численного значения и лишь некоторым комбинациям цифр соответствует определенный количественный эквивалент. Подобные системы называют символическими (невзвешенными, невесомозначными). Главным преимуществом взвешенных позиционных систем счисления является удобство представления чисел и простота выполнения арифметических операций. Недостаток таких систем — невозможность выполнения арифметических операций как поразрядных. Поразрядной операцией называют такую, результат которой в любом разряде не зависит от результата выполнения этой операции во всех остальных разрядах. Примером системы счисления, в которой арифметические операции могут выполняться поразрядно, является символическая система счисления остаточных классов (СОК). В СОК целое положительное число представляется в виде набора остатков (вычетов) от его деления на ряд взаимно простых чисел — оснований системы:

$$X = (x_1, x_2, \dots, x_m),$$

где x_i — целочисленный остаток от деления X на основание p_i , т. е.

$$x_i = \text{rest } X \pmod{p_i} = X - \left\lfloor \frac{X}{p_i} \right\rfloor p_i, \quad (i = 1, 2, \dots, m).$$

Запись $\lceil A \rceil$ означает округление в сторону ближайшего меньшего числа, если A дробное ($\lceil A \rceil = \text{int } A$).

Максимальное число, представимое в СОК, определяется так

$$X_{\max} = \prod_{i=1}^m p_i$$

Однако применение СОК в ЦВТ ограничивается рядом ее недостатков, а именно: отсутствием наглядной связи между числом и его представлением в СОК, не позволяющим оценивать количественный эквива-

лент числа без перевода его в позиционную систему; трудность выполнения операций деления и округления; отсутствием признаков выхода результата за допустимые пределы.

Алгоритмы выполнения арифметических операций в канонической взвешенной системе счисления с основанием k устанавливаются аналогично соответствующим алгоритмам в обычной десятичной системе счисления. Результаты сложения, вычитания и умножения цифр x и y в одном разряде представляются двумя цифрами: цифрой s результата соответствующей операции в данном разряде и цифрой r переноса (заема) в старшем разряде. Правила формирования цифр s и r при любом k для всех указанных операций можно записать так:

а) для сложения

$$\begin{aligned} s_{\text{слож}} &= \begin{cases} x + y & \text{при } x + y < k; \\ x + y - k & \text{при } x + y \geq k, \end{cases} \\ r_{\text{слож}} &= \begin{cases} 0 & \text{при } x + y < k; \\ 1 & \text{при } x + y \geq k, \end{cases} \end{aligned} \quad (1.6)$$

б) для вычитания

$$\begin{aligned} s_{\text{выч}} &= \begin{cases} x - y & \text{при } x \geq y; \\ k + x - y & \text{при } x < y, \end{cases} \\ r_{\text{выч}} &= \begin{cases} 0 & \text{при } x \geq y; \\ 1 & \text{при } x < y, \end{cases} \end{aligned} \quad (1.7)$$

в) для умножения

$$\begin{aligned} s_{\text{умн}} &= \begin{cases} xy & \text{при } xy < k; \\ xy - \left(\left\lfloor \frac{xy}{k} \right\rfloor - 1 \right) k & \text{при } xy \geq k, \end{cases} \\ r_{\text{умн}} &= \begin{cases} 0 & \text{при } xy < k; \\ \left\lfloor \frac{xy}{k} \right\rfloor - 1 & \text{при } xy \geq k. \end{cases} \end{aligned} \quad (1.8)$$

Операции, выполняемые в соответствии с (1.6)—(1.8), называют также операциями сложения, вычитания и умножения по модулю k (сокращенно: по $\text{mod } k$). При $k \neq 2$ функция $r_{\text{умн}} = 0$, что упрощает выполнение операции умножения в двоичной системе счисления.

Что касается операции деления, то ее выполнение не может быть сведено к операциям над отдельными цифрами. Поэтому при любом $k \geq 2$ деление выполняется по аналогии с делением в десятичной системе счисления, но с использованием правил формирования цифр $s_{\text{умн}}$, $s_{\text{выч}}$, $r_{\text{умн}}$, $r_{\text{выч}}$ для k -й системы. Заметим, что в случае $k > 10$ в качестве некоторых цифр необходимо вводить специальные символы, которых нет среди десятичных. Обычно для этого используют буквы.

Выбор системы счисления для цифровых ЭВМ оказывает влияние не только на алгоритмы выполнения операций, но и на структуру узлов, блоков и устройств машины. Исторически сложилось так, что наибольшее распространение в практике вычислительных работ полу-

чила десятичная позиционная однородная система счисления. Однако эта система не является самой удобной для реализации ее в ЭВМ, где, как правило, используют системы счисления с недесятичным основанием — двоичную, восьмиричную и другие, а также двоично-кодированные системы (т. е. такие системы, цифры которых закодированы двоичными символами). Объясняется это в первую очередь простотой, высокой надежностью и высоким быстродействием технических средств, используемых для представления цифр и выполнения операций в двоичной системе счисления. Из сравнения цифровых ЭВМ различного назначения следует, что обычно машины, решающие задачи с относительно большим числом операций ввода—вывода, приходящихся на одну операцию по обработке информации, строят с использованием десятичной (двоично-десятичной) системы счисления. В машинах же, решающих задачи, где время ввода—вывода относительно невелико по сравнению со временем обработки информации, применяют двоичную систему счисления. В связи с этим возникает задача преобразования (перевода) чисел из одной системы счисления в другую.

Нетрудно заметить, что правая часть выражения (1.4) определяет правило вычисления количественного эквивалента числа, записанного в форме (1.5). На этом основан один из алгоритмов перевода чисел из одной позиционной системы в другую. Обозначим $X_{(k_1)}$ число в k_1 -й системе счисления. Для перевода числа в систему с основанием k_2 необходимо записать $X_{(k_1)}$ в форме (1.4), заменить цифры x_i и основание k_1 их k_2 -ми представлениями, а затем выполнить операции умножения и сложения в системе с основанием k_2 . Рассмотрим примеры.

Пример 1. Пусть задано число $633_{(7)}$ и $k_2 = 2$, необходимо найти $X_{(2)}$. Согласно вышеизложенному получим

$$X_{(7)} = 633_{(7)} = 7^2 \cdot 6 + 7^1 \cdot 3 + 7^0 \cdot 3 = (111)^2 \cdot 110 + (111)^1 \cdot 011 + (111)^0 \cdot 011 = 110001 \cdot 110 + 111 \cdot 011 + 011 = 100111110_{(2)}.$$

Пример 2. Перевести в десятичную систему двоичное число из предыдущего примера

$$X_2 = 100111110_{(2)} = 2^8 \cdot 1 + 2^7 \cdot 0 + 2^6 \cdot 0 + 2^5 \cdot 1 + 2^4 \cdot 1 + 2^3 \cdot \times 1 + 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0 = 256 + 32 + 16 + 8 + 4 + 2 = 318_{(10)}.$$

Пример 3. Задано число $36,57_{(10)}$, необходимо записать это число в двоичной системе

$$X_{(10)} = 36,57_{(10)} = 10^1 \cdot 3 + 10^0 \cdot 6 + 10^{-1} \cdot 5 + 10^{-2} \cdot 7 = (1010)^1 \times 0011 + (1010)^0 \cdot 0110 + (1010)^{-1} \cdot 0101 + (1010)^{-2} \cdot 0111 = 011110 + 000110 + 0,10000 + 0,00010001... = 100100,10010001..._{(2)}.$$

Описанный способ перевода чисел из одной системы в другую получил название способа непосредственного замещения. Наибольшее распространение этот способ получил в так называемом табличном варианте его реализации. В этом случае в памяти ЭВМ хранятся таблицы k_2 -х представлений k_1 -х цифр и степеней основания k_1^i . Перевод чисел сводится к выборке из этих таблиц k_2 -х эквивалентов цифр и степеней основания, а также к выполнению сложения и умножения

по правилам k_2 -й арифметики. Этот способ удобно использовать в случае, когда $k_1 < k_2$ и при переводе чисел в такую систему, где просто выполняются операции сложения и умножения (например, из двоичной системы в десятичную). Для упрощения вычислений при этом можно воспользоваться таким выражением, полученным из (1.4):

$$X = (\dots((kx_n + x_{n-1})k + x_{n-2})k + \dots + x_1)k + x_0 + \\ + (\dots((k^{-1}x_{-m} + x_{-m+1})k^{-1} + x_{-m+2})k^{-1} + \dots + x_{-1})k^{-1}.$$

Однако при переводе чисел в системы с «непривычными» основаниями, особенно в случае $k_1 > k_2$, применение этого способа связано с довольно громоздкими вычислениями. Поэтому во многих случаях удобнее пользоваться отдельными способами перевода целых чисел и правильных дробей.

Способ перевода целых чисел из системы с основанием k_1 в систему с основанием k_2 , ($k_1 > k_2$), заключается в следующем. Число $X_{(k_1)}$ делят на k_2 по правилам деления с основанием k_1 до получения остатка. Если частное от деления не нуль, то частное становится делимым и процесс деления на k_2 продолжают. Как только очередное частное станет равным нулю, процесс деления на k_2 прекращают. Остаток, полученный при первом делении на k_2 , представляет цифру разряда результата с весом k_2^0 , остаток от второго деления представляет цифру результата с весом k_2^1 и т. д. Последний остаток является цифрой результата, имеющей вес k_2^n .

Пример 4. Перевести число $X_{(10)} = 1247$ в пятиричную и двоичную системы счисления.

[illegible]

В случае, когда $k_1 < k_2$, выполняют умножение цифры с весом k_1^n числа $X_{(k_1)}$ (старшей цифры числа $X_{(k_1)}$) на основание k_1 , после чего к произведению прибавляют следующую (в порядке убывания весов) цифру числа $X_{(k_1)}$. Результат предыдущей операции умножают на k_1 и прибавляют очередную цифру числа $X_{(k_1)}$. Этот процесс заканчивают, когда будет прибавлена цифра с весом k_1^0 (младшая цифра). Все вычисления при этом выполняются в k_2 -й системе счисления.

Пример 5. Перевести из двоичной системы в троичную, а затем в десятичную число $X_{(2)} = 10110110$:

$$\begin{aligned} 1 \cdot 2 + 0 &= 2; & 2 \cdot 2 + 1 &= 12; & 12 \cdot 2 + 1 &= 102; \\ 102 \cdot 2 + 0 &= 211; & 211 \cdot 2 + 1 &= 1200; & 1200 \cdot 2 + 1 &= 10101; \\ & & 10101 \cdot 2 + 0 &= 20202; \\ & & X_3 &= 20202; \\ 2 \cdot 3 + 0 &= 6; & 6 \cdot 3 + 2 &= 20; & 20 \cdot 3 + 0 &= 60; & 60 \cdot 3 + 2 &= 182; \\ & & X_{(10)} &= 182. \end{aligned}$$

Перевод правильных дробей из системы счисления с основанием k_1 в систему с основанием k_2 ($k_1 > k_2$) осуществляется так. Дробь, соответствующая числу $X_{(k_1)}$, умножается на k_2 по правилам умножения в системе с основанием k_1 . В полученном произведении отделяется целая часть, которая может быть равной нулю, а дробная часть снова умножается на k_2 с последующим отделением целой части. Эти операции повторяют либо до получения нулевой дробной части произведения, либо до получения необходимого количества разрядов числа $X_{(k_2)}$ в новой системе счисления. Старшая цифра результата перевода (т. е. первая после запятой) совпадает с первой отделенной целой частью, вторая цифра результата — со второй отделенной целой частью и т. д.

Пример 6. Требуется записать число $X_{(10)} = 0,314$ в восьмеричной и двоичной системах, ограничившись в обоих случаях шестью разрядами после запятой.

$$\begin{aligned} 0,314 \cdot 8 &= 2,512; & 0,512 \cdot 8 &= 4,096; & 0,096 \cdot 8 &= 0,768; \\ 0,768 \cdot 8 &= 6,144; & 0,144 \cdot 8 &= 1,152; & 0,152 \cdot 8 &= 1,216; \\ & & X_{(8)} &= 0,240611; \\ 0,314 \cdot 2 &= 0,628; & 0,628 \cdot 2 &= 1,256; & 0,256 \cdot 2 &= 0,512; \\ 0,512 \cdot 2 &= 1,024; & 0,024 \cdot 2 &= 0,048; & 0,048 \cdot 2 &= 0,096; \\ & & X_{(2)} &= 0,010100. \end{aligned}$$

При $k_1 < k_2$ для перевода правильной дроби, имеющей m цифр после запятой, необходимо разделить цифру младшего разряда числа X_{k_1} на k_1 и сложить со следующей цифрой этого числа. Такую операцию необходимо повторить еще $m - 1$ раз, используя на каждом шаге в качестве делимого сумму, полученную на предыдущем шаге. Все операции выполняются в k_2 -й системе.

Пример 7. Перевести в десятичную систему число $X_{(2)} = 0,101101$.

$$\begin{aligned} \frac{1}{2} + 0 &= 0,5; & \frac{0,5}{2} + 1 &= 1,25; & \frac{1,25}{2} + 1 &= 1,625; \\ \frac{1,625}{2} + 0 &= 0,8125; & \frac{0,8125}{2} + 1 &= 1,40625; \\ \frac{1,40625}{2} + 0 &= 0,703125; \\ & & X_{(10)} &= 0,703125. \end{aligned}$$

В том случае, когда основания позиционных однородных систем счисления связаны соотношением $k_1 = k_2^g$, где $g > 0$, перевод чисел выполняется очень просто. Если g — целое, перевод сводится к замене каждой k_1 -й цифры ее g -разрядным k_2 -м представлением. При дробном g исходное число разбивают на $\frac{1}{g}$ разрядные группы (начиная с младших разрядов) и каждую группу заменяют ее k_2 -м представлением.

Пример 8. Перевести из четверичной системы в двоичную, а затем в шестнадцатеричную число $X_{(4)} = 23013311$. В шестнадцатеричной системе количественным эквивалентам 10, 11, ..., 15 соответствуют цифры A, B, ..., F.

Согласно вышеизложенному

$$X_{(2)} = 1011000111110101;$$

$$X_{(16)} = B \ 1 \ F \ 5.$$

Для перевода числа X из канонической k_1 -й системы счисления в квазиканоническую систему с онованием k_2 вначале необходимо представить X в канонической k_2 -й системе. Затем те цифры канонической k_2 -й системы, которых нет в квазиканонической, заменяют комбинациями цифр квазиканонической системы. После этого по правилам k_2 -й системы суммируют все полученные комбинации цифр с учетом их весов.

Пример 9. Перевести число $X_{(10)} = 765289$ в квазиканоническую десятичную систему счисления.

В данном случае $l = 5$ и допустимыми цифрами квазиканонической системы будут $\{-5, -4, \dots, -1, 0, 1, \dots, 5, 6\}$. Поэтому цифры 7, 8, 9 канонической системы следует представить посредством двух цифр квазиканонической системы, а именно: $7 = 1\bar{3}$, $8 = 1\bar{2}$, $9 = 1\bar{1}$, где $\bar{3} = -3$; $\bar{2} = -2$; $\bar{1} = -1$. Просуммируем теперь преобразованные цифры и непреобразованные разряды исходного числа с учетом их весов

$$\begin{array}{r} + \quad 1\bar{3} \\ + \quad \quad 6 \\ + \quad \quad \quad 5 \\ + \quad \quad \quad \quad 2 \\ + \quad \quad \quad \quad \quad 1\bar{2} \\ + \quad \quad \quad \quad \quad \quad 1\bar{1} \\ \hline 1\bar{3}653\bar{1}\bar{1} \end{array}$$

Следовательно, в квазиканонической десятичной системе заданное число можно записать как $1\bar{3}653\bar{1}\bar{1}$. Заметим, что в виде комбинации двух цифр квазиканонической системы могут быть представлены и другие цифры канонической системы. Если, например, записать $5 = 1\bar{5}$, $6 = 14$, то исходное число в десятичной квазиканонической системе можно представить как

$$\begin{array}{r} + \quad 1\bar{3} \\ + \quad \quad 14 \\ + \quad \quad \quad 1\bar{5} \\ + \quad \quad \quad \quad 2 \\ + \quad \quad \quad \quad \quad 1\bar{2} \\ + \quad \quad \quad \quad \quad \quad 1\bar{1} \\ \hline 1\bar{2}3\bar{5}3\bar{1}\bar{1} \end{array}$$

Таким образом, перевод чисел из канонической системы в квазиканоническую может быть неоднозначным.

Для перевода чисел из квазиканонической системы в каноническую достаточно просуммировать цифры числа в квазиканонической системе с учетом их знаков и весов.

Пример 10. Перевести в десятичную каноническую систему счисления число $4562352\bar{1}$

$$X_{(10)} = -1 + 2 \cdot 10 + 5 \cdot 10^2 - 3 \cdot 10^3 - 2 \cdot 10^4 + 6 \cdot 10^5 - 5 \cdot 10^6 + 4 \cdot 10^7 = 35577519.$$

Перевод чисел из канонической системы в модифицированную квазиканоническую систему и обратный перевод выполняются аналогично.

Описанные выше способы предназначены; в основном, для ручного или программного перевода чисел из одной системы счисления в другую. Другие способы, удобные для реализации их с помощью технических средств, рассмотрены в гл. 5.

1.5. ПРЕДСТАВЛЕНИЕ ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ В ЦИФРОВЫХ ЭВМ

В цифровых ЭВМ целесообразно представлять знаки чисел с помощью тех же символов, посредством которых записывается само число в k -й системе счисления. Для этого выделяется дополнительный разряд, называемый знаковым и располагаемый обычно слева от старшего разряда числа. Будем обозначать знаковые разряды чисел X , Y , Z , ... как X_0 , Y_0 , Z_0 , ... и отделять их точками от целого числа и запятыми от дробного. В системах счисления с непосредственным представлением цифр принимают $X_0 = 0$, если знак числа «+» и $X_0 = k - 1$, если число отрицательное. Это позволяет представлять знак числа посредством тех же символов, которые используются для представления цифр. В двоичной системе счисления $k - 1 = 1$, т. е. здесь $X_0 = 1$ соответствует знаку «-». В двоично-кодированных системах знаки чисел кодируют так же, как и в двоичной системе.

Для правильного выполнения арифметических операций над числами, у которых знаки закодированы цифрами, используют специальные способы представления чисел: прямой, обратный и дополнительный коды.

Представление числа X в прямом коде определяется выражением

$$X_{\text{пр}} = X_0 | X|, \quad (1.9)$$

т. е. прямые коды чисел $+X$ и $-X$ отличаются только знаковыми разрядами.

Из (1.9) видно, что нуль в прямом коде может быть представлен двояко: «положительный» нуль $+0 = 0_{\text{пр}} = 0.00...0,00...0$ и «отрицательный» нуль $-0 = 0_{\text{пр}} = 1.00...0,00...0$.

Используется прямой код в цифровых ЭВМ для ввода и вывода информации, а также для представления чисел в ЗУ. Сложение чисел с одинаковыми знаками в прямом коде выполняется в соответствии с приведенными выше правилами и не вызывает затруднений. Наличие цифры переноса $r_{\text{слож}} = 1$ при сложении старших разрядов $n + m + 1$ -разрядных чисел X и Y , представленных в форме (1.4), свидетельствует о том, что

$$Z = X + Y > X_{\text{max}} + k^{-m} = k^{n+1},$$

т. е. для представления результата операции требуется больше, чем $n + m + 1$ разрядов. В этом случае говорят, что происходит переполнение разрядной сетки. Однако складывать числа с разными знаками, т. е. выполнять алгебраическое сложение в прямых кодах не-

удобно, так как в составе цифровых ЭВМ в этом случае должно быть специальное оборудование для вычитания чисел и определения знака разности.

Операцию алгебраического сложения чисел можно свести к операции сложения при использовании обратных и дополнительных кодов. Способ представления числа $X = \pm \sum_{i=-m}^n k^i x_i$ в обратном коде определяется выражением

$$X_{\text{обр}} = \begin{cases} X_0 |X| & \text{при } X = 0; \\ X_0 k^{n+1} - k^{-m} - |X| & \text{при } X_0 = 1. \end{cases} \quad (1.10)$$

Сравнивая (1.9) и (1.10), легко заметить, что обратный код положительного числа совпадает с его прямым кодом. Так как при $k = 2$

$$2^{n+1} - 2^{-m} = \underbrace{11 \dots 11}_{n+1}, \underbrace{11 \dots 11}_m,$$

то для получения обратного кода отрицательного числа в двоичной системе счисления необходимо в знаковом разряде записать 1, а в остальных разрядах единицы заменить нулями и нули единицами. Аналогичную замену необходимо выполнить и при преобразовании обратного кода отрицательного числа в прямой код. В отличие от прямого кода, в обратном коде нельзя отбрасывать нули после знакового разряда в целой части и нули в конце дробной части отрицательных чисел, но можно отбрасывать записанные на этих местах единицы при $k = 2$. Нуль в обратном коде, так же, как и в прямом коде, имеет два изображения.

Рассмотрим выполнение операции алгебраического сложения с использованием обратных кодов. Если числа X и Y положительные, то их сложение не отличается от сложения в прямом коде. Когда же одно из чисел (например, Y) отрицательное и представлено обратным кодом, то можно записать, что

$$Z = X - Y = X_{\text{обр}} + Y_{\text{обр}} = |X| + (k^{n+1} - k^{-m} - |Y|) = (|X| - |Y|) + (k^{n+1} - k^{-m}). \quad (1.11)$$

Дальнейшее выполнение операции зависит от знака разности

$$\Delta = |X| - |Y|.$$

При $\Delta < 0$ выражение (1.11) можно переписать так:

$$Z = k^{n+1} - k^{-m} - \Delta < X_{\text{max}}; \quad Z_0 = 0.$$

В этом случае перенос при сложении старших разрядов равен нулю, а сам результат будет представлен обратным кодом. Для образования кода знака результата достаточно сложить коды знаков чисел.

При $\Delta > 0$ получим

$$Z = k^{n+1} - k^{-m} + \Delta > X_{\text{max}}; \quad Z_0 = 0.$$

Следовательно, возникает единичный перенос при сложении старших разрядов, имеющий вес k^{n+1} , а в $n + m + 1$ числовых разрядах бу-

дет записано число $\Delta - k^{-m}$. Для формирования знака результата в данном случае необходимо сложить знаковые разряды чисел с учетом переноса с весом k^{n+1} . Если образующейся при этом единице переноса приписать вес k^{-m} и сложить ее с числом $\Delta - k^{-m}$, то получим правильную сумму чисел X и Y . Перенос, возникающий при сложении знаковых разрядов и прибавляемый к младшему разряду суммы чисел, называют циклическим переносом.

Если числа X и Y отрицательные, то

$$\begin{aligned} Z &= -X - Y = X_{\text{обр}} + Y_{\text{обр}} = k^{n+1} - k^{-m} - |X| + k^{n+1} - k^{-m} - |Y| = \\ &= k^{n+1} + k^{n+1} + k^{-m} - (|X| + |Y|) - k^{-m} > X_{\text{max}}. \end{aligned}$$

Первое число k^{n+1} в полученном выражении соответствует переносу из старшего разряда. Поэтому в числовых разрядах будет находиться число $Z = k^{n+1} - k^{-m} - (|X| + |Y|) - k^{-m}$, которое после прибавления единицы циклического переноса, образующейся при сложении знаковых разрядов, будет представлять собой сумму чисел X и Y , записанную в обратном коде. Из изложенного следует, что при использовании обратных кодов для алгебраического сложения положительная сумма всегда представлена прямым кодом, а отрицательная сумма — обратным кодом.

Существенным недостатком алгебраического сложения с использованием обратных кодов является наличие циклического переноса, для учета которого в цифровых ЭВМ необходимы дополнительные затраты времени (см. гл. 5). От этого недостатка свободен способ выполнения алгебраического сложения в дополнительных кодах.

Дополнительный код числа X образуют в соответствии со следующим правилом:

$$X_{\text{доп}} = \begin{cases} X_0 |X| & \text{при } X_0 = 0; \\ X_0 k^{n+1} - |X| & \text{при } X_0 = 1. \end{cases} \quad (1.12)$$

Представление положительного числа в дополнительном коде совпадает с его прямым кодом. Из сравнения вторых строк выражений (1.10) и (1.12) следует, что дополнительный код отрицательного числа отличается от обратного кода того же числа на величину k^{-m} . На этом основано правило формирования дополнительного кода отрицательного числа — достаточно получить обратный код числа и прибавить 1 в его младший разряд. При $k = 2$ тот же результат может быть получен, если в прямом коде отрицательного числа оставить без изменения крайние справа нули и первую единицу за ними, а все цифры левее этой единицы (кроме знакового разряда) поменять на обратные. Преобразование же дополнительного кода отрицательного числа в прямой код может быть осуществлено либо обратным путем (т. е. вычитанием 1 в младшем разряде и последующим преобразованием обратного кода), либо путем формирования дополнительного кода от дополнительного. Легко убедиться, что оба пути приводят к одинаковому результату. Нуль, в отличие от прямого и обратного кодов, в дополнительном коде имеет единственное представление

$$0_{\text{доп}} = 0.00 \dots 0.00 \dots 0.$$

Сложение положительных чисел X и Y в дополнительном коде выполняется так же, как и в прямом и обратном кодах. Если число Y отрицательное, то

$$Z = X - Y = X_{\text{доп}} + Y_{\text{доп}} = |X| + k^{n+1} - |Y| = \\ = k^{n+1} + (|X| - |Y|) = k^{n+1} + \Delta.$$

При $\Delta > 0$ число k^{n+1} соответствует переносу при сложении старших разрядов. Поэтому для получения правильного результата в прямом коде этот перенос необходимо учесть при сложении знаковых разрядов. При $\Delta < 0$ перенос из старших разрядов в знаковые отсутствует, а отрицательный результат будет представлен дополнительным кодом.

Сложение отрицательных чисел X и Y в дополнительном коде дает следующий результат:

$$Z = -X - Y = k^{n+1} - |X| + k^{n+1} - |Y| = k^{n+1} + k^{n+1} - \\ - (|X| + |Y|) > X_{\text{max}}.$$

Следовательно, при сложении старших разрядов будет возникать перенос, о чем здесь и свидетельствует первое число k^{n+1} . Для формирования правильного знака результата этот перенос необходимо прибавить к сумме знаковых разрядов. Перенос же из разряда знаков не учитывается. Поэтому отрицательный результат в этом случае будет представлен дополнительным кодом.

Таким образом, использование обратного или дополнительного кодов позволяет упростить операции сложения и вычитания чисел с разными знаками. Из сравнения двух способов представления отрицательных чисел в ЭВМ следует, что дополнительные коды формируются сложнее, чем обратные. Однако алгебраическое сложение проще выполнять в дополнительном коде, так как при этом не возникает циклический перенос.

Как указывалось выше, при выполнении операции сложения чисел с одинаковыми знаками может произойти переполнение разрядной сетки, когда для представления результата требуется большее количество разрядов, чем для представления чисел. Поскольку знаковые разряды складываются с учетом переноса из старшего разряда, то это может привести к неправильному результату. Независимо от кода для представления отрицательных чисел (обратный или дополнительный) переполнение разрядной сетки происходит всегда, если только при сложении положительных чисел возникает перенос в знаковый разряд, а при сложении отрицательных чисел такой перенос отсутствует.

Уменьшить вероятность переполнения разрядной сетки можно путем введения масштабных коэффициентов, на которые умножаются все исходные данные перед решением задачи. Однако для полного исключения переполнения необходимо оценивать возможные значения результата после каждой операции, что практически невыполнимо ввиду очень большого числа выполняемых операций. Поэтому в ЭВМ принимаются специальные меры для обнаружения переполнения разрядной сетки. Наиболее широко для этой цели используют два способа.

При обнаружении переполнения по первому способу учитывается то обстоятельство, что переполнение может появиться только при сложении чисел с одинаковыми знаками. В этом случае о переполнении будет свидетельствовать несовпадение знака слагаемых и знака суммы. Недостатком этого способа является необходимость реализации двукратного сравнения знаков чисел и их запоминания до конца операции.

Сущность второго способа состоит в использовании так называемых модифицированных прямого, обратного и дополнительного кодов. Особенностью этих кодов является наличие двух знаковых разрядов, а также кодирование знака числа двумя одинаковыми цифрами. Появление в знаковых разрядах модифицированного кода, обрабатываемых так же, как и числовые разряды, различных цифр (например, 01 при сложении положительных чисел и 10 при сложении отрицательных чисел для $k = 2$) свидетельствует о переполнении разрядной сетки. Циклический перенос в случае модифицированного обратного кода возникает при сложении вторых (крайних слева) знаковых разрядов. Этот способ обнаружения переполнения получил наибольшее распространение в цифровых ЭВМ.

При использовании квазиканонических систем счисления знак числа равен знаку цифры в разряде с наибольшим весом.

1.6. ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В ЦИФРОВЫХ ЭВМ

В настоящее время в ЦВТ преимущественно используются две формы представления чисел. Первая из них получила название представления чисел с фиксированной запятой (естественной формы), а вторая — представления чисел с плавающей запятой (нормальной или полулогарифмической формы).

Любое число X в позиционной системе счисления с основанием k можно записать как

$$X = k^P M.$$

Обычно M называют мантиссой числа X , а P — порядком того же числа. Если в ЭВМ каждому числу X однозначно соответствует хранимая в ЗУ мантисса M , а порядок фиксирован для всех чисел, то говорят, что число представлено в форме с фиксированной запятой. Порядок P в этом случае устанавливается при подготовке задачи к решению, причем число $\frac{1}{k^P}$ в сущности является масштабным коэф-

фициентом, на который должны быть умножены исходные данные для исключения переполнения разрядной сетки. Если же каждому числу X однозначно соответствует пара P и M , хранимых в ЗУ, то такое представление называют формой с плавающей запятой (следует иметь в виду, что положение запятой в разрядной сетке ЭВМ физически никак не обозначается) или полулогарифмической формой.

При представлении чисел в форме с фиксированной запятой положение запятой сохраняется неизменным для всех чисел, с которыми оперирует машина. С целью упрощения вычисления масштабных

коэффициентов запятую обычно фиксируют перед старшим разрядом мантиссы или после младшего.

В первом случае ЭВМ оперирует с числами, которые меньше единицы. Если число разрядов для записи мантиссы составляет n , то минимальное по абсолютной величине число, которое может быть представлено в машине, равно k^{-n} , а максимальное — $1 - k^{-n}$.

Следует отметить, что при таком представлении чисел удобно выполнять операцию умножения, так как при этом не происходит переполнения разрядной сетки.

Во втором случае в машине выполняются операции над целыми числами, абсолютные величины которых находятся в пределах от 0 до $k^n - 1$. Иногда запятую сдвигают за пределы разрядной сетки вправо на l разрядов, т. е. вводят l условных разрядов. При этом абсолютная величина максимального числа в машине составляет $k^{n+l} - k^l$, а минимального — k^l .

Разрядная сетка для представления чисел (формат данных) в ЭВМ с фиксированной запятой состоит из двух частей: один разряд для представления знака, остальные разряды — для представления мантиссы M .

При представлении чисел в форме с плавающей запятой порядок P может быть положительным или отрицательным целым числом. Мантисса же в большинстве случаев является положительной или отрицательной правильной дробью, причем $k^{-1} \leq M < 1$. Если в ЭВМ для записи порядка используется m разрядов, а для записи мантиссы n разрядов, то в этой машине может быть представлено следующее максимальное по абсолютной величине число

$$k^{k^m-1} - (1 - k^{-n}) = k^{k^m-1} - k^{k^m-n-1}.$$

В свою очередь, минимальное по абсолютной величине отличающееся от нуля число в такой машине

$$k^{-k^m+1}k^{-1} = k^{-k^m}.$$

Кроме указанных выше разрядов для представления порядка и мантиссы, в разрядной сетке ЭВМ с плавающей запятой есть также два разряда для представления знаков порядка и мантиссы.

В ЭВМ, где используется представление чисел в форме с плавающей запятой, арифметические операции выполняются как над мантиссами чисел, так и над их порядками. Часто также необходимо выполнять операцию нормализации чисел, сущность которой состоит в выполнении условия $k^{-1} \leq M < 1$ (см. гл. 5). Поэтому АЛУ машин с плавающей запятой является более сложным и менее быстродействующим, чем АЛУ машин с фиксированной запятой. По этим причинам совокупность средств ЦВТ, в которых используется представление чисел в форме с фиксированной запятой, составляют, в основном, специализированные машины для управления технологическими процессами, обработки статистической информации и результатов измерений, решения задач с большим числом логических операций и т. д. Представление же чисел в форме с плавающей запятой используется в универсальных ЭВМ средней и большой производительности для решения широкого круга инженерных и научных задач.

Так как каждой из двух рассмотренных форм представления чисел присущи свои положительные качества и недостатки, то в универсальных больших цифровых ЭВМ 2-го поколения, а также почти во всех современных универсальных ЭВМ используют обе формы. Например, при работе в режиме с фиксированной запятой ЭВМ Единой системы (ЕС) запятая располагается справа от младшего разряда. Основной формат данных машины показан на рис. 1.3 и имеет 32 двоичных разряда. На этом же рисунке указан вес каждого разряда. Отрицательные числа представляются здесь дополнительными кодами. Для повышения производительности и для лучшего использования 3У в ЭВМ ЕС могут быть заданы числа длиной в 16 разрядов. Кроме того, для вычислений с повышенной точностью (например, при умножении и делении) предусмотрена возможность использования 64-разрядных двоичных чисел. В режиме с фиксированной запятой эти машины могут производить не только целочисленные вычисления, но и выполнять операции над командами.

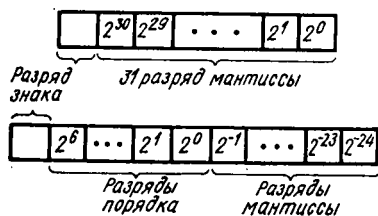


Рис. 1.3.

Числа с плавающей запятой в тех же ЭВМ могут быть двух форматов: короткого и длинного. Короткий формат (рис. 1.3) имеет 32 разряда, из которых 7 разрядов предназначены для представления порядка, 24 —

для представления мантииссы и 1 разряд — для знака мантииссы. Специального разряда для знака порядка в ЭВМ ЕС нет. Порядки в машине отсчитываются относительно числа 64, т. е. вводится «смещенный порядок» $P_{см} = 64 + P$. Таким образом, отрицательные порядки P будут представлены «смещенным порядком» $P_{см}$, меньшими 64, а положительные порядки — «смещенными порядками», большими 64. Введение «смещенного порядка» позволяет свести операции над порядками к арифметическим действиям над целыми положительными числами. Следовательно, максимальный положительный порядок числа в машинах ЕС равен $1111111_{(2)} - 1000000_{(2)} = 0111111_{(2)} = 63_{(10)}$, а максимальный отрицательный порядок $0000000_{(2)} - 1000000_{(2)} = -64_{(10)}$. Мантииссы отрицательных чисел при этом представляются прямым кодом. Длинный формат, используемый для вычислений с повышенной точностью, имеет 64 разряда. Для порядка отводится столько же разрядов, что и в коротком формате, а для мантииссы — 56 разрядов. Запятая находится слева от старшего разряда мантииссы.

Кроме рассмотренных выше наиболее распространенных форм представления чисел в специализированных цифровых ЭВМ могут использоваться формы, получаемые путем определенного функционального преобразования чисел. Одним из примеров такой формы является логарифмическая форма, когда числа представляются их логарифмами по некоторому основанию. Это дает возможность заменить операции умножения и деления чисел операциями сложения и вычитания их логарифмов. Известны также представления чисел в ЭВМ в форме с трансформирующейся запятой, когда число X

подвергается преобразованию вида $\frac{hX}{1+X}$, $0 < h < 1$, в форме цепных (непрерывных) дробей, разложений в некоторые ряды и др.

1.7. ДВОИЧНО-КОДИРОВАННЫЕ И ДРУГИЕ СИСТЕМЫ СЧИСЛЕНИЯ

Одной из важнейших характеристик цифровых ЭВМ является количество оборудования, необходимого для представления чисел. Если числа, с которыми оперирует машина, не превосходят N , то для их представления в канонической системе счисления с основанием k требуется $\log_k N$ условных единиц оборудования (k -х разрядов). Предполагая, что аппаратные затраты на один k -й разряд пропорциональны k , количество оборудования для представления чисел составит $k \log_k N$. Эффективность различных (по основанию k) позиционных систем счисления в этом случае удобно оценивать функцией

$$f_1(k) = \frac{k \log_k N}{2 \log_2 N} = \frac{k \ln N \ln 2}{2 \ln N \ln k} = \frac{k \ln 2}{2 \ln k},$$

показывающей, во сколько раз количество оборудования для представления чисел в k -й системе счисления больше аппаратных затрат в двоичной системе. Нетрудно убедиться, что функция $f_1(k)$ имеет минимум, если $k = e$. Поэтому при указанном выше предположении наиболее экономичной будет система с основанием $k = |e| = 3$. Если аппаратные затраты на один k -й разряд не зависят от k , то эффективность различных систем счисления оценивается функцией

$$f_2(k) = \frac{\log_k N}{\log_2 N} = \frac{\ln N \ln 2}{\ln k \ln N} = \frac{\ln 2}{\ln k}.$$

Из выражения для $f_2(k)$ следует, что с ростом k количество оборудования для представления чисел уменьшается.

Построение наиболее удобных для человека десятичных универсальных ЭВМ, в которых цифры отображались бы сигналами, квантованными по десяти уровням, является практически целесообразным только при наличии технических средств (схем) для представления чисел, основные технико-экономические характеристики которых (стоимость, надежность, быстродействие и др.) были бы не хуже соответствующих характеристик схем для представления двоичных цифр. Однако, как правило, некоторые основные характеристики известных схем для такого представления десятичных цифр уступают характеристикам двоичных схем. Поэтому при построении десятичных ЭВМ используют кодирование двоичными цифрами десятичных цифр.

Для двоично-кодированных систем счисления сравнительная оценка оборудования для представления чисел может быть записана как

$$f_3(k) = \frac{\log_k N \log_2 k}{\log_2 N} = \frac{\log_2 N \log_2 k}{\log_2 k \log_2 N} = \frac{\log_2 k}{\log_2 k}.$$

Функция $f_3(k)$ равна единице, если только k равно целой степени числа 2. В таком случае для представления чисел в двоичной и k -й

системах требуется одинаковое количество оборудования. При $k = 10$

$$f_3(10) = \frac{\lceil \log_2 10 \rceil}{\log_2 10} = \frac{4}{3,322} \approx 1,204,$$

т. е. в двоично-десятичной системе для представления чисел требуется примерно в 1,2 раза больше оборудования по сравнению с двоичной системой.

Способ кодирования цифр k -й системы счисления двоичными цифрами должен удовлетворять условию единственности, в соответствии с которым между каждой k -й цифрой и комбинацией двоичных цифр должно быть установлено взаимно однозначное соответствие. Для этого число двоичных разрядов для представления k -й цифры должно быть не меньше $h = \lceil \log_2 k \rceil$. Очевидно, что любой k -й цифре можно поставить в соответствие любое из 2^h двоичных чисел, записываемых с помощью h разрядов. Следовательно, число способов двоичного кодирования k цифр равно числу размещений из 2^h по k , т. е.

$$2^h (2^h - 1) (2^h - 2) \dots (2^h - k + 1) = \frac{2^h!}{(2^h - k)!}.$$

Это число очень быстро растет с увеличением k , что делает невозможным анализ ручными методами всех способов двоичного кодирования даже для практически наиболее важного случая $k = 10$. Поэтому к настоящему времени исследовано всего несколько способов двоичного кодирования k -х цифр.

Наиболее широкое распространение в ЦВТ получили двоично-десятичные системы счисления, в которых десятичные цифры записываются как четырехразрядные двоичные числа — двоичные тетрады. Иногда используют также представление десятичных цифр с помощью пяти, шести и семи разрядов. В принципе, можно построить ЭВМ, работающую при любом двоично-десятичном кодировании. Однако, как показывают теоретические исследования и опыт разработки десятичных ЭВМ, наилучшие результаты могут быть получены при использовании двоично-десятичных кодов (ДДК), удовлетворяющих сформулированному выше условию единственности, а также обладающих свойством упорядоченности, четности, дополненности и взвешенности (перечисленные условия имеют общее название условий Рутисхаузера).

Упорядоченность ДДК состоит в выполнении одного из условий

$$\begin{aligned} 0_{(2)} < 1_{(2)} < \dots < 9_{(2)}; \\ 0_{(2)} > 1_{(2)} > \dots > 9_{(2)} \end{aligned}$$

для двоичных кодов $0_{(2)}, 1_{(2)}, \dots, 9_{(2)}$ десятичных цифр 0, 1, ..., 9. Наличие упорядоченности ДДК необходимо для реализации логических операций.

Свойство четности ДДК должно проявляться в том, чтобы всем четным десятичным цифрам соответствовали либо только четные, либо только нечетные двоичные коды. Аналогично, всем нечетным десятичным цифрам должны соответствовать либо только нечетные, либо

только четные двоичные коды. Это свойство упрощает выполнение операций умножения, деления и округления.

Сущность свойства дополнительности ДДК заключается в том, что сумма двоичного кода любой десятичной цифры и ее инверсного двоичного кода (получаемого заменой 0 на 1 и наоборот) должна быть равна двоичному коду цифры 9. Это необходимо для обнаружения переноса в старший разряд, а также для формирования обратного и дополнительного кодов при десятичном сложении (см. гл. 5).

ДДК называют взвешенным, если каждому из h разрядов двоичного представления $x_h x_{h-1} \dots x_1$ десятичной цифры X поставлены в соответствие веса $a_h, a_{h-1}, \dots, a_1$, причем

$$a_h x_h + a_{h-1} x_{h-1} + \dots + a_1 x_1 = X. \quad (1.13)$$

Использование взвешенных ДДК облегчает перевод чисел из одной системы счисления в другую.

Простейшим ДДК является так называемый код «8, 4, 2, 1» (код прямого замещения), получаемый путем записи десятичных цифр в двоичной позиционной системе счисления с естественным порядком весов (табл. 1). Перевод чисел из десятичной системы в двоично-десятичную в этом случае сводится к замене десятичных цифр двоичными тетрадами в соответствии с табл. 1. Обратный переход также осуществляется очень просто: необходимо число в двоично-десятичной системе разбить на тетрады и каждую из них заменить ее десятичным эквивалентом. Код «8, 4, 2, 1» использован в ЭВМ ЕС. Следует отметить, что правильная запись числа в ДДК не должна содержать тетрад, превышающих 9.

Таблица 1

ДДК	Десятичные цифры									
	0	1	2	3	4	5	6	7	8	9
«8, 4, 2, 1»	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
«С избыт- ком 3»	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
«2, 4, 2, 1»	0000	0001	0010	0011	0100	0101	0110	0111	1110	1111
«3, 3, 2, 1»	0000	0001	0010	1000	1001	1010	1011	1100	1101	1110
				0100	1001	1010	1011	1100	1101	
				1000	0011	0100	0101	0110	0111	
«7, 4, 2, 1»	0000	0001	0010	0011	0100	0101	0110	0111	1001	1010
«8, 4, —2, —1»	0000	0111	0110	0101	0100	1011	1010	1001	1000	1111
								1000		
«Рефлекс- ный»	0000	0001	0011	0010	0110	0111	0101	0100	1100	1000
«3а + 2»	00010	00101	01000	01011	01110	10001	10100	10111	11010	11101
«2 из 5»	11000	00011	00101	00110	01001	01010	01100	10001	10010	10100

Код «8, 4, 2, 1» обладает всеми перечисленными выше свойствами, кроме свойства дополнительности, что очень затрудняет формирование переносов из младшей тетрады (т. е. из младшего десятичного раз-

ряда) в старшую. Примером ДДК, обладающего свойством дополнительности, является код «с избытком 3» (табл. 1). Для записи десятичных чисел в коде «с избытком 3» необходимо каждую двоичную тетраду этого числа в коде «8, 4, 2, 1» сложить с двоичным кодом цифры 3. Преобразование чисел, заданных в ДДК «с избытком 3» в десятичную систему выполняется также путем замены тетрад их десятичными аналогами. Недостаток кода «с избытком 3» — отсутствие свойства взвешенности.

Заметим однако, что код «8, 4, 2, 1» и коды с избытком (построенные по аналогии с ДДК «с избытком 3») обладают еще одним важным свойством, называемым аддитивностью. Свойство аддитивности заключается в том, что ДДК суммы любых десятичных цифр равен двоичной сумме ДДК этих цифр или отличается от нее на некоторую константу. Это свойство позволяет выполнять многие арифметические операции с ДДК по правилам двоичной арифметики.

Коду «2, 4, 2, 1» присущи все указанные ранее свойства, за исключением аддитивности и единственности, так как старший разряд тетрады этого ДДК имеет искусственный вес 2. Преобразование чисел из одной системы счисления в другую осуществляется в соответствии с табл. 1. Особенностью взвешенных ДДК с искусственным порядком весов является отсутствие взаимно однозначного соответствия между десятичными цифрами и их двоичными кодами. Любой двоичный код здесь соответствует одной десятичной цифре, но десятичная цифра может быть представлена несколькими двоичными кодами (см. табл. 1). Причиной тому служит множественность решений уравнения (1.13) относительно переменных $x_n, x_{n-1}, \dots, x_1$. Для выполнения же условия единственности кодирования уславливаются считать «запрещенными» некоторые, хотя и правильные, двоичные коды десятичных цифр.

Примерами ДДК с искусственным порядком весов являются также коды «3,3,2,1» и «7,4,2,1» (табл. 1). В двоичных тетрадах кода «7,4,2,1», как нетрудно заметить, число единиц не превышает двух. Это обуславливает высокую экономичность цифровых схем, работающих в таком коде, если энергопотребляющему состоянию схемы соответствует единица.

В рефлексном коде (табл. 1) двум соседним десятичным цифрам соответствуют двоичные тетрады, отличающиеся только в одном разряде. Этот код называют также кодом Грея и используют чаще всего в устройствах, преобразующих линейные или угловые механические перемещения в цифровую форму.

Условие единственности кодирования десятичных цифр может быть выполнено введением отрицательных весов (например, в ДДК «8, 4, —2, —1»). Из ДДК, в которых каждая десятичная цифра записывается пятью двоичными разрядами, укажем коды «3a + 2» и «2 из 5» (табл. 1). Первый код получают, записывая в двоичной системе с естественным порядком весов число $3a + 2$, где a — заданная десятичная цифра. Такой код обладает свойством дополнительности. В коде «2 из 5» каждая десятичная цифра изображается пятью двоичными разрядами, из которых только два содержат единицы. Можно считать, что код «2 из 5» получается из кода «7, 4, 2, 1» путем добав-

ления справа разряда с весом 0. В этот разряд записывают такую двоичную цифру, чтобы общее число единиц было равно двум за исключением кода десятичного нуля. Используются коды «3а + 2» и «2 из 5» в основном для передачи информации, так как позволяют обнаруживать наиболее вероятные ошибки при передаче кодов.

В современных ЭВМ, обрабатывающих не только числовую информацию, но и текстовую (алфавитно-цифровую), отдельные символы чаще всего представляют словами длиной в один байт. Так как один байт равен двум двоичным тетрадам, то в цифровых ЭВМ с байтовым представлением информации можно записывать две десятичные цифры в один байт. Вместе с тем можно записывать в каждый байт и по одной десятичной цифре, используя остальные четыре разряда байта для записи знаков и вспомогательных символов. В ЭВМ ЕС, например, реализованы обе возможности, причем первая используется для обработки числовой десятичной информации, а вторая — для обработки текстовой информации.

Во всех рассмотренных выше двоичных и двоично-десятичных системах счисления использовались цифры 0 и 1. Однако при выполнении некоторых арифметических операций (деления, извлечения корня) удобно использовать неканоническую двоичную систему с цифрами 1 и $\bar{1}$. В этой системе вес каждого разряда является целой степенью числа 2 и может быть как положительным, так и отрицательным — в зависимости от значения 1 или $\bar{1}$ цифры в данном разряде. Отрицательные числа в этой системе могут быть записаны без использования специального разряда знака. Диапазон представления чисел при естественном порядке весов заключен между максимальным положительным числом $11\dots 1 = 2^n - 1$ и минимальным числом $\bar{1}\bar{1}\dots\bar{1} = -(2^n - 1)$. Однако в такой системе с естественным порядком весов могут быть записаны лишь нечетные числа и, вследствие этого, отсутствует изображение нуля. Использование искусственного порядка весов позволяет устранить этот недостаток, но делает неоднозначным представление чисел и сужает его диапазон. Для преобразования положительного числа из десятичной системы счисления в двоичную неканоническую систему с цифрами $\{1, \bar{1}\}$ необходимо вначале перевести это число в систему с цифрами $\{0, 1\}$. Затем, просматривая полученную запись слева направо, необходимо выделить все группы цифр вида $00\dots 01$, даже если они включают один ноль. Эти группы необходимо заменить на группы вида $1\bar{1}\dots\bar{1}\bar{1}$. Остальные цифры остаются без изменения. При преобразовании отрицательных чисел группы цифр $00\dots 01$ необходимо заменить на группы $\bar{1}\bar{1}\dots 11$, а остальные цифры заменить на $\bar{1}$.

В ЦВТ нашла ряд применений также троичная система счисления с цифрами $\{1, 0, \bar{1}\}$ и естественным порядком весов (такая система, например, использовалась в одной из первых отечественных ЭВМ «Сетунь»). Диапазон представления чисел в этой системе симметричен относительно нуля и ограничивается максимальным по абсолютной величине числом $\frac{1}{2}(3^n - 1)$. Для перевода целого положительного

десятичного числа в рассматриваемую систему необходимо вначале перевести его в троичную систему с цифрами $\{0,1,2\}$. Затем, начиная с младших разрядов, следует заменить цифры 2 на $\bar{1}$, прибавляя при этом 1 к соседнему старшему разряду. При переводе отрицательных целых десятичных чисел описанную процедуру выполняют над абсолютной величиной числа, после чего заменяют все $\bar{1}$ на 1 и наоборот.

Для преобразования смешанных чисел необходимо, чтобы абсолютная величина их дробной части D не превышала $\frac{1}{2}$. Если же это условие не выполнено, то следует прибавить 1 к целой части числа, а вместо D подвергнуть преобразованию в систему с цифрами $\{0,1,2\}$ отрицательную правильную дробь $D - 1$. Далее, как и при преобразовании целых чисел, необходимо заменить цифры $\{0,1,2\}$ цифрами $\{1,0,\bar{1}\}$.

Особенностью арифметических операций в рассматриваемой системе, выполняемых в соответствии с правилами (1.6)—(1.8), является то, что цифра 2 результата представляется как $\bar{1}$ в данном разряде и перенос 1 в старшем разряде.

В последние годы появились специализированные средства ЦВТ, использующие коды Фибоначчи. При этом p -кодом Фибоначчи называют представление числа X в виде

$$X = \sum_{i=1}^n x_i \varphi_i(p),$$

где $x_i \in \{0,1\}$ — двоичная цифра в i -м разряде; $\varphi_i(p)$ — вес i -го разряда, определяемый как

$$\varphi_i(p) = \begin{cases} 0 & \text{при } i \leq 0; \\ 1 & \text{при } i = 1; \\ \varphi_{i-1}(p) + \varphi_{i-p-1}(p) & \text{при } i > 1. \end{cases}$$

Если $p = 0$, то код Фибоначчи совпадает с представлением чисел в двоичной канонической системе счисления. В случае $p = 1$ код Фибоначчи соответствует системе счисления с искусственным порядком весов, а именно: 1, 1, 2, 3, 5, 8, 13, 21... Легко заметить, что в этой последовательности любой вес равен сумме двух соседних младших весов. Как и в других системах с искусственным порядком весов представление чисел в коде Фибоначчи не является однозначным. К числу основных достоинств кодов Фибоначчи следует отнести повышение быстродействия устройств обработки информации и удобство организации контроля правильности их функционирования.

СХЕМОТЕХНИЧЕСКИЕ ОСНОВЫ ЦИФРОВОЙ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

2.1. ОСНОВЫ ТЕОРИИ ПЕРЕКЛЮЧАТЕЛЬНЫХ ФУНКЦИЙ

Большинство инженерных задач можно разделить на две категории: задачи анализа, сущностью которых является предсказание поведения конкретной заданной системы, и задачи синтеза, состоящие в построении системы по заданному закону ее поведения и заданным параметрам. В настоящей главе рассматриваются методы решения указанных задач на уровне логических схем ЦВТ (см. п. 1.3).

Современная цифровая ЭВМ представляет собой сложную систему, состоящую из большого числа компонент. Решение указанных задач для любой системы, в том числе и для цифровых ЭВМ, начинается с построения ее модели. Под моделью будем понимать упрощенное представление действия исследуемой системы, которое, с одной стороны, обеспечивает достаточно точное отражение важнейших свойств системы, и, с другой стороны, является достаточно простым, чтобы его можно было изучать.

В теории цифровых ЭВМ наиболее часто применяются три различных типа моделей: структурные схемы (блок-схемы), математические модели и графы. Модели типа структурных схем используются для общего (качественного) описания системы и соответствующих ей функциональных взаимосвязей между отдельными подсистемами (блоками). Широко известным примером модели такого типа является структурная схема цифровой ЭВМ (рис. 1.2), по которой легко проследить взаимодействие основных устройств машины. Наиболее общей моделью типа структурной схемы является многополюсный «черный ящик» (рис. 2.1) с n входами и m выходами. Выясним свойства этой модели применительно к аппаратным средствам ЦВТ.

Вся информация (числовая и управляющая) в цифровых ЭВМ кодируется совокупностью цифр. В свою очередь цифры отображаются электрическими сигналами, принимающими конечное число значений. Поэтому любая схема, узел, устройство или даже вся цифровая ЭВМ может быть представлена моделью — «черным ящиком», на входы которого поступают, а на выходах появляются сигналы, соответствующие буквам некоторых алфавитов. Термин «черный ящик» указывает на неопределенность внутренней структуры такой модели. Будем обозначать входные сигналы переменными x_i ($i = 1, 2, \dots, n$), а выходные y_j ($j = 1, 2, \dots, m$). В большинстве современных цифровых ЭВМ переменные x_i и y_j имеют алфавит из двух букв, которые отождествляются с цифрами 0 и 1.



Рис. 2.1

Пусть $A_i = \{a_{i1}, a_{i2}, \dots, a_{ik_i}\}$ — алфавит переменной x_i . Множество всех возможных упорядоченных наборов вида

$$(a_{1s_i}, a_{2s_i}, \dots, a_{ns_i}), \quad (2.1)$$

где $s_i \in \{1, 2, \dots, k_i\}$ называют входным алфавитом схемы. Если k_i — число букв в алфавите A_i , то число R букв во входном алфавите схемы будет равно числу различных наборов (2.1), т. е.

$R = \prod_{i=1}^n k_i$. Число Q букв в выходном алфавите будет равно

$Q = \prod_{j=1}^m l_j$, где l_j — число букв в алфавите $B_j = \{b_{j1}, b_{j2}, \dots, b_{jl_j}\}$ переменной y_j . Если $k_i = l_j = 2$, то $R = 2^n$; $Q = 2^m$. Так как число входов n и выходов m , а также число букв в алфавитах всех переменных конечно, то входной и выходной алфавиты схемы также будут конечными.

Любая реальная физическая схема вносит задержку в распространение сигналов. Поэтому входные сигналы не могут изменяться сколь угодно часто, так как в противном случае схема окажется под воздействием очередного входного сигнала и переходного процесса, вызванного предыдущим входным сигналом, что может привести к ее неправильному функционированию. В связи с этим считают, что все переменные могут изменяться только в дискретные (тактовые) моменты времени $s = rT$, где $r = 0, 1, 2, \dots$, а T — время, достаточное для окончания переходных процессов в схеме. Временной интервал между двумя соседними дискретными моментами называют тактом. Значения переменных x_i и y_j в s -м тактовом моменте, там, где это необходимо, будем обозначать $x_i^{(s)}$, $y_j^{(s)}$.

Зависимость между входными и выходными переменными схемы может быть описана посредством некоторой системы уравнений. Применительно к рассматриваемой модели система уравнений будет иметь вид

$$y_j^{(s)} = f_j(x_1^{(s)}, x_2^{(s)}, \dots, x_n^{(s)}, z_1^{(s)}, z_2^{(s)}, \dots, z_p^{(s)}) \\ (j = 1, 2, \dots, m), \quad (2.2)$$

где переменные $z_1, z_2, \dots, z_p$ соответствуют внутренним состояниям схемы.

Если переменные y_j не зависят от переменных z_r ($r = 1, 2, \dots, p$), то значения выходных переменных в s -м такте зависят только от значений входных переменных в этом же такте. Это означает, что двум одинаковым наборам входных переменных всегда соответствует один и тот же набор выходных переменных. Такое соответствие обусловлено неизменностью внутреннего состояния схемы и независимостью этого состояния от внешних воздействий. В связи с этим такие схемы получили название схем без памяти или комбинационных схем. При этом система (2.2) может быть записана в виде

$$y_j = f_j(x_1, x_2, \dots, x_n), \quad (j = 1, 2, \dots, m). \quad (2.3)$$

Функции (2.3) могут принимать только конечное число значений и зависят от аргументов, также принимающих конечное число значений. Функции такого вида получили название переключательных. Если входные и выходные переменные принимают значения из двухбуквенного алфавита, то такие функции называют также двузначными или булевыми (по фамилии английского математика Дж. Буля (1815—1864), впервые исследовавшего этот класс функций). Далее, если не будет оговорено специально, переключательными функциями или просто функциями будем называть именно двузначные функции, которые так же, как и их аргументы, принимают только два значения 0 и 1.

Если выходные переменные зависят не только от входных переменных, но и от состояния схемы в s -м такте, то для полного описания схемы необходимо указать еще одну систему уравнений

$$z_r^{(s+1)} = \varphi_r(x_1^{(s)}, x_2^{(s)}, \dots, x_n^{(s)}, z_1^{(s)}, z_2^{(s)}, \dots, z_p^{(s)}), \quad (2.4)$$

$$(r = 1, 2, \dots, p),$$

отражающую зависимость внутреннего состояния схемы в $s + 1$ -й момент времени и от ее состояния и от входных сигналов в s -й момент. Схемы, которые описываются системами уравнений (2.2) и (2.4), называют автоматами с конечной памятью или просто конечными цифровыми автоматами.

Переключательную функцию обычно задают с помощью таблицы, называемой таблицей соответствия или таблицей истинности, путем указания ее значений на всех наборах значений аргументов. Для определенности наборы аргументов нумеруют, при этом набору $(a_1, a_2, \dots, a_n)$, где $a_i \in \{0, 1\}$, приписывают номер

$$N = a_n 2^{n-1} + a_{n-1} 2^{n-2} + \dots + a_2 2^1 + a_1 2^0 = \sum_{i=1}^n a_i 2^{i-1},$$

т. е. номер набора равен двоичному числу, соответствующему данному набору. Если функция зависит от n аргументов, то число различных наборов аргументов функции равно 2^n , поскольку каждый набор имеет свой номер, а общее число номеров равно количеству различных двоичных n -разрядных чисел. Две функции считают отличающимися друг от друга, если они принимают различные значения хотя бы на одном наборе аргументов. Число различных функций от n аргументов равно 2^{2^n} , так как для задания функции $f(x_1, x_2, \dots, x_n)$ необходимо указать набор из 2^n констант $f(a_1, a_2, \dots, a_n)$, а число различных 2^n -разрядных наборов равно 2^{2^n} . В таблице истинности значения функций на некоторых наборах могут быть неопределены, т. е. могут принимать как значение 0, так и значение 1. Такие функции называют частично определенными, в отличие от полностью определенных, принимающих значения 0 и 1 на всех наборах аргументов. Наборы, на которых функция не определена, называют несущественными (фиктивными). Число переключательных функций, зависящих от одного аргумента, равно четырем (табл. 2). Из табл. 2 следует, что $f_0(x)$ и $f_3(x)$ являются константами, а $f_1(x)$ совпадает с x , т. е.

$$f_0(x) = 0, f_1(x) = x, f_3(x) = 1.$$

Единственной нетривиальной функцией будет функция $f_2(x)$, которую называют функцией отрицания или инверсии и записывают в виде $f_2(x) = \bar{x}$ (читается «не x » или « x с отрицанием»).

Рассмотрим теперь функции, зависящие от двух аргументов, число которых составит $2^{2^2} = 16$ (табл. 3). Из представленных в этой таблице 16 функций 6 функций

$$\begin{aligned} f_0(x, y) &= 0, f_3(x, y) = x, f_5(x, y) = y; \\ f_{10}(x, y) &= \bar{y}, f_{12}(x, y) = \bar{x}, f_{15}(x, y) = 1 \end{aligned}$$

являются константами или функциями одного аргумента. Остальные 10 функций зависят от двух аргументов и имеют свои общепринятые обозначения и названия, указанные в таблице. Рассмотреть таким же образом все функции, зависящие от трех аргументов, трудно, так как их число составляет $2^{2^3} = 256$.

Таблица 2

x	0	1
$f_0(x)$	0	0
$f_1(x)$	0	1
$f_2(x)$	1	0
$f_3(x)$	1	1

Суперпозицией переключательных функций называют подстановку одних функций вместо аргументов в другие функции. Так как множество значений переключательных функций совпадает со множеством значений их аргументов, то функции от большого числа аргументов могут быть представлены как суперпозиции от меньшего числа аргументов.

Система переключательных функций, с помощью которых путем суперпозиций можно представить любую сколь угодно сложную функцию, называется функционально полной (иногда такую систему называют алгеброй, однако эти понятия не тождественны, так как понятие алгебры значительно шире понятия функционально полной системы).

В теории переключательных функций важную роль играет следующая задача. Пусть задана система переключательных функций

$$f_1, f_2, \dots, f_n. \quad (2.5)$$

Какими свойствами должны обладать функции (2.5) для того, чтобы система обладала функциональной полнотой? Решение этой задачи основано на понятии замкнутого относительно суперпозиции класса функций и состоит в следующем. Число функций, зависящих от n аргументов, конечно и равно 2^{2^n} . Путем суперпозиции из функций заданной системы получают все всевозможные функции до тех пор, пока не будет получен замкнутый класс функций, когда любая суперпозиция функций из этого класса будет функцией этого же класса. Взяв в качестве исходной различные системы функций, получают различные замкнутые классы. Так как функций конечное число, то и число различных классов функций конечно. Среди них будут классы особого типа, так называемые предполные классы, обладающие следующими свойствами. Предполный класс не совпадает с классом всех 2^{2^n} функций, однако если в него включить любую не входящую ранее функцию и с помощью суперпозиции из этой функции и функций

предполного класса образовать новый класс, то он будет совпадать с классом всех 2^{2^n} функций.

• Проведенные исследования показали, что предполных классов пять: K_0 , K_1 , K_2 , K_3 и K_4 . Для того чтобы система функций (2.5) была функционально полной, необходимо, чтобы она не содержалась полностью ни в одном из пяти предполных классов. Другими словами, для полноты системы (2.5) необходимо, чтобы она содержала, по крайней мере: 1) одну функцию, не принадлежащую K_0 ; 2) одну функцию, не принадлежащую K_1 , и т.д.; 5) одну функцию, не принадлежащую K_4 . Условия 1)–5) являются не только необходимыми, но и достаточными. Перечислим теперь все предполные классы:

Класс K_0 образуют все функции, сохраняющие 0, т. е. такие, которые на наборе $(0, 0, \dots, 0)$ равны 0, $f(0, 0, \dots, 0) = 0$.

Класс K_1 образуют все функции, сохраняющие 1, т. е. такие, которые на наборе $(1, 1, \dots, 1)$ равны 1, $f(1, 1, \dots, 1) = 1$.

Класс K_2 образуют все линейные функции, т. е. такие, которые могут быть представлены линейным полиномом

$$f(x_1, x_2, \dots, x_n) = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n.$$

Класс K_3 составляют так называемые самодвойственные функции. Функции $f(x_1, x_2, \dots, x_n)$ и $\varphi(x_1, x_2, \dots, x_n)$ называются двойственными (дуальными) друг другу, если

$$f(x_1, x_2, \dots, x_n) = \overline{\varphi(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)}.$$

Самодвойственные функции двойственны сами себе

$$f(x_1, x_2, \dots, x_n) = \overline{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)}.$$

Класс K_4 состоит из всех монотонных функций. Функция $f(x_1, x_2, \dots, x_n)$ называется монотонной, если для любых двух наборов $(\alpha_1, \alpha_2, \dots, \alpha_n)$ и $(\beta_1, \beta_2, \dots, \beta_n)$ таких, что $\alpha_i \leq \beta_i$ для всех $i = 1, 2, \dots, n$ имеет место неравенство

$$f(\alpha_1, \alpha_2, \dots, \alpha_n) \leq f(\beta_1, \beta_2, \dots, \beta_n).$$

Для доказательства замкнутости любого из перечисленных классов достаточно взять две функции, принадлежащие одному и тому же классу, и показать, что их суперпозиция также принадлежит этому классу. Пусть $f(x_1, x_2, \dots, x_n)$ и $\varphi(x_1, x_2, \dots, x_n)$ — функции, сохраняющие 1. Возьмем произвольную суперпозицию функций

$$f(\varphi(x_1, x_2, \dots, x_n), x_2, \dots, x_n).$$

Но тогда

$$f(\varphi(1, 1, \dots, 1), 1, \dots, 1) = 1.$$

Таким образом, произвольная суперпозиция этих функций также является функцией, сохраняющей 1. Аналогично доказывается замкнутость остальных классов.

С учетом вышеизложенного критерии функциональной полноты системы функций можно сформулировать так. Для того чтобы система была функционально полной, необходимо и достаточно, чтобы эта

Таблица 3

Название функций	Обозначения функций	Значения функций на наборах аргументов x и y			
		00	01	10	11
Константа 0	f_0 0	0	0	0	0
Конъюнкция, логическое умножение, И	f_1 $xy, x \wedge y, x \& y$	0	0	0	1
Запрет по y	f_2 $\bar{x}y$	0	0	1	0
Переменная x	f_3 x	0	0	1	1
Запрет по x	f_4 $\bar{x}y$	0	1	0	0
Переменная y	f_5 y	0	1	0	1
Неравнозначность, сумма по модулю 2	f_6 $x \oplus y, (x + y) \bmod 2$	0	1	1	0
Дизъюнкция, логическое сложение, ИЛИ	f_7 $x \vee y$	0	1	1	1
Функция (стрелка) Пирса, отрицание дизъюнкции	f_8 $x \downarrow y, \overline{x \vee y}$	1	0	0	0
Равнозначность, эквивалентность	f_9 $x \equiv y$	1	0	0	1
Отрицание y	f_{10} $\bar{y}$	1	0	1	0
Импликация	f_{11} $y \rightarrow x, x \vee \bar{y}$	1	0	1	1
Отрицание x	f_{12} $\bar{x}$	1	1	0	0
Импликация	f_{13} $x \rightarrow y, \bar{x} \vee y$	1	1	0	1
Функция (штрих) Шеффера, отрицание конъюнкции	f_{14} $x/y, \overline{xy}$	1	1	1	0
Константа 1	f_{15} 1	1	1	1	1

система содержала хотя бы по одной из следующих функций: не сохраняющую 1, не сохраняющую нуль, нелинейную, несамодвойственную и немонотонную.

В табл.4 показано распределение функций от двух переменных по предполным классам. Знак X означает вхождение функции в класс.

Таблица 4

Функции	Классы				
	K_0	K_1	K_2	K_3	K_4
0	X	—	X	—	X
1	—	X	X	—	X
x	X	X	X	X	X
$\bar{x}$	—	—	X	X	—
xy	X	X	—	—	X
$\bar{x}\bar{y}$	X	—	—	—	—
$x \downarrow y$	—	—	—	—	—
$x \vee y$	X	X	—	—	X
$y \rightarrow x$	—	X	—	—	—
x/y	—	—	—	—	—
$x \oplus y$	X	—	X	—	—
$x \equiv y$	—	X	X	—	—

В таблицу не включены функции $y, \bar{y}, xy, x \rightarrow y$, которые с точностью до перестановки аргументов совпадают с функциями $x, \bar{x}, \bar{x}y, y \rightarrow x$ и, вместе с тем, принадлежат к одним и тем же классам. Из табл. 4 видно, что константы 0 и 1 являются несамодвойственными функциями. Кроме того, константа 0 не сохраняет 1, а константа 1 не сохраняет 0. Отсюда можно заключить, что для полноты системы, содержащей константы 0 и 1 (иначе — для ослабленной функциональной полноты), необходимо и достаточно, чтобы она содержала хотя бы одну нелинейную и одну немонотонную функции.

Из табл. 4 следует, что системы функций

- 1) $x \vee y, xy, \bar{x}$; 2) $x \oplus y, xy, \bar{x}$; 3) $x \downarrow y, xy$; 4) $x \vee y, x/y$; 5) x/y ; 6) $x \downarrow y$; 7) $x \downarrow y, x \vee y$; 8) $x/y, xy$

Таблица 5

x	y	$x \vee y$	$\overline{x \vee y}$	$\bar{x}$	$\bar{y}$	$\bar{x}\bar{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

являются функционально полными. Особенностью систем 5, 6 является то, что каждая из них состоит только из одной функции. Системы 1, 3, 4, 7 и 8 называют избыточно полными системами, так как одна из функций ($x \vee y$ или xy) может быть исключена без потери функциональной полноты системы.

На практике наиболее часто используется система 1. Рассмотрим некоторые свойства функций, входящих в эту систему, выражаемые

через приведенные ниже тождественные соотношения. Все тождественные соотношения могут быть легко доказаны с помощью таблиц истинности рассматриваемых функций. Например, так называемое правило де Моргана $x\bar{y} = \overline{x\vee y}$ можно доказать с помощью табл. 5. Из этой таблицы видно, что левая часть исследуемого выражения равна его правой части.

Ниже указаны основные тождественные преобразования переключательных функций:

$$\begin{aligned}
 x\vee 0 &= x; x1 = x; x\vee \bar{x} = 1; x\bar{x} = 0; \\
 x\vee 1 &= 1; x0 = 0; \bar{0} = 1; \bar{1} = 0; \\
 x\vee y &= y\vee x; xy = yx; & (\text{коммутативность}); \\
 (x\vee y)\vee z &= x\vee (y\vee z); (xy)z = x(yz); & (\text{ассоциативность}); \\
 x(y\vee z) &= xy\vee xz; x\vee yz = & (\text{дистрибутивность}); \\
 &= (x\vee y)(x\vee z); \\
 x\vee x &= x; xx = x; & (\text{идемпотентность}); \\
 \bar{\bar{x}} &= x; & (\text{правило двойного отрицания}); \\
 x\vee xy &= x; x(x\vee y) = x; & (\text{поглощение}); \\
 xy\vee x\bar{y} &= x; (x\vee y)(x\vee \bar{y}) = x; & (\text{склеивание}); \\
 \overline{x\vee y} &= \bar{x}\bar{y}; \overline{\bar{x}\bar{y}} = (x\vee y); & (\text{правило де Моргана}); \\
 x\vee y &= \overline{\bar{x}\bar{y}}; xy = \overline{\bar{x}\vee \bar{y}}; \\
 x(\bar{x}\vee y) &= xy; x\vee \bar{x}y = x\vee y.
 \end{aligned}$$

С помощью этих соотношений можно производить все остальные преобразования переключательных функций. Порядок выполнения действий следующий: при отсутствии в выражении скобок первыми должны выполняться операции отрицания, затем конъюнкции и последними — дизъюнкции. В процессе таких преобразований часто необходимо находить отрицания некоторых выражений. Это можно осуществить с помощью следующего правила: если A — произвольное выражение (построенное, в свою очередь, из выражений $a_1, a_2, \dots, a_n$ с помощью операций дизъюнкции, конъюнкции и отрицания), то отрицанием A будет выражение, полученное из A путем замены операций ($\&$) на ($\vee$), операций ($\vee$) на ($\&$) и подстановки вместо a выражений $\bar{a}$. Легко заметить, что выражения, записанные в одной строке, являются двойственными друг другу. Например, $x\vee 0 = \bar{x}1$; $x\vee y = \overline{\bar{x}\bar{y}}$ и т. п.

Рассмотрим теперь систему 2, получившую название алгебры Жегалкина (исследована советским ученым И. И. Жегалкиным), состоящую из двух бинарных функций конъюнкции xy и суммы по модулю 2 $x \oplus y$, а также константы 1. В отличие от системы 1 алгебра Жегалкина не содержит унарных функций. Основными тождествами в алгебре Жегалкина являются следующие:

$$\begin{aligned}
 x \oplus 0 &= x; x1 = x; x \oplus 1 = \bar{x}; x0 = 0; \\
 x \oplus y &= y \oplus x; xy = yx; & (\text{коммутативность}) \\
 (x \oplus y) \oplus z &= x \oplus (y \oplus z); (xy)z = x(yz); & (\text{ассоциативность}) \\
 x(y \oplus z) &= xy \oplus xz; & (\text{дистрибутивность}) \\
 x \oplus x &= 0; x x = x.
 \end{aligned}$$

Связь между системами 1 и 2 выражается с помощью следующих тождеств:

$$x \vee y = x \oplus y \oplus xy; \bar{x} = x \oplus 1; x \oplus y = x\bar{y} \vee \bar{x}y.$$

С помощью этих же соотношений можно переходить от задания функции в алгебре Жегалкина к заданию ее в системе 1 и наоборот.

В заключение рассмотрим систему 5 и двойственную ей систему 6.

$$\begin{aligned} x \downarrow y &= \overline{x \vee y}; & x/y &= \overline{\bar{x}y}; \\ x \downarrow y &= y \downarrow x; & x/y &= y/x. \end{aligned}$$

Функции стрелка Пирса и штрих Шеффера являются неассоциативными, поэтому при преобразованиях необходимо с помощью скобок указывать порядок выполнения действий. Если же порядок действий не указан, то выражения $x_1 \downarrow x_2 \downarrow \dots x_n$ и $x_1/x_2/\dots/x_n$ воспринимают как n -местные функции $x_1 \vee x_2 \vee \dots \vee x_n$ и $x_1 x_2 \dots x_n$.

Рассматриваемые системы характеризуются следующими тождественными соотношениями:

$$\begin{aligned} xy &= \bar{x} \downarrow \bar{y}; & x \vee y &= \bar{x}/\bar{y}; \\ xy &= x/y; & x \vee y &= x \downarrow y; \\ \bar{x} &= x \downarrow x; & \bar{x} &= x/x; \\ x \downarrow y \downarrow y &= x \downarrow y; & x/y/y &= x/y; \\ (x \downarrow y) \downarrow \bar{z} &= (\bar{x} \downarrow \bar{z}) \downarrow (\bar{y} \downarrow \bar{z}); & (x/y)/\bar{z} &= (\bar{x}/\bar{z})/(\bar{y}/\bar{z}); \\ x \downarrow y \downarrow z \downarrow u &= (\bar{x} \downarrow \bar{u}) \downarrow z \downarrow u; & x/y/z/u &= (\bar{x}/\bar{u})/z/u; \\ x \downarrow y \downarrow z \downarrow 0 &= x \downarrow y \downarrow z; & x/y/z/1 &= x/y/z; \\ x \downarrow y \downarrow z \downarrow 1 &= 0; & x/y/z/0 &= 1; \\ x \downarrow (x \downarrow y) &= x \downarrow \bar{y}; & x/(x/y) &= x/\bar{y}. \end{aligned}$$

Доказательство этих тождеств может быть получено, например, заменой стрелки Пирса и штриха Шеффера функциями системы с последующими преобразованиями.

2.2. КАНОНИЧЕСКИЕ ФОРМЫ ПРЕДСТАВЛЕНИЯ ПЕРЕКЛЮЧАТЕЛЬНЫХ ФУНКЦИЙ

Одной и той же переключательной функции могут соответствовать различные суперпозиции функций функционально полной системы (в этом нетрудно убедиться, используя приведенные в п. 2.1 тождественные соотношения). В связи с этим возникает задача нахождения такой формы записи функций, при которой каждой функции будет соответствовать одна и только одна формула стандартного типа и каждой формуле стандартного типа будет соответствовать одна и только одна функция. Такие формы записи функций называют каноническими. С помощью соотношения

$$f(x_1, x_2, \dots, x_n) = x_1^0 (0, x_2, \dots, x_n) \vee x_1^1 f(1, x_2, \dots, x_n),$$

$$\text{где } x^\alpha = \begin{cases} \bar{x} & \text{при } \alpha = 0, \\ x & \text{при } \alpha = 1, \end{cases}$$

любую n -местную функцию можно разложить по переменной x_1 на две составляющие функции, каждая из которых зависит уже от $n - 1$ аргументов. В свою очередь, функции $f(0, x_2, \dots, x_n)$ и $f(1, x_2, \dots, x_n)$ можно разложить по переменной x_2 и т. д.

Разложение функции по m -переменным будет иметь вид

$$f(\vec{x}_n) = \bigvee_{\vec{\alpha}_m} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_m^{\alpha_m} f(\alpha_1, \alpha_2, \dots, \alpha_m, x_{m+1}, x_{m+2}, \dots, x_n),$$

где символ $\bigvee_{\vec{\alpha}_m}$ означает дизъюнкции по всем наборам $\vec{\alpha}_m = (\alpha_1, \alpha_2, \dots, \alpha_m)$, а $x_n = (x_1, x_2, \dots, x_n)$.

При $m = n$ получим разложение функции по всем своим аргументам

$$f(\vec{x}_n) = \bigvee_{\vec{\alpha}_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} f(\vec{\alpha}_n), \quad (2.6)$$

где $f(\vec{\alpha}_n) = f(\alpha_1, \alpha_2, \dots, \alpha_n)$.

Произвольная функция может быть представлена в виде формулы (2.6). При этом из (2.6) следует, что между всеми n -местными функциями и всеми функциями (2.6) можно установить взаимно однозначные соответствия. Поэтому представление функций в виде (2.6) является канонической формой записи переключательных функций.

Каноническую форму (2.6) можно записать в несколько ином виде. Для этого используют свойства конъюнкции ($0x = 0$; $1x = x$) и опускают 1, а также исключают члены, содержащие 0. Поэтому

$$f(\vec{x}_n) = \bigvee_{\vec{\alpha}_n}^* x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}, \quad (2.7)$$

где знак $\bigvee_{\vec{\alpha}_n}^*$ означает, что дизъюнкция берется только по тем наборам

$\vec{\alpha}_n$, на которых $f(\vec{\alpha}_n) = 1$.

Канонические формы (2.6) и (2.7) получили название совершенных дизъюнктивных нормальных форм (сокращенно СДНФ). Термин «совершенная» указывает на то, что дизъюнктивные члены (выражения вида $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$) формируются из всех аргументов $x_1, x_2, \dots, x_n$ функции $f(\vec{x}_n)$. Термин «дизъюнктивная» указывает на то, что внешней функцией разложения является дизъюнкция, а внутренней — конъюнкция, так как для вычисления значений функции надо сначала определить значения всех конъюнкций, а затем вычислить их дизъюнкцию. Термин «нормальная» указывает на то, что форма является двухуровневой, т. е. дизъюнкция конъюнкций.

СДНФ называют также совершенной нормальной формой (сокращенно СНФ) типа И/ИЛИ. Такое название указывает, что в данной СНФ внутренней функцией является конъюнкция (функция И), а внешней — дизъюнкция (функция ИЛИ). Из СДНФ можно полу-

читать еще семь типов СНФ. Для этого над правой частью выражения (2.6) поставим два отрицания. От этого значение функции не изменится. Применим к полученному выражению правило де Моргана:

$$f(\vec{x}_n) = \overline{\bigvee_{\vec{\alpha}_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} f(\vec{\alpha}_n)} = \overline{\&_{\vec{\alpha}_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} f(\vec{\alpha}_n)}, \quad (2.8)$$

где символ $\&_{\vec{\alpha}_n}$ указывает на то, что конъюнкция берется по всем наборам $\vec{\alpha}_n$.

Каноническая форма (2.8) называется СНФ типа И—НЕ/И—НЕ. В этой форме внешней и внутренней функциями является штрих Шеффера.

Применив к выражению (2.7) правило де Моргана, получим СНФ типа ИЛИ/И—НЕ

$$f(\vec{x}_n) = \overline{\&_{\vec{\alpha}_n} (x_1^{\alpha_1} \vee x_2^{\alpha_2} \vee \dots \vee x_n^{\alpha_n} \vee \bar{f}(\vec{\alpha}_n))}. \quad (2.9)$$

Здесь внутренней функцией является дизъюнкция (ИЛИ), а внешней — штрих Шеффера (И—НЕ). При получении этого выражения использовалось тождество $\bar{x}^{\bar{\alpha}} = x^{\bar{\alpha}}$.

Применяя далее к выражению (2.9) правило де Моргана, получим СНФ типа ИЛИ—НЕ/ИЛИ

$$f(\vec{x}_n) = \overline{\bigvee_{\vec{\alpha}_n} (x_1^{\alpha_1} \vee x_2^{\alpha_2} \vee \dots \vee x_n^{\alpha_n} \vee \bar{f}(\vec{\alpha}_n))}, \quad (2.10)$$

у которой внутренней и внешней функциями будут соответственно стрелка Пирса (ИЛИ—НЕ) и дизъюнкция (ИЛИ).

Если к выражению (2.10) применить еще раз правило де Моргана, то вернемся к форме (2.6). Поэтому для получения остальных канонических форм представим в СДНФ отрицание функции

$$\bar{f}(\vec{x}_n) = \bigvee_{\vec{\alpha}_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \bar{f}(\vec{\alpha}_n). \quad (2.11)$$

Если поставить знак отрицания над левой и правой частями выражения (2.11), то получим СНФ типа И/ИЛИ—НЕ

$$\bar{f}(\vec{x}_n) = \overline{\bigvee_{\vec{\alpha}_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \bar{f}(\vec{\alpha}_n)}. \quad (2.12)$$

с внутренней функцией И и внешней ИЛИ—НЕ. По правилу де Моргана из (2.12) получим СНФ типа И—НЕ/И

$$f(\vec{x}_n) = \overline{\&_{\vec{\alpha}_n} x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \bar{f}(\vec{\alpha}_n)}, \quad (2.13)$$

из которой аналогично получим СНФ типа ИЛИ/И

$$f(\vec{x}_n) = \bigwedge_{\alpha_n} (x_1^{\alpha_1} \vee x_2^{\alpha_2} \vee \dots \vee x_n^{\alpha_n} \vee f(\vec{\alpha}_n)). \quad (2.14)$$

Каноническую форму (2.14) называют также совершенной конъюнктивной нормальной формой (СКНФ). Поставив над правой частью выражения (2.14) два отрицания и следуя правилу де Моргана, получим СНФ типа ИЛИ—НЕ/ИЛИ—НЕ

$$f(\vec{x}_n) = \bigvee_{\alpha_n} (\overline{x_1^{\alpha_1}} \vee \overline{x_2^{\alpha_2}} \vee \dots \vee \overline{x_n^{\alpha_n}} \vee \overline{f(\vec{\alpha}_n)}). \quad (2.15)$$

Если к выражению (2.15) еще раз применить правило де Моргана, то получим СНФ (2.12). Таким образом, получены все восемь канонических форм типа СНФ, которые широко используются при анализе и синтезе цифровых схем. Еще одну каноническую форму можно получить из (2.6), заменив функции ИЛИ и НЕ эквивалентными суперпозициями $x \vee y = x \oplus y \oplus xy$, $\bar{x} = x \oplus 1$ в алгебре Жегалкина и выполнив необходимые преобразования. Полученную в результате этого каноническую форму называют полиномом Жегалкина

$$f(\vec{x}_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \dots \oplus a_{12 \dots n} x_1 x_2 \dots x_n. \quad (2.16)$$

Зависимость коэффициентов a от значений функции $f(\vec{x}_n)$ на различных наборах в общем виде получить трудно. Используя свойства конъюнкции и дизъюнкции ($0x = 0$; $1x = x$; $0 \vee x = x$; $1 \vee x = 1$) из канонических форм (2.8)—(2.15), можно по аналогии с формой (2.7) исключить константы и брать дизъюнкции и конъюнкции во внешней функции разложения только по тем наборам, на которых функция $f(\vec{x}_n)$ равна единице (для форм (2.8)—(2.10)) или нулю (для форм (2.12)—(2.15)). Точно так же из канонической формы (2.16) можно исключить все члены, у которых коэффициенты равны нулю, а у остальных членов опустить коэффициент 1.

Таблица 6

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Пример 1. Представим функцию $f(x, y, z)$, заданную табл. 6 в СДНФ, СКНФ и в форме

полинома Жегалкина.

$$\text{СДНФ } f(x, y, z) = \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee xyz;$$

$$\text{СКНФ } f(x, y, z) = (x \vee y \vee z)(x \vee y \vee \bar{z})(x \vee y \vee \bar{z})(\bar{x} \vee \bar{y} \vee \bar{z}).$$

Полином Жегалкина построим, воспользовавшись СНФ типа И—НЕ/И—НЕ, имеющей вид

$$f(x, y, z) = \overline{(\bar{x}y\bar{z})(\bar{x}yz)(x\bar{y}\bar{z})(xyz)}.$$

Тогда

$$\begin{aligned} f(x, y, z) &= (\bar{x}y\bar{z} \oplus 1)(\bar{x}yz \oplus 1)(x\bar{y}\bar{z} \oplus 1)(xyz \oplus 1) \oplus 1 = \bar{x}y\bar{z} \oplus \bar{x}yz \oplus \\ &\oplus x\bar{y}\bar{z} \oplus x\bar{y}z \oplus 1 \oplus 1 = (x \oplus 1)y(z \oplus 1) \oplus (x \oplus 1)yz \oplus x(y \oplus 1)(z \oplus 1) \oplus \\ &\oplus xy(z \oplus 1) = xyz \oplus xy \oplus yz \oplus y \oplus xyz \oplus yz \oplus xyz \oplus xy \oplus xz \oplus \\ &\oplus x \oplus xyz \oplus xy = xy \oplus y \oplus xz \oplus x. \end{aligned}$$

При построении канонических форм используются простейшие выражения, представляющие собой конъюнкции или дизъюнкции нескольких попарно различных переменных, взятых с отрицаниями или без них (например, x ; xy ; $(\bar{x} \vee y)$; $\bar{x}y\bar{z}$; $(x \vee y \vee z)$ и т. п.). Эти выражения называют соответственно элементарным произведением (конъюнкцией, термом) и элементарной дизъюнкцией (дизъюнктивным термом). Элементарное произведение, содержащее все переменные $x_1, x_2, \dots, x_n$ функции и равное 1 на одном наборе аргументов и 0 на остальных, называют конституентой единицы или минтермом. Элементарная дизъюнкция, содержащая все переменные функции и равная 0 на одном наборе аргументов и 1 на остальных, называется конституентой нуля или макстермом. Использование свойств дистрибутивности позволяет получать так называемые скобочные формы переключательных функций, у которых общие части нескольких элементарных произведений или дизъюнкций вынесены за скобки. Например, для функции, заданной табл. 6, одними из возможных скобочных форм, полученных из СДНФ и СКНФ, будут следующие:

$$f(x, y, z) = y(\bar{x}z \vee \bar{x}z \vee xz) \vee xy\bar{z};$$

$$f(x, y, z) = (x \vee y \vee z)(\bar{z} \vee (x \vee y)(\bar{x} \vee y)(\bar{x} \vee \bar{y})).$$

Заметим при этом, что СКНФ и другие ей подобные СНФ не являются скобочными формами, так как наличие скобок в этих формах обусловлено лишь особенностями используемой символики.

С целью упрощения записи канонических форм переключательных функций иногда используют их числовое представление. При числовом представлении в СДНФ или СКНФ вместо конституент 0 и 1 записывают номера наборов, на которых равны 1 (для СДНФ) или 0 (для СКНФ) соответствующие конституенты. Например, рассмотренную выше функцию можно в числовом представлении записать как

$$f(x, y, z) = 2 \vee 3 \vee 4 \vee 6 = \vee (2, 3, 4, 6);$$

$$f(x, y, z) = 0 \& 1 \& 5 \& 7 = \& (0, 1, 5, 7).$$

Использование цифровых ЭВМ для автоматизации процесса проектирования схем цифровых ЭВМ способствовало развитию геометрических форм представления переключательных функций. Каждый фиксированный набор аргументов может быть интерпретирован как набор координат (вектор) некоторой точки в n -мерном пространстве. Поэтому всему множеству наборов будет однозначно соответствовать множество вершин n -мерного куба (гиперкуба). Если же вершины этого куба, где функция принимает единичные значения, отметить каким-либо образом (например, цифрами или точками), то тем самым будет получено геометрическое представление переключательной функции.

На рис. 2.2. приведено геометрическое представление функции $f(x, y, z) = \vee (0, 2, 3, 4, 6)$. Набор аргументов, соответствующий отмеченной вершине куба, называют также 0-кубом. Множество всех 0-кубов (т. е. множество отмеченных вершин) получило название кубического комплекса K^0 . Две вершины куба (два 0-куба) считаются соседними, если соответствующие им наборы отличаются только по

одной переменной. Число переменных, которые изменяются при переходе от одного 0-куба к другому (т. е. от одной вершины к другой), равно расстоянию между ними. Из двух соседних 0-кубов можно сформировать 1-куб, в состав которого входят общие части 0-кубов и знак «—» (или «X»), который ставится на место изменяющейся координаты. Множество всех 1-кубов составляет кубический комплекс K^1 . Два соседних 1-куба образуют 2-куб (соседство 1-кубов определяют, считая совпадающими координаты, отмеченные знаком «—»). Все 2-кубы составляют кубический комплекс K^2 . Аналогично можно определить 3-кубы, 4-кубы и т. д., а также комплексы K^3 , K^4 и т. д. Объединение всех кубических комплексов K^0 , K^1 , ..., K^n называют кубическим комплексом K функции. Для функции на рис. 2.2. можно образовать следующие кубические комплексы:

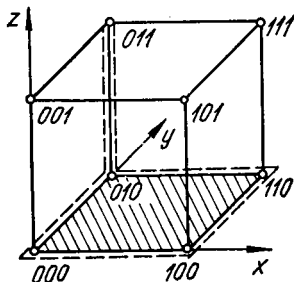


Рис. 2.2

$$K^0 = \begin{Bmatrix} 000 \\ 010 \\ 011 \\ 100 \\ 110 \end{Bmatrix}; \quad K^1 = \begin{Bmatrix} 0-0 \\ -00 \\ 01- \\ -10 \\ 1-0 \end{Bmatrix}; \quad K^2 = \{- - - 0\};$$

$$K = \begin{Bmatrix} 000 \\ 010 \\ 011 \\ 100 \\ 110 \\ 0-0 \\ -00 \\ 01- \\ -10 \\ 1-0 \\ - - - 0 \end{Bmatrix}.$$

Легко заметить, что в случае $n = 3$ комплекс K^1 соответствует множеству ребер куба между отмеченными вершинами, а комплекс K^2 — множеству граней куба, ограниченных ребрами комплекса K^1 (рис. 2.2).

2.3. МИНИМИЗАЦИЯ ПЕРЕКЛЮЧАТЕЛЬНЫХ ФУНКЦИЙ

В п. 2.2 показано, что одна и та же переключательная функция может быть представлена различными суперпозициями функций функционально полной системы. Эти суперпозиции являются моделями, по которым строятся схемы цифровых ЭВМ. Поэтому модели должны быть оптимальными по минимуму требуемого для их реализации оборудования. Количество оборудования, используемого в схеме, называют ее ценой. Процесс нахождения представления функции с минимальной ценой получил название минимизации. Для большинства полных систем эффективные алгоритмы (т. е., не сводящиеся к прямому перебору всех возможных суперпозиций) не известны. Наиболее существенные результаты по решению задачи минимизации получены для класса нормальных двухуровневых форм представления переключательных функций.

При реализации переключательных функций каждой элементарной дизъюнкции и каждому элементарному произведению соответствует некоторая достаточно простая электрическая цепь, называемая элементом. Поэтому цену какой-либо схемы обычно определяют как

суммарное число входов всех элементов, используемых для ее построения. Это число называют ценой схемы по Квайну. Если функцию реализуют по ее нормальной форме, то число входов элементов первого уровня равно числу букв в представлении функции. Число входов элементов второго уровня равно числу элементарных дизъюнкций или конъюнкций в представлении функций.

Функцию $\varphi(\vec{x}_n)$ называют импликантой функции $f(\vec{x}_n)$, если на любом наборе выполняется условие $\varphi(\vec{x}_n) \leq f(\vec{x}_n)$. Если φ и ψ являются импликантами функции f , то их дизъюнкция $\varphi \vee \psi$ также является импликантой функции f .

Простой импликантой функции f называют такое элементарное произведение, которое является импликантой функции f , но никакая его собственная часть уже не является импликантой функции f . Под собственной частью произведения $A = x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \dots x_{i_m}^{\alpha_m}$ понимают такое произведение B , которое можно получить из A вычеркиванием одной или нескольких переменных $x_{i_j}^{\alpha_j}$.

Любая функция имеет конечное множество простых импликант, число которых не больше числа конституент 1 в СДНФ. Дизъюнкция всех простых импликант функции f равна этой функции. Дизъюнкция всех простых импликант функции называется сокращенной (так как число членов в ней не больше, чем в совершенной) ДНФ функции. Каждая функция имеет единственную сокращенную ДНФ.

Один из методов получения сокращенных ДНФ, называемый методом Квайна, основан на применении операции неполного склеивания $xy \vee x\bar{y} = xy \vee x\bar{y} \vee x$. Для получения сокращенной ДНФ функции f необходимо в СДНФ функции f произвести все возможные операции неполного склеивания, а затем все возможные операции поглощения.

Произвести все возможные операции склеивания можно следующим образом. Каждая конституента 1 сравнивается с остальными. Если конституента A отличается от конституенты B только наличием отрицания у одной из переменных, то к этим конституентам применяется операция склеивания, т. е. выписывается их общая часть. В результате получают группу элементарных произведений, состоящих из $n - 1$ переменных. Описанный процесс повторяется до тех пор, пока не получим группу, к любым двум членам которой уже нельзя применить операцию склеивания. Метод Квайна применим к функциям, заданным СДНФ. Если же функция задана произвольной формой, то с помощью тождественных соотношений необходимо получить ДНФ данной функции, а затем путем умножения на $1 = x \vee \bar{x}$ получить СДНФ функции.

Говорят, что функция $\varphi(\vec{x}_n)$ накрывается функцией $f(\vec{x}_n)$, если на любом наборе $\vec{\alpha}_n$

$$f(\vec{\alpha}_n) \geq \varphi(\vec{\alpha}_n).$$

В общем случае сокращенная ДНФ не является минимальной, так как некоторые простые импликанты могут накрываться дизъюнк-

цией других членов и их можно исключить. Минимальная ДНФ (МДНФ) должна состоять исключительно из простых импликант.

Нахождение МДНФ функции f сводится к выделению некоторого числа простых импликант данной функции из всего их множества. Один из методов решения этой задачи основан на применении импликантной таблицы (матрицы). Рассмотрим построение такой матрицы для функции $f(x, y, z)$ на рис. 2.2. В импликантной табл. 7 столбцы отмечаются конституентами 1, а строки — простыми импликантами. В строке против каждой простой импликанты ставятся знаки $\times$ под теми конституентами, которые поглощаются данной простой импликантой. Непосредственно из таблицы

Таблица 7

	$\bar{x}\bar{y}\bar{z}$	$\bar{x}yz$	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz
$\bar{x}\bar{z}$	$\times$	$\times$	—	—	—
$\bar{y}\bar{z}$	$\times$	—	—	$\times$	—
$\bar{x}y$	—	$\otimes$	$\otimes$	—	—
$y\bar{z}$	—	$\times$	—	—	$\times$
$x\bar{z}$	—	—	—	$\times$	$\times$
$\bar{z}$	$\otimes$	$\otimes$	—	$\otimes$	$\otimes$

выписывают все так называемые тупиковые ДНФ, у которых ни один из дизъюнктивных членов исключить нельзя. В тупиковую ДНФ должны войти все импликанты, отмеченные только одним знаком $\times$, а также минимальное число простых импликант, на пересечении которых с любой конституентой есть хотя бы один знак $\times$. Подсчитывая число букв в тупиковой форме, определяют минимальную ДНФ.

Из табл. 7 видно, что у функции $f(x, y, z)$ имеется пять трехчленных и одна двухчленная тупиковая ДНФ, которая и является минимальной $f(x, y, z) = \bar{x}y \vee \bar{z}$.

Заметим, что задача перехода от тупиковой ДНФ к абсолютно минимальной ДНФ получила название задачи факторизации. Известные к

настоящему времени способы решения задачи факторизации не намного эффективнее перебора всех возможных суперпозиций, состоящих из двух, трех и более импликант до тех пор, пока не будут получены представления функций с абсолютно минимальным числом импликант. Вполне естественно, что такие способы из-за своей громоздкости не имеют практического значения. Поэтому задача факторизации решается приближенно путем вынесения за скобки некоторых переменных в МДНФ функции, хотя доказано, что таким способом не всегда удается получить абсолютно минимальное представление функций.

Процесс отыскания простых импликант и МДНФ можно совместить. Для этого составляют таблицу, столбцы которой отмечают конституентами. Сравнивая конституенты друг с другом, получают первую группу импликант. Каждой импликанте соответствует отдельная строка. Произведя в первой группе импликант все возможные склеивания, получим вторую группу импликант и т. д.

Существуют и другие методы отыскания простых импликант и МДНФ. Например, метод Нельсона состоит в следующем. Исследуе-

мая функция представляется в СКНФ. Далее, применив дистрибутивный закон (раскрыв скобки), получают выражение, которое содержит все простые импликанты. Для выделения их необходимо произвести все возможные операции поглощения. Затем с помощью импликантной матрицы находят МДНФ.

Эффективный способ нахождения МДНФ при небольшом числе (не более 6) переменных состоит в применении так называемых диаграмм Вейча (карт Карно) и являющихся разновидностью таблиц истинности. Можно считать также, что диаграммы Вейча представляют собой развертки n -мерных кубов на плоскость. На рис. 2.3 показаны диаграммы Вейча для функций двух, трех и четырех, пяти и шести аргументов. Каждому набору значений аргументов соответ-

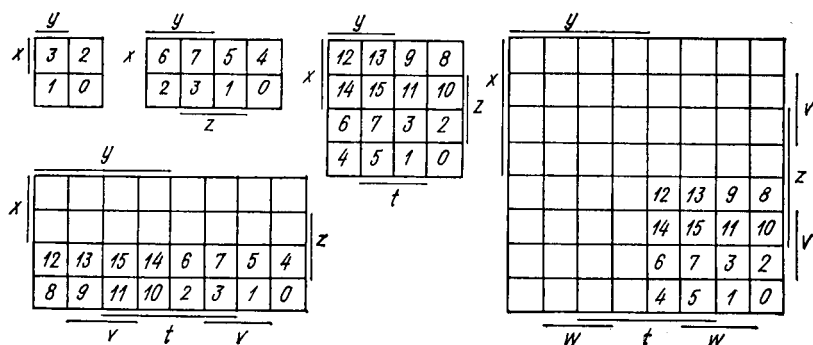


Рис. 2.3

ствует одна клетка диаграммы Вейча. Если на данном наборе аргументов функция равна 1, то в соответствующей данному набору клетке записывается 1. Клетки, соответствующие наборам, на которых функция равна 0, либо заполняют нулями, либо оставляют пустыми. Наборам (0, 0), (0, 0, 0), ... и конституентам $\bar{x}\bar{y}$, $\bar{x}\bar{y}\bar{z}$, ... соответствуют клетки с номером 0, наборам (0, 1), (0, 0, 1), ... и конституентам $\bar{x}y$, $\bar{x}yz$, ... — клетки с номером 1, наборам (1, 0), (0, 1, 0), ... и конституентам $x\bar{y}$, $\bar{x}y\bar{z}$, ... — клетки с номером 2 и т. д.

Аналогично могут быть построены диаграммы для большего числа аргументов. Однако из-за того, что в этом случае алгоритм поиска минимальных ДНФ существенно усложняется, то диаграммы для семи и более аргументов не получили распространения на практике.

Операция склеивания может быть применена только к конституентам (в общем случае элементарным произведениям), у которых все буквы, за исключением одной, совпадают. Например, $xy\bar{z}$ можно склеить с $\bar{x}y\bar{z}$, но нельзя склеить с $\bar{x}\bar{y}\bar{z}$. Такие конституенты называются соседними. Из распределения конституент на диаграмме Вейча следует, что все соседние конституенты имеют на диаграмме соседние клетки. Исключение составляют клетки, расположенные у границ диаграммы. Для устранения этого исключения условно отождествляют противоположные границы (верхнюю с нижней и левую с правой) диаграммы, т. е. диаграмму мысленно сворачивают. Любым двум

соседним клеткам соответствует элементарное произведение, являющееся общей частью соответствующих двух конъюнктов и имеющее на одну букву меньше. Любым четырем соседним ячейкам соответствует элементарное произведение, являющееся общей частью соответствующих четырех конъюнктов и имеющее на две буквы меньше и т. д. Поэтому склеивать можно соседние две, четыре, восемь, шестнадцать и т. д. единиц на диаграмме Вейча, а отыскание минимальной ДНФ сводится к определению минимального числа наиболее коротких элементарных произведений, накрывающих все единицы на диаграмме данной функции. На рис. 2.4 приведена диаграмма Вейча функции $f(x, y, z) = \vee (0, 2, 3, 4, 6)$.

Задача минимизации функций, представленных в геометрической форме, часто формулируется как задача отыскания покрытия функции, имеющего минимальную цену по Квайну. При этом покрытием переключательной функции называют множество M кубов комплекса K таких, что каждая вершина комплекса K^0 включена по крайней мере в один куб множества M . Легко заметить, что кубический комплекс K^0 представляет собой множество конъюнктов переключательной функции от n аргументов, кубический комплекс K^1 — множество импликант, имеющих $n - 1$ букв, комплекс K^2 — множество импли-

	y			
	0	1	2	3
x	0	1	0	1
	1	1	0	1
	z			

Рис. 2.4

кант, имеющих $n - 2$ букв и т. д. Комплекс же K представляет собой все множество конъюнктов и импликант данной функции. Поэтому поиск минимального покрытия может производиться теми же методами, что и при использовании аналитических представлений переключательных функций (с помощью импликантных матриц, диаграмм Вейча и др.) с учетом лишь особенностей двоичной числовой нумерации наборов. Известны также методы (например, метод Рота), специально разработанные для минимизации на цифровых ЭВМ геометрических представлений переключательных функций.

Получение минимальных нормальных форм (НФ) функций, заданных каноническими формами (2.8)—(2.15), может быть сведено к нахождению МДНФ функции или ее отрицания. Канонические формы (2.8)—(2.10) были получены из СДНФ функции путем применения правила де Моргана. Но применение правила де Моргана не изменяет числа букв в выражении. Поэтому, если функцию предварительно представить в СДНФ, найти МДНФ этой функции, то, используя затем правило де Моргана, можно получить искомую минимальную НФ. Канонические формы (2.11)—(2.15) получены из отрицания функции. Поэтому для получения минимальных НФ, соответствующих этим каноническим формам, необходимо найти МДНФ отрицания функции, исходя из которой, с помощью правила де Моргана, можно получить искомую минимальную НФ.

2.4. СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ

Комбинационной схемой называют совокупность элементов, реализующих заданную систему переключательных функций и соединенных по правилам, которые соответствуют суперпозиции функций.

Функционирование элемента описывается одной переключательной функцией, которую называют элементарным логическим оператором. Об элементе в этом случае говорят, что он реализует данную функцию (оператор). На практике чаще всего используют логические элементы, реализующие функции ИЛИ, И, НЕ, ИЛИ—НЕ, И—НЕ, И—ИЛИ—НЕ, ЗАПРЕТ и др. Поэтому их называют соответственно элементами ИЛИ, И, НЕ и т. д. Условные обозначения некоторых логических элементов на схемах, принятые в общегосударственной единой системе конструкторской документации (ЕСКД), приведены на рис. 2.5.

Назовем цепью упорядоченную последовательность элементов, у которых хотя бы один вход соединен с выходом предыдущего элемента. Петлей называют замкнутую цепь, по которой сигнал с выхода i -го элемента непосредственно или через другие элементы может поступать на вход того же i -го элемента. Общим свойством комбинационных схем является отсутствие петель. Если же схемы, составленные из логических элементов, имеют петли (иначе такие схемы называют схемами с обратными связями), то их функционирование не может быть полностью описано системой переключательных функций.

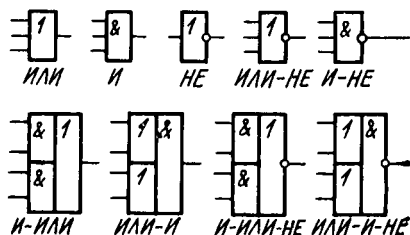


Рис. 2.5

Общеинженерные задачи анализа и синтеза в теории комбинационных схем сводятся к следующему. Задача анализа состоит в определении переключательных функций, реализуемых комбинационной схемой, по известной ее структуре. Задача синтеза заключается в построении из заданного набора логических элементов комбинационной схемы, реализующей заданную систему переключательных функций.

Задача анализа комбинационных схем без петель решается тривиально и не требует специальной методики.

Задача синтеза схем с n входами и одним выходом из элементов И, ИЛИ, НЕ и их комбинаций — И—НЕ, ИЛИ—НЕ и т. д. решается в несколько этапов. На первом этапе подлежащая реализации переключательная функция или ее отрицание (в зависимости от типов логических элементов и, следовательно, от типа используемой нормальной формы) представляется в СДНФ. Затем находится минимальная ДНФ функции. На втором этапе МДНФ функции представляют в виде суперпозиции элементарных логических операторов. На третьем этапе по операторному представлению переключательной функции составляют искомую комбинационную схему.

Задача синтеза схем с n входами и одним выходом из элементов И, ИЛИ, НЕ и их комбинаций — И—НЕ, ИЛИ—НЕ и т. д. решается в несколько этапов. На первом этапе подлежащая реализации переключательная функция или ее отрицание (в зависимости от типов логических элементов и, следовательно, от типа используемой нормальной формы) представляется в СДНФ. Затем находится минимальная ДНФ функции. На втором этапе МДНФ функции представляют в виде суперпозиции элементарных логических операторов. На третьем этапе по операторному представлению переключательной функции составляют искомую комбинационную схему.

Целью первого этапа синтеза является получение минимальной ДНФ заданной функции или ее отрицания. Подлежащая реализации переключательная функция $f(\vec{x}_n)$ может быть задана не на всех 2^n наборах аргументов $x_1, x_2, \dots, x_n$, т. е. быть неполностью определенной. На тех наборах, где функция не определена, ее определяют так, чтобы соответствующая ей схема была наиболее простой.

Решение этой задачи можно получить различными способами.

Один из них основан на методе Квайна и состоит в следующем. Неполностью определенную функцию приравнивают 1 на всех тех наборах, на которых она не определена. Для полученной таким путем функции $f_1(\vec{x}_n)$ находят все ее простые импликанты. Затем с помощью импликантной матрицы определяют наименьшее число наиболее коротких простых импликант, которые накрывают все конstituенты функции $f_0(\vec{x}_n)$, полученной из исходной путем приравнивания ее 0 на всех наборах, на которых исходная функция не определена. Дизъюнкция найденных простых импликант будет соответствовать МДНФ оптимально доопределенной функции. Такой метод является универсальным, однако при малом числе (3—4) переменных более удобно пользоваться картами Вейча.

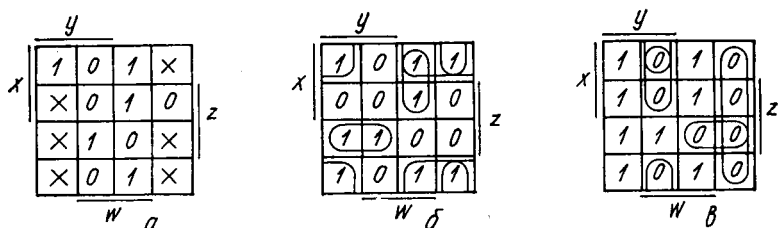


Рис. 2.6

Пример 1. Найдем МДНФ неполностью определенной функции $f(x, y, z, w)$ и ее отрицания, диаграмма Вейча которой показана на рис. 2.6, а. Знаком X отмечены наборы, на которых функция не определена.

Непосредственно из диаграммы получаем МДНФ функции

$$f_1(x, y, z, w) = \bar{z}\bar{y} \vee \bar{z}\bar{w} \vee \bar{x}y\bar{z} \vee x\bar{y}w.$$

Этой МДНФ соответствует функция, заданная диаграммой на рис. 2.6, б. При нахождении МДНФ отрицания неполностью определенной функции ее оптимальное доопределение будет иным (рис. 2.6, в). Такому доопределению соответствует функция

$$\bar{f}_2(x, y, z, w) = \bar{y}\bar{w} \vee xy\bar{w} \vee y\bar{w}\bar{z} \vee \bar{x}\bar{y}\bar{z}.$$

На втором этапе синтеза задача представления функции в виде суперпозиций операторов логических элементов решается тривиально с помощью правила де Моргана, если число входов у логических элементов достаточно велико.

Если число входов у операторов не достаточно для представления функции в нормальной форме, то производят группировку по p переменных (p — число переменных у соответствующих операторов) с помощью следующих правил:

$$\begin{aligned} A_1 A_2 \dots A_n &= (A_1 A_2 \dots A_p) (A_{p+1} A_{p+2} \dots A_{2p}) \dots (A_{mp+1} A_{mp+2} \dots A_n), \\ A_1 \vee A_2 \vee \dots \vee A_n &= (A_1 \vee A_2 \vee \dots \vee A_p) \vee (A_{p+1} \vee A_{p+2} \vee \dots \vee A_{2p}) \vee \dots \\ &\quad \vee (A_{mp+1} \vee A_{mp+2} \vee \dots \vee A_n), \end{aligned}$$

$$\bigwedge_{i=1}^n A_i = \bigwedge_{j=0}^m \left(\bigwedge_{i=1}^p A_{jp+i} \right), \quad \bigvee_{i=1}^n A_i = \bigvee_{j=0}^m \left(\bigvee_{i=1}^p A_{jp+i} \right)$$

позволяющих представить произвольную минимальную нормальную форму функции в операторной форме. Здесь m выбирается таким, чтобы $n - mp \leq p$.

Следует, однако, отметить, что использование приведенных выше соотношений не всегда приводит к операторному представлению функции с наименьшим числом операторов. Задача представления функции в форме с минимальным числом операторов аналогична задаче факторизации переключательных функций.

Третий этап синтеза состоит в составлении по операторному представлению функции первого варианта комбинационной схемы. При этом обычно считают, что заданы переменные и их отрицания. На третьем этапе часто возникают ситуации, когда суммарное число входов μ , подключенных к выходу некоторого i -го элемента, превышает

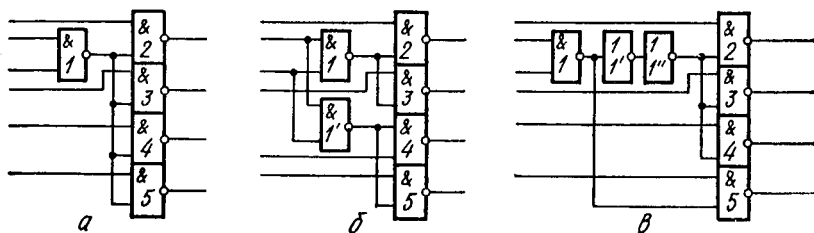


Рис. 2.7

максимально допустимое число входов N , при котором работоспособность i -го элемента еще не нарушается. В этом случае говорят, что i -й элемент перегружен. Для устранения таких перегрузок применяют дублирование элементов и усилители. Пусть, например, имеется схема (рис. 2.7, а), построенная из элементов, для которых $N = 3$. Тогда элемент 1 перегружен, так как для него $\mu > 3$. При дублировании элемент 1 заменяется двумя элементами (рис. 2.7, б). При использовании усилителей к выходу перегруженного элемента подключается развязывающий усилитель, например два элемента НЕ (рис. 2.7, в). Следует иметь в виду, что дублирование увеличивает нагрузку на элементы, являющиеся источниками входных сигналов для данной схемы. Это, в свою очередь, может обусловить необходимость дублирования этих элементов и т. д. Использование усилителей не приводит к увеличению числа элементов на всех уровнях схемы, однако вносит дополнительную задержку в распространение сигналов. Поэтому оптимальным способом разгрузки является комбинированный: там, где это допустимо по быстродействию, вводят усилители, а остальные перегруженные элементы дублируют.

Решение задачи синтеза комбинационных схем с многими выходами может быть сведено, в принципе, к синтезу схем с одним выходом. Для этого достаточно каждую из функций системы, описывающей схему, реализовать отдельно. При этом получим схему с m выходами, состоящую из m независимых комбинационных схем. Однако такая схема в общем случае не оптимальна, так как некоторые части схемы могут дублироваться. Так же, как и при синтезе схем с одним выходом, получение оптимальных (содержащих наименьшее число логи-

ческих элементов) схем с многими выходами связано с практически неосуществимым перебором большого числа вариантов схемы. Известные методы синтеза позволяют сократить перебор, однако, при этом получаются не оптимальные, а только в той или иной степени близкие к ним схемы.

Существующие методы синтеза схем с многими выходами разделяют на две группы. К первой группе относятся методы, использующие нормальные (двухуровневые) канонические формы функций. Ко второй группе относятся методы, использующие многоуровневые представления функций.

Наиболее распространенный метод синтеза, относящийся к первой группе, основан на совместной минимизации системы функций.

Простой импликантой системы функций

$$f_1, f_2, \dots, f_q \quad (2.17)$$

называют такое элементарное произведение, которое является импликантой каждой из функций системы, но никакая его собственная часть уже не является импликантой хотя бы для одной из этих функций.

На первом этапе совместной минимизации находят все простые импликанты всех возможных совокупностей функций из данной системы. Например, если минимизируется система функций (2.17), то сначала находят все простые импликанты каждой функции системы (2.17). Затем из функций системы образуют все возможные подсистемы, состоящие из двух функций. Для каждой из полученных подсистем функций находят все простые импликанты. Затем образуют все возможные подсистемы из трех функций и т. д. Заметим, что множество простых импликант системы функций $f_1, f_2, \dots, f_q$ совпадает со множеством простых импликант функций вида $F = f_1 f_2 \dots f_q$. Поэтому первый этап минимизации системы функций (2.17) сводится к нахождению простых импликант функций

$$f_1, f_2, \dots, f_q, f_1 f_2, f_1 f_3, \dots, f_1 f_q, f_2 f_3, \dots, f_1 f_2 \dots f_q,$$

число которых равно $2^q - 1$.

Пример 2. Для системы функций f_1, f_2, f_3 , заданных с помощью диаграмм Вейча (рис. 2.8, а, б, в), найти все простые импликанты всех возможных совокупностей функций. Для этого образуем все возможные произведения функций данной системы (рис. 2.8, г, д, е, ж). Непосредственно из полученных диаграмм найдем искомые импликанты:

$$\begin{aligned} p_1^{(1,2)} &= xy; & p_2^{(1,2)} &= xz; & p_1^{(1,3)} &= \bar{x}\bar{y}; & p_2^{(1,3)} &= \bar{y}z; & p_1^{(2,3)} &= p_1^{(1,2,3)} = x\bar{y}z; \\ p_1^{(1)} &= xy; & p_2^{(1)} &= xz; & p_3^{(1)} &= \bar{y}z; & p_4^{(1)} &= \bar{x}\bar{y}; & p_1^{(2)} &= y; & p_2^{(2)} &= xz; & p_1^{(3)} &= \bar{y}z; \\ & & & & & & & & & & & p_2^{(3)} &= \bar{x}\bar{y}. \end{aligned}$$

где $p_j^{(1)}$ — j -я импликанта функции f_1 ; $p_j^{(2)}$ — j -я импликанта функции f_2 ; $p_j^{(1,2)}$ — j -я импликанта системы функций $f_1 f_2$ и т. д. Соответствующая схема приведена на рис. 2.8, з.

На втором этапе минимизации из полученных простых импликант путем перебора различных вариантов отыскивают наиболее простые формулы для представления функций системы. Отыскание таких

формулы удобно производить с помощью импликантных матриц для систем функций, которые аналогичны соответствующим матрицам для одной функции. Импликантная матрица для функций из вышерассмотренного примера представлена табл. 8. Число столбцов в матрице

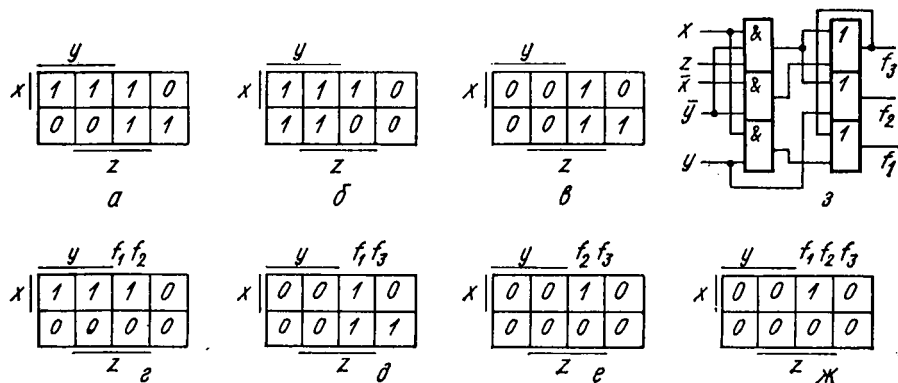


Рис. 2.8

равно суммарному числу конstituент во всех функциях системы. Каждому столбцу ставится в соответствие одна конstituента какой-либо функции системы. Для упрощения матрицы конstituенты удобно

Таблица 8

	$0^{(1)}$	$0^{(3)}$	$1^{(1)}$	$1^{(3)}$	$2^{(2)}$	$3^{(2)}$	$5^{(1)}$	$5^{(2)}$	$5^{(3)}$	$6^{(1)}$	$6^{(2)}$	$7^{(1)}$	$7^{(2)}$
$p_1^{(1, 2, 3)} = x\bar{y}z$	—	—	—	—	—	—	⊗	⊗	⊗	—	—	—	—
$p_1^{(1, 2)} = xy$	—	—	—	—	—	—	—	—	—	⊗	⊗	⊗	⊗
$p_2^{(1, 2)} = xz$	—	—	—	—	—	—	×	×	—	—	—	×	×
$p_1^{(1, 3)} = \bar{y}\bar{x}$	⊗	⊗	⊗	⊗	—	—	—	—	—	—	—	—	—
$p_2^{(1, 3)} = \bar{y}z$	—	—	×	×	—	—	×	—	×	—	—	—	—
$p_1^{(2)} = y$	—	—	—	—	⊗	⊗	—	—	—	—	⊗	—	⊗

обозначать номерами тех наборов, на которых конstituента равна 1. Показатель в скобках при этом номере указывает на ту функцию исходной системы, которой соответствует данная конstituента. Например, столбцу, обозначенному $5^{(2)}$, соответствует конstituента $\bar{x}\bar{y}z$ функции f_2 . Все столбцы, отмеченные одинаковыми конstituентами различных функций, являются соседними.

Число строк у импликантной матрицы равно числу простых импликант различных совокупностей из данной системы. Каждой строке соответствует одна простая импликанта. Если M является простой импликантой, например, функций f_1 и f_2 и, одновременно с этим, системы функций f_1, f_2 и f_3 , то эта импликанта указывается в матрице, как импликанта системы функций f_1, f_2 и f_3 . Поэтому импликантная матрица имеет 6 строк, так как система функций f_1, f_2 и f_3 имеет 6 различных импликант. Если какая-либо конституента функции f_i поглощается простой импликантой подсистемы, в которую входит функция f_i , то на пересечении соответствующих столбцов и строк ставят знак $\times$. При этом надо иметь в виду, что простые импликанты подсистемы функций могут поглощать только те конституенты, которые входят в эти функции. Например, импликанта xz является простой импликантой системы функций $f_1 f_2$ и поглощает конституенты $5^{(1)}, 5^{(2)}$, однако конституенту $5^{(3)}$ она поглощать не может. Для того чтобы импликанта $p^{(E)}$ поглощала конституенту $b^{(j)}$, необходимо $j \in E$, т. е. показатель j принадлежал множеству E показателей системы функций, для которой $p^{(E)}$ является простой импликантой.

После того, когда все необходимые знаки $\times$ проставлены, по импликантной матрице выбирают систему простых импликант, которая поглощает все конституенты всех функций системы. Такую систему простых импликант принято называть полной. Из всех полных систем импликант выбирают ту, которая содержит наименьшее число букв (так называемую минимальную полную систему). Из импликант минимальной полной системы формируют ДНФ функций. При этом, если в каком-либо столбце имеется только один знак $\times$, то соответствующая импликанта должна обязательно присутствовать в любой полной системе импликант. Такой импликантой в данном случае является $p_1^{(2)} = y$. Следовательно, минимальная полная система импликант будет $xyz, xy, \bar{x}\bar{y}, y$. Отсюда окончательно получаем

$$f_1 = \bar{x}\bar{y} \vee xy \vee x\bar{y}z;$$

$$f_2 = y \vee x\bar{y}z;$$

$$f_3 = \bar{x}\bar{y} \vee xyz.$$

Для реализации этих функций требуется 3 элемента И и 3 элемента ИЛИ с общим числом входов, равным 13. Если же реализовать функции f_1, f_2 и f_3 по их МДНФ, то для этого потребуется 6 элементов И и 3 элемента ИЛИ с общим числом входов, равным 19.

В случае, когда функции, описывающие работу комбинационной схемы, являются неполностью определенными, образуют две системы функций. Первую систему получают путем приравнивания значений функций 1 на тех наборах, на которых они не определены. Вторую систему получают путем приравнивания значений функций 0 на тех же наборах аргументов. Далее находят все простые импликанты всех возможных совокупностей функций первой системы. Из найденных простых импликант и конституент второй системы составляют импликантную матрицу для системы функций, с помощью которой формируют МДНФ оптимально доопределенных функций заданной си-

стемы. Так же, как и для одной функции, доказательство оптимальности данного метода основано на том, что наиболее короткие простые импликанты будут у первой системы, а наименьшее число конституент, необходимое для ДНФ функций, будет содержать вторая система.

Второй и третий этапы синтеза комбинационных схем с многими выходами аналогичны соответствующим этапам синтеза схем с одним выходом.

Наиболее распространенный метод синтеза схем, относящийся ко второй группе, получил название метода каскадов. В основе метода лежит формула разложения функций

$$f(x_1, x_2, \dots, x_n) = x_1 f_1 \vee \bar{x}_1 f_0,$$

где $f_i = f(i, x_2, \dots, x_n)$; $i = 0, 1$. В процессе упрощения функций можно одновременно получать их операторные представления. Если схемы строятся из двухвходовых элементов, то помимо приведенного используются также следующие соотношения:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= \overline{(x_1 \bar{f}_1)(\bar{x}_1 \bar{f}_0)} = \overline{(x_1 \vee \bar{f}_1)(x_1 \vee \bar{f}_0)} = \\ &= \overline{(x_1 \vee \bar{f}_1) \vee (x_1 \vee \bar{f}_0)} = \overline{x_1 \bar{f}_1 \vee x_1 \bar{f}_0} = \overline{(x_1 \bar{f}_1)(x_1 \bar{f}_0)} = \\ &= \overline{(x_1 \vee f_1)(x_1 \vee f_0)} = \overline{(x_1 \vee f_1) \vee (x_1 \vee f_0)}. \end{aligned}$$

В свою очередь, к функциям $f_0, f_1, \bar{f}_0, \bar{f}_1$ применяют эти же соотношения до тех пор, пока в результате очередного разложения не получат переменные или их отрицания.

Если схемы строятся из p -входовых элементов, то используют разложение функций по $p-1$ переменной:

$$\begin{aligned} f(\vec{x}_n) &= \vee x_1^{\alpha_1} x_2^{\alpha_2} \dots x_{p-1}^{\alpha_{p-1}} f(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n) = \\ &= \& (x_1^{\alpha_1} x_2^{\alpha_2} \dots x_{p-1}^{\alpha_{p-1}} f(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)) = \\ &= \& (x_1^{\bar{\alpha}_1} \vee x_2^{\bar{\alpha}_2} \vee \dots \vee x_{p-1}^{\bar{\alpha}_{p-1}} \vee \bar{f}(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)) = \\ &= \vee (x_1^{\bar{\alpha}_1} \vee x_2^{\bar{\alpha}_2} \vee \dots \vee x_{p-1}^{\bar{\alpha}_{p-1}} \vee \bar{f}(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)) = \\ &= (\vee x_1^{\alpha_1} x_2^{\alpha_2} \dots x_{p-1}^{\alpha_{p-1}} \bar{f}(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)) = \\ &= \& (x_1^{\alpha_1} x_2^{\alpha_2} \dots x_{p-1}^{\alpha_{p-1}} \bar{f}(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)) = \\ &= \& (x_1^{\bar{\alpha}_1} \vee x_2^{\bar{\alpha}_2} \vee \dots \vee x_{p-1}^{\bar{\alpha}_{p-1}} \vee f(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)) = \\ &= \vee (x_1^{\bar{\alpha}_1} \vee x_2^{\bar{\alpha}_2} \vee \dots \vee x_{p-1}^{\bar{\alpha}_{p-1}} \vee f(\alpha_1, \alpha_2, \dots, \alpha_{p-1}, x_p, \dots, x_n)). \end{aligned}$$

Здесь дизъюнкция $\vee$ и конъюнкция $\&$ берутся по всем наборам типа $(\alpha_1, \alpha_2, \dots, \alpha_n)$, т. е. внутренние функции p -местные, а внешние — 2^{p-1} -местные.

Применяя этот метод, например, для синтеза рассмотренных выше функций f_1, f_2, f_3 на двухвходовых элементах, получим

$$\begin{aligned} f_1 &= \bar{x}\bar{y} \vee x(y \vee z); \\ f_2 &= \bar{x}y \vee x(y \vee z); \\ f_3 &= \bar{x}\bar{y} \vee x(\bar{y}z). \end{aligned}$$

Поскольку метод каскадов использует многоуровневые представления функций, задержка сигналов, обусловленная конечностью времени переключения реальных логических элементов, будет больше, чем у схем, построенных по первому методу. Однако, метод каскадов дает, как правило, более простые схемы, так как учитывает ассоциативные свойства реализуемых функций.

Наряду с элементами типа И, ИЛИ, НЕ и их композициями И—НЕ, ИЛИ—НЕ и т. п., в ЦВТ используются и другие системы элементов (базисы). Задача синтеза схем в любом базисе может быть сведена к задаче синтеза в базисе И, ИЛИ, НЕ следующим образом. Пусть требуется построить комбинационную схему из элементов, реализующих операторы $\varphi_1, \varphi_2, \dots, \varphi_m$. Для этого из элементов с такими операторами строят схемы, реализующие функции И, ИЛИ, НЕ. Затем из схем, реализующих функции И, ИЛИ, НЕ, известными методами строят искомую комбинационную схему. Этот метод является универсальным, однако получаемые с его помощью схемы в большинстве случаев не оптимальны по числу используемых элементов и быстродействию. Поэтому, наряду с рассмотренным методом, разработаны различные частные методы, пригодные для данных конкретных систем операторов.

2.5. ЗАДАЧИ АНАЛИЗА И СИНТЕЗА КОНЕЧНЫХ АВТОМАТОВ

В современной теории автоматов в зависимости от принятого уровня абстрагирования выделяют два основных подхода к описанию законов функционирования и исследованию процессов переработки информации в них: абстрактный и структурный.

Автоматом (абстрактным автоматом) называют дискретное устройство, задаваемое тремя непустыми множествами, $X = \{x_1, x_2, \dots, x_n\}$; $Y = \{y_1, y_2, \dots, y_m\}$; $Z = \{z_1, z_2, \dots, z_l\}$, а также двумя функциями: функцией переходов δ и функцией выходов λ . Множества X , Y и Z называют соответственно алфавитами входных сигналов, выходных сигналов и состояний. Функция переходов δ определяет состояние автомата $z^{(s+1)}$ в зависимости от состояния $z^{(s)}$ и значения входной буквы $x^{(s)}$, т. е.

$$z^{(s+1)} = \delta(z^{(s)}, x^{(s)}),$$

где $s = 0, 1, 2, \dots$ — дискретное время; $z^{(s)}$ и $x^{(s)}$ — значения букв соответствующих алфавитов в s -е моменты дискретного времени, а функция выходов определяет значение выходной буквы $y^{(s)}$ в зависимости от состояния автомата $z^{(s)}$ и входной буквы $x^{(s)}$

$$y^{(s)} = \lambda(z^{(s)}, x^{(s)}). \quad (2.18)$$

Иногда говорят, что формально абстрактный автомат описывается пятеркой $A = \langle X, Y, Z, \delta, \lambda \rangle$. На множестве Z обычно фиксируют одно из состояний z_0 в качестве начального. Абстрактный автомат реализует отображение множества слов входного алфавита в множество слов выходного алфавита и имеет только один входной и один выходной каналы, являясь, таким образом, лишь математической моделью реальных устройств. При построении таких моделей реальный автомат с g входами и h выходами, имеющий k_1 букв во входном алфавите и k_2 букв в выходном алфавите, заменяется автоматом с одним входом и выходом, входной алфавит которого состоит из k_1^g букв, а выходной из k_2^h букв.

Структурный автомат, в отличие от абстрактного, имеет несколько элементарных входных и выходных каналов, по которым могут передаваться элементарные сигналы. Здесь учитывается структура входных и выходных сигналов, а также внутреннее устройство автомата на уровне простейших схем. Конечное множество символов, отображаемых различными элементарными сигналами и подаваемых (принимаемых) на один вход (выход) структурного автомата, называют структурным алфавитом. Чаще всего используют двузначный структурный алфавит, состоящий из двух символов 0 и 1. Каждый входной и выходной сигналы абстрактного автомата в структурном автомате представляются наборами букв структурного алфавита.

Функция выходов абстрактного автомата может быть задана двумя способами. Если выходной сигнал автомата определяется выражением (2.18), т. е. зависит как от состояния автомата, так и от входного сигнала, то автомат называют автоматом Мили. Если же выходной сигнал автомата не зависит от входного, а однозначно определяется состоянием автомата

$$y^{(s)} = \lambda(z^{(s)}),$$

то такой автомат называют автоматом Мура.

Рассматриваемые автоматы являются автоматами детерминированного типа, у которых каждой паре $z^{(s)}$ и $x^{(s)}$ однозначно соответствует пара $z^{(s+1)}$ и $y^{(s)}$. Функционирование детерминированных автоматов подчиняется следующим условиям:

1) любому входному слову из l букв (слову длиной l) соответствует выходное слово той же длины;

2) двум входным словам, в которых первые l букв совпадают, соответствуют выходные слова с совпадающими l первыми буквами, если только перед подачей входных слов автомат находился в одном и том же состоянии.

Функции $\delta(z^{(s)}, x^{(s)})$ и $\lambda(z^{(s)}, x^{(s)})$, описывающие работу автомата Мили, задаются таблицами переходов и выходов. Столбцы таких таблиц (табл. 9 и 10) обозначают символами из множества Z , а строки — символами из множества X . В клетку таблицы переходов, находящуюся на пересечении столбца z_i и строки x_j , записывают состояние автомата $\delta(z_i^{(s)}, x_j^{(s)})$, в которое он переходит из состояния $z_i^{(s)}$ под воздействием сигнала $x_j^{(s)}$. В таких же клетках таблицы выходов записывается сигнал $\lambda(z_i^{(s)}, x_j^{(s)})$. Таблицы переходов и выходов

автомата Мили могут быть представлены совмещенной таблицей, в клетках которой указаны как значения функции δ , так и значения функции λ .

Более наглядным является задание автоматов Мили с помощью графов. Граф автомата состоит из узлов (вершин), соединенных ветвями (ребрами). Узлы графа отождествляются с состояниями автомата, а ветви отмечаются входными сигналами x , вызывающими пере-

Таблица 9

	z_0	z_1	z_2	z_3
x_1	z_1	z_2	z_0	z_2
x_2	z_3	z_0	z_0	z_0
x_3	z_2	z_3	z_2	z_3

Таблица 10

	z_0	z_1	z_2	z_3
x_1	y_3	y_2	y_1	y_1
x_2	y_3	y_1	y_3	y_3
x_3	y_1	y_2	y_2	y_3

ход автомата из состояния z_i в состояние z_j и выходными сигналами $\lambda(z_i, x)$. На рис. 2.9, а приведен граф автомата Мили, заданного табл. 9 и 10.

Функции переходов и выходов автомата Мура задают одной отмеченной таблицей переходов, которая строится так же, как и таблица переходов автомата Мили, но над символами z_i отмечают символы выходных сигналов, соответствующих данному состоянию (табл. 11).

На графах автоматов Мура значения выходных сигналов записываются около узлов, а ветви обозначаются лишь входными сигналами. Пример автомата Мура, заданного графом, соответствующим табл. 11, приведен на рис. 2.9, б.

Таблица 11

	$\frac{y_3}{z_0}$	$\frac{y_2}{z_1}$	$\frac{y_2}{z_2}$	$\frac{y_1}{z_3}$
x_1	z_2	z_2	z_2	z_0
x_2	z_1	z_3	z_3	z_0

состояниями автомата. В клетку такой матрицы, находящуюся на пересечении i -й строки и j -го столбца, вписывают все входные сигналы, вызывающие переход автомата из состояния z_i в состояние z_j . Для автоматов Мили каждый входной сигнал дополнительно отмечают соответствующим выходным сигналом. Для автоматов Мура выходными сигналами отмечают строки квадратной автоматной матрицы.

Два автомата, у которых совпадают как входные, так и выходные алфавиты, называют эквивалентными, если для любого входного слова выходное слово одного автомата совпадает с выходным словом другого автомата, при условии, что оба автомата находились в одном

и том же начальном состоянии. Для любого автомата Мили можно построить эквивалентный автомат Мура и наоборот. Рассмотрим алгоритмы таких переходов на примере автомата Мили, заданного табл. 9 и 10. Поставим в соответствие каждой паре (z_i, x_j) автомата Мили состояние z_{ij} автомата Мура. Кроме того, во множество состояний автомата Мура включим начальное состояние z_0 автомата Мили. Для

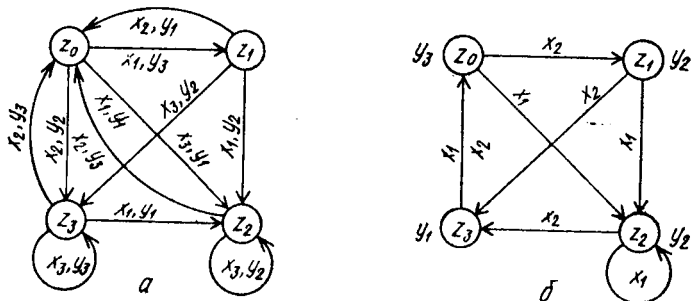


Рис. 2.9

рассматриваемого случая такое соответствие представлено табл. 12. Если автомат Мили имеет l состояний и k входных сигналов, то эквивалентный автомат Мура будет иметь $kl + 1$ состояний. Из табл. 12 видно, что состояние z_0 автомата Мили совпадает с состояниями z_{32} , z_{00} , z_{21} , z_{12} , z_{22} , автомата Мура, т. е.

$$z_0 = (z_{00}, z_{12}, z_{21}, z_{22}, z_{32}),$$

и далее

$$z_1 = (z_{01}, z_2 = (z_{03}, z_{11}, z_{23}, z_{31}), \\ z_3 = (z_{02}, z_{13}, z_{33}).$$

Поэтому переход автомата Мили из состояния z_0 в состояние z_1 должен соответствовать всем переходам автомата Мура из состояний z_{00} , z_{12} , z_{21} , z_{22} , z_{32} в состояние z_{01} , переход из z_1 в z_3 должен соответствовать всем переходам автомата Мура из z_{01} в z_{02} , z_{13} , z_{33} и т. д. Отсюда следует, что если состояние z_{ij} входит во множество состояний, совпадающих с состоянием z_r , то столбец таблицы переходов для состояния z_{ij} будет совпадать со столбцом таблицы переходов для состояния z_r . Значение же функции выходов для эквивалентного автомата Мура определяется соотношением

$$\lambda(z_{ij}^{(s)}) = \lambda(z_i^{(s)}, x_j^{(s)}) \text{ при } z_{ij} \neq z_{00}.$$

Для начального состояния z_{00} значение выходного сигнала выбирается произвольно. Таким образом, переходы и выходные сигналы эквивалентного автомата Мура определяются табл. 13.

Задача перехода от автомата Мура к эквивалентному автомату Мили решается очень просто. Если δ_1 и λ_1 — функции переходов и выходов автомата Мура, то функции переходов $\delta_2(z, x)$ и выходов

Таблица 12

	z_0	z_1	z_2	z_3
x_1	z_1 z_{01}	z_2 z_{11}	z_0 z_{21}	z_2 z_{31}
x_2	z_3 z_{02}	z_0 z_{12}	z_0 z_{22}	z_0 z_{32}
x_3	z_2 z_{03}	z_3 z_{13}	z_2 z_{23}	z_3 z_{33}

Таблица 13

	$\frac{y_3}{z_{00}}$	$\frac{y_3}{z_{01}}$	$\frac{y_2}{z_{02}}$	$\frac{y_1}{z_{03}}$	$\frac{y_2}{z_{11}}$	$\frac{y_1}{z_{12}}$	$\frac{y_2}{z_{13}}$	$\frac{y_1}{z_{21}}$	$\frac{y_3}{z_{22}}$	$\frac{y_2}{z_{23}}$	$\frac{y_1}{z_{31}}$	$\frac{y_3}{z_{32}}$	$\frac{y_3}{z_{33}}$
x_1	z_{01}	z_{11}	z_{31}	z_{21}	z_{21}	z_{01}	z_{31}	z_{01}	z_{01}	z_{21}	z_{21}	z_{01}	z_{31}
x_2	z_{02}	z_{12}	z_{32}	z_{22}	z_{22}	z_{02}	z_{32}	z_{02}	z_{02}	z_{22}	z_{22}	z_{02}	z_{32}
x_3	z_{03}	z_{13}	z_{33}	z_{23}	z_{23}	z_{03}	z_{33}	z_{03}	z_{03}	z_{23}	z_{23}	z_{03}	z_{33}

$\lambda_2(z, x)$ эквивалентного автомата Мили определяются следующими соотношениями:

$$\delta_2(z, x) = \delta_1(z, x); \lambda_2(z, x) = \lambda_1(\delta_1(z, x)).$$

Следовательно, таблица переходов эквивалентного автомата Мили совпадает с таблицей переходов автомата Мура, а в каждую клетку таблицы выходов записывают символ, которым отмечено состояние автомата Мура в данной клетке. При таком преобразовании граф автомата Мили отличается от графа автомата Мура только тем, что выходные сигналы из узлов графа перенесены на все ветви, входящие в данный узел.

Число состояний автомата существенно влияет на его сложность. Поэтому важное значение имеет определение абстрактного автомата с наименьшим возможным числом состояний, реализующего заданное отображение слов входного алфавита в слова выходного алфавита. Рассмотрим алгоритм минимизации числа состояний полностью определенного автомата. Будем называть k -эквивалентными два состояния автомата Мили ($k = 1, 2, \dots, \infty$), если автомат, находясь в любом из них, при подаче входного слова из k букв выдает одинаковые выходные сигналы. Если состояние z_i k -эквивалентно состоянию z_j , а z_j k -эквивалентно состоянию z_l , то и z_i k -эквивалентно z_l . Объединение всех k -эквивалентных состояний автомата образует класс k -эквивалентных состояний. Например, для автомата; заданного табл. 14 и 15, классы 1-эквивалентных состояний, определяемые одинаковыми столбцами таблицы выходов, следующие:

$$B_{11} = \{z_0, z_3, z_4\}; B_{12} = \{z_1, z_2, z_5, z_6\} \quad (2.19)$$

(первый индекс при B является степенью эквивалентности k , а второй — порядковым номером класса для данного k).

Для того чтобы из классов k -эквивалентных состояний сформировать классы $k+1$ -эквивалентных состояний, необходимо найти классы 1-эквивалентных состояний внутри каждого класса k -эквивалентных состояний. Признаком разбиения множества состояний автомата на классы ∞ -эквивалентности является совпадение разбиения на классы k -эквивалентных состояний с разбиением на классы $k+1$ -эквивалентных состояний. Таким образом, разбиение на классы ∞ -эквивалентных состояний может быть достигнуто за конечное число

шагов. Если теперь из каждого класса ∞ -эквивалентных состояний выбрать по одному состоянию, то полученный автомат будет иметь минимальное число состояний, равное числу классов ∞ -эквивалентности и, вместе с тем, при подаче любого входного слова будет формировать те же выходные сигналы, что и исходный автомат. Определим классы ∞ -эквивалентных состояний для автомата, заданного табл. 14 и 15. На основе табл. 14 и выражений (2.19) построим таблицу разбиения на классы 1-эквивалентности (табл. 16), заменяя при этом состояния z_i соответствующими классами эквивалентности. По табл. 16 получаем разбиение на классы 2-эквивалентности:

$$B_{21} = \{z_0, z_4\}; \quad B_{22} = \{z_3\}; \quad B_{23} = \{z_1, z_2, z_5\}; \quad B_{24} = \{z_6\}. \quad (2.20)$$

Таблица 14

	z_0	z_1	z_2	z_3	z_4	z_5	z_6
x_1	z_0	z_3	z_3	z_6	z_0	z_3	z_4
x_2	z_1	z_2	z_2	z_5	z_1	z_2	z_0
x_3	z_3	z_1	z_1	z_4	z_3	z_1	z_6

Таблица 15

	z_0	z_1	z_2	z_3	z_4	z_5	z_6
x_1	y_1	y_2	y_2	y_1	y_1	y_2	y_2
x_2	y_1	y_2	y_2	y_1	y_1	y_2	y_2
x_3	y_2	y_3	y_3	y_2	y_2	y_3	y_3

Аналогично, на основе табл. 14 и выражений (2.20) строим таблицу разбиения на классы 2-эквивалентных состояний (табл. 17), из которой получаем разбиение на классы 3-эквивалентности:

$$B_{31} = B_{21} = \{z_0, z_4\}; \quad B_{32} = B_{22} = \{z_3\}; \quad B_{33} = B_{23} = \{z_1, z_2, z_5\}; \\ B_{34} = B_{24} = \{z_6\}.$$

Таблица 16

	B_{11}			B_{12}			
	z_0	z_3	z_4	z_1	z_2	z_5	z_6
x_1	B_{11}	B_{12}	B_{11}	B_{11}	B_{11}	B_{11}	B_{11}
x_2	B_{12}	B_{12}	B_{12}	B_{12}	B_{12}	B_{12}	B_{11}
x_3	B_{11}	B_{11}	B_{11}	B_{12}	B_{12}	B_{12}	B_{12}

Таблица 17

	B_{21}		B_{22}	B_{23}			B_{24}
	z_0	z_4	z_3	z_1	z_2	z_5	z_6
x_1	B_{21}	B_{21}	B_{24}	B_{22}	B_{22}	B_{22}	B_{21}
x_2	B_{23}	B_{23}	B_{23}	B_{23}	B_{23}	B_{23}	B_{21}
x_3	B_{22}	B_{22}	B_{21}	B_{23}	B_{23}	B_{23}	B_{24}

Таким образом, разбиение множества состояний автомата на классы ∞ -эквивалентности совпадает с разбиением его на классы 2-эквивалентных состояний. Выберем теперь из каждого класса 2-эквивалентности произвольно по одному состоянию, например, z_0, z_3, z_1, z_6 . Заме-

няя в табл. 14 состояние z_4 ∞ -эквивалентным ему состоянием z_0 , состояния z_2 и z_5 — состоянием z_1 и, вычеркивая лишние столбцы в табл. 15, получаем автомат с минимальным числом состояний (табл. 18 и 19).

Для минимизации числа состояний автоматов Мура необходимо дополнительно рассматривать классы 0-эквивалентных состояний. При этом 0-эквивалентными называют любые одинаково отмеченные состояния автомата Мура. Все дальнейшие классы k -эквивалентности в данном случае определяются так же, как и для автомата Мили.

Таблица 18

	z_0	z_1	z_3	z_6
x_1	z_0	z_3	z_6	z_4
x_2	z_1	z_2	z_5	z_0
x_3	z_3	z_1	z_4	z_6

Таблица 19

	z_0	z_1	z_3	z_6
x_1	y_1	y_2	y_1	y_2
x_2	y_1	y_2	y_1	y_2
x_3	y_2	y_3	y_2	y_3

В схемах ЦВТ часто встречаются автоматы, на входы которых некоторые последовательности сигналов никогда не подаются. Такие последовательности называют запрещенными входными словами. При наличии запрещенных входных слов функции переходов и выходов будут определены не для всех пар значений z и x . Такие автоматы называют частичными автоматами. Графы частичных автоматов имеют узлы, у которых количество выходящих ветвей меньше числа букв входного алфавита. При синтезе частичного автомата производят доопределение функций переходов и выходов с тем, чтобы максимально упростить его реализацию.

Как и в теории переключательных функций, в теории автоматов основными задачами являются задачи анализа и синтеза автоматов. Под анализом автомата понимают установление закона его функционирования по заданной схеме автомата, а под синтезом — построение схемы автомата из более простых (элементарных) автоматов по заданному закону функционирования.

С инженерной точки зрения наибольший интерес представляет задача структурного синтеза автоматов.

Для решения этой задачи прежде всего необходимо описать условия функционирования автомата на некотором формальном языке, т. е. ввести стандартную форму задания автоматов. Выше рассмотрены несколько таких стандартных форм задания автоматов с помощью таблиц функций переходов и выходов.

Закон функционирования автомата часто может быть задан в описательной форме — на обычном языке. Поскольку такая форма не является математически строгой, то принципиально нельзя получить алгоритм перехода от описательной формы задания автомата к стандартной. Такой переход должен осуществляться интуитивно. Поэтому

стандартная форма задания автомата должна быть достаточно удобной не только для решения задачи синтеза, но и для перехода к ней от различных других форм задания автоматов, и, прежде всего, от описательной формы. На этапе структурного синтеза в качестве такой стандартной формы задания автоматов обычно используют кодированные буквами соответствующих структурных алфавитов таблицы переходов и выходов. Кроме того, задают или выбирают набор элементарных автоматов и логических элементов. В результате выполнения этапа структурного синтеза получают структурную схему автомата, т. е. способ соединения между собой элементарных автоматов и логических элементов, обеспечивающий заданные условия функционирования автомата.

2.6. ЭЛЕМЕНТАРНЫЕ АВТОМАТЫ, МЕТОДЫ СТРУКТУРНОГО СИНТЕЗА АВТОМАТОВ

В цифровых ЭВМ в качестве запоминающих элементов чаще всего применяют триггерные схемы (см. 2.10), являющиеся элементарными автоматами Мура с двумя внутренними состояниями, которым соответствуют два различных выходных сигнала. Это дает возможность обозначать состояния и выходные сигналы элементарных автоматов одинаковыми символами и кодировать их буквами двузначного структурного алфавита, в качестве которых обычно используют цифры 0 и 1. Входной структурный алфавит таких элементарных автоматов также является двузначным.

Если для каждой пары z_i и z_j автомата найдется хотя бы один входной сигнал, переводящий его из состояния z_i в состояние z_j при $i \neq j$ и $i = j$, то такой автомат называют автоматом с полной системой переходов.

Если автомат Мура в каждом состоянии выдает выходной сигнал, отличающийся от сигналов, соответствующих другим состояниям, то такой автомат называют автоматом с полной системой выходов. Все реальные автоматы Мура с двумя внутренними состояниями имеют полную систему выходов.

Набор элементарных автоматов и логических элементов называют структурно полным, если из элементов этого набора можно построить любой автомат. Необходимыми и достаточными условиями структурной полноты являются следующие:

1) набор должен содержать хотя бы один элементарный автомат с двумя внутренними состояниями с полными системами переходов и выходов;

2) набор должен содержать функционально полную систему логических элементов.

Рассмотрим типы элементарных автоматов с полной системой переходов.

Существует только четыре элементарных автомата детерминированного типа с одним входом и полной системой переходов. В табл. 20 столбцы 1 и 2 представляют собой все возможные наборы значений входного сигнала $x^{(s)}$ и состояния автомата $z^{(s)}$. Остальные столбцы представляют состояния $z^{(s+1)}$. Для автомата с полной системой пере-

Таблица 20

$x^{(s)}$	$z^{(s)}$	$z^{(s+1)}$			
1	2	3	4	5	6
0	0	0	0	1	1
0	1	0	1	1	0
1	0	1	1	0	0
1	1	1	0	0	1

Таблица 21

$x_0^{(s)}$	$x_1^{(s)}$	$z^{(s)}$	$z^{(s+1)}$		
1	2	3	4	5	6
0	0	0	—	0	0
0	0	1	—	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	—	—	1
1	1	1	—	—	0

ходов таблицу переходов необходимо заполнять так, чтобы в столбцах $z^{(s)}$ и $z^{(s+1)}$ были указаны все четыре типа переходов: $0 \rightarrow 0$; $0 \rightarrow 1$; $1 \rightarrow 0$; $1 \rightarrow 1$. Сравнивая столбцы 3 и 5 табл. 20, а также столбцы 4 и 6, легко заметить, что соответствующие автоматы различаются только способами кодирования входных сигналов $x^{(s)}$. Если же в указанных парах для одного из автоматов в качестве входного сигнала использовать сигнал $\bar{x}^{(s)}$, то их таблицы переходов совпадут. Следовательно, имеется лишь два основных принципиально отличающихся типа элементарных автоматов с одним входом. Будем далее считать, что этим автоматам соответствуют столбцы 3 и 4 табл. 20. Состояния $z^{(s+1)}$ первого из этих элементарных автоматов совпадают со значениями входного сигнала $x^{(s)}$. Поэтому первый автомат является обычным элементом задержки входных сигналов на один такт, а его функцию переходов можно аналитически записать в следующем виде:

$$\begin{aligned} z^{(s+1)} &= \delta(z^{(s)}, x^{(s)}) = \\ &= x^{(s)} \bar{z}^{(s)} \vee x^{(s)} z^{(s)} = x^{(s)}. \end{aligned} \quad (2.21)$$

Функцию переходов второго элементарного автомата (столбец 4 в табл. 20) можно записать в виде

$$z^{(s+1)} = \delta(z^{(s)}, x^{(s)}) = x^{(s)} \bar{z}^{(s)} \vee \bar{x}^{(s)} z^{(s)}, \quad (2.22)$$

т. е. такой автомат реализует сложение по модулю 2 значений входных сигналов. Выражения (2.21) и (2.22) полностью характеризуют функционирование элементарных автоматов с одним входом при всех возможных значениях входных сигналов и представляют собой аналитическую форму записи функций переходов.

Элементарные автоматы с двумя входами часто строятся так, что если на входе x_0 автомата действует сигнал 1 (0), а на входе x_1 — 0 (1), то автомат устанавливается в нулевое (единичное) состояние независимо от того, в каком состоянии он находился ранее. Поэтому автомат, заданный столбцами 1—4 табл. 21, является автоматом детерминированного типа и имеет полную систему переходов. Пустые места в столбце 4 табл. 21 можно заполнять различными способами, получая при этом различные типы элементарных автоматов. Например, технически реализуем такой автомат, который при $x_0^{(s)} = x_1^{(s)} = 0$ сохраняет свое предыдущее состояние, но при $x_0^{(s)} = x_1^{(s)} = 1$ его состояние $z^{(s+1)}$ неопределено (столбец 5 табл. 21). Комбинации входных сигналов, вызывающие неопределенные переходы автомата, на-

зывают запрещенными. Синтез автоматов необходимо производить так, чтобы запрещенные комбинации не могли появиться на их входах. Для автомата, заданного столбцом 5 табл. 21, допустимыми комбинациями сигналов $x_0^{(s)}$ и $x_1^{(s)}$ являются лишь те, для которых выполняется условие

$$x_0^{(s)} x_1^{(s)} = 0. \quad (2.23)$$

Вход $x_0^{(s)}$ автомата называют нулевым, а вход $x_1^{(s)}$ — единичным.

Аналитическую форму функции $z^{(s+1)}$ рассматриваемого автомата можно получить, если считать его состояния $z^{(s+1)}$ неполностью определенной функцией аргументов $x_0^{(s)}$, $x_1^{(s)}$ и $z^{(s)}$. В этом случае

$$z^{(s+1)} = x_1^{(s)} \vee \bar{x}_0^{(s)} z^{(s)}. \quad (2.24)$$

Таким образом, реакция автомата на входные сигналы определяется выражениями (2.23) и (2.24).

Автомат, заданный столбцом 5 табл. 21, может быть переведен из состояния $z^{(s)} = 0$ (1) в состояние $z^{(s+1)} = 0$ (1) двумя комбинациями входных сигналов, а именно

$$x_0^{(s)} = 0(0), \quad x_1^{(s)} = 0(0) \quad \text{и} \quad x_0^{(s)} = 1(0), \quad x_1^{(s)} = 0(1).$$

Автоматы, переходящие из состояния z_i в состояние z_j (при $i = j$ и $i \neq j$) под воздействием нескольких комбинаций входных сигналов, называют автоматами с избыточной системой переходов. Избыточность системы переходов элементарных автоматов можно использовать для упрощения схем синтезируемых автоматов. Поэтому, если схемы двух элементарных автоматов характеризуются одинаковой сложностью, следует выбирать автомат с большей избыточностью системы переходов.

Столбец 6 табл. 21 определяет функционирование элементарного автомата без запрещенных входных комбинаций. При $x_0^{(s)} = x_1^{(s)} = 1$ этот автомат меняет свое состояние на противоположное, а его функция переходов имеет вид

$$z^{(s+1)} = \delta(z^{(s)}, x^{(s)}) = \bar{x}_0^{(s)} z^{(s)} \vee x_1^{(s)} \bar{z}^{(s)}.$$

Любой переход в таком автомате может быть осуществлен под воздействием двух различных комбинаций входных сигналов.

Рассмотренные выше элементарные автоматы с одним и двумя входами являются наиболее распространенными и не исчерпывают всего многообразия их типов. В общем случае, если элементарный автомат имеет n входов, на которые могут подаваться сигналы 0 и 1, а на его выходе возможны 5 состояний (0, 1, $z^{(s)}$, $\bar{z}^{(s)}$ и неопределенное), то можно задать 5^{2^n} различных законов функционирования автомата. Например, при $n = 1$ можно задать 25 различных элементарных автоматов (табл. 22), при $n = 2$ — 625 автоматов. Однако при этом значительная часть таких автоматов или оказывается тривиальной или же не находит практического применения (см. п. 2.9).

Общие методы структурного синтеза автоматов из произвольных структурно полных наборов элементарных автоматов и логических

$x(s)$	$z(s+1)$											
	Номера вариантов элементарных											
	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	1	1	0	$z^{(s)}$	$z^{(s)}$	0	$\bar{z}^{(s)}$	$\bar{z}^{(s)}$	1	$z^{(s)}$
1	0	1	0	1	$z^{(s)}$	0	$z^{(s)}$	$\bar{z}^{(s)}$	0	$z^{(s)}$	$z^{(s)}$	1

элементов к настоящему времени еще не созданы. Так называемый канонический метод структурного синтеза предполагает сведение задачи синтеза произвольных автоматов к задаче синтеза комбинационных схем. Этот метод ориентирован на использование структурно полного набора, включающего в себя элементарные автоматы Мура, и на обобщенное представление структурной схемы автомата в виде, показанном на рис. 2.10.

Обобщенная схема автомата состоит из элементарных автоматов, $Q_1, Q_2, \dots, Q_m$ и комбинационной схемы F , связанных между собой так, чтобы выполнялись заданные условия работы автомата. Каждое состояние z_i синтезируемого автомата кодируется набором состояний элементарных автоматов.

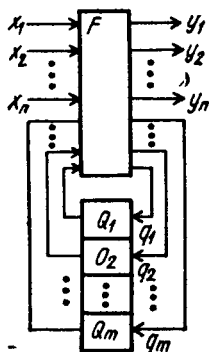


Рис. 2.10

Структурный синтез сводится к построению такой схемы автомата, которая функционирует в соответствии с заданными таблицами переходов и выходов автомата. Для такого построения необходимо:

1. Поставить в соответствие каждой из букв x_i и y_j входного и выходного алфавитов абстрактного автомата совокупности букв структурного алфавита. Такие процедуры называют кодированием входных и выходных сигналов.
2. Определить количество и выбрать типы элементарных автоматов.
3. Поставить в соответствие каждому состоянию z_i абстрактного автомата совокупность состояний $Q_1, Q_2, \dots, Q_m$ элементарных автоматов. Этот этап называют кодированием состояний автомата.
4. Найти переключательные функции $q_1, q_2, \dots, q_m$ и $y_1, y_2, \dots, y_p$. Эти функции определяют структуру комбинационной схемы F .

Количества r и p входов и выходов структурных автоматов, а также количество m элементарных автоматов при использовании двужанчного структурного алфавита определяются из условий

$$r \geq \log_2 n, \quad p \geq \log_2 k, \quad m \geq \log_2 l,$$

где n и k — соответственно число букв во входном и выходном алфавитах абстрактного автомата; l — число состояний автомата.

автоматов с одним входом												
13	14	15	16	17	18	19	20	21	22	23	24	25
1	$\bar{z}^{(s)}$	0	—	1	—	$z^{(s)}$	—	$\bar{z}^{(s)}$	—	$z^{(s)}$	$\bar{z}^{(s)}$	—
$\bar{z}^{(s)}$	1	—	0	—	1	—	$z^{(s)}$	—	$\bar{z}^{(s)}$	$\bar{z}^{(s)}$	$z^{(s)}$	—

Функция $q_i^{(s)}$ ($i = 1, 2, \dots, m$) называется функцией возбуждения элементарного автомата Q_i и определяет значение входного сигнала автомата в зависимости от состояний $Q_1^{(s)}, Q_2^{(s)}, \dots, Q_m^{(s)}$ элементарных автоматов и значений сигналов $x_1^{(s)}, x_2^{(s)}, \dots, x_r^{(s)}$ на входах автомата

$$q_i^{(s)} = q_i(Q_1^{(s)}, Q_2^{(s)}, \dots, Q_m^{(s)}, x_1^{(s)}, x_2^{(s)}, \dots, x_r^{(s)}).$$

Значения аргументов и функции в этом выражении определены для одного и того же момента дискретного времени, поэтому q_i является переключательной функцией. Для элементарного автомата, имеющего несколько входов, на этапе структурного синтеза следует получить функцию возбуждения каждого его входа. Определить значения функций возбуждения i -го элементарного автомата можно подстановкой в его функцию переходов значений $Q_i^{(s)}$ и $Q_i^{(s+1)}$, взятых из кодированной таблицы переходов заданного автомата, и последующего решения полученного уравнения относительно значений $q_i^{(s)}$. Так как число различных уравнений такого типа равно всего 4 (для наборов 00, 01, 10, 11 состояний $Q_i^{(s)}$ и $Q_i^{(s+1)}$), то на практике часто пользуются готовыми таблицами решения таких уравнений, называемых таблицами возбуждения элементарных автоматов. Значение $q_i^{(s)}$ может иногда оказаться неопределенным, то есть может быть равным как 1, так и 0. В таких случаях значение $q_i^{(s)}$ можно выбирать произвольно, но так, чтобы выполнялись условия появления допустимых комбинаций входных сигналов элементарных автоматов.

Функция $y_j^{(s)}$ ($j = 1, 2, \dots, p$) называется функцией выходов и определяет значение сигнала на j -м выходе автомата в зависимости от состояний элементарных автоматов и входных сигналов.

Следует иметь в виду, что выбор варианта кодирования входных и выходных сигналов, а также состояний автомата, во многом определяет его сложность. Проиллюстрируем применение канонического метода на примере синтеза автомата, заданного табл. 9 и 10, при следующем кодировании состояний, входных и выходных сигналов: $z_0 \rightarrow 00, z_1 \rightarrow 01, z_2 \rightarrow 10, z_3 \rightarrow 11, x_1 \rightarrow 00, x_2 \rightarrow 10, x_3 \rightarrow 01, y_1 \rightarrow 11, y_2 \rightarrow 10, y_3 \rightarrow 00$.

Таблица 23

$x_1^{(s)}$	$x_2^{(s)}$	$Q_1^{(s)}$	$Q_2^{(s)}$	$Q_1^{(s+1)}$	$Q_2^{(s+1)}$	$q_{01}^{(s)}$	$q_{11}^{(s)}$	$q_2^{(s)}$	$y_1^{(s)}$	$y_2^{(s)}$
0	0	0	0	0	1	—	0	1	0	0
0	0	0	1	1	0	0	1	1	1	0
0	0	1	0	0	0	0	1	0	1	1
0	0	1	1	1	0	0	—	1	1	1
0	1	0	0	1	0	0	1	0	1	1
0	1	0	1	1	1	0	1	0	1	0
0	1	1	0	1	0	0	—	0	1	0
0	1	1	1	1	1	0	—	0	0	0
1	0	0	0	1	1	0	1	1	1	0
1	0	0	1	0	0	—	—	1	1	1
1	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	1	0	0
1	1	0	0	—	—	—	—	—	—	—
1	1	0	1	—	—	—	—	—	—	—
1	1	1	0	—	—	—	—	—	—	—
1	1	1	1	—	—	—	—	—	—	—

Очевидно, что для рассматриваемого случая можно принять $r = 2$, $p = 2$, $m = 2$. Будем считать, что элементарный автомат Q_1 описывается функцией (2.24), а автомат Q_2 — функцией (2.22). Составим кодированную таблицу переходов и выходов автомата (табл. 23), где функции возбуждения элементарного автомата Q_1 обозначены как q_{01} и q_{11} , а функции возбуждения автомата Q_2 — q_2 . При заполнении столбцов для функций возбуждения элементарных автоматов воспользуемся таблицами возбуждения 24 и 25, в которых приведены все решения уравнений, получаемых из функций переходов при фиксированных $Q_i^{(s)}$ и $Q_i^{(s+1)}$. Такие таблицы могут быть также построены на основе табл. 20 и 21. Считая функции q_{01} , q_{11} , q_2 , y_1 , y_2 неполностью определенными функциями аргументов x_1 , x_2 , Q_1 и Q_2 , найдем их МДНФ:

$$\begin{aligned} q_{01} &= x_1 Q_1 \vee \bar{x}_2 \bar{Q}_1 \bar{Q}_2; & q_{11} &= x_2 \vee \bar{x}_1 Q_2 \vee x_1 \bar{Q}_1 \bar{Q}_2; \\ q_2 &= \bar{x}_2 Q_2 \vee \bar{x}_2 \bar{Q}_1; & y_1 &= x_1 \bar{Q}_1 \vee x_2 \bar{Q}_2 \vee \bar{Q}_1 Q_2 \vee x_1 \bar{x}_2 Q_1; \\ y_2 &= \bar{x}_1 \bar{x}_2 Q_1 \vee x_1 x_2 \bar{Q}_1 \vee x_1 \bar{Q}_1 Q_2. \end{aligned}$$

Структурная схема автомата, построенного по этим выражениям, представлена на рис. 2.11. Заметим, что для правильной работы авто-

Таблица 24

$Q_1^{(s)}$	$Q_1^{(s+1)}$	q_{01}	q_{11}
0	0	—	0
0	1	0	1
1	0	1	0
1	1	0	—

Таблица 25

$Q_2^{(s)}$	$Q_2^{(s+1)}$	q_2
0	0	0
0	1	1
1	0	1
1	1	0

мата необходимо, чтобы сигналы, представляющие функции возбуждения, и выходные сигналы автомата были разнесены во времени. Для этого в точках A (рис. 2.11) необходимо включить элементы временной задержки.

При использовании графического метода структурного синтеза автомат задают графом, в узлы которого перенесены коды состояний

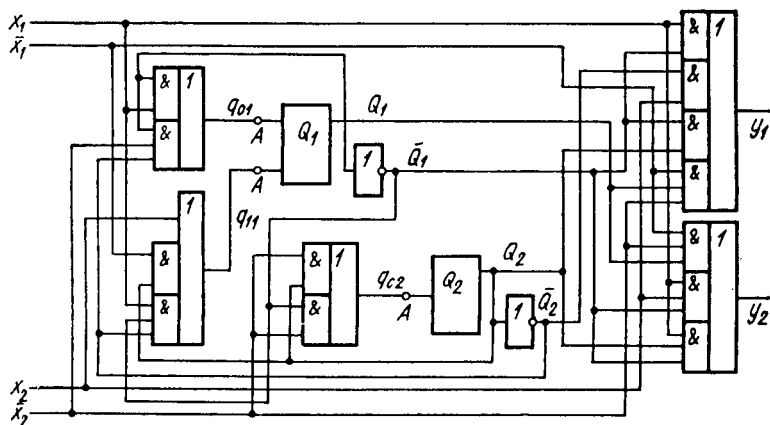


Рис. 2.11

автомата. Ветви графа также обозначают кодированными комбинациями входных и выходных сигналов. Пример кодированного графа автомата Мура приведен на рис. 2.12. Выражения для функций возбуждения и функций выходов получают следующим образом. Если при переходе автомата из одного состояния в другое i -й элементарный автомат ($i = 1, 2, \dots, m$) описывается функцией (2.22) и меняет свое состояние с 0 на 1, то соответствующая ветвь отмечается символом q_i . Если же i -й элементарный автомат описывается функцией (2.24), то указанная ветвь отмечается символом q_{1i} . При переходе i -го элементарного автомата из 1 в 0 соответствующая ветвь отмечается также символом q_i в первом случае или же символом q_{0i} во втором. Выражение для q_i (q_{0i} , q_{1i}) получают как дизъюнкцию всех выражений, каждое из которых представляет собой конъюнкцию кода узла, из которого выходит ветвь, отмеченная символом q_i (q_{0i} , q_{1i}), и соответствующего этой ветви входного сигнала (или дизъюнкции входных сигналов, если их несколько). Выражение для выходного сигнала y_j ($j = 1, 2, \dots, p$) получается как дизъюнкция всех выражений, каждое из которых представляет собой конъюнкцию кода узла, из которого выходит ветвь, отмеченная символом y_j и соответствующего этой ветви входного сигнала (или дизъюнкции входных сигналов, если их несколько). Для автомата, заданного графом на рис. 2.12, и элементарных автоматов, описываемых функциями (2.24), получим (q_{01} , q_{11} ,

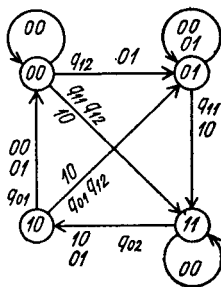


Рис. 2.12

q_{02}, q_{12} — неполностью определенные функции аргументов Q_1, Q_2, x_1 и x_2);

$$\begin{aligned} q_{01} &= Q_1 \bar{Q}_2; \quad q_{11} = \bar{Q}_1 x_1; \quad q_{02} = Q_1 Q_2 (x_1 \vee x_2); \\ q_{12} &= \bar{Q}_2 x_1 \vee \bar{Q}_1 \bar{Q}_2 x_2. \end{aligned}$$

Синтез частичных автоматов не отличается от изложенного выше. При тех комбинациях входных сигналов и внутренних состояний частичного автомата, которые не входят в кодированные таблицы переходов и выходов, значения функций возбуждения можно выбирать произвольно. Для правильной работы автоматов в их схемы необходимо вводить цепи установки одного из разрешенных состояний, так как при подаче питания автомат может устанавливаться в произвольное состояние, в том числе и в запрещенное.

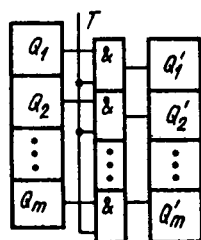


Рис. 2.13

При структурном синтезе автоматов может возникнуть ситуация, когда автомат из состояния z_i входным сигналом x_r переводится в состояние z_j , а из z_j тем же входным сигналом x_r — в состояние z_i . Если длительность входного сигнала x_r превышает время перехода автомата из одного состояния в другое, то, если не приняты специальные меры, автомат «проскочит» состояние z_j , которое является неустойчивым. Следовательно, для обеспечения устойчивости состояний автомата длительность входного сигнала должна быть меньше, чем

сумма времени прохождения сигнала по самому короткому пути в комбинационной схеме F и времени переключения элементарного автомата. Практически это условие реализуется путем стробирования входных сигналов x_r короткими синхронизирующими сигналами (СС). Входной сигнал x'_r , равный в этом случае конъюнкции сигналов x_r и СС, принимает нулевое значение к моменту окончания перехода автомата в очередное состояние. Вследствие этого, после каждого нулевого слова на вход автомата будет подаваться слово $00\dots 0$ и таблицы переходов и выходов автомата необходимо кодировать так, чтобы входное слово $00\dots 0$ не вызывало его переключения.

Другим способом обеспечения устойчивости состояний автомата является введение двойной памяти, то есть дублирование каждого элементарного автомата Q_i элементарным автоматом Q'_i (рис. 2.13). Передача информации из ряда элементарных автоматов Q_i в ряд Q'_i происходит по сигналу T , подаваемому во время отсутствия входных сигналов. Для формирования функций возбуждения и функций выходов автомата используются выходы Q'_i . Автомат из состояния z_i перейдет в состояние z_j лишь после переписи информации из ряда Q_i в ряд Q'_i . Поэтому во время действия входных сигналов автомат не может пропустить состояние z_j .

Устойчивость состояний автомата можно также обеспечить, используя две непересекающиеся во времени последовательности СС — $T1$ и $T2$. Если синхронизировать переход автомата из z_i в z_j сигналом $T1$, а переход из z_j в z_i сигналом $T2$, то эти переходы будут вызываться различными входными сигналами $x_r T1$ и $x_r T2$. Однако

этот способ применим лишь в том случае, когда все переходы в состоянии z_i могут быть синхронизированы сигналом $T1$, а все переходы из z_j — сигналом $T2$. При этом для любого узла графа автомата все входящие ветви должны быть отмечены одним из символов $T1$, $T2$, а все выходящие ветви — другим. В свою очередь, такую разметку графа можно всегда выполнить, если только все замкнутые последовательности его ветвей содержат четное их число. Для графа же, имеющего хотя бы одну нечетную замкнутую последовательность ветвей, указанную разметку осуществить нельзя. В некоторых случаях замкнутую нечетную последовательность ветвей можно преобразовать в четную путем разрыва одной ветви последовательности и включения в разрыв дополнительного состояния автомата z'_i , соответствующего нулевому выходному сигналу.

При работе автомата могут возникнуть так называемые гонки (состязания). Причиной гонок является то, что элементарные автоматы имеют различные времена переключения, различна также задержка сигналов, поступающих на входы элементарных автоматов по различным цепям схемы F . При переходе автомата из одного состояния в другое, сопровождающемуся переключением нескольких элементарных автоматов, тот из них, который изменит свое состояние раньше других, через схему F может изменить сигналы на входах элементарных автоматов до того, как они переключатся. Поэтому в процессе перехода из состояния z_i в состояние z_j под действием входного сигнала x_r автомат может оказаться в некотором состоянии z_w , переход в которое не указан в таблице переходов или на графе. Пусть, например, сигналом x_r автомат, содержащий элементарные автоматы Q_1 , Q_2 и Q_3 , переводится из состояния 011 в состояние 100. Предположим далее, что элементарные автоматы Q_2 и Q_3 уже переключились, а Q_1 остается в исходном состоянии. Тогда автомат окажется в состоянии 000. Если к тому же в автомате имеется переход из 000, например, в 010 под действием того же сигнала x_r , то автомат в результате может оказаться в состоянии 010, а не в 100.

Исключить гонки можно путем введения двойной памяти. В этом случае изменение состояний автомата происходит во время отсутствия входных сигналов. Гонки не возникают и при выборе длительности СС меньше времени прохождения сигнала по самому короткому пути в комбинационной схеме F . Кроме того, избежать гонок можно и с помощью соответствующего кодирования состояний автомата. Способы кодирования состояний автомата, исключающие гонки, называются противогоночными. Предположим, что состоянию z_i соответствует код K_i . Обозначим через $\{K_{i-j}\}$ множество кодов (наборов) состояний элементарных автоматов, которые могут возникнуть при переходе из z_i в z_j за счет одновременного переключения последних. Например, если $K_i = 0011$ и $K_j = 0100$, то $\{K_{i-j}\} = \{0011, 0000, 0001, 0010, 0101, 0110, 0111, 0100\}$. Если для любых пар переходов автомата из z_i в z_j и из z_s в z_p , вызываемых одним и тем же входным сигналом, во множествах $\{K_{i-j}\}$ и $\{K_{s-p}\}$ отсутствуют одинаковые коды, то гонки в таком автомате отсутствуют. Если же имеется хотя бы одно состояние, код которого входит как в $\{K_{i-j}\}$, так и в $\{K_{s-p}\}$, то гонки могут возникать.

Пары кодов K_i , K_j и K_s , K_p называют развязанными, если хотя бы один разряд кода принимает одно значение в паре K_s , K_p и противоположное в паре K_i , K_j . В противном случае пары кодов называются связанными. Например, пары кодов 0100, 0101 и 0000, 0001 развязаны, так как третий разряд в первой паре противоположен третьему разряду во второй паре. Связанными являются пары кодов 0100, 0101 и 0111, 0001. Для исключения гонок в автомате необходимо и достаточно закодировать его состояния так, чтобы для любых пар переходов из z_i в z_j и из z_s в z_p , вызываемых одним и тем же входным сигналом, соответствующие пары кодов K_i , K_j и K_s , K_p были развязаны.

В некоторых случаях можно использовать также способ противогоночного кодирования, получивший название соседнего кодирования. По этому способу любые два состояния автомата, соединенные ветвью на графе, кодируются наборами двоичных цифр, отличающихся лишь в одном разряде. Таким образом, при любом переходе автомата из одного состояния в другое изменяется состояние лишь одного элементарного автомата, и условие отсутствия гонок всегда будет выполнено.

2.7. СИСТЕМЫ ЭЛЕМЕНТОВ ЦИФРОВЫХ ЭВМ, ИХ КЛАССИФИКАЦИЯ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

Процесс обработки информации на устройствах цифровых ЭВМ может быть представлен как последовательность определенных операций над словами. Эти операции, в свою очередь, могут рассматриваться как совокупности некоторых элементарных (простейших) операций над буквами структурного алфавита.

Назовем блоком цифровой ЭВМ часть оборудования вычислительной машины, выполняющую определенные законченные действия, вызываемые командами программы. Любой блок состоит из двух частей. Одна часть оперирует с обрабатываемой информацией и называется операционным блоком или операционным автоматом. Другая часть воспринимает управляющую информацию и вырабатывает сигналы для организации работы операционного блока. Эта часть называется блоком местного управления или управляющим автоматом.

Процесс работы блока цифровой ЭВМ может быть представлен как совокупность конечного числа элементарных операций (микроопераций), таких как суммирование, сдвиг, установка в «0», инвертирование, прием и выдача кода и т. д. Узлом (операционной схемой) цифровой ЭВМ называем то оборудование, с помощью которого выполняется одна или несколько микроопераций. Примерами узлов является регистр, сумматор, счетчик, дешифратор, многоразрядный инвертор и др. Каждый узел выполняет заданную микрооперацию при помощи логических, запоминающих и вспомогательных элементов, входящих в его состав.

Элементами (функциональными элементами) цифровой ЭВМ называют те наименьшие единицы, на которые может быть разбита какая-либо функциональная структура (операционный узел, блок, устройство) и которые могут быть выполнены в виде элементарных, но элект-

рически законченных схем, реализующих простейшие логические функции. Таким образом, элементы являются структурно-функциональными единицами оборудования цифровой ЭВМ.

В конструктивном же отношении простейшими схемами цифровых ЭВМ являются модули — единицы электронного оборудования, имеющие законченное оформление и стандартные средства механического и электрического сопряжения с другими подобными единицами. Модульный принцип построения схем ЦВТ в настоящее время является основным принципом их конструирования, обеспечивающим высокую технологичность, регулярность, взаимозаменяемость, наращиваемость структур технических средств. Этот принцип реализуют на всех иерархических уровнях, например, в виде модуля может быть выполнено ЗУ, процессор, дешифратор, регистр. Модуль элементов содержит один или несколько элементов.

Все показатели, определяющие качество цифровых ЭВМ, в значительной степени зависят от характеристик элементов, на базе которых построена машина. Число элементов в современных цифровых ЭВМ достигает сотен тысяч. Этим и объясняется огромное внимание специалистов к совершенствованию известных и разработке новых элементов.

Назовем системой элементов цифровой ЭВМ набор элементов, использующих одинаковый принцип представления букв структурного алфавита и одинаковый тип межэлементных связей, на основе которого может быть построена любая схема цифровой ЭВМ.

Системы элементов характеризуются функциями, реализуемыми элементами, конструктивно-технологическим выполнением, показателями стоимости, надежности и быстродействия, а также электрическими параметрами. Рассмотрим подробнее некоторые из этих характеристик.

Для характеристики надежности элементов пользуются понятием интенсивности отказов λ , определяемой по формуле

$$\lambda = \frac{n}{NT} \left(\frac{1}{q} \right),$$

где N — общее количество испытываемых элементов; T — время испытания; n — количество элементов, вышедших из строя за время T .

Каждая система элементов характеризуется номиналами питающих напряжений, допусками на величины этих напряжений, полярностью и амплитудой (уровнем) входных и выходных сигналов. Кроме того, в руководящих технических материалах (РТМ) по системам элементов приводятся минимальные и максимальные значения входных и выходных сигналов, при которых элементы работают без сбоев.

Важнейшей характеристикой элементов является коэффициент разветвления (нагрузочная способность), указывающий число элементов, аналогичных рассматриваемому или той же серии, которые могут быть подключены к выходу данного элемента без нарушения его работоспособности.

Коэффициент объединения указывает максимальное число входов элемента. В РТМ указываются значения коэффициентов разветвления и объединения для наихудших, но допустимых условий работы элементов.

Элементы ЭВМ должны быть нечувствительными к действию помех. Помехой будем называть самопроизвольное изменение уровня сигнала вследствие, например, пульсации напряжения питания, нарушения временных соотношений в цепях машины, действия паразитных параметров, влияния внешних полей и т. п. Статическая помехоустойчивость элемента определяется величиной сигнала, который может быть подан на вход элемента относительно уровня 0

или 1, не вызывая ложного срабатывания элемента (например, перехода выходного сигнала из 0 в 1 и наоборот). Различают помехоустойчивость (запас помехоустойчивости) по нулевому уровню $U_{п0}$ и по единичному уровню $U_{п1}$. Определяют величины $U_{п0}$ и $U_{п1}$, используя семейство передаточных характеристик элементов $U_{вых} = f(U_{вх})$, построенных при различных, в том числе и при предельно допустимых U_{min0} и U_{max0} значениях напряжений, отображающих цифру 0, и напряжений U_{min1} и U_{max1} , отображающих цифру 1. На семействе передаточных характеристик строятся две огибающие (рис. 2.14), касательные к которым, проведенные под углом 45° к координатным осям, определяют точки А и В. В системе высоких потенциалов (СВП) величины $U_{п0}$ и $U_{п1}$ определяют так

$$U_{п0} = U_B - U_{max0}; \quad U_{п1} = U_{min1} - U_A.$$

В системе низких потенциалов (СНП) соответственно

$$U_{п0} = U_{min0} - U_A; \quad U_{п1} = U_B - U_{max0}.$$

Очевидно, что $U_{п0}$ и $U_{п1}$ будут принимать предельные значения, если абсциссы точек А и В совпадают. Запас статической помехоустойчивости определяется как

$$U_n = 0,5 (U_{п0} + U_{п1}).$$

Мощность, потребляемая элементами, определяет количество выделяемой в процессе работы тепловой энергии и в значительной степени влияет на их конструктивное исполнение. Различают мощность, потребляемую в статическом и динамическом режимах. В статическом режиме потребляемая мощность $P_{ст}$ определяется по формуле

$$P_{ст} = 0,5 (P_0 + P_1),$$

где P_0 и P_1 — соответственно мощности, потребляемые элементом в нулевом и единичном («выключенном» и «включенном») состояниях. Мощность, потребляемая в динамическом режиме, определяется при максимальной частоте переключения элементов.

Быстродействие системы элементов характеризуется следующими параметрами:

1. Максимальной частотой входных сигналов f_{max} в наилучших, но до-

пустимых условиях эксплуатации. При определении этой частоты требуют, чтобы за время одного такта $T = \frac{1}{f_{max}}$ в схеме элемента полностью заканчивались все

переходные процессы.

2. Временем переключения $t_{пер}$, которое определяется как $\max(t_{вкл}, t_{выкл})$, где $t_{вкл}$ и $t_{выкл}$ — соответственно время включения и выключения элемента. Заметим, что на временных диаграммах отсчетные точки, по которым определяют $t_{вкл}$ и $t_{выкл}$, обычно располагают на уровнях 0,1 и 0,9 номинальных значений $U_{вх}$ и $U_{вых}$, если это специально не оговорено в РТМ.

3. Временем задержки распространения сигнала (для логических элементов), которое определяется как

$$t_3 = 0,5 (t_{3.п} + t_{3.л}),$$

где $t_{3.п}$ — время задержки переднего фронта выходного сигнала относительно переднего фронта входного сигнала; $t_{3з}$ — время задержки заднего фронта выходного сигнала относительно заднего фронта входного сигнала. Измерение $t_{3.п}$ и $t_{3з}$ обычно производят на уровне половины амплитуды сигнала.

Отметим также, что качество современных интегральных элементов ЦВТ часто характеризуют одним обобщенным параметром, представляющим собой произведение средней потребляемой мощности одним элементом на время его переключения — $P\tau$. Этот параметр получил название фактора добротности или энергии единичного переключения.

2.8. ИНТЕГРАЛЬНЫЕ СИСТЕМЫ ЭЛЕМЕНТОВ ЦВТ

Основными активными компонентами цифровых интегральных микросхем являются биполярные и униполярные каналные транзисторы.

Для построения элементов ЭВМ могут использоваться биполярные транзисторы как с проводимостью типа $p-n-p$, так и с проводимостью типа $n-p-n$. Известно три основных способа включения биполярных транзисторов: с общей базой, с общим эмиттером, с общим коллектором. В схемах элементов ЭВМ наиболее часто встречаются последние два способа включения транзисторов.

Функционирование простейшего элемента на основе транзисторного каскада с общим эмиттером с учетом уровней сигналов на входе x может быть описано табл. 26 (В означает высокий уровень входного напряжения, а Н — низкий), из которой видно, что в СВП и СНП элемент реализует отрицание двузначной переменной.

Параллельное включение таких транзисторных каскадов с общим коллекторным сопротивлением позволяет реализовать в СВП операцию ИЛИ—НЕ а в СНП — операцию И—НЕ (табл. 27). При последовательном же включении таких каскадов можно реализовать операцию И—НЕ в СВП и операцию ИЛИ—НЕ в СНП (табл. 28).

Принцип действия МОП-транзисторов основан на изменении проводимости полупроводникового материала под воздействием электрического поля,

Таблица 26

x	f
Н	В
В	Н

Таблица 27

x_1	x_2	...	x_{n-1}	x_n	f
Н	Н	...	Н	Н	В
Н	Н	...	Н	В	Н
.	.	...	.	.	.
.	.	...	.	...	...
.	.	...	.	.	.
В	В	...	В	Н	Н
В	В	...	В	В	Н

Таблица 28

x_1	x_2	...	x_{n-1}	x_n	f
Н	Н	...	Н	В	В
Н	Н	...	Н	В	В
.	.	...	.	.	.
.	.	...	.	.	.
.	.	...	.	.	.
В	В	...	В	В	В
В	В	...	В	Н	Н

то есть в отличие от биполярных транзисторов каналные транзисторы управляются не током, а напряжением.

Различают МОП-транзисторы двух типов проводимости: p -типа (p -канальные) и n -типа (n -канальные). МОП-транзисторы p -типа открываются при отрицательном смещении затвора относительно истока, а транзисторы n -типа — при положительном напряжении на затворе относительно истока.

Схемы элементов часто строятся также на так называемых дополняющих МОП (МДП)-транзисторах (сокращенно МОПДТ или МДПДТ), использующих МОП-транзисторы обоих типов.

Схемы на МОП-транзисторах разделяют на статические, квазистатические и динамические. Последние два класса схем используют специфические свойства

МОП-транзисторов — сверхвысокое входное сопротивление порядка 10^8 МОм и способность затвора длительное время сохранять заряд. Наиболее широко эти классы схем используются для построения запоминающих элементов. Из числа других полупроводниковых компонент в интегральных элементах ЦВТ широко используют диоды. Простейшие схемы диодных логических элементов приведены на рис. 2.15. Функционирование этих схем описывается соответственно табл. 29 и 30 (при составлении таблиц предполагаем, что диоды идеальны, т. е. их прямое сопротивление равно нулю, обратное сопротивление бесконечно велико и, кроме того, $E \geq V$). Поскольку в СВП уровень Н соответствует 0, а уровень В — 1, то схема на рис. 2.15, а реализует конъюнкцию входных переменных, а в СНП, где уровень Н соответствует 1, а уровень В — 0, эта схема реализует дизъюнкцию n переменных. В СВП схема на рис. 2.15, б реализует дизъюнкцию n переменных, а СНП — конъюнкцию n переменных.

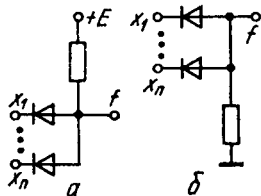


Рис. 2.15

Помимо реализации логических функций диоды в интегральных элементах выполняют также функции фиксации уровней сигналов, смещения порога срабатывания транзисторов, ограничения амплитуды помех и др.

Элементную основу современной ЦВТ составляют потенциальные элементы, реализованные методами микроэлектроники. При построении схем логических элементов проводник (шина), по которому передается двузначный сигнал, может рассматриваться как технический аналог булевой переменной, а проводники, на которые постоянно поданы напряжения, соответствующие буквам двузначного алфавита — как технические аналоги констант 0 и 1.

Существующие системы интегральных потенциальных элементов принято классифицировать по типу радиокомпонент, на которых реализуются логические функции. Можно выделить следующие наиболее часто употребляемые типы интегральных логических элементов:

1. Элементы резисторно-транзисторной логики (РТЛ).
2. Элементы резисторно-конденсаторной (емкостной) транзисторной логики (РКТЛ, РЕТЛ).
3. Элементы транзисторной логики с непосредственными связями (ТЛНС).
4. Элементы диодно-транзисторной логики (ДТЛ).
5. Элементы транзисторно-транзисторной логики (ТТЛ).
6. Элементы транзисторно-транзисторной логики с диодами Шоттки (ТТЛШ).
7. Элементы транзисторной логики с эмиттерными связями (ЭСЛ).
8. Элементы интегральной инжекционной логики (ИИЛ, И²Л).
9. Элементы на МОП-транзисторах (МОПТЛ).
10. Элементы на дополняющих МОП-транзисторах (МОПДТЛ).

Рассмотрим схемные особенности каждого из перечисленных типов элементов.

Элементы РТЛ содержат минимальное число типов радиокомпонент-транзисторы и резисторы. Схема типового элемента ИЛИ—НЕ (СВП) РТЛ показана на рис. 2.16, а. Типовой модуль РТЛ обычно содержит несколько элементов типа ИЛИ—НЕ. Параметры, характерные для лучших серий элементов РТЛ, приведены в табл. 31.

Возможности использования элементов РТЛ в схемах ЦВТ ограничиваются малыми значениями N и M . Часто при проектировании схем ЭВМ требуются

Таблица 29

x_1	x_2	$\dots$	x_{n-1}	x_n	f
H	H	$\dots$	H	H	H
H	H	$\dots$	H	B	B
$\cdot$	$\cdot$	$\dots$	$\cdot$	$\cdot$	$\cdot$
$\cdot$	$\cdot$	$\dots$	$\cdot$	$\cdot$	$\cdot$
B	B	$\dots$	B	H	B
B	B	$\dots$	B	B	B

Таблица 30

x_1	x_2	$\dots$	x_{n-1}	x_n	f
H	H	$\dots$	H	H	H
H	H	$\dots$	H	B	H
$\cdot$	$\cdot$	$\dots$	$\cdot$	$\cdot$	$\cdot$
$\cdot$	$\cdot$	$\dots$	$\cdot$	$\cdot$	$\cdot$
B	B	$\dots$	B	H	H
B	B	$\dots$	B	B	B

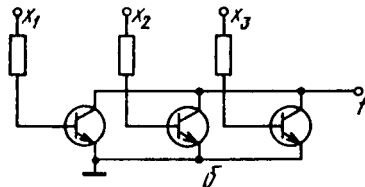
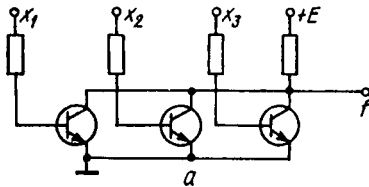


Рис. 2.16

элементы с коэффициентами объединения и разветвления, равными 15—20. Для этого при разработке систем интегральных элементов в их состав включаются логические расширители и усилители мощности. Схема типового расширителя для элементов РТЛ показана на рис. 2.16, б. Подключается расширитель к коллекторному сопротивлению схемы на рис. 2.16, а. Максимальное значение M в таком случае определяется величиной предельно допустимого тока по этому сопротивлению и допустимым изменением уровня H .

К числу положительных качеств элементов РТЛ относятся их простота и низкая стоимость. К числу же недостатков следует отнести их неэкономичность по энергопотреблению, сравнительно невысокое быстродействие и жесткие требования к допустимому разбросу величин сопротивлений.

Элементы РКТЛ в схемном отношении отличаются от элементов РТЛ только наличием конденсаторов, которые шунтируют базовые сопротивления. Введение этих конденсаторов позволило улучшить передачу фронтов переключающих сигналов и уменьшить неравномерность распределения базовых токов, свойственную элементам РТЛ, за счет увеличения базовых сопротивлений. Набор элементов РКТЛ в системе элементов повторяет набор РТЛ. Параметры лучших элементов РКТЛ приведены в табл. 31.

К положительным качествам элементов РКТЛ следует отнести большие допустимые разбросы параметров радиокомпонент и питающих напряжений, меньшую, по сравнению с РТЛ, потребляемую мощность. Недостатки этих элементов — для выполнения емкостей требуется большая площадь подложки, средняя задержка сигналов может изменяться в довольно широких пределах, элементы неустойчивы к помехам в виде коротких импульсов.

Элементы ТЛНС отличаются схемной простотой и малым потреблением энергии. Схема типового элемента имеет вид, показанный на рис. 2.17. Такие элементы технологичны в производстве и обладают высоким быстродействием. Однако эти элементы критичны к разбросу входных характеристик транзисторов и обладают низкой помехоустойчивостью. Типовые параметры элементов ТЛНС приведены в табл. 31.

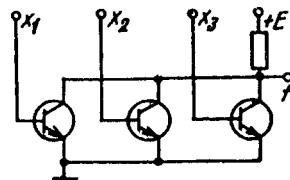


Рис. 2.17

Таблица 31

Параметры элементов	РГЛ	РКТЛ	ГЛНС	ДТЛ	ГТЛ	ГТЛШ	ЭСЛ	ИЛ	МОПТЛ	МОПДТЛ
Время задержки сигналов t_3 , мкс	$> 0,3$	$> 0,02$	$> 0,02$	$> 0,01$	$> 0,005$	$> 0,001$	$> 0,003$	$> 0,01$	$> 0,6$	$> 0,05$
Рассеиваемая мощность P_p , мВт	> 4	> 2	> 2	> 12	> 10	> 2	> 35	$> 6 \cdot 10^{-6}$	$> 0,5$	$> 50 \cdot 10^{-3}$
Коэффициент разветвления N	≤ 4	≤ 5	≤ 5	≤ 12	≤ 30	≤ 25	≤ 25	≤ 5	≤ 15	≤ 20
Коэффициент объединения M	≤ 4	≤ 6	≤ 8	≤ 10	≤ 8	≤ 8	≤ 8	≤ 5	≤ 10	≤ 10
Напряжение питания $E, В$	4	4	4	5	5	5	3	5	15	15
Уровень В, В	2,2	2,4	0,8	2,6	2,4	2,5	1,45	0,75	9,5	9,5
Уровень Н, В	0,4	0,3	0,2	0,3	0,3	0,3	0,95	0,056	1	1
Рабочая частота f_p , МГц	$\leq 0,3$	≤ 2	≤ 10	≤ 20	≤ 20	≤ 125	≤ 25	≤ 10	$\leq 0,5$	≤ 2
Запас помехоустойчивости U_n , В	$\leq 0,15$	$\leq 0,2$	$\leq 0,15$	$\leq 0,6$	$\leq 0,7$	$\leq 0,8$	$\leq 0,1$	$\leq 0,1$	≤ 2	≤ 2
Фактор добротности, пДж	> 100	> 70	> 70	> 50	> 50	> 10	> 20	$> 0,1$	> 10	$> 0,6$

Следует отметить, что в процессе развития интегральной технологии элементы РТЛ, РКТЛ, ТЛНС появились одними из первых и были характерны для аппаратных средств ЦВТ в 60-е годы. К настоящему времени эти элементы в значительной степени уступили свое место более совершенным элементам других типов.

Элементы ДТЛ применяются там, где первостепенное значение имеют не требования минимального энергопотребления, а требования высокого быстродействия и помехоустойчивости. Типовая схема элемента ДТЛ показана на рис. 2.18, а и реализует функцию И—НЕ в СВП и ИЛИ—НЕ в СНП. В элементах ДТЛ как правило, отсутствует базовое сопротивление, а функцию смещения уровня открывания транзистора выполняет пара последовательно включенных диодов. Недостатком схемы является наличие высокоомного

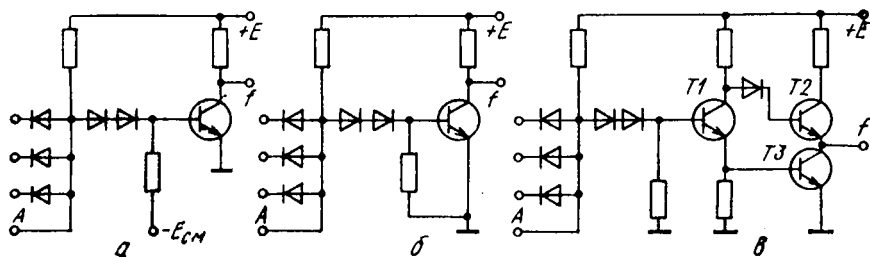


Рис. 2.18

сопротивления смещения, занимающего сравнительно большую площадь подложки и необходимость в дополнительном источнике питания. Схему на рис. 2.18, а можно упростить, исключив этот источник и уменьшив сопротивление смещения (рис. 2.18, б). Однако помехоустойчивость такой схемы снижается и возрастает критичность к разбросу параметров радиокомпонент. Диодная часть элементов ДТЛ обычно имеет вывод общего анода (А на рис. 2.18), который служит для увеличения числа входов элемента путем внешнего подключения диодных схем (рис. 2.15), т. е. для «расширения входной логики». В состав систем элементов ДТЛ включают также схемы с большой нагрузочной способностью. Один из вариантов такой схемы приведен на рис. 2.18, в. Здесь на транзисторе $T1$ выполнен буферный усиительный каскад, обеспечивающий такое управление транзисторами $T2$ и $T3$, что когда один из них открыт, то другой закрыт и наоборот. Коллекторное сопротивление транзистора $T2$ невелико и служит для ограничения коллекторного тока при замыкании выхода элемента на землю.

Для уменьшения мощности, потребляемой схемами на элементах ДТЛ, в цифровых устройствах невысокого быстродействия применяют модифицированные элементы ДТЛ, в которых отсутствуют коллекторные сопротивления. Иначе такие ДТЛ-элементы называют элементами со свободным коллектором. В схемах на таких элементах функции коллекторного сопротивления выполняет сопротивление нагрузки. Параметры элементов ДТЛ лучших серий указаны в табл. 31. К ДТЛ, например, относятся элементы серий К104, К121, К128 и др.

Элементы ТТЛ характерны тем, что с целью уменьшения влияния емкостной нагрузки (следовательно — с целью повышения быстродействия) входная цепь таких элементов выполняется на основе многоэмиттерного транзистора, имеющего низкое входное сопротивление. В таких схемах необходимо, чтобы цепь, являющаяся генератором для многоэмиттерного транзистора, при отключении последнего была низкоомной и обеспечивала бы быстрое рассасывание неосновных носителей, накопленных в базе многоэмиттерного транзистора. Для этого выходной каскад предыдущего однотипного элемента

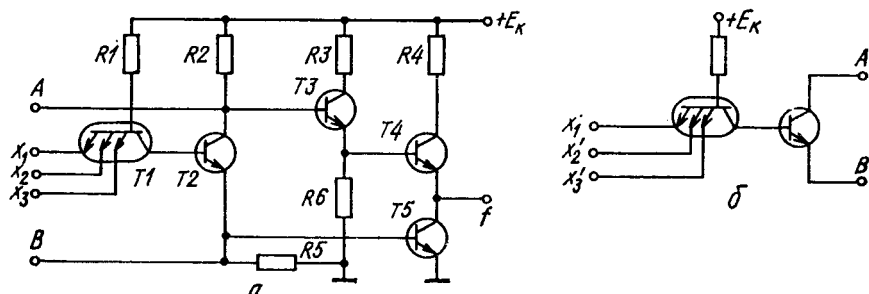


Рис. 2.19

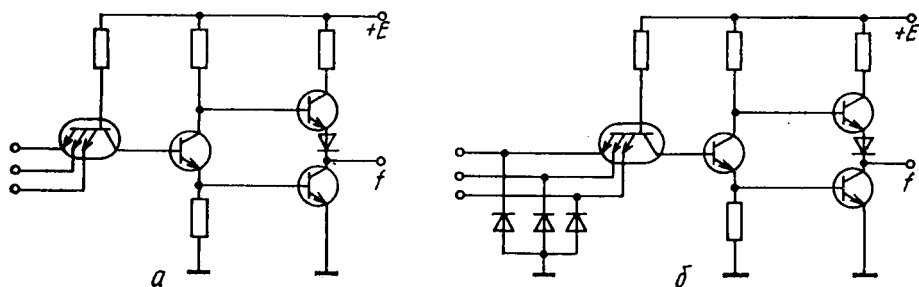


Рис. 2.20

должен быть достаточно мощным и низкоомным. Схема типового элемента ТТЛ представлена на рис. 2.19, а. Элемент реализует функцию И—НЕ в СВП и ИЛИ—НЕ в СНП.

Рассмотренный элемент имеет специальные входы А и В для расширения входной логики. Переключательную функцию такого элемента, в котором ко входам А и В подключен элемент-расширитель (рис. 2.19, б) можно в СВП записать как И—ИЛИ—НЕ или же
$$f = x_1 x_2 x_3 \vee x'_1 x'_2 x'_3.$$

Элементы ТТЛ часто используются в различных модификациях. Например, схемный вариант на рис. 2.20, а отличается экономичностью (диод в этой схеме обеспечивает запираение верхнего транзистора при низком напряжении на выходе), а вариант на рис. 2.20, б — повышенным быстродействием. Здесь диоды ограничивают максимальную амплитуду отрицательных помех на входе.

На практике системы элементов ТТЛ имеют в своем составе до нескольких десятков элементов. Основным логическим элементом является И—НЕ (ИЛИ—НЕ) с возможностью расширения входной логики по входам И и ИЛИ. Например, серия микросхем К155 — одна из самых распространенных серий ТТЛ — содержит модули логических элементов, условные обозначения которых на структурных схемах приведены на рис. 2.21. Кроме логических элементов модули серии К155 содержат также запоминающие элементы и ряд операционных схем небольшой разрядности (см. гл. 3). В настоящее время элементы ТТЛ являются одними из наиболее распространенных элементов в ЦВТ, в частности, на таких элементах построены цифровые

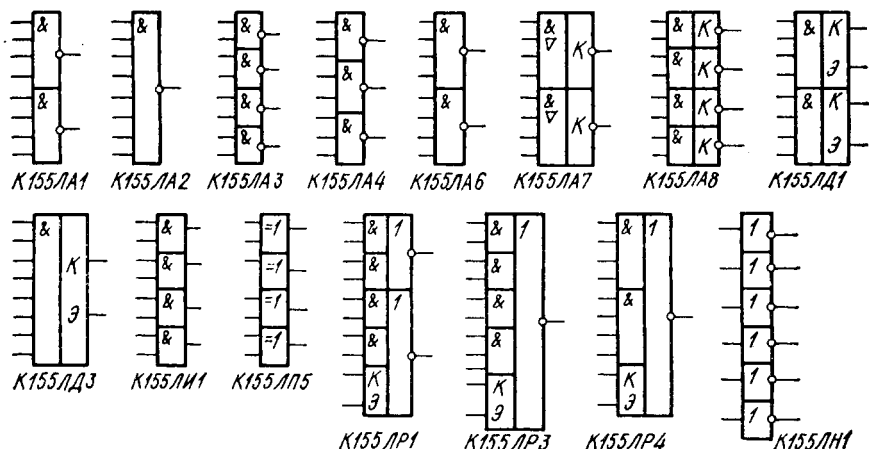


Рис. 2.21

ЭВМ ЕС-1010—ЕС-1040. Кроме того, ТТЛ является компонентной основой при построении цифровых схем со средней и большой степенью интеграции. Кроме элементов серии К155 к ТТЛ относятся серии К130, К131, К133 и др. Параметры лучших элементов ТТЛ иллюстрируются табл. 31. К преимуществам элементов ТТЛ относятся: высокая скорость работы, высокий коэффициент разветвления, большое число активных радиокомпонент, простота согласования элементов по электрическим параметрам. К числу недостатков следует отнести жесткие требования к технологическому процессу при изготовлении многоэмиттерных транзисторов и сравнительно большую потребляемую мощность.

Схемы элементов ТТЛ с диодами Шоттки (ТТЛШ) во многом повторяют схемы элементов ТТЛ, однако элементы ТТЛШ характеризуются более высоким быстродействием. Диоды Шоттки являются приборами, работа которых основана на использовании потенциального барьера в зоне контакта металла с полупроводником. Вольт-амперные характеристики диодов Шоттки по форме аналогичны характеристикам диффузионных плоскостных диодов. Принципиальное же отличие диодов Шоттки заключается в том, что они имеют более низкое прямое падение напряжения и не обладают свойствами инжекции

и накопления неосновных носителей. Поэтому время рассасывания неосновных носителей в диодах Шоттки равно нулю, а их быстродействие определяется только емкостью $p-n$ перехода. Если диод Шоттки включен параллельно переходу коллектор — база транзистора (путем технологического совмещения электродов диода и транзистора), то такой транзистор называют транзистором Шоттки. В транзисторе Шоттки снижается степень насыщения, так как втекающий в базу ток ответвляется диодом. Поскольку транзисторы Шоттки практически не подвержены накоплению зарядов, то схемы на их основе характеризуются примерно вдвое большим быстродействием и улучшенной помехоустойчивостью без обычного в таких случаях увеличения потребляемой мощности.

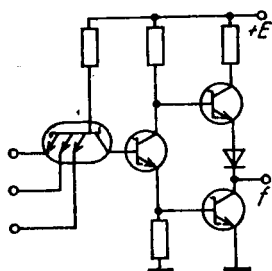


Рис. 2.22

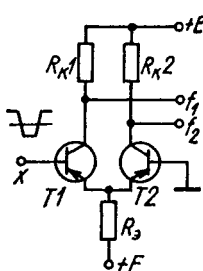


Рис. 2.23

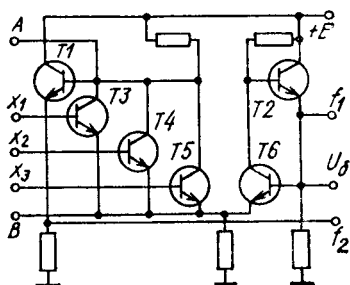


Рис. 2.24

Схема типового логического элемента ТТЛШ приведена на рис. 2.22. В СВП такой элемент реализует функцию И—НЕ, в СНП — ИЛИ—НЕ. Параметры лучших элементов ТТЛШ указаны в табл. 31. КТТЛШ, например, относятся элементы серий К530, К531.

Элементы ЭСЛ, в отличие от всех рассмотренных выше интегральных потенциальных элементов, при работе не входят в насыщение, что обуславливает существенное повышение их быстродействия. Элементы ЭСЛ работают по принципу переключения токов при малых изменениях входных напряжений. Поэтому эти элементы часто называют элементами с переключателями тока. Простейшая схема элемента ЭСЛ показана на рис. 2.23.

Для полного и надежного переключения транзисторов достаточно изменения потенциалов на базе $T1$ на 0,15—0,2 В. Уровни выходных сигналов такого элемента существенно отличаются от уровней входных сигналов и поэтому непосредственное соединение элементов невозможно. Необходимо использовать специальные схемы согласования уровней.

Схема типового элемента ЭСЛ представлена на рис. 2.24.

Такой элемент реализует в СНП следующие функции:

$$f_1 = \overline{x_1 x_2 x_3}; f_2 = x_1 x_2 x_3.$$

Число входов элемента может быть увеличено при подключении к выводам А и В транзисторов с соединенными коллекторами и эмиттерами. Типовые параметры элементов ЭСЛ приведены в табл. 31.

К числу положительных качеств ЭСЛ следует отнести большое быстродействие и большую нагрузочную способность. К числу недостатков отнесем большую потребляемую мощность и малую разность уровней выходных сигналов. На элементах ЭСЛ серий К137 и К187 построены цифровые ЭВМ ЕС-1050 и ЕС—1060.

Создание элементов ИИЛ явилось результатом дальнейшего развития и совершенствования элементов ТЛНС. Элементы ИИЛ представляют собой безрезисторные схемы на биполярных транзисторах с инжекционным питанием. Основным компонентом схемы (рис. 2.25, а) является транзистор $T2$ типа $n-p-n$, который может иметь несколько

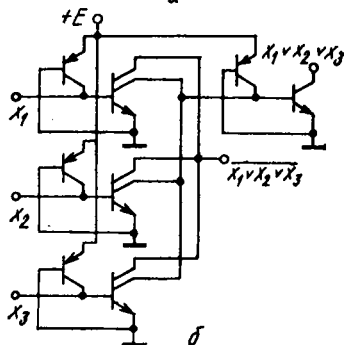
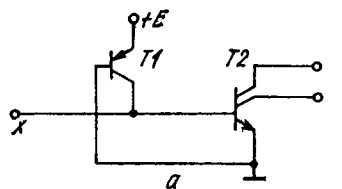


Рис. 2.25

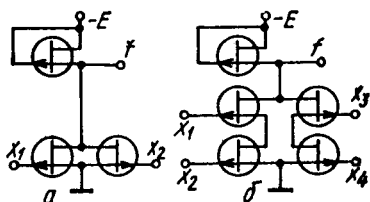


Рис. 2.26

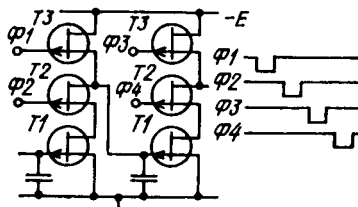


Рис. 2.27

коллекторов (многоколлекторный транзистор) и работает как инвертор. Транзистор $T1$ типа $p-n-p$ предназначен для инжекционного питания инвертора и работает в качестве нагрузочного элемента и генератора тока базы многоколлекторного транзистора. Переключение такой схемы основано на изменении входным сигналом x направления и величины тока в базовой цепи $T2$. Если $x = В$, то ток из коллектора $T1$ течет в базу $T2$, вызывая смещение перехода база — эмиттер и перехода коллектор — база этого транзистора в прямом направлении. Транзистор $T2$ при этом насыщается, а напряжение между его коллектором и базой не превышает несколько десятков милливольт. Если же $x = Н$, то ток инжекции прекращается, так как база $T2$ оказывается под потенциалом коллектора. При этом в базу $T2$ перестает течь ток, что приводит к увеличению выходного сопротивления транзистора, попадающего на границу активного режима и отсечки. Напряжение на коллекторе $T2$ в этом случае составляет 0,6—0,7 В. Таким образом, схема на рис. 2.25, а реализует функцию отрицания. Схема элемента ЗИЛИ—НЕ/ЗИЛИ (СВП) показана на рис. 2.25, б.

Схемы элементов ИИЛ не содержат резисторов, занимающих боль-

шую площадь на подложке и рассеивающих большую мощность. Уменьшению площади, занимаемой элементами ИИЛ на подложке, способствует также объединение базовой и эмиттерной областей *n-p-n* транзистора соответственно с коллекторной и базовой областями *p-n-p* транзистора. Из-за такой физической интеграции весь инвертор ИИЛ занимает площадь одного многоэмиттерного транзистора в элементах ТТЛ. ИИЛ является перспективной схмотехнической основой построения больших интегральных схем, содержащих десятки и сотни тысяч элементов. Некоторые параметры лучших из известных элементов ИИЛ приведены в табл. 31.

Интегральные элементы МОПТЛ являются одними из наиболее технологичных изделий микроэлектроники. Это обусловлено, в первую очередь, тем, что для МОПТЛ требуется выполнять одну технологическую операцию диффузии, в то время как для схем на биполярных транзисторах необходимо выполнять три таких операции. Для схем МОПТЛ уменьшается и количество других технологических операций (окисления, маскирования и т. п.). Простота производства схем МОПТЛ обеспечивает большой процент выхода годных изделий. Плотность размещения МОП-транзисторов на подложке выше по сравнению с биполярными транзисторами. Этому способствует и использование МОП-транзисторов в качестве нагрузочных резисторов. В отличие от биполярных схем в схемах МОПТЛ не требуется никаких изолирующих зон, так как компоненты изолируются полупроводниковым материалом за счет его низкой естественной проводимости. Мощность, потребляемая МОП-транзисторами, значительно меньше, чем мощность, потребляемая биполярными транзисторами. Эти факторы позволяют создать схемы МОПТЛ, содержащие десятки и сотни тысяч компонент на одном кристалле.

Вместе с тем схемам МОПТЛ присущ и ряд недостатков. Главный из них состоит в сравнительно невысоком быстродействии таких схем. Это связано с тем, что заряд схемных емкостей происходит через большие сопротивления нагрузочных транзисторов и это удлиняет время переходных процессов. Быстродействие *n*-канальных схем МОПТЛ несколько выше быстродействия *p*-канальных, однако ниже быстродействия схем на биполярных транзисторах. Схемы логических элементов ИЛИ—НЕ (СНП) и И—ИЛИ—НЕ (СНП) статического типа на *p*-канальных МОП-транзисторах приведены на рис. 2.26. Значения типовых параметров элементов лучших серий МОПТЛ приведены в табл. 31. К МОПТЛ, например, относятся элементы серий К120, К164, К172, К176, К764.

В динамических запоминающих элементах МОПТЛ (рис. 2.27) вместо нагрузки используется МОП-транзистор, работающий в ключевом режиме и включаемый в определенные моменты времени тактирующими сигналами $\Phi 1$ — $\Phi 4$. В такой схеме по сигналу $\Phi 1$ ($\Phi 3$) заряжается паразитная емкость затвора транзистора $T1$, а по $\Phi 2$ ($\Phi 4$) эта емкость разряжается или остается заряженной в зависимости от того, открыт или закрыт транзистор $T1$ (т. е. в зависимости от сигнала X). Сигналы $\Phi 1$ — $\Phi 4$ должны быть сдвинуты на $\frac{1}{4}$ периода. Логические значения напряжения на емкости затвора сохраняются

только в пределах периода тактирующих сигналов, который не должен быть слишком большим из-за разряда емкости токами утечки. По этой причине динамические элементы МОПТЛ не могут работать на низких частотах. Быстродействие динамических элементов МОПТЛ выше быстродействия статических, так как перезаряд паразитных емкостей происходит через открытые МОП-транзисторы.

Основу логических элементов МОПДТЛ составляет инвертор на дополняющих МОП-транзисторах. Затворы канальных транзисторов n - и p -типа включены параллельно, а сами транзисторы последовательно. Поэтому, когда один из них включен, то второй всегда выключен. В статическом режиме ток через такую схему определяется током утечки и составляет всего несколько наноампер. В режиме переключения ток достигает нескольких микроампер и мощность, потребляемая схемой, возрастает, однако рост мощности ограничивается тем, что n - и p -канальные транзисторы одновременно не открываются полностью.

При построении многовыходных логических элементов каждый вход подключают к соединенным вместе затворам пары n - и p -транзисторов. Комбинируя параллельным и последовательным включением ключевых и нагрузочных транзисторов в таких парах можно реализовать различные функции. На рис. 2.28 показаны схемы элементов ИЛИ—НЕ и И—НЕ МОПДТЛ. Параметры элементов МОПДТЛ лучших серий указаны в табл. 31.

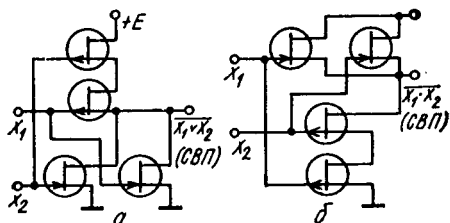


Рис. 2.28

Заметим, что особенности технологии МОПДТЛ позволяют реализовать схемы, работоспособность которых сохраняется при очень широком диапазоне изменения питающих напряжений. Такие схемы могут работать даже при напряжении, меньшем суммы пороговых напряжений переключения ключевых и нагрузочных транзисторов, тогда как для схем МОПТЛ напряжение питания должно не меньше, чем в пять раз превышать пороговое напряжение. Быстродействие схем МОПДТЛ выше быстродействия статических схем МОПТЛ, поскольку перезаряд емкости затвора, длительность которого определяет время переключения схемы, всегда происходит через открытый транзистор. К числу недостатков элементов МОПДТЛ следует отнести то, что технология их изготовления значительно сложнее технологии элементов МОПТЛ, так как требует выполнения трех операций диффузии. Поэтому плотность размещения компонент в таких схемах примерно в три раза меньше, чем в схемах на МОП-транзисторах, хотя и больше, чем у многих эквивалентных по быстродействию схем на биполярных транзисторах.

Развитие интегральной технологии производства элементов цифровых ЭВМ сопровождалось увеличением (в среднем в два раза за год) числа радиокомпонент в пределах одного интегрального модуля, т. е. ростом степени интеграции. Это привело к созданию нового

класса функциональных изделий микроэлектроники — больших интегральных схем (БИС), которые становятся основной элементной базой цифровых ЭВМ четвертого поколения. Появление БИС оказывает существенное влияние не только на принципы конструирования средств ЦВТ, но и на функциональную организацию их структур, а также на программное обеспечение. В процессе производства БИС используются все рассмотренные выше способы реализации элементов на уровне электрических схем. Достигнутая к настоящему времени степень интеграции позволяет производить модули БИС, соответствующие различным иерархическим уровням аппаратных средств ЦВТ, т. е. БИС-элементы, БИС-операционные узлы, БИС-операционные блоки и т. д. Структуры таких БИС будут рассмотрены в следующих главах.

В заключение отметим, что отечественной промышленностью освоен выпуск нескольких десятков серий интегральных микросхем элементов, причем число типов микросхем в каждой серии непрерывно увеличивается за счет включения новых модулей повышенной степени интеграции. Более подробные сведения об интегральных микросхемах, необходимые для построения принципиальных схем средств ЦВТ и изготовления другой конструкторской документации, приведены в соответствующих справочниках.

2.9. ТРИГГЕРЫ В СХЕМАХ ЦВТ

В операционных схемах ЦВТ в качестве запоминающих элементов используются главным образом триггерные схемы (триггеры) нескольких типов. Триггер содержит схему, которая может находиться в двух устойчивых состояниях (бистабильную схему) и схему управления (рис. 2.29, а). Схема управления преобразует входные сигналы таким образом, чтобы выполнялся заданный закон переключения триггера в целом. Для обеспечения надежного переключения в схемах некоторых триггеров в точках А должны быть включены элементы задержки. Однако при построении триггеров на потенциальных элементах не могут быть применены элементы задержки, содержащие реактивные компоненты и нарушающие принцип прямых гальванических связей. Поэтому для задержки сигналов x_1 и x_2 используют, например, запоминающие элементы на основе бистабильных схем (БС) того же типа, что уже имеются в триггере. Структурная схема такого триггера приведена на рис. 2.29, б. Следует заметить, что в некоторых конкретных схемах триггеров функции управляющей схемы могут быть реализованы логическими элементами, входящими в состав бистабильных схем. В триггере на рис. 2.29, б значения сигналов x_1 и x_2 запоминаются дополнительной БС, а затем по внешнему сигналу С передаются на основную БС. Поскольку состояния дополнительной БС повторяют состояния основной БС, то первую БС называют также ведомой, а вторую ведущей, а схему триггера в соответствии с рис. 2.29, б — схемой типа М—S (от слов master и slave).

Разновидности триггеров, применяемых в ЦВТ, отличаются законом функционирования, схемной реализацией, способом записи информации в триггер и т. п. Это вызывает необходимость специальной

классификации триггеров. В настоящее время принято разделять триггеры на классы по виду функции переходов триггеров $Q^{(s+1)} = \lambda(Q^{(s)}, X^{(s)})$ (функциональное разделение) и по способу записи информации в триггеры. При функциональном разделении различают триггеры D , T , RS , E , S , JK , DV , RST и других типов. По способу же записи информации триггеры делят на асинхронные и синхронные (тактируемые). Кроме того, оба этих класса триггеров разделяют на два подкласса: триггеры с внутренней задержкой и триггеры, переключающиеся по уровню синхронизирующих сигналов. Триггеры первого подкласса переключаются после окончания входного сигнала (по заднему фронту), триггеры второго подкласса переключаются одновременно с поступлением входного сигнала. В зависимости от

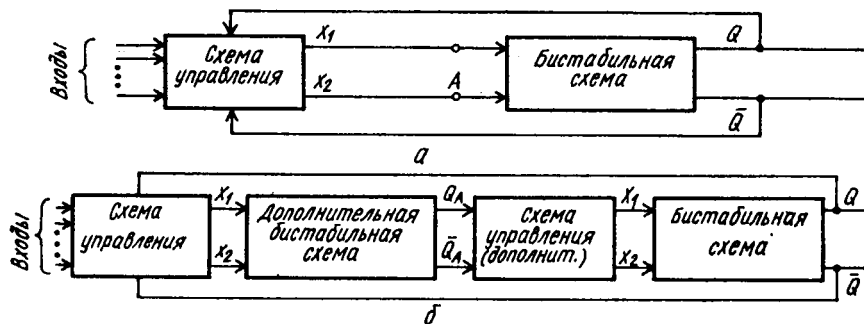


Рис. 2.29

числа серий синхронизирующих сигналов, используемых в триггерах, переключающихся по уровню, для формирования состояния $Q^{(s+1)}$ такие триггеры делят на одноктактные и многотактные. Синхронные триггеры с внутренней задержкой являются, преимущественно, одноктактными.

В основу построения триггеров положено свойство логических схем с обратными связями (схем с петлями, их определение приведено в п. 2.4), заключающееся в том, что состояния выходов схемы зависят не только от значений входных сигналов в данный момент времени, но и от сигналов, действовавших в предыдущие моменты времени. Говорят, что схема с обратными связями находится в устойчивом состоянии, если состояние ее выходов может сохраняться неограниченно долго. Неустойчивым состоянием схемы будет такое, которое существует лишь короткое время, соизмеримое с длительностью переходных процессов в схеме. Наличие в схеме двух и более устойчивых состояний указывает на то, что схема может быть использована для запоминания некоторых сигналов, поступающих на схему по внешним цепям.

В качестве примера рассмотрим схему, построенную из элементов ИЛИ—НЕ (рис. 2.30). Нетрудно определить, что выходная переменная z удовлетворяет следующему булевому уравнению

$$\overline{\overline{(x \vee z)} \vee (\overline{y \vee \bar{z}})} = \bar{x} \bar{z} \vee \bar{y} z = z, \quad (2.25)$$

Рис. 2.30

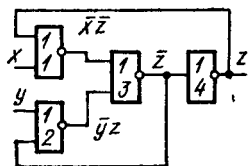


Рис. 2.30

Следовательно, логическая схема с обратными связями в зависимости от комбинации входных сигналов может быть конечным автоматом или вообще будет неправильно функционировать (выдавать нестандартный выходной сигнал, либо генерировать колебания).

Однако схемы с обратными связями, имеющими много входов и выходов, анализировать подобным образом трудно, так как таблицы соответствия в форме табл. 32 становятся очень громоздкими. В таком случае используют другую форму таблицы соответствия — так называемую карту Карно. Рассмотрим методику анализа схем с применением карт Карно на примере двух схем с обратными связями, получивших очень широкое распространение в ЦВТ. Первая из них представлена на рис. 2.31, а и описывается уравнениями

Таблица 32

x	y	z	$\overline{xz} \vee \overline{yz}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$\begin{aligned} y_1 &= \overline{x_1 y_2} = \overline{x_1} \vee \overline{y_2}; \\ y_2 &= \overline{x_2 y_1} = \overline{x_2} \vee \overline{y_1}. \end{aligned}$$

Для решения этой системы уравнений составим карту Карно (табл. 33), столбцы которой обозначим всевозможными комбинациями

независимых переменных x_1 и x_2 , а строки — комбинациями зависимых переменных y_1 и y_2 . В клетки этой карты запишем значения функций y_1 и y_2 , определенные в соответствии с приведенной системой уравнений.

Очевидно, что состояние схемы является устойчивым, если значения функций y_1 и y_2 совпадают с обозначением соответствующей строки таблицы (такие состояния в табл. 33 обведены жирной линией). Например, на пересечении столбца 01 и строки 10 находится устойчивое состояние 10, а на пересечении того же столбца и строки 00 — неустойчивое состояние 11. Иногда вышеприведенную таблицу представляют в другой форме (табл. 34) и называют таблицей переходов. Здесь кружками обозначены устойчивые состояния, точками — неустойчивые, а стрелками указывают направления переходов в устойчивые состояния.

Рассмотрим подробнее, как осуществляется переход схемы из неустойчивого состояния в устойчивое. Возможны при этом три случая: 1) код неустойчивого состояния на карте Карно совпадает с кодом устойчивого; 2) код неустойчивого состояния не совпадает с кодом устойчивого; 3) переход из одного состояния в другое вызван разбросом времени переключения элементов (т. е. вызван гонками).

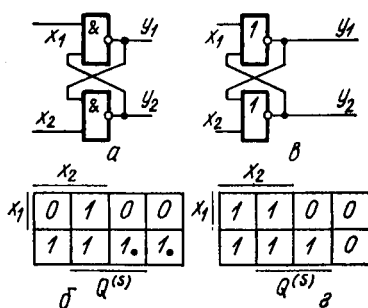
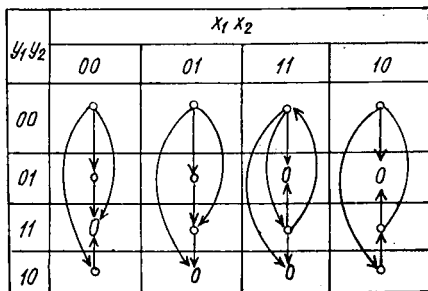


Рис. 2.31

Таблица 33

$y_1 y_2$	$x_1 x_2$			
	00	01	11	10
00	11	11	11	11
01	11	11	01	01
11	11	10	00	01
10	11	10	10	11

Таблица 34



В первом случае, при фиксированных значениях независимых переменных x_1 и x_2 выходные сигналы y_1 и y_2 , соответствующие неустойчивому состоянию, подаются на входы y_1 и y_2 схемы, тем самым обуславливая переход к строке карты Карно, соответствующей устойчивому состоянию. Например, пересечение столбца 00 и строки 01 соответствует неустойчивому состоянию 11. Однако при подаче $y_1 = 1$ и $y_2 = 1$ и при прежних значениях x_1 и x_2 схема переходит в устойчивое состояние 11.

Во втором случае при фиксированных x_1 и x_2 выходные сигналы y_1 и y_2 обуславливают переход к новой строке карты Карно, где

эти же значения y_1 и y_2 являются входными до тех пор, пока не возникнет ситуация, предусмотренная первым случаем.

В третьем случае вследствие конечности и разброса времени переключения элементов при переходе схемы из одного состояния в другое могут появляться промежуточные наборы значений зависимых переменных. Например, если исходным состоянием является состояние 01, а конечным (т. е. находящимся в клетке карты Карно) — состояние 10, то промежуточными состояниями будут 00 и 11. Заметим, что для рассматриваемых карт Карно строки, соответствующие промежуточным состояниям, всегда будут соседними для строк, соответствующих исходному состоянию. Для определения всех возможных переходов, вызванных гонками, необходимо определить промежуточные состояния между состоянием, обозначающим строку карты Карно, и неустойчивым состоянием, находящимся в этой строке. Затем в строки карты, обозначенные промежуточными состояниями, необходимо провести стрелки переходов.

Из табл. 34 видно, что в схеме при $x_1 = 1$ и $x_2 = 1$ может начаться циклический процесс перехода из состояния 00 в 11 и наоборот, т. е. могут возникнуть колебания. Очевидно, что такое явление возможно лишь в случае, когда оба элемента характеризуются абсолютно одинаковым временем переключения и, кроме того, сигналы $x_1 = 1$ и $x_2 = 1$ устанавливаются строго одновременно. В рассматриваемом столбце из каждого неустойчивого состояния возможен переход в любое из устойчивых состояний. В других же столбцах возможны несколько путей перехода из неустойчивого состояния в устойчивое. Первое явление свидетельствует о наличии в схеме так называемых критических гонок, второе — о наличии некритических гонок. Для того, чтобы рассматриваемую схему можно было использовать для запоминания сигналов, поступающих по входам x_1 и x_2 , необходимо обеспечить три режима ее работы: установку в состояние, соответствующее цифре 0; установку в состояние, соответствующее цифре 1; режим хранения, при котором входные сигналы не изменяют состояния схемы. Необходимо также исключить возможность возникновения критических гонок и «неудобное» устойчивое состояние 11 (исключить первый столбец карты Карно). Состояние 11 неудобно тем, что, во-первых, после изменения независимых входных переменных от значений 00 к значениям 11 схема может перейти как в состояние 01, так и в состояние 10 (иначе говоря, переход будет неопределенным) и, во-вторых, состояние 11 не сохраняется при других значениях независимых входных переменных. Исключить первый столбец карты Карно можно, наложив ограничения на допустимые комбинации входных сигналов, а именно: $x_1 \vee x_2 = 1$. Критические гонки исключаются, если разрешенными комбинациями входных сигналов, производящими установку схемы в состояние 0 и 1, будут комбинации 01 и 10. В этом случае при подаче сигналов $x_1 = x_2 = 1$ схема будет сохранять то устойчивое состояние, которое установилось предыдущей разрешенной комбинацией входных сигналов.

До сих пор процессы в схеме анализировались при фиксированных x_1 и x_2 . Рассмотрим теперь поведение схемы при изменении независимых входных переменных. Поскольку состояние 11 схемы исключено

в силу условия $x_1 \vee x_2 = 1$, то обозначим для удобства записи $Q^{(s)} = 0$ — состояние 01 схемы в s -й момент времени, $Q^{(s)} = 1$ — состояние 10. Тогда зависимость состояния $Q^{(s+1)}$ от $Q^{(s)}$, x_1 и x_2 можно представить в табл. 35. Для определения закона функционирования схемы по отношению к переменным $Q^{(s)}$, x_1 и x_2 составим диаграмму Вейча функции $Q^{(s+1)}$, соответствующую табл. 35. Неопределенные значения $Q^{(s+1)}$ отметим на этой диаграмме точками (рис. 2.31, б). Доопределим функцию $Q^{(s+1)}$ на тех наборах, где она неопределена, с тем, чтобы найти ее минимальную форму. Очевидно, что доопределенной диаграмме Вейча будет соответствовать следующая минимальная форма

$$Q^{(s+1)} = \bar{x}_1 \vee Q^{(s)} x_2.$$

Эту функцию называют функцией переходов БС на элементах И—НЕ, а совместно с условием $x_1 \vee x_2 = 1$ называют характеристическими уравнениями БС. Схема с обратными связями на элементах ИЛИ—НЕ (рис. 2.31, в) описывается уравнениями

$$y_1 = \overline{x_1 \vee y_2} = \bar{x}_1 \bar{y}_2; \quad y_2 = \overline{x_2 \vee y_1} = \bar{x}_2 \bar{y}_1.$$

Составим карту Карно и таблицу переходов этой схемы (табл. 36, 37). Из таблиц видно, что в этом случае критические гонки могут начаться при $x_1 = x_2 = 0$, а для бистабильности схемы надо исклю-

Таблица 35

x_1	x_2	$Q^{(s)}$	$Q^{(s+1)}$
0	0	0	—
0	0	1	—
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Таблица 36

$y_1 y_2$	$x_1 x_2$			
	00	01	11	10
00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

Таблица 37

$y_1 y_2$	$x_1 x_2$			
	00	01	11	10
00				
01				
11				
10				

чить столбец карты Карно с $x_1 = x_2 = 1$, т. е. «неудобное» состояние 00, что достигается при условии $x_1 x_2 = 0$. Функцию переходов этой схемы можно легко определить, воспользовавшись табл. 38 и диаграммой Вейча на рис. 2.31, г. Следовательно, характеристические уравнения БС на элементах ИЛИ—НЕ будут следующие:

$$Q^{(s+1)} = x_2 \vee \bar{x}_1 Q^{(s)}; \quad x_1 x_2 = 0.$$

Таким образом, характеристические уравнения выражают зависимость состояния БС в $s+1$ -й момент времени от разрешенных комбинаций входных сигналов и состояния в s -й момент времени.

Решения характеристических уравнений относительно x_1 и x_2 при фиксированных $Q^{(s+1)}$ и $Q^{(s)}$ определяют значения входных сигналов, переключающих БС из состояния $Q^{(s)}$ в состояние $Q^{(s+1)}$, т. е. определяют функции возбуждения БС (см. п. 2.6). Найти решения характеристических уравнений можно, например, путем составления таблиц значений их левых и правых частей для всех возможных значений x_1 и x_2 . Другой способ решения состоит в следующем. Пусть для БС на элементах И—НЕ (сокращенно: БС (И—НЕ)) $Q^{(s+1)} = 0$ и $Q^{(s)} = 0$. Тогда

$$0 = \bar{x}_1 \vee x_2 0; \quad x_1 \vee x_2 = 1.$$

Из второго уравнения следует, что решениями могут быть только следующие наборы значений x_1 и x_2 : 01, 10 и 11. Однако первый на-

Таблица 38

x_1	x_2	$Q^{(s)}$	$Q^{(s+1)}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	—
1	1	1	—

Таблица 39

$Q^{(s)}$	$Q^{(s+1)}$	БС (И—НЕ)		БС (ИЛИ—НЕ)	
		x_1	x_2	x_1	x_2
0	0	1	—	—	0
0	1	0	1	0	1
1	0	1	0	1	0
1	1	—	1	0	—

бор не удовлетворяет первому уравнению, поэтому $x_1 = 1$, а x_2 при этом может быть равным как 0, так и 1 (т. е. x_2 — неопределено). Аналогичные рассуждения можно провести и для других наборов $Q^{(s+1)}$ и $Q^{(s)}$.

Так как при фиксированных $Q^{(s+1)}$ и $Q^{(s)}$ возможны всего 4 типа характеристических уравнений, то в дальнейшем для определения сигналов x_1 и x_2 будем пользоваться таблицей решений характеристических уравнений (табл. 39), где прочерки соответствуют неопределенным значениям. Из этой таблицы видно, что

$$x_1 \text{ (или — не)} = \bar{x}_2 \text{ (и — не)}; \quad x_2 \text{ (или — не)} = \bar{x}_1 \text{ (и — не)}.$$

Для построения триггеров из рассмотренных БС следует так определить функции управляющей схемы, чтобы переключение БС происходило в соответствии с таблицей функции $\lambda(Q^{(s)}, X^{(s)})$ триггера. С этой целью необходимо найти совместное решение (относительно x_1 и x_2) характеристических уравнений БС и уравнения $Q^{(s+1)} = \lambda(Q^{(s)}, x^{(s)})$. Один из способов решения такой системы уравнений состоит в подстановке в характеристические уравнения БС значений $Q^{(s+1)}$ и $Q^{(s)}$ из таблицы переходов триггера и применении таблицы решений характеристических уравнений. Рассмотрим особенности и схемные варианты построения триггеров различных типов.

D — триггер (триггер задержки) представляет собой триггер с одним информационным входом D , состояние которого в $s+1$ -й мо-

мент времени повторяет значение входного сигнала D в s -й момент, т. е.

$$Q^{(s+1)} = D^{(s)}.$$

Название триггера происходит от слова delay — задержка. Практический интерес представляют лишь синхронные D -триггеры, в которых прием информации осуществляется по внешнему сигналу C . В таблицу переходов такого триггера (табл. 40) включены также

Таблица 40

C	$D^{(s)}$	$Q^{(s)}$	$Q^{(s+1)}$	БС (И—НЕ)		БС (ИЛИ—НЕ)	
				x_1	x_2	x_1	x_2
0	0	0	0	1	—	—	0
0	0	1	1	—	1	0	—
0	1	0	0	1	—	—	0
0	1	1	1	—	1	0	—
1	0	0	0	1	—	—	0
1	0	1	0	1	0	1	0
1	1	0	1	0	1	0	1
1	1	1	1	—	1	0	—

столбцы функций x_1 и x_2 для соответствующих БС, значения которых определены с помощью таблицы решений характеристических уравнений. Минимальные формы функций x_1 и x_2 от аргументов C , $D^{(s)}$ и $Q^{(s)}$ найдем, составив диаграммы Вейча этих функций и оптимальным образом доопределив их на неопределенных наборах. При этом для БС (И—НЕ)

$$x_1 = \overline{CD}^{(s)}, \quad x_2 = \overline{CD}^{(s)} = \overline{CD}^{(s)} \vee \overline{CC} = \overline{C} (\overline{D}^{(s)} \vee \overline{C}) = \overline{Cx_1},$$

а для БС (ИЛИ—НЕ)

$$x_1 = \overline{C} \vee \overline{D}^{(s)}, \quad x_2 = \overline{CD}^{(s)} \vee \overline{CC} = \overline{C} \vee (\overline{D}^{(s)} \vee \overline{C}) = \overline{C} \vee x_1$$

Этим выражениям соответствуют схемы D -триггеров на рис. 2.32, которые переключаются по уровню синхронизирующих сигналов. В эти триггеры при $C = 1$ записывается та информация, которая была на входе D до прихода синхронизирующего сигнала. Для устойчивой работы триггера необходимо, чтобы сигнал D не менялся во время действия сигнала C . В одном из схемных вариантов D -триггера — так называемом «триггере—защелке» (рис. 2.33, а) — прием информации производится по сигналу $C = 0$. Пусть $Q^{(s)} = 0$, $D = 1$ и на вход C подан нулевой сигнал. Тогда через время, равное сумме задержек сигнала на элементах 1, 2, 4, на выходе $\overline{Q}$ появится нулевой сиг-

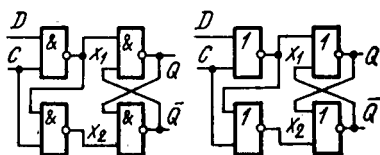


Рис. 2.32

нал, а еще через время задержки сигналов инвертором 5 на выходе Q появится единичный сигнал, чем и завершится установка триггера в единичное состояние. После окончания сигнала $C = 0$ (т. е. при $C = 1$) состояние триггера не изменится, поскольку будет открыт элемент 3, который открывается раньше, чем закрывается элемент 2. Для установки триггера в нулевое состояние необходимо подать сигналы $D = 0$ и $C = 0$, закрывающие элементы 2 и 3. На выходе $\bar{Q}$ формируется единичный сигнал, а на выходе Q — нулевой. При $C = 1$ состояние триггера сохраняется, так как элемент 2 закрыт сигналом $D = 0$, а элемент 3 — сигналом $Q = 0$.

Однофазный D -триггер отличается от рассмотренных наличием только одного выхода Q . Проще всего такие триггеры реализуются на основе элементов с инверсными входами (рис. 2.33, б). Для устой-

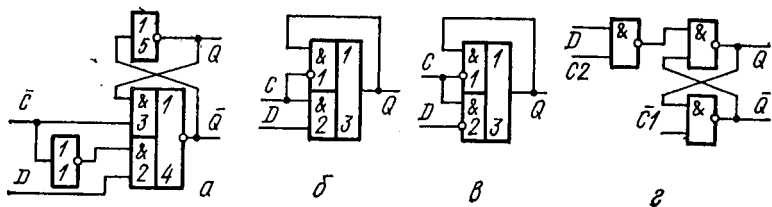


Рис. 2.33

чивой работы рассматриваемого однофазного триггера длительность сигнала D не должна превышать длительность сигнала C . Преимуществом однофазных триггеров по сравнению с другими D -триггерами является их схемная простота. Известны также однофазные инверсные D -триггеры ($\bar{D}$ -триггеры), функция переходов которых имеет вид $Q^{(s+1)} = \bar{D}^{(s)}$ (рис. 2.33, в).

В схемах ЦВТ используются также двутактные D -триггеры (рис. 2.33, г), переключение которых происходит за два такта синхронизирующих сигналов $C1$ и $C2$, единичные значения которых не пересекаются во времени. По сигналу $C1 = 1$ производится установка БС в состояние 0, а по сигналу $C2 = 1$ осуществляется запись информации со входа D . Последовательное соединение двух таких триггеров образует четырехтактный D -триггер, для работы которого необходимы четыре серии внешних синхронизирующих сигналов $C1, C2, C3, C4$, не пересекающихся во времени.

D -триггеры с внутренней задержкой строят, в основном, по схеме М—S и по схеме трех триггеров, причем каждая из этих схем имеет несколько разновидностей. На рис. 2.34, а показано построение D -триггера по схеме М—S с запрещающими связями. При $C = 0$ состояние триггера не меняется, при $C = 1$ и $D = 1$ происходит установка в 1 основной БС и одновременно блокируется дополнительная БС нулевым сигналом с выхода элемента 1. Если $C = 1$ и $D = 0$, блокирующий сигнал формируется элементом 2, а основная БС устанавливается в 0. Когда $C = 0$, то элементы 1 и 2 формируют единичные сигналы и дополнительная БС принимает состояние основной БС.

D -триггер, выполненный по схеме трех триггеров, приведен на рис. 2.34, б. Здесь элементы 5 и 6 образуют основную БС, а элементы

1, 2 и 3, 4 — две коммутирующих БС. Будем считать, что основная БС находится в единичном состоянии, т. е. $Q = 1$, $\bar{Q} = 0$, а сигналы D и C равны нулю. В этом случае на выходе элемента 2 нулевой сигнал, на выходе элемента 1 — единичный, на выходе элемента 4 — нулевой и на выходе элемента 3 — единичный сигналы. Поэтому обе коммутирующие БС находятся в единичном состоянии. При подаче сигнала $C = 1$ на выходе элемента 3 появится нулевой сигнал и нижняя коммутирующая БС (элементы 3 и 4) установится в нуль. Единичный сигнал с выхода элемента 4 поступает на вход элемента 1, формируя на его выходе нулевой сигнал. Однако верхняя коммутирующая БС (элементы 1 и 2) в нулевое состояние не переключается, так как на выходе элемента 2 также нулевой сигнал, обусловленный тем, что $C = 1$. Таким образом, при $C = 1$ на выходах основной БС действуют нулевые сигналы, не вызывающие ее переключений. Когда $C = 0$, на выходе элемента 2 появится единичный сигнал, вслед за которым основная и верхняя коммутирующая БС установятся в 0. Как видно, установка основной БС в нулевое состояние задерживается по отношению к моменту подачи сигнала $C = 1$ на длительность этого сигнала.

Пусть, далее $D = 1$ и $C = 0$. Тогда на выходе элемента 4 появится нулевой сигнал, однако нижняя коммутационная БС при этом в единичное состояние не переключается, так как на выходе элемента 3 нулевой сигнал удерживается единичным сигналом с выхода элемента 2. При поступлении сигнала $C = 1$ верхняя коммутационная БС устанавливается в единичное состояние, а сигналы на выходах остальных элементов не изменяются. Когда сигнал C станет равным нулю, на входах элемента 3 будут действовать нулевые сигналы, что приведет к установке нижней коммутирующей и основной БС в единичное состояние. Как и ранее, переключение основной БС будет запаздывать на длительность сигнала $C = 1$. Если теперь снова поступит сигнал $C = 1$, то состояние верхней коммутирующей и основной БС не изменится.

Триггер со счетным входом, или T -триггер (от слова toggle — кувыркаться), также как и D -триггер, является триггером с одним информационным входом T . Таблица переходов такого триггера (табл. 41) содержит столбцы для функций x_1 и x_2 БС, определенные с помощью таблицы решений характеристических уравнений БС. Функция переходов такого триггера имеет вид

$$Q^{(s+1)} = Q^{(s)}\bar{T}^{(s)} \vee \bar{Q}^{(s)}T^{(s)}.$$

Такой триггер изменяет свое состояние на противоположное каждый раз при подаче единичного входного сигнала. Легко заметить,

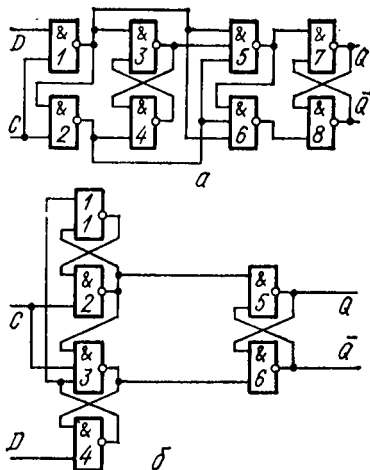


Рис. 2.34

Таблица 41

$T^{(s)}$	$Q^{(s)}$	$Q^{(s+1)}$	БС (И—НЕ)		БС (ИЛИ—НЕ)	
			x_1	x_2	x_1	x_2
0	0	0	1	—	—	0
0	1	1	—	1	0	—
1	0	1	0	1	0	1
1	1	0	1	0	1	0

что если отождествить входной сигнал T с одной переключательной переменной, а состояние триггера в s -й момент времени — с другой переменной, то функция $Q^{(s+1)}$ будет функцией суммы по mod 2 количества входных сигналов. Этим и объясняется другое название этого триггера — триггер со счетным входом. Из табл. 41 следует

$$x_1 = \overline{Q^{(s)}}T^{(s)}, \quad x_2 = \overline{Q^{(s)}}\overline{T^{(s)}} \quad \text{для БС (И—НЕ);}$$

$$x_1 = \overline{Q^{(s)}} \vee \overline{T^{(s)}}, \quad x_2 = Q^{(s)} \vee \overline{T^{(s)}} \quad \text{для БС (ИЛИ—НЕ).}$$

Этим выражениям соответствуют структурные схемы T -триггеров на рис. 2.35. Как видно из этих схем, сигналы $Q^{(s)}$ и $\overline{Q^{(s)}}$ используют-

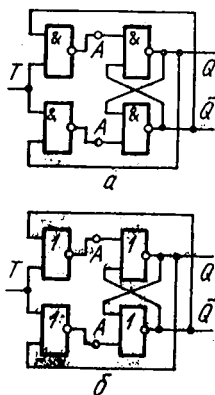


Рис. 2.35

ся для формирования значений функций возбуждения БС, т. е. для перевода триггера в новое состояние. Так как сигнал $T^{(s)}$ имеет конечную длительность, то это может привести к неопределенной работе триггера, а именно: под воздействием одного и того же сигнала $T^{(s)}$ схема может многократно переключаться из состояния 01 в 10 и наоборот. Отсюда следует практически важный вывод о том, что в схемах триггеров, где сигналы $Q^{(s)}$ и $\overline{Q^{(s)}}$ используются для формирования функций возбуждения БС, сигналы записи информации (сигналы x_1 и x_2) и сигналы съема информации (выходные сигналы Q и $\overline{Q}$) не должны пересекаться во времени. Этого можно достичь, включив в точках A (рис. 2.35) элементы временной задержки сигналов (в импульсно-потенциальных системах элементов) или же вводя в указанные места промежуточные запоминающие элементы (рис. 2.29, б). В связи с этим асинхронный T -триггер, а также одноктактный синхронный T -триггер, переключаемые по уровню сигналов, реализуемы только на импульсно-потенциальных элементах.

Синхронный T -триггер (иногда называемый TV -триггером) отличается от рассмотренных наличием дополнительного входа C (V). Если $C = 1$, то такой триггер переключается как T -триггер, если же $C = 0$, то триггер сохраняет свое предыдущее состояние.

Двухтактные T -триггеры, переключаемые по уровню, могут быть построены по аналогии со схемой на рис. 2.36, а. Синхронные и асинхронные T -триггеры с внутренней задержкой, выполненные по схеме

$M-S$, могут быть легко получены из рассматриваемого двухтактного триггера. Для этого достаточно сформировать сигналы $C2$ путем инверсии сигналов $C1$ — для синхронного триггера (показано штриховой линией на рис. 2.36, а) или сигналов T — для асинхронного триггера. Еще один вариант асинхронного T -триггера с внутренней задержкой приведен на рис. 2.36, б. Особенностью такой схемы является наличие запрещающих связей, блокирующих передачу информации с основной БС на вспомогательную БС при $T = 1$. При $T = 0$ запрет на передачу снимается и состояние основной БС дублируется состоянием вспомогательной БС.

T -триггеры с внутренней задержкой, построенные по схеме трех триггеров (рис. 2.36, в) переключаются следующим образом. Пусть основная БС находится в нулевом состоянии и $T = 0$. Тогда и верхняя коммутационная БС находится в нулевом состоянии, а на выходах элементов 3 и 4 будут нулевые сигналы. На выходах элементов 2 и 3 таким образом будет сформирована комбинация 10, подтверждающая состояние основной БС. При $T = 1$ на выходах этих же элементов появятся нулевые сигналы, не изменяющие состояния основной БС, а верхняя коммутирующая БС установится в 1. Когда входной сигнал опять станет равным 0, нижняя коммутирующая БС установится в 1, что повлечет за собой и переключение основной БС в 1. На выходах элементов 1 и 2 при этом установятся нулевые сигналы. Следующий сигнал $T = 1$ установит нижнюю коммутирующую БС в 0, а на

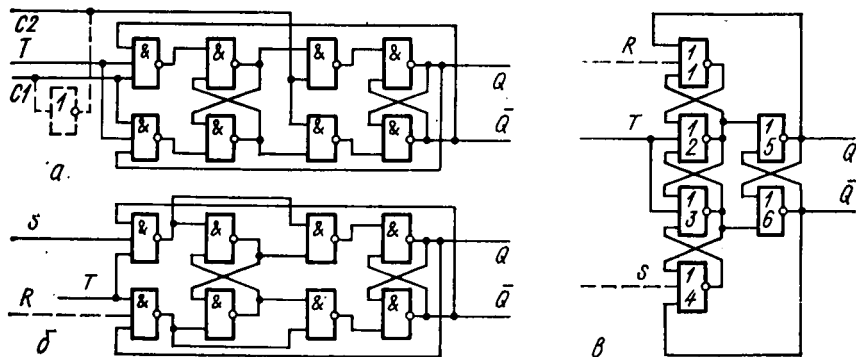


Рис. 2.36

выходах элементов в верхней коммутирующей БС будут нулевые сигналы и т. д. Как видно, T -триггер, построенный по схеме трех триггеров, содержит на 2 логических элемента И—НЕ (ИЛИ—НЕ) меньше, чем T -триггер, построенный по схеме $M-S$.

Триггер RS -типа (от слов reset и set — сбрасывать и устанавливать) или триггер с установочными (раздельными) входами описываются табл. 42. Сравнивая эту таблицу с таблицей переходов БС (ИЛИ—НЕ) легко убедиться, что они совпадают, если отождествить R с x_1 и S с x_2 . Следовательно, при таком отождествлении будут идентичны и схемы, реализующие эти таблицы. При использовании элементов И—НЕ

$$x_{1(\text{И—НЕ})} = \bar{x}_{2(\text{ИЛИ—НЕ})} = \bar{S}; \quad x_{2(\text{И—НЕ})} = \bar{x}_{1(\text{ИЛИ—НЕ})} = \bar{R},$$

Таблица 42

$R^{(s)}$	$S^{(s)}$	$Q^{(s)}$	$Q^{(s+1)}$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	—
1	1	1	—

т. е. БС (И—НЕ) будет функционировать как RS -триггер по отношению к инверсиям сигналов R и S . Функция переходов RS -триггера имеет вид

$$Q^{(s+1)} = S^{(s)} \vee Q^{(s)} \bar{R}^{(s)}.$$

При этом должно выполняться условие $R^{(s)} S^{(s)} = 0$. Как и БС, RS -триггер находится в единичном состоянии, если на его выходе Q присутствует единичный сигнал, а на выходе $\bar{Q}$ — нулевой сигнал. Триггер находится в нулевом состоянии, если на выходе

Q — нулевой сигнал, а на выходе $\bar{Q}$ — единичный. С учетом способа отображения букв двузначного алфавита вышеизложенное можно проиллюстрировать табл. 43.

Работа RS -триггера в тех случаях, когда $R = S = 1$, не определена. Поэтому в схемах ЦВТ надо принимать меры к недопущению таких состояний на входах триггера. Реализация асинхронных RS -триггеров не отличается от реализации бистабильных схем. Синхронные RS -триггеры, в отличие от асинхронных, имеют на каждом установочном входе дополнительные элементы (рис. 2.37, а), первые входы которых объединены и образуют шину для син-

Таблица 43

Состояние триггера	Выходы триггера					
	СВП		СНП		Логические значения	
	$\bar{Q}$	Q	$\bar{Q}$	Q	$\bar{Q}$	Q
0	В	Н	Н	В	1	0
1	Н	В	В	Н	0	1

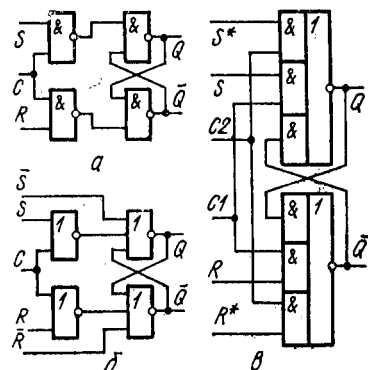


Рис. 2.37

хронизирующих сигналов C . Информация, поданная на входы R и S , записывается на триггер только при $C = 1$. Синхронные триггеры могут иметь также входы для асинхронной установки (рис. 2.37, б). Двухтактные триггеры получают путем раздельной синхронизации входов R и S сигналами $C1$ и $C2$. Еще один вариант синхронного двухтактного триггера (так называемого $R - S - R^* - S^*$ -триггера) показан на рис. 2.37, в. Здесь входы R и S синхронизируются сигналом $C1$, а входы R^* и S^* — сигналом $C2$.

RS -триггеры с внутренней задержкой (по схеме М—S) могут быть построены по аналогии со схемами на рис. 2.38. В схеме на рис. 2.38, а между основной и вспомогательной БС включен инвертор, обеспечивающий однократный режим работы и блокирующий передачу информации на вспомогательную БС при записи ее на основную БС.

В схеме на рис. 2.38, б сигналы, блокирующие передачу информации с основной БС на вспомогательную, формируются элементами, введенными в схему для ее синхронизации. По схеме М—S могут быть построены также двухтактные, трехтактные, четырехтактные RS-триггеры.

В RS-триггере, выполненном по схеме трех триггеров (рис. 2.38, в) при $C = 0$ основная и коммутирующие БС находятся в одинаковых состояниях. Например, состояния БС нулевые и $S = 1, R = 0$. Тогда при $C = 1$ верхняя коммутирующая БС переключится в единичное состояние, а на выходах нижней БС будут нулевые сигналы. Поэтому на входах основной БС действуют нулевые сигналы, не изменяющие

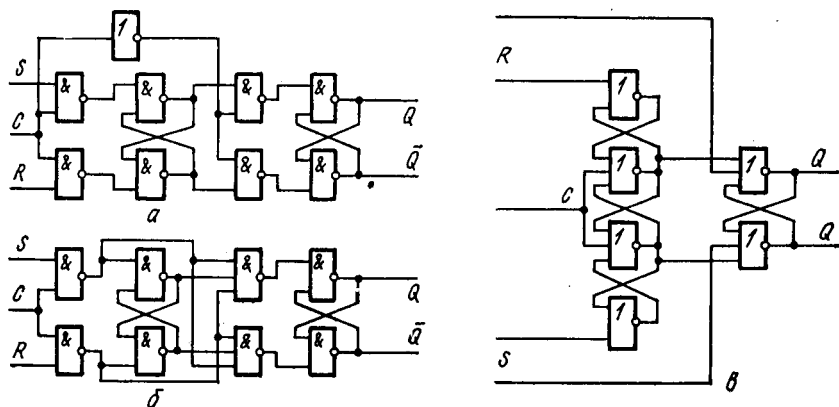


Рис. 2.38

ее состояния. Когда C станет равным 0, нижняя коммутирующая БС переключится в 1, что приведет к установке в 1 и основной БС. Переключение триггера из 1 в 0 происходит аналогично. Если $R = S = C = 0$, то состояния коммутирующих БС не изменяются. Если же $R = S = 0, C = 1$, то не меняется состояние одной из коммутирующих БС, которая при $C = 0$ восстанавливает исходное состояние другой БС.

Примером триггера с тремя входами является RST-триггер (триггер с установочными и счетным входами), функционирование которого описывается табл. 44. Сигнал $T = 1$ переводит RST-триггер в противоположное состояние, сигнал $R = 1$ устанавливает триггер в состояние $Q = 0$, а сигнал $S = 1$ — в состояние $Q = 1$. Подача единичного сигнала на два или три входа запрещена, так как состояние $Q^{(s+1)}$ триггера в этом случае не определено. Функция переходов RST-триггера имеет вид

$$Q^{(s+1)} = S^{(s)} \vee T^{(s)} \bar{Q}^{(s)} \vee \bar{T}^{(s)} \bar{R}^{(s)} Q^{(s)},$$

а выражение, ограничивающее допустимые комбинации входных сигналов, можно записать как

$$T^{(s)} R^{(s)} \vee T^{(s)} S^{(s)} \vee R^{(s)} S^{(s)} = 0.$$

Таблица 44

$T(s)$	$R(s)$	$S(s)$	$Q(s)$	$Q(s+1)$	БС (И—НЕ)		БС (ИЛИ—НЕ)	
					x_1	x_2	x_1	x_2
0	0	0	0	0	1	—	—	0
0	0	0	1	1	—	1	0	—
0	0	1	0	1	0	1	0	1
0	0	1	1	1	—	1	0	—
0	1	0	0	0	1	—	—	0
0	1	0	1	0	1	0	1	0
0	1	1	0	—	—	—	—	—
0	1	1	1	—	—	—	—	—
1	0	0	0	1	0	1	0	1
1	0	0	1	0	1	0	1	0
1	0	1	0	—	—	—	—	—
1	0	1	1	—	—	—	—	—
1	1	0	0	—	—	—	—	—
1	1	0	1	—	—	—	—	—
1	1	1	0	—	—	—	—	—
1	1	1	1	—	—	—	—	—

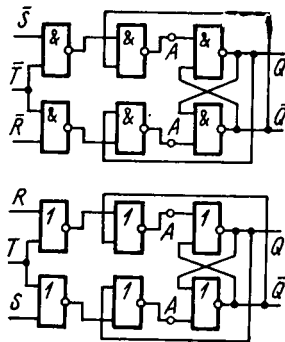


Рис. 2.39

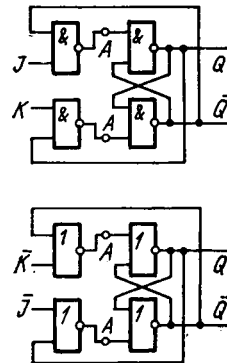


Рис. 2.40

Функции возбуждения БС, определенные в соответствии с табл. 44, имеют вид

$$\left. \begin{aligned} x_1 &= \overline{Q^{(s)} \vee T^{(s)} \bar{S}^{(s)}} = \overline{\bar{Q}^{(s)} (\bar{T}^{(s)} \bar{S}^{(s)})}; \\ x_2 &= \overline{Q^{(s)} \vee T^{(s)} \bar{R}^{(s)}} = \overline{Q^{(s)} (\bar{T}^{(s)} \bar{R}^{(s)})}, \end{aligned} \right\} \text{ для БС (И—НЕ)}$$

$$\left. \begin{aligned} x_1 &= \overline{Q^{(s)} (\bar{T}^{(s)} \vee \bar{R}^{(s)})} = \overline{\bar{Q}^{(s)} \vee (\bar{T}^{(s)} \vee \bar{R}^{(s)})}; \\ x_2 &= \overline{Q^{(s)} (\bar{T}^{(s)} \vee \bar{S}^{(s)})} = \overline{Q^{(s)} \vee (\bar{T}^{(s)} \vee \bar{S}^{(s)})}. \end{aligned} \right\} \text{ для БС (ИЛИ—НЕ)}$$

Этим функциям соответствуют структурные схемы триггеров на рис. 2.39. Для обеспечения устойчивой работы RST -триггеров, как и T -триггеров, в точках A необходимо включить элементы временной задержки сигналов. Практические схемы RST -триггеров совпадают

со схемами T -триггеров, у которых имеются дополнительные установочные входы (показаны штриховой линией на рис. 2.36).

JK -триггеры, как и RS -триггеры, являются триггерами с двумя входами. Но в отличие от RS -триггеров они не имеют запрещенных комбинаций входных сигналов. Кроме того, на их основе могут быть построены все другие типы триггеров. По этой причине JK -триггеры называют также универсальными. Функционирование JK -триггера описывается табл. 45, а его функция переходов имеет вид

$$Q^{(s+1)} = Q^{(s)} \bar{K}^{(s)} \vee J^{(s)} \bar{Q}^{(s)}.$$

Как видно, при $J^{(s)} = K^{(s)} = 1$ состояние $J^{(k)}$ -триггера изменяется на противоположное. Минимальными формами функций возбуждения БС будут следующие:

$$\left. \begin{aligned} x_1 &= \overline{J^{(s)} \bar{Q}^{(s)}}; \\ x_2 &= \overline{K^{(s)} Q^{(s)}} \end{aligned} \right\} \text{ для БС (И — НЕ)}, \quad \left. \begin{aligned} x_1 &= \overline{\bar{K}^{(s)} \vee \bar{Q}^{(s)}}; \\ x_2 &= \overline{\bar{J}^{(s)} \vee Q^{(s)}} \end{aligned} \right\} \text{ для БС (ИЛИ — НЕ)}.$$

Соответствующие этим функциям структурные схемы триггеров приведены на рис. 2.40. Реализация JK -триггеров на потенциальных элементах может быть выполнена по любой из схем T -триггера, у которого вход T разделен на два входа J и K .

Таблица 45

$J^{(s)}$	$K^{(s)}$	$Q^{(s)}$	$Q^{(s+1)}$	И — НЕ		ИЛИ — НЕ	
				x_1	x_2	x_1	x_2
0	0	0	0	1	—	—	0
0	0	1	1	—	1	0	—
0	1	0	0	1	—	—	0
0	1	1	0	1	0	1	0
1	0	0	1	0	1	0	1
1	0	1	1	—	1	0	—
1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	0

DV -триггер (в некоторых литературных источниках DF -триггер) также является универсальным триггером с двумя входами. Когда $V = 1$, то такой триггер работает как D -триггер. Если же $V = 0$, то триггер сохраняет свое предыдущее состояние. Табл. 46 является таблицей переходов такого триггера и значений функций возбуждения БС. На ее основе получим функцию переходов DV -триггера

$$Q^{(s+1)} = D^{(s)} V^{(s)} \vee \bar{V}^{(s)} Q^{(s)},$$

а также функции возбуждения БС

$$\left. \begin{aligned} x_1 &= \overline{D^{(s)} V^{(s)}}; \\ x_2 &= \overline{\bar{D}^{(s)} V^{(s)}} \end{aligned} \right\} \text{ для БС (И — НЕ)}, \quad \left. \begin{aligned} x_1 &= \overline{\bar{D}^{(s)} \vee \bar{V}^{(s)}}; \\ x_2 &= \overline{\bar{D}^{(s)} \vee V^{(s)}} \end{aligned} \right\} \text{ для БС (ИЛИ — НЕ)}$$

Таблица 46

$D(s)$	$V(s)$	$Q(s)$	$Q(s+1)$	И-НЕ		ИЛИ-НЕ	
				x_1	x_2	x_1	x_2
0	0	0	0	1	—	—	0
0	0	1	1	—	1	0	—
0	1	0	0	1	—	—	0
0	1	1	0	1	0	1	0
1	0	0	0	1	—	—	0
1	0	1	1	—	1	0	—
1	1	0	1	0	1	0	1
1	1	1	1	—	1	0	—

Легко заметить, что если в этих выражениях заменить $V(s)$ на C , то получим функции возбуждения синхронного D -триггера. Поэтому DV -триггер реализуется на основе D -триггера, у которого параллельно входу синхронизации C включен информационный вход V . Для устойчивой работы DV -триггера необходимо, чтобы сигнал $V = 1$ по длительности был не меньше синхронизирующего сигнала $C = 1$.

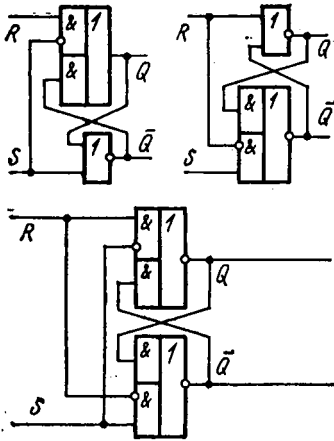


Рис. 2.41

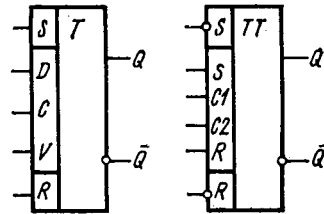


Рис. 2.42

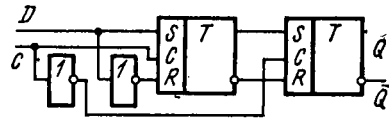


Рис. 2.43

Триггеры типов S , R и E являются триггерами с двумя входами S и R . Если $SR = 0$, то такие триггеры переключаются как RS -триггеры. При $S = R = 1$ S -триггеры устанавливаются в 1, R -триггеры — в 0, E -триггеры сохраняют предыдущее состояние. Такие триггеры могут быть легко построены на основе RS -триггеров (рис. 2.41).

На функциональных схемах узлов ЦВТ триггеры всех типов принято обозначать так, как показано на рис. 2.42. Здесь в левой части условного обозначения триггера указывают вход C синхронизирующий

щих сигналов и функциональные назначения других входов. В правой части условного обозначения ставят идентификатор триггера — букву T (или TT , если триггер построен по схеме на рис. 2.29, б). В некоторых случаях левую часть разделяют на синхронную и асинхронную части. В асинхронной части указывают символы S и R асинхронной установки триггера, а в синхронной ставят буквы, указывающие тип триггера. При этом кружок на каком-либо входе триггера означает, что прием информации происходит при инверсном значении соответствующего сигнала.

Рассмотрим теперь, как можно, имея один тип триггера, построить на его основе другой. В общем случае это можно сделать так же, как ранее строились различные триггеры на основе БС, т. е. необходимо найти решение системы уравнений, полученных из функций перехо-

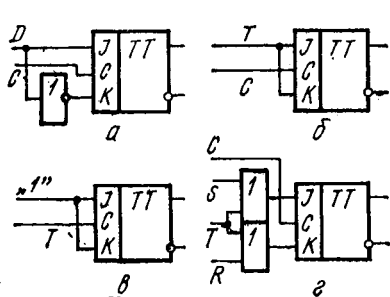


Рис. 2.44

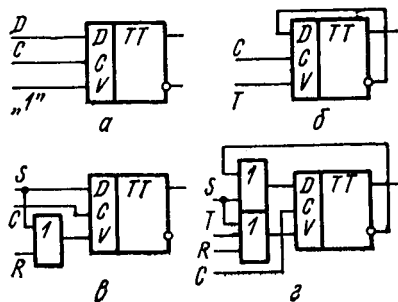


Рис. 2.45

дов проектируемого и заданного триггеров, а затем в соответствии с полученным решением реализовать функции возбуждения известного триггера. Для наиболее же важных практических случаев приведем примеры таких построений.

D -триггер можно построить путем каскадного включения двух синхронных RS -триггеров (рис. 2.43). При этом передача информации из одного триггера в другой происходит после окончания приема информации в первый триггер. Как видно из схемы, информационный сигнал $D = 1$, записанный в левый триггер сигналом $C = 1$, на выход схемы не пройдет, так как передача блокируется инвертором. Только после окончания данного такта, когда $C = 0$, информация может быть снята с выхода правого триггера. JK -триггер будет работать как D -триггер, если на его вход J подать сигнал D , а на вход K — сигнал $\bar{D}$ (рис. 2.44, а). При объединении входов J и K JK -триггер может работать как синхронный (рис. 2.44, б) или асинхронный (рис. 2.44, в) T -триггер. RS -триггер получают из JK -триггера простым наложением ограничения на комбинации входных сигналов, а именно $JK = 0$. RST -триггер может быть построен из JK -триггера по схеме на рис. 2.44, г.

Способы использования DV -триггеров для построения других типов триггеров (T , D , RS , RST) показаны на рис. 2.45.

Выше рассмотрены законы функционирования и принципы построения некоторых триггеров на основе БС из элементов И—НЕ и ИЛИ—НЕ. При использовании БС, построенных из других элементов (например, И—ИЛИ—НЕ, ИЛИ—И—НЕ и др.), схемная реализация триггеров может отличаться от приведенной. Конфигурация схемы триггера зависит также от способа доопределения функций возбуждения и функций переходов, от числа входов логических элементов и от используемой технически полной системы элементов.

ОПЕРАЦИОННЫЕ УЗЛЫ ЦИФРОВЫХ ЭВМ И ОСНОВЫ ИХ ПРОЕКТИРОВАНИЯ

3.1. ОБЩИЕ ВОПРОСЫ ПОСТРОЕНИЯ ОПЕРАЦИОННЫХ УЗЛОВ ЦИФРОВЫХ ЭВМ

Несмотря на большое разнообразие узлов, из которых составляются цифровые ЭВМ и их устройства, среди них имеется ряд узлов, наиболее часто встречающихся на практике. К таким узлам относятся, например, дешифраторы, преобразователи кодов, схемы сравнения кодов, сдвигатели, регистры, счетчики, сумматоры и др. На долю этих узлов приходится значительная часть оборудования современных цифровых ЭВМ. Поэтому весьма важным является изучение таких методов проектирования операционных узлов, которые обеспечивают достижение оптимальных значений основных технико-экономических показателей — сложности (стоимости) и быстродействия. В такой ситуации необходимы единые критерии оценки сложности и быстродействия вариантов операционных узлов, получаемых при различных требованиях к ним, при использовании различных методов проектирования, принципов представления букв структурного алфавита, различных систем элементов и т. п.

Наиболее полной оценкой L сложности какого-либо цифрового устройства является линейная форма вида

$$L = \sum_{\text{по всем } i} \omega_i L_i,$$

где ω_i — число элементов типа i с индексом сложности L_i , используемых для построения данного устройства. Значение L_i обычно выражают в условных единицах. Например, считают, что индекс сложности дискретного резистора примерно равен индексу сложности дискретного конденсатора и примерно равен 0,2—0,1 индекса сложности дискретного диода. Приблизительно о величине L_i можно судить по стоимости дискретных электронных компонент или элементов. Однако величина L_i сильно зависит от технологических особенностей изготовления элементов. Это обуславливает незначительные отличия в индексах сложности для интегральных элементов одного типа (например, все модули одной и той же серии микросхем можно считать равными по параметру L_i). По этим причинам, а также ввиду невозможности установления величины L_i безотносительно к конкретной серии интегральных элементов, сложность операционных узлов будем оценивать величиной

$$L = \sum_{\text{по всем } i} \omega_i.$$

Заметим, что в операционных узлах отдельные буквы обрабатываемых слов обычно хранятся на триггерах, обеспечивающих получение как прямых, так и инверсных их значений. Поэтому в оценку L

нет необходимости включать элементы НЕ, формирующие инверсии входных сигналов. Однако элементы, используемые для получения инверсий сигналов внутри схемы, следует учитывать.

Планарная технология производства больших интегральных схем предполагает выполнение всех соединений между электронными компонентами и элементами в виде многоуровневой системы плоских связей, напыляемых непосредственно на монокристаллическую кремниевую подложку. На первом уровне этой системы реализуются связи между отдельными компонентами и образуются простейшие логические элементы И—НЕ, ИЛИ—НЕ, И, ИЛИ и т. д. На втором уровне реализуются связи, объединяющие элементы в триггеры и другие разрядные схемы обработки информации. На третьем уровне реализуются связи в многоразрядных схемах и т. д. Поэтому сложность операционного узла, выполненного в виде большой интегральной схемы, можно характеризовать суммарным числом связей всех уровней, которое, в свою очередь, можно считать приблизительно пропорциональным общему числу входов логических элементов, используемых в схеме. Такую оценку сложности, известную в ЦВТ под названием цены по Квайну (см. п. 2.3), обозначим как K и впредь будем использовать наряду с оценкой L .

Способ оценки быстродействия операционных узлов зависит от их типа. Быстродействие комбинационных операционных узлов оценивают, определяя самую длинную цепь логических элементов из всех возможных путей следования информационных сигналов в схеме. Для узлов с памятью (накапливающих узлов) указывают число тактов, необходимых для выполнения заданной операции над машинным словом, а также длительность одного такта, которая должна быть не меньше времени переключения элементов в самой длинной цепи.

При проектировании операционных узлов необходимо учитывать то, что длительность сигнала на выходе логического элемента может отличаться от длительности входного сигнала. Основными причинами этого являются наличие порога срабатывания элементов, насыщение транзисторных элементов, а также различные паразитные эффекты. В качестве примера рассмотрим влияние порога срабатывания элементов на длительность выходного сигнала. При этом возможны два граничных случая, показанных на диаграмме (рис. 3.1). Первый случай соответствует переключению элемента при пороговом напряжении $U_1 \rightarrow H$. Длительность выходного сигнала в этом случае превышает длительность входного сигнала на время, приблизительно равное длительности фронта входного сигнала t_ϕ . Второй случай соответствует переключению элемента при пороговом напряжении $U_2 \rightarrow B$. При этом выходной сигнал приблизительно на t_ϕ короче входного. Поэтому последовательная цепь из n элементов, переключающихся по уровню U_1 , будет удлинять выходной сигнал на nt_ϕ . Частота входных сигналов для такой цепи не должна превышать величину

$$f = \frac{1}{t_0 + nt_\phi},$$

где t_0 — длительность входного сигнала. В противном случае два соседних сигнала, поступивших на вход схемы, не будут разделены вре-

менным интервалом на ее выходе. Если же последовательная цепь будет составлена из элементов, переключающихся по уровню U_2 , то сигнал на выходе n -го элемента будет иметь длительность $t_0 - nt_\Phi$. Следовательно, для такой схемы входной сигнал должен иметь длительность не меньше, чем

$$t_0 = t_{0\text{кр}} + nt_\Phi,$$

где $t_{0\text{кр}}$ — некоторая критическая длительность входного сигнала для данной схемы. В противном случае входной сигнал не достигнет выхода схемы.

В схемах ЦВТ часто возникают ситуации, когда сигналы, действующие на входах какого-либо элемента, имеют так называемую разность хода, т. е. проходят по цепям, состоящим из неодинакового числа элементов. Пример такой ситуации показан на рис. 3.2. Без

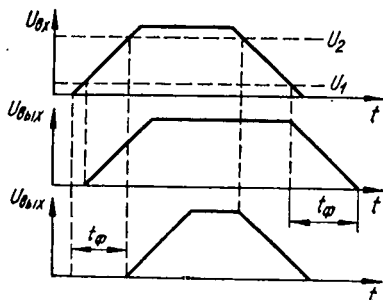


Рис. 3.1

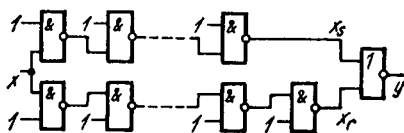


Рис. 3.2

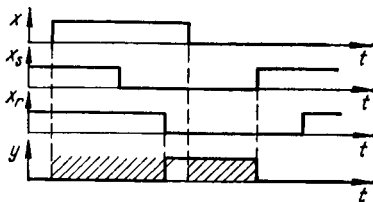


Рис. 3.3

учета задержек, вносимых элементами, сигнал на выходе этой схемы должен быть равен значению входного сигнала (считаем, что число элементов в нижней r и верхней s ветвях схемы нечетное). Однако в реальной схеме указанное равенство установится лишь через некоторое время, обусловленное конечностью задержек элементов. Таким образом, в течение этого времени на выходе схемы будет ложный сигнал, хотя на ее входе действует истинный сигнал (рис. 3.3, ложный сигнал заштрихован). При достаточно большой разности хода $r - s$ длительность ложных сигналов может во много раз превышать время переключения элементов, а их амплитуда равна номинальной амплитуде сигналов для данного типа элементов. На эти ложные сигналы могут реагировать другие элементы в схеме, они могут вызвать неправильное переключение триггеров, образовывать запрещенные кодовые комбинации и др. Явление возникновения ложных сигналов при взаимодействии сигналов, имеющих разность хода, как и в схемах с обратными связями, называют гонками.

Основная трудность в устранении гонок заключается в том, что заранее неизвестны истинные задержки сигналов в каждой цепи. Для элементов какой-либо серии в нормативно-технической документации указывают лишь максимальную задержку, задержки же конкрет-

ных элементов не превышают указанное значение. Поэтому не исключены ситуации, когда цепь из меньшего числа элементов будет вносить большую задержку, чем цепь из большего числа однотипных элементов. Подбор элементов по задержке практически неосуществим, в особенности при серийном производстве цифровых схем высокой степени интеграции.

Для предупреждения возможности возникновения гонок применяя различные структурные способы, среди которых одним из самых простых является синхронизация. В этом случае на тот логический элемент, где взаимодействуют сигналы, имеющие разность хода, подают внешний синхронизирующий сигнал, задержанный на время, большее максимально возможной задержки в самой длинной цепи. Все переходные процессы в схеме будут происходить при запертом выходе и ложные сигналы не появятся. Синхронизирующий сигнал должен заканчиваться не позже окончания входных сигналов. Сущность других способов устранения гонок состоит во введении структурной избыточности и некоторых эквивалентных преобразованиях систем переключательных функций, описывающих схему.

Проектирование операционных узлов цифровых ЭВМ можно разделить на два этапа — этапы абстрактного и структурного синтеза. Исходными данными для выполнения этапа абстрактного синтеза служат результаты операционного проектирования цифровой схемы, при котором определяются все реализуемые на ней микрооперации. Целью этапа абстрактного синтеза является определение закона функционирования проектируемого узла в терминах теории автоматов и теории переключательных функций, т. е. получение таблиц переходов и таблиц истинности. На этапе структурного синтеза выполняется кодирование этих таблиц буквами двужначного структурного алфавита, синтез и оптимизация функциональной схемы узла и оценка ее качества.

Следует отметить, что по отношению к операционным узлам цифровых ЭВМ этап абстрактного синтеза во многих случаях выполняется очень просто, а способ кодирования таблиц переходов и таблиц истинности буквами двужначного алфавита известен заранее, например, обусловлен используемой системой счисления, формой представления чисел, способом представления отрицательных чисел, двоично-десятичными кодами и т. п. Поэтому в дальнейшем процедуры составления кодированных таблиц часто будем опускать и сразу переходить от операционных описаний узлов цифровых ЭВМ к их структурной реализации.

3.2. ДЕШИФРАТОРЫ

Полным дешифратором будем называть комбинационную схему, имеющую n входов и 2^n выходов, которая реализует 2^n функций

$$D_i(X) = \begin{cases} 1 & \text{при } X = j; \\ 0 & \text{при } X \neq j; \end{cases}$$

где $j = 0, 1, \dots, 2^n - 1$; $X = \sum_{i=1}^n x_i 2^{i-1}$; $x_i \in \{0, 1\}$.

Таким образом, дешифратор преобразует n -разрядный двузначный код с естественным порядком весов в 2^n -значный пространственный код, так как каждому слову на входе дешифратора (всего 2^n слов) соответствует единичный сигнал только по одной из выходных шин.

Каждая из функций $D_i(X)$ может быть записана в следующем виде:

$$\begin{aligned} D_0(X) &= \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_3 \bar{x}_2 \bar{x}_1; \\ D_1(X) &= \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_3 \bar{x}_2 x_1; \\ D_2(X) &= \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_3 x_2 \bar{x}_1; \\ &\vdots \\ D_{2n-2}(X) &= x_n x_{n-1} \dots x_3 x_2 \bar{x}_1; \\ D_{2n-1}(X) &= x_n x_{n-1} \dots x_3 x_2 x_1. \end{aligned}$$

Следовательно, система функций $D_i(X)$ является совокупностью всех возможных конститuent единицы, которые можно образовать от n переменных. Известно два основных способа реализации системы функций $D_i(X)$: линейный и каскадный.

При линейном способе построения дешифратора каждая из функций $D_j(X)$ реализуется отдельно на n -входовом элементе типа И. На входы элементов подаются все возможные комбинации прямых и инверсных значений разрядов дешифрируемого слова X . (Как правило, в цифровых ЭВМ машинные слова представлены прямыми и инверсными значениями букв. Поэтому получение отрицаний входных переменных не сопряжено с дополнительными аппаратными затратами). Например, при $n = 3$ схема линейного дешифратора имеет вид, показанный на рис. 3.4, а. На рис. 3.4, б приведено условное обозначение дешифратора на функциональных схемах. Если для построения дешифратора используются элементы И—НЕ и ИЛИ—НЕ, то каждая из функций $D_j(X)$ должна быть реализована по таким операторным выражениям (см. п. 2.4):

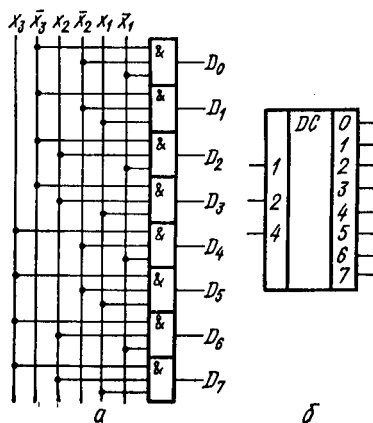


Рис. 3.4

$$D_j(X) = \overline{\tilde{x}_n \tilde{x}_{n-1} \dots \tilde{x}_2 \tilde{x}_1};$$

где $\tilde{x}_i$ означает, что в качестве переменной в соответствующем выражении может выступать либо прямое, либо инверсное значение x_i .

Нетрудно убедиться что для линейного дешифратора наилучшими оценками сложности и быстродействия будут следующие:

$$L = 2^n; K = n2^n; t = t_3.$$

Однако, если число разрядов машинного слова больше числа входов логических элементов, то элементы И в линейном дешифраторе должны представлять собой каскадные соединения элементов И с меньшим числом входов. В таких случаях рациональнее применять другие способы построения дешифраторов.

При каскадном построении дешифратора дешифрируемое слово разбивается на несколько подслов, представляющих участки данного слова. Число таких подслов может находиться в диапазоне от 2 до $\left\lfloor \frac{n}{2} \right\rfloor$. Для каждого подслова формируются все его конstituенты еди-

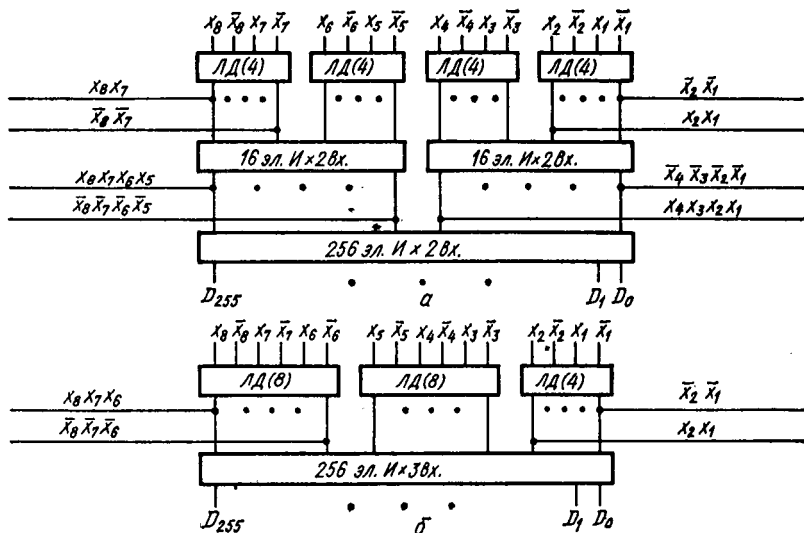


Рис. 3.5

ницы путем построения отдельных линейных дешифраторов. Все дешифраторы подслов составляют первый каскад дешифратора. В каждом последующем каскаде образуют конъюнкции конstituент из предыдущего каскада. Пусть, например, $n = 8$ и для построения дешифратора должны использоваться только двухвходовые элементы. Тогда структурная схема дешифратора имеет вид, показанный на рис. 3.5, а. Здесь $ЛД(4)$ матричные дешифраторы на 4 выхода. Если число входов логических элементов равно 3, то этот же дешифратор может быть построен по схеме на рис. 3.5, б.

В зависимости от n и числа входов логических элементов M каскадный дешифратор можно выполнять в различных вариантах, отличающихся числом каскадов, числом линейных дешифраторов и т. п. Поэтому оценим сложность только так называемого прямоугольного каскадного дешифратора с минимальным (равным 2) числом каскадов, быстродействие которого наибольшее. Пусть каждое подслово содержит m букв. Будем также считать, что $\frac{n}{m}$ — целое. Тогда ли-

нейные дешифраторы будут содержать $\frac{n}{m} 2^m$ элементов И, а во втором каскаде содержится 2^n элементов. Итого

$$L = \frac{n}{m} 2^m + 2^n.$$

Оценка K для рассматриваемого дешифратора имеет вид

$$K = \frac{n}{m} 2^m m + 2^n \frac{n}{m} = n 2^m + \frac{n}{m} 2^n = n \left(2^m + \frac{2^n}{m} \right).$$

Взяв производную $\frac{dK}{dm}$ и приравняв ее нулю, можно убедиться, что оптимальное m , при котором K минимально, равно $\frac{n}{2}$. В этом случае

$$L = 2^{0,5n} 2 + 2^n = 2^{0,5n+1} + 2^n;$$

$$K = n \left(2^{0,5n} + \frac{2^{n/2}}{n} \right) = 2^{0,5n} n + 2^{n+1}.$$

Из последних формул видно, что при $n > 3$ у каскадного прямоугольного дешифратора число элементов на $2^{0,5n+1}$ больше, чем у линейного, но общее число их входов меньше, причем с возрастанием n эта разность увеличивается. Поэтому при больших n целесообразнее строить каскадные дешифраторы, хотя они по быстродействию и по общему числу логических элементов уступают линейным.

Каскадный способ построения дешифраторов является наиболее распространенным в современной ЦВТ. Поэтому особый интерес представляют способы проектирования каскадных дешифраторов, обеспечивающие их минимальную сложность и максимальное быстродействие. Для этого при наличии элементов с $M < n$ входами входные переменные разбиваются на M

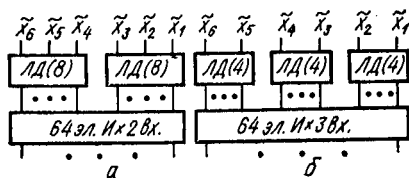


Рис. 3.6

групп по $\left\lfloor \frac{n}{m} \right\rfloor$ или $\left\lceil \frac{n}{m} \right\rceil - 1$ перемен-

ных; если в какой-либо группе переменных будет больше, чем M , то они в свою очередь, разбиваются на M групп с равными или отличающимися на 1 количествами переменных. Такое разбиение продолжают до тех пор, пока не будут получены группы с M или меньшим числом переменных. На всех каскадах дешифратора, начиная с последнего (выходного), входные переменные необходимо группировать так, чтобы полностью использовались входы логических элементов. Неиспользуемые входы элементов допускаются только на первом каскаде (естественно, их должно быть как можно меньше). Например, при $n = 6$, $M = 3$, дешифратор по схеме на рис. 3.6, а характеризуется $L = 80$. Однако такой дешифратор не является оптимальным, так как у элементов второго каскада используются лишь два входа. Вместе с тем дешифратор на рис. 3.6, б, у которого входные переменные разбиты на три группы, при том же быстродействии содержит на 4 элемента меньше.

В общем случае быстродействие оптимального дешифратора оценивается как

$$t = \lg_{\mathcal{M}} n [t_s.$$

Точную же оценку сложности оптимального дешифратора в общем случае дать трудно. Однако при конкретных значениях n и M это можно сделать, воспользовавшись формулой

$$L(n, M) = 2^n + \sum_{i=1}^M L(n_i, M),$$

где n_i — целые числа, удовлетворяющие условию $\sum_{i=1}^M n_i = n$. Сумма $\sum_{i=1}^M L(n_i, M)$ принимает наименьшее значение, когда n_i принимают ближайшие к $\frac{n}{M}$ значения, т. е. $n_i \approx \frac{n}{M}$. Тогда

$$L(n, M) = 2^n + M 2^{\frac{n}{M}} + M \sum_{i=1}^M L\left(\frac{n}{M^2}, M\right).$$

Отсюда получим

$$L(n, M) \approx 2^n + M 2^{\frac{n}{M}} + M^2 2^{\frac{n}{M^2}} + M^3 2^{\frac{n}{M^3}} + \dots$$

Оценку сложности K оптимального дешифратора можно получить, умножив одно из предыдущих выражений на M .

Пирамидальный дешифратор отличается от каскадного тем, что в его каскадах каждая конституента единицы формируется как конъюнкция одной из конституент предыдущего каскада и одного из разрядов входного слова, еще не использовавшегося для формирования конституент. Принцип построения пирамидального дешифратора иллюстрируется рис. 3.7. Количество каскадов пирамидального дешифратора равно $n - 1$, следовательно $t = t_3 (n - 1)$. Так как первый каскад пирамидального дешифратора содержит 4 элемента, а в каждом последующем каскаде число элементов удваивается, то

$$L = 2^{n+1} - 4 = 4(2^{n-1} - 1);$$

$$K = 8(2^{n-1} - 1).$$

В процессе проектирования выбор того или иного типа дешифратора определяется конкретными требованиями к нему и особенностями элементов, из которых строится дешифратор. В потенциальной системе элементов целесообразно строить линейные или каскадные дешифраторы. Проектировать на потенциальных элементах пирамидальные дешифраторы не имеет смысла, так как они, обладая лишь незначительно худшими показателями по сложности, значительно уступают каскадным дешифраторам по быстродействию. Если линейный дешифратор строится из элементов, у которых $M \geq n$, то он является самым экономичным и самым быстродействующим.

Пирамидальные дешифраторы находят применение в основном в некоторых типах запоминающих устройств (например, на ферритовых сердечниках). При этом для их построения используются диодно-трансформаторные или диодно-конденсаторные вентили (ДТВ или ДКВ) — импульсно-потенциальные элементы И с двумя входами, один из которых импульсный, другой — потенциальный. Структура дешифратора на таких элементах

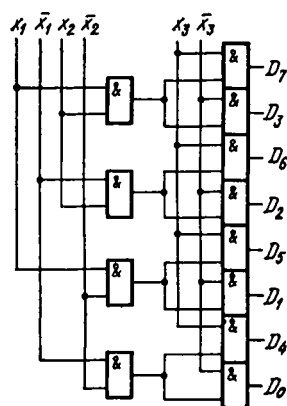


Рис. 3.7

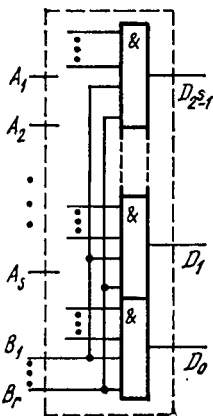


Рис. 3.8

одн из которых импульсный, другой — потенциальный. Структура дешифратора на таких элементах

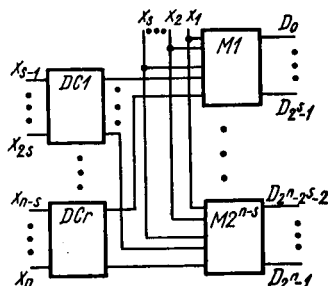


Рис. 3.9

отличается от представленной на рис. 3.7 наличием еще одного каскада из двух элементов, служащего для преобразования потенциального представления переменной x_1 в импульсное.

При разработке интегральных схем для реализации дешифраторов основная трудность состоит в том, что схема дешифратора представляет собой нерегулярную структуру, т. е. такую, способ соединения элементов внутри которой зависит от числа разрядов обрабатываемых слов. Это затрудняет выполнение требования наращиваемости структуры дешифратора. Выполнение этого требования («регуляризация» структуры дешифратора) означает, что дешифратор n_1 -разрядных слов может быть построен на основе модулей-дешифраторов для n_2 -разрядных слов ($n_1 > n_2$) без каких-либо изменений во внутренней структуре последних. При этом внутренняя структура модуля-дешифратора и распределение m его выводов на входы и выходы должны быть такими, чтобы общее число модулей для дешифратора было минимальным. Схема модуля, удовлетворяющего требованию наращиваемости, имеет вид, показанный на рис. 3.8, где входы $A_1, A_2, \dots, A_s$ называют информационными, а входы $B_1, B_2, \dots, B_r$ — управляющими. Такую структуру имеют, например, серийно выпускаемые дешифраторы серии 155 — К155ИДЗ. Значение s в этом случае выбирают максимально возможным при данном m , но таким, чтобы выполнялись условия

$$r = m - s - 2^s; 1 \leq r \leq 2^s + 1.$$

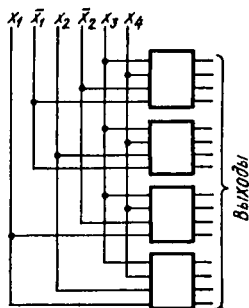


Рис. 3.10

Для реализации дешифратора на n входов ($n > s$) из набора однотипных модулей строится многокаскадная схема. Принцип ее построения поясняется рис. 3.9. К управляющим входам каждого из модулей $M1, M2, \dots, M2^{n-s}$ подключено по одному выходу от каждого дешифратора $DC1, DC2, \dots, DCr$, причем наборы выходов дешифраторов, подключаемых к модулям, не повторяются. Принцип построения дешифраторов $DC1, DC2, \dots, DCr$ аналогичен. Рассмотрим конкретные примеры. Пусть $m = 8$, тогда $s = 2$; $r = 2$. При наличии инверсий входных переменных такой дешифратор можно построить с помощью одного каскада модулей (рис. 3.10). На

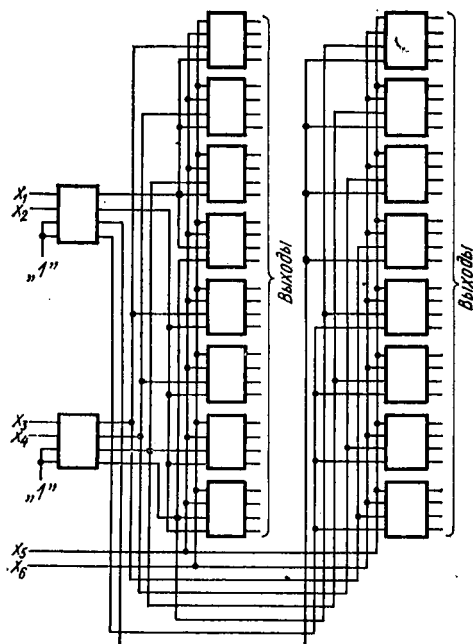


Рис. 3.11

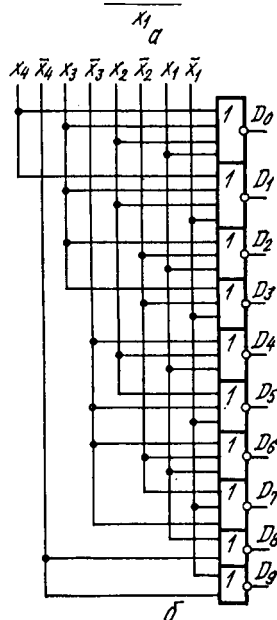
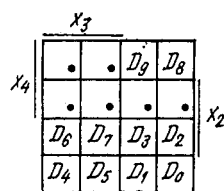


Рис. 3.12

рис. 3.11 показан дешифратор на 6 входов, построенный на модулях с $m = 8$, причем на входы поступают только прямые значения переменных.

Помимо полных дешифраторов на практике встречаются неполные (частичные) дешифраторы, реализующие лишь некоторые из функций $D_i(X)$. Рассмотрим особенности проектирования таких дешифраторов на примере построения дешифратора знакового индикатора, в котором для свечения одного из символов $0, 1, \dots, 9$ необходимо подать единичный сигнал по одной из десяти шин. Будем считать, что индицируемые символы представлены в коде с весами $8, 4, 2, 1$. Следовательно, дешифратор должен реализовать лишь 10 функций $D_0, D_1, \dots, D_9$ четырех аргументов x_4, x_3, x_2, x_1 . При этом 6 наборов таких

аргументов 1010, 1011, 1100, 1101, 1110, 1111 являются избыточными. Это обстоятельство можно использовать для упрощения реализации тех функций $D_j(X)$, которые принимают единичное значение на наборах, соседних с избыточными (рис. 3.12, а, избыточные наборы отмечены точками). Таким образом

$$\begin{aligned} D_0 &= \overline{x_4 \vee x_3 \vee x_2 \vee x_1}; & D_1 &= \overline{x_4 \vee x_3 \vee x_2 \vee \bar{x}_1}; & D_2 &= \overline{x_3 \vee \bar{x}_2 \vee x_1}; \\ D_3 &= \overline{x_3 \vee \bar{x}_2 \vee \bar{x}_1}; & D_4 &= \overline{\bar{x}_3 \vee x_2 \vee x_1}; & D_5 &= \overline{\bar{x}_3 \vee x_2 \vee \bar{x}_1}; \\ D_6 &= \overline{\bar{x}_3 \vee \bar{x}_2 \vee x_1}; & D_7 &= \overline{\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1}; & D_8 &= \overline{\bar{x}_4 \vee x_1}; & D_9 &= \overline{\bar{x}_4 \vee \bar{x}_1}. \end{aligned}$$

Схема неполного дешифратора знакового индикатора приведена на рис. 3.12, б. Легко убедиться, что для такого дешифратора $K = 30$ (без учета соседства заданных наборов с избыточными $K = 40$).

Рассмотренные выше дешифраторы называют также дешифраторами первого рода, так как выходные функции принимают единичное значение только на одном наборе аргументов. В дешифраторах второго рода выходные функции принимают единичное значение на нескольких входных наборах. Построение дешифраторов второго рода сводится к реализации заданной системы переключаемых функций и может быть выполнено известными методами (п. 2.4).

3.3. СХЕМЫ СРАВНЕНИЯ

В цифровых ЭВМ имеется несколько разновидностей операций сравнения слов: сравнение по модулю, сравнение с учетом знаков операндов, сравнение порядков. Здесь рассмотрим такую операцию сравнения, при которой устанавливается факт выполнения одного из условий

$$X = Y; \quad X > Y; \quad X < Y, \quad (3.1)$$

где $X = \sum_{i=1}^n x_i 2^{i-1}$; $Y = \sum_{i=1}^n y_i 2^{i-1}$. Наибольшее распространение получили два способа реализации операции сравнения. Первый способ заключается в том, что из одного числа вычитают второе и по знаку разности судят о выполнении двух последних из указанных выше условий. Выполняется такая операция на сумматоре, к которому дополнительно подключается схема для фиксации нулевого кода разности, который соответствует выполнению первого условия.

С целью сокращения времени выполнения операции сравнения и разгрузки сумматора для выполнения других операций используют другой способ выполнения сравнения. В этом случае строится комбинационная схема сравнения.

Будем называть схемой сравнения слов X и Y схему с $2n + 2$ входами и тремя выходами, реализующую функции

$$\begin{aligned} H_1(X, Y) &= \begin{cases} 1 & \text{при } X = Y; \\ 0 & \text{при } X \neq Y, \end{cases} & H_2(X, Y) &= \begin{cases} 1 & \text{при } X > Y; \\ 0 & \text{при } X \leq Y, \end{cases} \\ H_3(X, Y) &= \begin{cases} 1 & \text{при } X < Y; \\ 0 & \text{при } X \geq Y. \end{cases} \end{aligned}$$

Очевидно, что $H_1(X, Y) = 1$ только в том случае, когда $x_n = y_n$, $x_{n-1} = y_{n-1}$, ..., $x_1 = y_1$. Таким образом, для установления соотношения $X = Y$ необходимо n раз выполнить операцию равнозначности, т. е.

$$H_1(X, Y) = \bigwedge_{i=1}^n (x_i y_i \vee \bar{x}_i \bar{y}_i) = \bigwedge_{i=1}^n (x_i \vee \bar{y}_i) (\bar{x}_i \vee y_i) = \bigwedge_{i=1}^n F_i.$$

Функция $H_2(X, Y) = 1$ в том случае, когда $x_n > y_n$ или же когда $x_n = y_n$, но $x_{n-1} > y_{n-1}$ и т. д. Следовательно,

$$\begin{aligned} H_2(X, Y) &= x_n \bar{y}_n \vee (x_n y_n \vee \bar{x}_n \bar{y}_n) \& x_{n-1} \bar{y}_{n-1} \vee \dots \\ &\dots \vee (x_n y_n \vee \bar{x}_n \bar{y}_n) \dots (x_2 y_2 \vee \bar{x}_2 \bar{y}_2) x_1 \bar{y}_1 = \\ &= x_n \bar{y}_n \vee F_n x_{n-1} \bar{y}_{n-1} \vee \dots \vee F_n F_{n-1} \dots F_2 x_1 \bar{y}_1 = \\ &= x_n \bar{y}_n \vee F_n (x_{n-1} \bar{y}_{n-1} \vee F_{n-1} (\dots \vee F_3 (x_2 \bar{y}_2 \vee F_2 x_1 \bar{y}_1) \dots)). \end{aligned} \quad (3.2)$$

Аналогично рассуждая, можно убедиться, что

$$H_3(X, Y) = (\bar{x}_n y_n \vee F_n (\bar{x}_{n-1} y_{n-1} \vee F_{n-1} (\dots \vee F_3 (\bar{x}_2 y_2 \vee F_2 \bar{x}_1 y_1) \dots))). \quad (3.3)$$

Однако, если схемы для H_1 и H_2 уже построены, то H_3 проще реализовать по формуле

$$H_3 = \bar{H}_1 \bar{H}_2 = \overline{H_1 \vee H_2}. \quad (3.4)$$

Заметим также, что аналогичным способом можно получить значения любой из функций, реализуемых схемой сравнения, при известных двух других функциях, т. е.

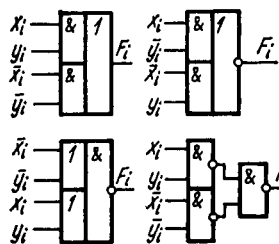


Рис. 3.13

$$\begin{aligned} H_1 &= \bar{H}_2 \bar{H}_3 = \overline{H_2 \vee H_3}; \\ H_2 &= \bar{H}_1 \bar{H}_3 = \overline{H_1 \vee H_3}. \end{aligned}$$

Из приведенных выше выражений видно, что для реализации функций H_1 , H_2 и H_3 необходимо предварительно реализовать функции F_i . Это можно сделать, например, в соответствии с одной из схем на рис. 3.13. На рис. 3.14 представлена схема, реализующая функцию $H_1(X, Y)$ при $n = 4$.

Для такой схемы (при произвольном числе разрядов n)

$$L_1 = 3n + 1; K_1 = 7n; t_1 = 3t_3.$$

Функция H_2 реализуется схемой на рис. 3.15 ($n = 4$), для которой при произвольном n

$$L_2 = 3(n - 1); K_2 = 6n - 5; t_2 = 2nt_3.$$

В оценки L_2 и K_2 не включены элементы, необходимые для построения схем F_i . Если функцию H_3 реализовать по выражению (3.4), то

$$L_3 = 1; K_3 = 2; t_3 = (2n + 1) t_3.$$

Таким образом, для всей схемы сравнения

$$\begin{aligned} L &= L_1 + L_2 + L_3 = 6n - 1; \\ K &= K_1 + K_2 + K_3 = 13n - 3; \\ t &= \max(t_1, t_2, t_3) = (2n + 1)t_3. \end{aligned}$$

При использовании элементов с большим коэффициентом объединения функцию H_2 (или H_3) можно реализовать по выражению, в котором члены F_i не вынесены за скобки. В этом случае $t_2 = 4t_3$. Схема на рис. 3.15 является регулярной и может быть построена на модулях (рис. 3.16), имеющих такую же внутреннюю структуру, как и указанные ранее схемы F_i . Недостаток схемы на рис. 3.16 — рост t пропорционально увеличению n . Поэтому при больших n целесообраз-

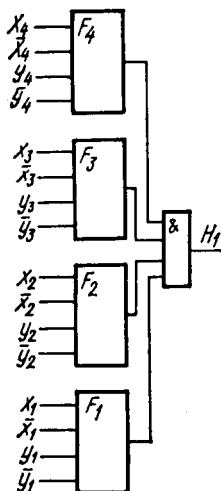


Рис. 3.14

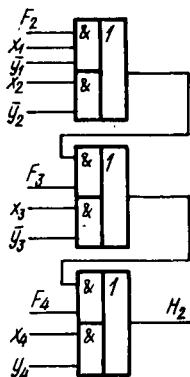


Рис. 3.15

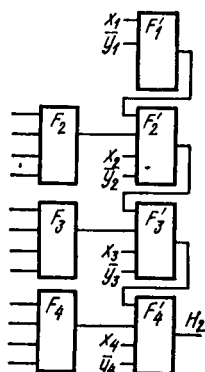


Рис. 3.16

но разбивать сравниваемые слова на группы по m разрядов и строить схемы сравнения для каждой группы. Результаты сравнения групп разрядов будут представлены функциями H_{1j} , H_{2j} (или H_{3j}), где $j = 1, 2, \dots, s$; $s = \left\lfloor \frac{n}{m} \right\rfloor$; $j = 1$ соответствует младшей группе разрядов. На основе функций H_{1j} , H_{2j} формируются функции H_1, H_2, H_3 :

$$H_1 = \bigwedge_{j=1}^s H_{1j};$$

$$\begin{aligned} H_2 &= H_{2s} \vee H_{1s} H_{2s-1} \vee H_{1s} H_{1s-1} H_{2s-2} \vee \dots \vee H_{1s} H_{1s-1} \dots H_{12} H_{21} = \\ &= H_{2s} \vee \bigvee_{j=1}^{s-1} \bigwedge_{r=j+1}^s H_{2j} \bigwedge_{r=j+1}^s H_{1r}; \\ H_3 &= H_1 \vee H_2. \end{aligned}$$

Схема, реализующая сравнение таким способом, представлена на рис. 3.17 ($n = 9$, $m = 3$). Для такой схемы

$$\begin{aligned} L &= s(L_1(m) + L_2(m)) + s + 2 = s(3m + 3m - 3) + \\ &\quad + s + 2 = s(6m - 1) + 2; \\ K &= s(K_1(m) + K_2(m)) + 2s + 2 + \frac{s^2 + s + 2}{2} = s(7m + 6m - 5) + \\ &\quad + \frac{s^2 + 5s + 2}{2}; \quad t = t_2(m) + 3t_2 = (2m + 3)t_3. \end{aligned}$$

Для распознавания соотношения между числами X и Y достаточно иметь схему с двумя выходами H_2 и H_3 , причем выполнению каждого из условий (3.1) будет соответствовать определенная комбинация выходных сигналов, например, 00, 10, 01 (комбинация 11 не используется). Такую схему сравнения называют также схемой с кодированным выходом. С целью получения форм переключательных функ-

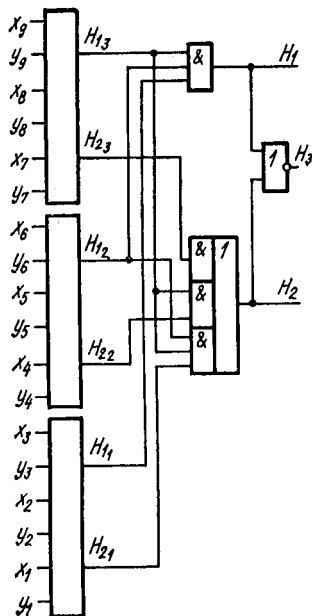


Рис. 3.17

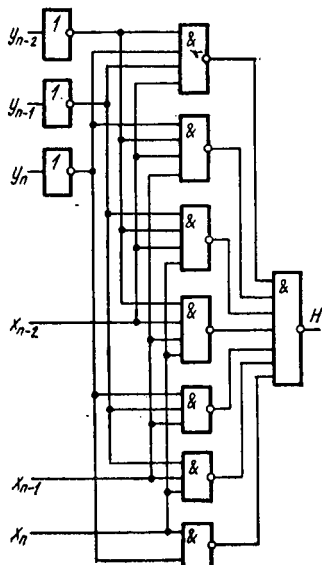


Рис. 3.18

ций H_2 и H_3 , пригодных для их реализации на одноконтурных интегральных модулях, раскроем скобки в выражениях (3.2) и (3.3):

$$H_2(X, Y) = x_n \bar{y}_n \vee x_n y_n x_{n-1} \bar{y}_{n-1} \vee \bar{x}_n \bar{y}_n x_{n-1} \bar{y}_{n-1} \vee \vee x_n y_n x_{n-1} y_{n-1} x_{n-2} \bar{y}_{n-2} \vee x_n y_n \bar{x}_{n-1} \bar{y}_{n-1} x_{n-2} y_{n-2} \vee \dots \vee \bar{x}_n \bar{y}_n \bar{x}_{n-1} \bar{y}_{n-1} \dots \bar{x}_2 \bar{y}_2 x_1 \bar{y}_1; \quad (3.5)$$

$$H_3(X, Y) = \bar{x}_n \bar{y}_n \vee x_n y_n \bar{x}_{n-1} y_{n-1} \vee \bar{x}_n \bar{y}_n \bar{x}_{n-1} y_{n-1} \vee x_n y_n x_{n-1} y_{n-1} \bar{x}_{n-2} y_{n-2} \vee x_n y_n \bar{x}_{n-1} \bar{y}_{n-1} \bar{x}_{n-2} y_{n-2} \vee \dots \vee \bar{x}_n \bar{y}_n \bar{x}_{n-1} \bar{y}_{n-1} \dots \bar{x}_2 \bar{y}_2 x_1 y_1. \quad (3.6)$$

Покажем справедливость тождественного соотношения

$$\bar{x}A \vee xAB = \bar{x}A \vee AB, \quad (3.7)$$

где A и B — произвольные выражения. Действительно,

$$\bar{x}A \vee xAB = \bar{x}A\bar{B} \vee \bar{x}AB \vee xAB = xA \vee AB.$$

Используя (3.7), упростим выражения (3.5) и (3.6) и запишем их в форме, удобной для реализации на элементах И—НЕ:

$$\begin{aligned} H_2(X, Y) &= x_n \bar{y}_n \vee x_n x_{n-1} \bar{y}_{n-1} \vee \bar{y}_n x_{n-1} \bar{y}_{n-1} \vee x_n x_{n-1} x_{n-2} \\ &\quad \bar{y}_{n-2} \vee \bar{y}_n x_{n-1} x_{n-2} \bar{y}_{n-2} \vee x_n \bar{y}_{n-1} x_{n-1} \bar{y}_{n-2} \vee \dots \vee \bar{y}_n \bar{y}_{n-1} \dots \\ &\quad \dots \bar{y}_2 x_1 \bar{y}_1 = \overline{(x_n \bar{y}_n)} \overline{(x_n x_{n-1} \bar{y}_{n-1})} \overline{(\bar{y}_n x_{n-1} \bar{y}_{n-1})} \dots \overline{(\bar{y}_n \bar{y}_{n-1} \dots \bar{y}_2 x_1 \bar{y}_1)}; \\ H_3(X, Y) &= \bar{x}_n \bar{y}_n \vee \bar{x}_n \bar{x}_{n-1} y_{n-1} \vee y_n \bar{x}_{n-1} y_{n-1} \vee \bar{x}_n \bar{x}_{n-1} \bar{x}_{n-2} y_{n-2} \vee \\ &\quad \vee y_n \bar{x}_{n-1} \bar{x}_{n-2} y_{n-2} \vee x_n y_{n-1} \bar{x}_{n-2} \vee \dots \vee y_n y_{n-1} \dots y_2 \bar{x}_1 y_1 = \\ &= \overline{(\bar{x}_n \bar{y}_n)} \overline{(\bar{x}_n \bar{x}_{n-1} y_{n-1})} \overline{(y_n \bar{x}_{n-1} y_{n-1})} \dots \overline{(y_n y_{n-1} \dots y_2 \bar{x}_1 y_1)}. \end{aligned}$$

Каждая из этих функций реализуема модулем на рис. 3.18. Если же $n > M$, где M — разрядность модуля (по входу), то на каждой паре модулей выполняется сравнение одной из $\left\lfloor \frac{n}{M} \right\rfloor$ групп разрядов.

Результаты сравнения групп будут представлены двумя $\left\lfloor \frac{n}{M} \right\rfloor$ разрядными числами, первое из которых H_2^* формируется из значений функций H_{2j} , $(j = 1, 2, \dots, \left\lfloor \frac{n}{M} \right\rfloor)$, а второе H_3^* из значений функций H_{3j} .

При указанном выше способе кодирования выходных функций равенство чисел H_2^* и H_3^* будет соответствовать равенству X и Y , $H_2^* > H_3^*$ соответствует $X > Y$, $H_2^* < H_3^*$ соответствует $X < Y$. Следовательно, для установления соотношения между X и Y необходимо реализовать сравнение слов H_2^* и H_3^* и т. д. до тех пор, пока не будут получены функции H_2 и H_3 . Принцип построения такой схемы сравнения из модулей на рис. 3.18 иллюстрируется на рис. 3.19. Для уменьшения числа внешних выводов целесообразно в одном модуле реализовать обе функции H_2 и H_3 (рис. 3.20). В этом случае схема сравнения строится в соответствии с рис. 3.21.

На практике часто требуется реализовать только одну из функций H_1 , H_2 , H_3 . Схему, реализующую функцию H_1 , называют схемой равенства кодов. Например, в блоке защиты памяти ЭВМ ЕС-1020 используется схема равенства кодов, показанная на рис. 3.22. На функциональных схемах обозначают схемы равенства в соответствии с рис. 3.23. При необходимости построения схемы, реализующей только функцию H_2 , выражение (3.2) целесообразно упростить следующим образом. Запишем функцию H_2 для двух разрядов

$$H_2(n=2) = x_2 \bar{y}_2 \vee (x_2 y_2 \vee \bar{x}_2 \bar{y}_2) x_1 \bar{y}_1 = x_2 \bar{y}_2 \vee x_2 y_2 x_1 \bar{y}_1 \vee \bar{x}_2 \bar{y}_2 x_1 \bar{y}_1.$$

Истинность этого выражения не изменится, если в него ввести дизъюнктивный член $x_2 \bar{y}_2 x_1 \bar{y}_1$, поглощаемый первым членом $x_2 \bar{y}_2$,

$$\begin{aligned} H_2(n=2) &= x_2 \bar{y}_2 \vee x_2 y_2 x_1 \bar{y}_1 \vee \bar{x}_2 \bar{y}_2 x_1 \bar{y}_1 \vee x_2 \bar{y}_2 x_1 \bar{y}_1 = \\ &= x_2 \bar{y}_2 \vee x_2 x_1 \bar{y}_1 \vee \bar{y}_2 x_1 \bar{y}_1 = x_2 \bar{y}_2 \vee x_1 \bar{y}_1 (x_2 \vee \bar{y}_2). \end{aligned}$$

Аналогично можно убедиться, что

$$H_2(n=3) = x_3 \bar{y}_3 \vee x_2 \bar{y}_2 (x_3 \vee \bar{y}_3) \vee x_1 \bar{y}_1 (x_3 \vee \bar{y}_3) (x_2 \vee \bar{y}_2).$$

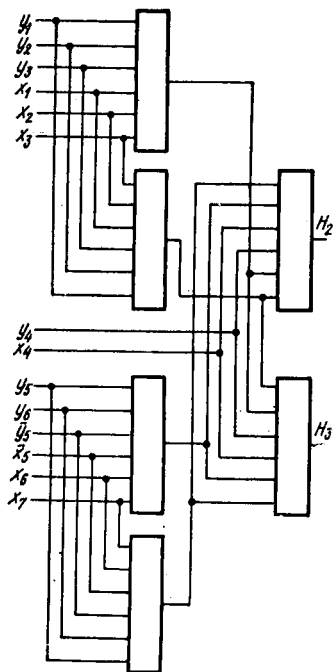


Рис. 3.19

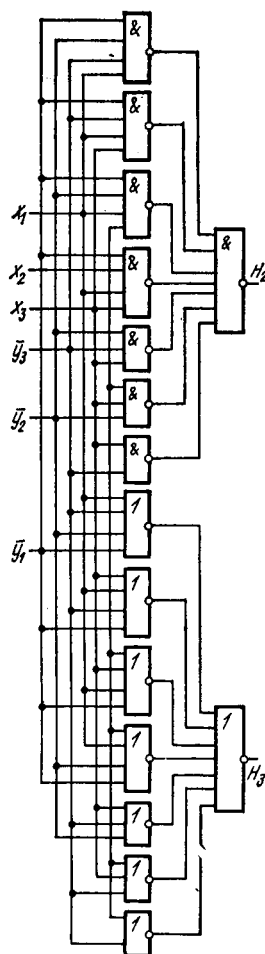


Рис. 3.20

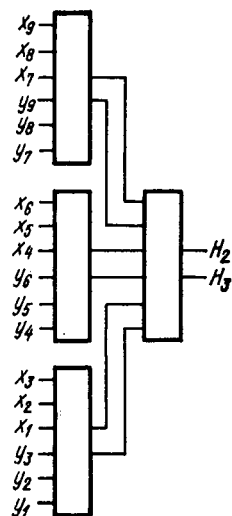


Рис. 3.21

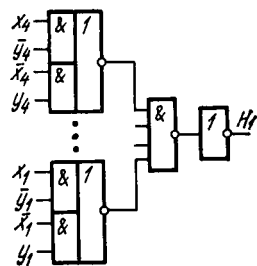


Рис. 3.22

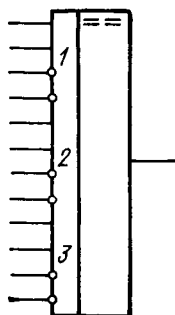


Рис. 3.23

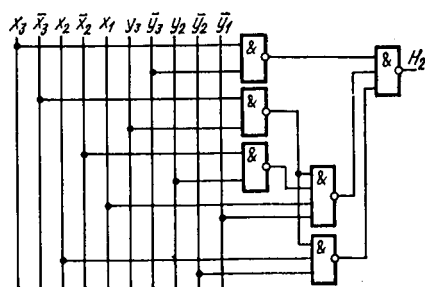


Рис. 3.24

В общем случае при произвольном n

$$H_2 = x_n \bar{y}_n \bigvee \bigvee_{i=1}^{n-1} x_i \bar{y}_i \bigwedge_{j=i+1}^n (x_j \bigvee \bar{y}_j).$$

На рис. 3.24 приведена схема, реализующая следующее операторное выражение для функции H_2 ($n = 3$),

$$H_2(n=3) = \overline{x_3 \bar{y}_3 x_2 \bar{y}_2 \bar{x}_3 y_3 x_1 \bar{y}_1 \bar{x}_2 y_2 \bar{x}_3 y_3}.$$

Для такой схемы $L = 2n$; $K = \frac{n^2 + 9n - 4}{2}$; $t = 3t_3$.

Если X и Y — числа со знаками (как и раньше, знаковые разряды обозначим X_0 и Y_0), то выполнение условий (3.1) может быть установлено с помощью схемы, реализующей функции

$$\begin{aligned} H_1(X, Y) &= (X_0 Y_0 \bigvee \bar{X}_0 \bar{Y}_0) H_1(|X|, |Y|) = F_0 H_1(|X|, |Y|); \\ H_2(X, Y) &= \bar{X}_0 Y_0 \bigvee F_0 \bar{X}_0 H_2(|X|, |Y|) \bigvee F_0 X_0 H_3(|X|, |Y|) = \\ &= \bar{X}_0 Y_0 \bigvee F_0 \bar{X}_0 H_2(|X|, |Y|) \bigvee F_0 X_0 \bar{H}_1(|X|, |Y|) \bar{H}_2(|X|, |Y|). \end{aligned}$$

Здесь $H_1(|X|, |Y|)$, $H_2(|X|, |Y|)$ и $H_3(|X|, |Y|)$ — функции сравнения H_1 , H_2 и H_3 абсолютных величин X и Y .

3.4. ШИФРАТОРЫ И СДВИГАТЕЛИ

Шифратором называют схему для преобразования пространственного k -значного кода в некоторый двузначный позиционный код. Шифратор имеет k входов, n выходов и выполняет функции, обратные функциям дешифратора, т. е. преобразует единичный сигнал на одном из k входов (иначе, унитарный k -разрядный код) в двузначный код номера этого входа. Если выходной код имеет естественный порядок весов числа 2, то

$$n = \lceil \log_2 k \rceil.$$

Метод построения схемы шифратора очень прост и заключается в следующем. Если номеру j возбужденного входа шифратора соответствует двузначный выходной код C_j , то j -ю входную шину необходимо подвести ко входам элементов ИЛИ в тех разрядах, где C_j содержит единицы. Пусть $k = 10$ и выходной код имеет естественный порядок двоичных весов. Тогда $n = 4$, а систему функций, реализуемых шифратором, можно записать как

$$\begin{aligned} y_1 &= 1 \bigvee 3 \bigvee 5 \bigvee 7 \bigvee 9; \\ y_2 &= 2 \bigvee 3 \bigvee 6 \bigvee 7; \\ y_3 &= 4 \bigvee 5 \bigvee 6 \bigvee 7; \\ y_4 &= 8 \bigvee 9. \end{aligned}$$

Цифры в правой части этих выражений соответствуют номерам возбужденных входов. Схема такого шифратора приведена на рис. 3.25, а, а его условное обозначение — на рис. 3.25, б.

Построенные таким образом шифраторы, по аналогии с дешифраторами, могут быть названы линейными. При использовании элемен-

тов И—НЕ целесообразно выходные функции шифратора представить в такой операторной форме

$$y_1 = \overline{1 \ 3 \ 5 \ 7 \ 9}; \quad y_2 = \overline{2 \ 3 \ 6 \ 7} \text{ и т. д.}$$

Впредь будем считать, что k равно целой степени числа 2. Тогда для линейного шифратора

$$L = n = \log_2 k; \quad K = \frac{k}{2} n = \frac{k \log_2 k}{2}; \quad t = t_3.$$

В случае, когда коэффициент объединения M используемых логических элементов меньше $\frac{k}{2}$, целесообразно проектировать каскадные шифраторы. Один из способов каскадного построения шифраторов состоит в следующем. Вначале формируют дизъюнкции по M входных шин, коды номеров которых C_i имеют единицу в первом (младшем) разряде, затем формируют дизъюнкции по M входных шин, коды номеров которых C_j имеют единицу во втором разряде и т. д. Элементы, реализующие эти дизъюнкции, составляют первый каскад шифратора. Во втором и всех последующих каскадах формируют дизъюнкции по M функций предыдущего каскада, соответствующих единице в

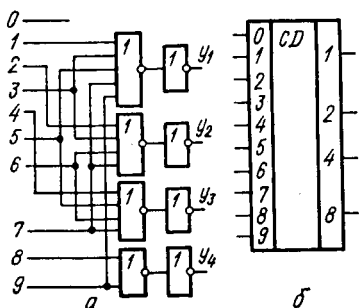


Рис. 3.25

первом разряде C_i , затем во втором разряде C_j , и т. д. Общее число каскадов составляет $\left\lceil \log_M \frac{k}{2} \right\rceil$.

При таком построении шифратора на каждом r -м каскаде ($r = 1, 2, 3, \dots$) некоторые дизъюнкции по M функций предыдущего каскада будут общими для выходных функций $y_s, y_{s+1}, \dots, y_n$, где $s \geq \log_2 M^r + 1$. Пусть, например, $k = 32$, $M = 4$, тогда

$$\begin{aligned} y_1 &= (1 \vee 3 \vee 5 \vee 7) \vee (9 \vee 11 \vee 13 \vee 15) \vee (17 \vee 19 \vee 21 \vee 23) \vee \\ &\quad \vee (25 \vee 27 \vee 29 \vee 31); \\ y_2 &= (2 \vee 3 \vee 6 \vee 7) \vee (10 \vee 11 \vee 14 \vee 15) \vee (18 \vee 19 \vee 22 \vee 23) \vee \\ &\quad \vee (26 \vee 27 \vee 30 \vee 31); \\ y_3 &= (4 \vee 5 \vee 6 \vee 7) \vee (12 \vee 13 \vee 14 \vee 15) \vee (20 \vee 21 \vee 22 \vee 23) \vee \\ &\quad \vee (28 \vee 29 \vee 30 \vee 31); \\ y_4 &= (8 \vee 9 \vee 10 \vee 11) \vee (12 \vee 13 \vee 14 \vee 15) \vee (24 \vee 25 \vee 26 \vee 27) \vee \\ &\quad \vee (28 \vee 29 \vee 30 \vee 31); \\ y_5 &= (16 \vee 17 \vee 18 \vee 19) \vee (20 \vee 21 \vee 22 \vee 23) \vee (24 \vee 25 \vee 26 \vee 27) \vee \\ &\quad \vee (28 \vee 29 \vee 30 \vee 31). \end{aligned}$$

Из этих выражений видно, что дизъюнкция $(12 \vee 13 \vee 14 \vee 15)$ является общей для функций y_3 и y_4 , дизъюнкция $(20 \vee 21 \vee 22 \vee 23)$ является общей для функций y_3 и y_5 , дизъюнкция $(24 \vee 25 \vee 26 \vee 27)$

$\vee 27)$ — общей для функций y_4 и y_5 , дизъюнкция $(28 \vee 29 \vee 30 \vee 31)$ — для функций y_3, y_4, y_5 . Кроме того, дизъюнкция $(24 \vee 25 \vee 26 \vee 27) \vee (28 \vee 29 \vee 30 \vee 31)$ является общей для функций y_4 и y_5 . Если считать, что $s = \log_2 M^r + 1$, то число общих дизъюнктивных членов на каждом каскаде равно

$$\frac{1}{M^r} \sum_{j=s}^{n-1} 2^j = n + s.$$

Поэтому для каждого r -го каскада (за исключением последнего)

$$L_r = \frac{1}{M^r} \left(\frac{k}{2} \log_2 M^r + \sum_{j=\log_2 M^r + 1}^{n-1} 2^j \right) + 1;$$

$$K_r = \frac{1}{M^{r-1}} \left(\frac{k}{2} \log_2 M^r + \sum_{j=\log_2 M^r + 1}^{n-1} 2^j \right) + M.$$

Число элементов на последнем каскаде равно n . В табл. 47 приведены оценки L каскадного шифратора при некоторых значениях k и M . На рис. 3.26 приведена схема шифратора, построенного по описанному способу с использованием элементов ИЛИ—НЕ и И—НЕ при $k = 32$, $M = 4$.

Таблица 47

k	M		
	2	4	8
4	2	2	2
8	8	3	3
16	22	11	4
32	52	20	14
64	114	48	25
128	240	93	46

Еще один способ каскадного построения шифраторов представляет собой «свертку» k -входового шифратора к шифратору с меньшим числом входных шин. Его целесообразно использовать, если уже имеются готовые шифраторы с числом входов k_1 (например, выполненные в виде интегральных модулей), а число входов проектируемого шифратора равно $k > k_1$. Сущность этого способа состоит в следующем (считаем, что $k_1 = 2^m$). Сначала входные шины проектируемого шифратора разделяют на k_1 групп так, что каждая группа объединяет шины, двузначные коды номеров которых имеют одинаковые цифры в m старших разрядах. В таком случае первая группа содержит входные шины с номерами $0, 1, \dots, \frac{k}{k_1} - 1$, вторая группа — шины с номерами $\frac{k}{k_1}, \frac{k}{k_1} + 1, \dots, \frac{2k}{k_1} - 1$ и т. д. Последняя k_1 -я группа содержит входные шины с номерами $\frac{(k_1 - 1)k}{k_1}, \frac{(k_1 - 1)k}{k_1} + 1, \dots, k - 1$. В каждой из этих групп входные шины подключают к элементу ИЛИ с $\frac{k}{k_1}$ входами.

Выходы элементов ИЛИ соединены с k_1 -входным шифратором, формирующим старшие m разрядов выходного слова. Затем входные шины, двузначные коды номеров которых отличаются только в m

старших разрядах, объединяют в $\frac{k}{k_1}$ групп по k_1 шин в каждой с помощью k_1 -входовых элементов ИЛИ. Выходы этих элементов ИЛИ являются входными шинами для второго каскада шифратора. При этом номера входных шин второго каскада определяются содержимым $n - m$ младших разрядов номеров входных шин первого каскада. Второй и все последующие каскады шифратора строятся аналогично. Заметим, что на каждом каскаде нет необходимости формиро-

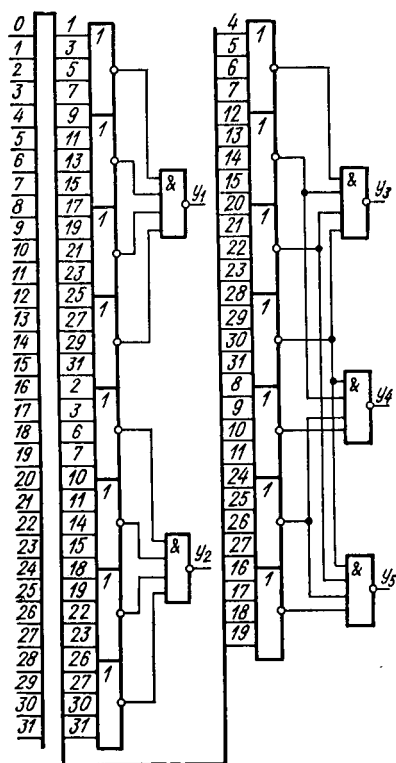


Рис. 3.26

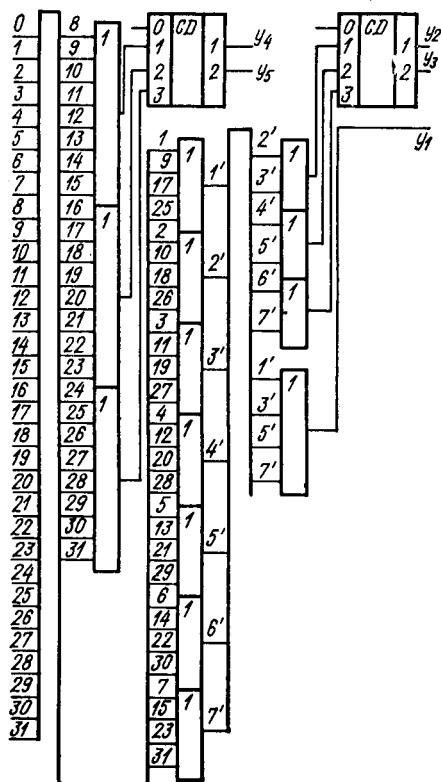


Рис. 3.27

вать дизъюнкции, соответствующие входным шинам следующего каскада с номером 0, а также дизъюнкции входных шин, в m старших разрядах номеров которых находятся нули. Общее число каскадов r в таком шифраторе равно $\left\lfloor \frac{n}{m} \right\rfloor - 1$. Оценки сложности шифратора (без учета сложности k_1 -входовых шифраторов) имеют вид:

$$L = r(2^m - 2) + \frac{2^n - 2^{n-mr}}{2^m - 1} = r(k_1 - 2) + \frac{k(k_1^r - 1)}{k_1^r(k_1 - 1)};$$

$$K = \frac{(2^{mr} - 1) 2^n}{2^{mr} (2^m - 1)} (2^{m+1} - 1) - r 2^m = \frac{(k_1^r - 1) k}{k_1^r (k_1 - 1)} (2k_1 - 1) - r k_1.$$

Если $k = 32$, $k_1 = 4$, то построение шифратора по рассматриваемому способу иллюстрируется схемой на рис. 3.27. Здесь цифрами $1'$, $2'$, ... обозначены входные шины второго каскада.

Полным двусторонним сдвигателем будем называть схему для выполнения операции $X2^Y$, где $X = \sum_{i=1}^n x_i 2^{i-1}$, $Y = \pm \sum_{j=1}^m y_j 2^{j-1}$, т. е.

полный двусторонний сдвигатель осуществляет сдвиг на Y разрядов влево или вправо (в зависимости от знака Y) слова, представляющего X . Такой сдвигатель имеет n входов, по которым поступают разряды сдвигаемого слова X , а также $m + 1$ входов, по которым поступают разряды и знак управляющего слова Y . Впредь для определенности будем считать, что Y может принимать значения $0, 1, \dots, n - 1$, так как на практике редко возникает необходимость сдвига слов больше, чем на $n - 1$ разрядов (в противном случае сдвигаемое слово выходит за пределы разрядной сетки цифровой ЭВМ). Поэтому $m = \lceil \log_2 n \rceil$, а число выходов полного двустороннего сдвигателя равно $3n - 2$. Схема сдвигателя на уровне «черного ящика» приведена на рис. 3.28. Будем обозначать выходы сдвигателя следующим образом: те выходы, на которые попадает входное слово при $Y = 0$, обозначим s_i ($i = 1, 2, \dots, n$); выходы, на которые попадает хотя бы один разряд входного слова при $Y_0 = 0$, обозначим s_{n+j} ($j = 1, 2, \dots, n - 1$); выходы, на которые попадает хотя бы один разряд при $Y_0 = 1$, обозначим s_{-j} . Тогда работу сдвигателя можно описать функциями

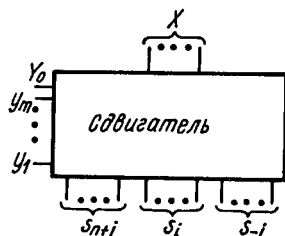


Рис. 3.28

$$s_i = \begin{cases} x_i & \text{при } Y = 0; \\ x_{i-Y} & \text{при } Y_0 = 0; \\ x_{i+Y} & \text{при } Y_0 = 1; \\ 0 & \text{в остальных случаях;} \end{cases}$$

$$s_{-j} = \begin{cases} x_{|Y|-j+1} & \text{при } Y_0 = 1, Y \geq j; \\ 0 & \text{в остальных случаях;} \end{cases}$$

$$s_{n+j} = \begin{cases} x_{n+j-|Y|} & \text{при } Y_0 = 0, Y \geq j; \\ 0 & \text{в остальных случаях.} \end{cases}$$

Реализовать эти функции можно с помощью дешифратора на n выходов и коммутирующей схемы. Рассмотрим пример. Пусть $n = 3$, тогда $m = 2$. Обозначим выходы дешифратора, на вход которого поступает слово Y , буквами D_j , где j будет соответствовать численному значению Y . Тогда функции s_i , s_{-j} и s_{n+j} можно записать в таком виде:

$$\begin{aligned} s_1 &= x_1 D_0 \vee x_2 D_1 Y_0 \vee x_3 D_2 Y_0; \\ s_2 &= x_2 D_0 \vee x_3 D_1 Y_0 \vee x_1 D_1 \bar{Y}_0; \\ s_3 &= x_3 D_0 \vee x_2 D_1 \bar{Y}_0 \vee x_1 D_2 \bar{Y}_0; \\ s_{-1} &= x_1 D_1 \bar{Y}_0 \vee x_2 D_2 Y_0; \\ s_{-2} &= x_1 D_2 Y_0; \\ s_{3+1} &= x_3 D_1 \bar{Y}_0 \vee x_2 D_2 \bar{Y}_0; \quad s_{3+2} = x_3 D_2 \bar{Y}_0. \end{aligned}$$

Схема такого сдвигателя на элементах И—ИЛИ—НЕ представлена на рис. 3.29. Число элементов И и ИЛИ, необходимых для реализации функций s_i в таком сдвигателе, составляет $n^2 + n$, а число элементов для реализации функций s_{-j} и s_{n+j} (кроме функций s_{-n+1} и s_{n+n-1}) определяется суммой членов арифметической прогрессии, разность которой равна 1, первый член равен 3, а последний n . Таким обра-

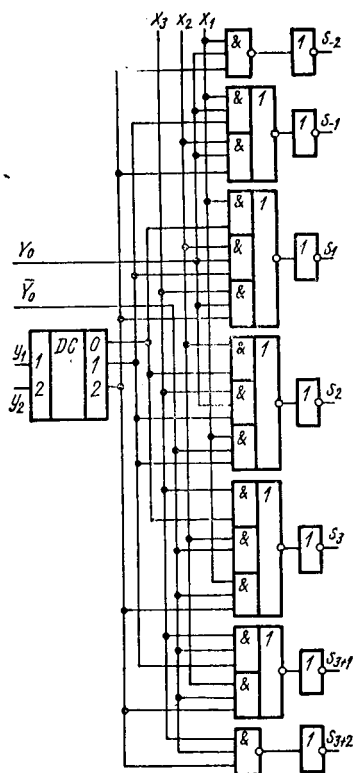


Рис. 3.29

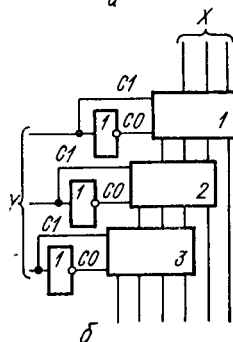
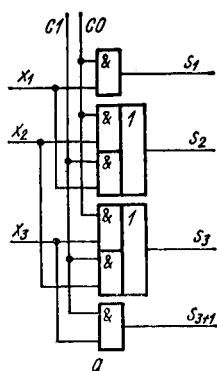


Рис. 3.30

зом, сложность рассматриваемого полного двустороннего сдвигателя определяется выражениями

$$L = 2n^2 + 2n - 4 + L_{DC};$$

$$K = 8n^2 - 5n - 2 + K_{DC}.$$

Здесь L_{DC} и K_{DC} — соответствующие оценки сложности дешифратора. Задержка сигналов таким сдвигателем составляет

$$t = 2t_3 + t_{DC},$$

где t_{DC} — задержка сигналов дешифратором. Несколько улучшить оценки L и t сдвигателя можно, если совместить на одних и тех же элементах реализацию функций D_j и произведений $\tilde{x}_j \tilde{Y}_0 D_j$. Однако для этого требуются $m + 2$ -входные элементы, что приводит к значи-

тельному ухудшению оценки K и росту нагрузки на элементы, формирующие управляющее слово Y . Улучшить оценку K можно путем вынесения за скобки некоторых переменных (например, Y_0 и $\bar{Y}_0$) в операторных формах функций s_i, s_{-j}, s_{n+j} , но при этом ухудшается оценка L и снижается быстродействие сдвигателя.

В практике проектирования операционных узлов значительно чаще, чем двусторонние сдвигатели, встречаются односторонние, выполняющие операции либо $X2Y$, либо $X2-Y$, а также частичные сдвигатели, реализующие лишь некоторые из функций s_i, s_{-j}, s_{n+j} . Такие сдвигатели характеризуются значительно меньшей сложностью.

При построении сдвигателей с использованием интегральных модулей повышенной степени интеграции возникает задача такого их выполнения, которое обеспечивает наращиваемость их структуры по числу реализуемых функций. Рассмотрим некоторые способы решения этой задачи на примерах построения односторонних сдвигателей. Один из способов состоит в том, что каждый модуль выполняют как сдвигатель, обеспечивающий сдвиг входного n -разрядного слова на 0 и 1 разряд (рис. 3.30, а, $n = 3$). Такой модуль имеет $n + 1$ выходов, а величина сдвига определяется единичным сигналом по одной из шин СО или С1. Для правильной работы сдвигателя необходимо выполнение условия $(CO) (C1) = 0$. Путем каскадного включения таких модулей можно построить сдвигатель на требуемое число разрядов (рис. 3.30, б). Сдвиг слова на i разрядов обеспечивается подачей единичных сигналов С1 на i каскадов, начиная с первого, т. е. управляющее слово Y должно быть представлено в непозиционной унитарной системе счисления (см. п. п. 1.4, 3.7). Для рассматриваемого одностороннего сдвигателя

$$L = 3n l + L_{np}; \quad K = 6n l - l + K_{np};$$

$$t = (2l + 1) t_3 + t_{np},$$

где L_{np}, K_{np}, t_{np} — соответствующие оценки преобразователя позиционного представления слова Y в унитарное; l — число каскадов сдвигателя, равное максимальному значению Y . Если принять $l = 2^m - 1$, то

$$L = 3n (2^m - 1) + L_{np};$$

$$K = 5n 2^m - 6n + 1 + K_{np};$$

$$t = (2^{m+1} - 1) t_3 + t_{np}.$$

Односторонний наращиваемый сдвигатель можно построить и без использования унитарного представления управляющего слова Y , если модули на каждом i -м каскаде сдвигателя будут реализовывать сдвиг на 0 и на 2^{i-1} разрядов (рис. 3.31). Здесь сдвиг на 1 разряд выполняется модулем на 1-м каскаде по сигналу $C1 = 1$, сдвиг на 2 разряда — модулем на 2-м каскаде по сигналу $C2 = 1$, сдвиг на 3 разряда — модулями 1 и 2 по сигналам $C1 = C2 = 1$, сдвиг на 7 разрядов — модулями 1, 2, 3 по сигналам $C1 = C2 = C4 = 1$ и т. д. В таком сдвигателе для каждого i -го модуля

$$L_i = 3n - 2^{i-1}; \quad K_i = 6n - 2^i; \quad t_i = 2t_3,$$

а для всего сдвигателя

$$L = \sum_{i=1}^m (3n - 2^{i-1}) = 3mn - 2^m + 1;$$

$$K = 6mn - 2^{m+1} + 2;$$

$$t = 2m t_3.$$

Дальнейшего упрощения каскадного наращиваемого сдвигателя и повышения его быстродействия можно достичь, если использовать представление управляющего слова Y в троичной системе счисления с цифрами 1, 0, $\bar{1}$. В этом случае модуль 1-го каскада должен реализовать сдвиг на 0 разрядов (C0), на 1 разряд влево (C1) и на 1 разряд вправо (C $\bar{1}$). Модуль второго каскада выполняет сдвиг C0, C3,

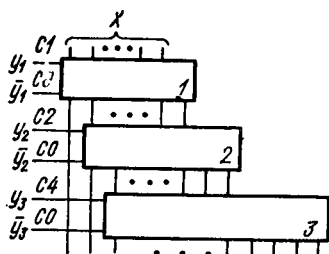


Рис. 3.31

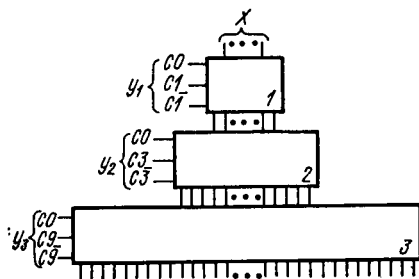


Рис. 3.32

C $\bar{3}$, модуль третьего каскада — C0, C9, C $\bar{9}$, модуль i -го каскада — C0, C 3^{i-1} —C $\bar{3}^{i-1}$. Схема организации описываемого сдвигателя показана на рис. 3.32. Если, например, $Y = 7$, то такой сдвиг реализуется как совокупность сдвигов C1 на первом каскаде, C $\bar{3}$ на втором каскаде и C9 на третьем каскаде. Управляющие сигналы для i -го каскада определяются цифрой в i -м разряде троичного представления Y . Поскольку в этом случае $m = \lceil \log_3 n \rceil$, то число каскадов такого сдвигателя в $\log_2 n / \log_3 n = \log_2 3$ раз меньше числа каскадов предыдущего сдвигателя. Формулы для оценок сложности L и K рассматриваемого сдвигателя при произвольном n очень громоздки и неудобны для использования в инженерном проектировании цифровых схем. Заметим только, что при конкретных значениях n двусторонний сдвигатель по схеме на рис. 3.32 всегда проще сдвигателя по схеме на рис. 3.29.

3.5. КОМБИНАЦИОННЫЕ СУММАТОРЫ

Комбинационным сумматором будем называть схему, осуществляющую арифметическое сложение (суммирование) чисел $X \sum_{i=1}^n x_i 2^{i-1}$ и $Y = \sum_{i=1}^n y_i 2^{i-1}$.

Следует отметить, что полная операция сложения, выполняемая в цифровых ЭВМ, значительно сложнее операции суммирования, реализуемой сумматором, так как при сложении необходимо учитывать знаки чисел, выравнивать порядки, нормализовать числа и т. д.

Будем называть одноразрядным сумматором схему с 3 входами x_i , y_i и z_i и двумя выходами s_i и p_i (рис. 3.33, а), которая по раз-

рядным значениям x_i и y_i слагаемых X и Y и по значению переноса z_i из младшего разряда формирует значения разрядной суммы s_i и переноса p_i в старший разряд. Диаграммы Вейча функций s_i и p_i приведены на рис. 3.34.

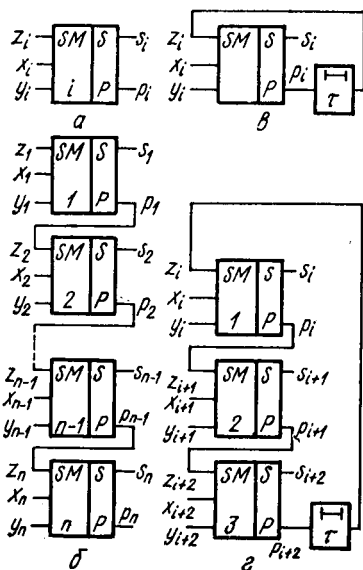


Рис. 3.33

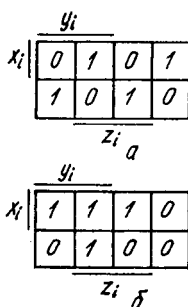


Рис. 3.34

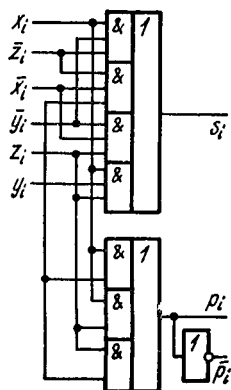


Рис. 3.35

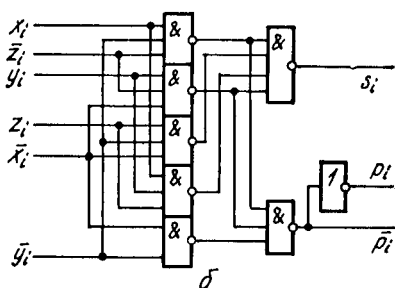
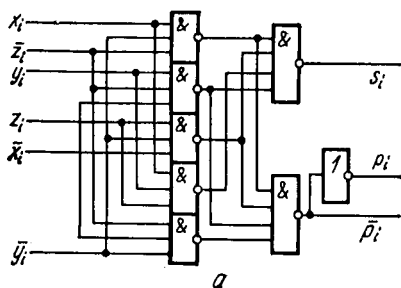


Рис. 3.36

Полный n -разрядный сумматор может строиться из одноразрядных сумматоров различными способами. Различают параллельные, последовательные и параллельно-последовательные сумматоры. Построение параллельного сумматора иллюстрируется схемой на рис. 3.33, б. Количество одноразрядных сумматоров в такой схеме равно количеству разрядов складываемых чисел. На вход z_1 при обычном суммировании подается 0, а при суммировании с циклическим переносом ко входу z_1 подключается выход p_n этого же сумматора.

Последовательный сумматор строится по блок-схеме на рис. 3.33, в, где τ — задержка на один такт. В отличие от параллельного сумматора здесь имеется всего один одноразрядный сумматор. Поразрядные значения слагаемых поступают на входы x_i и y_i по очереди: сначала младшие разряды, затем следующие по весу и т. д. Значение переноса p_i , появляющееся в i -м такте, с помощью задержки τ (триггера, линии задержки) запоминается на время одного такта и в начале $(i+1)$ -го такта подается на вход z_{i+1} одновременно с подачей x_{i+1} и y_{i+1} .

Построение параллельно-последовательного сумматора иллюстрируется блок-схемой на рис. 3.33, г. В этом случае n -разрядные слова X и Y разбиваются на несколько подслов по m разрядов каждое (на рис. 3.33, г $m = 3$). На схему сумматора подслова подаются последовательно во времени, однако суммирование подслов происходит параллельно.

Рассмотрим способы построения схем одноразрядного сумматора. СДНФ функций s_i и p_i имеют вид

$$s_i = x_i \bar{y}_i \bar{z}_i \vee \bar{x}_i y_i \bar{z}_i \vee \bar{x}_i \bar{y}_i z_i \vee x_i y_i z_i; \quad (3.8)$$

$$p_i = x_i y_i \bar{z}_i \vee x_i \bar{y}_i z_i \vee \bar{x}_i y_i z_i \vee x_i y_i z_i. \quad (3.9)$$

Нетрудно заметить, что ни одна пара конstituент 1 в (3.8) не может быть подвергнута операции склеивания. Выражение же (3.9) допускает упрощение

$$p_i = x_i y_i \vee x_i z_i \vee y_i z_i. \quad (3.10)$$

Схема одноразрядного сумматора, реализующего функции (3.8) и (3.10), приведена на рис. 3.35. Инвертор в этой схеме необходим для формирования сигнала $\bar{p}_i$, используемого в следующем старшем разряде в качестве сигнала $\bar{z}_{i+1}$. Для такого одноразрядного сумматора

$$L = 10; K = 26; t = 3t_3.$$

Заметим, что если сумматор проектировать на элементах И—ИЛИ—НЕ, то при этом целесообразно исходить из МДНФ функций $\bar{s}_i$ и $\bar{p}_i$

$$\bar{s}_i = \overline{x_i \bar{y}_i \bar{z}_i \vee \bar{x}_i y_i \bar{z}_i \vee \bar{x}_i \bar{y}_i z_i \vee x_i y_i z_i};$$

$$\bar{p}_i = \overline{\bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{z}_i \vee \bar{y}_i \bar{z}_i}.$$

Однако при рассматриваемом выше построении одноразрядных сумматоров не учитывается то обстоятельство, что функции s_i и p_i зависят от одних и тех же аргументов и, кроме того, их СДНФ имеют общие члены. Наиболее полно учесть эти факторы можно при реализации функции s_i по выражению (3.8) и при следующем подходе к реализации схемы для p_i . Запишем СДНФ для отрицания функции p_i

$$\bar{p}_i = x_i \bar{y}_i \bar{z}_i \vee \bar{x}_i y_i \bar{z}_i \vee \bar{x}_i \bar{y}_i z_i \vee \bar{x}_i \bar{y}_i \bar{z}_i. \quad (3.11)$$

Сравнивая это выражение с (3.8) убеждаемся, что первые три дизъюнктивных члена в них совпадают. Следовательно, одноразряд-

ный сумматор можно строить по схеме на рис. 3.36,а. Для такого сумматора

$$L = 8; K = 24; t = 3t_3.$$

Схема сумматора может быть еще несколько упрощена (рис. 3.36,б), если в выражении (3.11) выполнить операцию склеивания двух последних членов:

$$p_i = x_i \bar{y}_i \bar{z}_i \vee \bar{x}_i y_i \bar{z}_i \vee \bar{x}_i \bar{y}_i. \quad (3.12)$$

В таком случае

$$L = 8; K = 22; t = 3t_3.$$

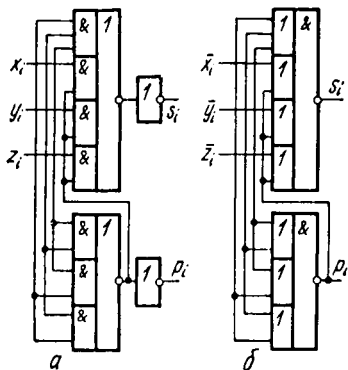


Рис. 3.37

К числу недостатков последних схем сумматоров можно отнести то, что выходы некоторых элементов нагружаются на два входа. Это недопустимо, например, при использовании пассив-

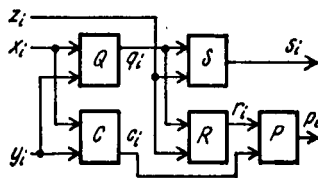


Рис. 3.38

ных элементов, или элементов И—ИЛИ—НЕ, ИЛИ—И—НЕ. Для устранения этого недостатка поступим следующим образом. Припишем к СДНФ функции s_i конъюнкции вида $x_i \bar{x}_i \bar{y}_i$, $x_i \bar{x}_i \bar{z}_i$, $\bar{x}_i y_i \bar{y}_i$, $\bar{x}_i y_i \bar{z}_i$, $\bar{x}_i z_i \bar{z}_i$, $\bar{y}_i z_i \bar{z}_i$, имеющие нулевое значение. Получим

$$\begin{aligned} s_i &= x_i \bar{y}_i \bar{z}_i \vee \bar{x}_i y_i \bar{z}_i \vee \bar{x}_i \bar{y}_i z_i \vee x_i y_i z_i \vee x_i \bar{x}_i \bar{y}_i \vee x_i \bar{x}_i \bar{z}_i \vee \\ &\vee \bar{x}_i y_i \bar{y}_i \vee y_i \bar{y}_i \bar{z}_i \vee \bar{x}_i z_i \bar{z}_i \vee \bar{y}_i z_i \bar{z}_i = x_i y_i z_i \vee x_i (\bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{z}_i \vee \\ &\vee \bar{y}_i \bar{z}_i) \vee y_i (\bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{z}_i \vee \bar{y}_i \bar{z}_i) \vee z_i (\bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{z}_i \vee \bar{y}_i \bar{z}_i) = x_i y_i z_i \vee \\ &\vee (x_i \vee y_i \vee z_i) (\bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{z}_i \vee \bar{y}_i \bar{z}_i). \end{aligned}$$

Выражение в последних скобках есть МДНФ отрицания функции (3.10). Таким образом, можно записать, что

$$\begin{aligned} s_i &= x_i y_i z_i \vee (x_i \vee y_i \vee z_i) \bar{p}_i = \overline{\overline{x_i y_i z_i \vee x_i \bar{p}_i \vee y_i \bar{p}_i \vee z_i \bar{p}_i}} = \\ &= \overline{(\bar{x}_i \vee \bar{y}_i \vee \bar{z}_i) (\bar{x}_i \vee \bar{p}_i) (\bar{y}_i \vee \bar{p}_i) (\bar{z}_i \vee \bar{p}_i)}. \end{aligned} \quad (3.13)$$

$$p_i = \overline{\overline{x_i y_i \vee x_i z_i \vee y_i z_i}} = \overline{(\bar{x}_i \vee \bar{y}_i) (\bar{x}_i \vee \bar{z}_i) (\bar{y}_i \vee \bar{z}_i)} \quad (3.14)$$

а соответствующие схемы сумматоров представлены на рис. 3.37 (так построены, например, микросхемы 155ИМ1—155ИМ3). Положительным качеством таких схем является также то, что на их входах используются либо только прямые значения независимых переменных, либо только инверсные.

Выше рассматривалось проектирование одноразрядного сумматора, реализующего функции s_i и p_i от трех переменных. Построение сумматора можно свести к реализации функций двух переменных, если воспользоваться блок-схемой сумматора на рис. 3.38. Здесь блок Q реализует функцию q_i сложения по mod2 поразрядных значений x_i и y_i слагаемых, блок C — функцию c_i переносов при сложении x_i и y_i , блок S — функцию s_i сложения q_i и переноса z_i из младшего разряда, блок R — функцию переносов r_i при сложении q_i и z_i , блок P формирует значение переноса p_i в старший разряд. Функционирование блоков Q , C , S , R и P можно описать табл. 48 (работа блоков Q и S , а также блоков C и R описывается одинаковыми таблицами, что обуславливает одинаковую внутреннюю структуру этих блоков). Функция p_i (c_i , r_i) на наборе 11 не определена, так как такой набор не может никогда появиться на входах блока P . Доопределять эту функцию надо, исходя из условия максимальной простоты реализации схемы. Запишем СДНФ функций q_i , c_i и p_i :

$$q_i = x_i \bar{y}_i \vee \bar{x}_i y_i, \quad c_i = x_i y_i, \quad p_i = c_i \vee r_i.$$

Таблица 48

		y_i, z_i				y_i, z_i				r_i	
		0	1			0	1			0	1
x_i, q_i	0	0	1	x_i, q_i	0	0	0	c_i	0	0	1
	1	1	0		1	0	1		1	1	—
		q_i, s_i				c_i, r_i				p_i	

Функция p_i при таком ее построении на наборе 11 будет принимать единичные значения. Очевидно, что записанные выражения являются минимальными формами функций c_i и p_i . Что касается функции q_i , то ее минимальная форма зависит от того, имеются ли в наличии инверсные значения независимых переменных. При наличии инверсии аргументов минимальной формой функции q_i помимо приведенной выше, является также форма $q_i = (x_i \vee y_i) (\bar{x}_i \vee \bar{y}_i)$. При отсутствии инверсий аргументов минимальным является выражение

$$q_i = (x_i \vee y_i) (\bar{x}_i \bar{y}_i) = (x_i \vee y_i) \bar{c}_i. \quad (3.15)$$

Операторными формами функций q_i , c_i и p_i будут следующие:

$$\begin{aligned} q_i &= \overline{(x_i \bar{y}_i) (\bar{x}_i y_i)} = \overline{(\bar{x}_i \vee y_i) (x_i \vee \bar{y}_i)} = \overline{(x_i \vee y_i) \vee c_i} = \\ &= \overline{(x_i \bar{y}_i) \vee c_i}; \quad c_i = \overline{x_i y_i} = \bar{x}_i \vee \bar{y}_i; \quad p_i = \overline{c_i \vee r_i} = \bar{c}_i \vee \bar{r}_i. \end{aligned}$$

Схемы, реализующие функции q_i и c_i в совокупности часто называют полусумматорами или одноразрядными сумматорами на два входа в отличие от рассмотренного ранее полного одноразрядного сумматора на три входа. Некоторые варианты схемной реализации полусумматоров приведены на рис. 3.39. Полный одноразрядный сумматор может быть построен из полусумматоров по схеме на

рис. 3.40. Если при этом структура полусумматоров соответствует схеме на рис. 3.39,а, то для полного одноразрядного сумматора

$$L = 9; K = 16; t_s = 5t_3; t_p = 3t_3.$$

В этих оценках учтены инверторы, необходимые для получения $\bar{q}_i$ и $\bar{p}_i$; t_s и t_p — задержки сигналов в цепях формирования сигналов s_i и p_i .

В последние годы большинство новых цифровых ЭВМ, предназначенных для инженерных расчетов, строится так, чтобы вычисления

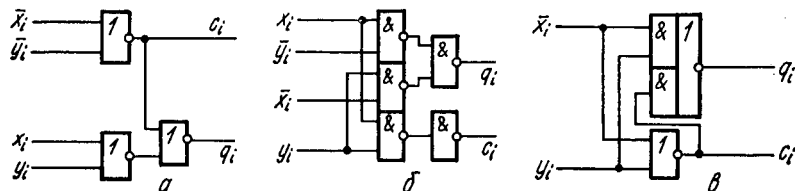


Рис. 3.39

в них выполнялись в десятичной системе счисления. Это делает более удобным программирование на таких цифровых ЭВМ, обеспечивает простоту индикации промежуточных и конечных результатов, исключает блоки преобразования чисел из одной системы счисления в другую.

Задача построения десятичного одноразрядного сумматора (точнее двоично-десятичного) в принципе может решаться так же, как и задача построения двоичного сумматора. Однако если в случае двоичного сумматора требовалось реализовать две функции от трех двузначных переменных, то в случае десятичного сумматора при представлении десятичных цифр двоичными тетрадами, необходимо реализовать пять функций от 9 двузначных переменных — по 4 переменных для каждой десятичной цифры и одна переменная — перенос из младшего разряда. При этом выходные функции будут определены лишь на 200 наборах из $2^9 = 512$. Можно было бы записать СДНФ этих функций и минимизировать их известными формальными методами. Однако СДНФ в этом случае содержали бы десятки дизъюнктивных членов, что значительно затрудняет минимизацию. Кроме того, вследствие большого числа неопределенных значений функций трудно решить вопрос об их оптимальном доопределении. Поэтому при использовании двоично-десятичных кодов, обладающих свойством аддитивности (см. п. 1.7), задачу суммирования десятичных цифр решают в два этапа. На первом этапе суммируют двоичные коды десятичных цифр по правилам двоичной арифметики. На втором этапе производят коррекцию результата путем прибавления или вычитания некоторой поправки, величина которой связана с величиной некорректированного результата, а также формируют перенос в старший десятичный разряд. Если же двоично-десятичный код не обладает свойством

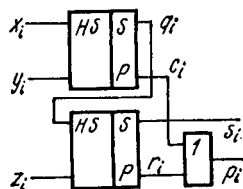


Рис. 3.40

аддитивности (например, код «3, 3, 2, 1»), то цифры слагаемых следует преобразовать из такого кода в код, обладающий указанным свойством. Пусть, например, десятичные цифры представлены в коде «8, 4, 2, 1» и в таком же коде должен быть получен результат. Составим таблицу десятичных сумм, их представлений в коде «8, 4, 2, 1», корректирующих поправок и десятичного переноса (табл. 49). Здесь сумма i -х десятичных цифр, полученная на первом этапе, представлена четырьмя разрядами двоичной суммы $s'_{i,j}$ ($j = 1, 2, 3, 4$) и разрядом двоичного переноса $p'_{i,4}$. Поразрядные значения коррекции $c_{i,j}$ в этом случае определяются двоичной разностью слова $p'_{i,4} s'_{i,4} s'_{i,3} s'_{i,2} s'_{i,1}$, являющегося правильным изображением десятичной суммы в коде «8, 4, 2, 1», и слова $p'_{i,4} s'_{i,4} s'_{i,3} s'_{i,2} s'_{i,1}$. Считая функции $c_{i,j}$ неполностью определенными функциями аргументов $p'_{i,4}, s'_{i,4}, s'_{i,3}, s'_{i,2}, s'_{i,1}$, можно легко найти их минимальные формы. В данном случае

$$c_{i,4} = c_{i,1} = 0;$$

$$c_{i,3} = c_{i,2} = p'_{i,4} \vee s'_{i,4} s'_{i,3} \vee s'_{i,4} s'_{i,2}.$$

Таблица 49

Десятичная сумма	Код суммы, полученной на первом этапе					Сумма в коде «8», 1, 2, 1 ²					Знак кор- рекции	Корректирующая поправка			
	$p'_{i,4}$	$s'_{i,4}$	$s'_{i,3}$	$s'_{i,2}$	$s'_{i,1}$	$p_{i,4}$	$s_{i,4}$	$s_{i,3}$	$s_{i,2}$	$s_{i,1}$		$c_{i,4}$	$c_{i,3}$	$c_{i,2}$	$c_{i,1}$
0	0	0	0	0	0	0	0	0	0	0	+	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	+	0	0	0	0
2	0	0	0	1	0	0	0	0	1	0	+	0	0	0	0
3	0	0	0	1	1	0	0	0	1	1	+	0	0	0	0
4	0	0	1	0	0	0	0	1	0	0	+	0	0	0	0
5	0	0	1	0	1	0	0	1	0	1	+	0	0	0	0
6	0	0	1	1	0	0	0	1	1	0	+	0	0	0	0
7	0	0	1	1	1	0	0	1	1	1	+	0	0	0	0
8	0	1	0	0	0	0	1	0	0	0	+	0	0	0	0
9	0	1	0	0	1	0	1	0	0	1	+	0	0	0	0
10	0	1	0	1	0	1	0	0	0	0	+	0	1	1	0
11	0	1	0	1	1	1	0	0	0	1	+	0	1	1	0
12	0	1	1	0	0	1	0	0	1	0	+	0	1	1	0
13	0	1	1	0	1	1	0	0	1	1	+	0	1	1	0
14	0	1	1	1	0	1	0	1	0	0	+	0	1	1	0
15	0	1	1	1	1	1	0	1	0	1	+	0	1	1	0
16	1	0	0	0	0	1	0	1	1	0	+	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1	+	0	1	1	0
18	1	0	0	1	0	1	1	0	0	0	+	0	1	1	0
19	1	0	0	1	1	1	1	0	0	1	+	0	1	1	0

Из табл. 49 видно, что если результат, полученный на первом этапе, находится в пределах от 0 до 9, то никакой коррекции не требуется. Если же этот результат находится в пределах от 10 до 19, то коррекция состоит в прибавлении слова 0110. Десятичный перенос в старшую тетраду в данном случае может быть получен двумя способами: как двоичный перенос на втором этапе и как функция $c_{i,3}$ или $c_{i,2}$ (это видно из табл. 49, где $c_{i,3} = c_{i,2} = p_{i,4}$).

С помощью аналогичной таблицы можно убедиться, что, например, для двоично-десятичного кода с избытком 3

$$c_{i,4} = c_{i,3} = \bar{p}'_{i,4}; c_{i,2} = p_{i,4}; c_{i,1} = 1,$$

а перенос в старший десятичный разряд совпадает с переносом $p'_{i,4}$. С учетом вышеизложенного легко построить схемы одноразрядных двоично-десятичных сумматоров. Для кода «8, 4, 2, 1» сумматор представлен на рис. 3.41, а. Здесь левый ряд сумматоров реализует первый этап сложения. Наличие сигнала переноса на выходе старшего двоичного разряда означает, что результат первого этапа больше или

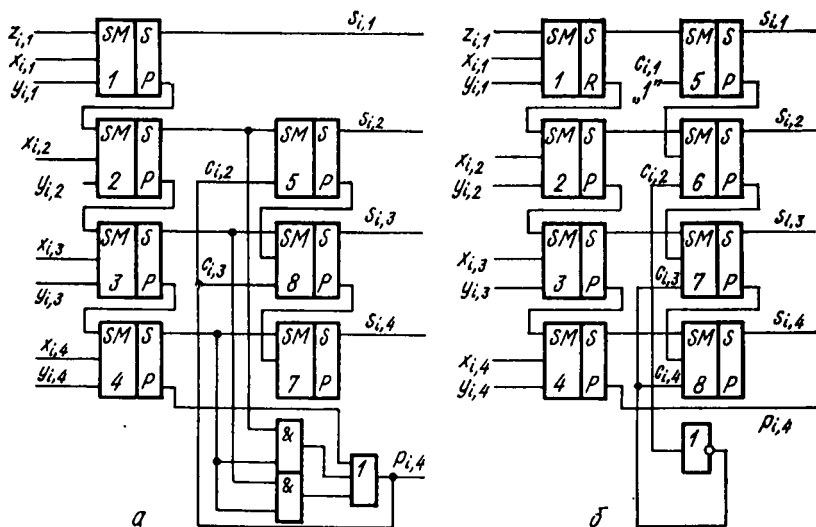


Рис. 3.41

равен 16. Кроме того, перенос в старший десятичный разряд должен формироваться в тех случаях, когда сумма больше или равна 10, т. е. когда ее двоичное представление будет иметь вид 1010, 1011, 1100, 1101, 1110, 1111. Для обнаружения таких слов необходима схема из двух элементов И. Второй ряд сумматоров предназначен для прибавления корректирующей поправки.

Для кода с избытком 3 схема десятичного сумматора представлена на рис. 3.41, б. Как и в предыдущей схеме левый ряд сумматоров реализует первый этап десятичного сложения. Правый ряд сумматоров производит либо прибавление слова 0011 (если имеется перенос из данного десятичного разряда), либо прибавление слова 1101 (если сигнал переноса равен 0).

Приведенные схемы можно упростить, если в сумматоре для кода «8, 4, 2, 1» заменить одноразрядные сумматоры 5 и 7 полусумматорами. Кроме того, из сумматора 7 можно исключить элементы, формирующие перенос. Если, например, каждый двоичный сумматор выполнен на элементах И—НЕ по схеме на рис. 3.36, б, то в таком

случае оценки сложности и быстродействия сумматора на рис. 3.41, а имеют вид

$$L = 54; K = 135; t_s = 20t_3; t_p = 13t_3.$$

В схеме на рис. 3.41, б сумматор 5 может быть заменен инвертором, так как этот сумматор реализует сложение с единицей, а $s_{i,1} + 1 \pmod{2} = \bar{s}_{i,1}$. На вход переноса следующего сумматора при такой замене необходимо подавать непосредственно сигнал $\bar{s}_{i,1}$, поскольку $p(\bar{s}_{i,1}, 1) = s_{i,1}$. В сумматоре 8 можно исключить цепи переноса. Для такого сумматора при указанных выше предположениях

$$L = 58; K = 153; t_s = 20t_3; t_p = 12t_3.$$

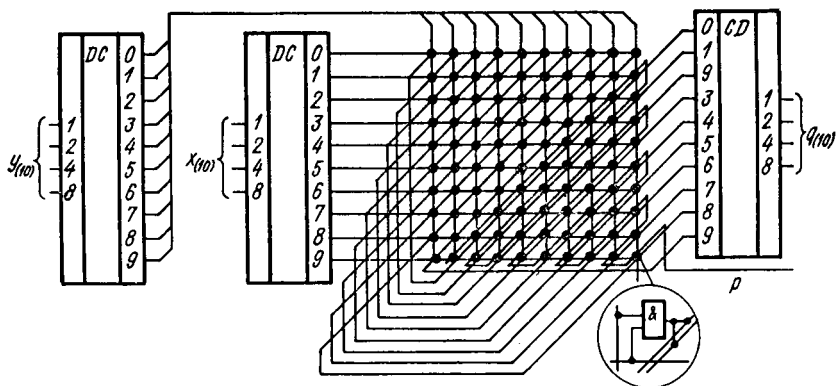


Рис. 3.42

В связи с развитием интегральной технологии возрос интерес к так называемым матричным схемам десятичных сумматоров. Такие схемы могут быть построены с помощью формальных методов теории многозначных переключательных функций. На рис. 3.42 приведена схема матричного десятичного сумматора, построенная без учета сигналов переноса из младшего десятичного разряда. Здесь каждому

узлу решетки шин $x_{(10)}$ и $y_{(10)}$ соответствует элемент И, подключенный своими входами к вертикальным и горизонтальным шинам, а выходом — к диагональной шине. Эти элементы могут быть реализованы, например, на основе двухэмиттерных транзисторов или униполярных транзисторов. Выходы элементов И должны быть подключены к выходным шинам через разделительные диоды для устранения возможности возникновения ложных сигналов. Шифратор СД пре-

Таблица 50

		$y_{(10)}$									
		0	1	2	3	4	5	6	7	8	9
$x_{(10)}$	0	0	1	2	3	4	5	6	7	8	9
	1	1	2	3	4	5	6	7	8	9	0
	2	2	3	4	5	6	7	8	9	0	1
	3	3	4	5	6	7	8	9	0	1	2
	4	4	5	6	7	8	9	0	1	2	3
	5	5	6	7	8	9	0	1	2	3	4
	6	6	7	8	9	0	1	2	3	4	5
	7	7	8	9	0	1	2	3	4	5	6
	8	8	9	0	1	2	3	4	5	6	7
	9	9	0	1	2	3	4	5	6	7	8

Таблица 51

образует пространственный десятичный код в двоично-десятичный. Выходы элементов И в узлах решетки объединяются выходными шинами в соответствии с квадратными таблицами истинности функций $x + y \pmod{10}$ и переносов при десятичном сложении (табл. 50 и 51). Полный одноразрядный десятичный сумматор может быть построен на основе рассмотренной схемы при включении между одним из входных дешифраторов и решеткой сдвигателя, управляемого сигналом z_i . Сточки зрения интегральной технологии матричный сумматор удобен тем, что в нем все элементы просты и однородны, а связи между ними регулярны. Матричный десятичный сумматор, изготовленный в виде БИС, характеризуется очень высоким значением так называемого показателя функционального разбиения β — отношения числа B эквивалентных элементов (вентилей) в БИС к числу v внешних выводов БИС. Для полного одноразрядного десятичного сумматора на основе схемы на рис. 3.42 $\beta \approx 10,3$, а для схем на рис. 3.41 — $\beta \approx 4$.

		$y_{(10)}$									
		0	1	2	3	4	5	6	7	8	9
$x_{(10)}$	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	1
	2	0	0	0	0	0	0	0	0	1	1
	3	0	0	0	0	0	0	0	1	1	1
	4	0	0	0	0	0	0	1	1	1	1
	5	0	0	0	0	0	1	1	1	1	1
	6	0	0	0	0	1	1	1	1	1	1
	7	0	0	0	1	1	1	1	1	1	1
	8	0	0	1	1	1	1	1	1	1	1
	9	0	1	1	1	1	1	1	1	1	1

3.6. ОРГАНИЗАЦИЯ ЦЕПЕЙ ПЕРЕНОСА В ПАРАЛЛЕЛЬНЫХ СУММАТОРАХ

Параллельный сумматор в простейшем случае представляет собой n одноразрядных сумматоров, последовательно — от младших разрядов к старшим — соединенных цепями переноса. Однако такая схема сумматора характеризуется сравнительно невысоким быстродействием, так как формирование сигналов s_i^j и p_i в каждом i -м разряде производится лишь после того, как поступит сигнал переноса с $(i-1)$ -го разряда. Таким образом, быстродействие сумматора определяется временем прохождения сигнала по цепи переноса. Поэтому при построении параллельного сумматора следует особое внимание обращать на реализацию цепи переноса. Для повышения скорости распространения переноса применяют самые различные средства: используют в цепи переноса наиболее быстродействующие элементы, тщательно выполняют монтаж без длинных проводников и паразитных емкостей, применяют специальные структурные методы ускорения прохождения сигнала переноса. Рассмотрим некоторые из них.

Сущность самых простых структурных методов составляет такая реализация цепи переноса, при которой между входом z_i и выходом p_i одноразрядного сумматора оказывается наименьшее число логических уровней. С этой целью в выражении (3.9) выполним склеивание крайних дизъюнктивных членов, а из средних вынесем за скобки общую часть:

$$p_i = x_i y_i \bar{z}_i \vee x_i \bar{y}_i z_i \vee \bar{x}_i y_i z_i \vee x_i y_i z_i = x_i y_i \vee z_i (x_i \bar{y}_i \vee \bar{x}_i y_i) = D_i \vee z_i \bar{D}_i \bar{z}_i \bar{F}_i,$$

где $D_i = x_i y_i$; $F_i = x_i y_i \vee \bar{x}_i y_i$. При реализации функции p_i по полученному выражению не требуются сигналы $\bar{z}_i$ и, кроме того, сигнал z_i проходит через два элемента (И—ИЛИ либо И—НЕ/И—НЕ). На другие входы этих элементов подаются сигналы D_i и F_i , которые зависят только от x_i и y_i и, следовательно, могут быть сформированы по всем разрядам параллельного сумматора одновременно. Параллельные сумматоры, у которых цепь переноса выполнена таким способом, называют также сумматорами со сквозным переносом. Задержка t_p для сумматора со сквозным переносом составляет $2t_3$.

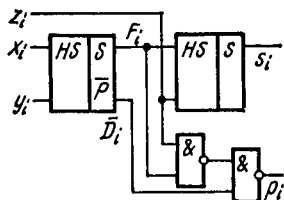


Рис. 3.43

Функции D_i и F_i совпадают с функциями, реализуемыми полусумматорами. Поэтому цепи сквозного переноса могут быть построены с минимальными аппаратными затратами в многоразрядных сумматорах на основе полусумматоров (рис. 3.43). Выход p второго сумматора в этом случае не используется.

В цифровых ЭВМ на интегральной элементной базе получил распространение еще один способ ускорения переноса в параллельных сумматорах, при котором $t_p = t_3$. При наличии элементов ИЛИ—НЕ функцию p_i и ее отрицание можно реализовать по выражениям

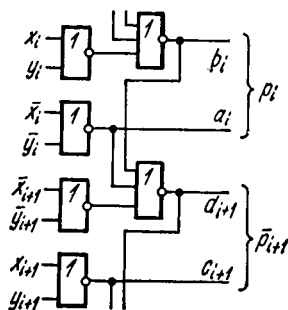


Рис. 3.44

$$\begin{aligned} p_i &= x_i y_i \vee x_i z_i \vee y_i z_i = \\ &= \overline{x_i y_i} \vee \overline{z_i (x_i \vee y_i)} = (\bar{x}_i \vee \bar{y}_i) \vee \\ &\quad \vee (\bar{z}_i \vee (x_i \vee y_i)); \bar{p}_i = \\ &= \bar{x}_i \bar{y}_i \vee \bar{x}_i \bar{z}_i \vee \bar{y}_i \bar{z}_i = \overline{\bar{x}_i \bar{y}_i} \vee \overline{z_i (\bar{x}_i \vee \bar{y}_i)} = \\ &= (x_i \vee y_i) \vee (z_i \vee (\bar{x}_i \vee \bar{y}_i)). \end{aligned}$$

Из этих выражений следует, что одним из аргументов функции p_i является $\bar{z}_i$, а функции $\bar{p}_i$ — z_i . В связи с этим достаточно в одних разрядах (например, нечетных) формировать только сигналы p_i , а в соседних (четных) — только сигналы $\bar{p}_i$. Введем обозначения:

$$\bar{x}_i \vee \bar{y}_i = a_i; \quad \bar{z}_i \vee (x_i \vee y_i) = b_i; \quad x_i \vee \bar{y}_i = c_i; \quad z_i \vee (\bar{x}_i \vee \bar{y}_i) = d_i.$$

Тогда $p_i = a_i \vee b_i$, $\bar{p}_i = c_i \vee d_i$. В выражения для p_i и $\bar{p}_i$ переменные z_i и $\bar{z}_i$ входят как отдельные дизъюнктивные члены. Поэтому можно записать, что

$$\begin{aligned} p_{i+1} &= (\bar{x}_{i+1} \vee \bar{y}_{i+1}) \vee (c_i \vee d_i \vee (x_{i+1} \vee y_{i+1})) = a_{i+1} \vee b_{i+1}; \\ \bar{p}_{i+1} &= (x_{i+1} \vee y_{i+1}) \vee (a_i \vee b_i \vee (\bar{x}_{i+1} \vee \bar{y}_{i+1})) = c_{i+1} \vee d_{i+1}. \end{aligned}$$

Отсюда видно, что сигналы a_i , b_i , c_i , d_i , представляющие функции $\bar{p}_i$ и p_i , проходят в $(i+1)$ -м разряде только через один элемент ИЛИ—НЕ. Так как сигналы p_i и $\bar{p}_i$ в каждом разряде в сущности представляются двумя сигналами (a_i , b_i и c_i , d_i), то такая схема (рис. 3.44) получила название двухпроводной передачи сигналов пе-

реноса. При таком выполнении цепи переноса функции s_i должны строиться по выражениям, где в качестве аргумента в нечетных разрядах используется $\bar{z}_i$, а в четных разрядах — z_i .

Эффективным способом повышения быстродействия параллельных сумматоров является применение так называемого одновременного переноса. При всех рассмотренных выше способах организации цепей переноса функция p_i строилась как функция цифр слагаемых в i -м разряде и переноса в i -й разряд, т. е. $p_i = p(x_i, y_i, z_i)$. Для случая же одновременного переноса

$$p_i = p(x_i, y_i, x_{i-1}, y_{i-1}, \dots, x_1, y_1, z_1),$$

т. е. перенос p_i является функцией цифр слагаемых во всех i разрядах и переноса в первый разряд. Возможность такого выполнения цепей переноса видна из следующих преобразований. Пусть

$$p_i = D_i \vee z_i F_i.$$

Но, в свою очередь,

$$z_i = p_{i-1} = D_{i-1} \vee z_{i-1} F_{i-1},$$

следовательно,

$$p_i = D_i \vee F_i D_{i-1} \vee F_i F_{i-1} z_{i-1}.$$

Аналогично рассуждая, запишем

$$z_{i-1} = p_{i-2} = D_{i-2} \vee z_{i-2} F_{i-2}$$

и далее

$$\begin{aligned} p_i &= D_i \vee F_i D_{i-1} \vee F_i F_{i-1} D_{i-2} \vee F_i F_{i-1} F_{i-2} z_{i-2}; \\ p_i &= D_i \vee F_i D_{i-1} \vee F_i F_{i-1} D_{i-2} \vee \dots \vee F_i F_{i-1} \dots F_2 D_1 \vee \\ &\quad \vee F_i F_{i-1} \dots F_2 F_1 z_1. \end{aligned}$$

Если цепь переноса в каждый разряд построить по приведенному выражению, то формирование сигналов переноса происходило бы по всем разрядам одновременно за $2 t_3$. Однако для этого необходимы элементы на $n + 1$ входов и с коэффициентом разветвления, примерно равным $\frac{n^2}{4}$, так как число дизъюнктивных членов в выражениях для p_i растет очень быстро. Поэтому такой способ построения цепей переноса реализуем лишь при небольшой разрядности слагаемых. На рис. 3.45 приведен фрагмент схемы сумматора с одновременным переносом. Здесь схемы $M2$ сложения по модулю 2 во всех разрядах, за исключением первого, построены так, что на их входах действуют сигналы $\bar{p}_i$ (например, по выражению (3.15)).

Основная идея одновременного переноса — формирование сигналов p_i независимо от z_i — используется и в сумматорах с групповым (частично-групповым) переносом, где все разряды сумматора разбивают на группы по m разрядов в каждой. При этом m не превышает коэффициента объединения логических элементов, на которых проектируют сумматор. Возможны следующие варианты организации группового переноса:

- 1) с последовательным или сквозным переносом в группах и одновременным между группами;
- 2) с одновременным переносом в группах и последовательным между группами;
- 3) с одновременным переносом в группах и сквозным между группами;
- 4) с одновременным переносом в группах и одновременным между группами.

Группа разрядов ($m = 3$) сумматора со сквозным переносом между разрядами в группе и с одновременным переносом между группами

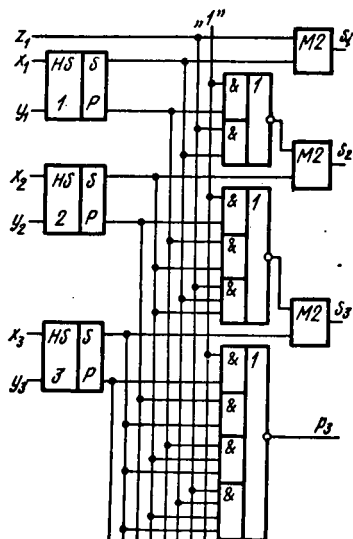


Рис. 3.45

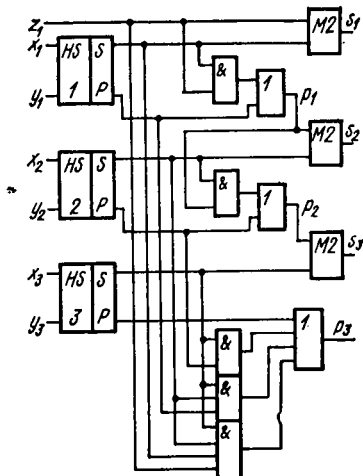


Рис. 3.46

показана на рис. 3.46. В многоразрядном сумматоре p_3 (в общем случае p_m) каждой такой группы соединяется со входом z_1 следующей группы. Таким образом, задержка переноса в группе разрядов будет составлять $2t_3(m-1)$.

Вместе с тем задержка в последовательно соединенных цепях одновременного переноса между группами составит $2\left(\frac{n}{m}-1\right)t_3$. Очевидно, что чем больше m , тем меньше задержка в цепи одновременного переноса и больше задержка внутри групп. Поэтому для рассматриваемого сумматора

$$t_p = 2t_3 \left(\left(\frac{n}{m} - 1 \right) + (m - 1) \right).$$

Найдем оптимальную величину m , обеспечивающую минимальное значение t_p ,

$$\frac{dt_p}{dm} = -\frac{n}{m^2} + 1 = 0.$$

Отсюда имеем $m = \sqrt{n}$. При таком m

$$t_p = 4 t_3 (\sqrt{n} - 1).$$

При последовательном переносе в группах и одновременном между группами

$$t_p = 2 t_3 \left(\frac{n}{m} - 1 \right) + 3 t_3 (m - 1),$$

а оптимальное значение m равно $\sqrt{\frac{2n}{3}}$.

В случае одновременного переноса в группах и последовательного переноса между ними каждая группа выполняется по аналогии со схемой на рис. 3.45, а выход p_m соединяется со входом z_1 следующей группы. Тогда

$$t_p = 2 t_3 \frac{n}{m}.$$

Сумматоры с одновременным переносом в группах и одновременным либо сквозным переносом между группами называют также сверхпараллельными сумматорами. В таких сумматорах стремятся выполнить максимальное число подготовительных операций в старших разрядах сумматора до поступления сигнала переноса из младших разрядов. В рассмотренных ранее сумматорах с этой целью формировались функции D_i и F_i , которые в дальнейшем будем называть подготовительными функциями нулевого порядка. На основе функций D_i и F_i можно сформировать подготовительные функции первого порядка, которыми будем называть выражения вида

$$\begin{aligned} D_{1,m}^1 &= D_m \vee F_m D_{m-1} \vee F_m F_{m-1} D_{m-2} \vee \dots \vee F_m F_{m-1} \dots F_2 D_1; \\ F_{1,m}^1 &= F_m F_{m-1} \dots F_2 F_1. \end{aligned}$$

Легко заметить, что перенос из данной группы, состоящей из m разрядов, может быть представлен в этом случае следующим выражением:

$$p_m = D_{1,m}^1 \vee F_{1,m}^1 z_1. \quad (3.21)$$

Если функции D_1^1 и $F_{1,m}^1$ реализовать с помощью отдельных схем, то перенос из данной группы в следующую в соответствии с выражением (3.21) может быть получен как сквозной перенос. При наличии же m групп разрядов (т. е. m пар подготовительных функций первого порядка) перенос из такой m^2 -разрядной группы может быть сформирован как одновременный перенос

$$\begin{aligned} p_{m^2} &= D_{m^2-m, m^2}^1 \vee D_{m^2-2m, m^2-m}^1 F_{m^2-m, m^2}^1 \vee \dots \\ &\dots \vee F_{m^2-m, m^2}^1 F_{m^2-2m, m^2-m}^1 \dots F_{1, m}^1 z_1. \end{aligned}$$

При построении таких сумматоров целесообразно проводить разбиение на группы не всех разрядов, а лишь тех, которые старше некоторого l -го разряда. Величина l зависит от соотношения между скоростью распространения сигналов переноса и скоростью сигналов в цепях функций D^1 и F^1 . При этом l надо выбирать так, чтобы время

прохождения сигнала переноса через l младших разрядов было равно времени формирования функций D^1 и F^1 . В зависимости от типа используемых элементов на практике $l = 1 - 6$. Через l групп разрядов, начиная с первой, т. е. через $lm + l$ разрядов от самого младшего, имеет смысл объединять группы разрядов в новые группы и для каждой такой группы групп формировать подготовительные функции второго порядка D^2 и F^2 . Эти функции строятся по тем же формулам, что и функции первого

порядка. Отличие состоит в том, что в левой части этих выражений будут функции второго порядка, а в правой — функции первого порядка. Отступив затем еще на l групп, объединяю-

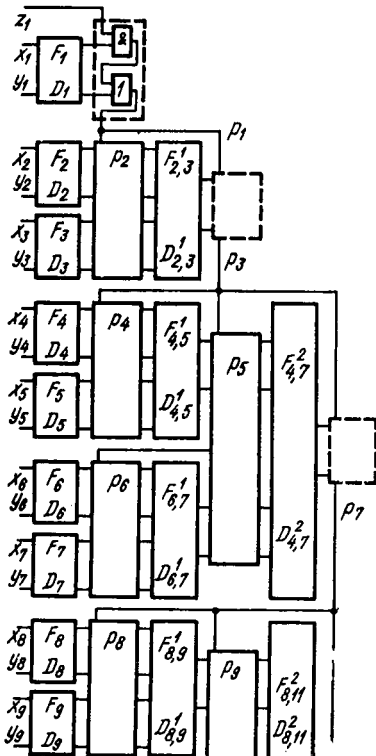


Рис. 3.47

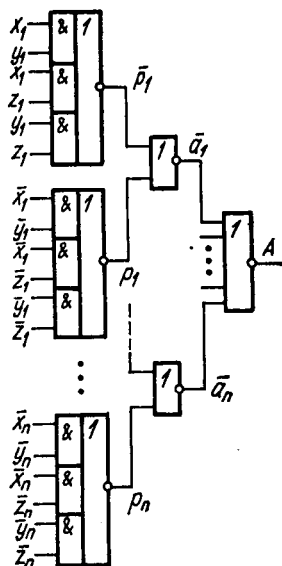


Рис. 3.48

щих группы по m разрядов, можно формировать подготовительные функции третьего порядка и т. д. Однако на практике, как правило, необходимости в формировании функций третьего порядка не возникает. Пример организации цепи переноса сверхпараллельного сумматора со сквозным переносом между группами показан на рис. 3.47. На этой схеме $n = 9$, $l = 1$, $m = 2$. Задержка сигналов переноса в подобных сумматорах пропорциональна количеству ступеней в схеме формирования подготовительных функций наивысшего порядка, т. е. величине $\log_m n$.

Во всех рассмотренных выше многоразрядных сумматорах время суммирования определялось длительностью прохождения сигналов по цепям переноса в самом неблагоприятном случае (перенос возникает в младшем разряде и распространяется через все разряды). Од-

Таким образом, такой случай достаточно маловероятен. Если считать, что вероятности появления любой из k цифр в разрядах слагаемых равны $\frac{1}{k}$, то среднее число разрядов, на которые распространяется перенос при суммировании n -разрядных чисел составляет $\log_k n$, где k — основание системы счисления. Поэтому среднее время распространения переносов в сумматоре в $\frac{n}{\log_k n}$ раз меньше максимального. На этом и основан один из способов повышения быстродействия сумматоров, заключающийся в построении специальной схемы, фиксирующей момент окончания распространения переносов. Поскольку время суммирования в этом случае зависит от конкретных значений слагаемых, то такой сумматор получил название асинхронного.

При проектировании двоичных асинхронных сумматоров исходят из того, что суммирование в i -м разряде двух единиц приводит к возникновению переноса, проходящего через старшие разряды, если только комбинация цифр слагаемых в них будут 01, 10 и 11. При наличии двух нулей в i -м разряде перенос не возникает ни в i -м разряде, ни в старших разрядах с комбинациями цифр слагаемых 01 и 10. Поэтому разряды с цифрами 11 и 00 могут быть источниками сигналов наличия p_i и отсутствия $\bar{p}_i$ переноса. Если один из сигналов p_i или $\bar{p}_i$ принимает единичное значение, то это означает, что к рассматриваемому моменту времени суммирование в i -м разряде закончено. Таким образом, окончанию суммирования в одном разряде соответствует равенство 1 функции

$$a_i = p_i \vee \bar{p}_i,$$

а окончанию суммирования в n -разрядном сумматоре — равенство 1 функции

$$A = \bigwedge_{i=1}^n a_i.$$

Функции p_i и $\bar{p}_i$ при этом должны быть реализованы независимыми схемами, например, по выражениям

$$\begin{aligned} p_i &= \overline{x_i y_i} \vee \overline{x_i z_i} \vee \overline{y_i z_i}; \\ \bar{p}_i &= x_i y_i \vee x_i z_i \vee y_i z_i. \end{aligned}$$

Схема реализации функции A в асинхронном сумматоре показана на рис. 3.48.

3.7. ПРЕОБРАЗОВАТЕЛИ КОДОВ

Класс операционных узлов, называемых в ЦВТ преобразователями кодов, весьма многочислен по функциональным типам. К преобразователям кодов относят схемы для получения заданного представления отрицательных чисел, для перевода чисел из одной системы счисления в другую, схемы для управления знаковыми индикаторами и др. Рассмотрим примеры построения некоторых из них.

Преобразователем прямого кода в обратный будем называть комбинационную схему с $n + 1$ входами и n выходами, выполняющую

операцию (1.10). Как показано в п. 1.5, получение обратного кода X в случае $X_0 = 1$ сводится к выполнению n поразрядных операций инвертирования. Поэтому можно записать, что

$$f(x_i) = x_i \bar{X}_0 \vee \bar{x}_i X_0.$$

Для реализации преобразователя на элементах И — НЕ удобна следующая форма функции $f(x_i)$:

$$f(x_i) = \overline{(x_i \bar{X}_0) (\bar{x}_i X_0)}.$$

Таким образом, для многоразрядного преобразователя

$$L = 3n; K = 6n; t = 2t_3;$$

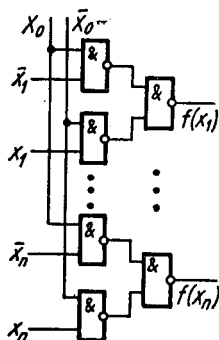


Рис. 3.49

Пример схемы преобразователя прямого кода в обратный приведен на рис. 3.49. Простота схемы такого преобразователя послужила причиной того, что он редко выполняется в виде функционально обособленной схемы, а чаще встраивается в другие операционные узлы, например регистры.

Преобразователем прямого кода в дополнительный будем называть схему с $n + 1$ входами и n выходами, выполняющую операцию (1.12). Для получения дополнительного кода в случае $X_0 = 1$ в i -м разряде выполняется операция

$$g(x_i) = \begin{cases} x_i, & \text{если все } x_{i-r} = 0; \\ \bar{x}_i, & \text{если } x_{i-r} \neq 0; \end{cases}$$

$$r = 1, 2, \dots, i-1.$$

Следовательно, можно записать, что

$$g(x_i) = x_i \bar{X}_0 \vee \bar{x}_i X_0 (\bigvee_{j=1}^{i-1} x_j) \vee x_i X_0 (\bigvee_{j=1}^{i-1} \bar{x}_j).$$

Учитывая, что

$$\bigvee_{j=1}^{i-1} x_j = \bigwedge_{j=1}^{i-1} \bar{x}_j,$$

последнее выражение можно несколько упростить:

$$g(x_i) = x_i \bar{X}_0 \vee \bar{x}_i X_0 (\bigvee_{j=1}^{i-1} x_j) \vee x_i X_0 (\bigwedge_{j=1}^{i-1} \bar{x}_j).$$

Такому представлению функции $g(x_i)$ соответствует схема на рис. 3.50, а. Эта схема реализует функции

$$g(x_1) = x_1 \bar{X}_0 \vee x_1 X_0 = x_1;$$

$$g(x_2) = x_2 \bar{X}_0 \vee \bar{x}_2 X_0 x_1 \vee x_2 X_0 \bar{x}_1 = (x_2 \bar{X}_0) (\overline{(x_2 X_0 x_1)}) (\overline{(x_2 X_0 \bar{x}_1)});$$

$$g(x_3) = x_3 \bar{X}_0 \vee \bar{x}_3 X_0 (x_1 \vee x_2) \vee x_3 X_0 \bar{x}_1 \bar{x}_2 = (x_3 \bar{X}_0) (\overline{(\bar{x}_3 X_0 (\bar{x}_1 \bar{x}_2))}) = (x_3 X_0 \bar{x}_1 \bar{x}_2);$$

$$g(x_4) = x_4 \bar{X}_0 \vee x_4 X_0 (x_1 \vee x_2 \vee x_3) \vee x_4 X_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 =$$

$$= (x_4 \bar{X}_0) (\overline{(\bar{x}_4 X_0 (\bar{x}_1 \bar{x}_2 \bar{x}_3))}) (x_4 X_0 \bar{x}_1 \bar{x}_2 \bar{x}_3).$$

Для такой схемы при произвольном n

$$L = 5n - 6; K = n^2 + 9n - 11; t = 3t_3.$$

Существенным недостатком такой реализации преобразователя прямого кода в дополнительный является рост числа входов некоторых логических элементов с ростом n , что затрудняет выполнение требования наращиваемости структуры преобразователя. Для устранения

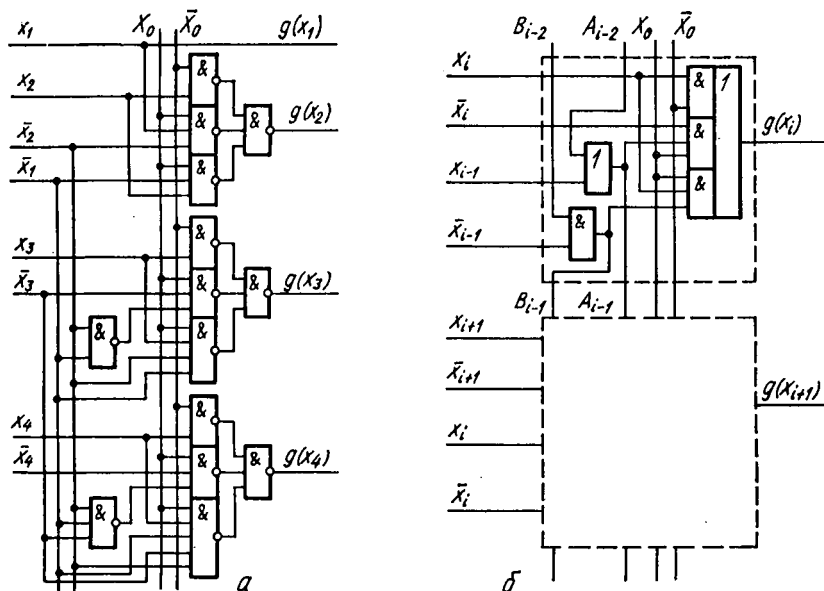


Рис. 3.50

этого недостатка можно поступить следующим образом. Введем обозначения:

$$A_i = x_i \vee A_{i-1}; B_i = \bar{x}_i B_{i-1}; \\ A_0 = 0; B_0 = 1.$$

Тогда выражение для $g(x_i)$ можно записать так:

$$g(x_i) = x_i \bar{X}_0 \vee \bar{x}_i X_0 A_{i-1} \vee x_i X_0 B_{i-1}.$$

Этому выражению соответствует схема на рис. 3.50, б. Структура такого преобразователя регулярна и может быть построена на однотипных интегральных модулях (обведены штриховой линией на рис. 3.50, б). Для этой схемы -

$$L = 6n - 6; K = 15n - 15; t = (n + 1) t_3.$$

Таким образом, сложность преобразователя прямого кода в дополнительный примерно в два раза выше сложности преобразователя прямого кода в обратный. Поэтому в цифровых ЭВМ дополнительный код часто получают путем прибавления единицы к младшему разряду обратного кода. Выполняется такая операция в АЛУ.

При разработке вычислительных устройств, работающих в десятичной системе счисления, часто требуются преобразователи одного двоично-десятичного кода в другой (см. п. 3.5). Особенности проектирования таких преобразователей рассмотрим на примере построения преобразователя кода «2, 4, 2, 1», не обладающего свойством аддитивности, в аддитивный код «8, 4, 2, 1». Функционирование такого преобразователя описывается табл. 52. Считая функции y_4, y_3, y_2, y_1 неполностью определенными функциями аргументов x_4, x_3, x_2, x_1 , нетрудно убедиться (например, с помощью диаграмм Вейча), что

$$y_4 = x_3 x_2 = \overline{\overline{x_3 x_2}}; \quad y_3 = x_3 \bar{x}_2 \vee x_4 \bar{x}_3 = \overline{\overline{(x_3 x_2) (x_4 x_3)}};$$

$$y_2 = x_4 \bar{x}_2 \vee \bar{x}_4 x_2 = \overline{\overline{(x_4 x_2) (x_4 x_2)}}; \quad y_1 = x_1.$$

Таблица 52

Десятич- ные цифры	Код «2, 4, 2, 1»				Код «8, 4, 2, 1»				Семисегментный код						
	x_4	x_3	x_2	x_1	y_4	y_3	y_2	y_1	z_7	z_6	z_5	z_4	z_3	z_2	z_1
0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0	1	0	1	0	0	1	0	0
2	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1
3	0	0	1	1	0	0	1	1	1	1	0	1	1	0	1
4	0	1	0	0	0	1	0	0	0	1	0	1	1	1	0
5	1	0	1	1	0	1	0	1	1	1	0	1	0	1	1
6	1	1	0	0	0	1	1	0	1	1	1	1	0	1	1
7	1	1	0	1	0	1	1	1	0	1	0	0	1	0	1
8	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1
9	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1

На рис. 3.51, а показано условное обозначение такого преобразователя на функциональных схемах, а на рис. 3.51, б — его схемная реализация, для которой

$$L = 8; K = 15; t = 2t_3.$$

Преобразователь позиционного кода в унитарный (см. п. 3.4) имеет n входов, $2^n - 1$ выходов и реализует $2^n - 1$ функций

$$u_j(X) = \begin{cases} 0 & \text{при } X < j; \\ 1 & \text{при } X \geq j, \end{cases} \quad (3.16)$$

где $X = \sum_{i=1}^n x_i 2^{i-1}$; $x_i \in \{0, 1\}$; $j = 1, 2, \dots, 2^n - 1$. Такой преобразователь легко реализуем по схеме дешифратор-шифратор, когда используется промежуточное пространственное представление информации. В данном случае каждая функция $u_j(X)$ может быть представлена как

$$u_j(X) = \bigvee_{s=j}^{2^n-1} D_s(X), \quad (3.17)$$

где $D_s(X)$ — функции дешифрации входного слова. Два варианта такого построения преобразователя показаны на рис. 3.52. Оценки сложности и быстродействия этих схем имеют вид

$$L = 2^n - 2 + L_{DC}; K = (2^n + 1)(2^{n-1} - 1) + K_{DC}, t = t_{DC} + t_3;$$

$$L = 2^n - 2 + L_{DC}; K = 2^{n+1} - 4 + K_{DC}; t = t_{DC} + (2^n - 2)t_3.$$

Заметим, что по схеме дешифратор—шифратор можно, в принципе, построить любой преобразователь кодов. Однако получаемые

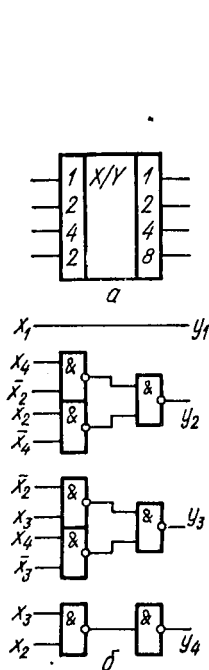


Рис. 3.51

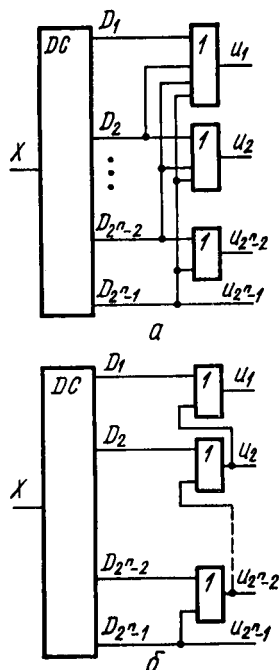


Рис. 3.52

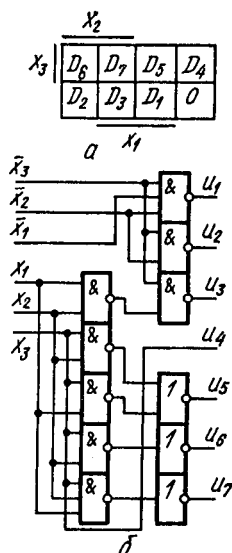


Рис. 3.53

при этом структуры часто не являются самыми экономичными и быстродействующими, несмотря на простоту методики их проектирования. Значительно лучшие результаты могут быть получены при использовании различных частных методик и приемов. Пусть, например, для преобразователя позиционного кода в унитарный $n = 3$. Нанесем на диаграмму Вейча функции $D_s(X)$ (рис. 3.53, а). Тогда, согласно (3.17) функцию $u_j(X)$ можно получить склеиванием тех D_s , индексы при которых больше или равны j . В этом случае

$$u_1 = x_3 \vee x_2 \vee x_1 = \overline{x_3 x_2 x_1}; u_2 = x_3 \vee x_2 = \overline{x_3 x_2};$$

$$u_3 = x_3 \vee x_2 x_1 = \overline{x_3 (x_2 x_1)}; u_4 = x_3, u_5 = x_3 x_2 \vee x_3 x_1 = \overline{(x_3 x_2) (x_3 x_1)};$$

$$u_6 = x_3 x_2 = \overline{x_3 x_2}; u_7 = x_3 x_2 x_1 = \overline{x_3 x_2 x_1}.$$

Схема преобразователя изображена на рис. 3.53, б. Для этого преобразователя

$$L = 11; K = 22; t = 2t_3,$$

что значительно лучше оценок преобразователя по схемам на рис. 3.52 при таком же n .

В аппаратуре отображения цифровой информации находят широкое применение преобразователи двоично-десятичных кодов в семи-элементные коды цифровых знаковых индикаторов. В таких индикаторах каждой десятичной цифре соответствует свечение определенной комбинации элементов (сегментов) $z_7, z_6, \dots, z_1$ (рис. 3.54, а). Рассмотрим построение такого преобразователя для случая, когда входным кодом является

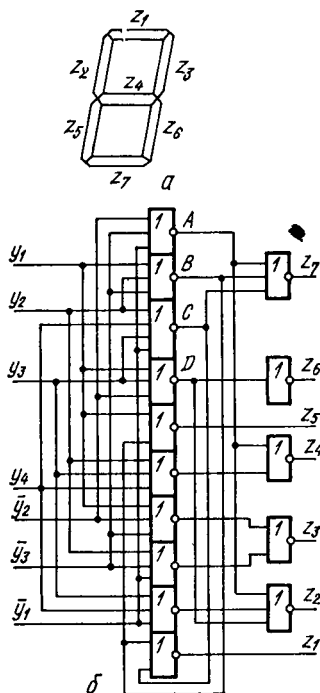


Рис. 3.54

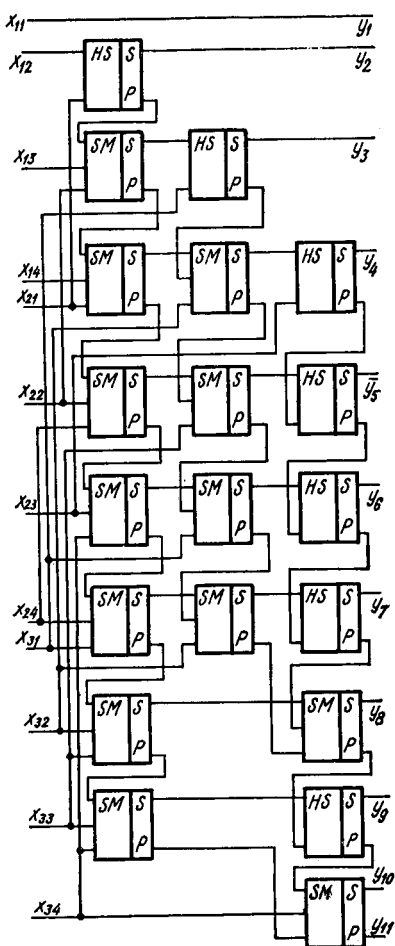


Рис. 3.55

код «8, 4, 2, 1». Тогда работу преобразователя можно описать с помощью табл. 52, где каждый набор значений функций $z_7, z_6, \dots, z_1$ определяется индицируемой десятичной цифрой. Считая $z_7, z_6, \dots, z_1$ системой неполностью определенных функций аргументов y_4, y_3, y_2, y_1 , получим

$$z_7 = (\bar{y}_3 \vee \bar{y}_2 \vee \bar{y}_1) \vee (\bar{y}_3 \vee y_2 \vee y_1) \vee (y_4 \vee y_3 \vee y_2 \vee \bar{y}_1) = \bar{A} \vee B \vee C;$$

$$\begin{aligned}
z_6 &= \overline{(y_3 \vee \bar{y}_2 \vee y_1)} = \bar{D}; \\
z_5 &= y_1 \vee \overline{(y_3 \vee y_2 \vee y_1)} = y_1 \vee B; \\
z_4 &= \overline{(\bar{y}_3 \vee \bar{y}_2 \vee \bar{y}_1) \vee (y_4 \vee y_3 \vee y_2)} = A \vee \overline{(y_4 \vee y_3 \vee y_2)}; \\
z_3 &= \overline{(y_3 \vee y_2 \vee y_1) \vee (\bar{y}_3 \vee y_2 \vee \bar{y}_1)}; \\
z_2 &= \overline{(y_4 \vee y_3 \vee \bar{y}_1) \vee (\bar{y}_3 \vee \bar{y}_2 \vee \bar{y}_1) \vee (y_3 \vee \bar{y}_2 \vee y_1)} = \\
&= A \vee D \vee \overline{(y_4 \vee y_3 \vee \bar{y}_1)}; \\
z_1 &= \overline{(\bar{y}_3 \vee y_2 \vee y_1) \vee (y_4 \vee y_3 \vee y_2 \vee y_1)} = B \vee C.
\end{aligned}$$

Схема преобразователя двоично-десятичного кода «8, 4, 2, 1» в семи-элементный код представлена на рис. 3.54, б.

Одними из важнейших операций, выполняемых при вводе и выводе информации в цифровых ЭВМ, являются операции преобразования (перевода) чисел из десятичной системы счисления в двоичную и наоборот. В высокопроизводительных универсальных и специализированных цифровых ЭВМ для выполнения этих операций часто используют аппаратные средства. Рассмотрим вначале построение комбинационных преобразователей из десятичной системы счисления в двоичную. Пусть, например, задано трехразрядное десятичное число

$$X = 10^2 x_3 + 10^1 x_2 + 10^0 x_1,$$

где $x_i \in \{0, 1, \dots, 9\}$. Обозначим $x_{i4}x_{i3}x_{i2}x_{i1}$ i -ю тетраду двоично-десятичного представления X , а вес каждого десятичного разряда в предыдущем выражении представим как сумму целых степеней числа 2:

$$\begin{aligned}
X &= (2^8 + 2^6 + 2^2)(x_{31} 2^3 + x_{33} 2^2 + x_{32} 2^1 + x_{31} 2^0) + (2^3 + 2^1) \times \\
&\times (x_{24} 2^3 + x_{23} 2^2 + x_{22} 2^1 + x_{21} 2^0) + (x_{14} 2^3 + x_{13} 2^2 + x_{12} 2^1 + x_{11} 2^0).
\end{aligned}$$

Раскрыв скобки и сгруппировав слагаемые по степеням числа 2, получим

$$\begin{aligned}
X &= 2^9 x_{34} + 2^8 (x_{34} + x_{33}) + 2^7 (x_{33} + x_{32}) + 2^6 (x_{32} + x_{31} + x_{24}) + \\
&+ 2^5 (x_{34} + x_{31} + x_{23}) + 2^4 (x_{33} + x_{24} + x_{22}) + 2^3 (x_{32} + x_{23} + \\
&+ x_{21} + x_{14}) + 2^2 (x_{31} + x_{22} + x_{13}) + 2^1 (x_{21} + x_{12}) + 2^0 x_{11}.
\end{aligned}$$

Таким образом, младший разряд числа в двоичной системе y_1 равен цифре x_{11} десятичного представления, второй разряд y_2 определяется двоичной суммой x_{21} и x_{12} , третий y_3 — суммой x_{31} , x_{22} , x_{13} и переноса из второго разряда и т. д. Структура преобразователя показана на рис. 3.55. Как видно из этого рисунка, для реализации преобразователя трехразрядных десятичных чисел в двоичные числа требуется 12 одноразрядных сумматоров и 7 полусумматоров.

Для построения преобразователя чисел из двоичной системы счисления в десятичную воспользуемся вышеописанным приемом, т. е. представим вес каждого двоичного разряда как сумму весов де-

сятных разрядов. Пусть, например, $n = 6$. Тогда число X можно записать в таком виде:

$$X = 2^5 x_6 + 2^4 x_5 + 2^3 x_4 + 2^2 x_3 + 2^1 x_2 + 2^0 x_1 = (3 \cdot 10^1 + 2 \cdot 10^0) \times \\ \times x_6 + (10^1 + 6 \cdot 10^0) x_5 + 8 \cdot 10^0 x_4 + 4 \cdot 10^0 x_3 + 2 \cdot 10^0 x_2 + \\ + 10^0 x_1 = 10^1 (2^1 + 2^0) x_6 + 10^0 2^1 x_6 + 10^1 2^0 x_5 + 10^0 (2^2 + 2^1) \times \\ \times x_5 + 10^0 2^3 x_4 + 10^0 2^2 x_3 + 10^0 2^1 x_2 + 10^0 2^0 x_1.$$

Раскрыв скобки и сгруппировав члены по степеням основания 10, получим

$$X = 10^1 (2^1 x_6 + 2^0 (x_6 + x_5)) + 10^0 (2^3 x_4 + 2^2 (x_5 + x_3) + 2^1 (x_6 + \\ + x_5 + x_2) + 2^0 x_1) = 10^1 (2^1 y_{22} + 2^0 y_{21}) + 10^0 (2^3 y_{14} + 2^2 y_{13} + \\ + 2^1 y_{12} + 2^0 y_{11}).$$

Отсюда видно, что значения десятичных цифр определяются выражениями в скобках, причем вычисление этих выражений должно производиться по правилам десятичной арифметики. Это значит, что каждый десятичный разряд необходимо снабдить схемой коррекции и выделения переноса в старшую тетраду. Схема коррекции и выделения переноса строится так же, как и для двоично-десятичного сумматора (см. п. 3.5). Структура шестизрядного преобразователя чисел из двоичной системы в десятичную приведена на рис. 3.56.

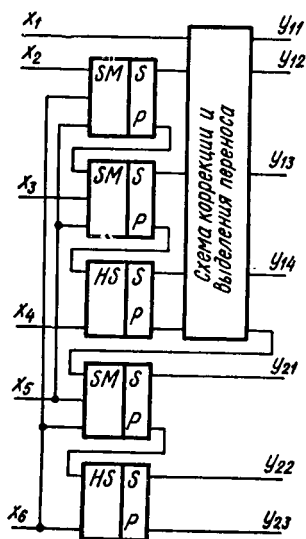


Рис. 3.56

Общим недостатком двух последних схем преобразователей является невозможность их построения на однотипных модулях, каждый из которых также представляет собой преобразователь определенной разрядности, так как эти схемы не обладают свойством наращиваемости структуры. Требование наращиваемости преобразователя чисел из десятичной системы в двоичную можно выполнить, если представить X в следующем виде:

$$X = 10^0 x_1 + 10^1 x_2 + 10^2 x_3 + \dots + 10^{n-2} x_{n-1} + 10^{n-1} x_n = x_1 + \\ + 10 (x_2 + 10 (x_3 + \dots + 10 (x_{n-1} + 10 (x_n + 0)) \dots)) = x_1 + \\ + (2^3 + 2^1) (x_2 + (2^3 + 2^1) (x_3 + \dots + (2^3 + 2^1) (x_{n-1} + (2^3 + \\ + 2^1) (x_n + 0)) \dots)).$$

Отсюда следует, что преобразование чисел сводится к n -кратному выполнению операции

$$A_i = 2^3 A_{i-1} + 2^1 A_{i-1} + x_{n+1-i}, \quad i = 1, 2, \dots, n, \quad (3.18)$$

при начальном условии $A_0 = 0$. Здесь x_{n+1-i} — десятичные цифры, представленные двоичными тетрадами, а вычисления проводятся по правилам двоичной арифметики (умножение на 2^1 и на 2^3 сводится

соответственно к сдвигу на 1 и на 3 разряда). Модуль для вычисления A_i может быть построен на основе сумматора (рис. 3.57, а), имеющего две группы входов — m входов для A_{i-1} и 3 входа для старших разрядов x_{n+1-i} . Младший разряд x_{n+1-i} в соответствии с (3.18) является младшим разрядом A_i . Если модуль предназначен для построения преобразователей n -разрядных десятичных чисел в двоичные, то $m \geq 3n - 3$, а число выходов модуля должно быть не меньше $3n$. На рис. 3.57, б показана схема подключения модулей рассмотренной структуры в преобразователе 4-разрядных десятичных чисел в двоичные. Число каскадов в преобразователе равно $n - 1$.

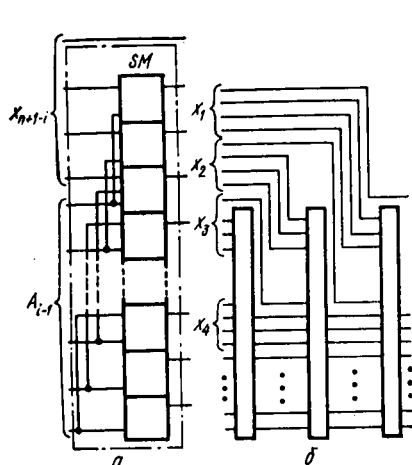


Рис. 3.57

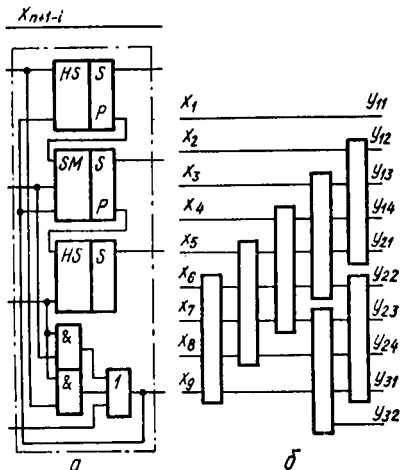


Рис. 3.58

Для определения структуры наращиваемого преобразователя двоичных чисел в десятичные представим двоичное число X в виде

$$X = 2^0 x_1 + 2^1 x_2 + 2^2 x_3 + \dots + 2^{n-2} x_{n-1} + 2^{n-1} x_n = \\ = x_1 + 2(x_2 + 2(x_3 + \dots + 2(x_{n-1} + 2(x_n + 0)) \dots)).$$

Таким образом, преобразование может быть сведено к n -кратному выполнению операции

$$B_i = 2B_{i-1} + x_{n+1-i}, \quad i = 1, 2, \dots, n \quad (3.19)$$

при начальном условии $B_0 = 0$ по правилам десятичной арифметики. Модуль, реализующий вычисления по формуле (3.19), можно построить на основе схемы коррекции и выделения переноса в двоично-десятичном сумматоре (рис. 3.58, а). На старшие входы такой схемы подаются младшие три разряда B_{i-1} , чем и реализуют умножение $2B_{i-1}$. Поскольку это произведение всегда имеет нуль в младшем разряде, то младший разряд B_i определяется значением x_{n+1-i} . Способ соединения однотипных модулей в преобразователе 9-разрядных двоичных чисел в десятичные показан на рис. 3.58, б. Число каскадов в таком преобразователе равно $n - 4$, а число модулей — $(n - 1,5) \left\lceil \frac{n}{3} \right\rceil - 1$.

3.8. ДРУГИЕ ОПЕРАЦИОННЫЕ УЗЛЫ КОМБИНАЦИОННОГО ТИПА

В процессе выполнения программ на ЭВМ встречается операция по определению старшего (младшего) не равного нулю разряда числа. Эти разряды называют приоритетными разрядами, а аппаратные средства для их поиска называют приоритетными схемами.

Приоритетная схема для поиска старшего единичного разряда имеет n входов x_i и n выходов y_i , а ее функционирование может быть описано выражениями

$$y_i = \begin{cases} x_i & \text{при } \bigvee_{j=i+1}^n x_j = 0; \\ 0 & \text{при } \bigvee_{j=i+1}^n x_j = 1, \quad (i = 1, 2, \dots, n). \end{cases}$$

Поэтому можно записать, что

$$y_i = x_i \left(\bigvee_{j=i+1}^n x_j \right) = x_i \& \bar{x}_j = \bar{x}_i \bigvee \bigvee_{j=i+1}^n x_j; \\ (i = 1, 2, \dots, n).$$

Реализация приоритетной схемы по последнему выражению показана на рис. 3.59, а. Для этой схемы

$$L = n - 1; K = \frac{n^2 + n - 2}{2}; t = t_3,$$

а число входов отдельных элементов достигает n . Улучшить оценку K и обеспечить наращиваемость приоритетной схемы можно следующим образом. Обозначим

$$A_i = \bar{x}_i \bar{x}_{i+1} \dots \bar{x}_n; \quad i = 1, 2, \dots, n;$$

$$A_i = \bar{x}_i A_{i+1}; \quad A_{n+1} = 1.$$

Тогда

$$y_i = x_i A_{i+1} = \bar{x}_i \bigvee \bar{A}_{i+1} = \bar{x}_i \bigvee \bar{x}_{i+1} A_{i+2}. \quad (3.20)$$

Приоритетная схема, реализующая систему функций (3.20), представлена на рис. 3.59, б. Для такой схемы

$$L = 3n - 3; K = 15n - 24; t = (2n - 4) t_3.$$

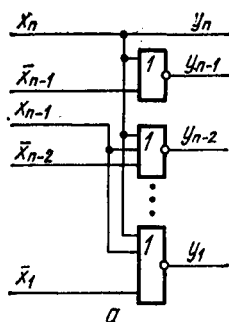
Для построения приоритетной схемы, осуществляющей поиск самой младшей единицы в слове и описываемой функциями

$$z_i = \begin{cases} x_i & \text{при } \bigvee_{j=1}^{i-1} x_j = 0; \\ 0 & \text{при } \bigvee_{j=1}^{i-1} x_j = 1, \end{cases}$$

необходимо во всех приведенных выше выражениях для y_i индекс i изменять в пределах $1 \leq j \leq i - 1$.

Мультиплексором называют операционный узел, служащий для коммутации информационных сигналов X в одном из P возможных

направлений. Селектор — узел, предназначенный для коммутации в одном направлении сигналов, принимаемых с P возможных направлений. Направление коммутации мультиплексора и селектора задается управляющим словом Y . Структуры параллельных мультиплексора и селектора показаны на рис. 3.60 ($n = 3$). Для таких схем



а

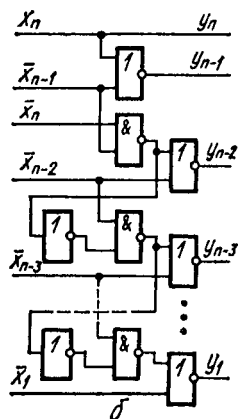


Рис. 3.59

а) $L = nP + L_{DC}$, $K = 2nP + K_{DC}$, $t = t_3 + t_{DC}$;

б) $L = n(P + 1) + L_{DC}$,

$K = 3nP + K_{DC}$, $t = 2t_3 + t_{DC}$.

Заметим, что с целью уменьшения оценки L мультиплексора и селектора можно совместить формирование функций дешифрации и конъюнкций $D_j x_i$ на одних и тех же элементах И (именно так выполнены, например, микросхемы К 155КП1, К 155КП2, К 155КП5, К 155КП7).

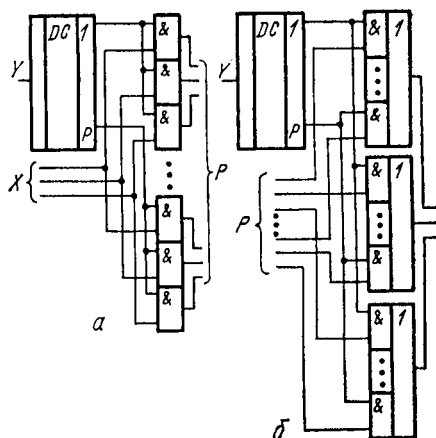


Рис. 3.60

Однако число входов элементов И в этом случае должно быть равно $\lceil \log_2 P \rceil + 1$, а оценки сложности и быстродействия имеют вид

а) $L = nP$, $K = nP (\lceil \log_2 P \rceil + 1)$, $t = t_3$;

б) $L = n(P + 1)$, $K = nP (\lceil \log_2 P \rceil + 2)$, $t = 2t_3$.

В системах обнаружения сбоев в работе цифровых ЭВМ часто используют так называемые схемы контроля по четности (точнее: схемы формирования контрольных разрядов при контроле по четности). Такая схема имеет n входов x_i и один выход S , причем выходной сигнал принимает единичное значение только при четном числе единиц во входном слове, то есть

$$S = \begin{cases} 1 & \text{при } \sum_{i=1}^n x_i \pmod{2} = 0; \\ 0 & \text{при } \sum_{i=1}^n x_i \pmod{2} = 1. \end{cases}$$

Отсюда видно, что функция S может быть реализована как отрицание суммы по модулю 2 разрядных значений x_i . Поскольку функция $\sum_{i=1}^n x_i \pmod{2}$ обладает свойством ассоциативности, то ее, в свою очередь, можно получить путем последовательного или пирамидального соединения схем, реализующих такую же функцию от m переменных ($m < n$). Если, например, $m = 2$, то обозначив

$$S_j = \sum_{i=1}^j x_i \pmod{2}, \quad S = \bar{S}_n,$$

получим

$$S_j = (x_j \vee S_{j-1}) \overline{(x_j S_{j-1})} = \overline{\overline{(x_j \vee S_{j-1})} \vee \overline{(x_j S_{j-1})}}.$$

На рис. 3.61, а показана структура модуля, реализующего функцию S_j , а на рис. 3.61, б, в — соответственно схемы последовательного и пирамидального соединения таких модулей. Эти схемы характеризуются одинаковой сложностью ($L = 4(n-1)$, $K = 7(n-1)$), но

различным временем выполнения операции. Для схемы на рис. 3.61, б $t = 3t_3(n-1)$, а для схемы на рис. 3.61, в $t = 3t_3 \log_2 n$, однако первая схема легко наращивается.

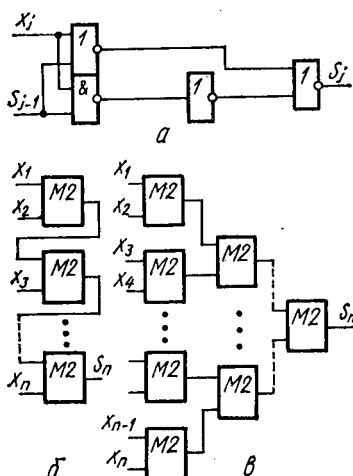


Рис. 3.61

При проектировании аппаратных средств для организации так называемого контроля по модулю (контроля в остатках) необходимо иметь специальные схемы для быстрого определения вычетов (остатков) чисел по модулю k . Эти схемы называют схемами свертки по соответствующему модулю, а сам модуль k обычно выбирают равным $2^m - 1$. Алгоритм функционирования схем свертки состоит в суммировании по модулю k соответствующих групп разрядов. Например, для определения вычета по модулю 3 необходимо разбить все число,

начиная с младших разрядов, на группы по два разряда. После этого, рассматривая группу как одну цифру в системе счисления с основанием 4, просуммировать их по модулю 3. В соответствии с этим, схема свертки должна состоять из преобразователей четвертичных цифр в троичные и сумматоров по модулю 3. Троичные цифры при этом представляют двоично-троичным или пространственным (трехфазным) кодом.

Работа преобразователя четвертичных цифр в двоично-кодированные троичные цифры описывается табл. 53, где x_i , x_{i+1} — группа разрядов исходного числа, f_i , f_{i+1} — функции, реализуемые преобразователем. Из таблицы следует, что

$$\begin{aligned} f_i &= \bar{x}_{i+1} x_i = \overline{x_{i+1} \vee x_i}; \quad \bar{f}_i = \overline{\bar{x}_{i+1} x_i}; \\ f_{i+1} &= x_{i+1} \bar{x}_i = \overline{x_{i+1} \vee x_i}; \quad \bar{f}_{i+1} = \overline{x_{i+1} \bar{x}_i}. \end{aligned}$$

Сумматор по модулю 3 троичных цифр, представленных двоичными кодами f_4f_3 и f_2f_1 , реализует функции h_2 и h_1 , заданные табл. 54. Операторные формы И—ИЛИ—НЕ этих функций имеют вид

$$h_1 = \overline{f_3 f_1 \vee f_4 \bar{f}_2 \vee \bar{f}_4 f_2 \vee \bar{f}_4 \bar{f}_3 \bar{f}_1};$$

$$h_2 = \overline{f_4 f_2 \vee f_3 \bar{f}_1 \vee \bar{f}_3 f_1 \vee \bar{f}_4 \bar{f}_3 \bar{f}_2}.$$

На рис. 3.62, а показан фрагмент пирамидальной схемы свертки по модулю 3, построенной по приведенным выражениям. Первый каскад этой схемы образуют преобразователи четверичных цифр в троич-

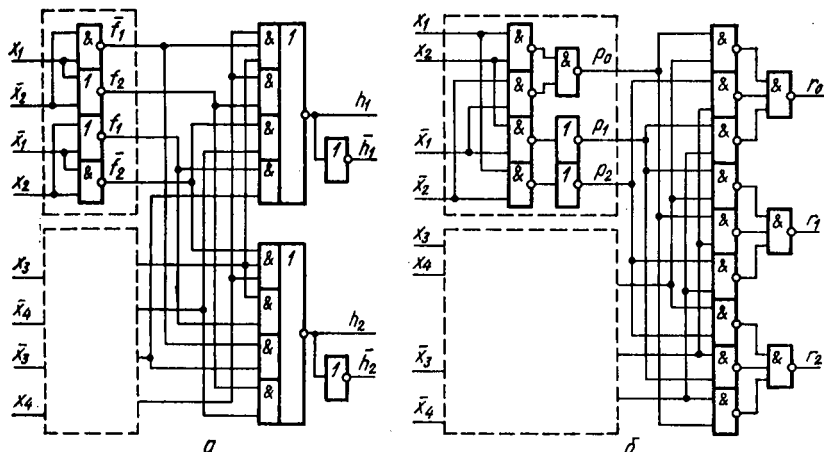


Рис. 3.62

ные, число которых равно $\frac{n}{2}$. Второй и все последующие каскады построены из сумматоров по модулю 3, общее число которых составляет $\frac{n}{2} - 1$. Таким образом, для всей схемы свертки

$$L = 4 \cdot \frac{n}{2} + 12 \left(\frac{n}{2} - 1 \right) = 8n - 12;$$

$$K = 2 \cdot 4 \cdot \frac{n}{2} + 28 \left(\frac{n}{2} - 1 \right) = 18n - 28;$$

$$t = (3 \log_2 n - 2) t_3.$$

Таблица 53

x_{i+1}	x_i	f_{i+1}	f_i
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

Таблица 54

f_4	f_3	f_2	f_1	h_2	h_1
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1

Таблица 55

x_{i+1}	x_i	p_0	p_1	p_2
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

Последовательное соединение сумматоров по модулю 3 не улучшает оценок L и K , а время выполнения операции при этом увеличивается $\left(t = \left(\frac{3}{2}n - 2\right)t_3\right)$.

При использовании пространственного представления троичных цифр преобразователь четверичных цифр в троичные должен реализовать функции p_0, p_1, p_2 (табл. 55). Функционирование сумматора по модулю 3 пространственных кодов $p_{21}p_{11}p_{01}$ и $p_{22}p_{12}p_{02}$ описывается табл. 56, где r_2, r_1 и r_0 — выходы сумматора. Перечисленным функциям соответствуют такие МДНФ и операторные формы:

$$p_0 = x_{i+1} x_i \vee \overline{x_{i+1}} \overline{x_i} = \overline{(x_{i+1} x_i)} \overline{(\overline{x_{i+1}} \overline{x_i})};$$

$$p_1 = \overline{x_{i+1}} x_i = x_{i+1} \vee \overline{x_i}, \quad p_2 = x_{i+1} \overline{x_i} = \overline{x_{i+1}} \vee x_i;$$

$$r_0 = p_{02} p_{01} \vee p_{12} p_{21} \vee p_{22} p_{11} = \overline{(p_{02} p_{01})} \overline{(p_{12} p_{21})} \overline{(p_{22} p_{11})};$$

$$r_1 = p_{02} p_{11} \vee p_{12} p_{01} \vee p_{22} p_{21} = \overline{(p_{02} p_{11})} \overline{(p_{12} p_{01})} \overline{(p_{22} p_{21})};$$

$$r_2 = p_{02} p_{21} \vee p_{12} p_{11} \vee p_{22} p_{01} = \overline{(p_{02} p_{21})} \overline{(p_{12} p_{11})} \overline{(p_{22} p_{01})}.$$

Схема свертки на рис. 3.62, б имеет то же функциональное назначение, что и предыдущая схема, однако она построена с использованием пространственного представления троичных цифр и характеризуется следующими оценками:

$$L = 7 \frac{n}{2} + 12 \left(\frac{n}{2} - 1 \right) = \frac{19n}{2} - 12;$$

Таблица 56

p_{22}	p_{12}	p_{02}	p_{21}	p_{11}	p_{01}	r_2	r_1	r_0
0	0	1	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	1	0
0	1	0	0	1	0	1	0	0
0	1	0	1	0	0	0	0	1
1	0	0	0	0	1	1	0	0
1	0	0	0	1	0	0	0	1
1	0	0	1	0	0	0	1	0

$$K = 12 \frac{n}{2} + 27 \left(\frac{n}{2} - 1 \right) = \frac{39n}{2} - 27;$$

$$t = 2(\log_2 n) t_3.$$

3.9. РЕГИСТРЫ

Регистром называют операционный узел, служащий для запоминания слов и обеспечивающий в общем случае выполнение следующих микроопераций:

- 1) установка регистра в нуль (сброс, гашение);
- 2) прием слова из другого регистра, сумматора, счетчика и т. п.;

- 3) передача слова на другой регистр, сумматор, счетчик и т. п.;
- 4) преобразование кодов хранимых слов в инверсные коды;
- 5) сдвиг слова влево или вправо на требуемое число разрядов;
- 6) преобразование последовательного кода в параллельный и наоборот;
- 7) поразрядные операции конъюнкции, дизъюнкции и сложения по mod 2.

Схемы конкретных регистров в частных случаях могут реализовать лишь некоторые из перечисленных микроопераций.

Обычно регистры строят на триггерах *RS*, *JK*, *DV*, *D* или *T*-типов, которые и реализуют микрооперацию запоминания слов. Остальные же микрооперации выполняются за счет подключения ко входам и выходам триггеров логических элементов, а также за счет организации соответствующих связей между ними.

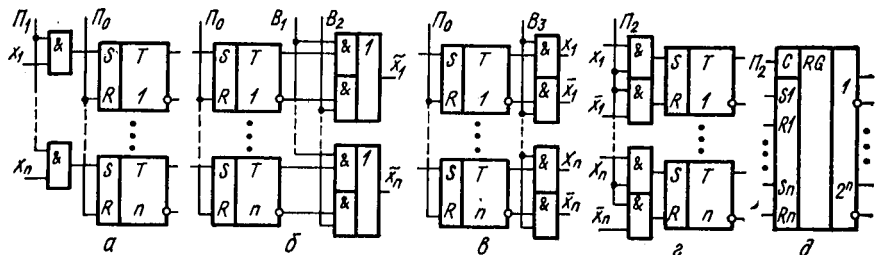


Рис. 3.63

Будем называть параллельным регистром такой регистр, который реализует все перечисленные операции с n -разрядными словами, за исключением сдвига и преобразования последовательного кода в параллельный и наоборот. Если в параллельном регистре на вход каждого разряда информация поступает по двум каналам в парафазном коде (парафазный код соответствует пространственному принципу представления букв двузначного структурного алфавита), то такой регистр называют парафазным. При наличии только одного канала (прямого или инверсного) поступления информации в каждом разряде регистр называют однофазным. Парафазные регистры строят, как правило, на триггерах с раздельными входами, а однофазные — на триггерах с одним входом. Различают также однотактные и многотактные регистры. В однотактных регистрах запись информации производится парафазным кодом без предварительной установки триггеров в «0», причем сами триггеры выполняют по схеме с внутренней задержкой. В многотактных регистрах, реализуемых на основе тактируемых триггеров, микрооперации выполняются при подаче двух или более тактовых сигналов.

Схема параллельного однофазного регистра, выполняющего первые две микрооперации из приведенного выше списка, показана на рис. 3.63, а. На нулевые входы всех триггеров подается сигнал установки нулевого состояния (сигнал сброса) Π_0 . После подачи этого сигнала все триггеры регистра будут находиться в нулевом состоянии до момента появления на входных шинах записываемого слова и сиг-

нала приема слова Π_1 . В тех разрядах, где $x_i = 1$, произойдет установка триггеров в единичное состояние. Там же, где $x_i = 0$, состояния триггеров не изменяются. Выдача информации с регистра может производиться в прямом, инверсном и парафазном кодах. Схема выдачи информации в прямом и инверсном кодах показана на рис. 3.63, б. Здесь B_1 — сигнал выдачи прямого кода, B_2 — сигнал выдачи инверсного кода. Очевидно, что одновременное появление сигналов B_1 и B_2 должно быть запрещено, т. е. должно выполняться условие $B_1 B_2 = 0$. Схема регистра с выдачей парафазного кода приведена на рис. 3.63, в. Схема приема информации на регистр без предварительной установки его в нуль представлена на рис. 3.63, г. Условное обозначение такого регистра на функциональных схемах показано на рис. 3.63, д. Здесь C — вход синхронизирующего сигнала. Использование парафазного кода

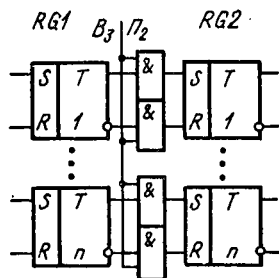


Рис. 3.64

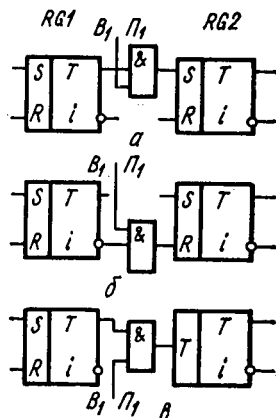


Рис. 3.65

обеспечивает установку триггеров в требуемое состояние независимо от той информации, которая была на них записана ранее.

Операция выдачи слова с одного регистра обычно совмещается с приемом слова на другой регистр (рис. 3.64).

В процессе передачи слов с одного регистра на другой можно выполнять поразрядную дизъюнкцию слов (рис. 3.65, а). В регистре $RG1$ хранится слово X , а в регистре $RG2$ — слово Y . Указанная операция выполняется путем передачи прямым кодом слова X без предварительного сброса слова Y в регистре $RG2$. В самом деле, если $y_i = 1$, то пришедший единичный или нулевой сигнал с i -го разряда регистра $RG1$ не меняет этого значения. Если же $y_i = 0$, то i -й триггер в $RG2$ установится в «1» при $x_i = 1$ или же останется в «0» при $x_i = 0$.

Поразрядная конъюнкция может выполняться в соответствии со схемой на рис. 3.65, б. Как и в предыдущем случае, операция выполняется путем передачи слова X с регистра $RG1$ на регистр $RG2$ без предварительного сброса последнего. Однако здесь используется инверсный код X , который подается на нулевые входы триггеров $RG2$. Если $y_i = 1$, то это состояние останется неизменным лишь в случае, когда $\bar{x}_i = 0$. Если же $y_i = 0$, то независимо от значения сигнала x_i i -й триггер в $RG2$ останется в том же состоянии.

Поразрядная операция сложения по mod 2 может быть реализо-

вана, если в $RG2$ использовать триггеры со счетным входом. Такая схема изображена на рис. 3.65, в. Если $x_i = 1$, то i -й триггер регистра $RG2$ всегда будет менять свое состояние на противоположное, что соответствует сложению по $\text{mod } 2$ y_i и 1. Если $x_i = 0$, то состояние i -го триггера не меняется, что соответствует сложению y_i с 0.

Сдвигающим регистром называют такой регистр, который выполняет микрооперацию сдвига, а также микрооперации приема и выдачи слов. Если две последних микрооперации реализуются только в крайних разрядах регистра (1-м или n -м), то сдвигающий регистр называют также последовательным регистром. При сдвиге слова в ре-

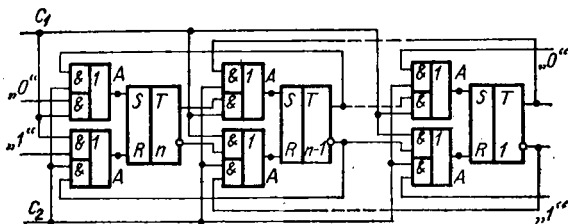


Рис. 3.66

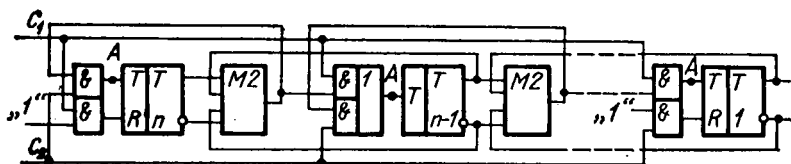


Рис. 3.67

гистре разряды слова, вышедшие за пределы разрядной сетки регистра, теряются, а освободившиеся разряды заполняются нулями.

Регистр, который может осуществлять сдвиг слов как влево, так и вправо, принято называть реверсивным. Схема реверсивного сдвигающего регистра с парафазной передачей слов (рис. 3.66) не требует сброса регистра в «0». На рис. 3.66 C_1 — сигналы сдвига содержимого регистра вправо на один разряд; C_2 — сигналы сдвига на один разряд влево. При необходимости выполнения сдвига сразу на j -разрядов выходы i -го триггера подключаются не к $(i - 1)$ -му или $(i + 1)$ -му разряду, а к $(i - j)$ -му или $(i + j)$ -му разряду.

Так как единичный сигнал всегда имеется на одной из разрядных шин парафазного кода, то установка триггеров производится в соответствии с этим сигналом. Для выполнения условий правильного обмена информацией в такой схеме в точках A должны быть включены элементы задержки. В противном случае считывающий и записывающий сигналы не будут разнесены во времени, что приведет к неопределенному переключению триггеров регистра.

При использовании в сдвигающем регистре T -триггеров их функции возбуждения реализуются с помощью схем неравнозначности цифр в соседних разрядах (рис. 3.67). Каждая из функций возбуждения в реверсивном сдвигающем регистре на T -триггерах имеет вид

$$f_i = (x_i \bar{x}_{i-1} \vee \bar{x}_i x_{i-1}) C_2 \vee (x_i x_{i+1} \vee \bar{x}_i \bar{x}_{i+1}) C_1.$$

Выходной сигнал схемы неравнозначности равен единице только тогда, когда в соседних разрядах записаны разные цифры. В этом случае на счетный вход триггера T_{i-1} или T_{i+1} (в зависимости от направления сдвига) поступает единичный сигнал и устанавливает его в состояние, соответствующее состоянию триггера T_i . Как и в предыдущей схеме, для выполнения условий правильного обмена информацией в точках А необходимо включить элементы задержки.

Сдвигающий регистр реализует и преобразование последовательного кода в параллельный и наоборот. При преобразовании последовательного в параллельный запись слова в регистр производится синхронно со сдвигом его содержимого влево, если последовательный код

поступает со старших разрядов, или вправо, если код поступает с младших разрядов. После заполнения всех разрядов регистра следует параллельная выдача слова через элементы И, подключенные к выходам триггеров регистра.

При преобразовании параллельного кода в последовательный информация заносится на регистр параллельным кодом, а затем следует серия из n сдвигающих сигналов. Последовательный код считывается с триггера T_1 (с младших разрядов) при сдвиге вправо или же с триггера T_n (со старших разрядов) при сдвиге влево.

Сдвигающие регистры на потенциальных элементах очень часто выполняют по двутактной схеме. Условия правильного обмена информацией в этом

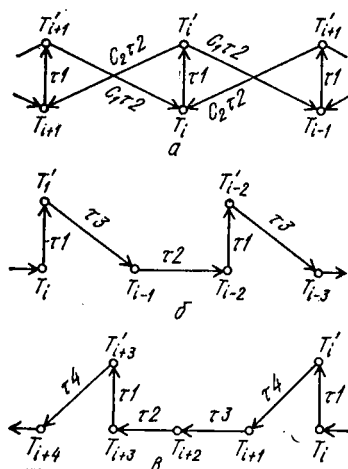


Рис. 3.68

случае выполняются за счет введения дополнительных (вспомогательных) триггеров. Каждый i -й разряд регистра содержит два синхронных одноктактных триггера — основной T_i и вспомогательный T_i' . Сдвиг информации в таких регистрах осуществляется в соответствии со схемой на рис. 3.68, а. Здесь τ_1 и τ_2 — тактирующие сигналы, не пересекающиеся во времени, а обозначения $C_2\tau_2$ и $C_1\tau_2$ на схеме означают, что передача информации по соответствующему каналу происходит при единичном значении конъюнкции сигналов C_2 и τ_2 или C_1 и τ_2 . Таким образом, для сдвига слова на один разряд влево или вправо необходимо подать два тактирующих сигнала τ_1 и τ_2 . Число одноктактных триггеров в таком регистре равно $2n$.

Сложность сдвигающего регистра можно уменьшить, реализовав трехтактный сдвиг (рис. 3.68, б) с помощью трех серий не пересекающихся во времени сигналов τ_1 , τ_2 и τ_3 . В этом случае каждым двум основным триггерам соответствует один вспомогательный, а общее число триггеров равно $1,5n$. Однако для сдвига слова на один разряд необходимо подать 3 тактирующих сигнала τ_1 , τ_2 , τ_3 . Число одноктактных триггеров можно еще уменьшить в четырехтактном регистре, где на каждые три основных триггера приходится один вспомога-

ный (рис. 3.68, а), а общее их число составляет $1,33n$. Сдвиг на один разряд в таком регистре длится 4 такта. Очевидно, что при наличии n серий тактирующих сигналов можно реализовать сдвиг в регистре, не содержащем вспомогательных триггеров. Длительность такого сдвига составляет n тактов. В общем случае при p -тактном сдвиге число однократных триггеров равно $n + \frac{n}{p-1}$, а время сдвига сло-

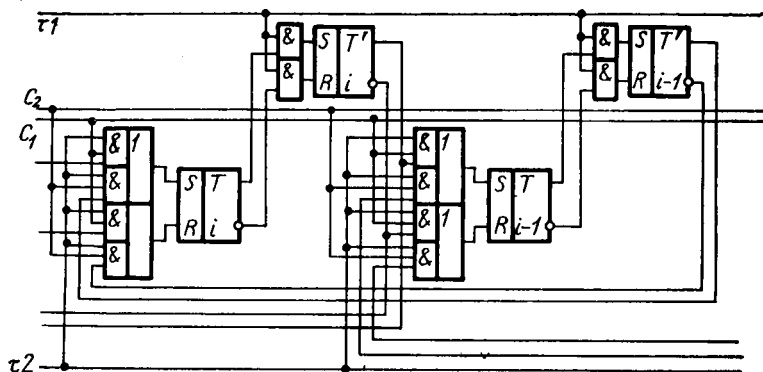


Рис. 3.69

ва на один разряд пропорционально p . Поэтому при проектировании регистров чаще всего выбирают $p \leq 4$, так как дальнейшее увеличение p не приводит к заметному упрощению регистра и, вместе с тем, существенно снижает его быстродействие.

Фрагмент схемы реверсивного двухтактного сдвигающего регистра на асинхронных RS -триггерах представлен на рис. 3.69. В такой схеме управляющие сигналы C_1 и C_2 синхронизируются сигналами τ_2 . Функции возбуждения i -го триггера в рассматриваемом реверсивном регистре можно записать в таком виде:

$$S_i = T'_{i-1} C_2 \tau_2 \vee T'_{i+1} C_1 \tau_2;$$

$$R_i = \bar{T}'_{i-1} C_2 \tau_2 \vee \bar{T}'_{i+1} C_1 \tau_2;$$

$$S'_i = T_i \tau_1; R'_i = \bar{T}_i \tau_1.$$

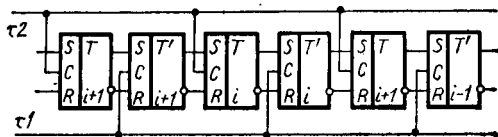


Рис. 3.70

Очевидно, что правильная работа регистра возможна лишь при выполнении условия

$$C_1 C_2 = 0.$$

Схемы как сдвигающих, так и параллельных регистров значительно упрощаются при использовании синхронных триггеров, в которых элементы, не входящие в бистабильную схему, имеют больше, чем два входа. В этом случае все функции возбуждения реализуются на самих триггерах. Схема сдвигающего регистра на синхронных RS -триггерах приведена на рис. 3.70. Здесь сигналы C_1 должны быть совмещены с тактирующим сигналом τ_2 . При использовании тригге-

ров с внутренней задержкой сдвигающий регистр можно строить по схеме на рис. 3.71, а.

Схема сдвигающего регистра на JK -триггерах приведена на рис. 3.71, б. Нетрудно заметить, что в этой схеме JK -триггер включен как D -триггер, так как входы J_i и K_i триггера подключены к вза-

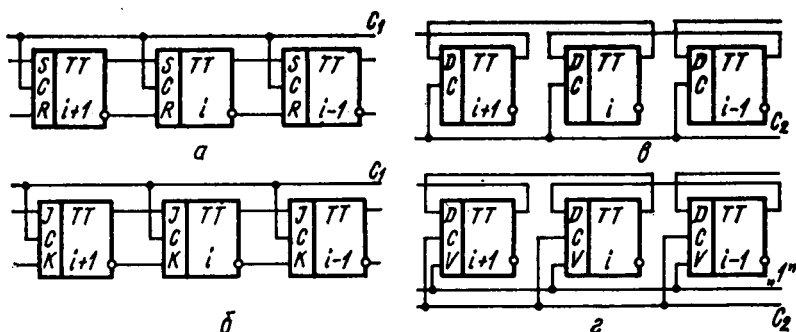


Рис. 3.71

имно инверсным выходам T_{i+1} и $\bar{T}_{i+1}$. Функции возбуждения JK -триггера в реверсивном сдвигающем регистре можно записать как

$$J_i = T_{i+1} C_1 \vee T_{i-1} C_2 = \overline{(T_{i+1} C_1)} \overline{(T_{i-1} C_2)};$$

$$K_i = \bar{T}_{i+1} C_1 \vee \bar{T}_{i-1} C_2 = \overline{(\bar{T}_{i+1} C_1)} \overline{(\bar{T}_{i-1} C_2)}.$$

Отсюда видно, что функция J_i принимает те значения, которые совпадают с состояниями T_{i+1} или T_{i-1} на предыдущем такте сдвига. Следовательно, используя D -триггеры, однофазный сдвигающий регистр можно построить так, как показано на рис. 3.71, в. По подобным схемам строятся однофазные многотактные сдвигающие регистры на основе рассмотренных в п. 2.9 D -триггеров на МОП-транзисторах. Сдвигающие регистры могут быть построены и на DV -триггерах (рис. 3.71, г), где эти триггеры включены как D -триггеры. При построении реверсивных регистров на DV -триггерах функция возбуждения D_i равна функции J_i регистра на JK -триггерах.

Условное обозначение сдвигающего регистра на схемах ЦВТ показано на рис. 3.72, где стрелкой указывают направление сдвига.

Схемы регистров на рис. 3.69—3.71 построены при предположении, что регистры выполняют только микрооперацию сдвига. Если регистр должен выполнять еще и другие микрооперации, то соответствующим образом должны быть изменены функции возбуждения триггеров в каждом разряде. Например, если реверсивный регистр на JK -триггерах помимо сдвига должен принимать слова в парафаз-

ном коде и сбрасываться в «0», то функции возбуждения его i -го триггера можно записать так:

$$J_i = x_i P_2 \vee T_{i+1} C_1 \vee T_{i-1} C_2 = \overline{(x_i P_2)} \overline{(T_{i+1} C_1)} \overline{(T_{i-1} C_2)};$$

$$K_i = \bar{x}_i P_2 \vee \bar{T}_{i+1} C_1 \vee \bar{T}_{i-1} C_2 \vee P_0 = \overline{(\bar{x}_i P_2)} \overline{(\bar{T}_{i+1} C_1)} \overline{(\bar{T}_{i-1} C_2)} \bar{P}_0.$$

Естественно, что такой регистр может правильно функционировать лишь при выполнении условия

$$P_0 P_2 \vee P_0 C_1 \vee P_0 C_2 \vee P_2 C_1 \vee P_2 C_2 \vee C_1 C_2 = 0.$$

В заключение заметим, что вследствие разрядного принципа построения схем сдвигающих и параллельных регистров их оценки сложности L и K пропорциональны n , а требование наращиваемости структуры легко выполнимо.

3.10. СЧЕТЧИКИ

Счетчиком называют автомат, выполняющий функции подсчета количества единичных сигналов, поступивших на его вход, а также функции формирования и запоминания слова, соответствующего этому количеству. Если это слово формируется в двужанчном алфавите, то счетчик можно определить как преобразователь числоимпульсного кода в некоторый двоичный код. Помимо указанных функций счетчики часто также могут выполнять микрооперации приема и выдачи слов.

Способ кодирования формируемых на счетчиках слов определяет их разделение на счетчики с позиционным (взвешенным) и непозиционным кодированием. Позиционное кодирование означает, что количество P единичных сигналов, поступивших на вход счетчика, определяется как

$$P = \sum_{i=1}^n p_i k_i,$$

где n — число разрядов; $p_i \in \{0, 1, \dots, k-1\}$ — состояние i -го разряда счетчика; k_i — вес i -го разряда. В счетчике с непозиционным кодированием разряды не имеют постоянных весов, а соответствие между P и состоянием счетчика выражается более сложным образом (обычно задается с помощью таблиц).

В аппаратных средствах ЦВТ получили преимущественное распространение счетчики с позиционным кодированием. Такие счетчики имеют разрядную структуру. Если $k_i = k^{i-1}$, то число разрядов счетчика определяется по формуле

$$n = \lceil \log_k N \rceil,$$

где N — важнейшая характеристика счетчика, называемая коэффициентом пересчета, модулем или периодом счетчика. Коэффициент пересчета равен максимальному числу внутренних состояний счетчика, которое, в свою очередь, равно максимальному числу входных сигналов, которое может сосчитать счетчик. При $k = 2$ разряды счет-

чика с позиционным кодированием выполняются на основе триггеров T , JK , RST или, реже, RS и D -типов.

К другим важнейшим характеристикам счетчиков относятся время регистрации — интервал времени между моментом поступления входного сигнала и окончанием самого длинного переходного процесса в схеме, а также разрешающая способность счетчика — минимально допустимый период следования входных сигналов, при котором счетчик работает без сбоев. Очевидно, что время регистрации определяется типом используемых триггеров и организацией межразрядных связей, а разрешающая способность зависит от динамических свойств триггера в разряде с самым малым весом.

По назначению счетчики разделяют на простые (суммирующие и вычитающие) и реверсивные. Простые счетчики могут вести счет только в одном направлении, т. е. или только прибавлять входные сигналы к слову, сформированному на счетчике, или же только вычитать входные сигналы из этого слова. Реверсивные счетчики, в зависимости от управляющих сигналов, могут вести счет и в прямом, и в обратном направлениях. Некоторые счетчики имеют цепи внешней установки начального состояния. Это состояние может также устанавливаться с помощью цепей обратной связи, имеющих в самом счетчике. Кроме того, счетчик может устанавливаться в начальное состояние после завершения одного цикла работы — подсчета числа входных сигналов, равного модулю счетчика.

Простейший счетчик выполняется на триггерах со счетным входом, которые реализуют одноразрядное сложение по $\text{mod } 2$ единичных входных сигналов. Таким образом, T -триггер является счетчиком с $N = 2$. Если же $N = 2^n$, где $n > 1$, то прибегают к каскадному соединению таких триггеров. При этом различают асинхронные и синхронные счетчики. В асинхронных счетчиках сигналы от одного каскада к другому передаются естественным образом в различные моменты времени, зависящие от сочетания входных сигналов и состояния счетчика. В синхронных счетчиках все межкаскадные передачи и переключения триггеров происходят одновременно под воздействием синхронизирующих сигналов.

По способу формирования сигналов переноса (заема) между разрядами различают счетчики с последовательным, сквозным, параллельным (одновременным) и групповым (частично-групповым) переносом. Счетчик с последовательным переносом является асинхронным счетчиком. В таком счетчике входные сигналы поступают на вход только первого триггера, а каждый последующий триггер переключается выходными сигналами предыдущего триггера. При использовании в счетчике T -триггеров реализуют однопроводную передачу переноса (рис. 3.73, а), а на RS -триггерах строят счетчики с двухпроводной передачей переноса (рис. 3.73, б). На рис. 3.73, в приведена временная диаграмма работы счетчика с последовательным переносом и однопроводной передачей. Из диаграммы легко заметить, что счетчик по модулю N является делителем частоты входных сигналов в N раз, так как на выходе i -го триггера появляется лишь 2^i -й сигнал по отношению к числу сигналов, поступивших на вход счетчика. Кроме того, из диаграммы видно, что в процессе работы счетчика воз-

можны ситуации, когда входной сигнал X вызывает последовательное переключение всех триггеров. Поэтому время регистрации в счетчике с последовательным переносом равно времени переключения n триггеров. Вычитающий счетчик с последовательной передачей сигналов заема получим, если в схеме на рис. 3.73, *а* использовать нулевые выходы триггеров, а в схеме на рис. 3.73, *б* — тактировать основной триггер в i -м разряде ($i = 2, 3, \dots$) сигналом $\bar{T}_{i-1}$, а вспомогательный

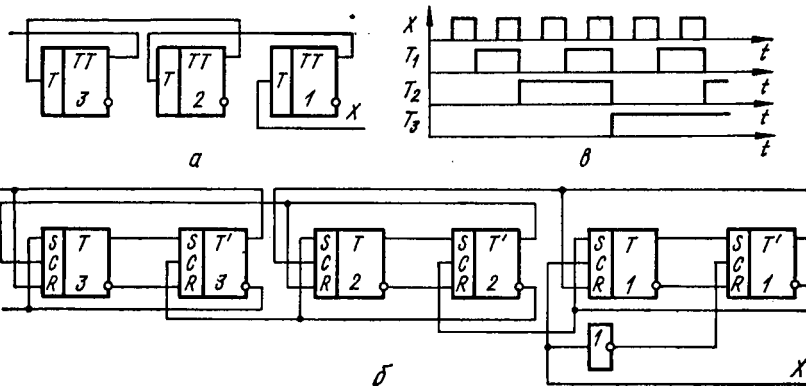


Рис. 3.73

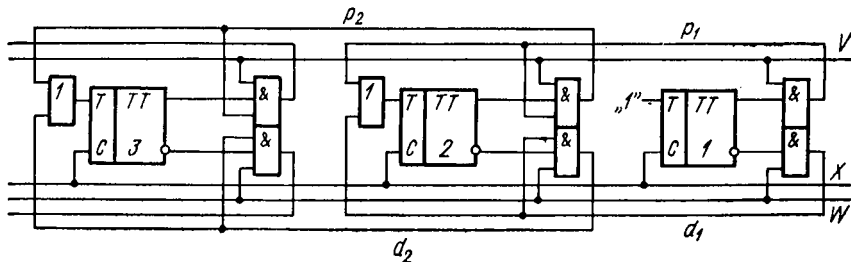


Рис. 3.74

триггер в том же разряде — сигналом T_{i-1} . Счетчики с другой организацией цепей переноса строятся как синхронные счетчики.

Принцип работы реверсивного счетчика со сквозным переносом иллюстрируется схемой на рис. 3.74. Здесь сигналы V и W — сигналы сложения и вычитания — определяют направление счета. Сигналы переноса p_i или заема d_i с i -го разряда формируются в соответствии с выражениями

$$p_i = T_i p_{i-1} V; \quad d_i = \bar{T}_i d_{i-1} W,$$

а функция возбуждения T -триггера в i -м разряде реверсивного счетчика имеет вид

$$\bar{f}_i = p_{i-1} \vee d_{i-1}.$$

В случае $i = 1, f_1 = 1$. Операторные формы функций $p_i, d_i, \bar{f}_i$ можно получить из приведенных выражений применением правила де Мор-

гана. Работает такой счетчик следующим образом. Если $T_i = 1$, то при поступлении сигнала переноса p_{i-1} (или сигнала V в случае $i = 1$) соответствующий элемент И будет открыт и тем самым будет сформирован сигнал переноса p_i . Триггер T_i вслед за этим установится в «0», причем сигнал p_i будет сформирован раньше, чем сумма цифры из данного разряда и переноса p_{i-1} . Быстродействие такого счетчика выше, чем счетчика с последовательным переносом, так как время регистрации здесь определяется временем переключения $n - 1$ элементов И (n элементов И в случае реверсивного счетчика) и одного триггера.

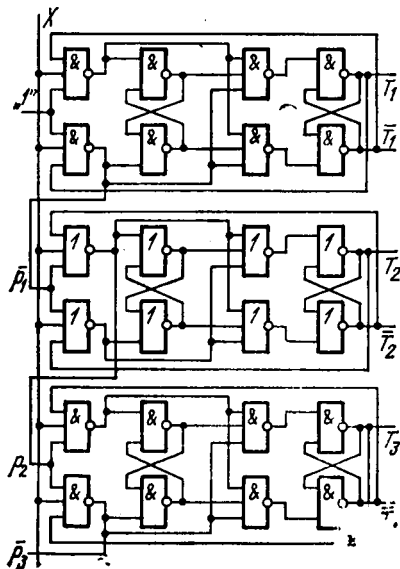


Рис. 3.75

переноса и сигналов возбуждения совмещено на одних и тех же элементах. Здесь триггеры в нечетных разрядах выполнены на элементах И—НЕ, а в четных разрядах — на элементах ИЛИ—НЕ. Счетчики, построенные по подобным схемам, называют также счетчиками с внутренним формированием переносов.

В счетчике с параллельным (одновременным) переносом сигналы p_i и d_i формируются как независимые функции содержимого всех младших разрядов счетчика:

$$p_i = \bigwedge_{j=1}^i T_j V = \overline{\bigvee_{j=1}^i \bar{T}_j \bigvee \bar{V}}, \quad d_i = \bigwedge_{j=1}^i \bar{T}_j W = \overline{\bigvee_{j=1}^i T_j \bigvee \bar{W}}.$$

Таким образом, сущность идеи параллельного переноса в счетчиках та же, что и в комбинационных сумматорах. Схема счетчика с параллельным переносом представлена на рис. 3.76. С возрастанием n в такой схеме увеличивается число входов элементов в цепи параллельного переноса. Из-за конечности коэффициента объединения этих элементов число разрядов в счетчике с параллельным переносом не

В простом счетчике, где используются триггеры, построенные по схеме $M - S$, сквозной перенос можно реализовать без дополнительных аппаратных затрат. Пусть, например, для суммирующего счетчика

$$p_i = T_i p_{i-1} = \bar{T}_i \bigvee \bar{p}_{i-1}, \quad \bar{p}_i = \bar{T}_i p_{i-1}.$$

Отсюда видно, что одним из аргументов функции p_i является $\bar{p}_{i-1}$, а функции $\bar{p}_i$ — p_{i-1} . Поэтому достаточно в одних разрядах (нечетных) формировать только сигналы $\bar{p}_i$, а в других разрядах (четных) — сигналы p_i . Вместе с тем выражения вида $\bar{T}_i \bigvee \bar{p}_{i-1}$ и $T_i p_{i-1}$ представляют собой функции возбуждения бистабильных схем в T -триггере (см. п. 2.9). На рис. 3.75 приведена схема суммирующего счетчика со сквозным переносом, где формирование сигналов

может быть большим. Поэтому в тех случаях, когда число входов элементов меньше числа разрядов счетчика, последний разбивают на группы и внутри каждой из них строят цепи параллельного переноса. Перенос между группами реализуется методом сквозного переноса. На рис. 3.77 приведена схема счетчика с групповым переносом и 4-разрядными группами. Время регистрации в счетчике с параллельным переносом равно времени переключения одного элемента И и одного триггера, а в счетчике с групповым переносом равно $(m - 1)t_3 + mt_{\text{рп}}$, где m — число групп; $t_{\text{рп}}$ — время регистрации для счетчика с параллельным переносом.

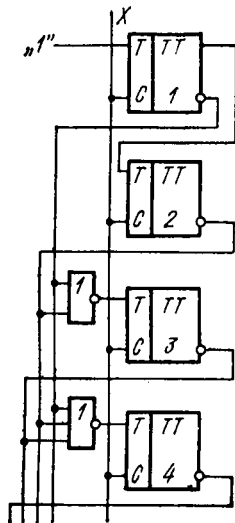


Рис. 3.76

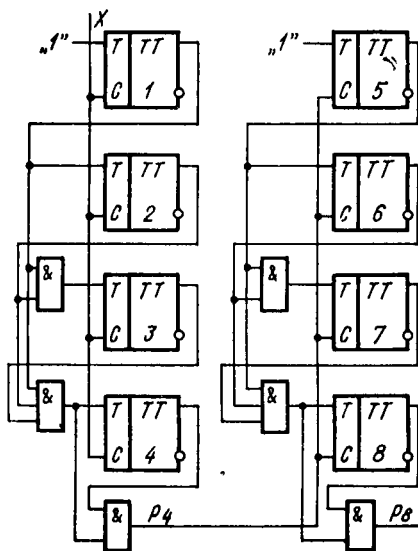


Рис. 3.77

Часто для построения схем ЭВМ необходимы счетчики, модуль которых не равен целой степени числа 2. Пусть необходимо построить счетчик по модулю M , причем

$$2^n > M > 2^{n-1}.$$

Очевидно, что счетчик по модулю M должен содержать не менее n разрядов. Однако при этом $m = 2^n M$ состояний счетчика будут «лишними» (запрещенными). Поэтому необходимо так построить счетчик, чтобы в процессе его работы не могло возникнуть ни одно из запрещенных состояний. Известно большое число способов исключения запрещенных состояний, ориентированных на использование в счетчиках различных типов триггеров и различную организацию цепей переноса. Рассмотрим некоторые из них.

В синхронном счетчике со сквозным переносом исключить состояния, соответствующие числам $0, 1, \dots, m - 1$ можно путем введения запрещающей обратной связи с выхода цепи переноса последнего (старшего) разряда ко входам тех триггеров, которые находятся в «1»,

если в счетчик записано двоичное число m . Например, при $M=10$, $n=4$ необходимо исключить состояния 0000, 0001, 0010, 0011, 0100, 0101 счетчика. Перенос со старшего разряда равен 1, когда счетчик установится в состояние 1111, а первым разрешенным состоянием будет 0110. Без введения обратной связи счетчик после состояния 1111 естественным образом перешел бы в состояние 0000. Поэтому сигнал по цепи обратной связи должен обеспечить сохранение состояний триггеров во втором и третьем разрядах при действии очередного входного сигнала. Схема счетчика для этого случая приведена на рис. 3.78, а, а временная диаграмма его работы на рис. 3.78, б.

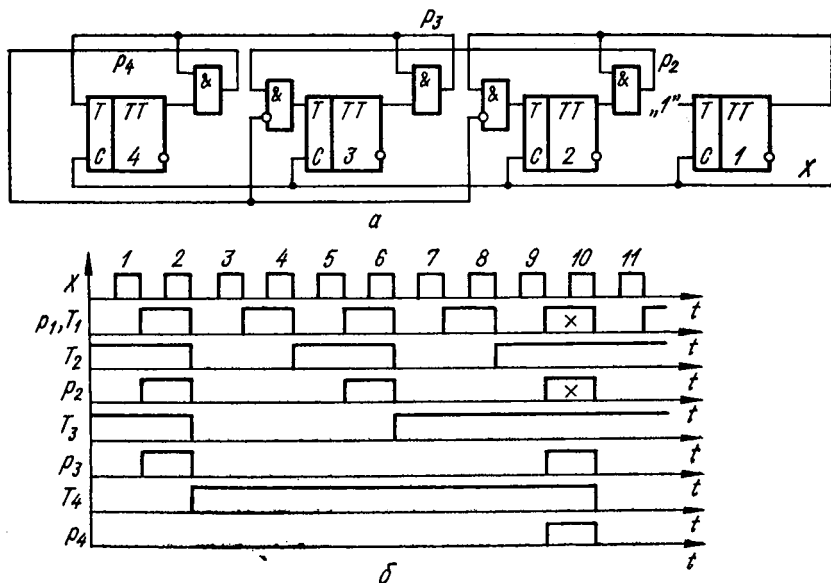


Рис. 3.78

Сигналы ρ_i , прохождение которых блокируется цепью обратной связи, на диаграмме отмечены знаком X. Однако сформированное слово в таком счетчике не соответствует количеству поступивших на его вход сигналов, так как счет начинается не с нулевого состояния, а с состояния m (количество сосчитанных сигналов определяется кодом с избытком m). Для устранения этого недостатка исключают последние m состояний триггерной схемы с 2^n состояниями. При использовании синхронных Т-триггеров с асинхронными входами R и S с этой целью к единичным выходам тех триггеров, которые находятся в «1», если в счетчик записать двоичное число $M-1$, подключают элемент И (или каскадно соединенные элементы И при большом числе разрядов). Единичный сигнал на выходе этого элемента должен заблокировать цепи передачи сигналов переноса на Т входы и открыть цепи передачи синхронизирующих сигналов на входы R триггеров во всех разрядах. Схема такого счетчика по модулю 10 приведена на рис. 3.79.

Если состояниям схемы на n триггерах поставить в соответствие множество точек на числовой оси, занумерованных числами $0, 1, \dots, 2^n - 1$, то первый из рассмотренных способов позволяет исключить m запрещенных состояний, расположенных подряд в начале числовой оси, а второй способ исключает m состояний подряд в конце числовой оси. Исключить в счетчике любые m состояний подряд можно следующим образом. Обозначим $\vec{\alpha} = (\alpha_n, \alpha_{n-1}, \dots, \alpha_1)$ — состояние счетчика перед первым исключаемым состоянием $\vec{\beta} = (\beta_n, \beta_{n-1}, \dots, \beta_1)$, а $\vec{\gamma} = (\gamma_n, \gamma_{n-1}, \dots, \gamma_1)$ — первое состояние после m подряд исключенных состояний, $\alpha_i, \beta_i, \gamma_i \in \{0, 1\}$. Сравним поразрядно значения α_i, β_i и γ_i . Если $\beta_i = \gamma_i$, то это означает, что сигнал возбуждения T -триггера в i -м разряде при переходе из состояния $\vec{\alpha}$ в состояние $\vec{\gamma}$ равен сигналу возбуждения при переходе из $\vec{\alpha}$ в $\vec{\beta}$ и, следовательно, в этом случае никакие схемные изменения не требуются. Если $\beta_i \neq \gamma_i$, но $\alpha_i = \gamma_i$, то это означает, что при переходе из α в γ сигнал возбуждения i -го триггера должен быть равен нулю. Поэтому перенос в i -й разряд надо умножить на $\bar{D}(\vec{\alpha})$, где $D(\vec{\alpha})$ — функция дешифрации состояния $\vec{\alpha}$. Если же $\beta_i \neq \gamma_i$ и $\alpha_i \neq \gamma_i$, то это свидетельствует о равенстве 1 сигнала возбуждения i -го триггера при переходе из состояния $\vec{\alpha}$ в состояние $\vec{\gamma}$. Схемные изменения в счетчике сводятся к формированию на входе i -го триггера дизъюнкции переноса из младшего разряда и $D(\vec{\alpha})$. Пусть, например, $M = 10$, $\vec{\alpha} = (0011)$, $\vec{\beta} = (0100)$, $\vec{\gamma} = (1010)$. Тогда на входе триггера в четвертом разряде необходимо реализовать дизъюнкцию $p_3 \vee D(0011)$, а на входах второго и третьего триггеров — соответственно конъюнкции $p_1 \bar{D}(0011)$ и $p_2 \bar{D}(0011)$. Схема такого счетчика приведена на рис. 3.80, а, а временная диаграмма его работы — на рис. 3.80, б.

Исключить запрещенные состояния счетчика можно и несколько иначе: в тех разрядах, где число m содержит единицы — формировать сигналы заема, а в тех разрядах, где число m содержит нули — формировать сигналы переноса. Выход цепи переноса (заема) последнего разряда подключается ко входам триггеров так же, как и в случае исключения последних m запрещенных состояний. Например при $M = 10$, $m = 6$, схема счетчика имеет вид, показанный на рис. 3.81, а, а временная диаграмма его работы приведена на рис. 3.81, б. Из диаграммы видно, что порядок изменения состояний в таком счетчике «неестественный», а именно: из состояния 0000 счетчик переходит в состояние 0001, затем в состояние 1110, затем в 1111, в 1100, 1101 и т. д.

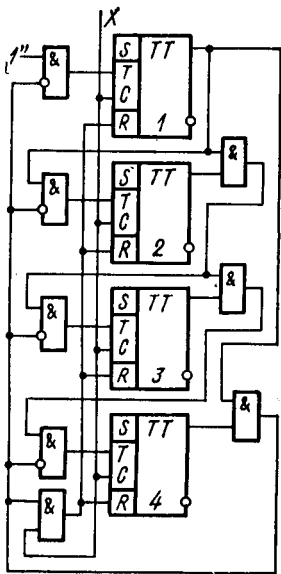


Рис. 3.79

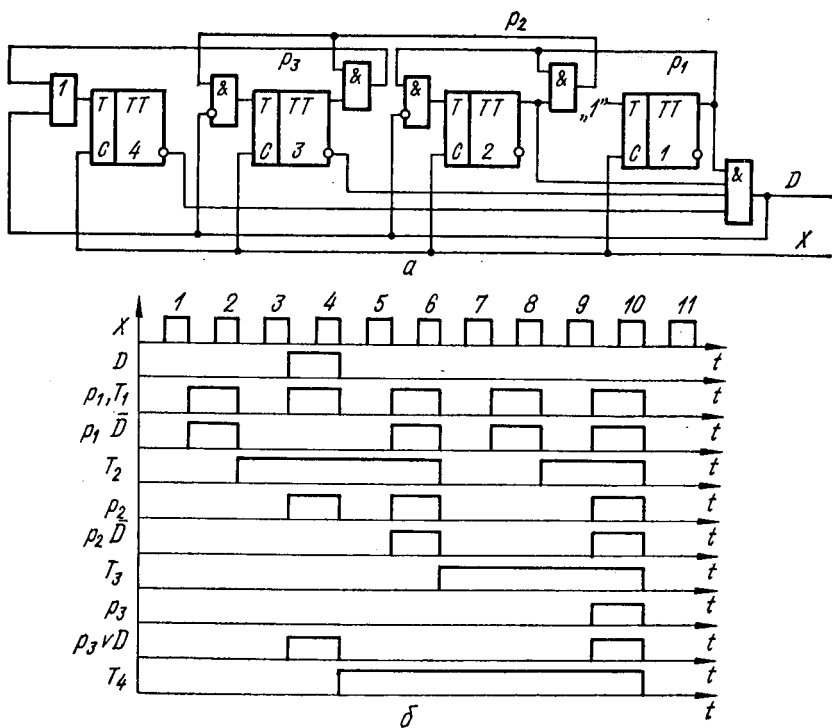


Рис. 3.80

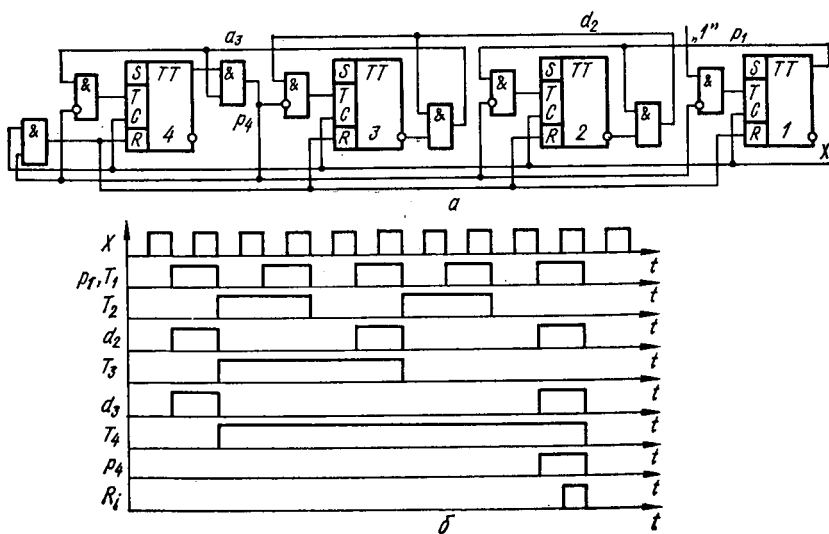


Рис. 3.81

Счетчики с модулем, не равным целой степени числа 2, в литературе получили также название пересчетных схем. Счетчики по модулю 10^i иногда называют счетными декадами. При построении многоразрядных десятичных счетчиков входным сигналом для декады с весом 10^i служит сигнал переноса из декады с весом 10^{i-1} . Для декад, выполненных по схемам на рис. 3.78, 3.80, это сигнал p_4 , а для декад по схемам на рис. 3.79, 3.81 — сигнал D , либо сигнал установки всех триггеров в «0».

На практике часто бывают необходимы счетчики, работающие в специальных кодах, при этом состояния счетчиков соответствуют цифрам некоторой двоичнокодированной системы счисления. Рассмотрим методику проектирования таких счетчиков на следующем примере. Пусть функционирование счетчика по модулю 6 описывается табл. 57, а в качестве элементарных автоматов могут использоваться триггеры RS -, JK - и T -типов. Исходя из табл. 57 можно определить функции возбуждения триггеров. Для этого будем сравнивать их состояния в s -й и $(s+1)$ -й моменты времени. Очевидно, что если $Q_j^{(s)} = 0$, а $Q_j^{(s+1)} = 1$, то функция возбуждения единичного входа j -го триггера S_j' равна 1, а функция возбуждения нулевого входа j -го триггера R_j' равна 0 ($j = 1, 2, \dots, n$). Аналогично, если $Q_j^{(s)} = 1$, а $Q_j^{(s+1)} = 0$, то $S_j' = 0$ и $R_j' = 1$. Если $Q_j^{(s)} = 0$, $Q_j^{(s+1)} = 0$, то $S_j' = 0$, а R_j' может принимать любое значение. При $Q_j^{(s)} = 1$, $Q_j^{(s+1)} = 1$, функция $R_j' = 0$, а S_j' — неопределена. Таким образом, S_j' всегда равна 1 при переходе триггера из 0 в 1 и может быть равна 1 при переходе триггера из 1 в 1. Функция R_j' всегда равна 1 при переходе триггера из 1 в 0 и может быть равна 1 при переходе триггера из 0 в 0. В остальных случаях функции S_j' и R_j' равны 0. Из табл. 57 следует, что

Таблица 57

X	$Q^{(s)}$			$Q^{(s+1)}$		
	Q_3	Q_2	Q_1	Q_3	Q_2	Q_1
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	1	0
1	1	1	0	0	0	1
1	0	0	1	0	1	1
1	0	1	1	0	0	0
—	1	0	1	Избыточные состояния		
—	1	1	1			

$$S_3' = \bar{Q}_3 Q_2 \bar{Q}_1 \vee (Q_3 \bar{Q}_2 \bar{Q}_1); \quad R_3' = Q_3 Q_2 \bar{Q}_1 \vee (\bar{Q}_3 \bar{Q}_2 \bar{Q}_1 \vee \bar{Q}_3 \bar{Q}_2 Q_1 \vee \bar{Q}_3 Q_2 Q_1).$$

В скобках указаны состояния счетчика, при которых соответствующие функции возбуждения могут принимать любое значение. Для других разрядов счетчика имеем

$$S_2' = \bar{Q}_3 \bar{Q}_2 \bar{Q}_1 \vee Q_3 \bar{Q}_2 \bar{Q}_1 \vee \bar{Q}_3 \bar{Q}_2 Q_1; \quad R_2' = \bar{Q}_3 Q_2 \bar{Q}_1 \vee Q_3 Q_2 \bar{Q}_1 \vee \bar{Q}_3 Q_2 Q_1;$$

$$S_1' = Q_3 Q_2 \bar{Q}_1 \vee (\bar{Q}_3 \bar{Q}_2 Q_1); \quad R_1' = \bar{Q}_3 Q_2 Q_1 \vee (\bar{Q}_3 \bar{Q}_2 \bar{Q}_1 \vee \bar{Q}_3 Q_2 \bar{Q}_1 \vee Q_3 \bar{Q}_2 \bar{Q}_1)$$

(здесь и далее переменную X , входящую в состав каждого произведения, будем опускать для упрощения записи). Однако функции S_3' , R_3' ,

S'_2, R'_2, S'_1, R'_1 , являются неполностью определенными. Доопределение их на избыточных наборах переменных Q_3, Q_2 и Q_1 может привести к существенному упрощению их реализации. Построив диаграмму Вейча этих функций, окончательно убеждаемся, что

$$S_3 = \bar{Q}_3 Q_2 \bar{Q}_1; R_3 = Q_3 Q_2; S_2 = \bar{Q}_2; R_2 = Q_2; S_1 = Q_3 Q_2 = R_3; R_1 = Q_2 Q_1.$$

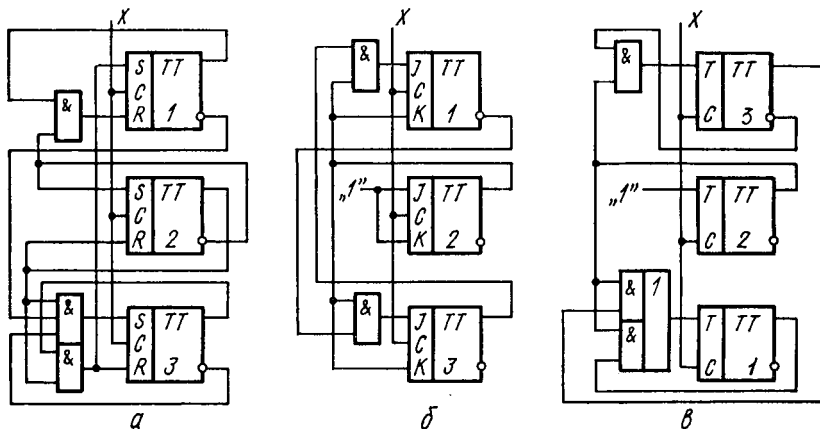


Рис. 3.82

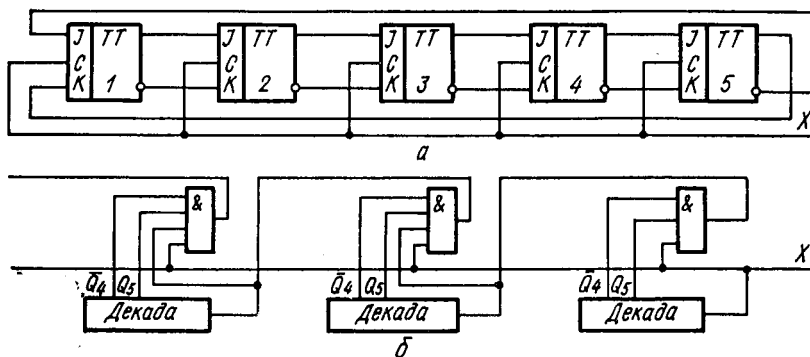


Рис. 3.83

Полученным выражениям соответствует схема счетчика на рис. 3.82, а, построенная на RS-триггерах.

При реализации счетчика на JK-триггерах схема может быть несколько упрощена за счет того, что внутри JK-триггера уже имеются связи, посредством которых на вход J_i подается сигнал Q_i , а на вход K_i — сигнал Q_i . Поэтому для JK-триггеров функции возбуждения рассматриваемого счетчика можно записать так:

$$J_3 = Q_2 \bar{Q}_1; K_3 = Q_2; J_2 = K_2 = 1; J_1 = Q_3 Q_2; K_1 = Q_2.$$

Равенство функций возбуждения $J_i = K_i$ означает, что JK -триггер включен как T -триггер. Схема счетчика для этого случая приведена на рис. 3.82, б.

При построении счетчика на T -триггерах функция возбуждения i -го триггера равна 1 только при переходе триггера из 0 в 1 и из 1 в 0. Поэтому можно записать, что

$$T_3 = S_3'' \vee R_3''; \quad T_2 = S_2'' \vee R_2''; \quad T_1 = S_1'' \vee R_1'',$$

где S_j'' и R_j'' — ранее полученные функции S_j' и R_j' , у которых опущены члены, заключенные в скобки. Минимальные формы функций T_3 , T_2 , T_1 имеют вид

$$T_3 = Q_2 Q_1; \quad T_2 = 1; \quad T_1 = \\ = Q_3 Q_2 \vee Q_2 Q_1.$$

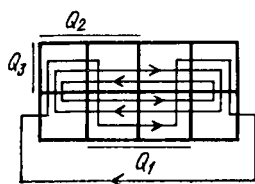


Рис. 3.84

Схема счетчика на T -триггерах приведена на рис. 3.82, в. Рассмотренные счетчики должны иметь цепи внешней установки разрешенного начального состояния, так как при подаче питания на схему триггеры могут устанавливаться произвольно и образовывать запрещенные состояния счетчика.

Счетчики с непозиционным кодированием могут быть построены различными способами. Один из возможных вариантов построения непозиционного десятичного счетчика иллюстрируется схемой на рис. 3.83, а. Здесь используются JK -триггеры, включенные так, что они образуют сдвигающий регистр с пятью разрядами. Благодаря наличию обратной связи с выхода последнего разряда на вход первого (обратная связь типа $\bar{x}_5$), этот регистр вначале последовательно заполняется единицами, а затем нулями. Будем считать, что начальным состоянием счетчика является нулевое. Тогда первый входной сигнал устанавливает $T1$ в «1», так как на его входе J будет единичный сигнал. Запись единицы в первый триггер будет продолжаться до тех пор, пока $T5$ не установится в «1». Вслед за этим начнется запись нуля в $T1$. Следовательно, взаимосвязь между числом входных сигналов и состояниями счетчика может быть определена с помощью табл. 58.

Для образования из таких декад многоразрядных десятичных счетчиков пользуются схемой на рис. 3.83, б. На выходах элементов И

Таблица 58

X	0	1	2	3	4	5	6	7	8	9	10
Q_1	0	1	1	1	1	1	0	0	0	0	0
Q_2	0	0	1	1	1	1	1	0	0	0	0
Q_3	0	0	0	1	1	1	1	1	0	0	0
Q_4	0	0	0	0	1	1	1	1	1	0	0
Q_5	0	0	0	0	0	1	1	1	1	1	0

в такой схеме сигнал переноса с декады может появиться лишь в том случае, когда $Q_5 = 1$, а $Q_4 = 0$, что соответствует состоянию 10000 декады. Таким образом десятый входной сигнал установит декаду в состояние 00000 и поступит на вход следующей декады. В технической литературе такие счетчики называют счетчиками Джонсона, Либау-Крейга, Мебиуса.

Другим примером непозиционных счетчиков являются счетчики с соседним кодированием, в которых два их последовательных состояния отличаются только в одном разряде. Последовательные переходы такого счетчика соответствуют перемещению из любой клетки диаграммы Вейча в любую другую, соседнюю с ней. На диаграмме Вейча (рис. 3.84) показаны некоторые варианты соседнего кодирования состояний трехразрядного счетчика. Реализованы такие счетчики могут быть тем же способом, что и ранее рассмотренные счетчики для двоично-кодированных систем счисления.

3.11. КОЛЬЦЕВЫЕ СЧЕТЧИКИ И НАКАПЛИВАЮЩИЕ СУММАТОРЫ

Кольцевыми счетчиками, или иначе счетчиками с единичным кодированием, называются такие счетчики, у которых число единичных входных сигналов определяется местоположением некоторого определенного сочетания двоичных символов, называемого маркирующим кодом, в регистре. При этом соседние местоположения маркирующего кода определяют числа, отличающиеся на единицу. В качестве маркирующего кода обычно используют слово с одной единицей, реже слово с двумя единицами подряд. Таким образом, кольцевой счетчик в простейшем случае представляет собой сдвигающий регистр с заранее введенным маркирующим кодом, причем первый и последний разряды регистра также связаны между собой цепями сдвига.

Такая схема кольцевого счетчика реализуема во всех системах элементов. Однако в потенциальной системе элементов возможны и другие схемные варианты кольцевых счетчиков. Так, например, в случае использования маркирующего кода 11 можно не удваивать число одноканальных триггеров в сдвигающем регистре. Маркирующий код 11 можно сдвигать на один разряд, исключив съём информации с переключаемых триггеров. Это достигается применением расщепленных цепей сдвига (рис. 3.85, а). Модуль таких счетчиков должен быть четным, так как здесь происходит попеременный сдвиг четных и нечетных разрядов. Такой сдвиг обеспечивается путем различной тактировки цепей сдвига, для чего служит специальный тактирующий триггер $T7$. В исходном состоянии (при отсутствии сигналов X) триггеры $T5$ и $T6$ сигналом $У$ «0» устанавливаются в «1», а остальные триггеры в «0». Временная диаграмма работы такого счетчика приведена на рис. 3.85, б. Для построения на основе этой схемы счетчика с единичным маркирующим кодом достаточно подключить к соседним выходам триггеров двувходовые элементы И.

Кольцевые счетчики можно также строить на основе схем с многими устойчивыми состояниями, составленных из элементов типа И—НЕ и ИЛИ—НЕ. Например, схема на рис. 3.86 имеет четыре

устойчивых состояния: 1000, 0100, 0010, 0001. Подобные схемы получили название полусчетных колец (ПК). Кольцевой счетчик из полусчетных колец при нечетном модуле M может быть построен по анало-

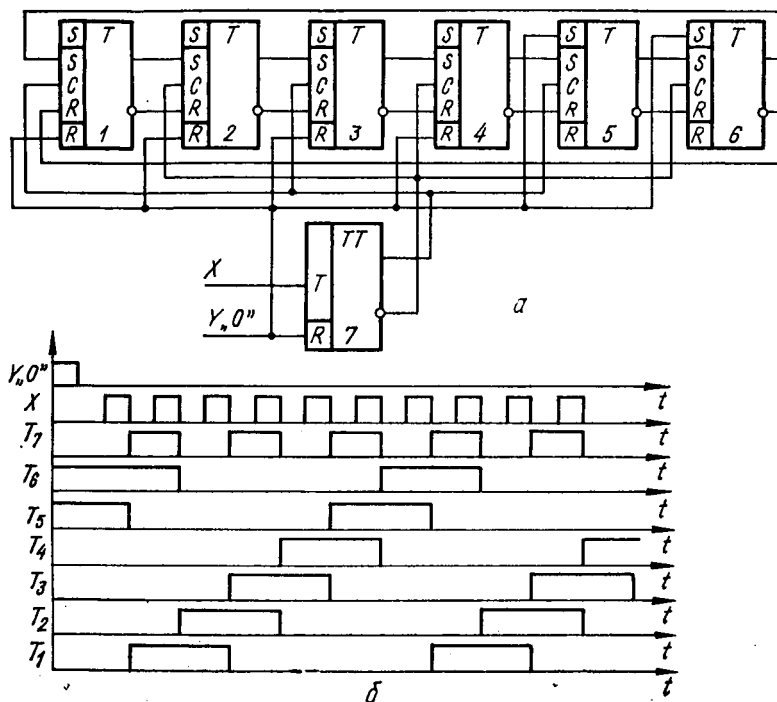


Рис. 3.85

гии со схемой на рис. 3.87. Процесс переключения такого кольцевого счетчика иллюстрируется табл. 59, где запись $b_i/\tau 1 = 1$ означает, что в соответствующей строке таблицы указано состояние выходов элементов b_i при $\tau 1 = 1$ и т. д. Если модуль кольцевого счетчика четный, то схему счетчика на основе полусчетных колец можно упростить. Полусчетное кольцо в этом случае строится по аналогии со схемой на рис. 3.88, а сама схема счетчика приведена на рис. 3.89. Состояния некоторых элементов этой схемы при ее переключении указаны в табл. 60. За исходное состояние полусчетного кольца принято 100...00.

Таблица 59

Сигналы в схеме	i				
	1	2	3	4	5
a_i	1	0	0	0	0
$d_i/\tau 2 = 0$	0	0	0	0	0
$d_i/\tau 2 = 1$	1	0	1	1	1
c_i	0	1	0	0	0
$b_i = \tau 1 = 0$	0	0	0	0	0
$b_i/\tau 1 = 1$	1	0	1	1	1
a_i	0	1	0	0	0
$d_i/\tau 2 = 1$	1	1	0	1	1
c_i	0	0	1	0	0
$b_i/\tau 1 = 1$	1	1	0	1	1
a_i	0	0	1	0	0

Важным достоинством кольцевых счетчиков на основе полусчетных колец является их большая экономичность по сравнению со схемами на обычных триггерах. Недостаток таких счетчиков — пропорциональный рост числа входов некоторых логических элементов с увеличением модуля M . От этого недостатка можно избавиться, если строить кольцевые счетчики по большому модулю M из кольцевых счетчиков по модулю $h < M$ в соответствии со схемой на рис. 3.90. В такой схеме первые $h - 1$

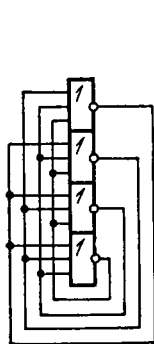


Таблица 60

a_i	100...00	a_i	010...00	a_i	0010...00
$S/\tau 2 = 1$	1	$S/\tau 2 = 1$	0	$S/\tau 2 = 1$	1
$R/\tau 2 = 1$	0	$R/\tau 2 = 1$	1	$R/\tau 2 = 1$	0
T	1	T	0	T	1
$A/\tau 1 = 1$	1	$A/\tau 1 = 1$	0	$A/\tau 1 = 1$	1
$B/\tau 1 = 1$	0	$B/\tau 1 = 1$	1	$B/\tau 1 = 1$	0

Рассмотрим еще один способ построения кольцевых счетчиков — так называемых счетчиков-дешифраторов. В таких кольцевых счетчиках совмещены в одной схеме счетчик с обычным позиционным кодированием и дешифратор.

Осуществляется такое совмещение заменой вспомогательного ряда триггеров в n -разрядном счетчике 2^n триггерами, функции возбуждения которых являются функциями дешифрации состояний триггеров основного ряда. Пусть, например, необходимо построить счетчик-дешифратор по модулю 8. Будем кодировать состояния основных триггеров такого счетчика числами натурального ряда в двоичной позиционной однородной системе счисления. Тогда таблица состояний основных триггеров будет иметь вид, показанный в табл. 61. Функции возбуждения основных триггеров можно записать как

Таблица 61

Состояние основных триггеров		
$Q^{(s)}$	Q_3	0 0 0 0 1 1 1 1
	Q_2	0 0 1 1 0 0 1 1
	Q_1	0 1 0 1 0 1 0 1
$Q^{(s+1)}$	Q_3	0 0 0 1 1 1 1 0
	Q_2	0 1 1 0 0 1 1 0
	Q_1	1 0 1 0 1 0 1 0

$$\begin{aligned}
 S_3 &= \bar{Q}_3 Q_2 Q_1; & K_3 &= Q_3 Q_2 Q_1; & S_2 &= \bar{Q}_3 \bar{Q}_2 Q_1 \vee Q_3 \bar{Q}_2 Q_1; \\
 R_2 &= \bar{Q}_3 Q_2 Q_1 \vee Q_3 Q_2 Q_1; & S_1 &= \bar{Q}_3 \bar{Q}_2 \bar{Q}_1 \vee \bar{Q}_3 Q_2 \bar{Q}_1 \vee Q_3 \bar{Q}_2 \bar{Q}_1 \vee Q_3 Q_2 \bar{Q}_1 \\
 R_1 &= \bar{Q}_3 \bar{Q}_2 Q_1 \vee \bar{Q}_3 Q_2 Q_1 \vee Q_3 \bar{Q}_2 Q_1 \vee Q_3 Q_2 Q_1.
 \end{aligned}$$

Как нетрудно заметить, каждый дизъюнктивный член в этих выражениях является не чем иным, как функцией дешифрации состояний триггеров. Введем теперь для запоминания значений функций дешифрации отдельные триггеры $Q'_1, Q'_2, \dots, Q'_8$. При этом на триггере Q'_1 хранится значение функции $\bar{Q}_3\bar{Q}_2\bar{Q}_1$, на триггере Q'_2 — значение функции $Q_3Q_2Q_1$ и т. д. Тогда функции возбуждения основных триггеров можно переписать в следующем виде:

$$\begin{aligned} S_3 &= Q'_4; & R_3 &= Q'_8; & S_2 &= Q'_2 \vee Q'_6; & R_2 &= Q'_4 \vee Q'_8; \\ S_1 &= Q'_1 \vee Q'_3 \vee Q'_5 \vee Q'_7; & R_1 &= Q'_2 \vee Q'_4 \vee Q'_6 \vee Q'_8. \end{aligned}$$

Схема рассматриваемого счетчика-дешифратора показана на рис. 3.91. При построении счетчика-дешифратора, модуль которого

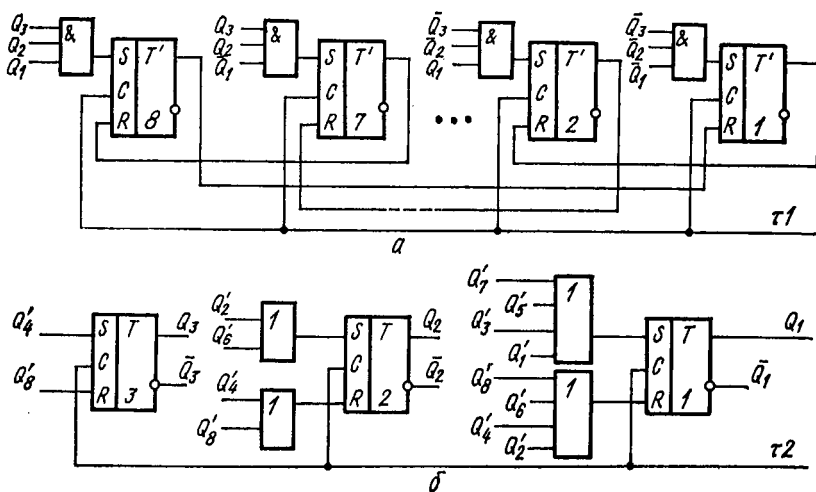


Рис. 3.91

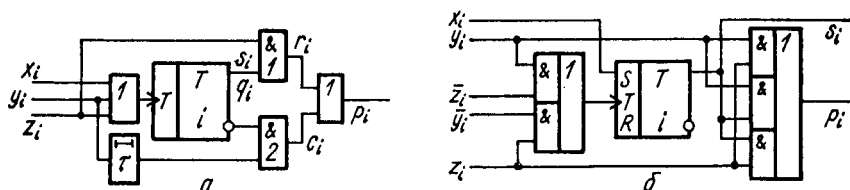


Рис. 3.92

не является целой степенью числа 2, на один из рядов триггеров необходимо подключить цепи установки разрешенного начального состояния, так как в этом случае при подаче питания на схему триггеры могут устанавливаться в произвольные состояния, в том числе и в такие, которые не будут дешифрироваться.

Накапливающим сумматором будем называть такой сумматор, который осуществляет арифметическое сложение слов X и Y при подаче их на сумматор одного за другим, независимо от того, каким кодом

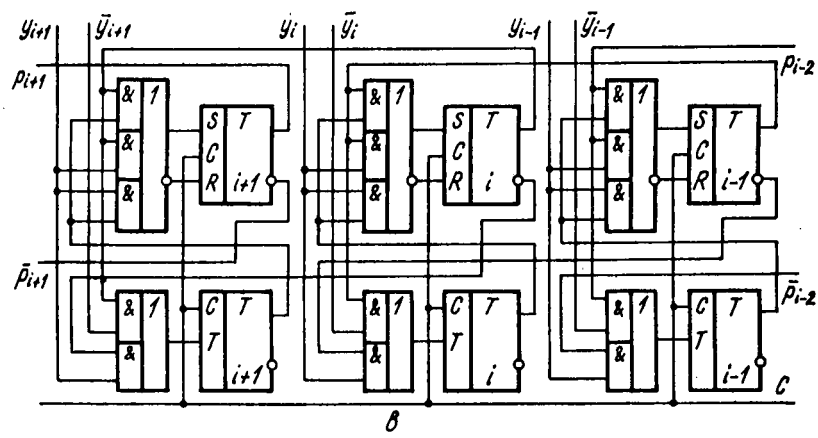
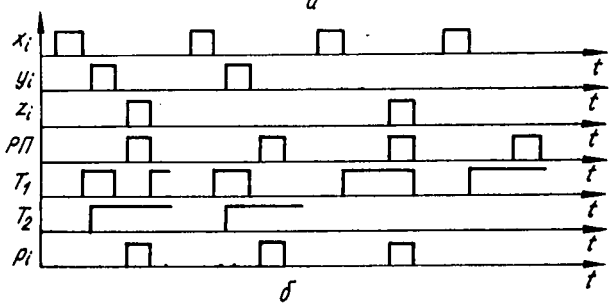


Рис. 3.93

$$\begin{aligned} c_i &= y_i \bar{q}_i = y_i (\overline{x_i \bar{y}_i \vee \bar{x}_i y_i}) = y_i (\overline{\bar{x}_i y_i}) (\overline{\bar{x}_i \bar{y}_i}) = \\ &= y_i (x_i \vee \bar{y}_i) (\bar{x}_i \vee y_i) = x_i y_i y_i = x_i y_i. \end{aligned}$$

Из этого выражения следует, что значение q_i должно сформироваться на триггере раньше, чем сигнал y_i достигнет выхода элемента И2. Поэтому необходима задержка τ сигналов y_i на время, большее суммы времени переключения триггера и задержки сигналов по его входному каналу. Вторая составляющая переноса r_i формируется при равенстве 1 суммы q_i и сигнала переноса z_i

Таблица 62

	Регистр суммы	Регистр переносов
$Y_1 = 3$	00000000 00000011 00000000	00000000 ←
X_1 $Y_2 = 2$	00000011 00000010 00000000	00000000 ←
X_2 $Y_3 = 140$	00000001 10001100 00000100	00000010 ←
X_3 $Y_4 = 65$	10001001 01000001 00001000	00000100 ←
X_4	11000000 00010010	00001001 ←
Сумма	11010010	

$$r_i = q_i z_i = \bar{x}_i y_i z_i \vee x_i \bar{y}_i z_i.$$

Следовательно, сигнал z_i должен достичь выхода элемента И1 раньше момента переключения триггера. Если время переключения триггера значительно больше задержки сигналов на одном логическом элементе, то это условие выполняется. Однако для надежной работы сумматора иногда вводят дополнительную задержку сигналов на входе T . Таким образом,

$$\begin{aligned} p_i &= x_i y_i \vee \bar{x}_i y_i z_i \vee x_i \bar{y}_i z_i = \\ &= x_i y_i z_i \vee x_i y_i \bar{z}_i \vee \bar{x}_i y_i z_i \vee \\ &\vee x_i \bar{y}_i z_i = x_i y_i \vee x_i z_i \vee y_i z_i, \end{aligned}$$

что является известным выражением для сигналов переноса. В сумматорах накапливающего типа время суммирования составляет три такта. Поскольку результат суммирования запоминается на триггерах, то такие сумматоры позволяют сравнительно просто осуществить суммирование многих чисел с накоплением результата.

На рис. 3.92, б приведена схема одного разряда так называемого комбинационно-накапливающего сумматора. В таком сумматоре перенос формируется комбинацион-

ной схемой, а окончательная сумма s_i образуется на триггере, на счетный вход которого подается результат суммирования по mod 2 цифр y_i и z_i . Сигнал переноса p_i формируется еще до момента установления результата на триггере. Суммирование в такой схеме выполняется за два такта.

Одноразрядные сумматоры по схемам на рис. 3.92 реализуемы лишь на импульсно-потенциальных элементах. Накапливающий одноразрядный сумматор на потенциальных элементах можно построить в соответствии со схемой на рис. 3.93, а, где РП — управляющий сигнал разрешения переноса, подаваемый на схему синхронного с сигналом z_i . Временная диаграмма работы такого сумматора при некото-

ых комбинациях значений входных сигналов приведена на рис. 3.93, б.

При построении параллельных накапливающих сумматоров необходимы отдельные сумматоры для каждого разряда, соединенные между собой цепями переноса. При этом для организации цепей переноса применимы все известные методы, используемые в комбинационных сумматорах и счетчиках.

Для ускорения суммирования при выполнении операций, требующих последовательного накопления результата (например, при умножении и делении) используют параллельные накапливающие сумматоры с запоминанием сигналов переноса (рис. 3.93, в). Такой сумматор имеет в своем составе два регистра — регистр суммы, в котором запоминаются промежуточные результаты суммирования, и регистр переносов. При выполнении очередного суммирования сигнал переноса из i -го разряда не прибавляется к $(i + 1)$ -му разряду суммы, а запоминается в i -м разряде регистра переносов. В каждом новом акте суммирования, задаваемом синхронизирующим сигналом C , в i -м разряде производится сложение трех цифр: i -й цифры из регистра суммы, $(i - 1)$ -й цифры из регистра переносов и i -й цифры нового слагаемого. Поэтому перенос в таком сумматоре не распространяется дальше, чем на один разряд. Для получения окончательного результата необходимо суммировать содержимое регистра суммы и регистра переносов на сумматоре с обычными цепями переноса. Пример выполнения сложения $Y_1 = 3$, $Y_2 = 2$, $Y_3 = 140$, $Y_4 = 65$ на таком сумматоре приведен в табл. 62, где буквой X_i обозначены промежуточные суммы.

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА ЦИФРОВЫХ ЭВМ

4.1. НАЗНАЧЕНИЕ, КЛАССИФИКАЦИЯ И ОСНОВНЫЕ ПАРАМЕТРЫ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Запоминающие устройства (ЗУ) служат для фиксации, хранения и выдачи информации, необходимой для решения задач на ЭВМ.

Процесс фиксации информации в ЗУ называют записью (З), а процесс выдачи — считыванием или просто чтением (Ч). Запись и чтение являются основными операциями в ЗУ. Независимо от назначения

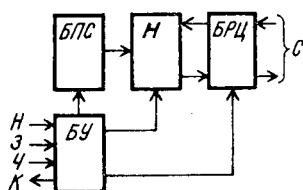


Рис. 4.1

и принципов построения в любом ЗУ можно выделить следующие четыре блока (рис. 4.1): блок поиска слов (БПС), накопитель (Н), блок разрядных цепей (БРЦ) и блок управления (БУ). Сигналы З и Ч задают вид выполняемой операции, а Н и К — ее начало и конец.

Накопителем называют ту часть ЗУ, в которой происходит фиксация и хранение слов и из которой при необходимости можно считать записанное ранее слово. Накопитель разбит на отдельные ячейки в каждой из которых может храниться слово, часть слова или несколько слов. Далее, за исключением специально оговоренных случаев, будем считать, что в одной ячейке хранится одно слово.

Блоком поиска слов называют часть ЗУ, осуществляющую поиск ячейки, в которой хранится искомое слово или в которую надо записать заданное слово. По способам поиска слов различают адресные, ассоциативные и безадресные ЗУ. В адресных ЗУ искомое слово определяется координатами ячейки в пространстве или во времени. В таких ЗУ каждой ячейке присваивают свой номер, который называют адресом. Поиск слова осуществляется путем задания адреса ячейки, где хранится искомое слово. Если адрес представляет собой пространственные координаты ячейки, то такой поиск слов называют адресным М-поиском (S-поиском) — по адресу места ячейки в ЗУ. Если же адрес является временными координатами ячейки, то такой поиск называют адресным В-поиском (Т-поиском) — поиском во времени. Известны и адресные ЗУ с комбинированным МВ-поиском, когда, например, некоторый массив слов находят с помощью В-поиска, слово в этом массиве — по М-поиску. В ассоциативном ЗУ поиск ведется по некоторым признакам искомого слова, т. е. по его содержанию. Таким признаком может быть особая часть искомого слова, придаваемая слову специально для обнаружения его среди других слов. Существуют также адресно-ассоциативные ЗУ, где одной основной операции соответствует адресный поиск, а другой операции — ассо-

циативный, либо же все операции реализуемы как с адресным, так и с ассоциативным поиском. В адресных ЗУ блок БПС называют адресными цепями, в ассоциативных ЗУ — ассоциативным накопителем. В безадресных ЗУ запись и чтение осуществляют в раз и навсегда заданной последовательности. Чаще всего используют следующие два варианта организации поиска слов в безадресных ЗУ:

1) чтение слов осуществляют в той последовательности, в которой слова были записаны в ЗУ, т. е. по принципу «первым записано, первым и считано»;

2) чтение слов осуществляют в последовательности, обратной той, в которой они были записаны в ЗУ, т. е. по принципу «первым записано, последним считано».

Блоком разрядных цепей называют ту часть ЗУ, по которой слово (С) при записи поступает в накопитель, а при чтении передается с накопителя на выход ЗУ. Существуют ЗУ, в которых некоторые функции БПС и БРЦ выполняются на одном и том же оборудовании. Такие блоки называют адресно-разрядными цепями.

Блоком управления называют ту часть ЗУ, которая по сигналу «Начало» операции (Н) и по коду операции «Запись» или «Чтение» вырабатывает последовательности управляющих сигналов, необходимые для выполнения заданной операции. Завершение операции фиксируется по сигналу «Конец» (К) на отдельном выходе БУ.

По способам доступа к отдельным ячейкам различают ЗУ с последовательным доступом, в которых чтение и запись осуществляются в некоторой последовательности, определяемой расположением ячеек в накопителе, и ЗУ с произвольным доступом, в которых основные операции можно выполнять в любой момент времени и использовать при этом любую ячейку.

По кратности записи информации выделяют ЗУ со стирающейся записью (активные ЗУ) и ЗУ с нестирающейся записью (односторонние, пассивные, постоянные ЗУ). В активных ЗУ запись и чтение могут производиться любое число раз вплоть до естественного износа носителя информации. В постоянные ЗУ информация записывается один раз (например, при изготовлении), а считывание может выполняться любое число раз.

В некоторых типах ЗУ чтение информации сопровождается ее разрушением. Поэтому различают ЗУ с разрушением и без разрушения информации при чтении.

Отдельные типы ЗУ позволяют сохранять информацию при отключении энергоснабжения. В соответствии с этим различают ЗУ, сохраняющие и не сохраняющие информацию при отключении питания. ЗУ могут быть классифицированы и по другим признакам, например, по их роли в вычислительном процессе.

Основными параметрами, определяющими качество ЗУ, являются емкость, быстродействие, стоимость и надежность. Для измерения емкости ЗУ используют единицы, указанные в п. 1.1. Кроме того часто емкость ЗУ выражают числом слов заданной длины. Наряду с емкостью используются понятия поверхностной и объемной плотности записи информации в ЗУ, которые измеряются числом бит соответственно на квадратный и кубический сантиметр.

Быстродействие ЗУ определяется его временными характеристиками. Выполнение одной операции называют обращением к ЗУ, а полное время ее выполнения — циклом обращения t_o . Так как в ЗУ выполняют две основные операции (чтение и запись), то различают цикл обращения при чтении $t_{оч}$ и записи $t_{оз}$. Промежуток времени от начала обращения при чтении до момента появления слова на выходах ЗУ называют временем выборки t_b . Циклы обращения к ЗУ связаны с другими временными характеристиками ЗУ соотношениями

$$t_{оз} = t_n + t_c + t_z; \quad t_{оч} = t_n + t_ч + t_p; \quad t_b = t_n + t_ч,$$

где t_n — время поиска ячейки в ЗУ; $t_ч$ — время считывания слова с ячейки; t_c — время стирания информации (установки ячейки в исходное состояние); t_z — время записи слова в ячейку; t_p — время регенерации (восстановления) информации (для ЗУ с неразрушающим чтением $t_p = 0$).

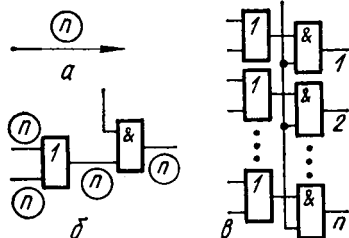


Рис. 4.2

Другими важнейшими временными характеристиками ЗУ являются скорость записи v_z и чтения $v_ч$ информации, выражаемые в битах за секунду. Обычно $v_z = v_ч$ и определяется максимально возможной скоростью записи информации в последовательно расположенные ячейки (т. е., адрес последующей ячейки на единицу больше адреса предыдущей ячейки).

Надежность ЗУ оценивается как по отказам, так и по сбоям. Количественной оценкой надежности по отказам служит вероятность безотказной работы ЗУ в заданных условиях в пределах заданного интервала времени, а по сбоям — число сбоев на некоторый объем информации, например, на 10^7 бит.

Стоимость ЗУ оценивается как общей стоимостью C всего ЗУ, так и приведенной стоимостью C' хранения одного бита информации в ЗУ

$$C' = \frac{C}{N},$$

где N — емкость ЗУ в битах.

Введем необходимые для дальнейшего условные обозначения. Канал связи будем обозначать линией со стрелкой. Если канал состоит более, чем из одной шины, то рядом с линией будем указывать число шин в канале (рис. 4.2, а). Рис. 4.2, б представляет собой пример «свернутого» изображения схемы на рис. 4.2, в.

4.2. СТРУКТУРНЫЕ СХЕМЫ АДРЕСНЫХ ЗУ С М-ПОИСКОМ

Записать информацию в ЗУ — это значит поставить в соответствие буквам структурного алфавита, из которых составлены слова, состояния некоторых объектов, имеющих в общем случае k устойчивых состояний, способных сохранять эти состояния неограниченно

долго и переходить из одного состояния в другое под воздействием внешних сигналов. Такие объекты получили название запоминающих элементов (ЗЭ). Если $k = 2$, то ЗЭ можно представить моделью— «черным ящиком» с определенным числом входов и выходов, по которым действуют сигналы, принимающие только два значения 0 и 1. Такое представление ЗЭ обуславливает независимость вида структурных схем ЗУ для самых различных физических эффектов и явлений, положенных в основу их построения. Каждый ЗЭ предназначен для хранения одного бита. Входы ЗЭ, служащие для записи информации, будем называть установочными. Сигналы, действующие на установочные входы ЗЭ, должны обеспечить режимы записи 0, записи 1 и хранения информации. Следовательно, число установочных входов ЗЭ должно быть не менее 2. Большое число установочных входов требует большого числа шин управления, что, в свою очередь, вызывает

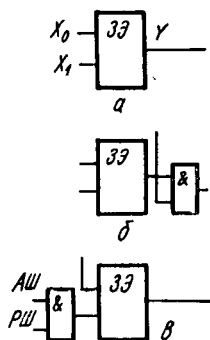


Рис. 4.3

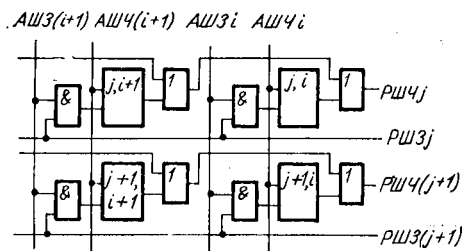


Рис. 4.4

значительные технологические трудности реализации ЗЭ и усложняет управление ими. Поэтому применяют, как правило, ЗЭ с минимальным числом установочных входов X_0 , X_1 и одним выходом Y (рис. 4.3, а). Сигнал $X_0 = 1$ устанавливает ЗЭ в состояние 0 ($Y = 0$), а сигнал $X_1 = 1$ в состояние 1 ($Y = 1$). Режим хранения обеспечивается сигналами $X_0 = X_1 = 0$, а одновременное действие сигналов $X_0 = X_1 = 1$ недопустимо.

Наиболее распространенными способами считывания информации с ЗЭ является способ установки ЗЭ в нулевое состояние путем подачи сигнала $X_0 = 1$ и способ считывания информации при помощи элемента И путем опроса значения Y (рис. 4.3, б). При первом способе считывание 1 происходит путем фиксации перехода сигнала Y из 1 в 0, а считывание 0 происходит при неизменном значении Y . Второй способ обеспечивает считывание без разрушения информации, чего, как правило, не удастся достичь при первом способе. В некоторых случаях при первом способе считывания используют нестандартный сигнал $X_0 = 1$ (например, малой длительности), за время действия которого ЗЭ не успевает изменить состояние 1 на 0 и после окончания действия сигнала X_0 возвращается в исходное состояние (например, триггер) или близкое к исходному (например, сердечник с прямоугольной петлей гистерезиса, обладающий множеством состояний безразличного равновесия). В последнем случае считывание без разрушения может производиться ограниченное число раз. Все ЗЭ объеди-

няются в ленточную, плоскую или объемную схему, образуя накопитель ЗУ.

Накопитель имеет систему адресных и разрядных проводников (шин). Адресные шины (АШ) служат для выбора искомой ячейки ЗУ, т. е., для выборки среди ЗЭ накопителя тех элементов, из которых состоит искомая ячейка. Как правило, данная адресная шина соединена со всеми ЗЭ данной ячейки. Различают АШ записи и АШ чтения. В некоторых типах ЗУ одни и те же адресные шины служат как для записи, так и для чтения.

Местоположение отдельных разрядов в накопителе определяют разрядные шины (РШ), которые соединены с выходами всех ЗЭ, относящихся к одному разряду всех (или группы) ячеек. Различают РШ записи и чтения. Последние называют также шинами воспроизведения (выходными, сигнальными). В некоторых типах ЗУ РШ чтения и записи удается объединить в одну разрядную шину. АШ и РШ носят общее название шин выборки. В зависимости от числа таких шин, соединенных с одним ЗЭ, различают двумерные (две шины выборки), трехмерные (три шины) и т. д. накопителе.

Для записи информации в ЗЭ к его входу X_1 должен быть подключен элемент И (рис. 4.3, в). Запись 1 в ЗЭ происходит при совпадении единичных сигналов по РШ и АШ. Запись 0 осуществляется путем сброса в 0 всех ЗЭ ячейки, для чего непосредственно перед записью осуществляется стирание информации. Запись 0 и 1 можно осуществить одновременно (без предварительного стирания информации), если имеется элемент И, подключенный ко входу X_0 ЗЭ.

Рассмотрим в качестве примера схему накопителя (рис. 4.4) на ЗЭ с рис. 4.3, в. Здесь АШ чтения (АШЧ) и АШ записи (АШЗ) объединяют все ЗЭ, принадлежащие одной ячейке ЗУ. Выходы ЗЭ с элементами ИЛИ образуют РШ чтения (РШЧ). РШ записи (РШЗ) и РШЧ объединяют все ЗЭ, принадлежащие одноименным разрядам всех ячеек. Поиск нужной ячейки как при записи, так и при чтении осуществляется путем задания ее адреса, для запоминания которого в составе ЗУ должен быть специальный регистр адреса (RGA). Преобразование адреса в сигнал по одной из АШ накопителя осуществляется с помощью дешифратора адреса (ДСА), включенного между RGA и накопителем. Считанное из искомой ячейки слово передается по РШЧ в специальный регистр слов (RGC). Подлежащие записи слова также предварительно записываются в RGC, а затем по РШЗ передаются в нужную ячейку ЗУ. Кроме перечисленных основных узлов, в состав ЗУ (рис. 4.5) входят также вспомогательные схемы, предназначенные для формирования сигналов требуемой длительности, фазы и амплитуды в адресных и разрядных цепях, которые впредь будем называть усилителями записи (УЗ) и чтения (УЧ), а также блок местного управления (БМУ), служащий для обеспечения необходимой последовательности работы узлов ЗУ. Число ячеек в накопителе, а следовательно, и число пар АШЧ, АШЗ равно числу выходов ДСА и равно $\frac{N}{n}$, где n — длина ячейки ЗУ. Для задания всех

$\frac{N}{n}$ адресов длина регистра адреса должна быть не менее $\log_2 \frac{N}{n}$ разрядов.

Стоимость C такого ЗУ емкостью N бит можно подсчитать по следующей формуле:

где C_1 — стоимость одного ЗЭ; C_2 — суммарная стоимость одного разрядного УЗ, УЧ и одного разряда RGC; C_3 — суммарная стоимость


$$C_n = C_0 \left(\frac{N}{n} + 2 \sqrt{\frac{N}{n}} + 4 \sqrt[4]{\frac{N}{n}} + \dots \right) \approx C_0 \frac{N}{n} + 2C_0 \sqrt{\frac{N}{n}},$$
$$n_0 \approx \sqrt{N \frac{C_3}{C_2}}.$$
$$33 \leq n_0 \leq 1000. \quad (4.2)$$

Нижний предел оптимальной длины ячейки ЗУ хорошо согласуется с общепринятой длиной n_c машинного слова ($n_c = 30 \div 40$ бит), в то время как верхний предел на порядок больше n_c . Поэтому если $n_c < n_0$, то с целью минимизации стоимости ЗУ следует увеличивать длину RG до величины, близкой к n_0 и кратной n_c . Например, если $n_0 = 130$, а $n_c = 30$, то длину RG следует выбирать равной 120 разрядам, т. е. за одно обращение к ЗУ будет выбираться одновременно 4 слова. Окончательную выборку слов можно производить на RG . Структурная схема такого ЗУ показана на рис. 4.6. Его особенностью является наличие двух коммутаторов $K1$ и $K2$, двух регистров

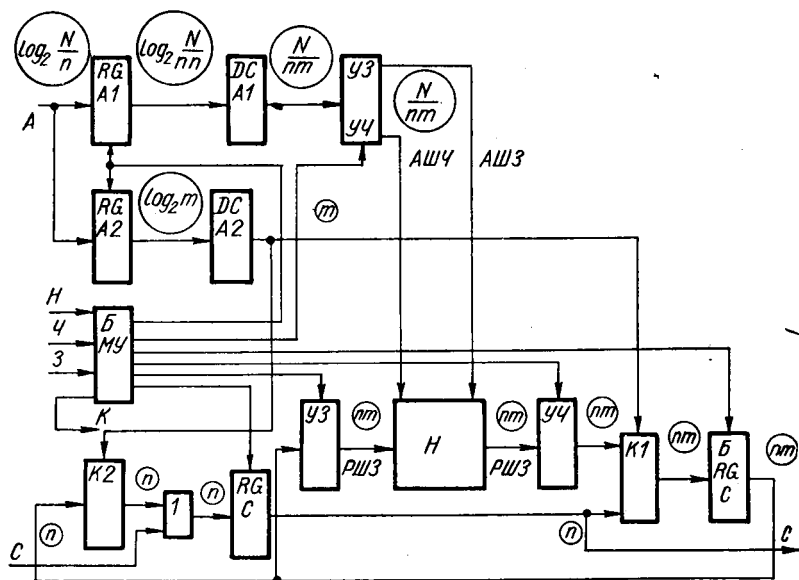


Рис. 4.6

адреса $RG A1$ и $RG A2$ со своими дешифраторами $DA1$ и $DA2$, буферного регистра слов $BRGC$. В $RG A1$ записываются старшие разряды адреса, а в $RG A2$ — младшие. Старшие разряды адреса выбирают нужную ячейку ЗУ, а младшие разряды адреса — искомое слово из m слов, записанных в одной ячейке. Входной коммутатор $K1$ представляет собой мультиплексор и служит для записи слова в $BRGC$. Выходной коммутатор является селектором и служит для выбора искомого слова из $BRGC$. Управляются коммутаторы $K1$ и $K2$ через $DA2$ младшими разрядами адреса. Например, если в $BRGC$ записано 8 слов, то $DA2$ имеет 8 выходов, а $RG A2$ является трехразрядным регистром. Нижний предел оптимальной длины ячейки такого ЗУ совпадает со значением нижнего предела из (4.2), верхний же предел несколько ниже, так как в функции стоимости данного ЗУ появляются члены, учитывающие стоимость коммутаторов.

Удлиненный регистр $BRGC$ в ЗУ по схеме на рис. 4.6 необходим только в том случае, если чтение происходит с разрушением, так как $BRGC$ используется для регенерации. Если чтение осуществ-

ляется без разрушения информации, то структура ЗУ несколько упрощается (рис. 4.7). В данном случае отпадает необходимость в *BRGC*. Пределы оптимальной длины ячейки для такого ЗУ несколько выше, чем указанные неравенством (4.2).

ЗУ по схемам на рис. 4.5—4.7 получили название ЗУ с прямой (непосредственной, линейной, типа *Z*) выборкой слов. Так как число шин выборки в таких ЗУ равно двум (одна адресная и одна разрядная), то их называют также двумерными или типа 2Д (от слова *dimension*). Весьма существенным недостатком таких ЗУ является то, что число адресных усилителей записи и чтения равно числу выходов ДСА и равно числу ячеек накопителя. Незначительное усложнение

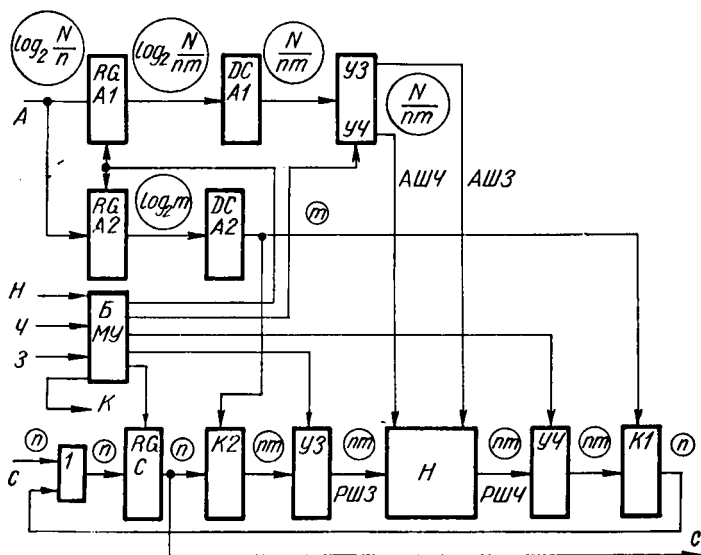


Рис. 4.7.

схемы ЗЭ (рис. 4.8, а) позволяет резко сократить число адресных усилителей за счет перенесения последней ступени ДСА непосредственно в накопитель. Ко входам элемента И1 подключены две адресные шины, а ко входам элемента И2 — две адресные и одна разрядная шины. По числу шин выборки ЗУ такого типа получили название трехмерных или сокращенно ЗД. В структурной схеме трехмерного ЗУ (рис. 4.8, б) имеется два дешифратора адреса ДСАХ и ДСАУ. Так как последний каскад дешифратора адреса реализован непосредственно в накопителе, то общее число выходов ДСАХ и ДСАУ равно $2\sqrt{\frac{N}{n}}$, где $\frac{N}{n}$ — число ячеек в ЗУ. Функция стоимости такого ЗУ имеет вид

$$C = C_1 N + C_2 n + 2C_3 \sqrt{\frac{N}{n}} + 4C_4 \sqrt[4]{\frac{N}{n}} + C_5 + C_6 \log_2 \frac{N}{n}. \quad (4.3)$$

Все буквы в этом выражении имеют такой же смысл, так и в (4.1). Оптимальную длину ячейки для ЗУ типа ЗД можно определить по

формуле

$$n_0 \approx \sqrt[3]{N \frac{C_2^2}{C_3}},$$

полученной аналогично (4.1). При сделанном ранее предположении относительно величин C_2 и C_3 получим

$$3 \leq n_0 \leq 50.$$

Следовательно, в ЗУ с выборкой 3Д верхний предел n_0 хорошо согласуется с общепринятой длиной машинного слова, в то время как

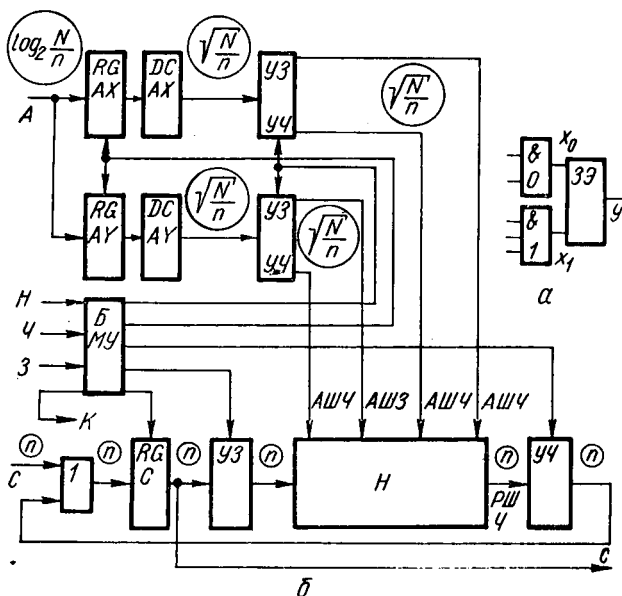


Рис. 4.8

нижний предел на порядок меньше n_c . Поэтому, если $n_c > n_0$, то, с целью минимизации стоимости ЗУ, машинное слово разбивают на слоги, близкие по длине к n_0 , и выборку слов из ЗУ производят последовательно по слогам. ЗУ с выборкой 3Д называют также матричными.

Дальнейшего упрощения адресных цепей можно добиться в ЗУ с многомерной выборкой (4Д, 5Д и т. д.), однако при этом возникают большие технологические трудности реализации многоходовых элементов И на входах ЗЭ. Поэтому ЗУ с многомерной выборкой не получили распространения. Более того, при построении быстродействующих ЗУ трудно реализовать даже трехходовой элемент И в ЗУ с выборкой 3Д. Вследствие этого в последнее время получили распространение ЗУ с комбинированной выборкой, которые занимают промежуточное положение между ЗУ с выборкой 2Д и 3Д и получившие название ЗУ «два с половиной Д» (сокращенно 2,5Д). Особенностью их является то, что трехходовой элемент И на входе ЗЭ реализуется

последовательным соединением двух двухвходовых элементов И, один из которых (ИЗ на рис. 49, а) выносится за пределы накопителя. На первый вход элемента ИЗ подключается разрядная шина, а на второй — адресная. Поэтому выходные шины элементов ИЗ получили название адресно-разрядных шин (АРШ). В таком ЗУ все ЗЭ, относящиеся к одноименным разрядам всех слов, группируются в отдельные разрядные накопители. Вследствие этого накопитель ЗУ с выборкой 2,5Д состоит из n разрядных накопителей (рис. 4.9, б). Каждый разрядный накопитель имеет один усилитель чтения (УЧ). Все элементы

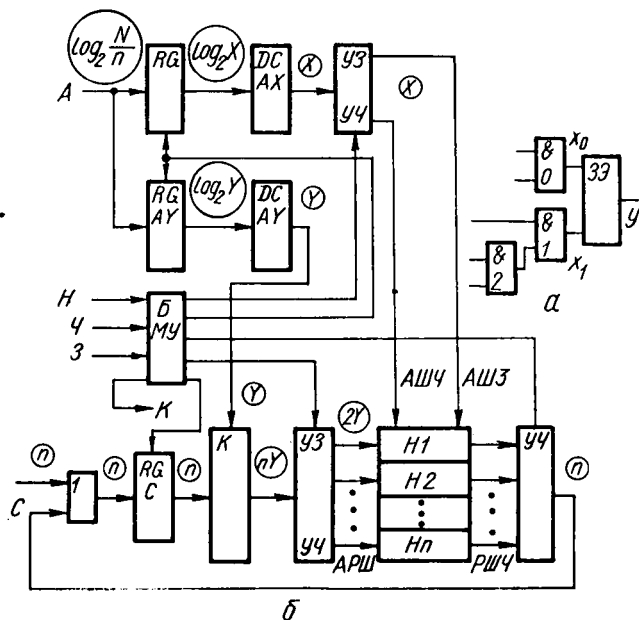


Рис. 4.9

ИЗ объединены в коммутатор K , который в зависимости от содержимого $RGAY$ осуществляет выборку одной АРШ в каждом накопителе. Стоимость ЗУ зависит не только от длины слова n , но и от числа адресных шин X и Y

$$C = C_1 N + an + bnY + bX + b_1 Y + 2g\sqrt{Y} + 2g\sqrt{X} + C_5 + C_6 \log_2 \frac{N}{n}, \quad (4.4)$$

где a — суммарная стоимость одного разряда RG и одного УЧ; b — суммарная стоимость одного усилителя записи УЗ, одного УЧ и одного выхода дешифратора ДСАХ или коммутатора K ; b_1 — стоимость одного выхода последнего каскада ДСВ; g — стоимость одного выхода предпоследнего каскада ДСАХ и ДСАУ, $XY = \frac{N}{n} = l$ — число ячеек в ЗУ. В связи с этим возникает задача такого разделения

адресных шин на группы X и Y , при котором стоимость ЗУ минимальна. С этой целью выражение (4.4) запишем в виде

$$C = \left(C_1 N + an + C_5 + C_6 \log_2 \frac{N}{n} \right) + ((bn + b_1)Y + bX) + 2g(\sqrt{Y} + \sqrt{X}).$$

Первый член в скобках не зависит от Y , второй член принимает наименьшее значение при

$$\frac{Y}{X} = \frac{b}{bn + b_1} \approx \frac{1}{n},$$

а третий минимален при $X = Y$. Следовательно, величина $\frac{Y}{X}$, при которой стоимость ЗУ минимальна, лежит в пределах

$$1 \geq \frac{Y}{X} \geq \frac{1}{n}.$$

Но так как основное влияние на функцию C оказывает второй член, то $X \approx nY$. Поскольку X — число АШ, а nY — число АРШ, то стоимость ЗУ минимальна при равном числе адресных и адресно-разрядных шин. При этом $Y = \frac{\sqrt{N}}{n}$. Следовательно, функцию стоимости при оптимальной конфигурации накопителя можно приближенно представить в виде

$$C = \left(C_1 N + an + C_5 + C_6 \log_2 \frac{N}{n} \right) + 2b\sqrt{N} + 2g\sqrt{N} \left(\frac{1}{\sqrt{n}} + 1 \right).$$

Членами, зависящими от n и оказывающими наибольшее влияние на стоимость ЗУ, являются an и $2g\sqrt{N} \left(\frac{1}{\sqrt{n}} + 1 \right)$, сумма которых принимает наименьшее значение при

$$n_0 = \sqrt[3]{\left(\frac{g}{a} \sqrt[4]{N} \right)^2}.$$

Так как $N \approx 10^5 - 10^7$; $\frac{g}{a} = 0,1 - 0,3$, то $1 \leq n_0 \leq 7$, т. е. стоимость ЗУ с выборкой 2,5Д минимальна при малой разрядности слов и при квадратном накопителе (квадратном в смысле равенства nY и X). ЗУ такого типа использованы, например в ЭВМ ЕС-1020.

4.3. СТРУКТУРНЫЕ СХЕМЫ ЗУ С В-ПОИСКОМ

Особенностями ЗУ с В-поиском является, как правило, наличие непрерывной запоминающей среды (например, магнитной поверхности) и последовательный доступ к отдельным ячейкам. Отдельные ячейки последовательно во времени с периодом T проходят мимо усилителей записи и усилителей чтения. Поэтому как при записи, так и при чтении необходимо фиксировать моменты времени, когда возможно обращение к данной ячейке. Для этого чтение и запись

в ЗУ с В-поиском осуществляют в моменты действия синхронизирующих импульсов (СИ). В зависимости от способа получения СИ различают ЗУ с внешней и внутренней синхронизацией. В ЗУ первого типа имеется специальный генератор, вырабатывающий СИ. В ЗУ с внутренней синхронизацией имеется отдельный усилитель записи, с помощью которого в начале работы ЗУ записываются СИ. В процессе работы ЗУ синхронизация осуществляется от специального усилителя чтения, воспроизводящего СИ. Период T , в течение которого мимо УЗ и УЧ проходят все ячейки ЗУ, разбивается синхронизирующими импульсами на Tf интервалов, где f — частота СИ. Емкость ЗУ равна RTf бит, где R — число пар УЗ—УЧ (число УЗ равно числу УЧ).

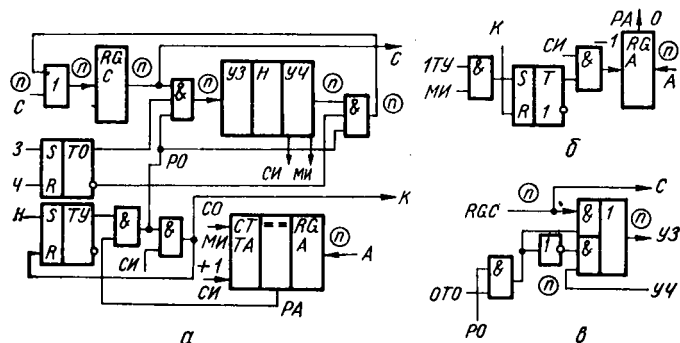


Рис. 4.10

По способам записи информации различают ЗУ с последовательной, параллельной и последовательно-параллельной записью.

В ЗУ с параллельной записью число R равно количеству разрядов в слове, причем все разряды слова записываются либо считываются одновременно (рис. 4.10, а). Чтение в таком ЗУ начинается с подачи адреса в RGA и сигналов H и $Ч$ на триггеры управления (ТУ) и операции (ТО). Каждому СИ соответствует одна ячейка ЗУ, а число СИ подсчитывается счетчиком текущего адреса (СТТА), период которого равен $Tf = 2^m$. Следовательно, в СТТА формируется номер ячейки, находящейся в данный момент времени возле УЗ и УЧ. При совпадении содержимого СТТА и RGA схема равенства кодов (СРК) выдает сигнал равенства адресов PA , по которому формируется сигнал разрешения обращения PO , в результате чего слово из накопителя H через УЧ записывается в RGC . В этот же момент формируется сигнал конца операции K .

Запись отличается от чтения только тем, что в RGC по внешнему каналу заносится подлежащее записи слово и при совпадении содержимого RGA и СТТА открываются УЗ. В некоторых ЗУ, с целью повышения надежности и упрощения адресных цепей, СТТА имеет период больший, чем Tf . Установка в нуль такого счетчика производится специальным маркерным импульсом (МИ), от которого производится начало отсчета адресов. Для получения МИ, так же как и для получения СИ, используется отдельная пара УЗ—УЧ. Изменения в схеме ЗУ при наличии МИ показаны на рис. 4.10, б. В этом

один за другим. Регистр RGC в этом случае условно разбивается на k частей. В таком ЗУ (рис. 4.12) имеются коммутаторы $K1$ и $K2$, поочередно подключающие выходы и входы RGC к УЗ и УЧ. Коммута-

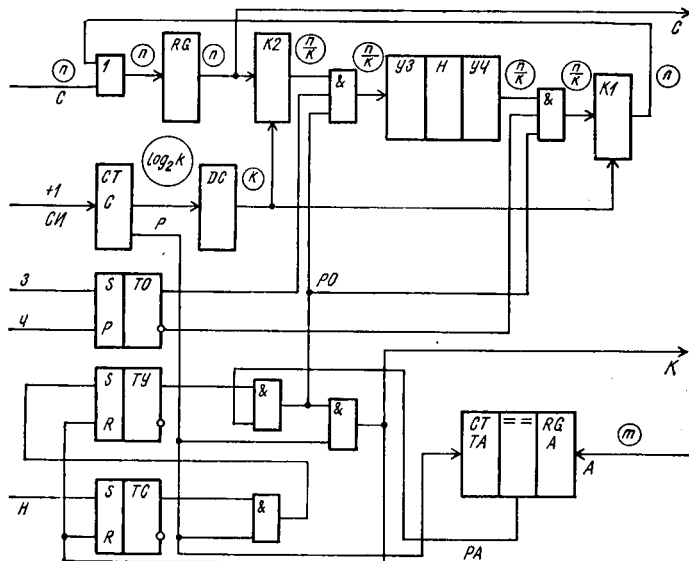


Рис. 4.12

торы $K1$ и $K2$ управляются счетчиком слогов $СТС$ с периодом k посредством дешифратора $ДС$ на k выходов. Триггер $ТС$ служит для привязки сигнала H к моментам действия переноса P со счетчика $СТС$. В остальном работа такого ЗУ аналогична работе схемы на рис. 4.11, а.

В ЗУ с последовательной записью для записи и чтения используется одна пара $УЗ—УЧ$. Слово в ЗУ записывается последовательно разряд за разрядом. Особенностью схемы ЗУ (рис. 4.13) является наличие счетчика разрядов $СТР$ с периодом n и последовательного входа и выхода на RGC . В остальной ЗУ аналогично схеме на рис. 4.12.

Наряду с В-поиском в ЗУ часто используется комбинированный М—В-поиск, при котором часть пар $УЗ—УЧ$ выбирается по М-поиску, а затем используется В-поиск. На рис. 4.14 показана структурная схема ЗУ с комбинирован-

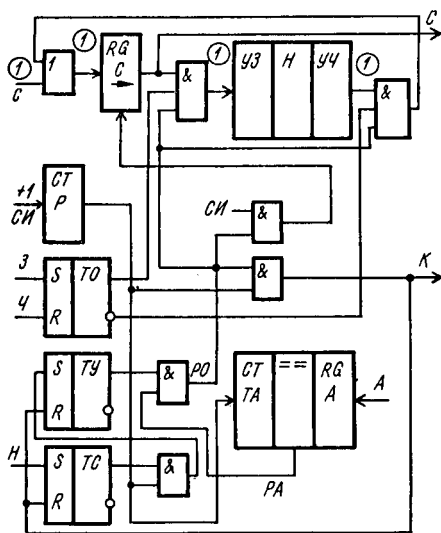


Рис. 4.13

ным М—В-поиском и последовательной записью с четырьмя парами УЗ—УЧ. Регистр адреса здесь условно разбивается на две части $RG A1$ и $RG A2$, первая из которых управляет поиском по В-способу, а вторая — выбирает пару УЗ—УЧ по М-способу с помощью дешифратора ДС.

Рассмотренные структуры не исчерпывают всех возможных вариантов ЗУ с В- и М—В-поиском. ЗУ такого типа могут быть с разрушающим и неразрушающим чтением, с маркерными импульсами и без них, адресные и ассоциативные, с последовательной, параллельной и последовательно-параллельной записью, с возможностью выборки одного слова и массивов. Сочетания этих признаков в конкретных

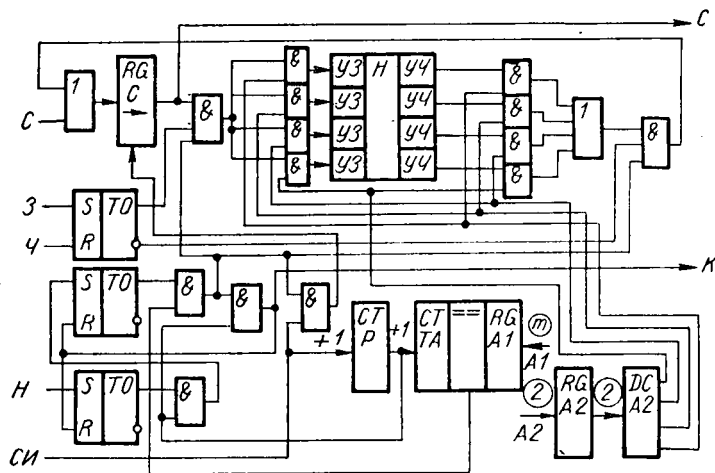


Рис. 4.14

схемах определяются физическими принципами построения накопителей и технико-экономическими требованиями, предъявляемыми к ЗУ.

Наличие непрерывной запоминающей среды в накопителе не всегда соответствует ЗУ с В-поиском, так же как и наличие отдельных ЗЭ не является признаком М-поиска. Например, накопитель на сдвигающих регистрах состоит из отдельных ЗЭ, однако структура такого ЗУ аналогична структуре ЗУ с В-поиском (рис. 4.13). Накопитель ЗУ на электронно-лучевых трубках является непрерывной запоминающей средой, но структура такого ЗУ соответствует ЗУ с М-поиском и выборкой 2Д, разумеется, если отклоняющую систему интерпретировать как дешифратор адреса. Однако приведенные примеры являются скорее исключением, чем правилом.

4.4. НАКОПИТЕЛИ ЗУ НА ФЕРРИТОВЫХ СЕРДЕЧНИКАХ

Одними из наиболее распространенных дискретных компонент, используемых для построения накопителей ЗУ, являются ферритовые сердечники с прямоугольной петлей гистерезиса (ППГ). Мате-

риал, из которого изготавливаются сердечники, при отсутствии внешних магнитных полей может неограниченно долго находиться в одном из двух состояний остаточной индукции $+B_r$ и $-B_r$, соответствующих буквам 0 и 1 двузначного алфавита. Запоминающий элемент представляет собой сердечник с одновитковыми обмотками (шинами), число которых обычно не превышает четырех. Коэффициент прямоугольности петли гистерезиса должен быть близок к 1. Для этого стремятся уменьшать толщину стенок сердечника. Кроме того, уменьшение общих размеров сердечника позволяет уменьшить количество тепла, выделяемого в нем, и улучшить его отвод, так как выделение тепла пропорционально объему сердечника, а отвод — пропорциона-

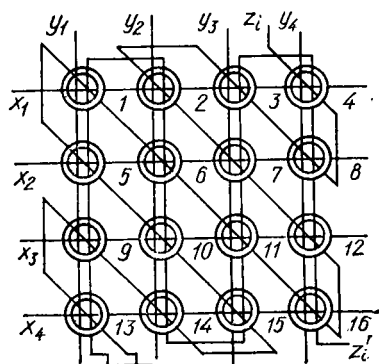


Рис. 4.15

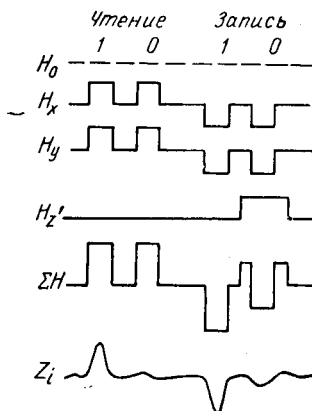


Рис. 4.16

лен общей его поверхности. Все это способствует уменьшению времени перемагничивания сердечника и привело к тому, что в настоящее время используются сердечники с очень малыми размерами (с наружным диаметром 0,3—1, внутренним диаметром 0,06—0,7, высотой 0,06—0,5 мм).

В накопителях ЗУ с выборкой ЗД сердечники, служащие для запоминания одноименных разрядов всех слов, объединяются в плоские матрицы (рис. 4.15). Каждая матрица имеет систему адресных шин X_s и Y_r ($s, r = 1, 2, \dots, \sqrt{\frac{N}{n}}$), а также разрядные шины записи Z_i' и чтения Z_i ($i = 1, 2, \dots, n$). По шинам X_s и Y_r могут действовать сигналы, создающие магнитное поле с напряженностью $H < H_0$ (H_0 — коэрцитивная сила), недостаточной для перемагничивания сердечника. Эти сигналы получили название полуточков, а сердечники, на которые действуют полуточки, называют полувывбранными. Для перемагничивания сердечника необходимо совпадение двух полуточков. Например, для считывания цифры, записанной в сердечнике 6, необходимо подать полуточки на шины X_2 и Y_2 . Разрядная шина чтения (воспроизведения) Z_i прошивается по диагонали с целью уменьшения помех, возникающих от действия полуточков. При такой прошивке одна часть сердечников наводит положительные помехи,

а другая — отрицательные. В результате суммарная помеха значительно уменьшается.

После считывания сердечник будет находиться в состоянии $+B_r$, независимо от того, в каком состоянии он находился раньше, т. е. происходит разрушение информации. Для восстановления прежней или записи новой информации в матрице имеется разрядная шина записи Z'_i , включенная согласно со всеми адресными шинами и проходящая через все сердечники. При записи или восстановлении 1 на выбранные адресные шины, например, на X_2 и Y_2 при обращении к сердечнику 6, подают полутоки, полярность которых противоположна полярности полутоков чтения. При записи 0, кроме этого, в провод Z'_i подают полуток запрета, направление которого совпадает с направлением полутоков чтения (рис. 4.16). В результате при записи 1 состояние сердечника изменится на $-B_r$, а при записи 0 — останется неизменным, т. е. $+B_r$. Таким образом,

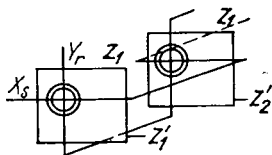


Рис. 4.17

чтение осуществляется путем сброса в 0 ЗЭ, на нулевом входе которого имеется двухвходовая схема совпадения, реализованная по принципу совпадения полутоков, а запись реализуется путем совпадения трех полутоков, т. е. логическая схема данного ЗЭ совпадает со схемой на рис. 4.8, а. Из плоских матриц, служащих для запоминания i -го разряда всех (в данном случае 16) слов, формируется пространственный накопитель — куб (рис. 4.17). Число матриц в кубе равно количеству разрядов в слове. На базе такого накопителя строится ЗУ по схеме на рис. 4.8, б.

Считывание начинается с подачи в регистр адреса (RGA) адреса искомого слова. Адрес разбивается на две части X и Y и декодируется дешифраторами ДСАХ и ДСАУ. Далее на выбранные шины куба X_s и Y_r поступают полутоки положительной полярности. Считанное слово записывается на регистр слов RGC , завершая тем самым такт выборки. После окончания переходных процессов начинается такт восстановления. На выбранные ранее шины X_s и Y_r поступают полутоки отрицательной полярности, а на разрядные шины записи тех матриц, с которых был считан 0 — полутоки положительной полярности. В результате считанное слово запишется в прежнюю ячейку.

Запись отличается от чтения только тем, что усилители чтения в такте выборки блокируются и считанное с заданной ячейки слово теряется, а на его место записывается новое слово, ранее занесенное в регистр слова. Временная диаграмма работы ЗУ при чтении показана на рис. 4.18, а, где t_1 — время, необходимое для приема слова на регистр (приема адреса ПА и слова ПС); t_2 — задержка в ДСА; t_3 — задержка в усилителе чтения или записи; t_4 — время перекрытия адресных полутоков током запрета; τ — время, необходимое для перемангничивания сердечника. Следовательно,

$$t_{0ч} = 2\tau + t_1 + t_2 + 2t_4. \quad (4.5)$$

Временная диаграмма работы ЗУ при записи отличается только тем, что отсутствует сигнал ПС, а прием слова на RGC происходит одновременно с сигналом ПА. Указанные последовательности управляю-

щих сигналов вырабатывает блок местного управления (БМУ) ЗУ. В качестве примера на рис. 4.18, б показана схема БМУ, построенная на линиях задержки (ЛЗ), входными сигналами БМУ являются сигналы Н, Ч (чтение) и З (запись). В случае необходимости на выходы БМУ подключают формирователи стандартных сигналов, преобразующих импульсы с ЛЗ в сигналы необходимой длительности и амплитуды.

Не представляет особых трудностей построить трехмерное ЗУ с циклом t_0 в несколько микросекунд. Однако дальнейшее повышение быстродействия требует применения специальных мер. Так как $\tau \gg t_1, t_2, t_3, t_4$, то прежде всего стремятся уменьшить τ . Известно, что величина τ обратно пропорциональна разности $H_1 - H_0$, где H_1 — напряженность магнитного поля, которую создает перемагничивающий ток. Поэтому для уменьшения τ увеличивают полутoki, создающие напряженность H_1 . Однако при этом должно выполняться условие $H_1 < 2H_0$ и, кроме того, с увеличением амплитуды полутoka возрастают помехи от полувыбранных сердечников. Дальнейшее повышение H_1 возможно за счет применения тока смещения, протекающего по специальной шине. Это позволяет увеличить верхнюю границу для H_1 в 1,5 раза, т. е. $H_1 < 3H_0$. Использование рассмотренных методов, применение миниатюрных сердечников и быстродействующего «электронного обрaмления» (т. е., ДСА, УЧ, УЗ, RGC, RGA) позволяет создавать трехмерные ЗУ с циклом до 0,6 мкс. Так как каждый сердечник пронизывается четырьмя шинами, то ЗУ такого типа получили название (ЗД, 4W).

Важнейшим показателем качества накопителей типа ЗД на ферритовых сердечниках является их помехоустойчивость. Если бы не было специальной диагональной прошивки и полярность помех от полувыбранных сердечников была одинаковой, то суммарная помеха намного бы превышала сигнал от выбранного сердечника. Помимо диагональной прошивки уменьшения амплитуды помехи добиваются также путем стробирования, т. е. открывания усилителей чтения в момент, когда действие помехи практически окончилось. Это возможно благодаря тому, что пиковые значения амплитуд помехи и сигнала не совпадают во времени.

Уменьшения амплитуды помехи в 2, 4, 8 и т. д. раз добиваются также путем разбиения (секционирования) разрядной шины чтения Z_i на две, четыре (EC-1020) и т. д. частей, каждая из которых охватывает соответствующую часть сердечников разрядной матрицы.

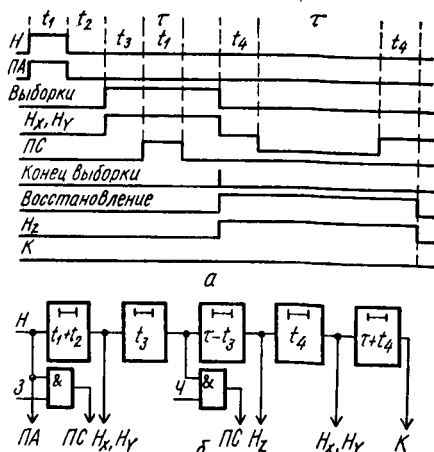


Рис. 4.18

Дальнейшее повышение отношения сигнал — помеха γ достигается за счет того, что полуток в шине Y_r подается после затухания помех от действия полутoka X_s (рис. 4.19). При этом целесообразно изготовлять матрицы прямоугольной формы, например, 128×32 вместо 64×64 , располагая больше сердечников на шине X_s . В сочетании со стробированием и секционированием, такой метод позволяет даже отказаться от диагональной прошивки считывающей шины

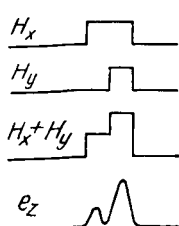


Рис. 4.19

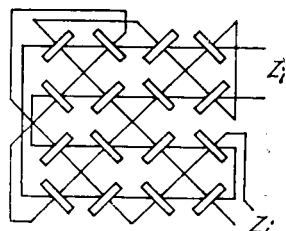
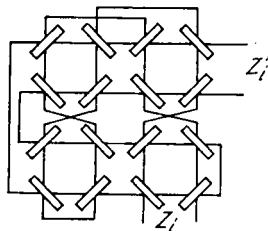


Рис. 4.20

и в качестве считывающей использовать шину записи Z'_i . Устранение четвертого провода упрощает конструкцию и технологию изготовления ЗУ, которое получило название ЗУ типа (ЗД, ЗW), однако при этом увеличивается время обращения к ЗУ.

Прошивка сердечников трехмерного накопителя разрядными шинами, показанная на рис. 4.15, не является единственно возможной.

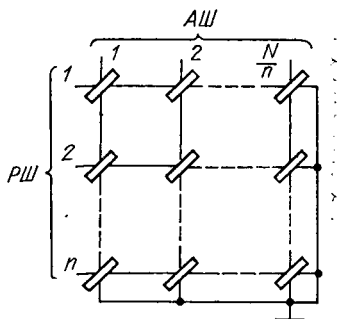


Рис. 4.21

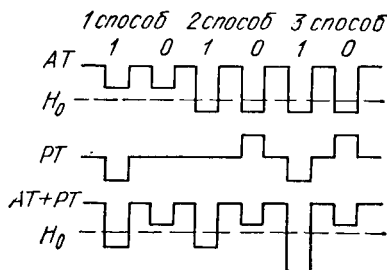


Рис. 4.22

С целью уменьшения длины разрядных проводников применяют также другие способы прошивки (рис. 4.20).

Схема накопителя ЗУ с выборкой 2Д показана на рис. 4.21. Через каждый сердечник проходят две шины: адресная (АШ) и разрядная (РШ), с помощью которых осуществляется как запись, так и чтение. Число АШ равно числу ячеек ЗУ, а количество РШ — длине ячейки. Чтение осуществляется полным (точнее неограниченным по амплитуде) током, действующим по одной из АШ, а запись — по совпадению полутокa, действующим по АШ и РШ, т. е. логическая структура запоминающего элемента совпадает со схемой на рис. 4.3, а схема всего ЗУ — со схемой на рис. 4.5. Если оптимальная длина n_0 ячейки больше длины машинного слова n_c , то ЗУ строят по схемам на

рис. 4.6 или 4. 7. При этом схема накопителя и принцип его работы не изменяются.

Чтение слова начинается с подачи адреса в *RGA* (рис. 4.5). Адрес дешифруется, в результате чего на выбранную АШ (рис. 4.21) поступает полный ток положительной полярности. Все сердечники переходят в состояние $+B_r$, а считанное слово записывается в *RGC*. После этого происходит восстановление информации. Известно три способа восстановления (рис. 4.22). При первом способе в выбранную адресную шину и разрядные шины тех разрядов, в которых необходимо

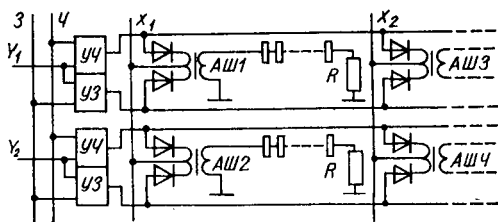


Рис. 4.23

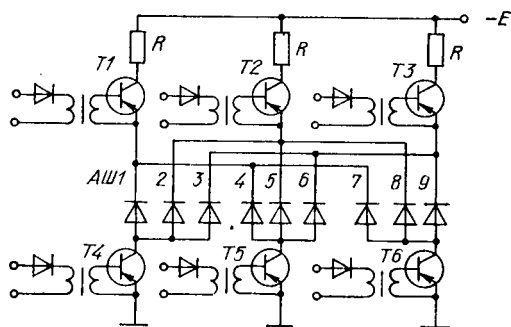


Рис. 4.24

записать 1, поступают отрицательные полутoki. При втором способе полный отрицательный адресный ток (АТ) при восстановлении 0 перекрывается положительным полутокom. При третьем способе полный адресный ток перекрывается положительным или отрицательным разрядным полутокom (РТ). Преимуществом первого способа являются малые амплитуды тока записи. Это уменьшает рассеиваемую мощность и облегчает режим работы усилителей записи, что особенно важно при изготовлении их на основе интегральной технологии. Третий способ обеспечивает наиболее быстрое переключение сердечников. Однако при этом расходуется больше энергии и происходит разрушение как 0, так и 1 под действием полутокon на невыбранные сердечники.

Запись в ЗУ типа 2Д отличается от чтения только тем, что в такте выборки разрядные усилители чтения блокируются и считанное с заданной ячейки слово теряется, а на его место в такте восстановления записывается новое слово, предварительно занесенное в *RGC*.

Значительная часть оборудования ЗУ типа (2Д, 2W) расходуется на адресные цепи. Наибольшее распространение получили адресные цепи на трансформаторах и транзисторах. При этом можно использовать сердечники с линейной магнитной характеристикой (ЛМХ) и диодно-трансформаторные вентили (ДТВ) (рис. 4.23). Токи чтения и записи вырабатываются адресными усилителями чтения (УЧ) и записи (УЗ) и с помощью ДТВ передаются в адресные шины. Набор ДТВ выполняет также роль последнего каскада дешифратора адреса и управляется предыдущими каскадами ДСАХ и ДСАУ.

Стоимость адресных цепей можно оценить как

$$C = C_1 \frac{N}{n} + C_2 m_X + (C_2 + C_3) m_Y,$$

где m_X и m_Y — количество выходов ДСАХ и ДСАУ; $\frac{N}{n} = m_X m_Y$ — число ячеек в ЗУ; C_1 — стоимость трансформатора и двух диодов; C_2 — стоимость ДСАХ или ДСАУ, приведенная к одному выходу; C_3 — суммарная стоимость одного АУЧ и АУЗ. Функция C будет минимальна при

$$m_X = \sqrt{\frac{N(C_2 + C_3)}{nC_2}}; \quad m_Y = \sqrt{\frac{NC_2}{n(C_2 + C_3)}}.$$

Наряду с ДТВ в адресных цепях широко применяются также транзисторные ключи, включенные по принципу токовых многополюсников (рис. 4.24). Для выбора, например, восьмой АШ необходимо подать открывающий сигнал на базу T_2 и T_6 . В свою очередь ДТВ, соединенные с базами транзисторов, могут быть использованы для построения предпоследних каскадов ДСА. По подобной схеме построены адресные цепи в машине ЕС-1020.

Сопروتивление протекающему по обмотке сердечника току будет различно при считывании «1» и «0». Поэтому сопротивление АШ будет максимальным при считывании слова 11...1 и минимальным при считывании слова 00...0. Переменное сопротивление адресной шины может привести к тому, что амплитуда сигнала «1» при чтении слова с большим числом 1 может быть равна амплитуде сигнала «0» при чтении слова с большим числом 0. Поэтому возникает задача стабилизации тока в АШ. К настоящему времени наибольшее распространение получили два способа стабилизации: с помощью балластного сопротивления (сопротивление R на рис. 4.23 и 4.24) и за счет использования двух сердечников на разряд. Недостатком первого способа является большая мощность, рассеиваемая на балластном сопротивлении, а второго — значительное усложнение ЗУ.

При использовании второго способа для запоминания одного разряда слова используется два сердечника (рис. 4.25, а), один из которых (например, A) в исходном состоянии (после считывания) намагничен до уровня $+B_r$, а второй (B) — до $-B_r$ (рис. 4.25, б). Запись осуществляется подачей отрицательного полутoka i_{3a} в АШ и отрицательного (положительного при записи 0) полутoka i_{3p} в РШ. При записи 1 магнитные потоки, создаваемые полутokaми, в сердечнике A суммируются, а в сердечнике B — вычитаются. В результате этого сердечник A перейдет в состояние $-B_r$, а сердечник B не изменит своего состояния (рис. 4.25, б). При записи 0 изменит свое состояние только сердечник B . Таким образом, после записи сердечники A и B будут находиться в одинаковых состояниях ($+B_r$ или $-B_r$) (рис. 4.25, б). Считывание производится подачей полного тока i_4 в АШ. Ток i_4 создает в сердечнике A положительное поле, а в B — отрицательное. В результате только один из сердечников перемагнитится и в выходной обмотке появится импульс (положительный при чтении 1, отрицательный при чтении 0), равный разности полезного импульса

и импульса помехи (рис. 4.25, в). Вследствие фазовой (знаковой) селекции 0 и 1 помехоустойчивость ЗУ такого типа повышается.

Для создания сверхбыстродействующих ЗУ на сердечниках повышенной помехоустойчивости иногда используют накопитель, в котором запись и чтение осуществляется полными токами (рис. 4.26). Отличие его от накопителя, показанного на рис. 4.21, состоит в том, что схема совпадения на входе X_1 ЗЭ (рис. 4.3, в) реализована по принципу ДТВ, а не по принципу совпадения токов. Логическая же структура ЗЭ и, следовательно, структурная схема ЗУ остается прежней.

Использование считывающих токов большой амплитуды и частичного перемагничивания ЗЭ с двумя сердечниками на разряд позволило

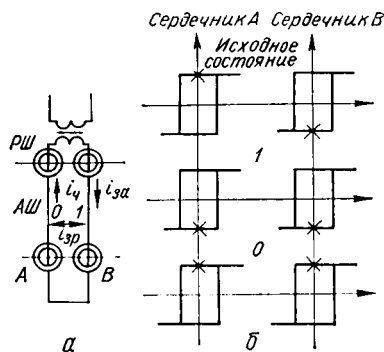


рис. 4.25

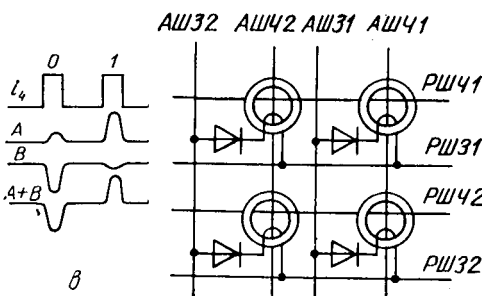


рис. 4.26

добиться рекордного для ЗУ на сердечниках цикла обращения, равного 110 нс при $N = 589824$ бит и $t_b = 65$ нс.

Для выяснения особенностей построения накопителей в ЗУ типа 2,5Д проследим, как можно преобразовать систему выборки ЗД в систему 2,5Д. Накопитель ЗУ типа ЗД состоит из ряда разрядных матриц с общими АШХ и АШУ (рис. 4.27, а). В каждой матрице при обращении выбирается один сердечник, находящийся на пересечении выбранных АШХ и АШУ, по которым протекают полутoki положительной полярности. Запись осуществляется подачей полутোকoв отрицательной полярности на АШХ и АШУ и полутока запрета положительной полярности на матрицы тех разрядов, в которые необходимо записать 0. В накопителе ЗУ типа 2,5Д АШУ выполняются отдельно для каждой разрядной матрицы (рис. 4.27, б), т. е. являются адресно-разрядными шинами (АРШ). Пусть, например, $N = 36$, $n = 3$. Тогда из условия равенства числа адресных и адресно-разрядных шин имеем $X = 6$, $Y = 2$, а схема прошивки шин в таком накопителе показана на рис. 4.28. Элементы И на этой схеме соответствуют коммутатору К на рис. 4.9. Чтение осуществляется подачей положительных полутোকoв по одной АШХ и по n АРШУ ($Y_{j1} - Y_{jn}$; $j = 1, 2, \dots, \sqrt{N}$; n — длина слова). Воспроизводится сигнал с помощью РШЧ с такой диагональной прошивкой, как и у разрядных матриц ЗУ типа ЗД. Запись осуществляется пропуском отрицательных полутোকoв

по АШХ (общей для всех разрядов) и по АРШУ в матрицах тех разрядов, в которых необходимо записать 1. Необходимость в специальном токе запрета отпадает. Через каждый сердечник проходят три шины (АШХ, АРШУ, РШЧ). Запись и чтение осуществляется по совпадению полутоков, запись — как в системе 2Д, чтение — как в системе 3Д. Логическая структура 3Э совпадает со схемой на рис. 4.9, а. Структурная схема всего 3У — со схемой на рис. 4.9, б. Таким образом, основными преимуществами системы (2,5Д, 3W) по сравнению с системой (3Д, 4W) являются упрощение накопителя за счет устранения РШЗ (шин запрета) и повышение быстродействия за счет отсутствия полутока запрета, который по длительности в системе (3Д, 4W) должен перекрывать адресные полутоки записи. По сравнению же с системой (2Д, 2W) преимущест-

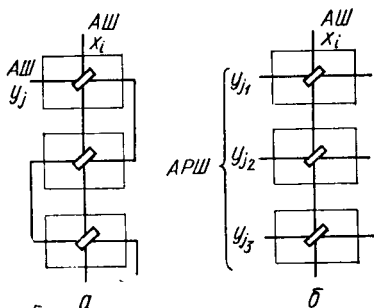


Рис. 4.27

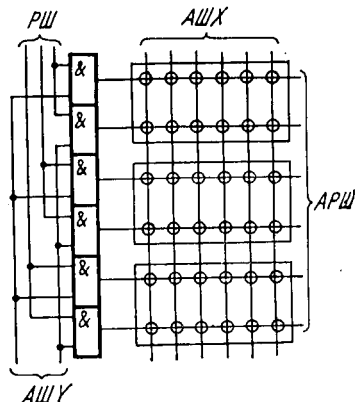


Рис. 4.28

вом является уменьшение стоимости адресных цепей. Для 3У с циклом порядка 0,5 мкс устранение шины запрета имеет большое значение, так как в таких 3У применяются малогабаритные сердечники (например, $0,3 \times 0,18 \times 0,06$ мм) с плотностью компоновки до 150 штук на см² и более. В накопителе на таких сердечниках прошивка четвертой шины привела бы к значительному увеличению его стоимости.

Так как в 3У типа 2,5Д АРШ являются независимыми для каждого разряда, то их можно использовать в качестве РШ чтения и сократить число шин, пронизывающих сердечник, до двух. Быстродействие 3У такого типа (2,5Д, 2W) значительно ниже, чем 3У типа (2,5Д, 3W), так как момент подачи полутока по АШХ, определяющего перемагничивание выбранных сердечников, должен быть задержан относительно переднего фронта полутоков У на время затухания переходных процессов в шинах У. Выходной сигнал снимается с РШЧ, в качестве которых выступают АРШУ. Для 3У такого типа характерна сложная система усилителей чтения и записи и простой накопитель. Так как в 3У большой емкости стоимость накопителя составляет 60...70 % от стоимости всего 3У, то уменьшение стоимости накопителя оказывается преобладающим. Поэтому система (2,5Д, 2W) применяется при создании 3У большой емкости с $t_0 \approx 2...8$ мкс.

Рассмотренные принципы построения и опыт практической реализации накопителей ЗУ на ферритовых сердечниках с различными системами выборки, такими как (ЗД, 4W), (ЗД, 3W), (2,5Д, 3W), (2,5Д, 2W) и (2Д, 2W) позволяют сделать следующие выводы.

1. В порядке возрастания быстродействия при фиксированной емкости, типы ЗУ образуют следующий ряд: (2,5Д, 2W), (ЗД, 3W), (ЗД, 4W), (2,5Д, 3W), (2Д, 2W). Наименьшее быстродействие систем (2,5Д, 2W) и (ЗД, 3W) объясняется тем, что в них полуточки выборки разнесены во времени. В ЗУ типа (2,5Д, 3W) большее быстродействие, чем в ЗУ типа (ЗД, 4W), достигается за счет отсутствия токов запрета и укорочения в 3—4 раза шин выборки. Наибольшее быстродействие в ЗУ типа (2Д, 2W) достигается форсированным переключением сердечников токами большой амплитуды. Наилучшие достигнутые значения циклов обращения для соответствующих систем приведенного ряда следующие: 2 мкс; 1 мкс; 0,6 мкс; 375 нс; 110 нс.

2. При данной емкости сложность (стоимость) накопителя возрастает в соответствии со следующим рядом (2,5Д, 2W), (2Д, 2W), (2,5Д, 3W), (ЗД, 3W), (ЗД, 4W). По сложности «электронного обрамления», без учета сложности накопителя, ЗУ образуют ряд (в порядке возрастания): (ЗД, 4W), (ЗД, 3W), (2,5Д, 3W), (2,5Д, 2W), (2Д, 2W).

3. В ЗУ типа 2Д отношение γ может быть значительно выше, чем в ЗУ типа ЗД и 2,5Д. При равных отношениях γ требования к таким показателям, как диапазон температур, стабильность импульсов тока по амплитуде и по длительности, коэффициент прямоугольности петли гистерезиса, однородность материала сердечников и т. д. в ЗУ типа 2Д менее жесткие, чем в ЗУ других типов.

4. В ЗУ типа 2Д и 2,5Д накопитель имеет плоскую конструкцию, что облегчает его ремонт и улучшает теплообмен.

Следует учитывать, что в процессе совершенствования технологии соотношение стоимостей накопителя и электронного обрамления могут изменяться и, следовательно, могут меняться и приведенные выше выводы.

Несмотря на то, что ЗУ на сердечниках в ЭВМ второго и третьего поколений занимали доминирующее положение, все время проводились попытки заменить ЗЭ на сердечниках более совершенным ЗЭ. Это объясняется двумя существенными недостатками ЗУ на сердечниках, а именно: считыванием с разрушением и большой трудоемкостью изготовления накопителя.

Было предложено большое число различных типов ЗЭ со считыванием без разрушения (биакс, трансфлюксор и др.). Однако практически все предложенные ЗЭ либо имели низкое быстродействие, либо приводили к значительному усложнению накопителя.

Более значительные результаты были достигнуты в снижении трудоемкости изготовления накопителей за счет применения ЗУ на монолитных ферритовых платах с ППГ (рис. 4.29). Одна плата служит для запоминания одного разряда всех слов. Каждые два отверстия образуют один ЗЭ по аналогии с ЗУ типа (2Д, 2W) с двумя сердечниками на разряд. Это делается для повышения отношения γ , так как магнитная плата характеризуется большей неоднородностью материала по сравнению с сердечником. Разрядные платы укладываются одна на другую, образуя пространственный накопитель. Количество плат равно длине слова. После сборки накопителя в отверстия пропускают АШ. Структурная схема ЗУ с таким накопителем соответствует рис. 4.5. Выборка типа 2Д обуславливает большую сложность адресных цепей ЗУ. Вследствие этого ЗУ на ферритовых платах непригодно для создания ЗУ большой емкости (10^8 бит и более).

Для создания ЗУ большой емкости был предложен другой интегральный метод изготовления накопителя, основанный на применении печатных сердечников, получаемых травлением пермалоевой фольги, нанесенной на подложку. С помощью печатного монтажа наносится также система адресных и разрядных шин. Разрядные платы собираются в накопитель типа ЗД. Запись выполняют обычным способом по совпадению трех полутоков. Чтение же осуществляется с помощью высокочастотных радиопульсов, обеспечивающих высокую помехоустойчивость и неразрушающее считывание.

Еще одно интересное направление в создании накопителей невысокой стоимости состоит в таком их выполнении, когда адресные и разрядные шины помещены в ферритовый монолит. В одной из конструкций накопителя использованы матрицы в виде слоистых ферри-

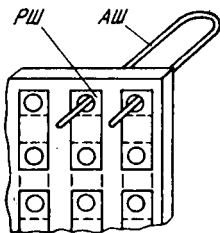


Рис. 4.29

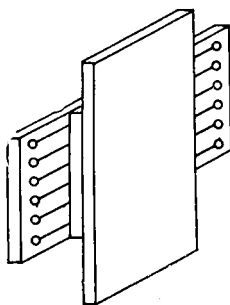


Рис. 4.30

товых пластин и палладиевые проводники (рис. 4.30). Здесь палладиевые шины наносятся химическим способом на поверхность двух тонких ферритовых пластин. Эти пластины складываются «крест-накрест» проводниками внутрь и разделяются третьей тонкой пластиной без проводников. В полученной трехслойной матрице цифрами 0 и 1 соответствуют состояния намаг-

нитненности феррита в перекрестьях проводников.

В заключение отметим, что в последнее время неоднократно делались предсказания о замене ферритовых сердечников другими запоминающими элементами. Однако технология производства сердечников хорошо отработана, на ее основе организовано массовое производство, а качество сердечников все время улучшается. Поэтому технологическим методам изготовления новых запоминающих элементов трудно конкурировать с технологией производства ферритов. В настоящее время почти все выпускающиеся серийно цифровые ЭВМ имеют в своем составе ЗУ на сердечниках. Однако время повсеместного применения ЗУ на сердечниках закончилось благодаря разработке и совершенствованию ЗУ на интегральных полупроводниковых элементах.

4.5. НАКОПИТЕЛИ ЗУ НА ТОНКИХ МАГНИТНЫХ ПЛЕНКАХ

Тонкими магнитными пленками (ТМП) называют слой магнитного вещества ($\text{Fe} \sim 19\%$, $\text{Ni} \sim 81\%$), толщина которого соизмерима с размерами доменов в монокристаллах (0,02—1,0 мкм). Различают плоские ТМП и цилиндрические. Последние наносятся на подложку, выполненную в виде провода. Один из способов получения ТМП состоит в испарении металлов в вакууме и осаждении их паров на подложку через маску с отверстиями прямоугольной формы.

Другой способ получения ТМП состоит в создании сплошной пленки с последующим травлением. Для создания ЗЭ используют анизотропные свойства ТМП. Анизотропность ТМП проявляется в наличии осей легкого и трудного перемагничивания, благодаря чему вектор намагниченности ТМП при отсутствии внешнего поля ориентируется вдоль оси легкого намагничивания. Одному направлению вектора намагниченности соответствует цифра 1, другому — цифра 0. Анизотропность ТМП получают, например, путем напыления пленок в присутствии внешнего магнитного поля, направленного вдоль подложки.

В качестве примера рассмотрим ЗУ на плоских ТМП, конструкция ЗЭ которого показана на рис. 4.31, а. Сплошная ТМП наносится

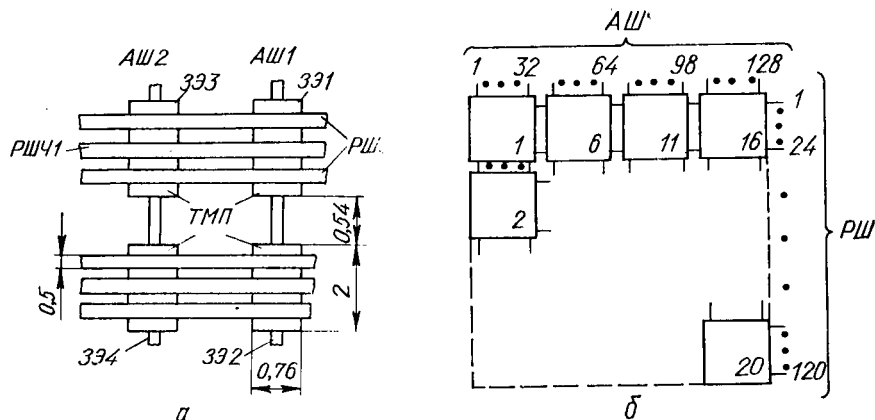


Рис. 4.31

на стеклянную подложку с размерами $70 \times 43 \times 0,2$ мм. Затем путем травления получают 768 «магнитных пятен» (32×24). Двадцать (5×4) подложек собираются в одну матрицу емкостью 128 120-разрядных ячеек (рис. 4.31, б). С одной стороны матрицы наклеиваются 128 АШ, а с другой — 120×2 РШЗ и 120 РШЧ. С целью создания более однородного поля РШЗ разделены на две шины.

Векторы намагниченности, соответствующие цифрам 0 или 1, в исходном состоянии направлены вдоль длинной стороны магнитного пятна, т. е. параллельны оси легкого намагничивания (ЛО), но имеют противоположную ориентацию (рис. 4.32). Характеристика $B(H)$ магнитного пятна при действии внешнего поля вдоль оси ЛО имеет вид обычной ППГ, характеристика же $B(H)$ вдоль оси трудного намагничивания (ТО) близка к линейной. При чтении на выбранную АШ поступает импульс тока i_a , который создает магнитное поле напряженностью H_a , параллельное оси трудного перемагничивания. Вектор намагниченности поворачивается в направлении трудной оси, вследствие чего магнитный поток, охватывающий РШЧ, изменится от $\pm \Phi_0$ до нуля. При этом в РШЧ наводится э.д.с., по направлению которой определяют считанную цифру (0 или 1).

Для записи или восстановления информации по РШЗ подают ток i_p положительной или отрицательной полярности (в зависимости от

записываемой цифры) в момент, когда вектор намагниченности пленки еще удерживается полем адресного тока в направлении трудной оси. После выключения адресного тока вектор намагниченности поворачивается на 90° и устанавливается в состояние 0 или 1. Таким образом, чтение осуществляется одним (адресным) током, а запись — двумя (адресным и разрядным) токами, что соответствует выборке типа 2Д.

При изготовлении ЗУ на цилиндрических ТМП подложкой для электролитического осаждения или напыления пленок обычно служит провод из бериллиевой бронзы диаметром $0,05 \div 0,25$ мм. Напыление пленки толщиной порядка 1 мкм производится при круговом магнитном поле, создаваемом током, протекающим по подложке, что

и обуславливает анизотропность свойств пленки. В результате ось легкого намагничивания совпадает с касательной к силовым линиям поля. В качестве РШ записи и чтения используется подложка. Каждое пересечение АШ с подложкой образует ЗЭ (рис. 4.33). Считывание

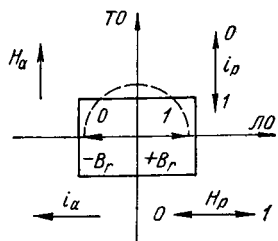


Рис. 4.32

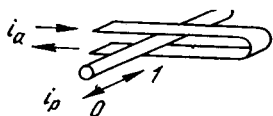
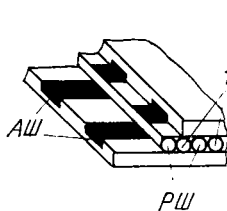
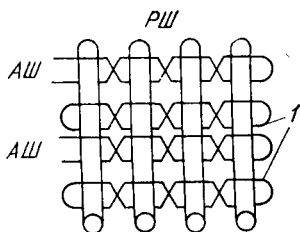


Рис. 4.33



а



б

Рис. 4.34

без разрушения информации производится путем подачи тока i_a на выбранную АШ. При этом вектор намагниченности поворачивается на угол $\alpha < 90^\circ$ к легкой оси и изменяется магнитный поток, сцепленный с РШЧ. Знак изменения потока, а следовательно, и знак наводимой э.д.с., зависит от цифры, хранящейся в ЗЭ. При снятии адресного тока вектор намагниченности возвращается в исходное состояние. Запись осуществляется совпадением токов. Когда вектор намагниченности повернут полем адресного потока, по РШ подается импульс тока i_p той или иной полярности. Этот ток создает круговое поле, которое ориентирует вектор намагниченности в нужном направлении. Величина тока должна быть такой, чтобы выбранный элемент переключался, а невыбранный остался в прежнем состоянии. Таким образом, логическая структура ЗЭ совпадает со схемой на рис. 4.3, в, а схема ЗУ — со схемой на рис. 4.7.

Накопитель ЗУ выполняется либо в виде системы адресных и разрядных шин, залитых компаундом (рис. 4.34, а), либо изготавливается методами ткацкого производства (рис. 4.34, б). На рис. 4.34 цифрой 1 обозначены соответственно разделительное отверстие и разделительная обмотка, предназначенные для создания заданного расстояния между шинами накопителя. Важным положительным качеством ЗУ на цилиндрических ТМП является замкнутость магнитного потока

здесь вдоль оси легкого намагничивания, что обуславливает их меньшую чувствительность к действию внешних полей по сравнению с плоскими ГМП, позволяет строить надежные накопители без их экранирования. Зная настоящее время на цилиндрических ТМП строят быстродействующие ЗУ небольшой емкости ($t_0 = 0,2-0,5$ мкс, $N \leq 10^5$ бит), либо же ЗУ среднего быстродействия ($t_0 = 1-2$ мкс) емкостью до нескольких миллионов бит.

4.6. ПОЛУПРОВОДНИКОВЫЕ НАКОПИТЕЛИ ЗУ

Время переключения современных интегральных логических элементов на переключателях тока достигает $1-2$ нс. Для эффективного использования таких элементов в ЭВМ необходимы сверхбыстродей-

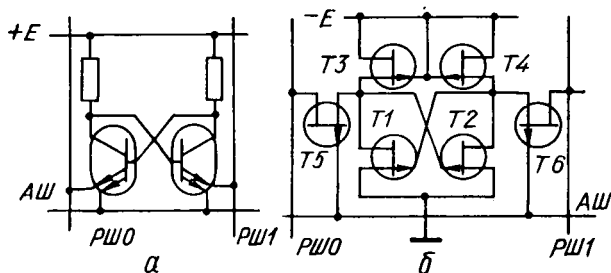


Рис. 4.35

ствующие ЗУ с t_0 порядка 10 нс. Обеспечить такое быстродействие ЗУ могут накопители, в которых запоминающие элементы представляют собой транзисторные триггеры.

С точки зрения технологии изготовления интегральных схем накопители ЗУ как с В-, так и с М-поиском представляют идеальную однородную структуру, пригодную для реализации в виде больших интегральных схем с высокой степенью интеграции. В качестве основных компонент в таких накопителях используют как биполярные, так и униполярные транзисторы.

На биполярных транзисторах накопители строят, как правило, с М-поиском. Простейшие ЗУ могут быть выполнены на многоэмиттерных транзисторах в виде триггеров с непосредственными связями (рис. 4.35, а). В режиме хранения на АШ подается потенциал $-U$, а на РШ0 и РШ1 — нулевой потенциал. Вследствие этого эмиттерный ток открытых транзисторов течет в АШ. Чтение без разрушения осуществляется подачей потенциала $+U$ на выбранную АШ. Потенциал РШ выбранных ЗУ становится ниже потенциала АШ и эмиттерный ток в зависимости от состояния ЗУ отводится в РШ0 или РШ1, что и обнаруживается усилителем чтения.

При записи на выбранную АШ поступает потенциал $+U$, а на РШ в зависимости от записываемой цифры — $+U$ или $-U$. Логическая структура ЗУ отличается от схемы на рис. 4.3 только наличием двух РШ (РШ0 и РШ1) вместо одной, что соответствует выборке 2Д. Простота описанного ЗУ позволяет получить высокую плотность компоновки, достичь большой степени интеграции. На основе аналогич-

ного ЗЭ с трехэмиттерными транзисторами может быть построен на накопитель с выборкой ЗД. Ток в одной из РШ при чтении возникнет только в том случае, если АШХ и АШУ будут выбранными.

Однако накопители на таких ЗЭ не позволяют осуществлять обращение к ЗУ по нескольким независимым каналам. В то же время наличие такой возможности в несколько раз повышает быстродействие ЗУ. Реализовать такую возможность удастся на базе регистров со встроенной операцией выдачи слов по нескольким каналам. Схема накопителя с двумя каналами для чтения и одним для записи показана на рис. 4.36. При чтении слова по первому (второму) каналу возбуждается выбранная АШЧ1 (АШЧ2) и на соответствующий регистр

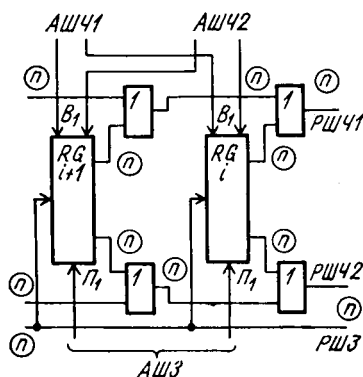


Рис. 4.36

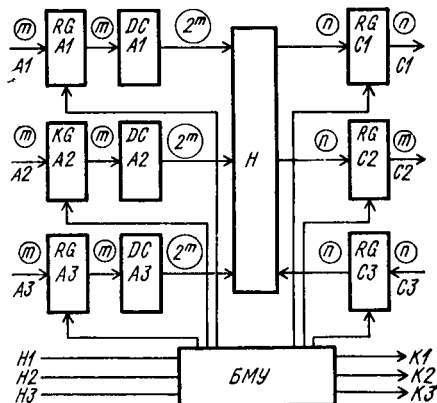


Рис. 4.37

поступает сигнал выдачи слова B_1 . Считанное с регистра RG слово поступает по РШЧ1 (РШЧ2) на выход накопителя. При записи возбуждается выбранная АШЗ, на соответствующий регистр поступает сигнал приема слова Π_1 , а само слово с РШЗ записывается в выбранный регистр. Структурная схема всего ЗУ отличается от ЗУ типа 2Д наличием трех независимых каналов обращения (рис. 4.37). Особенностью этой схемы является наличие трех регистров адреса $RGA1, RGA2, RGA3$ и трех регистров слов $RGC1, RGC2, RGC3$. Регистры $RGA1, RGA2$ и $RGC1, RGC2$ используются при чтении, а регистры $RGA3$ и $RGC3$ — при записи. Благодаря наличию трех RGA и трех RGC возможно одновременное чтение двух слов по независимым адресам $A1$ и $A2$ и одновременная с чтением запись слова по адресу $A3$. Вследствие этого эффективное быстродействие ЗУ в 2–3 раза выше фактического. Одно из ЗУ такого типа на биполярных транзисторах характеризуется следующими данными: $N = 64 \times 8$ бит, $t_{0ч} = 17$ нс, $t_{0з} = 10$ нс.

Кроме биполярных транзисторов, при построении ЗУ широко применяются и МДП-транзисторы. Простейший ЗЭ на МДП-транзисторах показан на рис. 4.35, б. Транзисторы $T5$ и $T6$ выполняют функции элементов И и работают в режиме двусторонней проводимости. При чтении возбуждается АШ, открывая $T5$ и $T6$. Запись осуществ-

ляется по совпадению сигналов на АШ и РШ, т. е. система выборки 2Д.

Накопители с выборкой 3Д могут быть построены на ЗЭ, показанных на рис. 4.38. Основу их составляет триггер T , выходы которого подключаются к РШ с помощью схем совпадений на транзисторах, работающих в режиме двусторонней проводимости.

Важнейшей проблемой, возникающей при построении полупроводниковых накопителей ЗУ, является снижение потребляемой мощности

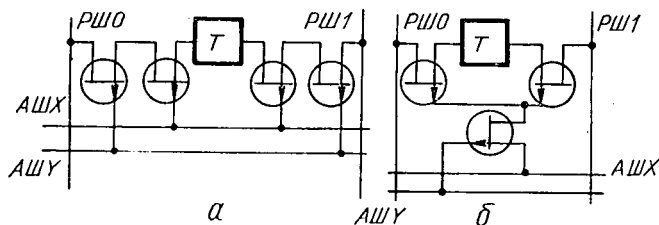


Рис. 4.38

сти. Наиболее простой способ снижения потребляемой мощности состоит в изменении величины сопротивления нагрузки. В схемах на биполярных транзисторах это можно осуществить подключением низкоомного резистора, шунтирующего в режиме обращения к данному ЗЭ его высокоомный коллекторный резистор. Шунтирующий резистор подключается с помощью эмиттерного повторителя, управляемого АШ, вследствие чего мощность, потребляемая ЗЭ в статическом режиме, уменьшается на порядок. В запоминающих элементах на униполярных транзисторах для изменения величины сопротивления нагрузки можно использовать транзисторы $T3$ и $T4$ (рис. 4.35, б), подавая на их затворы импульсы с малой скважностью. В промежутках между импульсами состояния триггеров запоминаются на емкостях затворов $T1$ и $T2$. Рассеиваемая мощность в этом случае может быть доведена до нескольких мкВт/бит.

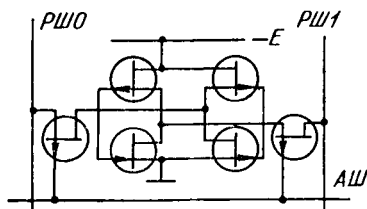


Рис. 4.39

Снизить потребляемую мощность при одновременном повышении быстродействия можно, используя ЗЭ на дополнительных МДП-транзисторах (рис. 4.39). Пороговые напряжения открывания транзисторов подбирают таким образом, чтобы в одном плече триггера при открытом нижнем транзисторе верхний был закрыт и наоборот. Вследствие этого потребляемая мощность теоретически равна нулю, но вследствие токов утечки составляет доли мкВт/бит.

В полупроводниковых ЗУ с В-поиском используют динамические ЗЭ с хранением информации на паразитных емкостях. Известны одно-, двух-, трех- и четырехтранзисторные динамические ЗЭ (рис. 4.40), в которых наличие заряда на конденсаторе соответствует 1, а отсутствие — 0. Наиболее простыми являются одно- и двухтранзисторные

ЗЭ. При чтении запоминающий конденсатор в этих ЗЭ подключается к РШ. Для надежного воспроизведения считанного сигнала необходимо, чтобы емкость РШ была много меньше емкости ЗЭ. Осуществить это трудно, поэтому такие ЗЭ не нашли распространения.

Постоянная времени разряда запоминающего конденсатора имеет порядок $10^{12} \text{ Ом} \times 10^{-12} \text{ Ф} = 1 \text{ с}$. Для предотвращения самопроизвольного стирания информации необходимо за это время обратиться к каждой ячейке. Естественно, что в процессе работы ЗУ гарантировать выполнение этого условия нельзя. Поэтому в ЗУ с динамическими ЗЭ должны быть предусмотрены специальные схемные средства для обновления информации. Для обновления цифры в ЗЭ с одним, двумя или тремя транзисторами слово из соответствующей ячейки

надо считать на регистр слов и вновь записать в ту же ячейку. Для обновления информации

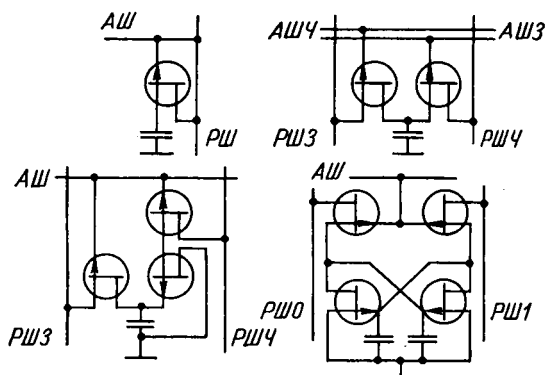


Рис. 4.40

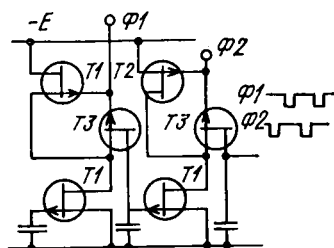


Рис. 4.41

в ЗЭ с четырьмя транзисторами достаточно подать на АШ, РШ0 и РШ1 питающее напряжение. В результате этого ЗЭ превращается в обычный триггер, в котором запоминающий конденсатор автоматически подзаряжается и, следовательно, отпадает необходимость в регенеративной перезаписи. Однако это преимущество достигается за счет усложнения схемы ЗЭ. Малая потребляемая мощность и относительная простота позволяют применять трех- и четырехтранзисторные ЗЭ в ЗУ большой емкости (10^7 бит).

Динамические ЗЭ в накопителях с В-поиском объединяют в последовательные сдвигающие регистры с двух- или четырехфазным тактированием. Каждый разряд двухфазного регистра (рис. 4.41) состоит из инвертора ($T1$ и $T2$) и элемента И ($T3$). Запоминающей является емкость затвора транзистора $T1$. Один разряд регистра состоит из двух каскадов, а для запоминания n -разрядного слова необходимо $2n$ каскадов. Передача цифры с каскада на каскад сопровождается ее инвертированием и происходит в моменты действия сигналов $\Phi1$ и $\Phi2$, открывающих транзисторы $T2$ и $T3$ соответствующих каскадов. Если конденсатор данного каскада заряжен (не заряжен), то транзистор $T1$ будет открыт (закрыт) и, следовательно, конденсатор следующего каскада разрядится (зарядится) через $T1$ и $T3$ ($T2$ и $T3$). Недостатком такой схемы является относительно большая потребляемая мощность (1—2 мВт/бит) вследствие сквозного тока, протекающего через

транзисторы $T1$ и $T2$. Устранить этот недостаток позволяет схема ЗЭ с четырехфазным тактированием (рис. 2.27), описанная в п. 2.9. В одном модуле (на одной подложке) обычно размещают регистры емкостью 64, 128, 256, 512, и т. д. до 8 К бит, на базе которых строят специализированные буферные ЗУ, а также ЗУ с В-поиском для замены накопителей с магнитной записью малой емкости.

Таблица 63

Микросхемы	Технология	Емкость (слов $\times$ $\times$ разряд)	Время выборки/ цикл чтения, нс	Потребляемая мощность: режим хран- ения/режим обращения, мВт/бит	Напряже- ние пита- ния, В	Время регенера- ции для динамиче- ских ЗУ, мс
K155PY5	ТТЛ	256×1	78/100	2,9/2,9	5	—
K176PY2	МДПДТ	256×1	650/—	0,02/—	9	—
K500PY410	ТЛЭС	256×1	45/60	2,8/2,8	5,2	—
K541PY1	ИИЛ	4096×1	120/200	0,12/0,12	5	—
K565PY2	МДП	1024×1	450/450	0,2/0,4	5	—
K565PY3	МДП	16384×1	300/510	0,004/0,04	5; 12	2

В табл. 63 приведены данные о некоторых серийно выпускаемых ЗУ на биполярных и униполярных транзисторах. ЗУ на статических ЗЭ более надежные и быстродействующие, чем ЗУ на динамических элементах, работают в широком диапазоне температур, но уступают им по потребляемой мощности, емкости и стоимости. ЗУ на МДП-транзисторах p -типа обеспечивают большее, чем у ЗУ на МДП-транзисторах n -типа, быстродействие, емкость и меньшее потребление мощности, однако уступают им по стоимости. ЗУ на биполярных транзисторах обеспечивают наибольшее быстродействие по сравнению с ЗУ других типов, но уступают ЗУ на МДП-транзисторах по всем остальным параметрам.

При построении ЗУ полупроводниковый накопитель разбивают на части, каждая из которых имеет технологически приемлемое число транзисторов. Разбиение накопителя на части можно выполнить различными способами. Чаще всего используют те способы, которые минимизируют число выводов интегрального модуля, в котором размещена часть накопителя. Если модуль стандартный с фиксированным числом выводов, то такая оптимизация может производиться для получения наибольшего числа компонент, приходящихся на один вывод.

Рассмотрим способы разбиения на части полупроводникового накопителя с выборкой 2Д. Пусть A и R — соответственно число адресных и разрядных шин, соединенных с одним ЗЭ. Тогда количество выводов модуля

$$B = Ak + Rm,$$

где k — число слов в модуле; m — разрядность этих слов. Так как $km = M$, где M — число ЗЭ в модуле (емкость модуля), то

$$B = Ak + R \frac{M}{k}.$$

Отсюда следует, что наименьшее количество выводов при заданном M будет иметь модуль, у которого $k = \sqrt{\frac{RM}{A}}$ и $m = \sqrt{\frac{AM}{R}}$. При этом на один вывод будет приходиться

$$M_1 = \frac{M}{B} \frac{M}{A \sqrt{\frac{RM}{A}} + R \sqrt{\frac{AM}{R}}} = \frac{1}{2} \sqrt{\frac{M}{RA}}$$

запоминающих элементов. Если же модуль имеет фиксированное число B_0 выводов, то максимальное число $3Э$ в модуле равно

$$M = B_0^2 \frac{1}{4RA}.$$

Обычно $A = 1$, $R = 2$, $B_0 = 14 - 2 = 12$ (два вывода для питания). Тогда $M = \frac{144}{8} = 18$, что является весьма низким значением для максимального числа $3Э$ в модуле. Кроме того, при таком разбиении накопителя для построения дешифратора адреса нужны специальные модули. Эти недостатки могут быть устранены включением ДСА в каждый модуль. Число выводов в таком модуле составляет

$$B = \log_2 k + Rm = \log_2 k + R \frac{M}{R}.$$

Отсюда для оптимального разбиения получаем

$$k = RM \ln 2; \quad m = \frac{1}{R \ln 2}.$$

Так как функция B принимает наименьшее значение при $m \approx 1$, то для отыскания целочисленного m , обеспечивающего минимум B , необходимо проверить его значения, наиболее близкие к указанному, а именно: $m_1 = 1$ и $m_2 = 2$. При $k = 2^i$, $m_1 = 1$, получаем $B_1 = i + R$, а при $k = 2^{i-1}$, $m_2 = 2$ — $B_2 = i - 1 + 2R$. При $R = 1$ имеем $B_1 = B_2$. Следовательно, разбиение накопителя будет оптимальным при одно- или двухразрядных словах ($R = 1$). На один вывод модуля в этом случае приходится запоминающих элементов

$$M_1 = \frac{M}{\log_2 M + R},$$

а их количество в модуле с фиксированным числом выводов B_0 будет равно

$$M = 2^{B_0 - R}.$$

При $R = 2$, $B_0 = 12$ получим $M = 1024$, что значительно больше, чем в случае разбиения накопителя на модули без дешифраторов.

Недостатком такого способа разбиения является избыточность модуля, что приводит к усложнению схемы и увеличению потребляемой мощности. Если эти факторы имеют решающее значение для данной серии модулей, то оптимизацию разбиения следует вести, минимизируя целевую функцию $f(B, C, P)$ вида, например,

$$f(B, C, P) = B^\alpha C^\beta P^\gamma$$

или

$$f(B, C, P) = \alpha B + \beta C + \gamma P,$$

где α , β , γ — весовые коэффициенты; B , C и P — соответственно число выводов, стоимость и потребляемая мощность.

Иногда с целью оптимизации стоимости модулей применяют два дешифратора адреса ДСА1 и ДСА2 и коммутатор, т. е. внутренняя структура модуля подобна схеме на рис. 4.7. Для выборки одного $m_1 n_1$ -разрядного слова используют ДСА1, а затем с помощью ДСА2 и коммутатора выбирают одно m_1 -разрядное слово. Число выводов в таком модуле

$$B = \log_2 k + \log_2 n_1 + m_1 = \log_2 k n_1 + m_1.$$

Сравнивая это выражение с выражением для B , полученным ранее, нетрудно заметить, что все выводы по k и m , полученные ранее, при $R = 1$ однозначно переносятся на $k n_1$ и m_1 . Следовательно, $m_1 = 1$, $m_2 = 2$, а ДСА1 и ДСА2 будут иметь наименьшее число выводов при $k = n_1$.

В ЗУ с выборкой 3Д накопитель можно разбивать на части отдельно и вместе с ДСА. Запись и чтение из модуля осуществляют по одному разряду, так как обращение к двум и более разрядам технологически неосуществимо. Модуль без дешифратора в этом случае имеет

$$B = Ak + Am + R$$

выводов, где k и m — число запоминающих элементов в модуле по горизонтали и по вертикали. Поэтому оптимальным будет модуль при $k = m = \sqrt{M}$, число M_1 ЗЭ на один вывод

$$M_1 = \frac{M}{B} = \frac{M}{2A\sqrt{M} + R} \approx \frac{\sqrt{M}}{2A},$$

а максимальное число ЗЭ в модуле с B_0 выводами

$$M = \frac{(B_0 - R)^2}{4A^2},$$

что несколько хуже, чем в модуле накопителя с выборкой 2Д (без дешифратора).

При разбиении накопителя с выборкой 3Д на модули совместно с дешифраторами адреса получим

$$B = \log_2 k + \log_2 m + R,$$

отсюда при $k = m$

$$M_1 = \frac{M}{B} = \frac{M}{\log_2 M + R}; \quad M = 2^{B_0 - R},$$

что совпадает с аналогичными данными для ЗУ с выборкой 2Д.

Таким образом, на основе изложенного, можно сделать вывод о том, что при одинаковом способе разбиения модули ЗУ с выборкой 3Д и 2Д практически равноценны. Поэтому для ЗУ на основе больших интегральных схем одинаково пригодны как выборка 2Д, так и 3Д.

4.7. НАКОПИТЕЛИ НА ПРИБОРАХ С ЗАРЯДОВЫМИ СВЯЗЯМИ И МАГНИТНЫХ ДОМЕННЫХ ЭЛЕМЕНТАХ

К числу наиболее перспективных ЗУ, разработанных в последние годы, относятся приборы с зарядовыми связями (ПЗС) и магнитные доменные (пузырьковые) элементы (МДЭ). Их используют для построения сдвигающих регистров, которые объединяют в накопители с В-поиском.

Структура одного из возможных регистров на ПЗС показана на рис. 4.42. На поверхности монокристаллического полупроводника n -типа (n -Si), покрытого слоем окисла SiO_2 , формируют электроды двух типов: исток (и) — приемник и затвор (з). Если к истоку такой МДП-структуры приложить напряжение, отталкивающее основные

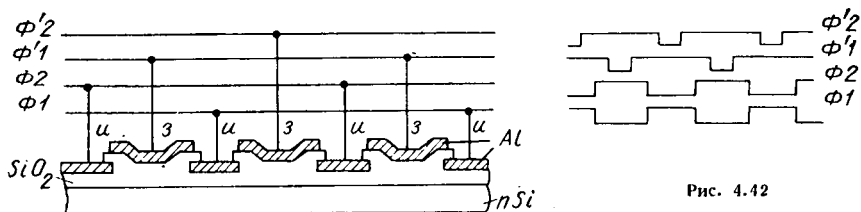


Рис. 4.42

носители, то у поверхности полупроводника образуется обедненная область, притягивающая неосновные носители («потенциальная яма» для неосновных носителей). Если это напряжение продержится достаточно долго без изменения, то вследствие термогенерации электронно-дырочных пар в объеме полупроводника в потенциальной яме накопится равновесное количество неосновных носителей. Если же период изменения напряжения на истоке будет значительно меньше, чем время наступления термравновесия, то наличие или отсутствие заряда неосновных носителей, введенных в потенциальную яму каким-либо образом, может служить признаком для хранения информации. Ввод заряда в потенциальную яму можно осуществлять освещением, вызывающим генерацию электронно-дырочных пар, или инжекцией заряда p - n переходом МДП-транзистора, выполненного на входе регистра. Сдвиг информации в регистрах на ПЗС осуществляют специальным тактированием (рис. 4.42). Для передачи заряда потенциал истока повышают относительно потенциала приемника, после чего на соответствующий затвор подают потенциал, меньший или равный потенциалу истока. Вследствие этого заряд из области истока через открытый потенциальный барьер под затвором перемещается в область приемника. В следующем такте исток и приемник меняются ролями. При указанном на рис. 4.42 порядке следования фаз сдвиг осуществляется вправо. Если изменить порядок следования фаз $\Phi'1$ и $\Phi'2$, то сдвиг будет осуществляться влево. За время действия фаз $\Phi'1$ и $\Phi'2$ полная передача заряда не происходит, т. е. по мере сдвига заряд уменьшается с коэффициентом 0,970—0,999. Следовательно, через некоторое число (например, 48—256) каскадов необходимо об-

овлять информацию. Обновление, так же как считывание и запись, осуществляется МДП-транзистором, выполненным на одной подложке ПЗС.

Основным достоинством элементов на ПЗС является простота изготовления, обусловленная отсутствием технологической операции диффузии. Диффузионные области необходимы только для создания МДП-транзисторов, используемых для записи, чтения и обновления информации. Это способствует повышению плотности упаковки ЗЭ до 10^4 бит/мм², увеличению степени интеграции и повышению быстродействия (частота СИ около 10^6 — 10^8 Гц). Кроме ЗУ, элементы ПЗС применяют в качестве аналоговых линий задержки (величина вводимого в первый ПЗС заряда может быть пропорциональна входному напряжению) и в полупроводниковых видеоканалах передающих

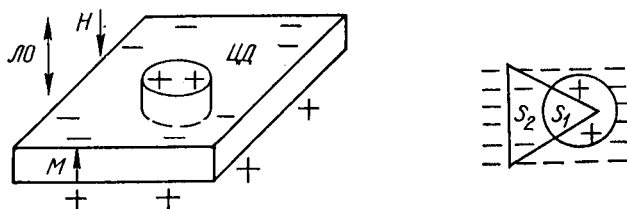


Рис. 4.43

телевизионных камер, где используется возможность преобразования светового сигнала в электрический при освещении ПЗС.

Принцип построения накопителей на магнитных доменных элементах основан на управляемом движении магнитных доменов (магнитных пузырьков) — областей самопроизвольной намагниченности, которых вектор намагниченности направлен противоположно направлению намагниченности остальной сплошной магнитной среды. Такие домены возникают в тонкой магнитной пленке, изготовленной из ортоферрита (кристаллического материала на основе окислов железа и редкоземельных металлов). Известно два вида магнитных доменных элементов: элементы с цилиндрическими доменами (ЭЦД) и элементы с плоскими доменами (ЭПД).

Для создания ЗУ на ЭЦД используют пленки или пластинки ортоферрита, единственная ось легкого намагничивания ЛО которых направлена перпендикулярно к плоскости пластинки. По различным причинам в таких пластинках возникают лабиринтные поперечные домены. Если перпендикулярно к поверхности пластины приложить магнитное поле смещения с напряженностью H , домены в виде полосок превратятся в цилиндрические (рис. 4.43), вектор намагниченности M которых направлен противоположно H (знаки «+» и «—» — на рис. 4.43 являются условными обозначениями направления векторов намагниченности). Увеличение напряженности поля смещения вызывает сначала сжатие, а затем и исчезновение доменов. При уменьшении напряженности происходит обратный процесс увеличения размеров доменов и превращение их в полоски.

Во многих схемах на ЭЦД используют свойство цилиндрических доменов находиться в контакте с областью, имеющей высокую

магнитную проницаемость. Такие области создаются нанесением, например, пермаллоевой пленки (аппликации) на пластину ортоферрита. Рассмотрим один из вариантов сдвигающего регистра, у которого пермаллоевые области имеют клиновидную форму (рис. 4.43). Экспериментально показано, что домен располагается возле аппликации так, что площади S_1 и S_2 равны, и его легче сместить к острию

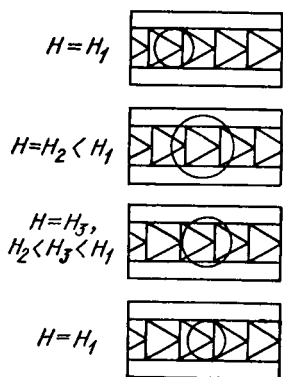


Рис. 4.44

аппликации, чем к широкому краю. Это свойство положено в основу принципа действия регистров с модуляцией поля смещения (рис. 4.44). Для сдвига домена уменьшают напряженность поля смещения. При $H_2 < H_1$ домены захватывают своей стенкой широкий конец следующего клина. После этого напряженность поля увеличивают, домен сжимается, его задняя стенка соскальзывает с острия и при $H = H_1$ домен занимает исходное положение, но уже на соседнем справа клине.

В последние годы были разработаны на ЭЦД и накопители с М-поиском. Один из вариантов такого накопителя показан на рис. 4.45, а. Накопитель состоит из спаренных адресных и разрядных шин записи, нанесенных на ортоферритовую пленку. Области, в которых перемещаются цилиндрические домены, имеют Г-образную форму (рис. 4.45, б). По положению домена в этой области и определяют состояния ЗЭ. Исходное положение магнитного домена отождествляется с состоянием 0. Запись 1 осуществляется подачей тока на АШЗ, создающего поле напряженностью H_a . Под действием этого поля

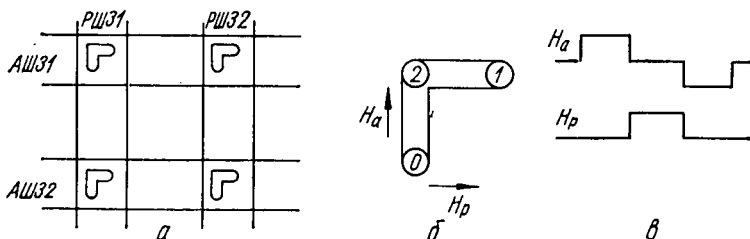


Рис. 4.45

домен перемещается и занимает положение 2. После этого (рис. 4.45 б, в) создают поле H_p с помощью тока в РШЗ, которое перемещает домен в положение, соответствующее 1. Для приведения полувыбранных доменов в исходное состояние создают поле $-H_a$, перемещающее домен из области 2 в область 0. Запись 0 осуществляется последовательной подачей токов, формирующих поля напряженностью $-H_p$, $-H_a$, H_p . Считывание информации в таком накопителе основано на использовании магнитооптических эффектов, заключающихся во вращении плоскости поляризации когерентного оптического излучения, получаемого с помощью лазера, при прохождении света через намаг-

ниченную пленку (эффект Фарадея) или при отражении света непрозрачной пленкой (эффект Керра). Сигнал на выходе накопителя воспроизводится фотодиодной матрицей.

Для создания накопителей на ЭПД используют пленки, единственная ось легкого намагничивания которых лежит в плоскости пленки. В таких пленках по разным причинам могут возникать домены с обратной намагниченностью. При приложении внешнего магнитного поля, совпадающего по направлению с вектором намагниченности доменов, последние начинают расти во всех направлениях. Ограничить боковой рост доменов можно за счет использования низкокоэрцитивных каналов. Для их получения на стеклянную подложку наносят слой алюминия, в котором с помощью фотолитографии протравливают каналы нужной конфигурации, после чего наносят магнитный слой, который над алюминием имеет большую, а над каналами — малую коэрцитивную силу.

Одна из конфигураций сдвигающего регистра, выполненного по такому способу, показана на рис. 4.46. Материал пленки вне и внутри канала K намагничен вдоль легкой оси. Наличие домена в области (узле) пересечения каналов соответствует «1», а отсутствие — «0». Для сдвига домена к пленке прикладывают продвигающее поле напряженностью H_n , вследствие чего размеры домена растут в направлении поля и занимают весь канал. После этого для восстановления состояния канала подают стирающее поле напряженностью H_c , уменьшающее размеры домена. Для того чтобы домен не был уничтожен этим полем, к нижним узлам канала прикладывают локальное удерживающее поле H_y , которое создается током, протекающим по удерживающим шинам, проходящим вдоль узлов. В следующем такте сдвига компонента H_T продвигающего поля H_n меняет свое направление на противоположное, вследствие чего доменная верхушка перемещается вверх. После подачи стирающего и удерживающего поля домен остается в верхнем узле, но уже сдвинут на одну позицию вправо. Направление сдвига в описанном регистре определяется направлением составляющей H_T продвигающего поля H_n , приложенного в первом такте. Изменяя ориентацию этой составляющей, можно менять и направление сдвига на противоположное.

Сдвигающие регистры на магнитных пузырьковых элементах обладают следующими достоинствами: большая плотность (10^3 — 10^5 бит/см²) размещения информации, высокое быстродействие (скорость записи составляет 10^5 — 10^7 бит/с), простота технологии, малое потребление мощности, возможность создания неограниченных по длине реги-

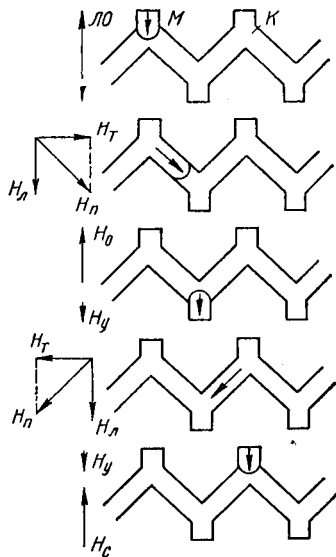


Рис. 4.46

стров. Широкое внедрение доменных элементов может значительно изменить структуру цифровых ЭВМ, так как при этом можно создавать устройства с распределенными по объему функциями хранения и переработки информации, а возможность визуального наблюдения доменов позволяет упростить процесс исследования и проектирования таких устройств.

4.8. ПОСТОЯННЫЕ И ПОЛУПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Для хранения часто встречающихся при решении различных задач констант, таблиц, стандартных подпрограмм и другой информации нашли широкое применение ЗУ, в которых информация записывается при их создании. Замена записанной ранее информации в таких ЗУ возможна только путем изменения схемы накопителя. Такие ЗУ получили название постоянных (ПЗУ).

Функции ПЗУ проще, чем у обычных ЗУ, поэтому они являются более быстродействующими, чем активные ЗУ, и имеют меньшую стоимость. Большинство ПЗУ строится по схеме с прямой выборкой (рис. 4.47). Накопитель ПЗУ часто называют шифратором, так как он преобразует адрес искомого слова в его код. Иногда, с целью упрощения ПЗУ, в его схему вводят выходную комбинационную схему (ВКС), управляемую частью RGA (рис. 4.47).

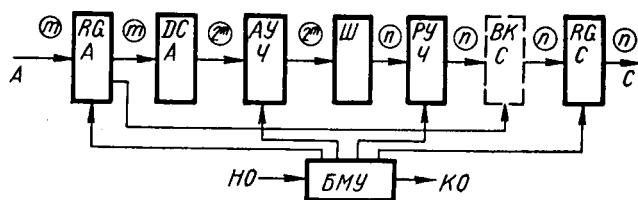


Рис. 4.47

Шифраторы (Ш) обычно строят по схеме, показанной на рис. 4.48. В зависимости от используемого связующего звена z различают линейные и нелинейные Ш. В линейных Ш используются резисторы, конденсаторы и трансформаторы с линейными характеристиками. Например, один из способов изготовления линейных резисторных шифраторов состоит в нанесении на изолирующую подложку адресных шин с резисторами R (рис. 4.49). Отверстия А пробиваются для нарушения гальванических связей между АШ и РШ при записи 0. При записи 1 отверстия не пробиваются. Отверстия В окружены печатными лепестками и служат для соединения карт между собой путем их прошивки РШ. Карты собираются в пакет и отверстия В заливаются припоем, образуя систему РШ.

На практике наибольшее распространение получили трансформаторные Ш. Известно два основных типа таких шифраторов: на разрядных трансформаторах (РТ) и на адресных трансформаторах (АТ). В шифраторе первого типа (рис. 4.50, а) с помощью одного сердеч-

ника воспроизводятся одноименные разряды всех слов, а число РТ равно числу разрядов. Прохождение АШ через РТ соответствует считываемой 1, а пропуск РТ соответствует 0.

В шифраторе на АТ (рис. 4.50, б) на одном сердечнике хранится одно слово, а число АТ равно числу слов, записанных в ПЗУ.

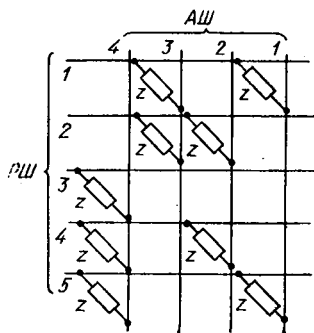


Рис. 4.48

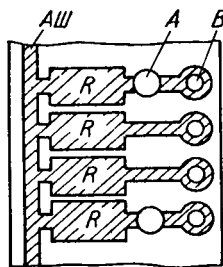


Рис. 4.49

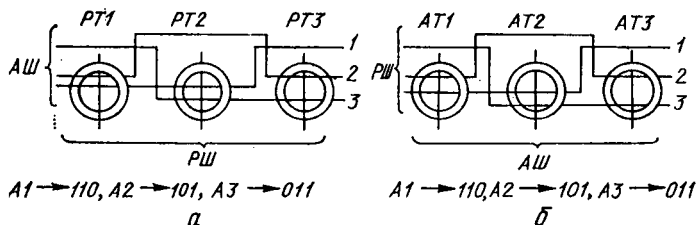


Рис. 4.50

С целью повышения помехоустойчивости и стабилизации нагрузки на АУЧ используют способы прошивки сердечников РТ и АТ, отличающиеся от показанных на рис. 4.50. Эти способы прошивки иллюстрируются на рис. 4.51. Выходные сигналы на РШ при считывании 1 и 0 в этом случае отличаются не только амплитудой, но и полярностью. Большим преимуществом трансформаторных ПЗУ является простота их адресных цепей, выполняемых по аналогии со схемами на рис. 4.23 и 4.24.

Недостатком всех линейных Ш является наличие различных помех и наводок. Нелинейные шифраторы позволяют уменьшить многие паразитные связи. Строятся такие шифраторы в основном на диодах и сердечниках с ППГ. Из-за большого числа диодов стоимость диодных ПЗУ на дискретных компонентах велика. Поэтому они нашли ограниченное применение в ЭВМ 1-го и 2-го поколений. Однако освоение интегральных способов изготовления диодов привлекло интерес к таким ПЗУ. Изготавливают их обычно в два приема. Сначала с помощью интегральной технологии изготавливается сетка РШ и АШ, в каждом узле которой имеется диод. Затем на выбранные АШ и РШ

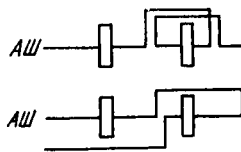


Рис. 4.51

подается высокое напряжение. В результате действия большого обратного напряжения ненужный диод пробивается, нарушая тем самым гальваническую связь АШ и РШ.

Первый способ построения ПЗУ на сердечниках с ППГ аналогичен построению ПЗУ на линейных трансформаторах (рис. 4.50—4.51). Однако пороговые свойства сердечников с ППГ позволяют строить ПЗУ с выборкой полуточками. В одном из таких ПЗУ каждый сердеч-

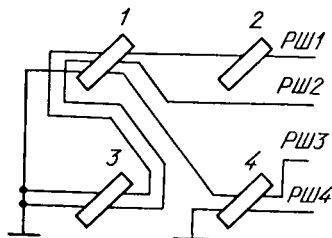


Рис. 4.52

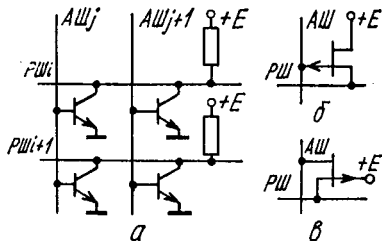


Рис. 4.53

ник служит для запоминания одного слова (рис. 4.52). Сердечники объединены в матрицу. Каждый сердечник имеет две АШ — X и Y и p РШ, где p — число единиц в слове, записанном в данном сердечнике. Кроме того, имеется еще одна вспомогательная шина, служащая для приведения всех сердечников в исходное состояние. В ПЗУ на рис. 4.52 в первом сердечнике записано слово 1110, во втором — 1000 и т. д.

ПЗУ на дискретных транзисторах ранее практически не использовались в ЦВТ, однако успехи в производстве больших интегральных схем сделали транзисторные ПЗУ наиболее распространенными и перс-

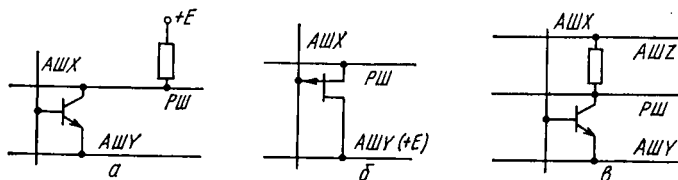


Рис. 4.54

пективными. В настоящее время ПЗУ, в которых шифраторы и «электронное обрамление» выполнены на транзисторах, выпускаются серийно как большие интегральные схемы. Как и двусторонние ЗУ, ПЗУ являются однородными структурами с простейшими компонентами. В качестве элементов связи в транзисторных шифраторах ПЗУ могут использоваться как биполярные, так и униполярные транзисторы. На рис. 4.53 показано включение транзисторов в накопители типа 2Д, причем включение МДП-транзистора на рис. 4.53, б называют активным, а на рис. 4.53, в — пассивным. Выборку типа 3Д в ПЗУ реализуют при включении транзисторов, показанном на рис. 4.54, а, б. Четырехмерная выборка (4Д) реализуема на биполярных транзисторах (рис. 4.54, в). Важным качеством транзисторных шифраторов является наличие у них свойств усиления и формирования сигналов.

В зависимости от способа записи информации в транзисторные ПЗУ различают ПЗУ с масочной и электрической записью (масочным и электрическим программированием). ПЗУ первого типа изготовляют с учетом записываемой в них информации, т. е. элементы связи — транзисторы — в технологическом процессе формируют с помощью масок (шаблонов) только в тех узлах АШ и РШ, где они необходимы. В ПЗУ второго типа шифраторы содержат полный набор транзисторов, а запись информации происходит путем электрического выжигания плавких соединений. При производстве больших партий ПЗУ экономически выгоднее использовать масочную запись, так как изготовление индивидуальных масок окупается упрощением технологии. При выпуске малых серий более выгодны ПЗУ с электрической записью. Обычно на одной подложке с шифратором размещают и другие схемы ПЗУ: регистры адреса и слова, дешифраторы адреса и др. Плотность записи информации в транзисторных ПЗУ достигает 2^{10} бит/мм². В табл. 64 приведены основные характеристики БИС ПЗУ различных типов, достигнутые к настоящему времени.

Таблица 64

Тип БИС ПЗУ	Технология	Емкость, бит	Цикл чтения, нс
С масочной записью	МДП	64К	300
	ТТЛ	4К	60
	ТЛЭС	1К	15
С электрической записью	МДП	16К	450
	ТТЛШ	4К	35
	ТЛЭС	1К	25

Несмотря на то что емкость серийных БИС ПЗУ постоянно увеличивается, в процессе проектирования аппаратных средств ЦВТ часто возникают ситуации, когда емкость одного модуля ПЗУ недостаточна для размещения в нем некоторого информационного массива. При этом требуется наращивать либо длину считываемых из ПЗУ слов, либо число адресов, либо то и другое одновременно. Рассмотрим возможные пути решения этой задачи на примерах. Пусть необходимо построить ПЗУ емкостью 16 12-разрядных слов из модулей ПЗУ емкостью 16 4-разрядных слов. Очевидно, что в этом случае необходимо использовать 3 модуля ПЗУ, в одном из которых хранятся младшие 16 4-разрядных слов, в другом — следующие 16 4-разрядных слов и т. д. (рис. 4.55, а). Адресные же цепи всех модулей ПЗУ в этом случае включаются параллельно.

Пусть теперь из модулей ПЗУ той же емкости требуется построить ПЗУ емкостью 64 4-разрядных слова. Для этого необходимо 4 модуля ПЗУ, мультиплексор и селектор. Дешифраторы, входящие в состав мультиплексора и селектора, управляют старшими разрядами адреса (рис. 4.55, б), т. е. с помощью старших разрядов задают номер модуля ПЗУ, где хранится искомое слово. Младшие же разряды адреса коммутируются мультиплексором и задают номер искомого слова в выбранном модуле ПЗУ. Селектор коммутирует выбранное слово с выходов модуля на выходы ПЗУ. Если селектор и мультиплексор выполнены не как функционально-обособленные узлы, то схему ПЗУ можно упростить за счет того, что дешифратор старших разрядов

адреса может быть общим как для мультиплексора, так и для селектора. Построение ПЗУ емкостью 64 12-разрядных слова из рассматриваемых модулей показано на рис. 4.55, в. Здесь используется такой же мультиплексор, что и в предыдущем случае (т. е. коммутирующий 4-разрядные слова в 4 направлениях). Однако селектор в этой схеме отличается от предыдущего — он должен коммутировать 12-разрядные слова, приходящие по 4 каналам.

Простейшим приемом сокращения емкости ПЗУ является использование выходной комбинационной схемы (ВКС, рис. 4.47).

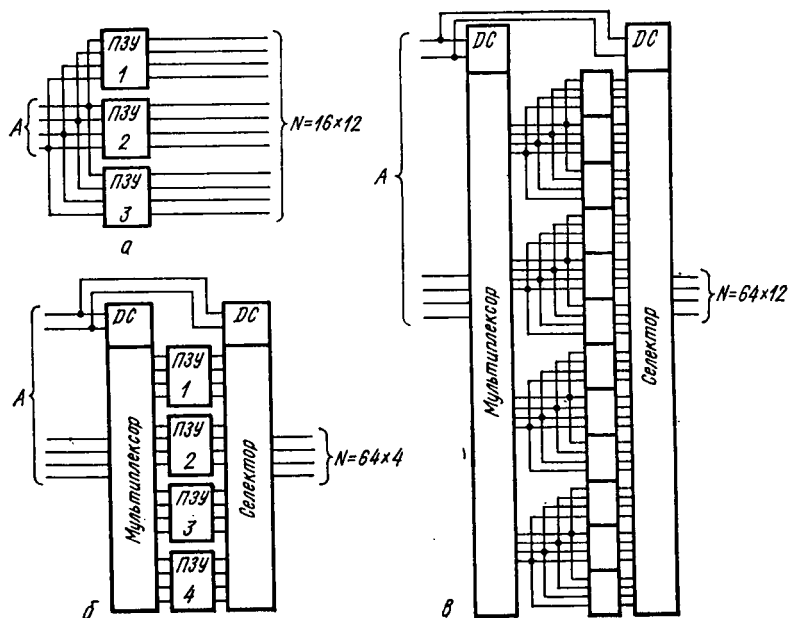


Рис. 4.55

Для этого необходимо выявить связи между адресами ячеек и словами, записанными в них. В некоторых случаях эта связь очевидна. Например, если в ПЗУ записана таблица функции $Y = X^2$, то 1 в последнем (младшем) разряде числа X вызовет появление в последних разрядах числа Y цифр 01, а 0 в младшем разряде X — цифр 00. ВКС в этом случае должна быть построена так, чтобы предпоследний разряд Y всегда был равен 0, а последний разряд Y равен последнему разряду X . Вследствие этого длину выходного слова можно сократить на 2 разряда, что приводит к уменьшению емкости ПЗУ на $2 \frac{N}{n}$ бит. Еще большего сокращения емкости удастся добиться, если введение ВКС приводит к уменьшению числа разрядов в адресе ПЗУ. Например, если ПЗУ используется для преобразования целых чисел из двоичной системы счисления в двоично-десятичную, то последний разряд любого числа в двоичной системе счисления совпадает с последним разрядом числа в двоично-десятичной системе и не влияет на остальные разряды двоично-десятичного представления числа.

Следовательно, длину адреса и выходного слова можно уменьшить на один разряд, что приводит к сокращению емкости ПЗУ более чем в 2 раза.

При проектировании интегральных модулей ПЗУ важное значение имеет выбор длины слова. Если модули предназначены, например, для построения ПЗУ большой емкости для хранения подпрограмм, то минимизация числа выводов требует уменьшения длины слова до одного или двух разрядов. Однако, в отличие от активных ЗУ, область применения ПЗУ не ограничивается техническими средствами для хранения информации (см. п. 4.9). Если ПЗУ используется, например, в качестве комбинационного сумматора двух l -разрядных слов, то при заданном числе B выводов модуля длину слова можно определить как $m = l + 1$, а l найти из неравенства

$$B \geq 3l + 2,$$

откуда

$$l \leq \frac{B-2}{3}.$$

Таким образом, оптимальная длина слов в интегральных модулях ПЗУ часто может быть значительно больше 1.

Рассмотренные выше ПЗУ не допускают изменения записанной в них информации. Между тем наличие возможности замены информации в ПЗУ часто бывает весьма полезным, так как встречаются информационные массивы, которые, хотя и редко, но изменяются. Примерами таких массивов являются программы решения массовых задач, выполняемые с некоторой периодичностью.

ПЗУ, допускающие смену записанной в них информации в специальном нерабочем режиме, получили название полупостоянных ЗУ (ППЗУ). В зависимости от способа перезаписи информации различают ППЗУ с механической и электрической сменой информации. Многие ППЗУ с механической сменой информации строят на тех же принципах, что и ПЗУ.

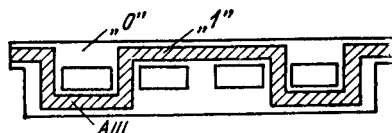


Рис. 4.56

Трансформаторные ППЗУ строят на разъемных сердечниках П-или Ш-образной формы. Адресные шины наносят на изолирующую подложку, например, печатным способом (рис. 4.56). Цифры записывают путем травления или перфорации соответствующих участков АШ. При использовании П-образных сердечников выполненные таким образом АШ либо огибают их, либо нет. Далее подложки складывают в пакет, в отверстия заводят разъемные сердечники, замыкают их магнитопроводы, а к контактным лепесткам на подложках подключают адресные усилители чтения. Смена информации осуществляется путем замены отдельных подложек или всего пакета.

Перспективный класс ППЗУ с механической сменой информации составляют оптические ЗУ. Один из способов построения таких ППЗУ состоит в использовании вращающихся фотоскопических дисков (рис. 4.57, а). Выбор дорожки здесь осуществляется с помощью

электроннолучевой трубки (ЭЛТ), управляемой посредством цифро-аналогового преобразователя словом, записанным в регистр адреса (М-поиск). Выбор места на дорожке осуществляется по В-способу. В дальнейшем луч удерживается на нужной дорожке с помощью следящей системы с фотозлектронным умножителем (ФЭУ) и интегратором (И). Следящая система удерживает луч в таком положении, чтобы коэффициент светопропускания вдоль дорожки был равен 0,5. Для того чтобы действие следящей системы не зависело от записываемой

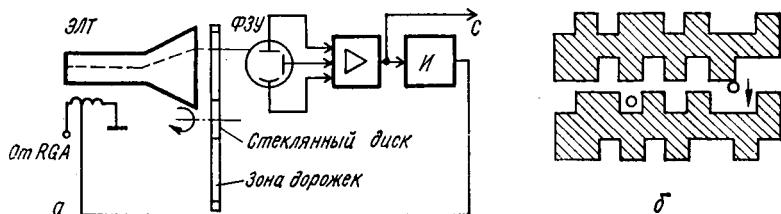


Рис. 4.57

мой информации, применяют специальный код, в котором за каждым разрядом следует его дополнение, т. е. 0 кодируется как 01, а 1 — как 10. Информационные дорожки отделены друг от друга прозрачной и непрозрачной границами (рис. 4.57, б). Действия следящей системы на луч при его отклонении от нормального положения показаны на рис. 4.57, б. Запись информации на фоточувствительный слой осуществляется с помощью луча лазера, сфокусированного до диаметра в 1 мкм. После записи диск такого ППЗУ подвергают обычному фотохимическому проявлению.

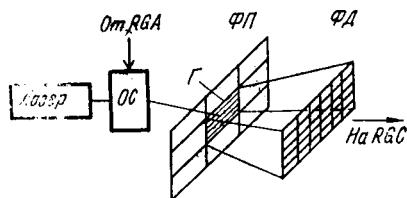


Рис. 4.58

в то время как на обычной фотографии фиксируется только квадрат амплитуды (интенсивность) световой волны. Работа ЗУ на голограммах поясняется схемой на рис. 4.58. Луч лазера поступает на отклоняющую систему (ОС), управляемую кодом адреса. Пройдя через ОС, луч попадает на выбранную голограмму (Г). Освещенная лучом лазера голограмма проецирует восстановленное изображение на матрицу фотодиодов (ФД). Изображение представляет собой матрицу из световых пятен. Яркое пятно возбуждает соответствующий фотодиод и воспроизводится как «1», а темное пятно — как «0». Запись информации осуществляют с помощью электроннолучевой трубки, на которой воспроизводится матрица пятен. Это изображение фотографируется на обычную фотопластинку (ФП), а затем с помощью специальной фотосистемы переносится на ФП голограмм, об-

разуя один кадр. На голограмме диаметром 1 мм может быть записано 10^4 пятен, а на одной ФП имеется до 10^4 (100×100) голограмм. Таким образом на одной ФП можно записать до 10^8 бит, при этом время выборки будет порядка 1 мкс. Наряду с плоскими предполагается использовать также и пространственные голограммы. При этом плотность записи может быть доведена до величины порядка 10^{12} бит/см².

Среди ППЗУ с электрической сменой информации необходимо прежде всего указать полупроводниковые ППЗУ на транзисторах со структурой металл-нитрид-окисел-полупроводник (МНОП). Запись и последующее запоминание информации основано здесь на явлении захвата заряда, инжектируемого в область между окислом и нитридом. Для этого к затвору транзистора прикладывается высокое положительное напряжение, превышающее некоторый критический уровень. Это приводит к появлению на границе слоев окисла и нитрида отрицательного заряда, снижающего порог U_1 переключения транзистора до величины U_2 (в качестве подложки используют кремний n -типа). Если на элементы связи в шифраторе, построенном на таких транзисторах, действовать адресным сигналом U_a , где $|U_2| < |U_a| < |U_1|$, то открываться будут только те транзисторы, в которых хранится заряд, т.е. записана «1». Для стирания информации к затвору прикладывается отрицательное напряжение такой же величины, как и при записи, что приводит к рассасыванию заряда. Кроме того, стирание информации может происходить самопроизвольно по мере утечки заряда. Поэтому время хранения информации в ППЗУ на МНОП-транзисторах зависит от величины накопленного заряда, которая, в свою очередь, зависит от величины и длительности записываемого напряжения. Изменяя эти величины, можно получать различные сроки сохранности записанной информации: от нескольких часов до многих лет.

Для создания ППЗУ применяют также транзисторы с лавинной инжекцией и плавающим затвором. Для записи информации к выводам стока и истока такого транзистора прикладывают напряжение, превышающее некоторую критическую величину. Это напряжение вызывает лавинную инжекцию заряда в область поликристаллического кремниевого затвора. Этот заряд захватывается и транзистор переходит в открытое состояние. Так как затвор не имеет внешнего вывода, то для стирания информации используют ультрафиолетовое или рентгеновское облучение. Срок хранения информации без перезаписи в таком ППЗУ исчисляется десятками лет. Применяют такие транзисторы в сочетании с обычными МОП-транзисторами, которые выполняют функции ключей, подключающих транзистор с плавающим затвором к адресным и разрядным шинам.

ППЗУ могут быть построены также на основе эффекта изменения сопротивления (в 10^8 — 10^6 раз) аморфного полупроводника при переходе его из аморфного состояния в кристаллическое. Перевод аморфного полупроводника в кристаллическое состояние осуществляют путем его электрического пробоя с последующим пропусканием небольшого тока. После окончания действия тока полупроводник остается в низкоомном состоянии. Для восстановления сопротивления (т.е.

для стирания информации) через рабочий участок пропускают коротко-временный импульс тока большой амплитуды.

В табл. 65 приведены некоторые технические характеристики БИС ПЗУ с электрической сменой информации, достигнутые к настоящему времени.

Таблица 65

Технология ПЗУ	Емкость, бит	$t_{\text{оз}}$, с	$t_{\text{оч}}$, нс	Время хранения, ч	Напряжение записи (стирания), В	Напряжение чтения, В
МНОП С плавающим затвором	4096	0,01	200	3000	≈ 50	≈ 10
	8192	30	400	8000	—	≈ 12

4.9. ТАБЛИЧНЫЕ И ТАБЛИЧНО-АЛГОРИТМИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ УСТРОЙСТВА НА ОСНОВЕ ПЗУ

Постоянные ЗУ в первых ЭВМ использовались для хранения констант и стандартных программ. Однако по мере совершенствования характеристик ПЗУ их применение расширялось. Освоение производства БИС сделало ПЗУ одной из важнейших интегральных компонент ЭВМ. Объясняется это прежде всего их универсальностью. На базе ПЗУ может быть построена комбинационная схема, цифровой автомат с памятью, арифметический блок и др.

Возможность построения произвольной комбинационной схемы может быть показана следующим образом. Известно, что функционирование комбинационной схемы с n входами и m выходами может быть описано таблицей соответствия наборов входных $\vec{x} = (x_1, x_2, \dots, x_n)$ и выходных $\vec{y} = (y_1, y_2, \dots, y_m)$ сигналов. Имея ПЗУ емкостью 2^n m -разрядных слов, можно в ячейки с адресами, указанными во входной части таблицы, записать слова, указанные в выходной части этой таблицы. ПЗУ, хранящее такую таблицу, будет работать как комбинационная схема: набор сигналов $\vec{x}$ на входе ПЗУ вызовет появление на его выходе набора сигналов $\vec{y}$. Если же число входов k и выходов p комбинационной схемы больше, чем входов m и выходов n ПЗУ, то такие схемы могут быть построены из логических элементов, реализуемых ПЗУ. При этом следует иметь в виду, что ПЗУ может рассматриваться как многофункциональный логический элемент, реализующий m произвольных функций от n переменных. Например, схема на рис. 4.59, а представляет собой комбинационную схему, реализующую систему функций

$$\begin{aligned} y_1 &= f_1(f_3(x_1, x_2, x_3, x_4), x_5, f_4(x_1, x_2, x_3, x_4), x_6); \\ y_2 &= f_2(f_3(x_1, x_2, x_3, x_4), x_5, f_4(x_1, x_2, x_3, x_4), x_6), \end{aligned}$$

где f_1, f_2, f_3, f_4 могут быть произвольными функциями четырех аргументов.

Если в ячейки ПЗУ записать их адреса, т. е. в ячейку с адресом i записать число i , и соединить выходы ПЗУ с его входами, то получим регистр. Действительно, слово i на входе ПЗУ вызовет появление этого слова на выходе ПЗУ, которое, поступая на вход, подтверждает i -е состояние ПЗУ. Для управления таким регистром необходимо иметь еще $n + 1$ вход, где n — длина слов в ПЗУ (рис. 4.59, б), а входное слово ПЗУ будет иметь структуру $(\vec{x}, \vec{z}, a)$, где $\vec{x} = (x_1, x_2, \dots, x_n)$ — входы обратной связи с выхода ПЗУ; $\vec{z} = (z_1, z_2, \dots, z_n)$ — входы для приема слов на регистр; a — управляющий сигнал для задания режимов приема слов и их хранения. Соответствие адресов и слов в ПЗУ должно определяться соотношениями:

$$(\vec{x}, \vec{z}, a) \rightarrow \vec{y}; (i, j, 0) \rightarrow i; (i, j, 1) \rightarrow j,$$

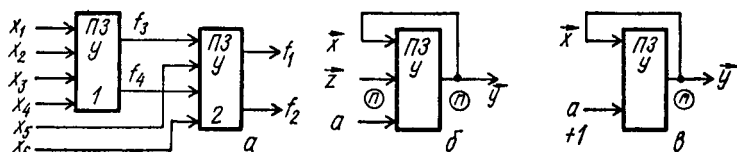


Рис. 4.59

т. е. при $a = 0$ слово на выходе ПЗУ равно слову i на входах $\vec{x}$ (режим хранения), а при $a = 1$ слово на выходе равно слову j на входах $\vec{z}$ (режим приема слова). Недостатком такой реализации регистра является большая избыточность. Если в ПЗУ каждому из $(2n + 1)$ -разрядных входных слов соответствует n -разрядная ячейка, то общая емкость ПЗУ составляет $n \cdot 2^{n+1}$ бит, а на один разряд регистра приходится 2^{n+1} бит. Избыточность ПЗУ-регистра можно уменьшить, включив на входе ПЗУ селектор, управляемый сигналом a . В зависимости от значения a на вход ПЗУ коммутируются либо сигналы $\vec{x}$, либо сигналы $\vec{z}$. На один разряд регистра в этом случае требуется 2^n бит.

Более экономной схемой с памятью на основе ПЗУ является счетчик (рис. 4.59, в). Для его реализации соответствие адресов и слов в ПЗУ определяется следующим образом:

$$(\vec{x}, a) \rightarrow \vec{y}; (i, 0) \rightarrow i; (i, 1) \rightarrow i + 1,$$

т. е. здесь в ячейку с адресом i должно быть записано слово $i + 1$. Каждый сигнал $a = 1$ вызывает увеличение слова на выходе ПЗУ на 1. Поэтому сигнал a может воздействовать непосредственно на цепи управления приемом адреса на ПЗУ. Очевидно, что число состояний такого счетчика равно числу ячеек ПЗУ, а на один его разряд необходимо 2^n бит. Чтобы исключить переход счетчика через все его состояния за время действия сигнала $a = 1$, длительность сигнала должна быть равна циклу обращения к ПЗУ. По аналогии с рассмотренными примерами на базе ПЗУ может быть построен любой сколь угодно сложный цифровой автомат. Кроме того, на основе ПЗУ могут

быть эффективно реализованы высокопроизводительные арифметические устройства. В таких устройствах ПЗУ может быть использовано для выполнения любой операции, однако наиболее целесообразно их применение для вычисления одноместных функций вида $\sqrt{x}$, $\sin X$, $\log X$, $\operatorname{tg} X$, X^a , e^X и др. Для этого достаточно по адресу X в ПЗУ записать значение функции $f(X)$. Вычисление функции $f(X)$ сводится, таким образом, к однократному обращению к ПЗУ, вместо длительного цикла вычисления значения функции по подпрограмме. Быстродействие ЭВМ при этом повышается на 2—3 порядка. Характерной чертой таких вычислительных устройств является то, что в ПЗУ хранится, по существу, таблица функции $f(X)$ для всех возможных значений аргумента X . Поэтому подобные вычислительные устройства называют табличными (матричными).

Важнейшей проблемой, возникающей при построении табличных вычислительных устройств, является сокращение емкости ПЗУ. Уменьшить емкость ПЗУ удастся за счет применения специальных методов (называемых иногда методами «сжатия» информации в ПЗУ). Сущность одного из самых простых методов состоит в объединении адресных шин ячеек ПЗУ, содержащих одинаковые слова. Число ячеек в таком случае равно числу различных слов, записанных в ПЗУ, и не превышает 2^m , где m — длина выходного слова. Стоимость накопителя в таком табличном вычислительном устройстве можно оценить как $c_1 m 2^m + c_2 2^m$, где c_1 — стоимость одного запоминающего элемента (элемента связи); c_2 — стоимость одного элемента в схеме объединения адресных шин. Тогда для выполнения условия

$$c_1 2^m m + c_2 2^m < c_1 2^n m$$

необходимо, чтобы

$$\begin{aligned} 2^n m c_1 &> 2^m (m c_1 + c_2); \\ 2^{n-m} &> \frac{m c_1 + c_2}{m c_1} = 1 + \frac{c_2}{m c_1}. \end{aligned}$$

Обычно $\frac{c_2}{m c_1} \leq 1$. Поэтому можно последнее условие записать так

$$2^{n-m} > 2,$$

т. е., рассматриваемый метод обеспечивает упрощение ПЗУ, если $n > m + 1$.

Уменьшить емкость ПЗУ в табличном вычислительном устройстве можно, если, например, вместо значений функции во всех точках диапазона изменения аргумента хранить в ПЗУ ее приращения Δf относительно некоторой константы C , а окончательный результат получать путем сложения. Если аргумент X представлен n -разрядами, а функция — m разрядами, то при традиционном воспроизведении такой функции емкость накопителя ПЗУ должна составлять $2^n m$ бит. Если $C \neq 0$, то $\Delta f_{\max} < 2^m - 1$, где $2^m - 1$ — максимальное значение функции $f(X)$. Поэтому емкость ПЗУ будет равна $2^n \log_2 (\Delta f_{\max}) < 2^n m$.

Еще один метод состоит в записи в ПЗУ значений функции в узловых точках с последующим уточнением ее значений, например, по формуле

$$f(X) = f(X_i) + \Delta f(X_i, \Delta X),$$

где $f(X_i)$ — значение функции в узловой точке X_i , $\Delta X = X - X_i$; $\Delta f(X_i, \Delta X)$ — приращение функции. Обычно узловая точка X задается старшими разрядами операнда X , а его младшие разряды определяют приращение ΔX .

На рис. 4.60, а приведен график функции $f(X)$, иллюстрирующий такой способ вычисления ее значений, и структурные схемы двух возможных вариантов построения вычислительного устройства. Схему на рис. 4.60, б целесообразно применять в том случае, когда вели-

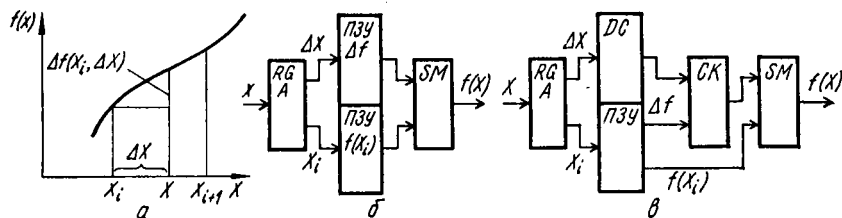


Рис. 4.60

чина $\Delta f(X_i, \Delta X)$ не зависит от X_i . В ПЗУ на рис. 4.60, в в одной ячейке хранятся как значения $f(X_i)$, так и все возможные приращения $\Delta f(X_i, \Delta X)$. Выбор нужного приращения $\Delta f(X_i, \Delta X)$ осуществляется с помощью селектора СК и дешифратора ДС, управляемых младшими разрядами аргумента, где записано приращение ΔX .

Если считать, что для представления абсциссы узловой точки X_i отводится n_1 разрядов, приращения ΔX — n_2 разрядов, значения $f(X_i)$ — m_1 разрядов, а приращения Δf — m_2 разрядов, причем $n_1 + n_2 = n$ и $m_1 + m_2 = m$, то суммарная емкость ПЗУ для вычислительного устройства на рис. 4.60, б будет равна $2^{n_1}m_1 + 2^{n_2}m_2$ бит, что в $\frac{2^{n_1}m_1}{2^{n_1}m_1 + 2^{n_2}m_2}$ раз меньше, чем при традиционной таблич-

ной реализации зависимости $f(X)$. В ПЗУ вычислительного устройства по схеме на рис. 4.60, в длина выходного слова равна $m_1 + 2^{n_2}m_2$, так как здесь одновременно считается значение $f(X_i)$ и 2^{n_2} значений Δf . Поэтому емкость (в битах) ПЗУ равна

$$\begin{aligned} 2^{n_1}(m_1 + 2^{n_2}m_2) &= 2^{n_1}m_1 + 2^{n_1+n_2}m_2 = 2^{n_1}m_1 + 2^{n_2}m_2 = \\ &= 2^n(2^{n_1-n_2}m_1 + m_2) = 2^n(2^{-n_2}m_1 + m_2) < 2^n m. \end{aligned}$$

Отсюда следует, что емкость ПЗУ в рассматриваемом вычислительном устройстве всегда меньше емкости ПЗУ в обычном табличном вычислителе, если только $n_2 \geq 2$.

При хранении в ПЗУ, кроме значений $f(X_i)$ еще и значений производных $f'(X_i)$, уточнение значений функции можно производить по формуле

$$f(X) = f(X_i) + f'(X_i) \Delta X.$$

Соответствующая схема организации вычислительного устройства показана на рис. 4.61. Здесь старшие разряды аргумента X_i используют для выборки из ПЗУ значений $f(X_i)$ и $f'(X_i)$ записанных в одной и той же ячейке, а младшие разряды (ΔX) — для формирования приращения $f'(X) \Delta X$ с помощью блока умножения БУ. Окончательно результат вычислений получают на сумматоре SM . Будем считать, что функция $f(X)$ при заданных $f(X_i)$ и $f'(X_i)$ может быть восстановлена с достаточной точностью для l значений аргумента, т. е. $\max(\Delta X) = l$ и $i = 1, 2, \dots, \frac{2^n}{l}$.

Если $f(X_i)$ и $f'(X)$ представлены m -разрядными словами, то для хранения их в $\frac{2^n}{l}$ узловых точках необходимо ПЗУ емкостью $\frac{2^n}{l} 2m$

бит. Следовательно, для того чтобы емкость ПЗУ в вычислительном устройстве по схеме на рис. 4.61 была меньше емкости ПЗУ в обычном табличном

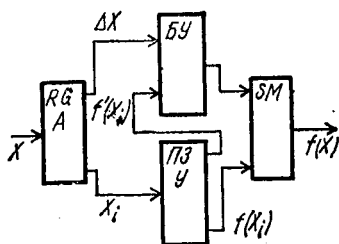


Рис. 4.61

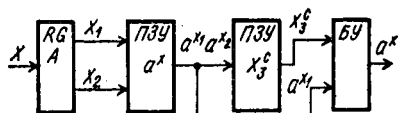


Рис. 4.62

вычислительном устройстве, необходимо выполнение условия

$$\frac{2^n}{l} 2m < 2^n m,$$

т. е. $l > 2$.

Поскольку в рассматриваемых устройствах значение $f(X)$ получают путем выполнения определенных операций (например, сложения, умножения и т. п.) с операндами, взятыми из таблиц некоторых функций, записанных в ПЗУ, то подобные вычислительные устройства называют таблично-алгоритмическими (таблично-аналитическими).

Рассмотрим еще один метод уменьшения емкости ПЗУ в таблично-алгоритмических вычислительных устройствах (ТАВУ), основанный на разбиении входного слова и реализации функциональной зависимости по частям. Основную идею метода покажем на примере построения ТАВУ для функции $f(X) = a^X$. Операнд X можно записать как

$$X = \sum_{i=1}^n 2^{i-1} x_i = \sum_{i=1}^{0,5n} 2^{i-1} x_i + \sum_{j=0,5n+1}^n 2^{j-1} x_j = \sum_{i=1}^{0,5n} 2^{i-1} x_i + 2^{0,5n} \sum_{j=0,5n+1}^n 2^{j-0,5n-1} x_j.$$

Обозначим

$$\sum_{i=1}^{0,5n} 2^{i-1} x_i = X_1, \quad \sum_{j=0,5n+1}^n 2^{j-0,5n-1} x_j = X_2, \quad 2^{0,5n} = C = \text{const}.$$

Тогда

$$a^X = a^{X_1 + CX_2} = a^{X_1} (a^{X_2})^C = a^{X_1} (X_3)^C.$$

Таким образом, вычисление a^X сводится к вычислению функций такого же вида, но с операндами X_1 и X_2 половинной длины, возведению в постоянную степень C и умножению. Для хранения функций a^{X_1} и a^{X_2} можно использовать одно и то же ПЗУ, длина входного слова которого составляет $0,5n$. Выборку значений a^{X_1} и a^{X_2} можно производить последовательно, либо параллельно по двум каналам (по двум адресам). Структура ТАВУ для этих случаев показана на рис. 4.62. Можно использовать также два одинаковых ПЗУ с длиной входного слова $0,5n$. В первых двух случаях суммарная емкость ПЗУ составляет $2^{0,5n}m + 2^m m$ бит, а накопители рассматриваемого ТАВУ будут проще накопителей табличного вычислителя, если выполнено условие

$$2^{0,5n} + 2^m < 2^n.$$

В третьем случае емкость всех накопителей ТАВУ составит $2^{0,5n+1}m + 2^m m$ бит, по отношению же к табличному вычислителю емкость уменьшится в $\frac{2^n}{2^{0,5n+1} + 2^m}$ раз. Заметим, что блок умножения в схеме на рис. 4.62 также может быть реализован таблично, для чего требуется $2^{2m}m$ бит. Тогда общая емкость накопителей ТАВУ будет равна $2^{0,5n+1}m + 2^m m + 2^{2m}m$ бит. Потребовав, чтобы

$$2^{0,5n+1} + 2^{2m} + 2^m < 2^n,$$

получим, что такой метод воспроизведения функции a^X обеспечивает уменьшение емкости накопителя по сравнению с традиционным табличным методом при любом $n \geq 4$ и $m < 0,5n$.

Выше рассмотрен случай разбиения входного слова X только на две части X_1 и X_2 . Эффект упрощения накопителя ТАВУ усиливается с ростом числа разрядов n и числа частей, на которые разбивается аргумент. Естественно, что при этом увеличивается время вычисления функциональной зависимости, а также сложность схем управления, коммутации и вычисления окончательного значения $f(X)$. Поэтому вопросы выбора структуры ТАВУ, числа частей разбиения входного слова и т. п. необходимо решать применительно к конкретным функциям $f(X)$, так, чтобы обеспечивались заданные точность и скорость вычислений, а выигрыш в аппаратных затратах за счет упрощения накопителей был не меньше проигрыша, обусловленного введением указанного дополнительного оборудования. Вообще метод разбиения входного слова и реализации функциональной зависимости по частям применим к функциям, допускающим их представление в виде

$$f(X) = F(\varphi_1(X_1), \varphi_2(X_2), \dots, \varphi(X_p)),$$

где $X_1, X_2, \dots, X_p$ — части входного слова. Функции $\varphi_i(X_i)$ в ТАВУ реализуются табличным способом, а функция F — специальным вычислительным устройством.

Ряд интересных ТАВУ можно построить при разбиении входного слова на n частей. Например, для функции a^X имеем

$$a^X = a^{x_1} a^{2x_2} a^{4x_3} \dots a^{2^{n-1}x_n},$$

где $X = \sum_{i=1}^n x_i 2^{i-1}$. Поскольку $x_i \in \{0, 1\}$, то в составе ТАВУ достаточно иметь ПЗУ из n ячеек, хранящих константы $a^1, a^2, a^4, \dots, a^{2^{n-1}}$, причем отдельные разряды x_i входного слова могут непосредственно воздействовать на адресные шины (без дешифратора) и управлять передачей содержимого ячеек на блок умножения БУ (рис. 4.63, а, $n = 4$). При $x_i = 0$ из соответствующей ячейки на БУ необходимо выдать константу 1. Для этого в каждом j -м разряде ячеек ПЗУ долж-

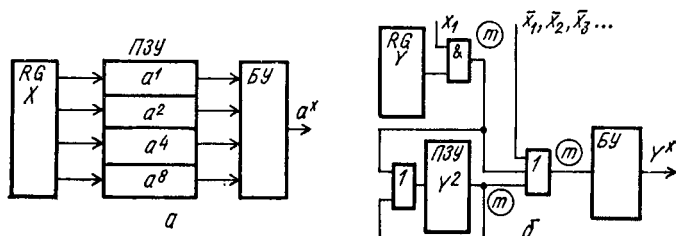


Рис. 4.63

на выполняться операция $\bar{x}_i \vee a_{ij}$, где a_{ij} — цифра в j -м разряде ($j = 1, 2, \dots, m$) константы $a_i = a^{2^{i-1}}$. Таким образом, емкость ПЗУ здесь равна m бит, время вычисления же определяется структурой БУ.

При последовательном поступлении разрядов x_i входного слова функции вида Y^X можно вычислять с помощью ТАВУ, схема которого приведена на рис. 4.63, в. В основу построения такого ТАВУ положено представление функции Y^X в виде

$$Y^X = Y^{x_1} Y^{2x_2} Y^{4x_3} \dots Y^{2^{n-1}x_n},$$

причем функции $Y^2, Y^4, Y^8, \dots$ генерируются с помощью одного ПЗУ для функции Y , так как $Y^4 = (Y^2)^2, Y^8 = (Y^4)^2$ и т. д. На первом шаге вычислений здесь на БУ передается Y^{x_1} , равное Y или 1 — в зависимости от значения x_1 . Одновременно Y поступает на вход ПЗУ Y^2 , из которого выбирается Y^2 . На втором шаге вычислений на БУ передается Y^{2x_2} и из ПЗУ выбирается $(Y^2)^2$. На третьем шаге на БУ поступает Y^{4x_3} и т. д. Емкость ПЗУ в таком ТАВУ равна $2^m m$ бит. При табличной реализации функции Y^X потребовалось бы ПЗУ емкостью $2^{2^n} m$ бит.

В заключение рассмотрим еще один пример построения ТАВУ для функции $f(X) = X^2$. Запишем эту функцию в следующем виде:

$$X^2 = \left(\sum_{i=1}^n x_i 2^{i-1} \right)^2 = \left(\sum_{i=1}^{0,5n} x_i 2^{i-1} + \sum_{j=0,5n+1}^n x_j 2^{j-1} \right)^2 =$$

$$\begin{aligned}
&= \left(\sum_{i=1}^{0,5n} x_i 2^{i-1} + 2^{0,5n} \sum_{j=0,5n+1}^n x_j 2^{j-1-0,5n} \right)^2 = (X_1 + 2^{0,5n} X_2)^2 = \\
&= X_1^2 + 2^n X_2^2 + 2^{0,5n+1} X_1 X_2,
\end{aligned}$$

где $X_1 = \sum_{i=1}^{0,5n} x_i 2^{i-1}$; $X_2 = \sum_{j=0,5n+1}^n x_j 2^{j-1-0,5n}$.

Для хранения функций X_1^2 и X_2^2 можно использовать одно и то же ПЗУ с длиной входного слова в $0,5n$ разрядов, а $X_1 X_2$ вычислять с помощью блока умножения. Заметим, что блок умножения в этом случае предназначен для выполнения операций с $0,5n$ -разрядными числами и значительно проще, чем блок, используемый для умножения $XX = X^2$. В ТАВУ для функции X^2 операнды X_2^2 и $X_1 X_2$ должны подаваться на сумматор со сдвигом влево на n и $0,5n + 1$ разрядов относительно X_1^2 . Емкость ПЗУ в такой схеме составляет $2^{0,5n} m$ бит, что в $2^{0,5n}$ раз меньше, чем в случае табличной реализации рассматриваемой функции.

Рассмотренные примеры не исчерпывают всех возможных применений ПЗУ, однако показывают их высокую универсальность.

Любая схема, для построения которой используется ПЗУ, может быть изготовлена в виде БИС. Однако это экономически оправдано лишь при большой серийности БИС. Кроме того, логическая структура БИС хотя и будет содержать меньшее число элементов, но может оказаться не регулярной и трудно реализуемой. Иначе обстоит дело с ПЗУ, имеющими очень простую (дешифратор — шифратор), регулярную структуру, удобную для реализации в виде БИС. Высокая универсальность ПЗУ влечет за собой большую серийность их выпуска и низкую стоимость.

4.10. АССОЦИАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

В настоящее время наибольшее распространение получили адресные ЗУ. Однако во многих случаях адресную организацию поиска слов нельзя считать ни самой удобной, ни самой естественной. Действительно, при решении информационно-логических задач требуется выбирать слова по их признакам. Например, при распределении слов в алфавитном порядке необходимо сначала выбрать слова, начинающиеся на букву «а», затем — начинающиеся на букву «б» и т. д. Для этого надо многократно пересмотреть содержимое всех ячеек ЗУ. При решении обычных вычислительных задач много времени и усилий программистов затрачивается на распределение данных в ячейки адресного ЗУ и последующий учет этого распределения. Кроме того, насколько известно в настоящее время, поиск информации в человеческой памяти осуществляется по ее содержанию.

Все это привело к созданию так называемых ассоциативных ЗУ (АЗУ), в которых слова отыскивают по их признакам, т. е., по их содержанию, а не по адресам ячеек, в которых записаны эти слова. Так же как и адресные, ассоциативные ЗУ могут быть с М- и В-поиском.

Выборка слов из АЗУ может происходить при выполнении следующих условий:

- 1) признак искомого слова P_n равен заданному признаку P_z , $P_n = P_z$;
- 2) $P_n > P_z$;
- 3) $P_n \geq P_z$;
- 4) $P_{z\min} \leq P_n \leq P_{z\max}$, где $P_{z\min}$ и $P_{z\max}$ предельные значения признака P_z и т. д.

Слова можно также выбирать при выполнении условий, являющихся отрицаниями вышеприведенных. Если два и более слов удовлетворяют заданному признаку, то получается так называемый многозначный ответ. При многозначном ответе выборку слов осуществляют по вторичным признакам или в порядке физических номеров ячеек. Поиск слов по равенству P и P_z называется простым поиском, а все другие виды поиска называются сложными.

Информация в АЗУ может записываться с помощью следующих способов: запись по адресу; запись по признаку; запись в первую свободную ячейку (шаговая запись) и запись с сортировкой информации. Первый способ аналогичен записи слов в адресном ЗУ. При записи по признаку нужную ячейку находят по ассоциативному признаку слова, записанному в ней. Шаговая запись является наиболее распространенным способом. При использовании этого способа производится учет всех свободных ячеек АЗУ и вновь поступающая информация записывается в первую из них. Запись с сортировкой состоит в упорядоченном расположении всей записываемой информации в соответствии с ее числовым значением. Такая запись облегчает упорядоченную выборку слов, например, в порядке возрастания ассоциативного признака.

Жесткие требования к ЗЭ АЗУ, а именно: неразрушающее считывание и необходимость в выполнении логической операции сравнения, привели к тому, что в настоящее время распространение получили только АЗУ небольшой емкости (порядка 100 ячеек), построенные на транзисторных триггерах, объединенных в регистры. В качестве примера рассмотрим АЗУ на триггерах, с простым поиском и записью по адресу (рис. 4.64). Чтение начинается с приема на регистр RGP_z заданного признака P_z . Этот признак опроса с помощью M схем равенства слов (СРС) сравнивается с M ассоциативными признаками слов (АП), записанными в регистрах $RGAP$. Все регистры признаков в совокупности называют ассоциативным накопителем. Если в АЗУ есть слово с заданным признаком, т. е. $P_z = P_n$, то возбуждается соответствующая адресная шина чтения (АШЧ) информационного накопителя H и в регистр слов записывается информационная часть слов. В противном случае формируется сигнал $K2$, свидетельствующий о том, что слова с заданным признаком P_z в АЗУ нет. Запись осуществляется так же, как и в адресном ЗУ.

АЗУ рассмотренной структуры находят применение в механизмах адресации цифровых ЭВМ, в системах защиты памяти, динамического распределения памяти, в сверхоперативных ЗУ. Однако возможности таких АЗУ ограничены, так как накопители (ассоциативный и инфор-

мационный) в них разделены и нельзя использовать любую часть слова в качестве ассоциативного признака. Создание АЗУ с объединенным накопителем стало возможным лишь с освоением выпуска интегральных накопителей. Строят такие накопители на биполярных и униполярных транзисторах, причем структура запоминающего элемента одинакова в обоих случаях (рис. 4.65, а). Здесь схема равнозначности реализует функцию $\omega_i = x_i \bar{y}_i \vee z_i y_i$, x_i и z_i — сигналы по РШО и РШП; y_i — переменная, характеризующая состояние триггера; ШП — шина поиска. Выходы отдельных схем равнозначности должны быть подключены к шине поиска так, чтобы при этом реализовалась конъюнкция $\omega = \bigwedge_{i=1}^n \omega_i$. Из-за этого $\omega = 1$ только при равенстве x_i и y_i по всем разрядам. В накопителе на таких ЗЭ может быть реализован также и адресный поиск информации.

Ассоциативный поиск слов, в которых в i -м разряде находится 1

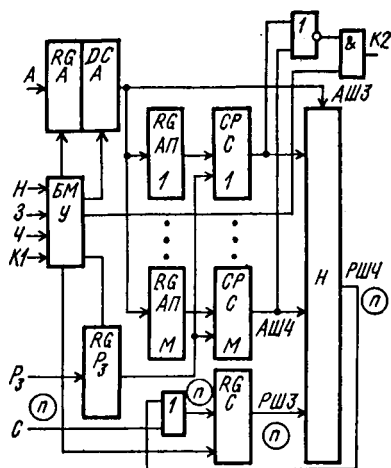


Рис. 4.64

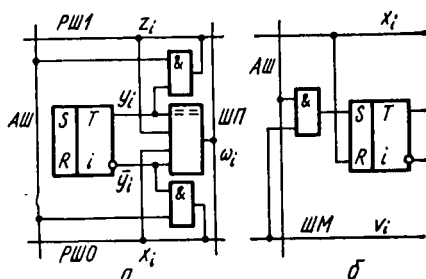


Рис. 4.65

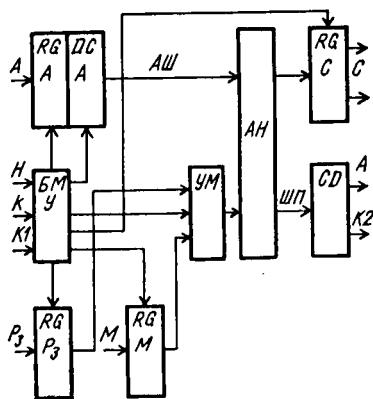


Рис. 4.66

(0), осуществляется подачей сигналов $z_i = \bar{x}_i = 1$ (0). Заметим, что ассоциативный поиск имеет смысл только в том случае, когда длина ассоциативного признака не равна длине искомого слова. В противном случае ассоциативный поиск укажет только на наличие или отсутствие в накопителе слова, представляющего собой заданный признак P_3 . Для организации поиска слова по его части другую часть необходимо из поиска исключить (маскировать).

Достигается это подачей сигналов $x_i = z_i = 1$ в маскируемых разрядах. В этом случае $\omega_i = 1 y_i \vee 1 \bar{y}_i = 1$ и маскируемый разряд не влияет на формирование сигналов ω .

В АЗУ часто необходимо изменять содержимое отдельных разрядов слов без изменения других разрядов. Осуществить это можно путем введения шины маскирования (ШМ), запрещающей запись в ЗЭ при возбужденной адресной шине (рис. 4.65, б). Введение этой шины не приводит к увеличению числа шин в накопителе, так как при ее наличии можно отказаться от одной из разрядных шин. Опрос i -го разряда по 1, 0 и маскирование в этом случае осуществляются за счет некоторого усложнения реализации функции ω_i , а именно:

$$\omega_i = \overline{x_i y_i} \vee x_i \overline{y_i} \overline{v_i}.$$

Режим поиска 0 обеспечивается подачей сигналов $x_i = v_i = 0$, режим поиска 1 — подачей сигналов $x_i = 1$, $v_i = 0$, маскирование — сигналами $x_i = v_i = 1$.

Структурная схема АЗУ с объединенным накопителем представлена на рис. 4.66. АЗУ состоит из ассоциативного накопителя АН, узла маскирования УМ, шифратора СД, регистров заданного признака RGP_3 и маски RGM . Здесь возможно выполнение следующих операций, задаваемых кодом k : чтение и запись по адресу, чтение и запись по признаку, запись в пустую ячейку. Первые две операции реализуются так же, как и в адресном ЗУ — с использованием RGA , ДСА, АН и RGC . При ассоциативном выполнении операций чтения и записи заданный признак P_3 поступает в RGP_3 и с помощью УМ маскируется словом, находящимся в RGM . По сигналу H маскированный ассоциативный признак (MP_3) поступает в АН. При наличии в АН слова с заданным MP_3 возбуждается одна из шин поиска (ШП) и на выходе шифратора СД появляется адрес ячейки с искомым словом. Этот адрес поступает далее в RGA , после чего выполняется операция записи или чтения, как в обычном адресном ЗУ. Если же слова с заданным MP_3 нет, то шифратор СД формирует сигнал $K2$. Если выполняется операция записи в пустую ячейку, то в качестве заданного ассоциативного признака используют нуль (маскирование в этом случае отсутствует) и в найденную таким образом ячейку записывают слово с RGC . Чтобы отличить пустую ячейку от ячейки с нулевым словом, АН должен содержать маркерные разряды занятости ячеек. В случае многозначного ответа несколько входных шин шифратора будут возбуждены одновременно. Поэтому шифратор должен обладать свойством приоритетного кодирования адреса, например, при возбуждении двух и более входных шин (ШП) шифратор должен формировать адрес ячейки с наибольшим номером. Если число входов шифратора равно 8 ($x_7 — x_0$), а выходов 3 (y_3, y_2, y_1), то функции, реализуемые таким шифратором, имеют вид:

$$y_1 = x_7 \vee \overline{x_6} (x_5 \vee x_4 (x_3 \vee \overline{x_2} x_1));$$

$$y_2 = (x_7 \vee x_6) \vee \overline{(x_5 \vee x_4)} (x_3 \vee x_2);$$

$$y_1 = x_7 \vee x_6 \vee x_5 \vee x_4.$$

При использовании такого шифратора выборка слов в случае многозначного ответа будет происходить по очереди. Для ее реализации необходимо иметь в каждой ячейке АЗУ специальный служебный разряд, в который после выборки слова записывается единица. Так

ак в служебных разрядах еще невыбранных ячеек будут нули, то и обеспечит их последовательную выборку. По окончании чтения всех слов многозначного ответа служебные разряды устанавливаются нуль. В описанном АЗУ реализована только операция простого поиска. Поэтому, если необходимо выполнить сложный поиск, то его надо представить как последовательность операций простого поиска. Например, поиск наименьшего слова в накопителе может быть выполнен в следующей последовательности. Первый поиск осуществляют по $P_3 = 0xx...x$, где x — маскируемые разряды. Если слово с таким признаком есть в АЗУ, то далее осуществляют поиск по признаку $P_3 = 00xx...x$, в противном случае поиск ведут по признаку $P_3 = 10xx...x$ и т. д. (табл. 66, $n = 3$). Из таблицы видно, что после n -го или $(n + 1)$ -го опроса содержимого АЗУ на выходе шифратора будет получен адрес наименьшего слова. Само слово может быть считано с АН или RGP_3 . Подобным же образом могут быть реализованы другие операции сложного поиска.

Таблица 66

Номер опроса	Признаки поиска и ответы									
1	P_3	0xx								
	Ответ	Да (слово с таким P_3 есть в АЗУ)					Нет (слова с таким P_3 нет в АЗУ)			
2	P_3	00x				10x				
	Ответ	Да		Нет		Да		Нет		
3	P_3	000		010		100		110		
	Ответ	Да (К)	Нет	Да (К)	Нет	Да (К)	Нет	Да (К)	Нет	
4	P_3		001		011		101		111	
	Ответ		Да (К)		Да (К)		Да (К)		Да (К)	

4.11. ОБРАБОТКА ИНФОРМАЦИИ В ЗУ

При решении информационно-логических задач значительную часть машинного времени цифровых ЭВМ занимает выполнение логических операций. Одним из эффективных способов повышения производительности цифровых ЭВМ в таком случае является реализация логических операций непосредственно в ЗУ. Это становится возмож-

ным благодаря свойству запоминающих элементов ЗУ выполнять поразрядную дизъюнкцию слов, записанных в данную ячейку, без предварительной ее очистки. Поскольку на регистре слова (*RGC*) можно получать поразрядное отрицание, а дизъюнкция и отрицание являются функционально полной системой операций, то непосредственно в ЗУ можно выполнить любую поразрядную логическую операцию. ЗУ, у которых реализована такая возможность, называют логическими ЗУ. Любое ЗУ, запоминающие элементы в котором обладают указанным свойством, можно преобразовать в логическое. Для этого достаточно внести такие изменения в его схему, чтобы наряду с обычной записью слов можно было осуществить запись в ячейку без предварительной ее очистки. Такая операция протекает примерно

в два раза быстрее по сравнению с обычной записью, что способствует росту быстродействия цифровых ЭВМ. Особенно это сказывается при выполнении групповых логических операций.

Таблица 67

Операция в ЗУ	Адрес	Содержимое		
		<i>RGC</i>	α	β
Исходное состояние		0	X	Y
Чтение	α	X	X	Y
Запись И	α	0	$\bar{X}$	Y
Чтение	β	Y	$\bar{X}$	Y
Запись И	α	0	$\bar{X} \vee \bar{Y}$	Y
Чтение	α	$\bar{X} \vee \bar{Y}$	$\bar{X} \vee \bar{Y}$	Y
Запись И	β	0	$\bar{X} \vee \bar{Y}$	$\bar{X} \vee \bar{Y}$

Например, пусть необходимо выполнить поразрядную дизъюнкцию над словом, написанным в ячейке α , и каждым словом, записанным в ячейках $\beta_1, \beta_2, \dots, \beta_n$. Для этого достаточно выбрать слово из ячейки α и записать его в ячейки $\beta_1, \beta_2, \dots, \beta_n$ без предварительной их очистки. Для выполнения этой операции непосредственно в ЗУ необходи-

мо одно полное обращение к ячейке α и n полуобращений (т. е. записи без очистки) к ячейкам β_i . Следовательно, длительность этой операции равна $t_0 \left(1 + \frac{n}{2}\right)$. При осуществлении этой же операции в АЛУ необходимо $n(2t_0 + t_1)$ единиц времени, где t_1 длительность операции поразрядной дизъюнкции в АЛУ. При $t_0 \approx t_1$ и $n \gg 1$ время выполнения операции в первом случае почти в 6 раз меньше, чем во втором. Выигрыш во времени будет еще значительнее, если учесть, что АЛУ освобождается для других операций.

Процедура выполнения в ЗУ других поразрядных логических операций несколько сложнее, чем для операции дизъюнкции. Например, реализация поразрядной конъюнкции по формуле $\bar{X} \vee \bar{Y}$ (табл. 67) требует 5,5 t_0 единиц времени. В табл. 67 «Запись И» означает операцию записи в ЗУ без предварительной очистки инверсного содержимого регистра слов. Выполнение операций в ЗУ можно значительно ускорить, если имеется возможность на *RGC* принимать инверсные слова. В этом случае поразрядная конъюнкция требует трех обращений к ЗУ (выборка $\bar{X}$ на *RGC*, выборка $\bar{Y}$ на *RGC*, запись $\bar{X} \vee \bar{Y}$ инверсным кодом в накопитель).

Любую, в том числе и арифметическую операцию, в ЗУ можно осуществить, если имеется возможность записывать слова в ЗУ со сдвигом на один разряд влево и вправо. Например, суммирование n -разрядных чисел X и Y может быть реализовано в соответствии со следующим алгоритмом:

1) сформировать поразрядные суммы, т. е. выполнить операцию $S = \overline{X \vee Y \vee X \vee \bar{Y}}$;

2) сформировать поразрядные переносы $P = \overline{\overline{X \vee \bar{Y}}}$;

3) если $\bigvee_{i=1}^n p_i = 0$, то закончить суммирование, если же $\bigvee_{i=1}^n p_i = 1$, то перейти к п. 4);

4) записать S в ячейку ЗУ вместо операнда X ;

5) записать P в ячейку ЗУ вместо операнда Y со сдвигом на один разряд влево;

6) повторить п. п.1) — 3) алгоритма.

Такое выполнение суммирования иллюстрируется табл. 68, где «Чтение I » означает операцию чтения инверсного слова, а «Запись C_2 » — запись со сдвигом записываемого слова на один разряд влево (без предварительной очистки).

Таблица 68

№ п. п.	Операция в ЗУ	Адрес	Содержимое ячеек				
			RGC	α	β	γ	δ
	Исходное состояние		0	X	Y		
1	Чтение I	α	$\bar{X}$	X	Y		
2	Чтение	β	$\bar{X} \vee Y$	X	Y		
3	Запись I	γ	0	X	Y	$\overline{\bar{X} \vee Y}$	
4	Чтение I	β	$\bar{Y}$	X	Y	$\overline{\bar{X} \vee Y}$	
5	Чтение	α	$\bar{Y} \vee X$	X	Y	$\overline{\bar{X} \vee Y}$	
6	Запись I	γ	0	X	Y	S	
7	Чтение I	α	$\bar{X}$	X	Y	S	
8	Чтение I	β	$\bar{X} \vee \bar{Y}$	X	Y	S	
9	Запись I	δ	0	X	Y	S	P
10	Чтение	γ	S	X	Y	S	P
11	Запись	α	0	S	Y	S	P
12	Чтение	δ	P	S	Y	S	P
13	Запись C_2	β	0	S	2P	S	P
14	Чтение	δ	P	S	2P	S	P
15	$\bigvee p_i = 0$	Конец операции					
16	$\bigvee p_i = 1$	Переход к операции 1					

Так как любая другая арифметическая операция может быть сведена к последовательности операций сдвига и суммирования, то приведенный пример показывает возможность реализации всех функций АЛУ в логическом ЗУ. Поскольку логическое ЗУ может хранить и микропрограммы отдельных операций, то на его основе, в принципе, может быть построена весьма простая одноадресная цифровая ЭВМ. Таким образом, незначительные изменения в регистре слов, блоке управления и накопителе ЗУ позволяют реализовать ряд логических и арифметических операций непосредственно в ЗУ, что повышает производительность ЭВМ особенно при решении задач с большим числом групповых поразрядных логических операций.

4.12. ПРИНЦИПЫ ПОСТРОЕНИЯ И РАЗНОВИДНОСТИ ЗУ С МАГНИТНОЙ ЗАПИСЬЮ

Запоминающие устройства с магнитной записью (МЗ) называют также электромеханическими. В настоящее время ЗУ с МЗ занимают доминирующее положение при построении ЗУ большой емкости (10^8 — 10^{11} бит) и невысокой стоимости. Все ЗУ с МЗ имеют специальные магнитные головки, с помощью которых информация записывается путем соответствующего намагничивания поверхности магнитного носителя. Считывание информации также осуществляется с помощью магнитных головок. Когда под считывающей головкой проходит намагниченный участок поверхности магнитного носителя, в обмотке считывающей головки наводится э.д.с., которая фиксируется в соответствии с принятым способом отображения «0» и «1». Чтение при этом происходит без разрушения.

Магнитная поверхность носителя представляет собой лак, содержащий частицы ферритового порошка, например, Fe_2O_3 . Толщина ферролака находится в пределах 2—20 мкм. Материал магнитного покрытия должен быть магнитно-твердым, т. е. иметь большое значение коэрцитивной силы и остаточной индукции, а также иметь петлю гистерезиса с высоким коэффициентом прямоугольности. Сердечники магнитных головок изготавливают из магнитно-мягких материалов. Материал сердечников должен иметь большую начальную магнитную проницаемость (чтобы головка воспринимала слабые поля при чтении), высокое значение индукции насыщения (чтобы сердечник головки не насыщался при насыщении носителя во время записи), малую величину остаточной индукции (чтобы головка не создавала сильного поля при отсутствии тока в ее обмотке). Одни и те же головки, как правило, используются и для записи и для чтения.

Одной из важнейших характеристик МЗ является ее плотность

$$\delta = \gamma\sigma, \text{ бит/см}^2, \text{ } \delta$$

где γ и σ — продольная и поперечная плотности МЗ. К настоящему времени в технике магнитной записи достигнуты следующие предельные значения: $\gamma = 3 \cdot 10^8$ бит/см, $\sigma = 400$ бит/см. Теоретический предел этих характеристик составляет 10^5 бит/см.

Для записи информации на магнитный носитель используют следующие наиболее известные методы

1. Запись по трем уровням осуществляется разнополярными импульсами тока малой длительности (1—2 мкс) на предварительно размагниченный носитель (рис. 4.67). Импульс тока положительной полярности в головке (ТГ) соответствует «1», а отрицательной — «0». Состояние носителя (СН) при этом будет соответствовать $+B_r$ и $-B_r$. При считывании в головке наводится э.д.с.

$$e = \alpha_1 \frac{dB}{dt} = \alpha_2 v \frac{dB}{dt},$$

где v — скорость движения носителя.

2. Запись по двум уровням осуществляется путем намагничивания носителя до уровня $-B_r$ и кратковременного перемагничивания в состояние $+B_r$.

при записи «1» (рис. 4.67). Так как ток в головке по форме совпадает с состоянием носителя, то диаграммы ТГ и СН совмещены.

3. Запись с переключением потока при записи «1» осуществляется с помощью триггера со счетным входом, который изменяет направление тока в головке на противоположное при записи «1» (рис. 4.67).

4. Запись с переключением потока при изменении цифр осуществляется так, что изменение тока в головке происходит только при изменении цифр от «1» к «0» или от «0» к «1» (рис. 4.67).

5. Запись с фазовой модуляцией осуществляется путем изменения намагниченности носителя от $-B_r$ до $+B_r$ при записи «0» и от $+B_r$ до $-B_r$ при записи «1» (рис. 4.67). В момент действия СИ намагниченность всегда меняет знак. В промежутке между СИ намагниченность меняет знак только тогда, когда записываются две одинаковые цифры.

6. Запись с частотной модуляцией характеризуется удвоением частоты изменения знака намагниченности при записи «1» (рис. 4.67). Под действием СИ намагниченность меняет знак. При записи же «1» намагниченность меняет знак еще раз через время $\frac{1}{2f}$, где f — частота СИ.

Чтение информации осуществляется схемами, принцип работы которых позволяет выявлять особенности воспроизведения «0» и «1» при данном способе записи.

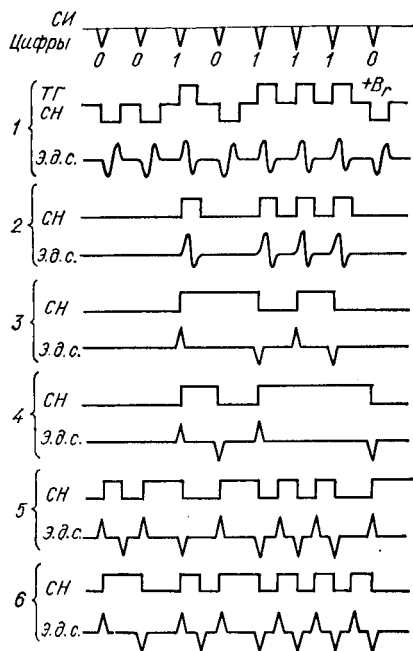


Рис. 4.67

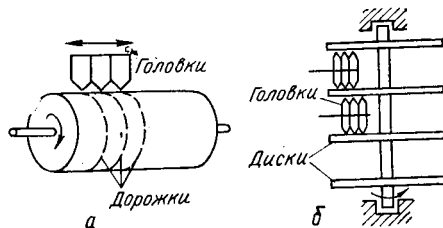


Рис. 4.68

Первый и второй методы записи нашли ограниченное применение из-за низкой плотности записи. Наибольшее распространение получили третий и четвертый методы. При их использовании частота изменения знака намагниченности в два раза меньше по сравнению с первым и вторым методами. Поэтому плотность записи по этим способам может быть выше почти в два раза. Однако у этих методов есть два существенных недостатка, снижающих помехоустойчивость ЗУ. Во-первых, не все цифры имеют свой «отпечаток» на носителе (например, «0» при третьем способе и группы цифр 11...1 и 00...0 при четвертом). Поэтому, например, при третьем способе малая амплитуда тока, соответствующего «1», может быть воспринята как «0», а помеха при чтении «0» — как 1. Во-вторых, имеют место большие амплитудные и фазовые искажения из-за широкого спектра считываемых сигналов и ограниченной полосы пропускания УЧ. Например, при четвертом способе частота считываемых сигналов может меняться от 0 (при чтении группы одинаковых цифр) до f (при чтении слова 010101...01).

Отмеченные недостатки отсутствуют у пятого и шестого методов. Амплитудные и фазовые искажения устраняются за счет ограниченного спектра (f , $2f$) считываемых сигналов. Кроме того, возможна самосинхронизация (т. е. получение СИ путем считывания их с любой информационной дорожки), так как независимо от записанной цифры в момент действия СИ намагниченность носи-

теля меняет знак. Вследствие этого пятый и шестой методы МЗ получили преимущественное распространение в настоящее время.

Конструктивно ЗУ с МЗ выполняются в виде магнитных барабанов (МБ), магнитных дисков (МД), магнитных карт (МК) и магнитных лент (МЛ). ЗУ на МБ (рис. 4.68, а) представляет собой полый цилиндр из немагнитного вещества (алюминиевых сплавов или латуни), покрытый тонким слоем ферромагнитного материала. Диаметр МБ равен 100—800 мм. Скорость вращения МБ может достигать 70000 об/мин, емкость накопителей на МБ достигает $8 \cdot 10^9$ бит. Среднее время поиска информации в ЗУ на МБ составляет $5 \div 30$ мс (в лучших моделях — 1,25 мс).

Технические характеристики ЗУ на МБ типа ЕС-5033 приведены в табл. 69.

Таблица 69

Тип устройства	N, байт	Среднее время поиска	γ , бит/мм	σ , бит/мм	Скорость	Диаметр, мм	Число дорожек (головок)
ЕС-5033 (МБ)	6М	20 мс	37	2,2	1500 $\frac{\text{об}}{\text{мин}}$	450	800
ЕС-5050 (МД)	7,2М	75 мс	40	4	2400 $\frac{\text{об}}{\text{мин}}$	360	203
ЕС-5014 (МЛ)	50М	~3 мин	63	1,4	2 $\frac{\text{м}}{\text{с}}$	—	9

ЗУ на МД имеют значительно большую емкость по сравнению с МБ при том же физическом объеме. Дорожки записи на МД имеют форму концентрических окружностей, а рабочими являются обе поверхности МД (рис. 4.68, б). Между МД размещают магнитные головки, которые бывают как подвижными (перемещаются вдоль радиуса МД), так и неподвижными. Поскольку биение рабочих поверхностей МД значительно больше, чем у МБ, то здесь применяют только плавающие магнитные головки. Все головки пакета дисков монтируются на общей каретке, обеспечивающей перемещение головок с одной дорожки на другую. В ЗУ на МД время обращения состоит из двух компонент: времени поиска, определяемого быстродействием системы выбора нужной дорожки, и времени ожидания, равного, в среднем, половине времени одного оборота диска. Для различных конструкций ЗУ на МД время поиска составляет от 25 до 75 мс. Существуют ЗУ с закрепленными и сменными пакетами МД. ЗУ с закрепленными пакетами имеют емкость 300—3000 Мбайт. Емкость одного сменного пакета МД достигает 300 Мбайт. Сменные пакеты могут содержать 1, 5, 6, 11, или 12 дисков в зависимости от назначения ЗУ. Диаметр сменного диска равен 356,25 мм, толщина диска 1,9 мм. Указанные размеры дисков и их количество в пакете установлены Международной организацией по стандартизации (ИСО). Продольная плотность записи на сменные диски достигает 240 бит/мм.

На стандартном диске зона дорожек имеет ширину 50 мм, а число дорожек в ней может быть равно 203, 411 или 822. Вертикальный набор дорожек (по всем дискам) называют цилиндром. Дорожкам и цилиндрам присваивают номера, возрастающие по направлению к центру диска. Все дорожки имеют одинаковую емкость и частоту синхронизирующих сигналов. Так как линейная скорость движения носителя изменяется с изменением радиуса диска, то плотность записи на внешних дорожках будет меньше, чем на внутренних. Для лучшего использования носителя рабочую часть МД иногда разбивают на ряд зон, по несколько дорожек в каждой, а частоту синхронизирующих сигналов меняют при переходе от зоны к зоне. Это выравнивает плотность записи информации в различных зонах.

Технические характеристики ЗУ на МД типа ЕС-5050 приведены в табл. 69. Помимо металлических (жестких) дисков в ЗУ с магнитной записью используются также так называемые гибкие диски, выполненные на майларовой (лавсановой) основе (толщиной порядка 0,12 мм), на которую нанесено ферролаковое покрытие. Каждый диск помещен в кассету из гибкой пластмассы и имеющую окна для установки диска на ступицу привода и для подвода головок. При работе диск вращается внутри кассеты. Диаметр диска около 200 мм, а его емкость около 300 Кбайт. Для уменьшения трения внутреннюю поверхность кассеты покрывают специальным пластиком. Известны также и другие конструкции ЗУ на гибких МД — без кассет, с плавающими головками, с плавающими дисками и т. п.

ЗУ практически неограниченной емкости со сменным накопителем могут быть построены на магнитных картах. Магнитная карта представляет собой лист гибкой пленки, покрытой ферромагнитным слоем.

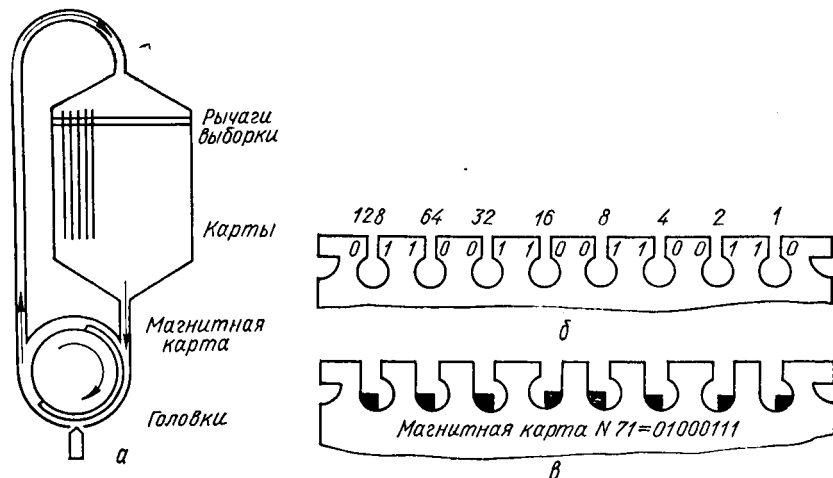


Рис. 4.69

Одна из конструкций такого ЗУ (рис. 4.69, а) состоит из магазина, содержащего оперативный комплект МК, и вращающегося барабана с вакуумными присосками. В магазине размещается до 256 МК размером 360 × 80 мм, подвешенных в произвольном порядке на двух удерживающих и восьми поворотных выборочных рычагах. Фигурные вырезы на картах (рис. 4.69, б) определяют их номера от 0 до 255. Одному из уступов выреза соответствует одно из значений 1, 2, 4, ..., 128, а другому — значение 0. Для присвоения карте номера отрезают соответствующие уступы. Выборочные рычаги в поперечном сечении представляют собой секторы, равные четверти круга. Каждый рычаг может поворачиваться на 90° и занимать два крайних положения, вследствие чего возможно 256 различных сочетаний положений выборочных рычагов. В любом из этих положений освобождается одна из 256 карт, а остальные удерживаются выборочными рычагами. На рис. 4.69, б показано, как удерживается МК № 71 при выборе карты № 70. Освобожденная МК падает в направляющий канал, из которого попадает на барабан и присасывается к нему. Барабан вращает карту неограниченное время, в течение которого производится запись и чтение слов. По окончании операции вакуум снимается и карта возвращается в магазин. Емкость подобных ЗУ без смены носителя доходит до $4 \cdot 10^6$ бит при среднем времени обращения порядка 0,2 с. Самым распространенным ЗУ большой емкости с магнитной записью является ЗУ на МЛ. Такие ЗУ позволяют создавать архивы данных, обмениваться информацией не только между отдельными ЭВМ, но и между вычислительными центрами. Различают ЗУ с катушками МЛ (рис. 4.70, а) и с петлями МЛ (рис. 4.70, б). В ЗУ на МЛ используется контактный способ записи и чтения слов, при котором МЛ скользит по поверхно-

сти магнитных головок, в то время как в ЗУ с магнитной записью других типов используется исключительно бесконтактный способ. В соответствии со стандартами ИСО магнитная лента может иметь ширину 3,81; 6,35 и 12,7 мм, причем лента шириной 3,81 и 6,35 мм используется в ЗУ кассетного типа. Современные конструкции лентопротяжных механизмов выполняют таким образом, чтобы исключить соприкосновение рабочего слоя ленты с какими-либо элементами механизма, кроме блока магнитных головок. Нерабочая поверхность ленты соприкасается с направляющими и охватывает ведущий ролик 1 (рис. 4.70, а). Для увеличения сцепления МЛ с поверхностью ролика иногда применяют

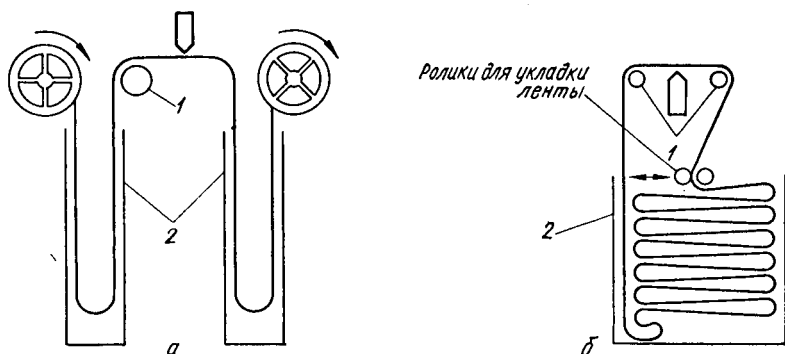


Рис. 4.70

ролики, внутри которых создается разрежение с помощью вакуумной системы. МЛ приводится в движение только при записи, чтении и поиске нужной информации, а в остальное время лента неподвижна. Чтобы не создавать механических перегрузок при быстром пуске и остановке МЛ, между катушками и ведущим роликом имеются буферные участки ленты, погруженные в вакуумные колонки 2. Кроме того, буферные участки ленты в колонках обеспечивают необходимую величину и постоянство натяжения ленты.

При использовании петель магнитной ленты последняя с помощью рычажной системы укладывается «змейкой» в бункер.

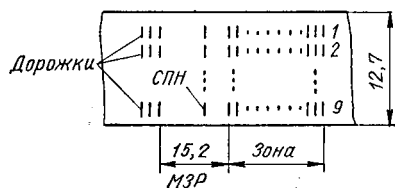


Рис. 4.71

Лентопротяжные механизмы обеспечивают движение ленты со скоростями от 0,9 до 6,3 м/с. Поэтому для стандартных катушек магнитной ленты среднее время поиска информации составляет не менее одной минуты. Для повышения быстродействия ЗУ на МЛ в некоторых их конструкциях применяют ускоренную обратную перемотку МЛ со скоростью 10—20 м/с. Продольная плотность записи на МЛ зависит от используемого способа магнитной записи и достигает 246 бит/мм.

Информация на ленте размещается последовательно или последовательно-параллельно, причем чтение и запись осуществляют исключительно массивами слов. Участок МЛ, на котором размещен один массив, называется зоной. Зоны отделены друг от друга межзонными промежутками, необходимыми для того, чтобы не терялась информация при пусках и остановках МЛ. Длина зон и межзонных промежутков определяется стандартами ИСО для каждого способа записи информации и скорости движения МЛ. Время пуска, остановки и реверса ленты в современных ЗУ составляет 1—10 мс. Для поиска зон используют как адресный, так и ассоциативный способы. С целью уменьшения износа ленты и магнитных головок конструкции ЗУ обычно предусматривается отвод головок от МЛ в режиме перемотки.

Размещение информации на МЛ также должно соответствовать стандартам ИСО. Например, в ЕС ЭВМ для обмена данными между отдельными машинами

используется размещение информации, показанное на рис. 4.71. Число дорожек записи 9, из них 8 информационных и одна контрольная, на которой записаны разряды поперечной нечетности. Контрольные разряды формируются путем подсчета числа единиц в строке. Если это число четное, то в контрольный разряд записывается «1», а если нечетное — записывается «0». Вследствие этого в любой строке всегда будет нечетное число единиц, благодаря чему удастся обнаруживать любое нечетное число сбоев в строке. Ширина каждой дорожки равна 1,2 мм. Номинальное расстояние между зонами составляет 15,2 мм, а минимальное — 12,7 мм. На расстоянии 0,5 мм от конца зоны находится строка продольной нечетности (СПН), разряды которой формируются путем подсчета числа «1» в пределах зоны на соответствующих дорожках. Плотность записи до 32 бит/мм. В различных типах ЗУ на МЛ, входящих в состав ЕС ЭВМ, используются последние 4 способа записи информации из числа выше рассмотренных. При таком размещении информации признаком начала зоны служит первый сигнал, считанный после прохождения междюзонного расстояния. Признак конца зоны — СПН. Специальной дорожки для СИ нет. СИ получают за счет того, что в каждой строке всегда записана по крайней мере одна «1».

Емкость ЗУ на МЛ (одной катушки) достигает 300 Мбайт. Технические характеристики ЗУ на МЛ типа ЕС-5014 приведены в табл. 69.

4.13. СТРУКТУРА ПАМЯТИ ЦИФРОВЫХ ЭВМ. СИСТЕМЫ АДРЕСАЦИИ

Запоминающее устройство ЭВМ должно удовлетворять трем основным требованиям, а именно: иметь большую емкость, высокое быстродействие, низкую стоимость. Малая емкость ($N < 10^5$ бит) ЗУ существенно ограничивает класс решаемых на цифровой ЭВМ задач, так как для программы, исходных и промежуточных данных может нехватить ячеек ЗУ. Низкое быстродействие ($t_0 \gg 1$ мкс) замедляет скорость решения задач, так как в первом приближении

$$v = \frac{1}{t_1 + kt_0},$$

где t_1 — среднее время выполнения одной операции в арифметическом устройстве; k — среднее число обращений к ЗУ при выполнении одной операции. Для трехадресной машины (см. гл. 5) $k = 4$, так как надо выбрать из ЗУ команду, два исходных числа и записать в ЗУ результат.

Предъявляемые к ЗУ требования противоречат друг другу. Действительно, с увеличением емкости ЗУ его стоимость растет, а быстродействие падает. Причем эта закономерность сохраняется не только для одного типа ЗУ, но и для ЗУ различных типов. Поэтому в современных цифровых ЭВМ имеется целая иерархия ЗУ различной емкости и различного быстродействия.

Основным ЗУ машины, предназначенным для хранения данных, непосредственно участвующих в вычислительном процессе, является оперативное ЗУ (ОЗУ).

Исследование статистики обращений к отдельным ячейкам ОЗУ показало, что существует некоторое число ячеек (примерно 10 % от общего числа ячеек в ОЗУ), к которым обращение производится наиболее часто. Поэтому, с целью повышения быстродействия, в современных цифровых ЭВМ часть ОЗУ выполняется на базе запоминающих элементов повышенного быстродействия. Указанная часть ОЗУ получила название сверхоперативного ЗУ (СОЗУ).

Для запоминания больших массивов данных и другой информации, относительно редко используемой в вычислительном процессе, применяют так называемые внешние ЗУ (ВЗУ) большой емкости, но малого быстродействия, построенные на МЛ, МК, МД и МБ. Для согласования скоростей работы медленнодействующего ВЗУ и быстродействующего ОЗУ часто в ЭВМ имеются буферные ЗУ (БЗУ) средней емкости и быстродействия.

Кроме перечисленных типов ЗУ на различных уровнях (СОПЗУ, ОПЗУ, ВПЗУ) могут находиться ПЗУ, в которых записаны справочные данные, управляющие, стандартные программы, подпрограммы и микропрограммы.

Систему всех ЗУ принято называть памятью машины. Каждый из уровней памяти в свою очередь может состоять из нескольких подуровней с одинаковыми параметрами, либо отличающихся друг от друга по емкости и быстродействию.

Такие многоуровневые системы памяти, рассматриваемые как единое целое и снабженные своими аппаратными и программными средствами управления, которые обеспечивают перемещение информации вверх и вниз по иерархическим уровням, используя при этом стратегию, рассчитанную на минимизацию среднего времени выборки информации, получили название виртуальной (кажущейся) памяти ЭВМ.

При решении задач на цифровых ЭВМ с виртуальной памятью местом хранения данных и команд программ является совокупность всех ЗУ емкостью $B = \frac{N}{n}$ слов. В командах может быть использован любой из B виртуальных адресов, т. е. для программиста любое слово в виртуальной памяти должно быть равнодоступным.

Управление выборкой информации из ЗУ осуществляется путем преобразования адресной части команды, где указываются виртуальные адреса, в физические (исполнительные) адреса ячеек. Требование равнодоступности всех ячеек в виртуальной памяти, многоуровневость памяти ЭВМ и ее большая емкость обусловили необходимость создания специальных способов такого преобразования — систем адресации.

Наиболее распространенной системой адресации памяти в первых ЭВМ являлась прямая адресация, при которой в адресной части команды указывался исполнительный адрес операнда. Однако по мере увеличения емкости ЗУ прямая адресация приводила к увеличению длины команд. Действительно, при емкости памяти, например, 10^6 слов для представления физического адреса требует 20 бит. В этом случае длина адресной части была бы соизмерима с общепринятой длиной машинного слова (25—35 бит), что, в свою очередь, привело бы к значительному увеличению объема памяти, занимаемого программой. С целью устранения указанного недостатка, а также создания удобств для программирования, расширения логических возможностей и повышения производительности ЭВМ была разработана целая система непрямой адресации. При непрямой адресации в адресной части команды вместо исполнительного находится условный адрес. Условный адрес преобразуется в исполнительный с помощью так называемого механизма адресации.

Одним из простых способов уменьшения длины адреса является присоединенная адресация, при которой в адресной части команды АЧК) указывается только часть исполнительного адреса. Вторая часть, общая для всей программы или отдельного ее участка, находится в специальном регистре базового адреса (*RGBA*). При обращении к памяти по сигналу *H* из указанных частей составляется исполнительный адрес (рис. 4.72, а).

В случае, если программа использует больше, чем 2^m ячеек, где n — число разрядов *RGBAЧК*, то ее разбивают на отдельные участки. Каждый из участков использует не более чем 2^m ячеек памяти. Перед началом каждого из участков программы содержимое *RGBA* меняется.

Для создания удобств при программировании часто вместо присоединенной используют относительную адресацию, при которой

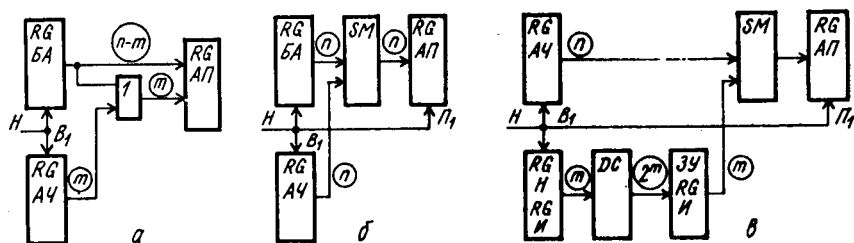


Рис. 4.72

исполнительный адрес получается путем суммирования базового адреса и кода из *RGBAЧК* (рис. 4.72, б). Для расширения возможностей ЭВМ над содержимым *RGBA* могут производиться различные арифметические и логические операции. В этом случае регистр *RGBA* принято называть индексным (*RGI*). В ЭВМ имеется, как правило, несколько *RGI* объединенных в *3Y RGI* (рис. 4.72, в). В старших разрядах *RGBAЧК* указывается номер *RGI* (*HRGI*), с которым необходимо производить операцию суммирования. Выборка нужного *RGI* осуществляется с помощью дешифратора. Наряду с описанными системами в настоящее время широко используется косвенная адресация, при которой в адресной части команды указывается номер регистра, хранящего адрес операнда (рис. 4.73, а). При большом числе таких регистров аппаратные затраты на механизм адресации велики. Поэтому, с целью их уменьшения, иногда в качестве указанных регистров используют первые *M* ячеек ОЗУ машины (рис. 4.73, б). Однако при этом время выборки операнда увеличивается в два раза. Первое обращение (когда *TU* в «1») необходимо для выборки адреса, а второе (*TU* в «0») — для выборки по этому адресу операнда.

Большинство современных ЭВМ работают в так называемом мультипрограммном (многопрограммном) режиме, при котором в машинной памяти находятся несколько программ, управляющих вычислительным процессом в некоторой последовательности. Поэтому заранее не известно, какие именно ячейки памяти в данный момент свободны и могут быть использованы выполняемой программой. Следова-

тельно, программа не может быть составлена в исполнительных адресах. В связи с этим возникает необходимость в переводе условных (виртуальных) адресов в исполнительные в процессе работы машины. Такой процесс перевода получил название динамического распределения памяти. Динамическое распределение памяти (ДРП) осуществляется особенно просто при страничной системе нумерации ячеек, при которой вся виртуальная память машины разбивается на страницы одинакового объема. В свою очередь адреса ячеек разбиваются на две части по α и β бит в каждой. Старшие α разрядов указывают номер страницы от 0 до $2^\alpha - 1$, а младшие β — номер ячейки внутри страницы от 0 до $2^\beta - 1$. При принятых обозначениях вся память, состоящая из $2^\alpha \times 2^\beta$ ячеек, разбивается на 2^α страниц по 2^β ячеек в каждой. При таком разбиении памяти ДРП состоит

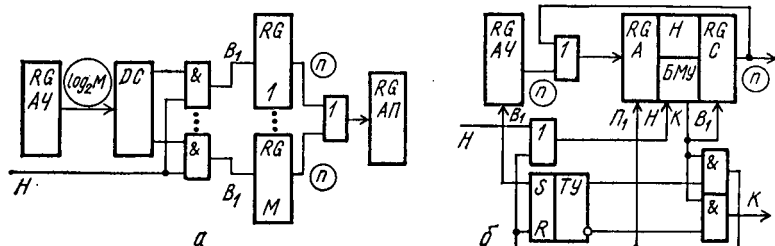


Рис. 4.73

в переводе условных адресов страниц данной программы в исполнительные. Для осуществления такого перевода необходимо иметь таблицу соответствия страниц (страничную таблицу), т. е. таблицу функции

$$\Phi = \Phi(P, A),$$

где Φ — физический адрес страницы памяти, в которой хранится условная страница с адресом A ; P — номер программы, к которой относится условная страница с адресом A . В зависимости от того, какое ЗУ (адресное или ассоциативное) используют для хранения таблицы функции Φ , различают страничные таблицы первого и второго типа.

Входом в страничную таблицу первого типа служит условный адрес страницы A и номер программы P (рис. 4.74). Каждая строка страничной таблицы содержит физический номер страницы, соответствующий заданным A и P . Выбранное по номеру P и адресу A слово из таблицы является физическим адресом Φ данной страницы, который совместно с условным адресом слова (УАС) внутри страницы по методу присоединенной адресации образует исполнительный адрес. Так как страничные таблицы первого типа используют адресное ЗУ, то их можно хранить в ОЗУ. При этом каждое обращение к памяти, задаваемое программой, требует двух обращений к ОЗУ: одного к страничной таблице за физическим адресом страницы, второго — за искомым словом по исполнительному адресу. Это существенно снижает быстродействие цифровых ЭВМ. Для устранения этого недостатка для хранения страничных таблиц иногда используют быстро-

действующие ЗУ на триггерах. Однако при большом числе программ и страниц, используемых программами, такая реализация динамического распределения памяти требует больших аппаратурных затрат.

В некоторых случаях, с целью экономии оборудования, страничные таблицы неактивных (т. е. не участвующих в управлении цифровых ЭВМ) программ хранят в ОЗУ. По мере приобретения активности какой-либо из программ соответствующую ей страничную таблицу вводят в быстродействующее ЗУ страничных таблиц. Однако при частой смене программ требуется выполнять большое число операций по обмену информацией между указанными ЗУ, что приводит к уменьшению производительности цифровых ЭВМ.

Для устранения указанных недостатков используют страничные таблицы второго типа на базе ассоциативных ЗУ, в которых хранятся

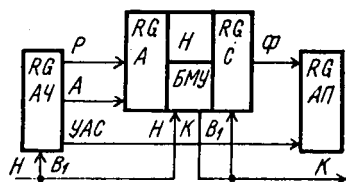


Рис. 4.74

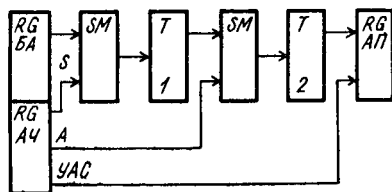


Рис. 4.75

номера условных страниц информации, находящейся в ОЗУ, и их физические адреса. При этом признаковую часть слов в ассоциативном ЗУ образуют номера условных страниц. Если в ассоциативном накопителе имеется слово с заданным признаком, в качестве которого используется номер программы P и условный адрес страницы A , то оно выбирается и вместе с УАС образует исполнительный адрес. Если данной страницы в ОЗУ нет, то вырабатывается сигнал, по которому специальная управляющая программа осуществляет поиск искомой страницы в БЗУ или ВЗУ и перезапись ее в ОЗУ. После этого номер программы, условный и физический адрес страницы записываются в АЗУ. В некоторых случаях с целью уменьшения аппаратурных затрат за счет сокращения емкости АЗУ часть страничных таблиц хранят в ОЗУ, а в АЗУ хранят данные о страницах, к которым наиболее часто обращаются. Если в АЗУ нужной информации нет, то производится просмотр части страничных таблиц, хранящейся в ОЗУ, путем последовательной выборки строк таблиц. Найденная таким способом строка страничной таблицы записывается в АЗУ, а на ее место из АЗУ переписывается строка таблицы, соответствующей наименее активной странице, так как предполагается, что к вновь полученным данным обращение будет происходить чаще.

Наряду со страничной организацией адресации в настоящее время распространена также сегментная система, при которой часть страниц объединяется в сегмент. Условный адрес состоит из номера сегмента S , адреса страницы и номера слова (рис. 4.75). Исполнительный адрес формируется с помощью двух таблиц: сегментной $T1$ и страничной $T2$. Формирование исполнительного адреса начинается

сложением слов из *РГБА* с *S*. Полученное слово используется для обращения к сегментной таблице. Выбранное из таблицы *T1* слово суммируется с *A*. Далее полученное слово используется для обращения к таблице *T2*. Слово из таблицы *T2* совместно с УАС образует исполнительный адрес. Таблицы *T1* и *T2* могут строиться как по первому, так и по второму способу.

4.14. СТРУКТУРЫ СОЗУ И ОЗУ

СОЗУ является частью ОЗУ и используется для хранения наиболее часто встречающейся информации. Цикл обращения к СОЗУ значительно меньше, чем к ОЗУ. Вследствие этого среднее время обращения к системе СОЗУ—ОЗУ меньше, чем время обращения к ОЗУ. Введение СОЗУ позволяет уменьшить поток запросов к ОЗУ и тем самым увеличить производительность за счет уменьшения времени обслуживания запросов. Кроме того, при наличии СОЗУ повышается быстродействие цифровых ЭВМ как за счет разницы в цикле СОЗУ и ОЗУ, так и за счет более глубокого совмещения операций, а именно: при выполнении длинных операций СОЗУ заполняется, а при выполнении коротких — освобождается.

Одним из основных вопросов, возникающих при введении СОЗУ, является способ адресации ячеек СОЗУ. Известно три способа адресации: прямая, магазинная и ассоциативная. При прямой адресации

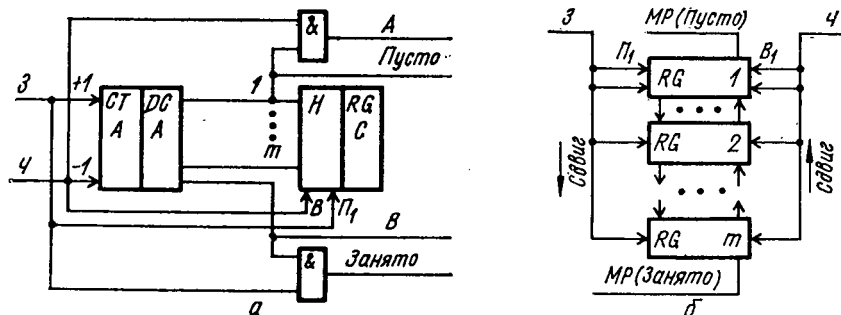


Рис. 4.76

ячейки СОЗУ имеют номера от 1 до *m*, а ячейки ОЗУ — от *m* + 1 и далее. Адреса ячеек СОЗУ указываются непосредственно в адресной части команды. Так как *m* обычно невелико, то для указания номера ячейки СОЗУ отводится небольшое число разрядов, например, 4 при *m* = 16. При прямой адресации программист все время должен следить за заполнением ячеек СОЗУ. Это повышает эффективность использования ячеек СОЗУ, однако сильно затрудняет программирование.

При магазинной (стековой, гнездовой) адресации адрес следующего операнда, записанного в СОЗУ, определяется из адреса предыдущего путем прибавления ± 1 , т. е. адресация осуществляется либо с помощью счетчика, либо путем перемещения массива слов в СОЗУ на одну ячейку вниз или вверх. Структура СОЗУ при магазинной адресации с помощью счетчика показана на рис. 4.76, а. При записи

нового слова в счетчик адреса (СТА) прибавляется 1, а при чтении — вычитается. Поэтому очередное слово записывается в ячейку, номер которой на 1 больше номера предыдущей ячейки, а чтение осуществляется в обратном порядке. Сигналы «СОЗУ пусто» и «СОЗУ занято» формируются по первому и $m + 1$ -му состояниям СТА. В случае необходимости вырабатываются сигналы ошибок А (чтение из пустого СОЗУ) и В (переполненное СОЗУ).

В СОЗУ с перемещением слов одноименные разряды регистров соединены цепями сдвига (рис. 4.76, б). Прием и выдача слов в СОЗУ осуществляется через первый регистр. При каждом чтении слова перемещаются на одну ячейку вверх, а при записи — вниз. В случае необходимости сигналы «Пусто», «Занято», А и В можно получить, вводя «1» в специальный маркерный разряд (МР) регистров. Заметим, что СОЗУ с перемещением слов емкостью m n -разрядных слов можно получить, используя n m -разрядных реверсивных сдвигающих регистров, у которых сигналы 3 и 4 управляют направлением сдвига.

Обмен информацией между ОЗУ и магазинным СОЗУ осуществля-

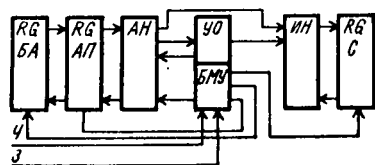


Рис. 4.77

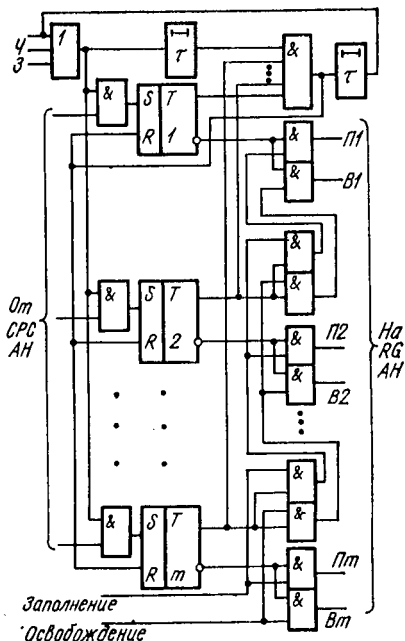


Рис. 4.78

ется с помощью специальных команд, что, как и в предыдущем случае, усложняет программирование. Автоматический обмен информацией между уровнями ЗУ является признаком виртуальной памяти. Такой обмен информацией между СОЗУ и ОЗУ возможно реализовать при ассоциативной адресации (рис. 4.77). СОЗУ содержит информационный накопитель (ИН), ассоциативный накопитель (АН) со схемами сравнения, блок местного управления (БМУ) с узлом очереди (УО) и регистры адреса памяти ($RG\ AП$) и слова ($RG\ C$). Признаками слов являются их адреса в ОЗУ. При обращении к памяти на $RG\ AП$ поступает адрес (А) слова, а на БМУ — сигнал характера обращения (Ч или З). Адрес слова сравнивается с адресами, хранящимися в АН. В случае совпадения адресов слово выбирается из ИН и записывается в $RG\ C$. Если же в АН отсутствует адрес, то вырабатывается сигнал P , по которому происходит обращение к ОЗУ. Выбранное по заданному адресу из ОЗУ слово записывается в первую свободную ячейку ИН, а его адрес — в соответствующую ячейку

АН. В дальнейшем обращение к памяти по данному адресу приводит к выборке слова из СОЗУ, а не из ОЗУ.

С течением времени все ячейки СОЗУ будут заполнены. Поэтому для записи нового слова требуется освободить одну из ячеек СОЗУ. Очевидно, следует освободить ту ячейку СОЗУ, на долю которой приходится наименьшее число обращений. Учет количества обращений к ячейкам осуществляется узлом очереди. Схема одного варианта узла очереди показана на рис. 4.78. Каждая ячейка СОЗУ имеет свой триггер очереди (ТО). При обращении к какой-либо ячейке ее ТО устанавливается в «1». Если ТО всех ячеек установлены в «1», то вырабатывается сигнал СО, который осуществляет сброс в «0» всех ТО. В случае необходимости освобождается первая ячейка, ТО которой находится в «0». Для этого по сигналу $P = 1$ на схему поступает сигнал «Освобождение», по которому адрес из *RGАП* переписывается в буферный регистр адреса и вырабатывается сигнал выдачи слова из i -й ячейки (B_i). По сигналу B_i извлекается адрес из АН и записывается в регистр *RGАП*. После этого возбуждается выход i -й схемы равенства слов (СРС) в АН и слово из i -й ячейки ИН выводится на *RGC*. Выведенное слово по полученному ранее адресу записывается в ОЗУ, а на его место в АН СОЗУ по сигналу «Заполнение» из *RGБА* заносится адрес новой ячейки ОЗУ, после чего по сигналу «Запись» в выбранную ячейку ИН заносится новое слово. Иногда, с целью повышения быстродействия за счет совмещения во времени операций освобождения ячейки СОЗУ с выполнением других операций, свободную ячейку СОЗУ готовят заранее. При этом отпадает необходимость в *RGБА*, однако один из регистров СОЗУ не будет участвовать в работе.

Существуют и другие способы построения узла очереди. Например, с целью его упрощения выбор слова, удаляемого из СОЗУ, можно производить наугад с помощью позиционного счетчика с дешифратором или кольцевого счетчика на сдвигающем регистре. При каждом обращении к ОЗУ на счетчик поступает сигнал +1 и все ячейки освобождаются по порядку.

Оценим среднее время обращения t_0 к виртуальной памяти типа СОЗУ-ОЗУ. Пусть t_1 и t_2 — циклы обращения соответственно к СОЗУ и ОЗУ. Обращение к памяти, при котором операция производится только с СОЗУ, назовем простым, а все остальные обращения — сложными. Время простого обращения равно t_1 , а сложного — зависит от выполняемой операции. При сложном обращении удаляемое слово выводится из СОЗУ и записывается в ОЗУ. Одновременно с этим в СОЗУ записывается новое слово. Следовательно, время выполнения сложной записи равно $t_1 + t_2$. При чтении после записи в ОЗУ удаленного из СОЗУ слова производится выборка новой информации. Выбранное из ОЗУ новое слово выводится на выходные шины памяти и одновременно с этим записывается в СОЗУ. Следовательно, время выполнения сложного чтения равно $t_1 + 2t_2$. Пусть W_1 , W_2 и W_3 соответственно вероятности появления операций простого обращения, сложной записи и чтения. Тогда

$$t_0 = t_1 W_1 + (t_1 + t_2) W_2 + (t_1 + 2t_2) W_3.$$

Так как $W_1 + W_2 + W_3 = 1$, то получаем окончательно

$$t_0 = t_1 + t_2 (W_2 + 2W_3).$$

Вероятности W_i зависят от алгоритма работы УО, емкости СОЗУ и характеристик программ. В большинстве случаев емкость СОЗУ должна быть порядка 32 ячеек и дальнейшее увеличение емкости практически не влияет на быстродействие виртуальной памяти типа СОЗУ-ОЗУ.

В отличие от чисел, команды размещаются и выбираются из памяти последовательно, в порядке возрастания их номеров. Учет этой особенности приводит к повышению эффективности использования СОЗУ команд. Простейшая схема СОЗУ команд состоит из накопителя, в котором реализована возможность обращения по двум адре-

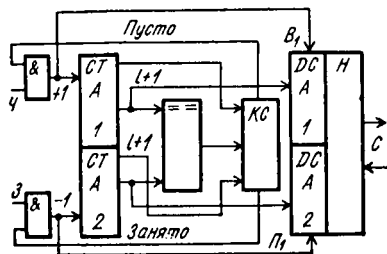


Рис. 4.79

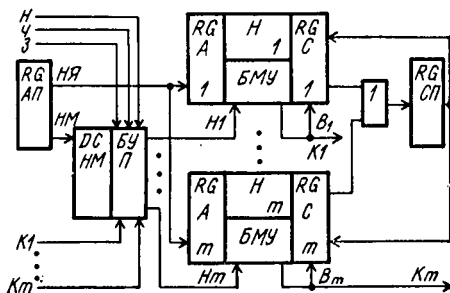


Рис. 4.80

сам (рис. 4.79). Заполнение и освобождение СОЗУ происходит по сигналам чтения и записи с помощью двух счетчиков адресов на СТА1 и СТА2. Для правильной работы СОЗУ необходимо фиксировать состояния «Пусто» и «Занято». При $m = 2^l$ для этого достаточно иметь l -разрядную схему равенства слов, а также $(l + 1)$ -разрядные счетчики адресов. Состояние «Пусто» обнаруживается при равенстве адресов на СТА1 и СТА2. Состояние «Занято» определяется равенством l разрядов содержимого счетчиков СТА1 и СТА2 и неравенством их $(l + 1)$ -х разрядов. Для формирования сигналов «Пусто» и «Занято» служит комбинационная схема КС, управляемая старшими разрядами счетчиков адресов и выходом схемы равенства слов. Сигналы, формируемые комбинационной схемой, блокируют выдачу слова из пустого СОЗУ и прием слова на занятое СОЗУ.

Правильная работа такого СОЗУ может быть нарушена появлением команд безусловного и условного переходов (см. гл. 6). Чтобы этого не происходило, команды перед записью в СОЗУ подвергаются анализу. Выявленные команды безусловного перехода в СОЗУ не записывают, а используют для изменения адресов следующих команд при выборке их из ОЗУ и записи в СОЗУ. Команды же условного перехода фиксируются в СОЗУ, после чего запись в СОЗУ новой информации прекращается до выполнения команды условного перехода, т. е. до освобождения СОЗУ.

Применение СОЗУ описанного типа не сокращает общего количества обращений к ОЗУ за командами, но увеличивает быстродействие

цифровых ЭВМ за счет более глубокого совмещения работы отдельных устройств, так как при выполнении длинных операций из ОЗУ можно выбирать следующие команды.

При построении ОЗУ широко используется агрегатный принцип, при котором ЗУ со своими адресными, разрядными и управляющими цепями конструктивно выполняется в виде модуля с унифицированными внешними связями. Из таких модулей может быть создано ЗУ требуемой емкости по блок-схеме, показанной на рис. 4.80. Регистр адреса памяти (RGAP) условно разбивается на две части. В первой части хранится номер ячейки (НЯ), а во второй — номер модуля (НМ). Номер модуля ЗУ можно задавать старшими разрядами адреса,

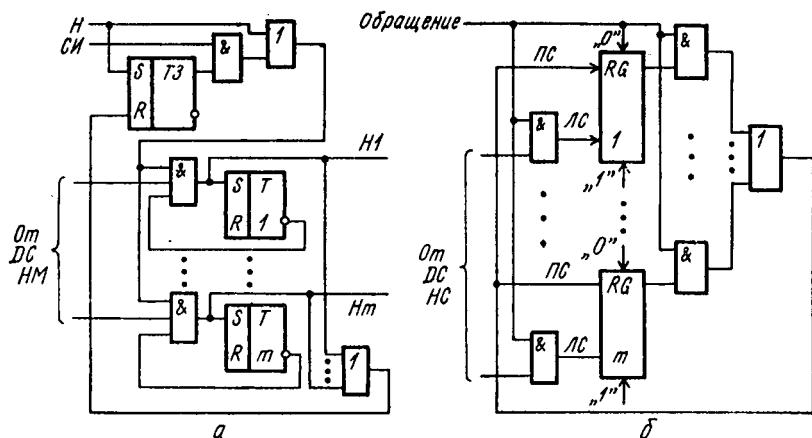


Рис. 4.81

но тогда две ячейки с соседними адресами будут находиться в одном модуле и совместить работу двух и более модулей будет трудно, так как обращение к ЗУ происходит, как правило, к ячейкам с соседними номерами.

Управление памятью осуществляет блок БУП, который состоит из узла анализа занятости (УАЗ) модулей и узла очереди. Каждому модулю ЗУ в УАЗ соответствует свой триггер (рис. 4.81, а). Обращение к данному модулю возможно только в случае, если соответствующий ему триггер находится в нуле. В противном случае триггер запроса ТЗ устанавливается в «1» и по каждому СИ запрос повторяется. Узел очереди (УО) осуществляет учет обращений к отдельным страницам виртуальной памяти с целью обнаружения страницы для удаления из ОЗУ. Принципы построения УО зависят от правила определения удаляемой страницы. Известно большое число таких правил. Наиболее известными являются: 1) выбор страницы наугад; 2) выбор страницы, к которой в течение определенного времени не производилось обращение; 3) подсчет числа обращений к отдельным страницам; 4) выбор страницы по максимуму временного интервала, в течение которого к ней не производилось обращение и др. Первые два правила и реализующие их УО аналогичны соответствующим правилам и УО, используемым в СОЗУ. В соответствии с третьим правилом произво-

дится подсчет числа обращений к отдельным страницам и удаляемая страница определяется по минимуму обращений к ней.

Подсчет удобно производить с помощью сдвигающих регистров. Каждой странице соответствует такой регистр (рис. 4.81, б). При обращении к какой-либо странице производится операция сдвига влево в соответствующем регистре, который выбирается с помощью дешифратора номера страницы (ДСНС). При этом в младший освободившийся разряд регистра записывается «1». При заполнении какого-либо из регистров производится сдвиг вправо всех регистров, а в освободившиеся старшие разряды заносятся нули. При определении удаляемой страницы отыскивается регистр с минимальным числом «1» путем поразрядного опроса всех регистров с помощью схем приоритета. На i -й разряд схемы приоритета поступает сигнал A с выхода $(i - 1)$ -го разряда. Сигнал A проходит до первого триггера, находящегося в «0», например, j -го и преобразуется в сигнал «Номер страницы, подлежащей удалению, равен j ». Если триггеры всех регистров находятся в «1», то формируется сигнал B опроса $(i + 1)$ -х разрядов регистров. Аналогично может быть построен УО и для четвертого правила. Выбор того или иного правила осуществляется обычно путем их моделирования на цифровых ЭВМ. При этом определяют оптимальное правило, обеспечивающее наименьшее число пересылок страниц при работе цифровых ЭВМ на данной совокупности программ.

4.15. ВЫБОРКА В ЗУ СЛОВ ПЕРЕМЕННОЙ ДЛИНЫ

Одним из методов повышения производительности цифровых ЭВМ является введение операций над словами переменной длины. Использование таких операций, кроме удобств программирования, приводит к сокращению числа команд в программах, а следовательно, и к сокращению объема памяти, занимаемой программой. Еще в большей степени сокращается объем памяти, занимаемой данными, так как для их представления отводится столько бит, сколько для этого необходимо и, следовательно, отсутствуют неиспользуемые разряды машинных слов. Уплотнение числовой и управляющей информации сокращает пересылки данных между АЛУ и ЗУ и исключает обработку малозначащих разрядов слов.

При выборке слов переменной длины всю память цифровых ЭВМ разбивают на участки, число разрядов в которых равно числу разрядов в одном слого. Каждый участок имеет свой номер (адрес). Для выборки из памяти слова необходимо задать адреса всех его слогов. При заданном адресе первого слога это можно сделать одним из следующих способов: а) задать адрес последнего слога; б) задать длину слова в слогах; в) задать число неиспользуемых разрядов в адресе; г) ввести специальный символ конца слова.

Адрес любого слога можно условно разбить на две части A и B . Часть A является адресом ячейки ЗУ, в которой хранится искомый слог. Часть B показывает номер позиции, которую занимает данный слог в ячейке. Так как первый слог слова может занимать любую позицию, то для выборки слова, состоящего из двух и более слогов, может понадобиться два обращения к памяти. Выборку слова можно

ускорить при одновременном обращении по адресам A и $A + 1$. Здесь и далее предполагается, что в слове не более k слогов, где k — длина ячейки ЗУ в слогах.

Из выбранных по адресам A и $A + 1$ слогов необходимо образовать одно слово. Известно два метода формирования слов: расшифровка и сдвиг. При расшифровке выбранные из памяти слоги заносятся в $RGC1$ и $RGC2$ (рис. 4.82). Далее сдвижателем (СДВ), управляемым посредством дешифратора ДС частью B адреса, слоги передвигаются на нужное число позиций влево и поступают на узел блокировки (УБ), который окончательно формирует слово. УБ управляется частью регистра адреса, в которой записана длина слова L (в слогах).

Процесс формирования слова при чтении показан на рис. 4.83, а. Процесс записи протекает в обратном порядке (рис. 4.83, б). Слово

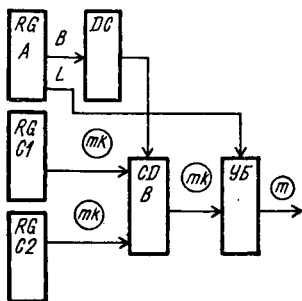


Рис. 4.82

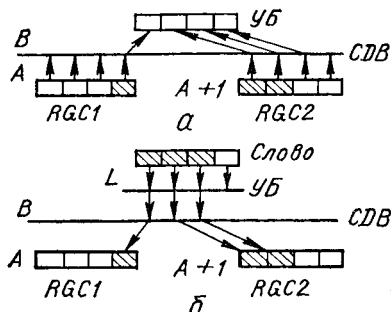


Рис. 4.83

поступает на входной блок, состоящий из УБ и СД, заносится на нужные позиции $RGC1$ и $RGC2$ после чего переписывается в накопитель ЗУ по адресам A и $A + 1$. Если чтение и запись осуществляется по одному слову и начало слова может занимать любую позицию в RGC с равной вероятностью, то в среднем для выборки и записи слова требуется $1 + \frac{q-1}{l}$ обращений к ЗУ, где q — длина слова в слогах, а l — длина ячейки ЗУ.

Формирование слова путем сдвига отличается от формирования путем расшифровки тем, что $RGC1$ и $RGC2$ выполняются сдвигающими. При этом сдвиг слова осуществляется на $RGC1$ и $RGC2$, а не на сдвижателе. Число сдвигов определяется адресом B позиции первого слога.

4.16. НЕКОТОРЫЕ СТРУКТУРНЫЕ СПОСОБЫ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ ЗУ

Увеличение емкости ЗУ при прочих равных условиях обычно сопровождается снижением их быстродействия и наоборот, улучшение их скоростных характеристик приводит к уменьшению емкости ЗУ. Поэтому важное значение имеет разработка способов организации структур ЗУ, направленных на существенное ослабление указанной взаимосвязи емкости и быстродействия. Одним из таких способов,

например, является иерархическая организация памяти, в простейшем случае, сводящаяся к двухуровневой структуре типа СОЗУ-ОЗУ. В общем случае эффект применения какого-либо способа повышения быстродействия ЗУ может быть охарактеризован коэффициентом повышения быстродействия (КПБ)

$$K_t = \frac{T}{T_t},$$

где T — суммарное время обращения к памяти при выполнении некоторой программы вычислений; T_t — время обращения к памяти для той же программы вычислений при использовании i -го способа повышения быстродействия. Для подсчета КПБ от введения СОЗУ с циклом T_{01} определим T и T_t , предполагая, что число обращений к СОЗУ равно a , а число обращений к ОЗУ — b . Тогда $T = T_0(a + b)$; $T_t = T_{01}a + T_0b$, где T_0 — цикл обращения к ОЗУ. Отсюда находим

$$K_1 = \frac{T_0(a+b)}{T_{01}a + T_0b} = \frac{1 + \frac{a}{b}}{1 + \frac{a}{b} \cdot \frac{T_{01}}{T_0}} = \frac{1+h}{1+h \frac{T_{01}}{T_0}},$$

что при $h \gg 1$ дает $K_1 \approx \frac{T_0}{T_{01}}$. Отношение $h = \frac{a}{b}$ называют отношением попадания, причем экспериментально установлено, что h сильно зависит от емкости СОЗУ и в меньшей степени от размеров пересылаемых массивов слов и от используемого алгоритма замены страниц в СОЗУ. Увеличение емкости ОЗУ приводит к уменьшению h . Поэтому при большой емкости ОЗУ целесообразно использовать трех- и более уровневую память. В этом случае

$$K_1 = \frac{\sum_{j=1}^n \frac{a_j}{a_n}}{1 + \sum_{j=1}^n \frac{a_j T_{0j}}{a_n T_0}},$$

где a_j — число обращений к j -му уровню памяти; T_{0j} — цикл обращения к j -му уровню ($T_{0n} = T_0$); n — число уровней памяти. При $\frac{a_1}{a_n} \gg \frac{a_2}{a_n} \gg \dots \gg \frac{a_{n-1}}{a_n} \gg 1$, как и в случае двухуровневой памяти, $K_1 = \frac{T_0}{T_{01}}$.

Другой способ повышения быстродействия памяти состоит в совмещении операций в ЗУ, реализуемом путем расслоения ячеек в модульных ЗУ, совмещения дешифрации адреса следующего слова и выборки предыдущего, применения длинных ячеек, хранящих два и более слов, обращения по многим адресам одновременно.

Основная идея расслоения ячеек в модульных ЗУ описана в п. 4.14, рис. 4.80. Здесь в начале обращения к ЗУ адреса поступают на *РГАП* и, если соответствующие модули еще не заняты, передаются на их *РГА*. Поступление адресов прекращается, если соответствующий

адресу модуль ЗУ уже занят. Количество поступивших в начале цикла адресов определяет число одновременно работающих модулей. Такая приближенная модель работы модульного ЗУ позволяет определить вероятность W_i одновременной работы i модулей. Она будет равна вероятности того, что в начале цикла поступило i неравных друг другу адресов (точнее, имеющих разные номера модулей), после чего поступил адрес уже занятого модуля. Предполагая, что вероятности обращения к отдельным модулям независимы и равны между собой, получим

$$W_i = \frac{m(m-1)(m-2)\dots(m-(i-1))}{m^{i+1}},$$

где m — число модулей в ЗУ. Отсюда находим среднее число одновременно работающих модулей

$$M = \sum_{i=1}^m i W_i.$$

При $m = 2, 4, 8$ получаем соответственно $M = 1,5; 2,2; 3,24$. Следовательно, за один цикл будет выбираться в среднем M слов и во столько же раз уменьшаться затраты времени на обращение к памяти, т. е. $K_2 = M$. Используя специальные аппаратные или программные средства для сортировки адресов, можно добиться почти полной загрузки модулей в каждом цикле. В этом случае $K_2 \rightarrow m$.

Совмещение дешифрации адреса следующего слова с выборкой очередного позволяет уменьшить цикл ЗУ на время $\min(t_d, T_0 - t_d)$, где t_d — время дешифрации адреса. Предположив, что $t_d < T_0 - t_d$, получим

$$K_3 = \frac{T_0}{T_0 - t_d} = 1 + \frac{t_d}{T_0 - t_d}.$$

В случае же, когда весь цикл работы ЗУ может быть разбит на c независимых операций, то $T_{03} = \max t_i$, где t_i — время выполнения i -й операции, $i = 1, 2, \dots, c$. При равенстве всех t_i получим

$$K_3 = \frac{T_0}{\frac{T_0}{c}} = c.$$

Хранение в каждой ячейке ЗУ нескольких слов с окончательной выборкой одного из них на регистре слов обеспечивает сокращение не только времени поиска информации, но и времени чтения с накопителя при условии, что выборка слов осуществляется в порядке расположения их в накопителе. Среднее время обращения к ЗУ в этом случае будет равно

$$T_{0ч} = \frac{T_0}{l} + t_{ов},$$

где l — среднее число слов, выбираемых из ячейки за одно обращение, $t_{ов}$ — время окончательной выборки слов на регистре. В этом случае

$$K_4 = \frac{T_0}{\frac{T_0}{l} + t_{ов}}.$$

Обычно $t_{0в} \ll T_0$, поэтому $K_2 \approx 1$. Величина l зависит от числа слов в одной ячейке m и от «разброса» адресов в программах. Если все слова выбираются последовательно, то $l = m$, если разброс адресов большой, то $l = 1$, т. е. $m \geq l \geq 1$.

Выборка слов в ЗУ по двум и более независимым адресам требует введения дополнительных адресных и разрядных цепей, что обуславливает целесообразность применения такого способа повышения быстродействия в ЗУ с полупроводниковыми накопителями (рис. 4.36, 4.37). При наличии q независимых каналов обращения к ЗУ эффективное быстродействие ЗУ в q раз выше фактического, т. е. $K_5 = q$. Чаще всего $q = 3$, так как это удобно при работе с ЭВМ — при каждом обращении выбираются два операнда и записывается результат предыдущей операции.

Ускорить выборку массивов слов в ЗУ с М-поиском можно путем введения в ЗУ распределителя сигналов РС (рис. 4.84). При обращении

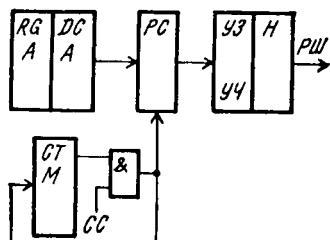


Рис. 4.84

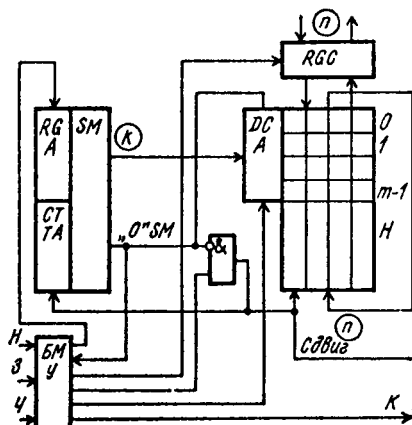


Рис. 4.85

нии к ЗУ адрес первого слова дешифрируется и соответствующий разряд РС устанавливается в единицу. Последующие обращения к накопителю будут уже происходить не по сигналам с ДСА, а при каждом переключении РС под действием синхронизирующих сигналов СС. После завершения операции с массивом, число слов в котором подсчитывается счетчиком СТМ, подача СС прекращается, а РС устанавливается в исходное состояние. Цикл обычного ЗУ при выборке массива из p слов равен $p(t_{a.ц} + t_{p.ц})$, где $t_{a.ц}$ — задержка сигналов в адресных цепях, $t_{p.ц}$ — задержка в разрядных цепях. При выборке массива с использованием РС

$$T_{05} = t_{a.ц} + pt_{p.ц}.$$

Поэтому

$$K_5 = \frac{p(t_{a.ц} + t_{p.ц})}{t_{a.ц} + t_{p.ц}p},$$

а при $pt_{p.ц} \gg t_{a.ц}$ имеем

$$K_5 = 1 + \frac{t_{a.ц}}{t_{p.ц}}.$$

Рассмотренные выше способы повышения быстродействия применимы в ЗУ с М-поиском. Однако реализация М-поиска в ЗУ емкостью

порядка 10^{15} бит приводит к очень сложным и медленно действующим адресным и разрядным цепям. Реализация же В-поиска в таких ЗУ не обеспечивает быстрого произвольного доступа к отдельным ячейкам. Поэтому с целью повышения быстродействия ЗУ большой емкости, по-видимому, наиболее целесообразно использовать комбинированный М-В-поиск. В этом случае накопитель с В-поиском имеет часть пространственно адресуемых ячеек (рис. 4.85) и состоит из n сдвигающих регистров, каждый из которых имеет длину в $\frac{N}{n}$ разрядов. В одном регистре хранятся одноименные разряды всех слов (в первом регистре — первые, во втором — вторые и т. д.). Обмен информацией между регистром слов и первыми пространственно адресуемыми ячейками накопителя организован на основе произвольного доступа с М-поиском. Будем считать, что адрес A слова совпадает с адресом ячейки α в тот момент, когда в СТТА хранится нуль. В случае, когда содержимое СТТА равно a , в нулевой ячейке накопителя находится слово с номером A . Поэтому разность $A - a$ по модулю $\frac{N}{n}$ будет равна номеру ячейки α , в которой в данный момент находится слово с адресом A . Если

$$\alpha = A - a \left(\bmod \frac{N}{n} \right) < m,$$

то это означает, что слово с адресом A хранится в одной из пространственно адресуемых ячеек и для его выборки достаточно адрес α подать на ДСА. Если же $\alpha \geq m$, то необходимо выполнять сдвиг слов в накопителе до тех пор, пока указанное неравенство будет выполняться. Проверку неравенства и формирование номера ячейки α можно производить с помощью сумматора SM , на котором определяется разность $A - a$. Так как $m = 2^k$ (k — целое положительное число), то при $\alpha < m$ в старших разрядах сумматора будут нули, а в k младших разрядах — номер ячейки с искомым словом. При наличии в старших разрядах SM хотя бы одной единицы обращение к накопителю блокируется и производится сдвиг слов в накопителе до появления сигнала «0» $SM = 1$.

Сравним быстродействие ЗУ с выборкой 2Д и рассматриваемого ЗУ. Если считать, что задержка сигналов в разрядных цепях ЗУ типа 2Д пропорциональна длине разрядных шин (т. е. числу ячеек), то

$$T_0 = \frac{N}{n} \tau,$$

где τ — время прохождения сигналов через одну ячейку накопителя. При оценке цикла обращения к ЗУ на рис. 4.85 будем считать, что выборка слов производится массивами, а число слов в массиве $Q \leq m$. Так как перед началом выборки массив может занимать произвольное место в накопителе, то для его перемещения в дешифрируемую секцию потребуется при $m, Q \ll \frac{N}{n}$ около $\frac{N}{2n}$ сдвигов, что на одно обращение составляет $\frac{N}{2nQ}$ сдвигов. Поэтому среднее время обращения к такому ЗУ составит

$$T_{07} = \frac{N}{2nQ} \tau_c + m\tau,$$

где τ_0 — время сдвига слов в накопителе на один разряд. Таким образом,

$$K_7 = \frac{\frac{N}{n} \tau}{\frac{N}{2nQ} \tau_0 + m\tau} = \frac{1}{\frac{\tau_0}{2Q\tau} + \frac{m}{N}} \approx \frac{2Q\tau}{\tau_0},$$

поскольку $N \gg mn$. Отсюда следует, что $K_7 > 1$, если только $2Q\tau > \tau_0$ или $Q > \frac{\tau_0}{2\tau}$. Реальность последнего неравенства подтверждается тем, что обычно Q имеет порядок $10^3 - 10^4$, а $\frac{\tau_0}{\tau}$ лежит в пределах от 10 до 100.

Заметим, что каждый из рассмотренных способов может использоваться как самостоятельно, так и в комбинации с другими способами. Повышение быстродействия ЗУ обычно достигается за счет аппаратных изменений в накопителе (Н), адресных цепях (АЦ), разрядных цепях (РЦ), блоке местного управления (БМУ). В табл. 70 приведены итоговые данные для каждого из рассмотренных выше способов и отмечено усложняемое оборудование ЗУ. Это упрощает выбор наиболее оптимального сочетания способов повышения быстродействия ЗУ еще на стадии их эскизного проектирования.

Таблица 70

Название способа	Коэффициент повышения быстродей- ствия	Усложнения			
		Н	АЦ	РЦ	БМУ
Иерархическая организация памяти	$\frac{T_0}{T_{01}}$	×	×	×	×
Расслоение ячеек в модульных ЗУ	M	—	×	×	×
Совмещение операций в ЗУ	c	—	×	×	×
Хранение в одной ячейке нескольких слов	l	—	×	×	×
Многоканальная выборка слов	q	—	×	×	×
Выборка слов массивами с использованием распределителей сигналов	$1 + \frac{t_{a.ц}}{t_{p.ц}}$	—	×	—	×
Применение комбинированного М-В-поиска	$2 \frac{Q\tau}{\tau_c}$	×	×	—	×

4.17. ПЕРСПЕКТИВНЫЕ ДЛЯ ЦВТ СИСТЕМЫ ПАМЯТИ

Первая универсальная цифровая вычислительная машина «Марк-1» имела ЗУ трех типов: на перфорированной бумажной ленте; на механических цифровых колесах и на тумблерах.

Первая электронная универсальная цифровая ЭВМ «ЭНИАК» имела ЗУ на ламповых триггерах емкостью 700 бит. Применение ЗУ на электронных лампах позволило повысить быстродействие и надежность ЭВМ. Однако большие

габариты и потребляемая мощность значительно ограничивали емкость памяти. Поэтому ЗУ на лампах были быстро вытеснены разработанными в 1946—1948 гг. ЗУ на ртутных линиях задержки (ЛЗ), на магнитных барабанах (МБ) и электроннолучевых трубках (ЭЛТ). ЗУ на ЭЛТ далее были полностью вытеснены другими типами ЗУ, в то время как ЗУ на ЛЗ и МБ в том или ином виде используются до настоящего времени.

Потребность в надежных и экономичных ЗУ большой емкости и низкой стоимости явилась причиной широких исследований в этой области, которые привели к созданию в 1951 г. ЗУ на сердечниках с ППГ. Благодаря высокому быстродействию, экономичности, надежности и малым габаритам они заняли ведущее место в ЭВМ и только в последние годы стала проявляться тенденция их замены ЗУ на полупроводниковых приборах. Современное состояние ЗУ характеризует табл. 71, где ПП — полупроводниковые приборы; ТМП — тонкие магнитные пленки; ФС — ферритовые сердечники; МД, МБ, МК, МЛ — магнитные диски, барабаны, карты, ленты; ФД — фотоскопические диски; t_0 — цикл обращения; v — скорость чтения; N — емкость; C — приведенная стоимость.

Таблица 71

	t_0 , мкс	v , МГц	N , бит	C , дол./бит
ПП	0,01—0,5	2—20	$< 10^6$	0,01—10
ТМП	0,1—0,5	2—12	10^3 — 10^5	0,5—2
ФС	0,3—4	0,2—3	10^4 — 10^8	$(5-2,5) \cdot 10^{-2}$
МБ	10^3	0,1—1	10^8	10^{-2} — 10^{-3}
МД	$(2-20) 10^3$	0,1—1	10^9	10^{-3} — 10^{-4}
МК	10^6	0,02—0,2	10^9	10^{-4} — 10^{-5}
ФД	10^3	0,02—0,2	10^{12}	10^{-6}

Важность улучшения технико-экономических показателей поддерживает высокие темпы исследований в области создания новых типов ЗУ. В настоящее время намечился ряд перспективных областей таких исследований.

Первая попытка использовать явление сверхпроводимости в ЗУ ЭВМ относится к 1955 году. С тех пор это направление считается перспективным, однако стоимость холодильного оборудования пока еще существенно превышает стоимость ЗУ. В первую очередь криотронные устройства предполагаются использовать для построения ассоциативных и адресных ЗУ большой емкости (10^8 — 10^9 бит).

Другим перспективным направлением в технике ЗУ является использование оптических явлений. Полупостоянные ЗУ такого типа (см. п. 4.8) уже успешно конкурируют с ЗУ на магнитных лентах, барабанах, дисках.

Много надежд разработчики памяти цифровых ЭВМ связывают с возможностью создания запоминающих устройств биологического типа. Память живых организмов и прежде всего человека по ряду основных показателей — гибкости, емкости, экономичности, быстродействию при данной емкости, удельной емкости — намного превосходит лучшие технические ЗУ. В связи с этим представляет интерес хотя бы в общих чертах рассмотреть основные свойства биологической памяти (БП).

В биологических системах различают три вида БП: генетическую (хранит наследственную информацию), программную (хранит программу «врожденного» поведения организма в окружающей среде) и «прижизненную» память, которая обеспечивает накопление опыта взаимодействия с окружающей средой и создает возможности для самообучения, приспособления и «разумного» поведения живых организмов. Прежде всего в БП чтение и запись осуществляется не по адресному, а по ассоциативному принципу, т. е. по связи вновь поступающей информации с информацией, уже хранящейся в памяти. Этим объясняются такие явления, как лучшая запоминаемость новой информации при наличии в БП сходной информации. Например, математик лучше запоминает новые математические понятия, чем человек, не знающий этой дисциплины.

Важным положительным свойством БП является то, что в ней совмещаются функции хранения и переработки информации. Многократное чтение записанной информации в БП закрепляет ее, в то время, как в технической памяти (ТП) это приводит к разрушению информации. При этом общее количество воспроизводимой при чтении из БП информации всегда меньше, чем количество воспринимаемой информации. В БП происходит сжатие информации и уменьшение ее избыточности, например, за счет фильтрации малозначащей информации. Время обращения к БП, в отличие от ТП, сильно зависит от самой информации и от ее важности. Существует много различных способов оценки емкости БП. Например, если емкость памяти человека оценить по способности усваивать 10^6 слов языка, то это даст минимальную емкость N порядка $4 \cdot 10^6$ бит. Исходя из предположения, что человек в течение 80 лет жизни по 16 часов ежедневно запоминает информацию со скоростью в 25 бит/с, получим $N \approx 4 \cdot 10^{10}$ бит. Если предположить, что количество нейронов в БП порядка 10^{10} и у каждого нейрона имеется 30 синапсов, каждый из которых может находиться по крайней мере в 10 состояниях, то это даст $N \approx 10^{12}$ бит. Если же исходить из того, что объем мозга равен 10^3 см³, а объем одной его молекулы равен 10^{-18} см³ и что каждая молекула может хранить один бит, то получим $N = 10^{21}$. Таким образом в зависимости от принятой методики объем наиболее совершенной БП (памяти человека) в битах оценивается величиной $10^7 < N < 10^{21}$. Истинная же емкость БП, по-видимому, лежит в середине указанного интервала, $N \approx 10^{14}$ бит. Всю огромность этого числа легко представить, если учесть, что общее количество информации, накопленной в виде печатных изданий и хранящейся в Государственной публичной библиотеке им. В. И. Ленина, составляет величину порядка 10^{13} бит.

Экономичность БП можно охарактеризовать следующим сравнением. Запоминающее устройство емкостью порядка 10^7 бит потребляет мощность порядка сотен ватт, в то время как БП при $N \approx 10^{14}$ бит расходует несколько десятков ватт, а в расчете на один бит БП потребляет мощность в 10^8 раз меньше, чем ТП. При этом во столько же раз плотность записи информации в БП больше, чем плотность записи данных в ТП (10^{11} и 10^3 бит/см³).

Надежность функционирования БП характеризуется двумя факторами. С одной стороны, БП свойственно самопроизвольное стирание (забывание) информации. С другой стороны, в БП отсутствует жесткая пространственная локализация информации, которая запоминается не отдельной группой нейронов, а как бы всей совокупностью элементов БП. Этим можно объяснить сохранение запомненной ранее информации и функций БП при повреждениях или удалении значительных участков мозга, например, при хирургических операциях.

Для реализации функций памяти в технике имеются три возможности: изменение состояний ЗЭ, изменение связей между ЗЭ и создание рециркуляционных процессов в замкнутых контурах. В БП, по-видимому, действуют все три указанных механизма, однако принципы их реализации до сих пор остаются в значительной мере еще не выясненными. По современным представлениям БП состоит из двух частей: оперативной (ОЧ) и долговременной (ДЧ). Внешняя информация воспринимается рецепторами (Р), которые кодируют ее частотно-импульсным кодом. Поступление этого кода приводит к возникновению в замкнутых контурах нервной сети рециркуляционных процессов (РП), которые представляют собой первую фазу запоминания. При достаточной длительности РП приводят к структурным изменениям в долговременной части БП. Природа этих изменений окончательно еще не выяснена. Некоторые специалисты полагают, что основой долговременной памяти является образование межнейронных связей. Воспроизведение информации из долговременной памяти снова связано с возникновением в оперативной части БП рециркуляционных процессов, возбуждающих эффекторы (Э), через которые осуществляется вывод информации, например, посредством речевого аппарата.

Вряд ли можно с уверенностью сказать, по какому пути пойдет дальнейшее развитие ЗУ, однако интенсивность исследований в этой области убеждает в том, что в ближайшее время будут созданы ЗУ сверхвысокой емкости (10^{12} — 10^{16} бит). Создание ЗУ такой емкости существенно повлияет на организацию и использование цифровых ЭВМ.

АРИФМЕТИКО-ЛОГИЧЕСКИЕ УСТРОЙСТВА И ОПЕРАЦИОННЫЕ БЛОКИ ЦИФРОВЫХ ЭВМ

5.1. НАЗНАЧЕНИЕ И КЛАССИФИКАЦИЯ, СОСТАВ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ АРИФМЕТИКО-ЛОГИЧЕСКИХ УСТРОЙСТВ

Арифметико-логические устройства (АЛУ) являются частью процессоров и предназначены для выполнения операций над машинными словами (т. е. числами, командами и кодами других объектов). Набор операций, выполняемых АЛУ, должен обладать функциональной полнотой для того, чтобы с их помощью можно было реализовать любой вычислительный алгоритм. Минимальный набор операций в цифровых ЭВМ, обладающий свойством полноты, включает всего четыре операции: пересылку информации из любой ячейки памяти в любую другую ячейку, сложение слова с $+1$ или -1 , условный переход по совпадению слов и безусловный останов. Однако с целью повышения быстродействия и упрощения программирования указанный набор обычно обладает значительной избыточностью, а число операций колеблется от нескольких операций (16 в АЛУ микропроцессоров) до нескольких сотен. При всем разнообразии наборов операций в их составе всегда имеются четыре основные арифметические операции и наиболее важные логические операции, такие как сравнение, поразрядная конъюнкция и др. Набор операций, выполняемых АЛУ, является важнейшей его характеристикой. К временным характеристикам АЛУ относятся производительность по Гибсону (см. п. 1.3), среднее время выполнения отдельных операций t_i и среднее время выполнения операций в АЛУ

$$t_0 = \sum_{i=1}^m t_i \gamma_i,$$

где γ_i — вес i -й операции (по Гибсону).

Любое АЛУ имеет в своем составе один или несколько регистров, сумматор и блок местного управления (БМУ), вырабатывающий необходимые последовательности управляющих сигналов.

По способу обмена информацией между регистрами и сумматором АЛУ делятся на последовательные, параллельные и последовательно-параллельные. В универсальных ЭВМ используют, как правило, параллельные АЛУ, так как их применение позволяет более чем в n раз повысить быстродействие по сравнению с последовательными АЛУ при увеличении аппаратных затрат меньше, чем в два раза. Последовательные и последовательно-параллельные АЛУ применяются в малых и специализированных цифровых ЭВМ в тех случаях, когда к быстродействию машин не предъявляется жестких требований.

По форме представления чисел различают АЛУ с плавающей запятой, АЛУ с фиксированной запятой и АЛУ, работающие как с пла-

вающей, так и с фиксированной запятой. В некоторых цифровых ЭВМ предусмотрен режим целых чисел, при котором запятая фиксируется после последнего разряда числа. Такой способ представления чисел удобен при решении экономических задач.

В зависимости от принятой системы счисления АЛУ разделяют на двоичные, троичные, десятичные, АЛУ в СОК, в избыточных системах, в системах Фибоначчи, в системах с искусственным порядком весов и т. п. Известны также АЛУ, работающие в системах счисления с основанием 2^k ($k = 2, 3, 4, \dots$) — восьмиричные, шестнадцатиричные и др.

По способу ввода входной информации АЛУ подразделяют на автономные и неавтономные. Автономные АЛУ в процессе функционирования не приобретают внешней информации. Такой режим является основным для АЛУ большинства универсальных цифровых ЭВМ. Неавтономные АЛУ в процессе работы получают информацию об уточненных значениях операндов через внешние каналы. Применяют неавтономные АЛУ в высокопроизводительных специализированных цифровых ЭВМ.

В зависимости от времени выполнения отдельных операций различают АЛУ синхронного и асинхронного типа. В синхронных АЛУ на выполнение различных операций отводится одно и то же время. В асинхронных АЛУ на выполнение каждой операции отводится столько тактов машинного времени, сколько требуется при каждой конкретной ее реализации, а следующая операция начинается лишь по сигналу конца текущей операции. Комбинированные АЛУ сочетают простоту синхронных и быстродействие асинхронных АЛУ. При этом все операции делятся на несколько групп, чаще всего на две: одно- и многотактные. Однотактные операции реализуются по синхронному способу, а многотактные — по асинхронному.

По структуре АЛУ разделяют на универсальные малорегистровые и многорегистровые (с закрепленными микрооперациями), блочного типа, магистральные (с общими микрооперациями, с сосредоточенной памятью и логикой), конвейерные. Более детально структурные отличия АЛУ рассмотрены в п. 5.12.

В АЛУ любого типа выполнение какой-либо операции сводится к выполнению последовательности микроопераций на операционных узлах. К таким микрооперациям относятся: B_1 — выдача прямого кода слова с некоторого операционного узла с памятью; B_2 — выдача инверсного кода слова; B_3 — выдача парафазного кода слова; P_0 — установка в «0» операционного узла (прием нулевого слова, гашение); P_1 — прием прямого кода слова на операционный узел; P_2 — прием инверсного кода слова; P_3 — прием парафазного кода слова; C_{1i} — сдвиг слова на i разрядов вправо; C_{2i} — сдвиг слова на i разрядов влево; P_i — прибавление (R_i — вычитание) константы i к слову. Далее в условных обозначениях микроопераций будем указывать и операционные узлы, на которых выполняются микрооперации. Правила образования таких условных обозначений микроопераций с использованием нотации Бэкуса (см. п. 1.2) могут быть записаны так:

⟨условное обозначение микрооперации⟩: := ⟨микрооперация⟩
идентификатор операционного узла, ⟨микрооперация⟩: := ⟨буква⟩

⟨индекс⟩|⟨буква⟩ ⟨индекс⟩ ⟨индекс⟩, ⟨идентификатор операционного узла⟩: : = ⟨идентификатор⟩ ⟨номер⟩, ⟨буква⟩: : = $P|B|C|D|R|P|S|T|U$, ⟨индекс⟩: : = $0|1|2|3|4$, ⟨идентификатор⟩: : = $RG|SM|CT|CD|TЗ$, ⟨номер⟩: : = $A|B|C|H|n|m|n|X|X'|Y|y|Z|Z'|z|\Delta$.

Например, условные обозначения P_0CTA , $C_{24}RGY$, R_1CTC соответствуют микрооперациям установки в нуль счетчика A , сдвига влево содержимого регистра Y на 4 разряда, вычитания «1» из слова на счетчике C .

Последовательность микроопераций, выполнение которой приводит к выполнению заданной операции, будем называть микроалгоритмом этой операции. В зависимости от способа реализации микроалгоритмов различают АЛУ со схемным и микропрограммным способами управления операциями.

В АЛУ со схемным управлением БМУ состоит из отдельных схем, каждая из которых управляет одной операцией. При этом некоторые узлы могут быть общими для всех схем. В БМУ из УУ машины поступают сигналы начала выполнения i -й операции H_i , а из БМУ в УУ поступают сигналы об окончании выполнения операции K_i . При микропрограммном способе управления операциями одной или несколькими микрооперациями ставят в соответствие некоторое слово, которое называют микрокомандой. Последовательности микроопераций, т. е. микроалгоритму, соответствует последовательность микрокоманд, которую называют микропрограммой данной операции. Микропрограммы операций хранятся в специальном ЗУ микропрограмм. Код операции задает адрес первой микрокоманды данной микропрограммы. Сущность микропрограммного способа управления операциями состоит в последовательном считывании из ЗУ и расшифровке микрокоманд, входящих в микропрограмму данной операции.

Наряду со схемным и микропрограммным способами существует также смешанное управление операциями, при котором часть микроопераций реализуется схемным, а другая — микропрограммным способом. Каждый из перечисленных способов имеет свои преимущества и недостатки. В цифровых ЭВМ малой и средней производительности используют преимущественно АЛУ с микропрограммным управлением. АЛУ со схемным управлением чаще применяются в больших цифровых ЭВМ и специализированных вычислительных средствах.

Далее для записи микроалгоритмов будем использовать графы и язык логических схем алгоритмов (операторные описания, см. п. 1.2), дополненный оператором безусловного перехода. Операторами такого языка являются микрооперации, а логические условия в микроалгоритмах будем записывать следующим образом:

⟨логическое условие⟩: : = ⟨буква⟩ ⟨индекс⟩ ⟨стрелка⟩ ⟨индекс⟩; ⟨буква⟩: : = $A|C|c|\bar{c}|E|F|L|R|S|s|T|X|x|\bar{X}|\bar{x}|x'|x'|z|\Delta$, ⟨стрелка⟩: : = $\uparrow$, ⟨индекс⟩: : = $0|1|\dots|9|n|n+1|n+2|\Delta$.

Например, запись $x_n \uparrow^2$ означает проверку логического условия x_n : если $x_n = 1$, то необходимо перейти к выполнению микрооперации или проверке условия, указываемого стрелкой $\downarrow^2$; если же $x_n = 0$, то далее выполняется следующая по порядку микрооперация или проверяется следующее по порядку логическое условие. Стрелка с индексом после микрооперации означает безусловный переход. За

метим, что на рассматриваемом языке микроалгоритмы можно записывать и без использования оператора безусловного перехода. Однако его наличие существенно упрощает записи ветвящихся микроалгоритмов. Микрооперации и логические условия, выполнение или проверку которых можно совместить во времени, будем записывать в круглых скобках. Для удобства чтения микроалгоритмов их начало и конец будем указывать с помощью символов H и K , а соседние микрооперации или логические условия — отделять пробелами. Важной отличительной особенностью рассматриваемого варианта операторных описаний является взаимно однозначное соответствие между условными обозначениями микроопераций и сигналами в цепях управления цифровыми ЭВМ, в то время как в других формальных языках, используемых для этих целей, операторам могут соответствовать последовательности управляющих сигналов.

Независимо от структуры АЛУ можно считать, что при выполнении некоторой операции оборудование АЛУ настраивается на ее реализацию, т. е. коммутируется в определенный операционный блок. Поэтому рассмотрим вначале алгоритмы выполнения отдельных операций в АЛУ и схемы организации соответствующих им операционных блоков.

Выполнение любой операции в АЛУ начинается с подготовительных микроопераций извлечения операндов из ЗУ и фиксации их в определенном порядке в регистрах АЛУ. Однако, с целью упрощения изложения, работа операционных блоков будет рассматриваться начиная с момента, когда все подготовительные микрооперации по извлечению и фиксации операндов завершены.

5.2. БЛОКИ ДЛЯ СЛОЖЕНИЯ И ВЫЧИТАНИЯ ЧИСЕЛ В ПРЯМЫХ И ДОПОЛНИТЕЛЬНЫХ КОДАХ

В ЗУ цифровых ЭВМ числа хранят в прямом или дополнительном кодах. В обратном коде числа хранить нецелесообразно, так как в случае необходимости обратный код может быть легко получен из прямого поразрядным инвертированием, для чего не требуется дополнительных затрат оборудования и времени. Из ЗУ извлекаются операнды и пересылаются в регистры АЛУ в том виде, в котором они хранились в ЗУ. Поэтому в зависимости от того, в каких кодах хранятся числа в ЗУ, различают операции в прямых и дополнительных кодах.

При проектировании суммирующих блоков (СБ) для сложения и вычитания чисел в прямых кодах основные затруднения связаны с реализацией операции вычитания. Если в СБ не используется специальная операционная схема-вычитатель (ее построение во многом аналогично построению сумматора), то вычитание реализуют путем его замены операцией сложения прямого кода одного операнда с дополнительным или обратным кодом второго операнда. При использовании дополнительного кода (ДК) результат операции также будет представлен ДК. Так как ДК положительного числа совпадает с прямым кодом этого же числа, то прямой код результата, необходимый для записи в ЗУ, получают тривиально. При отрицательном же ре-

результате для образования прямого кода необходимо провести его поразрядное инвертирование и добавить 1 в младший разряд. По времени это равноценно суммированию. Следовательно, для получения результата необходимо время $(1 + W)t$, где W — вероятность получения отрицательного результата; t — время однократного суммирования. Вследствие того, что фактическая задержка, вносимая СБ, увеличивается в $1 + W$ раз по сравнению с t , ДК не используется при выполнении операций в прямых кодах.

Алгоритм сложения-вычитания чисел в прямых кодах с использованием обратных кодов для вычитания состоит в следующем:

1. Если знак операнда X положительный (т. е. $X_0 = 0$), то подать на сумматор SM прямой код числа X . Если $X_0 = 1$, то подать на сумматор инверсный код X .

2. Если знак Y_0 второго операнда Y и знак S_0 заданной операции совпадают, то подать на SM прямой код Y . Если же $Y_0 \neq S_0$, то подать на SM инверсный код Y .

3. Образовать прямой код результата.

Сумматор в данном случае должен иметь цепь циклического переноса и оперировать с двумя знаковыми разрядами так же, как и с основными разрядами, а результат получается всегда в обратном коде. Так как обратный код положительного числа совпадает с его прямым кодом, то формировать прямой код результата следует только в случае получения отрицательного результата. Из приведенного алгоритма следует, что суммирующий блок должен в себя включать три параллельных регистра RGX , RGY , RGZ и сумматор SM . Выходы RGX и RGY должны быть подключены ко входам SM , а выходы сумматора — ко входам RGZ .

В соответствии с вышеизложенным микроалгоритм сложения-вычитания (MAC) в прямых кодах с использованием обратных кодов можно записать так:

$$MAC = H(X_0 \uparrow {}^1B_1RGX \uparrow {}^2\downarrow {}^1B_2RGX \downarrow {}^2F_0 \uparrow {}^3B_2RGY \uparrow {}^4\downarrow {}^3B_1RGY \downarrow {}^4) \\ (P_1RGZ)(Z_0 \uparrow {}^5B_1RGZ \uparrow {}^6\downarrow {}^5B_2RGZ \downarrow {}^6)K. \quad (5.1)$$

Здесь F_0 — логическое условие: $F_0 = 1$, если $Y_0 = S_0$ и $F_0 = 0$, если $Y_0 \neq S_0$. Легко заметить, что проверка этого условия может быть осуществлена комбинационной схемой, реализующей функцию

$$F_0 = \overline{Y_0 S_0} \vee \overline{Y_0} S_0 = Y_0 S_0 \vee \overline{Y_0} \overline{S_0}.$$

На рис. 5.1 приведен граф, соответствующий микроалгоритму (5.1). Совместимые во времени микрооперации здесь указаны в общей прямоугольной рамке.

Интерпретация рассматриваемого алгоритма сложения-вычитания в виде MAC (5.1) не является единственно возможной. Вид микроалгоритмов зависит от системы микроопераций, реализуемых используемыми операционными узлами. Приведенная запись ориентирована на использование в СБ регистров, реализующих микрооперации B_1 , B_2 и P_1 . Параллельные регистры на потенциальных элементах часто

могут реализовать набор микроопераций B_1, B_2, Π_1, Π_2 . В этом случае MAC можно записать так

$$MAC = H(X_0 \uparrow {}^1B_1RGX \uparrow {}^2 \downarrow {}^1B_2RGX \downarrow {}^2F_0 \uparrow {}^3B_2RGY \uparrow {}^4 \downarrow {}^3B_1RGY \downarrow {}^4 Z_0 \uparrow {}^5\Pi_1RGZ \uparrow {}^6 \downarrow {}^5\Pi_2RGZ \downarrow {}^6)K. \quad (5.2)$$

При сложении-вычитании чисел по MAC (5.2) в регистре RGZ будет получен прямой код результата, который можно записать в ЗУ.

На рис. 5.2 приведена структурная схема суммирующего блока, реализующего MAC (5.1). БМУ обеспечивает проверку логических условий X_0, Y_0, F_0 и формирует управляющие сигналы, по которым выполняются микрооперации в соответствии с MAC. Кроме того, БМУ формирует сигналы окончания операции K и переполнения раз-

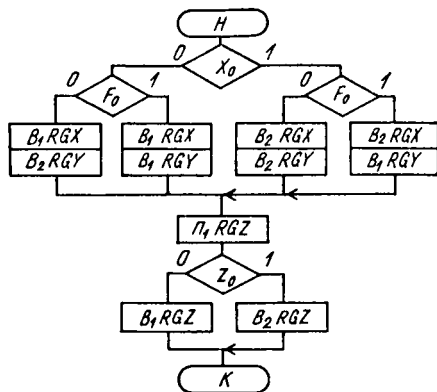


Рис. 5.1

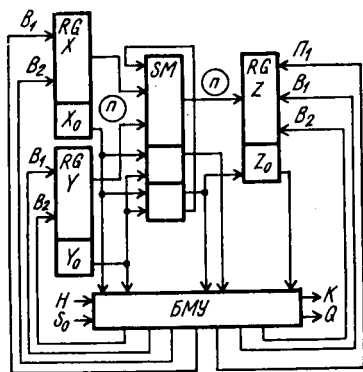


Рис. 5.2

рядной сетки Q . Как известно (см. гл. 1), сигнал $Q = 1$ фиксируется при появлении комбинаций 01 и 10 в знаковых разрядах сумматора. При этом знак результата определяется по первой слева цифре. Время сложения-вычитания двух чисел X и Y на рассматриваемом СБ может быть оценено как

$$t_+ \geq t_b + t_s + t_n,$$

где t_b и t_n — задержки сигналов в цепях выполнения микроопераций B и Π ; t_s — задержка сигналов на сумматоре.

Заметим, что в случае использования регистров на основе триггеров с внутренней задержкой функции RGZ может выполнять один из регистров операндов. В СБ на основе накапливающего сумматора функции RGX, RGZ и собственно сумматора реализуются на одном и том же оборудовании. Прямой код операнда X из ЗУ записывается в накапливающий сумматор SMH . Далее сложение-вычитание выполняется в соответствии со следующим микроалгоритмом

$$MAC = H(\bar{X}_0 \uparrow {}^1U_0SMH \downarrow {}^1)(F_0 \uparrow {}^2B_2RGY \uparrow {}^3 \downarrow {}^2B_1RGY \downarrow {}^3\Pi_1SMH)(D_0SMH)(Z_0 \uparrow {}^4B_1SMH \uparrow {}^5 \downarrow {}^4B_2SMH \downarrow {}^5)K, \quad (5.3)$$

где U_0SMH — микрооперация поразрядного инвертирования слова на SMH (без знакового разряда); D_0SMH — микрооперация разре-

шения прибавления переноса в SMH . Микрооперация U_0 может быть реализована подачей единичных сигналов на Т-входы триггеров в каждом разряде SMH . Если при выполнении сложения-вычитания по МАС (5.3) результат очередной операции не отсылается в ЗУ, а используется в качестве операнда X для следующей операции, то выполнять микрооперации B_1SMH , B_2SMH и проверять условие Z_0 нет необходимости.

Из изложенного видно, что аппаратные затраты на построение СБ, функционирующих по МАС (5.1) — (5.3), без учета сложности БМУ, могут быть в общем случае оценены как

$$C = \alpha n + \beta,$$

где значения коэффициентов α и β зависят как от используемого набора операционных узлов и реализуемой ими системы микроопераций, так и от особенностей построения самих узлов и применяемой системы элементов. Приблизительно оценить α и β можно следующим образом. Пусть a_1 — стоимость одного разряда регистра; a_2 — суммарная стоимость одного разряда цепей реализации микроопераций B_1 и B_2 ; a_3 — стоимость одного разряда цепей реализации микрооперации P_1 ; a_4 — стоимость одного разряда сумматора. Тогда, например, для схемы на рис. 5.2

$$\begin{aligned}\alpha &= 3(a_1 + a_2) + a_3 + a_4; \\ \beta &= 3(a_1 + a_2) + a_3 + 2a_4.\end{aligned}$$

Если числа в ЗУ хранятся в дополнительном коде, то в этом же коде они пересылаются в регистры СБ. Алгоритм сложения-вычитания в этом случае предельно прост:

1. Подать на SM код числа X .

2. Если $S_0 = 0$, то подать на SM код числа Y . Если $S_0 = 1$, то подать на SM инверсный код числа Y и 1 переноса в младший разряд.

В данном случае результат сложения-вычитания всегда будет в дополнительном коде, и никаких преобразований его перед записью в ЗУ не требуется. Составим микроалгоритм сложения-вычитания при использовании тех же операционных узлов, что и МАС (5.1):

$$\begin{aligned}MAC &= H(B_1RGX \ S_0 \uparrow \ ^1B_1RGY \uparrow \ ^2 \downarrow \ ^1B_2RGY \ P_1SMn \downarrow \ ^2) \\ &\quad (P_1RGZ) K.\end{aligned}\tag{5.4}$$

Здесь P_1SM_n — микрооперация прибавления константы 1 в младший (n -й) разряд сумматора.

Схема СБ с комбинационным сумматором, соответствующая МАС (5.4), показана на рис. 5.3. Быстродействие данного СБ такое же, как и у СБ на рис. 5.2, однако аппаратные затраты несколько ниже как за счет отсутствия цепей выдачи инверсного кода с регистров RGX и RGZ , так и за счет упрощения БМУ.

Так же как и в параллельных блоках, в последовательных СБ операции могут производиться в прямых и дополнительных кодах. При операциях в прямых кодах для реализации вычитания в свою очередь могут быть использованы как дополнительные коды, так и обратные. В первом случае с некоторой вероятностью результат может получаться в дополнительном коде. Поэтому в среднем потре-

буется $(1 + W)$ тактов суммирования n -разрядных чисел. При использовании обратных кодов для реализации циклического переноса также требуется два цикла суммирования. Вследствие этого операции в прямых кодах реализовать в СБ последовательного типа нецелесообразно.

Микроалгоритм сложения-вычитания чисел, представленных последовательными дополнительными кодами, отличается от MAC (5.4) тем, что микрооперации выдачи слов с регистров RGX и RGY и приема слов на RGZ заменяются выполнением $n + 2$ микроопераций сдвига вправо содержимого соответствующих регистров, т. е.

$$\begin{aligned} MAC = & H(P_0CTC)(S_0 \uparrow^1 \downarrow^5 C_1RGX \ C_1RGY \ C_1RGZ \ B_1RGY \uparrow^2 \\ & \downarrow^1 P_1SMn \ \downarrow^4 C_1RGX \ C_1RGY \ C_1RGZ \ B_2RGY \downarrow^2 P_1CTC) \\ & C_{n+2} \uparrow^3 S_0 \uparrow^4 \uparrow^5 \downarrow^3 K, \end{aligned}$$

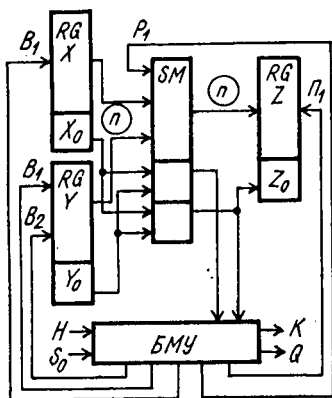


Рис. 5.3

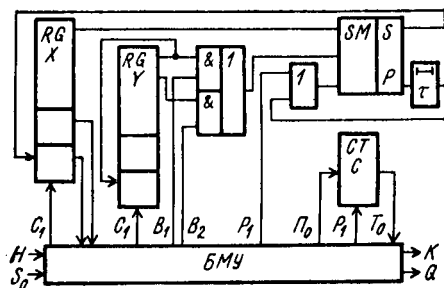


Рис. 5.4

где P_0CTC и P_1CTC — микрооперации установки в 0 и прибавления 1 к содержимому счетчика сдвигов CTC ; B_1RGY и B_2RGY — микрооперации выдачи содержимого младшего разряда RGY прямым или инверсным кодом; C_{n+2} — логическое условие «содержимое CTC равно $n + 2$ ». Заметим, что микрооперация P_1SMn выполняется только в первом такте работы СБ при вычитании ($S_0 = 1$). Схема СБ последовательного действия показана на рис. 5.4. Здесь в качестве RGZ используется RGX . БМУ выдает $n + 2$ сигналов C_1 , после чего вырабатывается сигнал K , и в случае необходимости — сигнал переполнения Q . Следовательно, для такого СБ

$$t_+ \geq (n + 2)(t_s + t_c),$$

где t_c — время сдвига слова в регистре на один разряд. Если обозначить a_5 — стоимость одного логического элемента или элемента задержки, a_6 — стоимость одного разряда цепей сдвига; a_7 — стоимость одного разряда счетчика, то без учета сложности БМУ аппаратурные затраты на последовательный СБ составят

$$C = \alpha n + \beta \log_2(n + 2) + \gamma$$

условных единиц, где

$$\alpha = 2(a_1 + a_8); \beta = a_7; \gamma = 4a_1 + a_4 + 5a_5 + 4a_6.$$

Сравнивая полученную оценку с аппаратурными затратами, необходимыми для реализации СБ параллельного типа, можно сделать вывод, что аппаратурные затраты в последовательном СБ уменьшаются всего лишь примерно в два раза, в то время как быстродействие падает более чем в n раз.

5.3. БЛОКИ ДЛЯ СЛОЖЕНИЯ И ВЫЧИТАНИЯ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Число X , представленное в форме с плавающей запятой, имеет вид

$$X = k^A M(X),$$

где A — порядок числа; $M(X)$ — мантисса; k — основание системы счисления.

Число X называется нормализованным, если его мантисса удовлетворяет условию

$$\frac{1}{k} \leq M(X) < 1$$

или при $k = 2$

$$\frac{1}{2} \leq M(X) < 1.$$

Производить сложение-вычитание чисел в форме с плавающей запятой можно лишь в том случае, если у чисел одинаковые порядки. Поэтому операция сложения-вычитания начинается с выравнивания порядков чисел. После суммирования мантисс возможно нарушение нормализации числа как влево на один разряд, так и вправо на любое число разрядов. Поэтому после выполнения суммирования производится нормализация результата. Нормализованный результат не всегда помещается в разрядной сетке машины. Поэтому далее необходимо выполнить округление результата и окончательную проверку факта нормализации результата. Следовательно, выполнение операций сложения и вычитания чисел с плавающей запятой состоит из следующих этапов:

1. Выравнивание порядков.
2. Суммирование мантисс.
3. Определение порядка результата.
4. Нормализация результата.
5. Округление результата.
6. Окончательная нормализация результата.

Рассмотрим каждый из этих этапов отдельно.

Выравнивание порядков осуществляется путем сдвига вправо числа, порядок которого меньше. Так как сдвиг вправо эквивалентен делению мантиссы на k , то при каждом сдвиге порядок должен увеличиваться на 1.

При выравнивании порядков необходимо прежде всего определить, порядок какого числа меньше. Это можно осуществить с помощью специальной схемы сравнения. Но, так как при выполнении

умножения и деления порядки чисел складываются и вычитаются, то в АЛУ всегда имеется сумматор порядков. Поэтому для определения числа с меньшим порядком используют этот сумматор, на котором из порядка первого числа вычитают порядок второго числа. По знаку разности определяют больший порядок, а абсолютная величина разности позволяет определить необходимое число сдвигов. Так как над порядками производятся только операции сложения или вычитания, то для представления их удобно использовать дополнительные коды. Таким образом алгоритм выравнивания порядков чисел X и Y сводится к следующему:

1. Подать на сумматор порядков $SMП$ прямой код порядка операнда X , инверсный код порядка операнда Y и 1 переноса в младший разряд $SMП$. Сформированную таким образом разность порядков записать на счетчик (регистр).

2. Если разность порядков Δ положительная, то необходимо сдвигать вправо мантиссу числа Y . Если разность порядков Δ отрицательная, то сдвигать вправо надо мантиссу числа X .

3. Если $\Delta > 0$, то при каждом сдвиге из разности порядков необходимо вычитать 1, а если $\Delta < 0$, то прибавлять 1.

4. Выравнивание порядков завершается, когда содержимое счетчика (регистра) разности порядков станет равным 0.

Из приведенного алгоритма выравнивания порядков следует, что блок, реализующий этот алгоритм, должен состоять из регистров порядков RGA , RGB , сумматора порядка $SMП$ и счетчика разности $CT\Delta$. Составим микроалгоритм выполнения операции выравнивания порядков на перечисленных узлах

$$MABП = H (B_1 RGA \ B_2 RGB \ P_1 SMm) (П_1 CT\Delta) (\downarrow^3 \Delta_0 \uparrow^1 S_\Delta \uparrow^2 C_1 RGY \ R_1 CT\Delta \uparrow^3 \downarrow^2 C_1 RGX \ P_1 CT\Delta \uparrow^3 \downarrow^1) K.$$

Здесь Δ_0 и S_Δ — логические условия «содержимое $CT\Delta$ равно «0» и «знак Δ ». Кроме того, предполагается, что $SMП$ имеет m разрядов. Схема блока, реализующего этот $MABП$, показана на рис. 5.5. После выравнивания порядков осуществляется сложение-вычитание мантисс, которое в принципе ничем не отличается от рассмотренных ранее в п. 5.2 способов сложения-вычитания чисел с фиксированной запятой. Результату присваивается порядок большого по абсолютной величине операнда, т. е., того, мантисса которого оставалась неподвижной при выравнивании порядков. Установить этот факт можно с помощью триггера $T1$ на рис. 5.5.

Сложение-вычитание мантисс может привести к тому, что результат окажется ненормализованным. Если, например, вычитание выполняется с использованием модифицированного обратного кода, то левое нарушение нормализации определяется по комбинациям цифр 01 и 10 в знаковых разрядах SM , а правое — по комбинациям цифр 00,0 и 11,1 в старшем и знаковых разрядах SM . При сложении-вычитании мантисс в дополнительном коде правое нарушение нормализации обнаруживается по комбинациям 00,0 и 11,11, а левое — так же как и в предыдущем случае. Осуществляется нормализация результата путем сдвига мантиссы результата влево (или вправо) и вычитания (или прибавления) 1 к порядку S результата при каждом сдвиге

результата до тех пор, пока будут сохраняться указанные комбинации цифр. Составим микроалгоритм нормализации:

$$MAN = H(Z_1 \uparrow^1 L_1 \uparrow^2 \downarrow^4 R_1 \uparrow^3 \uparrow^1)(\downarrow^2 C_1 RGZ \ P_1 RGC \uparrow^1 \downarrow^3 \\ C_2 RGZ \ R_1 RGC \uparrow^4) \downarrow^1 K.$$

Здесь Z_1 , L_1 , R_1 — логические условия соответственно «результат сложения-вычитания равен нулю», «левое нарушение нормализации», «правое нарушение нормализации». Схема операционного блока, производящего нормализацию результата сложения-вычитания в дополнительном коде, приведена на рис. 5.6.

При нормализации сдвиг вправо результата и отбрасывание его младшего разряда может привести к накоплению большой положительной ошибки. Для исключения этого вводят операцию округления. С этой

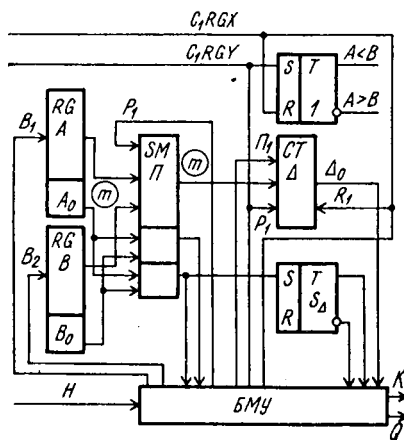


Рис. 5.5

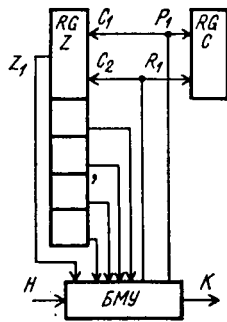


Рис. 5.6

целью SM и RGZ имеют дополнительный разряд со стороны младших разрядов. Округление осуществляется добавлением 1 в этот разряд с последующим отбрасыванием его содержимого. Если в дополнительном разряде была 1, то возникает 1 переноса в младший основной разряд. Вследствие этого погрешность будет знакопеременной и не будет больше $\frac{1}{2}$ веса младшего разряда.

Рассмотрим структурную схему устройства для сложения-вычитания чисел с плавающей запятой (рис. 5.7). Блок выравнивания порядков (БВП) выдает сигнал правого сдвига на RGX или RGY , находящиеся в блоке суммирования мантисс (БСМ), а также порядок результата C . По сигналу H из БВП запускается БСМ. Затем вступает в работу блок нормализации результата (БНР), нормализующий мантиссу результата (Z). По первому сигналу K из БНР производится округление результата, а по второму — с помощью триггера T вырабатывается сигнал K сложения чисел с плавающей запятой.

Среднее время выполнения операции сложения-вычитания рассматриваемым устройством составит

$$t_+ = t_v + t_m + t_{н1} + t_0 + t_{н2},$$

где $t_b, t_m, t_{n1}, t_0, t_{n2}$ — средние времена выравнивания порядков, сложения мантисс, первой нормализации, округления и второй нормализации. Определим каждое из них.

Среднее время выравнивания порядков равно

$$t_b = t_{s1} + t_c M,$$

где t_{s1} — время вычитания порядков; t_c — время сдвига; M — среднее число сдвигов. В свою очередь

$$M = \sum_{i=0}^{2q} i W \{ |\Delta| = i \}.$$

Здесь и далее $W \{A\}$ обозначает вероятность события A . В данном случае $W \{ |\Delta| = i \}$ — вероятность того, что абсолютная величина

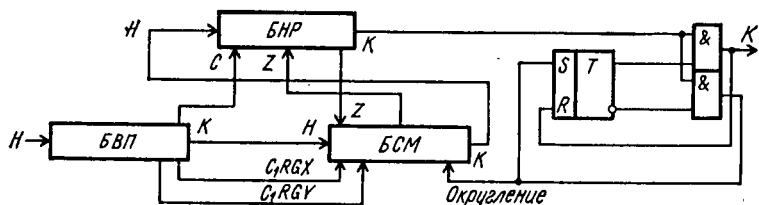


Рис. 5.7

разности порядков Δ равна i ; q — абсолютная величина максимально возможного порядка

$$q = \lceil \log_k N \rceil,$$

где k — основание системы счисления; N — величина, определяющая диапазон представления чисел, т. е.

$$\frac{1}{N} \leq |X| \leq N.$$

Предположим, что порядки A и B операндов статистически независимы и одинаково распределены. Событие $\{ |\Delta| = i \}$ может произойти, если имеет место одна из следующих пар событий:

$$\begin{aligned} & \{A = q\}, \quad \{B = q - i\}; \\ & \{A = q - 1\}, \quad \{B = q - 1 - i\}; \\ & \dots \dots \dots \\ & \{A = -q + i\}, \quad \{B = -q\}; \\ & \{A = q - i\}, \quad \{B = q\}; \\ & \{A = q - 1 - i\}, \quad \{B = q - 1\}; \\ & \dots \dots \dots \\ & \{A = -q\}, \quad \{B = -q + i\}. \end{aligned}$$

Поскольку перечисленные события составляют полную группу несовместимых событий, то по формуле полной вероятности получим

$$W \{ |\Delta| = i \} = 2 \sum_{j=-q}^{q-i} W \{ A = j \} W \{ B = j + i \}.$$

Порядки операндов A и B могут быть распределены по различным законам. Однако можно приближенно аппроксимировать эти распределения равномерным законом, для которого

$$W\{A=j\} = W\{B=j\} = \begin{cases} \frac{1}{2\alpha q + 1} & \text{при } |j| \leq \alpha q; \\ 0 & \text{при } |j| > \alpha q, \end{cases}$$

где α — параметр аппроксимации. Тогда

$$W\{|\Delta| = i\} = \frac{2(2\alpha q + 1 - i)}{(2\alpha q + 1)^2};$$

$$M = \frac{4\alpha q(\alpha q + 1)}{3(2\alpha q + 1)} \approx \frac{2\alpha q}{3}.$$

Так как $q \approx \log_k N = \frac{\ln N}{\ln k}$, то

$$M = \frac{2\alpha \ln N}{3 \ln k} = \frac{C}{\ln k},$$

где $C = \frac{2\alpha \ln N}{3}$ — константа, зависящая от диапазона представления чисел. Следовательно, среднее число сдвигов при выравнивании порядков обратно пропорционально логарифму основания системы счисления (константа C обычно равна или близка к 3).

Для сложения мантисс требуется время t_m , которое примерно равно времени получения разности порядков, т. е.

$$t_m \approx t_{s1}.$$

Оценим теперь среднее время $t_{н1}$, необходимое для первой нормализации. Для этого необходимо знать среднее число H сдвигов при нормализации. Для оценки H предположим, что операции сложения и вычитания, а также знаки чисел X и Y встречаются одинаково часто. Не ограничивая общности, можно считать, что $A \leq B$ и, следовательно, после выравнивания порядков будут справедливы неравенства

$$0 < |M(X)| < 1, \quad \frac{1}{k} \leq |M(Y)| < 1. \quad (5.5)$$

Среднее число сдвигов H равно

$$H = \sum_{i=-1}^q |i| W\{Q=i\} \approx W\{Q=-1\} + \sum_{i=1}^{\infty} W\{Q \geq i\},$$

где $W\{Q=i\}$ — вероятность нарушения нормализации на i разрядов вправо. Легко видеть, что

$$\begin{aligned} W\{Q=-1\} &= W\{|M(X) + M(Y)| \geq 1\}; \quad W\{Q \geq i\} = \\ &= W\{|M(X) + M(Y)| < k^{-i}\}. \end{aligned}$$

Если мантиссы $M(X)$ и $M(Y)$ статистически независимы и равномерно распределены в интервалах, определяемых условиями (5.5), то эти вероятности можно найти с помощью диаграммы на рис. 5.8,

задающей области значений суммы $M(X) + M(Y)$. Вероятность $W\{Q = -1\}$ равна отношению площади областей, где $|M(X) + M(Y)| > 1$, к площади областей всех возможных значений суммы мантисс (на рис. 5.8 эти области соответственно заштрихованы и выделены жирной линией)

$$W\{Q = -1\} = \frac{k+1}{4k}.$$

Аналогично, вероятность $W\{Q \geq i\}$ равна отношению площади областей, где $|M(X) + M(Y)| < k^{-i}$ к площади областей всех возможных значений суммы мантисс

$$\begin{aligned} W\{Q \geq i\} &= \frac{k(4k^i - 4k^{i-1} - 1)}{(k-1)4k^{2i}} = \\ &= \frac{1}{k^i} - \frac{1}{4k^{2i-1}(k-1)}. \end{aligned}$$

Отсюда, применяя формулу суммы бесконечной убывающей прогрессии, получим

$$\begin{aligned} H &= \frac{k+1}{4k} + \frac{1}{k-1} - \frac{k}{4(k-1)(k^2-1)} = \\ &= \frac{(k^2-1)^2 + 4(k^2-1) - k^2}{4k(k-1)(k^2-1)} \approx \frac{k+3}{4(k-1)}. \end{aligned}$$

Для оценки t_0 и t_{n2} предположим, что в каждом разряде числа с равными вероятностями могут быть как 0, так и 1. Следовательно, округление приводит к образованию переноса

в младший разряд с вероятностью $\frac{1}{2}$. Поэтому $t_0 = \frac{1}{2} t_m$. Нарушение нормализации после округления возможно только в случае, когда во всех разрядах результата будет 1. Вероятность этого события равна $\frac{1}{2^{n+1}} \approx 0$. Следовательно, $t_{n2} = 0$. Таким образом

$$\begin{aligned} t_+ &= t_{s1} + Mt_c + t_{s1} + Ht_c + \frac{1}{2} t_{s1} = 2,5t_{s1} + \\ &+ (M+H)t_c = 2,5t_{s1} + \left(\frac{3}{\ln k} + \frac{k+3}{4(k-1)} \right) t_c. \end{aligned}$$

Если $k = 2$, а время суммирования примерно равно времени сдвига, то

$$t_+ \approx (4 \div 5) t_{s1}.$$

и, следовательно, время сложения чисел в форме с плавающей запятой является в 4—5 раз более длительной операцией, чем сложение чисел с фиксированной запятой.

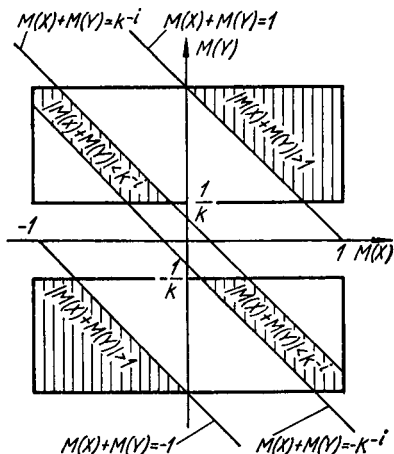


Рис. 5.8

5.4. ОСНОВНЫЕ АЛГОРИТМЫ И СТРУКТУРЫ ОПЕРАЦИОННЫХ БЛОКОВ ДЛЯ УМНОЖЕНИЯ В ПРЯМЫХ КОДАХ

При умножении чисел в прямых кодах знаковые разряды и разряды мантисс обрабатываются отдельно. Для определения знака результата производится суммирование цифр, записанных в знаковых разрядах операндов. Умножение мантисс можно производить с младших разрядов множителя либо со старших. При этом можно сдвигать сумму частичных произведений либо множимое. Комбинируя указанные возможности, получают четыре основных алгоритма умножения. Рассмотрим каждый из них в отдельности.

Пусть $X = \sum_{i=1}^n x_i 2^{-i}$, тогда

$$Z = YX = Y(x_1 2^{-1} + x_2 2^{-2} + \dots + x_n 2^{-n}) = 2^{-1}(Yx_1 + 2^{-1}(Yx_2 + 2^{-1}(Yx_3 + \dots + 2^{-1}(Yx_{n-1} + 2^{-1}(Yx_n + 0)) \dots))).$$

Отсюда следует, что умножение можно производить по следующим рекуррентным формулам: $\Pi_0 = 0$; $\Pi_1 = (\Pi_0 + Yx_n) \cdot 2^{-1}$;

$$\Pi_2 = (\Pi_1 + Yx_{n-1}) 2^{-1}, \dots, \Pi_{i+1} = (\Pi_i + Yx_{n-i}) \cdot 2^{-1}, \dots$$

$$Z = \Pi_n = (\Pi_{n-1} + Yx_1) 2^{-1},$$

т. е. умножение сводится к n -кратному повторению цикла

$$\Pi_{i+1} = (\Pi_i + Yx_{n-i}) 2^{-1}$$

при начальных условиях $\Pi_0 = 0$, $i = 0$. Такой алгоритм принято называть умножением с младших разрядов множителя и сдвигом вправо суммы частичных произведений.

Для реализации умножения по этому алгоритму необходимо иметь регистр множимого RGY с цепями выдачи слова, реализующими умножение на x_{n-i} , сумматор SM для реализации сложения $\Pi_{i-1} \dot{+} Yx_{n-i}$, регистр RGZ с $2n$ разрядами и цепями сдвига вправо для реализации умножения на 2^{-1} и хранения Π_{i-1} , а также сдвигающий регистр RGX для хранения и просмотра множителя X . Составим микроалгоритм реализации операции умножения (MA) на указанных узлах

$$MAU = H(\Pi_0 CTC \downarrow {}^3C_n \uparrow {}^1B_1 RGY \ B_1 RGZ)(\bar{x}_n \uparrow {}^2\Pi_1 RGZ \downarrow {}^2) \\ (C_1 RGZ \ C_1 RGX \ P_1 CTC \uparrow {}^3) \downarrow {}^1K,$$

где x_n — логическое условие, соответствующее анализу содержимого младшего разряда RGX ; C_n — логическое условие «содержимое CTC равно n ».

Схема, реализующая такой MAU , показана на рис. 5.9. Так как сдвиги в RGX и RGZ' осуществляются в одну и ту же сторону, и RGX по мере сдвигов освобождается, а RGZ' заполняется, то их функции могут быть совмещены в одном регистре RGX (см. штриховую линию на рис. 5.9). Аппаратурные затраты на реализацию данного MAU будут равны

$$C_1 = \alpha_1 n + \beta_1 \log_2 n + \gamma_1,$$

а быстродействие операционного блока определяется выражением

$$t_{y1} \geq n (t_0 + t_+),$$

где t_0 и t_+ — соответственно длительность тактов сложения и сдвига.

Пример 1. Составить цифровую диаграмму работы рассмотренного блока при $X = \frac{11}{16}$; $Y = \frac{12}{16}$; $n = 4$ (табл. 72).

Таблица 72

RGX	x_n	Микро- операция	RGY	RGZ	RGZ'	CTC
1011	1	$H + Y$	1100	00000	0000	000
		Сдв		01100		
0101	1	$+Y$		00110	0000	001
		Сдв		10010		
0010	0	$+0$		01001	0000	010
		Сдв		01001		
0001	1	$+Y$		00100	1000	011
		Сдв		10000		
0000	0			01000	0100	100
0000	0	K		01000	0100	100

В операционном блоке на рис. 5.9 используются регистры со встроенными микрооперациями B_1 , Π_1 , C_1 . При построении множительного блока на основе параллельных регистров, реализующих только микрооперацию Π_1 , сдвиг множителя X и суммы частичных произведений следует выполнять при передаче слов с основных регистров RGX и RGZ на вспомогательные RGX' и RGZ' . Кроме того, на RGZ необходимо выполнять две микрооперации: $\Pi_{11}RGZ$ — прием слова с сумматора SM , $\Pi_{12}RGZ$ — прием слова со вспомогательного регистра RGZ' . Микроалгоритм умножения в этом случае записывается так:

$$MAU = H(\Pi_0 CTC \downarrow^4 C_n \uparrow^1 x_n \uparrow^2 \Pi_{12} RGZ \uparrow^3 \downarrow^2 \Pi_{11} RGZ \downarrow^3)(\Pi_1 RGZ' \Pi_1 RGX' P_1 CTC)(\Pi_1 RGX) \uparrow^4 \downarrow^1 K,$$

а структура операционного блока показана на рис. 5.10, а. Управление таким блоком должно быть двутактным. При этом в первом такте выполняются микрооперации $\Pi_{12}RGZ$, $\Pi_{11}RGZ$, Π_1RGX , а во втором такте — Π_1RGZ' , Π_1RGX' , P_1CTC . Время выполнения операции умножения таким блоком составит

$$t_y \geq n (t_s + 2t_n).$$

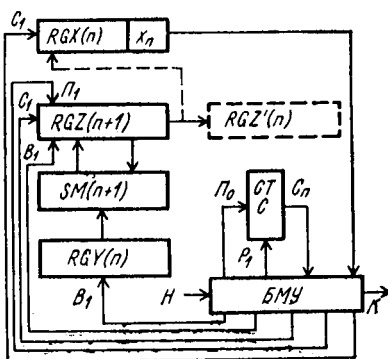


Рис. 5.9

При наличии регистров, выполняющих только микрооперацию Π_1 , для сдвига множителя X и суммы частичных произведений можно использовать один и тот же вспомогательный регистр RGX' , организовав работу всего множительного блока (рис. 5.10, б) по 4-тактному циклу. Перед началом первого такта сумма частичных произведений находится в RGX' . В первом такте в зависимости от цифры x_n выполняется микрооперация $\Pi_{11}RGZ$ (прием слова с сумматора) или же $\Pi_{12}RGZ$ (прием слова с регистра RGX'). Затем во втором такте — микрооперация $\Pi_{11}RGX'$ — прием слова на RGX' с регистра RGX , в третьем такте — микрооперация Π_1RGX , чем и обеспечивается сдвиг множителя на один разряд вправо. В четвертом такте выпол-

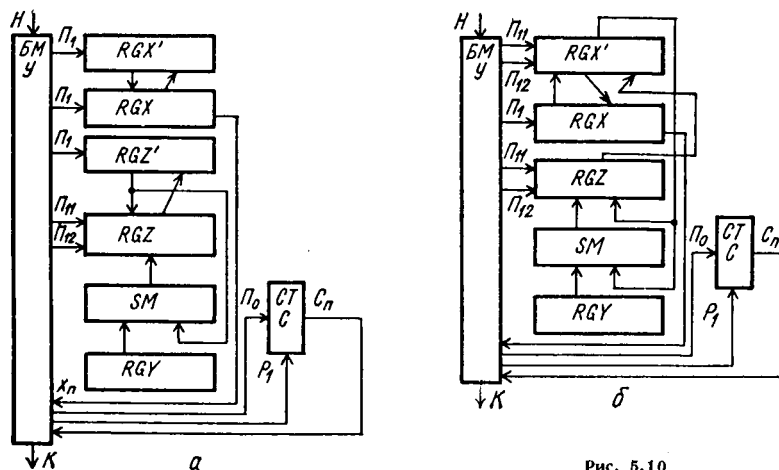


Рис. 5.10

няется микрооперация $\Pi_{12}RGX'$ — прием слова (со сдвигом вправо) с регистра RGZ . Микроалгоритм такого умножения можно представить в виде

$$MAU = H(\Pi_0 CTC \downarrow {}^4 C_n \uparrow {}^1 x_n \uparrow {}^2 \Pi_{12} RGZ \uparrow {}^3 \downarrow {}^2 \Pi_{11} RGZ \downarrow {}^3) \\ (\Pi_{11} RGX' P_1 CTC) (\Pi_1 RGX) (\Pi_{12} RGX' \uparrow {}^4) \downarrow {}^1 K,$$

а время умножения составит

$$t_y \geq n(t_s + 2t_n).$$

На рис. 5.11 показана схема четырехразрядного множительного блока, реализующего первый основной алгоритм умножения, построенного на основе регистров 155ИР1. Если на управляющий вход $V2$ такого регистра подать единичный сигнал, то регистр будет выполнять микрооперацию Π_1 синхронно с тактирующими сигналами $C2$. Если же $V2 = 0$, то регистр выполняет микрооперацию C_1 синхронно с сигналами $C1$ (младшим разрядом при этом будет разряд с весом 8). Вход $V1$ служит для записи информации в старший разряд при сдвиге содержимого регистра вправо. Тактирующие сигналы $C1$ и $C2$ не должны пересекаться во времени (могут быть взаимно инверсными).

Управление умножением осуществляется в соответствии с микроалгоритмом

$$MAY = H(P_0 CTC) (\downarrow^3 C_n \uparrow^1 \bar{x}_n \uparrow^2 P_1 RGZ) (\downarrow^2 C_1 RGX \\ C_1 RGZ P_1 CTC \uparrow^3) K.$$

После окончания умножения младшие разряды произведения находятся в RGX , а старшие — в RGZ . В качестве сумматора может быть использована микросхема 155ИМ3.

С целью выяснения особенностей машинного умножения по второму алгоритму представим произведение Z в следующем виде

$$Z = 2^{-n} (Y 2^{n-1} x_1 + Y 2^{n-2} x_2 + \dots + \\ + Y 2^1 x_{n-1} + Y 2^0 x_n) = 2^{-n} (Y_{n-1} x_1 + \\ + Y_{n-2} x_2 + \dots + Y_1 x_{n-1} + Y_0 x_n) = \\ = 2^{-n} (Y_{n-1} x_1 + (Y_{n-2} x_2 + \dots + \\ + (Y_1 x_{n-1} + (Y_0 x_n + 0)) \dots)),$$

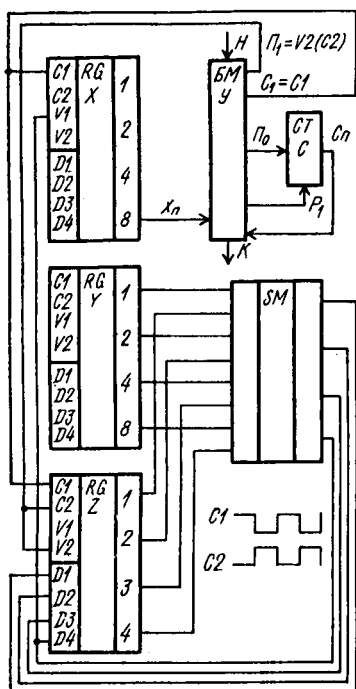


Рис. 5.11

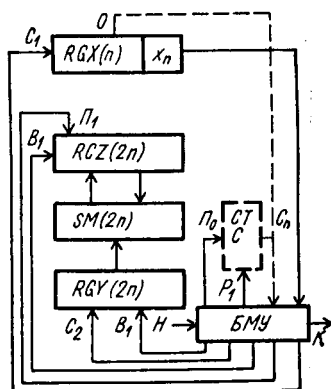


Рис. 5.12

где $Y_i = Y 2^i = Y_{i-1} 2$, а скобки указывают порядок действий. Следовательно, умножение в этом случае сводится к n -кратному выполнению цикла

$$P_{i+1} = P_i + Y_i x_{n-i}; \quad Y_i = Y_{i-1} 2$$

при начальных условиях $i = 0$, $Y_0 = Y$, $P_0 = 0$. Множитель 2^{-n} указывает на то, что после n -го цикла разряды произведения с весом 2^{-i} будут сдвинуты по отношению к разрядам операндов с таким же весом на n разрядов влево. Для приведения произведения к разрядной сетке операндов не требуются дополнительные затраты времени и оборудования — достаточно после окончания умножения переслать в ЗУ содержимое старших разрядов регистра результата (возможно с округлением). Такой алгоритм принято называть умножением с младших

разрядов множителя и сдвигом множимого влево. Составим микроалгоритм такого умножения

$$MAU = H(P_0 CTC \downarrow {}^3 C_n \uparrow {}^1 B_1 RGY \ B_1 RGZ) (\bar{x}_n \uparrow {}^2 P_1 RGZ \downarrow {}^2) \\ (C_2 RGY \ C_1 RGX \ P_1 CTC \uparrow {}^3) \downarrow {}^1 K.$$

Схема, реализующая данный МАУ, показана на рис. 5.12. Особенностью схемы является наличие $2n$ разрядов в регистрах RGY , RGZ и сумматоре SM . Регистр RGZ должен быть двухтактным, так как только в таком регистре можно осуществить выдачу слова и прием через SM этого же слова. Преимуществом схемы является неподвижность суммы частичных произведений, вследствие чего определять конец операции можно по нулевому содержимому регистра RGX (см. штриховую линию на рис. 5.12), т. е. отпадает необходимость в счетчике CTC , и, кроме того, можно совмещать микрооперации сдвига и суммирования. Как первое, так и второе повышает быстродействие и упрощает БМУ. Оценим эффект повышения быстродействия за счет того, что сигнал K будет вырабатываться по нулевому содержимому RGX . Пусть 0 и 1 в любом разряде встречаются с равными вероятностями. Тогда вероятность сокращения умножения на i тактов будет равна вероятности появления i нулей в старших разрядах RGX и единицы в $(i+1)$ -м разряде, т. е. величине $\frac{1}{2^{i+1}}$. Отсюда среднее число нулей получим как

$$\sum_{i=1}^n i \frac{1}{2^{i+1}} \approx \sum_{i=1}^{\infty} i \frac{1}{2^{i+1}} = 1.$$

Следовательно, в среднем операция умножения будет длиться $n - 1$ такт. Если в рассматриваемом множительном блоке суммирование и сдвиг совместить во времени и $t_+ \geq t_c$, что обычно имеет место, то

$$t_{y2} \geq (n - 1) t_+; \ C_2 = \alpha_2 n + \gamma_2.$$

Заметим, что $\alpha_2 > \alpha_1$ и $C_2 > C_1$, так как при умножении по второму алгоритму принципиально необходимы RGY , RGZ и SM двойной разрядности.

Умножение по третьему основному алгоритму сводится к вычислениям

$$Z = 2^{-n} (Y 2^{n-1} x_1 + Y 2^{n-2} x_2 + \dots + Y 2^1 x_{n-2} + Y 2^0 x_n) = \\ = 2^{-n} (x_n Y + 2(x_{n-1} Y + 2(x_{n-2} Y + \dots + 2(x_2 Y + 2(x_1 Y + 0)) \dots))).$$

Таким образом, умножение сводится к n -кратному выполнению цикла

$$P_{i+1} = x_{i+1} Y + 2P_i$$

при начальных условиях $i = 0$; $P_0 = 0$. Такой алгоритм получил название умножения со старших разрядов множителя и сдвигом суммы частичных произведений влево. Поскольку умножение в этом случае

должно заканчиваться суммированием, а не сдвигом, то микроалгоритм умножения следует записать так:

$$MAU = H(P_0 CTC \downarrow {}^3 B_1 RGY \ B_1 RGZ) (\bar{x}_1 \uparrow {}^1 P_1 RGZ \downarrow {}^1 P_1 CTC) \\ (C_n \uparrow {}^2 C_2 RGZ \ C_2 RGX \uparrow {}^3) \downarrow {}^2 K,$$

где логическое условие x_1 соответствует анализу старшей цифры множителя. Схема, реализующая данный МАУ, показана на рис. 5.13, а. Особенностью ее является то, что регистр RGZ' выполнен в виде счетчика, так как при суммировании возможно появление единицы переноса в младший разряд RGZ' . Сдвиги в RGX и RGZ' осуществляются в одну и ту же сторону. Поэтому функции RGZ' может выполнять

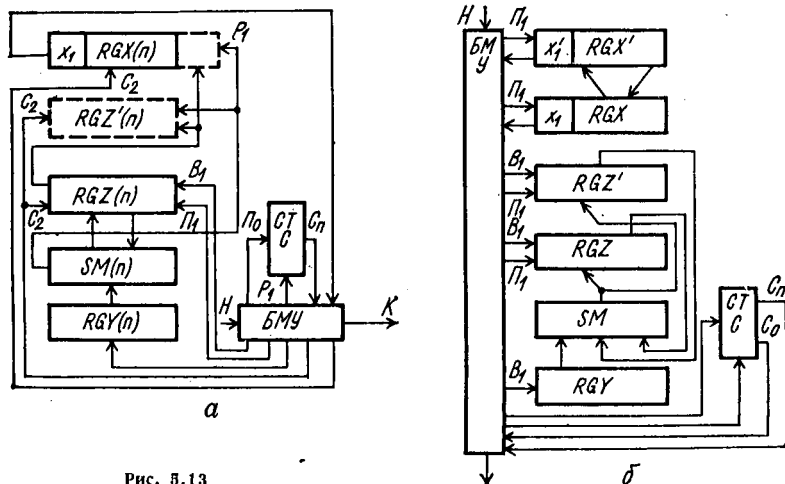


Рис. 5.13

RGX , при этом его длину надо увеличить на 1, чтобы перенос из старшего разряда суммы частичных произведений не попал в младший разряд множителя. Отсюда следует, что

$$t_{yz} \geq n(t_+ + t_c), \quad C_3 = \alpha_3 n + \beta_3 \log_2 n + \gamma_3.$$

При умножении по третьему алгоритму можно совмещать суммирование со сдвигом, используя регистры, выполняющие микрооперации P_1 и B_1 . Все циклы умножения в этом случае разделяют на нечетные и четные. В нечетных циклах производится передача содержимого RGX на RGX' (рис. 5.13, б) со сдвигом влево и суммирование содержимого RGY и RGZ' со сдвигом суммы влево. Управляет суммированием при этом старший разряд RGX . Указателем четного или нечетного цикла может служить содержимое c_0 младшего разряда CTC . В четных циклах ($c_0 = 0$) содержимое RGX' передают со сдвигом влево на RGX , управляет суммированием x'_1 — старший разряд RGX' , а сумму содержимого RGZ и RGY передают со сдвигом влево на RGZ' . Регистры RGZ и RGZ' , а также сумматор при этом должны иметь длину в $2n$

разрядов. Управление умножением реализуется в соответствии со следующим микроалгоритмом

$$\begin{aligned} MAU = & H (P_0 CTC \downarrow^4 \bar{c}_0 \uparrow^1 P_1 RGX \ B_1 RGZ \ \bar{x}_1 B_1 RGY \downarrow^2 \\ & P_1 RGZ' \ P_1 CTC \ C_n \uparrow^3 \uparrow^4) (\downarrow^1 P_1 RGX' \ B_1 RGZ' \bar{x}_1 \uparrow^5 \\ & B_1 RGY \downarrow^5 P_1 RGZ \ P_1 CTC \ C_n \uparrow^3 \uparrow^4) \downarrow^3 K. \end{aligned}$$

Для множительного блока на рис. 5.13, б

$$t_y \geq nt_+.$$

При четвертом основном алгоритме умножения произведение Z представляется в виде

$$\begin{aligned} Z = & (Y2^{-n}x_n + (Y2^{-n+1}x_{n-1} + \dots + (Y2^{-2}x_2 + \\ & + (Y2^{-1}x_1 + 0)) \dots)) = (Y_nx_n + (Y_{n-1}x_{n-1} + \dots + (Y_2x_2 + \\ & + (Y_1x_1 + 0)) \dots)), \end{aligned}$$

где $Y_i = Y \cdot 2^{-i} = Y_{i-1} \cdot 2^{-1}$

Следовательно, умножение можно представить как n -кратное повторение цикла

$$\Pi_i = \Pi_{i-1} + x_i Y_i, \ Y_i = Y_{i-1} 2^{-1}$$

при начальных условиях $i = 1$; $Y_0 = Y$; $\Pi_0 = 0$. Такой алгоритм называют умножением со старших разрядов множителя и сдвигом множимого вправо. Умножение сводится к реализации следующего микроалгоритма (предполагая, что Y_1 уже сформировано в RGY):

$$\begin{aligned} MAU = & H (\Pi_0 CTC \downarrow^3 C_n \uparrow^1 B_1 RGY \\ & B_1 RGZ) (\bar{x}_1 \uparrow^2 P_1 RGZ) \\ & (\downarrow^2 C_2 RGX \ C_1 RGY \ P_1 CTC \uparrow^3) \downarrow^1 K. \end{aligned}$$

Схема, реализующая этот MAU , показана на рис. 5.14. Так же как и во втором алгоритме, здесь конец умножения можно определить по нулевому содержимому RGX , а суммирование совмещать со сдвигом. Вследствие этого

$$t_y \geq (n-1)t_+, \ C_4 = \alpha_4 n + \gamma_4.$$

Рассмотренные выше схемы множительных блоков построены в предположении, что необходимо вычислять $2n$ цифр произведения и производить затем округление до n цифр. С учетом этого, аппаратные затраты на построение блоков умножения в некоторых случаях могут быть уменьшены. При использовании первого и третьего основных алгоритмов такого упрощения добиться нельзя, так как функции регистра RGZ' выполняются регистром RGX . В схемах, реализующих второй алгоритм, сократить аппаратные затраты можно за счет укорачивания RGZ и SM . При этом указанные узлы будут иметь длину $n + l$ разрядов, где l определяется из условия, чтобы погрешность от отбрасывания младших разрядов не превосходила половины веса

n -го разряда результата. Известно, что для этого l должно удовлетворять неравенству

$$l \geq 1 + \log_2(n - l - 1).$$

Отсюда нетрудно установить следующую систему соответствий между числом дополнительных разрядов и длиной чисел:

$$\begin{aligned} l = 2, \text{ если } n = 4, 5; l = 3, \text{ если } 6 \leq n \leq 8; \\ l = 4, \text{ если } 9 \leq n \leq 13; l = 5, \text{ если } 14 \leq n \leq 22; \\ l = 6, \text{ если } 23 \leq n \leq 39; l = 7, \text{ если } 40 \leq n \leq 72. \end{aligned}$$

Наибольшего упрощения можно добиться в схемах, реализующих четвертый алгоритм, так как в этом случае укороченным может быть также и регистр RGY .

Округление результата умножения до n разрядов можно осуществить путем прибавления единицы к старшему отбрасываемому разряду (т. е. к $n + 1$ -му разряду), для чего требуется время, равное времени суммирования двух чисел. После этого все разряды, правее n -го, отбрасывают. Погрешность округления при этом знакопеременна и по абсолютной величине не превышает половины веса младшего разряда.

Дополнительных затрат времени на округление можно избежать, если перед началом умножения записать единицу в тот разряд RGZ , который после умножения окажется старшим отбрасываемым разрядом. Для множительных блоков, реализующих, например, первый алгоритм, таким разрядом будет первый разряд RGZ , а для блоков, реализующих четвертый алгоритм — $(n + 1)$ -й разряд.

Сравнивая аппаратные затраты и быстродействие множительных блоков, можно сделать вывод, что реализация первого основного алгоритма требует минимальных затрат, а второго и четвертого алгоритмов — обеспечивает максимальное быстродействие. При этом помимо высокого быстродействия схемы с неподвижными суммами частичных произведений позволяют производить накопление суммы произведений вида $\sum_{i=1}^N X_i Y_i$, для чего достаточно результат очередной операции оставлять в RGZ .

В заключение отметим, что качество операционных блоков на уровне их структурных схем иногда удобно оценивать обобщенным технико-экономическим показателем $\epsilon = \frac{1}{C_i}$, называемым коэффициентом эффективности. Из числа рассмотренных выше блоков наибольшим коэффициентом эффективности характеризуется блок, реализующий четвертый основной алгоритм. По этим причинам первый и четвертый алгоритмы умножения получили наибольшее распространение.

5.5. МЕТОДЫ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ БЛОКОВ УМНОЖЕНИЯ. ЛОГИЧЕСКИЕ СПОСОБЫ УСКОРЕНИЯ УМНОЖЕНИЯ

Операция умножения встречается в программах так же часто, как и операция сложения. Однако, как показано выше, время выполнения умножения в n раз и более превышает время сложения.

Поэтому производительность АЛУ практически определяется скоростью выполнения операции умножения, а при проектировании АЛУ большое внимание уделяют ускорению умножения. Это ускорение, как правило, сопровождается увеличением аппаратных затрат. Так как блок умножения состоит из суммирующего блока (СБ) и БМУ, то в зависимости от того, какая из частей блока умножения усложняется, различают логические, аппаратные и комбинированные методы ускорения.

Логическими будем называть такие методы, в которых ускорение умножения достигается за счет усложнения схемы БМУ при неизменной структуре СБ.

При аппаратных методах ускорение достигается за счет введения дополнительного оборудования в СБ. При этом, в зависимости от того, какой величине (n или n^2) пропорциональны дополнительные затраты оборудования, различают аппаратные методы первого и второго порядков. При комбинированных методах используются как аппаратные, так и логические средства ускорения.

Рассмотрим логический метод ускорения умножения за счет исключения тактов суммирования, когда очередная цифра множителя равна 0.

Идею метода проиллюстрируем на примере первого основного алгоритма умножения. Микроалгоритм умножения при этом будет иметь вид

$$MAU = H(P_0 CTC \downarrow {}^3 C_n \uparrow {}^1 \bar{x}_n \uparrow {}^2 B_1 RGY \ B_1 RGZ) (P_1 RGZ) \\ (\downarrow {}^2 C_1 RGX \ C_1 RGZ \ P_1 CTC \uparrow {}^3 \downarrow {}^1) K. \quad (5.7)$$

Из сравнения выражений (5.6) и (5.7) видно, что для ускоренного умножения необходимы те же операционные узлы, что и для основного алгоритма. Отличие состоит в порядке выполнения микроопераций, а именно: если очередная цифра множителя равна 0, то последовательность микроопераций, соответствующую суммированию, пропускают и переходят сразу к сдвигу. Предполагая, что цифры 0 и 1 встречаются одинаково часто в любом разряде множителя, получим

$$t_y \geq \frac{1}{2} n t_+ + n t_c = n \left(\frac{t_+}{2} + t_c \right).$$

Применение этого метода в блоках, реализующих второй и четвертый основные алгоритмы, при $t_c = t_+$ не приводит к ускорению умножения. Однако, если совмещать два и более сдвигов с одним суммированием (т. е. $t_+ \geq 2t_c$), то это приводит к повышению быстродействия в 1,25 и более раз. Действительно, пусть $t_+ = 2t_c$. Обозначим вероятности появления двух очередных цифр 00, 01, 10 и 11 множителя в i -м такте через $W_j(i)$, ($j = 0, 1, 2, 3$). Для определенности предположим, что умножение производится с младших разрядов. При появлении цифр 00 в i -м такте производится сдвиг на один разряд, при появлении цифр 01 производится суммирование и два сдвига во время суммирования, при 10 — сдвиг, при 11 — суммирование и сдвиг. Составим зависимости, связывающие вероятности $W_j(i)$ для i -го такта с вероятностями $W_j(i+1)$ для $(i+1)$ -го такта. В $(i+1)$ -м такте комбинация цифр 00 может появиться, если: а) в i -м такте была ком-

бинация цифр 00 и после сдвига вновь появившейся цифрой оказался 0; б) в i -м такте была комбинация цифр 01 и после двух сдвигов, которые производятся в данном случае, появились цифры 00. Вероятность события а) равна $\frac{1}{2} W_0(i)$, а события б) $\frac{1}{2} \cdot \frac{1}{2} \cdot W_1(i)$. Так как события а) и б) несовместимы, то их сумма равна вероятности появления цифр 00 в $(i+1)$ -м такте, т. е.

$$W_0(i+1) = \frac{1}{2} W_0(i) + \frac{1}{4} W_1(i).$$

Аналогично для остальных вероятностей получаем

$$W_1(i+1) = \frac{1}{4} W_1(i) + \frac{1}{2} W_2(i) + \frac{1}{2} W_3(i);$$

$$W_2(i+1) = \frac{1}{2} W_0(i) + \frac{1}{4} W_1(i);$$

$$W_3(i+1) = \frac{1}{4} W_1(i) + \frac{1}{2} W_2(i) + \frac{1}{2} W_3(i).$$

К этим уравнениям следует добавить начальные условия

$$W_0(1) = W_1(1) = W_2(1) = W_3(1) = \frac{1}{4}.$$

Кроме того, $W_0 + W_1 + W_2 + W_3 = 1$ для любого такта.

Вероятности $W_j(i)$ изменяются от такта к такту только при малых i . При больших i значения $W_j(i)$ быстро стремятся к пределу и в дальнейшем практически не изменяются. Это свойство позволяет свести полученные рекуррентные уравнения к обычным алгебраическим уравнениям:

$$\left. \begin{aligned} W_0 &= \frac{1}{2} W_0 + \frac{1}{4} W_1; & W_1 &= \frac{1}{4} W_1 + \frac{1}{2} W_2 + \frac{1}{3} W_3; \\ W_2 &= \frac{1}{2} W_0 + \frac{1}{4} W_1; & W_3 &= \frac{1}{4} W_1 + \frac{1}{2} W_2 + \frac{1}{2} W_3, \end{aligned} \right\} \quad (5.8)$$

так как

$$\lim_{i \rightarrow \infty} W_j(i+1) = \lim_{i \rightarrow \infty} W_j(i) = W_j.$$

Из (5.8) получаем $W_0 = W_2 = \frac{1}{6}$; $W_1 = W_3 = \frac{1}{3}$. Следовательно, средняя длительность одного такта равна

$$(W_0 + W_2) t_c + (W_1 + W_3) t_+ = \frac{1}{3} t_c + \frac{2}{3} t_+,$$

а среднее число сдвигов за один такт составит $1W_0 + 2W_1 + 1W_2 + 1W_3 = \frac{4}{3}$. Поэтому время умножения будет определяться как

$$t_y \geq \frac{(n-1)^3}{4} \left(\frac{1}{3} t_c + \frac{2}{3} t_+ \right) = (n-1) \left(\frac{t_c}{4} + \frac{t_+}{2} \right).$$

Другой более эффективный логический метод ускорения умножения основан на использовании операций сложения и вычитания. В этом случае множитель представляют посредством цифр 1, 0, $\bar{1}$.

Представление двоичных чисел с помощью трех цифр является неоднозначным. Поэтому из всех возможных представлений числа выбирают то, которое содержит наименьшее количество цифр 1 и $\bar{1}$. Для этого группу из единиц 011...10 преобразуют в группу вида 100... $\bar{1}$ 0. В правомерности таких преобразований легко убедиться на следующем примере:

$$\begin{aligned} 30_{(10)} &= 11110_{(2)} = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = \\ &= 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + \bar{1} \cdot 2^1 + 0 \cdot 2^0 = 32_{(10)} - \\ &\quad - 2_{(10)} = 1000 \bar{1}0_{(2)}. \end{aligned}$$

При выполнении умножения множимое прибавляют к сумме частичных произведений, если очередная цифра множителя равна 1,

и вычитают, если цифра множителя $\bar{1}$. Преобразование множителя производится одновременно с выполнением умножения. Алгоритм преобразования зависит от того, со старших или младших разрядов выполняется умножение. Так как подвергаются преобразованию только группы из двух и более единиц, то одновременно необходимо анализировать не менее двух разрядов множителя. Однако при умножении со старших разрядов в преобразовании множителя участвуют

Таблица 73

x_{n-1}	x_n	φ_i	S	φ_{i+1}	E	j
0	0	0	+0	0	2	0
0	1	0	+Y	0	2	1
1	0	0	+0	0	1	2
1	1	0	-Y	1	2	3
0	0	1	+Y	0	2	4
0	1	1	+0	1	1	5
1	0	1	-Y	1	2	6
1	1	1	+0	1	2	7

иногда три разряда (например, группа цифр 011 преобразуется в группу 101). Поэтому при умножении со старших разрядов необходимо анализировать одновременно три разряда множителя. Далее для определенности будем считать, что умножение выполняется с младших разрядов множителя со сдвигом суммы частичных произведений вправо. Тогда алгоритм умножения может быть представлен табл. 73 (без учета столбцов E и j , необходимых в дальнейшем). Если очередные цифры множителя x_{n-1} и x_n равны 00, 01 или 10, то умножение происходит обычным способом, т. е. выполняются операции $S = +0$, либо $S = +Y$. Если же очередные цифры множителя будут 11, то вместо сложения производится вычитание и вырабатывается признак $\varphi_{i+1} = 1$, свидетельствующий о том, что первая единица данной группы преобразована в $\bar{1}$, все остальные единицы группы преобразованы в 0, а первый 0, следующий за данной группой единиц, должен быть преобразован в 1. При признаке $\varphi_i = 1$ умножение производится так, как будто эта единица просуммировалась с цифрой x_n (табл. 73). Признак $\varphi_{i+1} = 0$ вырабатывается в том случае, если две очередные цифры равны 0. Составим микроалгоритм умножения по такому методу:

$$\begin{aligned} MAU &= H(P_0 CTC) (\downarrow {}^5 C_{n+1} \uparrow {}^1 S_0 \uparrow {}^2 B_1 RGZ \ S_1 \uparrow {}^3 B_2 RGY \\ P_1 SMn \uparrow {}^4 \downarrow {}^3 B_1 RGY) (P_1 RGZ) (\downarrow {}^2 C_1 RGX' \ C_1 RGZ \ P_1 CTC \uparrow {}^5 \downarrow {}^1) K, \end{aligned}$$

где C_{n+1} — логическое условие «содержимое СТС равно $n+1$ »; S_0 и S_1 — логические условия, определяющие очередную операцию на суммирующем блоке: $S_0 = 1$ соответствует операции $S = +0$; $S_1 = 1$ соответствует операции $S = +Y$; $S_0 = 0$ и $S_1 = 0$ соответствует операции $S = -Y$.

Проверка условий S_0 и S_1 осуществляется комбинационной схемой, реализующей функции

$$S_0 = \bar{x}_n \bar{\varphi}_i \vee x_n \varphi_i, \quad S_1 = x_n \bar{x}_{n-1} \bar{\varphi}_i \vee \bar{x}_{n-1} \bar{x}_n \varphi_i = \bar{x}_{n-1} \bar{S}_0.$$

Так как в i -м такте вырабатывается признак φ_{i+1} для $(i+1)$ -го такта, то для его запоминания необходим триггер, функции возбуждения R_φ и S_φ которого будут иметь вид

$$R_\varphi = S_1, \quad S_\varphi = x_{n-1} x_n \bar{\varphi}_i \vee x_{n-1} \bar{x}_n \varphi_i = x_{n-1} \bar{S}_0.$$

Таблица 74

Цифры множителя			
до преобразования		после преобразования	
x_{i-1}	x_i	x_{i-1}	x_i
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	1

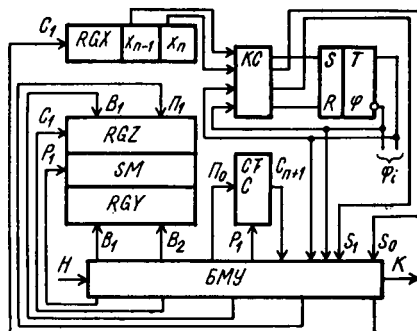


Рис. 5.15

Схема, реализующая рассмотренный МАУ, показана на рис. 5.15, где КС — комбинационная схема, реализующая функции S_0 , S_1 , R_φ , S_φ . Конец умножения определяется по $(n+1)$ -му состоянию счетчика СТС. Это связано с тем, что преобразованный множитель может быть на один разряд длиннее непреобразованного множителя.

Для оценки быстродействия множительного блока необходимо определить среднее число единиц в преобразованном множителе. В табл. 74 представлены значения $(i-1)$ -го и i -го разрядов множителя до преобразования и после него. Для определенности будем считать, что умножение начинается с n -го (младшего) разряда множителя. Из таблицы видно, что $(i-1)$ -я цифра преобразованного множителя равна 1 только в одном случае и при равенстве 0 i -й цифры. Вероятность же последнего равенства составляет $1 - W_i$, где W_i — вероятность появления 1 в i -м разряде. Поэтому

$$W_{i-1} = \frac{1}{2} (1 - W_i).$$

Если считать, что при $i = n$ $W_n = \frac{1}{2}$, то

$$W_{n-1} = \frac{1}{2} \left(1 - \frac{1}{2}\right) = \frac{1}{4}; \quad W_{n-2} = \frac{1}{2} \left(1 - \frac{1}{4}\right) = \frac{3}{8};$$

$$W_{n-3} = \frac{1}{2} \left(1 - \frac{3}{8}\right) = \frac{5}{16}; \quad W_{n-4} = \frac{1}{2} \left(1 - \frac{5}{16}\right) = \frac{11}{32}, \dots$$

Легко заметить, что по мере удаления в сторону старших разрядов $W_{i-1} \approx W_i$. Поэтому предельное значение W_i можно определить из уравнения

$$W_i = \frac{1}{2} (1 - W_i),$$

т. е. $W_i \approx \frac{1}{3}$. Таким образом, вероятность появления единицы в i -м разряде составляет $\frac{1}{3}$, а среднее число единиц в преобразованном множителе равно $\frac{n}{3}$. Отсюда получаем, что

$$t_y \geq n \left(t_c + \frac{1}{3} t_+ \right).$$

Коэффициент эффективности рассматриваемого множительного блока даже при $t_+ = t_c$ выше, чем у предыдущего множительного блока.

Высокая эффективность рассмотренных логических методов привела к тому, что при разработке быстродействующих АЛУ они используются в первую очередь. Вследствие этого даже при повышении быстродействия аппаратными методами стремятся применять их в сочетании с наиболее сильными логическими методами, а именно: с пропуском тактов суммирования с нулем и преобразованием множителя.

5.6. АППАРАТНЫЕ МЕТОДЫ УСКОРЕНИЯ УМНОЖЕНИЯ

Примером аппаратного метода ускорения умножения первого порядка является умножение с использованием дополнительных цепей сдвига в регистрах на один, два и более разрядов.

Рассмотрим наиболее простой случай, когда регистры RGX и RGY снабжены цепями сдвига на один и два разряда, а умножение производится по первому основному алгоритму. Алгоритм ускоренного умножения поясняется табл. 73, где E — количество сдвигов, производимых в данном такте умножения. Составим микроалгоритм умножения

$$\begin{aligned} MAU = & H(\Pi_0 CTC) (\downarrow {}^6 C_{n+1} \uparrow {}^1 S_0 \uparrow {}^2 B_1 RGZ \ S_1 \uparrow {}^3 B_2 RGY \ P_1 SMn \uparrow {}^4 \\ & \downarrow {}^3 B_1 RGY) (\downarrow {}^4 P_1 RGZ) (\downarrow {}^2 E_1 \uparrow {}^5 C_{12} RGZ \ C_{12} RGX \ P_2 CTC \uparrow {}^6 \\ & \downarrow {}^5 C_{11} RGZ \ C_{11} RGX \ P_1 CTC \uparrow {}^6 \downarrow {}^1) K. \end{aligned}$$

Здесь E_1 — логическое условие, соответствующее сдвигу на один разряд в данном такте ($E_1 = 1$), а S_0 и S_1 имеют тот же смысл, что и ранее. Из табл. 73 следует

$$E_1 = \bar{x}_{n-1} x_n \varphi_i \vee x_{n-1} \bar{x}_n \bar{\varphi}_i = \bar{E}_2,$$

где E_2 — логическое условие, определяющее необходимость сдвига на 2 разряда.

Схема множительного блока, реализующего рассмотренный микроалгоритм, будет отличаться от приведенной на рис. 260 тем, что КС должна реализовать проверку условия E_1 , в зависимости от значения которого БМУ формирует сигналы C_{11} и P_1 или C_{12} и P_2 . Сигналы правого сдвига поступают, как и в предыдущей схеме, на регистры

RGX и RGZ , а сигналы P_1 и P_2 — на счетчик сдвигов. Аппаратурные затраты на такой множительный блок несколько больше затрат на предыдущий блок за счет усложнения RGX , RGZ , KC , CTC , $БМУ$. Для оценки быстродействия необходимо знать вероятности появления отдельных строк в табл. 73. Пусть $W_j(i)$ будет вероятностью появления j -й строки таблицы в i -м такте умножения ($j = 0, 1, \dots, 7$). Тогда например, вероятность появления в n -м и $(n - 1)$ -м разрядах множителя двух нулей будет равна $\frac{1}{4}$, если только сдвиг выполнялся на два разряда и признак ϕ на предыдущем такте был равен нулю, т. е.

$$W_0(i+1) = \frac{1}{4}(W_0(i) + W_1(i) + W_4(i)).$$

Аналогично можно установить, что вероятности W_j для i -го и $(i + 1)$ -го тактов умножения будут связаны между собой следующей системой рекуррентных соотношений:

$$W_1(i+1) = \frac{1}{4}(W_0(i) + W_1(i) + W_4(i)) + \frac{1}{2}W_2(i);$$

$$W_2(i+1) = \frac{1}{4}(W_0(i) + W_1(i) + W_4(i));$$

$$W_3(i+1) = \frac{1}{4}(W_0(i) + W_1(i) + W_4(i)) + \frac{1}{2}W_2(i);$$

$$W_4(i+1) = \frac{1}{4}(W_3(i) + W_6(i) + W_7(i)) + \frac{1}{2}W_5(i);$$

$$W_5(i+1) = \frac{1}{4}(W_3(i) + W_6(i) + W_7(i));$$

$$W_6(i+1) = \frac{1}{4}(W_3(i) + W_6(i) + W_7(i)) + \frac{1}{2}W_5(i);$$

$$W_7(i+1) = \frac{1}{4}(W_3(i) + W_6(i) + W_7(i)).$$

при начальных условиях

$$W_0(0) = W_1(0) = W_2(0) = W_3(0) = \frac{1}{4};$$

$$W_4(0) = W_5(0) = W_6(0) = W_7(0) = 0.$$

Кроме того, для любого такта

$$\sum_{j=0}^7 W_j = 1.$$

Перейдя к пределу при $i \rightarrow \infty$ и решая полученную систему алгебраических уравнений, получим

$$W_0 = W_2 = W_5 = W_7 = \frac{1}{10}; \quad W_1 = W_3 = W_4 = W_6 = \frac{3}{20}.$$

Следовательно, за один такт в среднем производится умножение на

$$1(W_2 + W_5) + 2(1 - W_2 - W_5) = \frac{9}{5}$$

разрядов. При этом длительность одного такта будет равна

$$t_c + t_+(W_1 + W_3 + W_4 + W_6) = t_c + \frac{3}{5}t_+.$$

Отсюда имеем

$$t_y \geq \frac{n}{5} \left(t_c + \frac{3}{5} t_+ \right) = n \left(\frac{5}{9} t_c + \frac{1}{3} t_+ \right).$$

Как и следовало ожидать, число суммирований от введения цепей сдвига на два разряда не изменилось, в то время как количество сдвигов уменьшилось почти в два раза.

Таблица 75

k	x_{n-2}	x_{n-1}	x_n	$\varphi_i = 0$		$\varphi_i = 1$	
				S	φ_{i+1}	S	φ_{i+1}
1	0	0	0	+0	0	+Y	0
2	0	0	1	+Y	0	+2Y	0
3	0	1	0	+2Y	0	-Y	1
4	0	1	1	-Y	1	+0	1
5	1	0	0	+0	0	+Y	0
6	1	0	1	+Y	0	-2Y	1
7	1	1	0	-2Y	1	-Y	1
8	1	1	1	-Y	1	+0	1

Если использовать регистры, снабженные цепями сдвига на 1, 2 и 3 разряда, это позволяет добиться дальнейшего повышения быстродействия. Следует однако заметить, что замена регистров, сдвигающих на 1, 2, ..., m разрядов, регистрами с цепями сдвига на 1, 2, ..., m , $m+1$ разрядов сопровождается увеличением аппаратных затрат примерно на одну и ту же величину. В то

же время среднее число сдвигов, производимых за один такт умножения, увеличивается на величину, которая с ростом m уменьшается. Например, при $m = 1$ количество сдвигов равно n , при $m = 2$ — $0,56n$, а при $m = 3$ — $0,44n$. Отсюда следует, что данный метод вряд ли целесообразно использовать при $m \geq 3$.

Другой аппаратный метод ускорения умножения первого порядка основан на использовании сдвигателей, которые необходимы для передачи на сумматор удвоенного, учетверенного или, в общем случае, увеличенного в 2^i раз множимого. Рассмотрим на примере первого основного алгоритма умножения наиболее простой случай, когда сдвигатель может передавать на сумматор Y или $2Y$. Алгоритм ускоренного умножения поясняется табл. 75. Сдвиг множителя осуществляется всегда на два разряда. При этом анализировать надо три текущих разряда, так как в противном случае число суммирований будет больше, чем $\frac{1}{3}$ на разряд. Если, например, анализируются только

два разряда, то при комбинациях цифр 010 и 110 надо производить только одно действие: $+2Y$ или $-2Y$. Однако первое действие неоптимально во втором, а второе — в первом случае. Микроалгоритм умножения по данному способу будет иметь вид:

$$\begin{aligned} MAY &= H(P_0 CTC) (\downarrow^9 C_{n+1} \uparrow^1 B_1 RGZ \ S_0 \uparrow^2 S_1 \uparrow^3 S_2 \uparrow^4 B_2 RGY \\ P_1 SMn \ S_3 \uparrow^5 \downarrow^6 C_{11} CDB \uparrow^6 \downarrow^4 \downarrow^3 B_1 RGY \ S_1 \uparrow^7 \uparrow^8 \downarrow^5 \downarrow^7 C_{10} CDB) \\ &(\downarrow^6 P_1 RGZ) (\downarrow^2 P_2 CTC \ C_{12} RGX \ C_{12} RGZ \uparrow^9 \downarrow^1) K, \end{aligned}$$

$= 1, 2, 3, 7, 8$ или при $\varphi = 1$ и $k = 1, 2, 6, 7, 8$ сдвиг можно осуществлять сразу на три разряда, а в остальных случаях — на два (см. табл. 75).

Рассмотренные выше аппаратные методы направлены на уменьшение числа сдвигов, в то время, как число суммирований остается постоянным и равным $\frac{1}{3}$ на разряд. Уменьшить число суммирований позволяют методы, основанные на использовании дополнительных сумматоров. Например, по одному из таких методов множитель

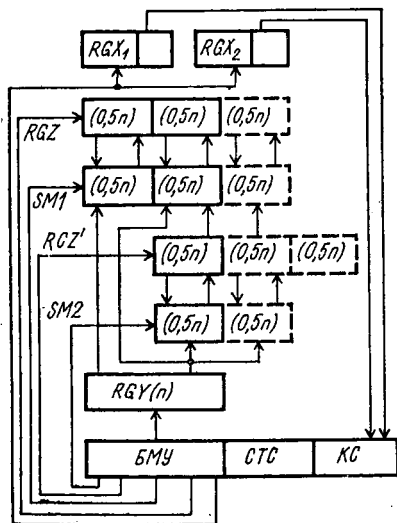


Рис. 5.17

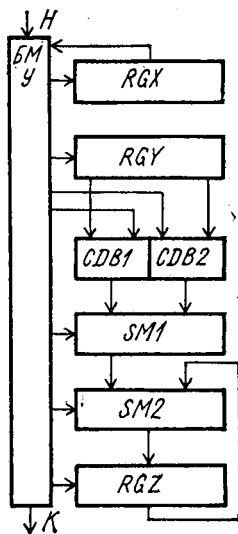


Рис. 5.18

$X = \sum_{i=1}^n x_i 2^{-i}$, условно разбивают на две части $X_1 = \sum_{i=1}^{0,5n} x_i 2^{-i}$ и $X_2 = \sum_{i=0,5n+1}^n x_i 2^{-i} = 2^{-0,5n} \sum_{j=0,5n+1}^n x_j 2^{-j+0,5n} = 2^{-0,5n} X_2'$, причем умножение на каждую из них производится независимо. Затем по формуле

$$Z = YX_1 + 2^{-0,5n} YX_2'$$

определяют результат. Структурная схема блока умножения, реализующего этот метод, показана на рис. 5.17. Для умножения на X_1 служит SM1 и RGZ, а для умножения на X_2 — SM2 и RGZ'. Так как здесь умножается n -разрядное число Y на $\frac{n}{2}$ -разрядное число X_1 или X_2 , то результат содержит $\frac{3}{2}n$ разрядов. Столько же разрядов должны иметь регистры RGZ, RGZ' и сумматор SM1. Схема множительного блока может быть существенно упрощена, если необходимо вычислять только n цифр произведения. В этом случае регистр RGZ и сумматор SM1 должны иметь только n разрядов, а RGZ' и SM2 — по $\frac{n}{2}$ разрядов.

В блоке умножения по схеме на рис.5.17 могут быть использованы любые рассмотренные ранее методы ускорения умножения. Однако наиболее удобными являются умножение на один или два разряда с преобразованием множителя. Дальнейшего повышения производительности блока умножения можно достичь, разбивая регистр RGX на три, четыре или более частей. При этом, однако, следует иметь в виду, что время умножения в таком блоке определяется наихудшей комбинацией цифр в частях регистра RGX . Вследствие этого нецелесообразно производить разбиение RGX больше, чем на две части.

Другая схема введения дополнительного сумматора, которая нашла применение в цифровой ЭВМ ЕС-1050, показана на рис. 5.18. Здесь Y передается на два сдвигателя $CDB1$ и $CDB2$, которые осуществляют сдвиг Y на нужное число позиций. Слова со сдвигателей суммируются на $SM1$, а на сумматоре $SM2$ происходит окончательное формирование частичного произведения. Использовать такую схему целесообразно только при умножении одновременно на четыре и более разрядов, так как при умножении, например, на три разряда число единиц в очередной тройке разрядов множителя редко равно двум. Поэтому сумматор $SM1$ не будет использоваться и, следовательно, добиться того, чтобы число суммирований было намного меньше, чем $\frac{1}{3}$ на разряд, не удастся. Именно такое число суммирований дает способ преобразования множителя.

Один из методов умножения на четыре разряда иллюстрируется табл. 77 на примере первого основного алгоритма, где буквами x_{n-3} , x_{n-2} , x_{n-1} , x_n обозначены цифры, зафиксированные в четырех младших разрядах регистра RGX , а $C1$ и $C2$ — слова на выходах сдвигателей $CDB1$ и $CDB2$. Сдвигатель $CDB1$ осуществляет сдвиг на два

Таблица 77

x_{n-3}	x_{n-2}	x_{n-1}	x_n	$\varphi_i = 0$		$\varphi_i = 1$	
				S		S	
				$C1+C2$	φ_{i+1}	$C1+C2$	φ_{i+1}
0	0	0	0	0+0	0	0+Y	0
0	0	0	1	0+Y	0	0+2Y	0
0	0	1	0	0+2Y	0	4Y-Y	0
0	0	1	1	4Y-Y	0	4Y+0	0
0	1	0	0	4Y+0	0	4Y+Y	0
0	1	0	1	4Y+Y	0	4Y+2Y	0
0	1	1	0	4Y+2Y	0	8Y-Y	0
0	1	1	1	8Y-Y	0	8Y+0	0
1	0	0	0	8Y+0	0	8Y+Y	0
1	0	0	1	8Y+Y	0	8Y+2Y	0
1	0	1	0	8Y+2Y	0	-4Y-Y	1
1	0	1	1	-4Y-Y	1	-4Y+0	1
1	1	0	0	-4Y+0	1	-4Y+Y	1
1	1	0	1	-4Y+Y	1	0-2Y	1
1	1	1	0	0-2Y	1	0-Y	1
1	1	1	1	0-Y	1	0+0	1

или три разряда, а сдвигатель *СДВ2* — на 0 или 1 разряд, т. е. *СДВ1* может выдавать слова 0, $+4Y$ или $+8Y$, а *СДВ2* — 0, $\pm Y$, $\pm 2Y$. В каждом такте умножения сдвиг осуществляется одновременно на четыре разряда. С помощью изложенной выше методики нетрудно определить время умножения

$$t_y \geq \frac{n}{4} \left(t_c + \frac{15}{16} t_+ \right).$$

Если при умножении на четыре разряда анализировать пять очередных цифр множителя, то это позволяет несколько уменьшить время выполнения операции.

$$t_y \geq \frac{n}{4} \left(t_c + \frac{11}{12} t_+ \right).$$

Некоторые аппаратные методы ускорения умножения направлены не на уменьшение числа суммирований, а на уменьшение времени самого суммирования. Это возможно осуществить благодаря многотактности операции умножения. Как известно, время t_2 суммирования равно

$$t_2 = t_1 + t_2 \cdot n,$$

где t_1 — время установления сигналов суммы, а t_2 — время прохождения сигналов переноса через один разряд сумматора.

Если же разорвать цепи переноса, то время суммирования будет определяться только величиной $t_1 \ll t_2 n$. При этом результат будет представлен поразрядной суммой Z_s и поразрядными переносами Z_p .

Схема блока умножения, реализующего эту возможность, приведена на рис. 5.19, где БМУ и *RGX* не показаны. Особенностью схемы является наличие двух регистров результата (регистра суммы *RGZ_s* и регистра переносов *RGZ_p*) и разрыв цепи переноса в сумматоре. В регистре *RGZ_s* накапливается поразрядная сумма частичных произведений, а в регистре *RGZ_p* — переносы. Выход каждого i -го разряда сумматора ($i = 1, 2, \dots, n$) соединен со входом i -го разряда *RGZ_p*, но выход этого же разряда *RGZ_p* подключен ко входу $(i - 1)$ -го разряда сумматора. После окончания умножения производят суммирование слов, записанных в *RGZ_s* и *RGZ_p*, на сумматоре с замкнутой цепью переноса, чем и формируется окончательный результат умножения в виде одного слова. Время умножения определяется n тактами «короткого» суммирования со сдвигами и одним полным суммированием, т. е.

$$t_y \approx t_+ + n t_c.$$

В некоторых быстродействующих цифровых ЭВМ рассмотренный метод применяется в сочетании с другими аппаратными и логическими способами ускорения умножения. Так, например, в машине «Стретч» умножение производится одновременно на 12 разрядов (рис. 5.20). Очередные 12 разрядов множителя разбиваются на 4 группы (Γ) по три разряда. Каждая тройка разрядов управляет своим сдвигателем *СДВ*. Слова со сдвигателей, являющиеся результатом умножения соответствующей группы разрядов X на Y , поступают на четыре сумматора с разорванными цепями переносов. На эти же сумматоры

в каждом такте умножения поступают слова с регистров RGZ_s и RGZ_p . В последнем такте слова с этих регистров поступают на SMZ с замкнутой цепью переноса, где окончательно формируется результат. Использование такого блока позволило выполнять умножение за время, примерно равное времени выполнения операции суммирования ($t_y = 1,4$ мкс при $n = 48$).

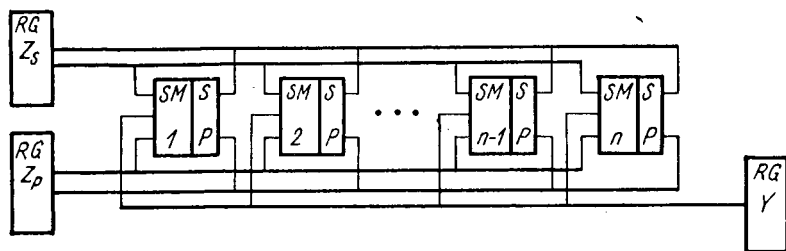


Рис. 5.19

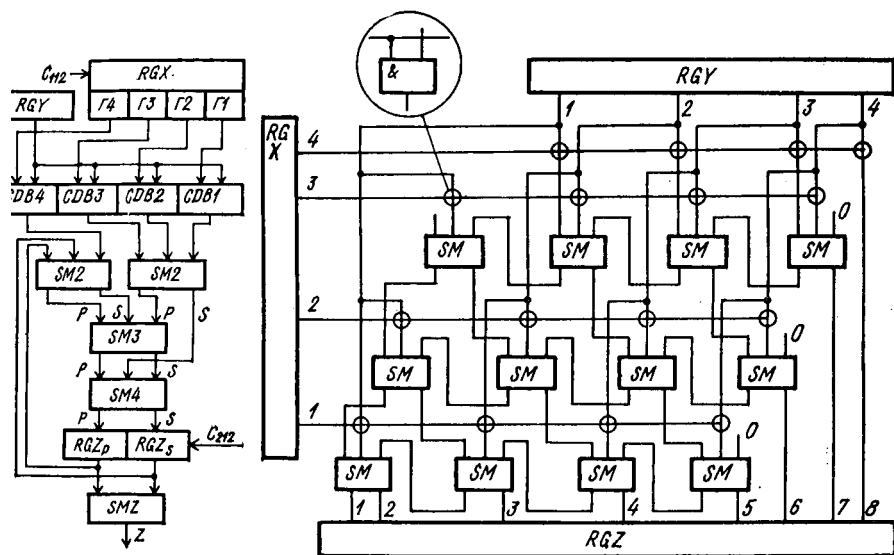


Рис. 5.20

Рис. 5.21

Рассмотрим теперь некоторые аппаратные методы ускорения второго порядка. Все они основаны на использовании блока умножения и комбинационной схемы. Функционирование таких схем может быть описано следующим образом. Пусть, например, $X = 0$, $x_1x_2x_3x_4$, $Y = 0$, $y_1y_2y_3y_4$.

Тогда $Z = XY$ можно представить в виде

$$\begin{aligned} & 2^{-5}x_4y_1 + 2^{-6}x_4y_2 + 2^{-7}x_4y_3 + 2^{-8}x_4y_4 \\ & 2^{-4}x_3y_1 + 2^{-5}x_3y_2 + 2^{-6}x_3y_3 + 2^{-7}x_3y_4 \\ & 2^{-3}x_2y_1 + 2^{-4}x_2y_2 + 2^{-5}x_2y_3 + 2^{-6}x_2y_4 \\ & 2^{-2}x_1y_1 + 2^{-3}x_1y_2 + 2^{-4}x_1y_3 + 2^{-5}x_1y_4 \end{aligned} \quad (5.9)$$

$$Z_1 + 2^{-1}Z_2 + 2^{-2}Z_3 + 2^{-3}Z_4 + 2^{-4}Z_5 + 2^{-5}Z_6 + 2^{-6}Z_7 + 2^{-7}Z_8 + 2^{-8}Z_9$$

Легко заметить, что в этой записи каждая строка, кроме последней, представляет собой частичное произведение множимого Y на одну из цифр множителя X . Отдельные разрядные произведения $x_i y_j$ могут быть получены с помощью элементов $И$. Суммирование частичных произведений сводится к суммированию разрядных произведений с одинаковыми весами (по столбцам) с учетом переносов из младших разрядов. Схема блока умножения, работающего по такому принципу, показана на рис. 5.21. Можно считать, что умножение здесь начинается с младших разрядов множителя. Для построения такого блока необходимо n^2 элементов $И$, формирующих разрядные произведения, и $n^2 - n$ одноразрядных сумматоров.

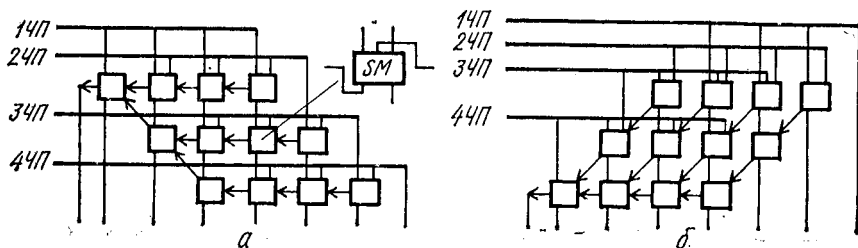


Рис. 5.22

Время умножения в таком блоке определяется в основном самым длинным путем прохождения сигналов через суммирующую схему — через $3n - 2$ одноразрядных сумматора:

$$t_y \geq 3nt_s,$$

где t_s — задержка сигналов одноразрядным сумматором (предполагаем, что $t_s \approx t_p$, см. гл. 3). Такой множительный блок обладает высоким быстродействием, однако требует значительных аппаратных затрат.

Другой вариант блока умножения с аппаратным методом ускорения 2-го порядка можно получить, если реализовать «перевернутое» представление (5.9):

$$\begin{aligned} &2^{-2}x_1y_1 + 2^{-3}x_1y_2 + 2^{-4}x_1y_3 + 2^{-5}x_1y_4 \\ &2^{-3}x_2y_1 + 2^{-4}x_2y_2 + 2^{-5}x_2y_3 + 2^{-6}x_2y_4 \\ &2^{-4}x_3y_1 + 2^{-5}x_3y_2 + 2^{-6}x_3y_3 + 2^{-7}x_3y_4 \\ &2^{-5}x_4y_1 + 2^{-6}x_4y_2 + 2^{-7}x_4y_3 + 2^{-8}x_4y_4 \\ \hline &2^{-1}Z_1 + 2^{-2}Z_2 + 2^{-3}Z_3 + 2^{-4}Z_4 + 2^{-5}Z_5 + 2^{-6}Z_6 + 2^{-7}Z_7 + 2^{-8}Z_8 \end{aligned}$$

В этом случае умножение начинается со старших разрядов множителя. На рис. 5.22, *a* показана организация суммирующей схемы (матрицы сумматоров) такого множительного блока. Обозначения 14П, 24П, ... соответствуют шинам частичных произведений, которые, как и раньше, формируют с помощью элементов $И$. Стелками на рисунке указаны направления движения сигналов переноса. Самая длинная цепь в суммирующей схеме состоит из $2n - 2$ одноразрядных сумматоров, следовательно $t_y \geq 2n t_s$.

Такого же быстроедействия множительного блока можно достичь и при умножении с младших разрядов, если в суммирующей схеме цепи переноса выполнить в соответствии с рис. 5.22, б. В двух рассмотренных ранее матрицах сумматоров переносы, возникающие в i -й строке при суммировании j -х разрядных произведений, учитывались в $(j - 1)$ -м разряде этой строки. Здесь же переносы из j -го разряда учитываются в $(j - 1)$ -м разряде $(i + 1)$ -й строки.

Аппаратурные затраты на рассмотренные блоки можно уменьшить, если необходимо получать не $2n$ разрядов произведения, а только n разрядов. В этом случае отбрасывается часть матрицы сумматоров, выходы которой соответствуют разрядам произведения с весами 2^{-n-l-1} , 2^{-n-l-2} , ..., 2^{-2n} , а l определяется так, как показано в п. 5.4.

5.7. ДРУГИЕ АЛГОРИТМЫ И БЛОКИ УМНОЖЕНИЯ В ЦИФРОВЫХ ЭВМ

В п. 5.2 было показано, что операция алгебраического сложения наиболее просто выполняется при представлении чисел в дополнительных кодах. При этом не только упрощается суммирующий блок, но и несколько повышается быстроедействие за счет отсутствия схемы определения знака фактически выполняемой операции. Вследствие этого в ряде цифровых ЭВМ отрицательные числа представляются и хранятся в памяти в дополнительных кодах. Однако при этом возникают трудности в выполнении операции умножения и деления, связанные с тем, что при представлении чисел в дополнительных кодах цифра в знаковом разряде имеет искусственный вес -1 . Из-за этого в общем случае умножение дополнительных кодов $[X]_д$ и $[Y]_д$ чисел X и Y не дает дополнительного кода результата:

$$[X]_д [Y]_д \neq [XY]_д.$$

Устранить этот недостаток можно либо путем придания цифре знакового разряда естественного веса 1, либо путем изменения алгоритма умножения таким образом, чтобы псевдопроизведение $[X]_д [Y]_д$ было равно произведению $[XY]_д$. Это можно осуществить, например, путем коррекции псевдопроизведения на последнем шаге умножения. Однако из-за того, что умножение на знаковый разряд (т. е. коррекция) осуществляется не так, как умножение на разряды мантиссы, способ умножения с коррекцией не нашел широкого распространения. Поэтому остановимся на первом способе.

Придать знаковому разряду естественный вес можно, например, путем преобразования цифр множителя с помощью формулы

$$x'_i = x_{i+1} - x_i,$$

где цифры x'_i принадлежат множеству $\{1, 0, \bar{1}\}$, а цифры x_i — множеству $\{0, 1\}$. Действительно

$$\begin{aligned} [X]_д &= \bar{1} X_0 + \frac{1}{2} x_1 + \dots + \frac{1}{2^n} x_n = 1(x_1 - X_0) + \frac{1}{2}(x_2 - x_1) + \\ &+ \frac{1}{4}(x_3 - x_2) + \dots + \frac{1}{2^{n-1}}(x_n - x_{n-1}) + \frac{1}{2^n}(0 - x_n) = \\ &= 1x'_0 + \frac{1}{2} x'_1 + \frac{1}{4} x'_2 + \dots + \frac{1}{2^{n-1}} x'_{n-1} + \frac{1}{2^n} x'_n. \end{aligned}$$

При $x'_i = 0$ такт сложения пропускается, при $x'_i = 1$ производится прибавление множимого, а при $x'_i = \bar{1}$ — вычитание. Так как умножение производится и на знаковый разряд, то число тактов умножения равно $n + 1$. Для того чтобы произведение заняло нужную позицию, сдвиг в последнем такте не производится. В преобразовании i -й цифры множителя участвуют i -я и $(i + 1)$ -я цифры. Поэтому в определении знака действия, производимого в данном такте, должны участвовать две очередные цифры множителя. Для этого при умножении с младших разрядов регистр RGX должен иметь на один разряд больше. Блок умножения, реализующий данный алгоритм, отличается от блока на рис. 5.15 способом формирования сигналов s_0, s_1, s_3

$$s_1 = \bar{x}_n x_{n+1}, \quad s_3 = x_n \bar{x}_{n+1}, \quad s_0 = \overline{s_1 \vee s_3},$$

а также тем, что знаковые разряды RGY соединены с аналогичными разрядами SM и обрабатываются так же, как и основные разряды.

Пример 1. Пусть

$$X = -\frac{5}{16} = -0,0101; \quad [X]_д = 1,1011 = 0,110\bar{1};$$

$$Y = -\frac{14}{16} = -0,1110; \quad [Y]_д = 1,0010;$$

$$[X \cdot Y]_д = X \cdot Y = \frac{70}{256} = 0,01000110.$$

Цифровая диаграмма состояний узлов блока умножения для данного примера при умножении по первому основному алгоритму приведена в табл. 78.

Таблица 78

RGX	RGY	Микро- операции	RGZ	RGZ'	$СТС$
1,10110	1,0010		000000	0000	000
(0,1101)	то же	$-Y$	00,1110	0000	001
111011	»	Сдв	00,0111	0000	
		$+0$	00,0111	0000	010
111101	»	Сдв	00,0011	1000	
		$+Y$	11,0101	1000	011
111110	»	Сдв	11,1010	1100	
		$-Y$	00,1000	1100	100
111111	»	Сдв	00,0100	0110	
		$+0$	00,0100	0100	101
					K

Другая возможность получения правильного произведения чисел в дополнительных кодах состоит в следующем. При отрицательном множителе считают, что его знаковый разряд является отрицательной цифрой, а сам множитель представлен в системе счисления с цифрами $\{1, 0, \bar{1}\}$. Далее по обычной методике выполняют умножение множимого (вместе со знаковыми разрядами) на все разряды множителя, причем на последнем такте (при умножении на знаковый разряд) вы-

полняют операцию $-Y$ вместо $+Y$ или $+0$. Если же множитель положительный, то его знаковый разряд не меняется, а умножение на знаковый разряд производится как обычно — выполняют операцию $+0$.

Пример 2. Составим цифровую диаграмму состояний узлов множительного блока, работающего по описанному алгоритму, при умножении чисел

$$X = -\frac{6}{16}, [X]_д = 1,1010, Y = -\frac{11}{16}, [Y]_д = 1,0101 \text{ (табл. 79).}$$

Таблица 79

RGX	RGY	Микро- операции	RGZ	RGZ'	$СТС$
$\bar{1},1010$	11,0101	$+0$	000000	0000	000
$\bar{1},101$	то же	Сдв	000000	0000	001
$\bar{1}10$	»	$+Y$	110101	0000	
$\bar{1}1$	»	Сдв	111010	1000	010
$\bar{1}$	»	$+0$	111010	1000	
	»	Сдв	111101	0100	011
	»	$+Y$	110010	0100	
	»	Сдв	111001	0010	100
		$-Y$	00,0100	0010	K

Легко видеть, что все рассмотренные ранее способы ускорения умножения применимы и при умножении чисел в дополнительных кодах. В частности, преобразовать множитель можно так, чтобы знающие цифры никогда не были соседними. Например, $[X]_д = 1,1011 = 0,\bar{1}10\bar{1} = 0,010\bar{1}$, т. е. число сложений можно сократить до $\frac{1}{2}$ на разряд.

В специализированных цифровых ЭВМ нашли применение блоки умножения последовательного типа. Особенности построения таких блоков рассмотрим на примере блока на динамических регистрах, в котором осуществляется умножение с младших разрядов и сдвиг суммы частичных произведений (применительно к последовательным блокам умножения этот способ можно было бы интерпретировать и как умножение с неподвижной суммой). Как уже отмечалось, микрооперация выдачи слова с последовательного регистра сводится к n -кратному сдвигу. Сдвиг двух слов друг относительно друга на один разряд также осуществляют путем n -кратного сдвига каждого из слов, при этом один из регистров удлиняется на один разряд. В остальном микроалгоритм работы последовательного блока умножения подобен микроалгоритму работы параллельного блока и поэтому здесь не приводится.

Схема последовательного блока умножения приведена на рис. 5.23, где также раскрыта структура и его БМУ. Сигнал H должен подаваться на блок умножения в момент времени, когда числа в RGX и RGY занимают исходную позицию — младшие разряды чисел находятся

в младших разрядах регистров. Взаимный сдвиг частичных произведений и множимого осуществляется путем удлинения регистра RGX на один разряд и использования счетчика тактов СТТ с периодом $T = n + 1$. Триггер $T1$ служит для запоминания очередной цифры множителя и управляет подачей множимого на сумматор. В конце каждого цикла умножения $T1$ устанавливается в «0» сигналом «СТТ = $n + 1$ », а запись очередной цифры множителя осуществляется в момент действия сигнала «СТТ = 0». Поочередная выборка цифр множителя осуществляется за счет того, что период СТТ на 1 больше,

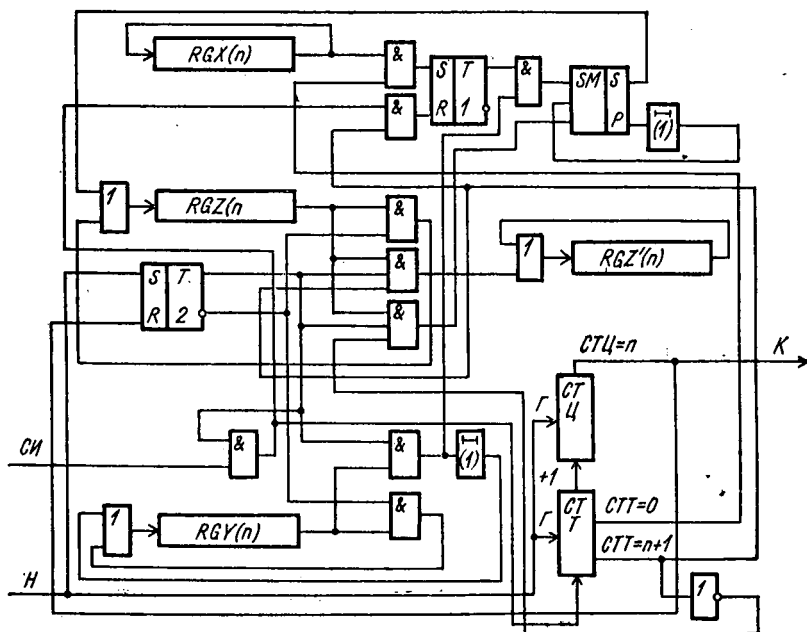


Рис. 5.23

чем число разрядов в регистре RGX . В каждом цикле по сигналу «СТТ = $n + 1$ » цифра младшего разряда частичных произведений поступает на регистр RGZ' . Поэтому после окончания умножения старшие разряды произведения будут находиться в регистре RGZ , а младшие — в регистре RGZ' , в качестве которого можно использовать регистр RGX .

Для получения сигналов «СТТ = 0» и «СТТ = $n + 1$ » первое и последнее состояния СТТ дешифрируются. Сигнал K вырабатывается счетчиком циклов СТЦ.

Для такого блока

$$G = \alpha n + \beta \log_2 n + \gamma \log_2 (n + 1) + \delta;$$

$$t_y \geq (n^2 + n) (t_c + t_+).$$

Сравнение этих оценок с соответствующими оценками для параллельных блоков умножения при конкретных значениях коэффициентов показывает, что аппаратные затраты уменьшились не более

чем в 10 раз, тогда как t_y увеличилось примерно в n раз. Следовательно коэффициент эффективности ϵ последовательных блоков умножения всегда меньше, чем параллельных.

Повышения быстродействия последовательных блоков умножения, как правило, достигают методами, аналогичными методам повышения быстродействия параллельных блоков (например, путем умножения на два и более разрядов). Вместе с тем, учет специфики представления и ввода чисел в операционный блок позволяет использовать для

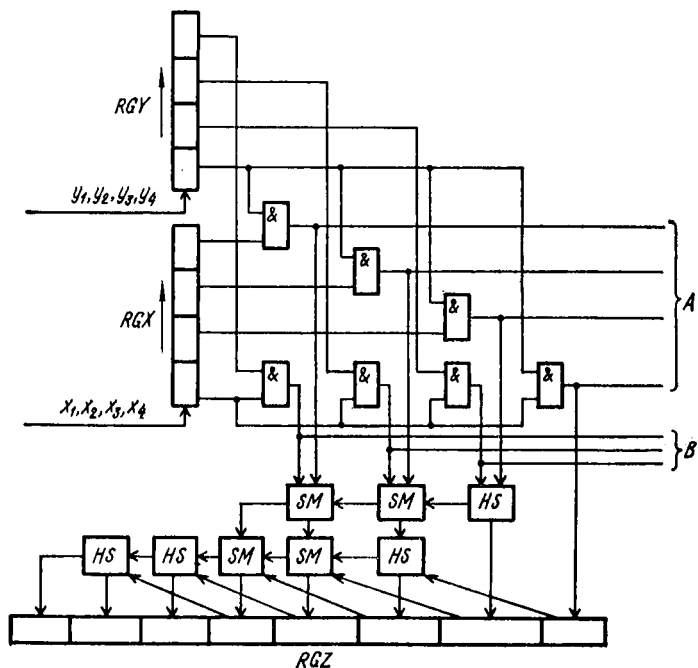


Рис. 5.24

ускорения умножения методы, не имеющие аналогов для параллельных кодов. Схема множительного блока, реализующего один из таких методов, приведена на рис. 5.24 ($n = 4$). В основу построения блока положена идея одновременного формирования и суммирования частичных произведений вида $X_{i-1}y_i$ и $Y_{i-1}x_i$, где X_{i-1} и Y_{i-1} — соответственно множитель и множимое, представленные $i - 1$ разрядами, x_i и y_i — очередные значения i -х разрядов сомножителей. Последовательные коды множителя и множимого поступают на регистры RGX и RGY со старших разрядов (их i -е разряды заносятся на регистры одновременно). С помощью элементов И в каждом такте умножения формируются слова $A = X_{i-1}y_i$ и $B = Y_{i-1}x_i - 2^{-i}x_i$. Например, в первом такте формируется произведение x_1y_1 ; во втором такте — произведения x_1y_2, x_2y_1, x_2y_2 ; в третьем такте — $x_1y_3, x_2y_3, x_3y_2, x_3y_3$ и т. д. Поскольку последнее разрядное произведение x_iy_i является общим для слов A и B, то его отнесем к одному из слов, на-

пример *A*. Этим и объясняется наличие члена $-2^{-i}x_i$ в приведенном выше выражении для *B*. Суммирующая часть блока предназначена для суммирования разрядных произведений с одинаковым весом. Сумма частичных произведений из регистра *RGZ* в каждом такте поступает на суммирующую схему со сдвигом на два разряда влево. Такой сдвиг можно реализовать с помощью соответствующей коммутации разрядов (рис. 5.24), либо выполнять в самом регистре *RGZ*.

Пример 3. Составим цифровую диаграмму состояний узлов блока по схеме на рис. 5.24 при умножении чисел $X = \frac{13}{16}$ и $Y = \frac{9}{16}$ (табл. 80).

Таблица 80

<i>RGX</i>	<i>RGY</i>	<i>A</i>	<i>B</i>	Суммируемые слова	<i>RGZ</i>	<i>СТС</i>
1	1	0001	000	0001 000	00000000	1
11	10	0000	001	00000000 0000 001	00000001	10
110	100	0000	000	00000100 0000 000	00000110	11
1101	1001	1101	100	00011000 1101 100 01100000	00011000 01110101	100 K

Как следует из рис. 5.24 и табл. 80, умножение последовательных кодов чисел *X* и *Y* здесь совмещено с вводом разрядов операндов в регистры операционного блока и длится *n* тактов, т. е.

$$t_y \geq n(t_s + t_+).$$

Аппаратурные затраты на рассматриваемый блок примерно равны затратам на параллельные блоки умножения.

Подобной схемой можно реализовать за *n* тактов и умножение последовательных кодов, поступающих младшими разрядами вперед.

Построение блоков умножения, работающих в десятичной системе числения, может быть выполнено на основе тех же принципов, что и двоичных блоков, т. е. имеется четыре основных алгоритма десятичного умножения, каждый из которых характеризуется своими преимуществами и недостатками, аналогичными преимуществам и недостаткам соответствующих алгоритмов двоичного умножения. Особенноностью десятичных блоков является то, что сдвиг в них осуществляется одновременно на *m* двоичных (один десятичный) разрядов и в каждом такте умножения в зависимости от значения очередной цифры множителя осуществляется несколько сложений множимого с суммой частичных произведений.

Для определенности будем считать, что умножение осуществляется по первому основному алгоритму. Составим микроалгоритм умножения

$$MAU = H(\Pi_0 CTC)(\downarrow^4 C_n \uparrow^1 \downarrow^3 A_0 \uparrow^2 B_1 RGY B_1 RGZ) \\ \Pi_1 RGZ R_1 RGX \uparrow^3)(\downarrow^2 C_1 RGX C_1 RGY P_1 CTC \uparrow^4) \downarrow^1 K$$

Здесь n равно числу десятичных разрядов, C_1 — микрооперация сдвига на один десятичный разряд, A_0 — логическое условие «содержимое младшего десятичного разряда множителя равно нулю», $R_1 RGX$ — микрооперация вычитания единицы из младшей цифры множителя.

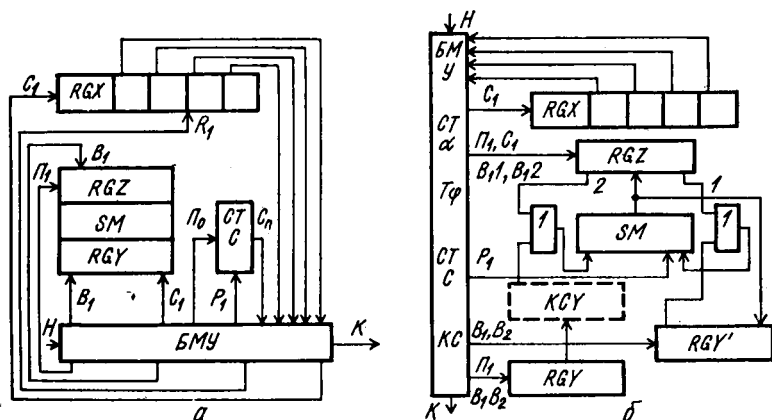


Рис. 5.25

Структура операционного блока, реализующего данный микроалгоритм и построенного на базе комбинационного сумматора, показана на рис. 5.25, а. Суммирующий блок, как обычно, состоит из двух регистров RGY и RGZ и сумматора SM . Регистры RGX и RGZ снабжены цепями сдвига на один десятичный разряд. Для реализации микрооперации $R_1 RGX$ последний десятичный разряд регистра RGX выполнен в виде счетчика, при нулевом содержимом в котором осуществляется переход к микрооперации C_1 . Если предположить, что цифры 0, 1...9 появляются во всех разрядах RGX с равной вероятностью, то на каждый сдвиг в среднем приходится 4,5 суммирований, а время умножения будет равно

$$t_y \geq n(t_c + 4,5t_+). \quad (5.10)$$

При этом аппаратные затраты определяются выражением

$$C = \alpha n + \beta \log_2 n + \gamma. \quad (5.11)$$

Представляет интерес сравнение эффективности десятичных и двоичных блоков умножения. Естественно, что при этом перемножаемые числа должны быть представлены с одинаковой точностью. Пусть δ будет точность, с которой заданы мантиссы чисел

$$\delta = \min(X_1 - X_2),$$

где X_1 и X_2 — ближайшие неравные друг другу числа. Отсюда следует, что

$$\delta = \left(\frac{1}{k}\right)^{n_k},$$

где k — основание системы счисления; n_k — количество разрядов в числе, соответствующих заданной точности δ их представления. Следовательно,

$$n_k = -[\log_k \delta] \approx -\frac{\lg \delta}{\lg k}.$$

При $k = 10$ и $k = 2$ получаем

$$n_{10} = -\lg \delta, \quad n_2 = -\frac{\lg \delta}{\lg 2}.$$

Таким образом,

$$n_{10} = n_2 \lg 2 \approx 0,3n_2.$$

Если полученные величины подставить в (5.10) и (5.11), то можно сделать вывод, что десятичные блоки умножения примерно в 1,2 раза сложнее и примерно в 1,5 раза (при $t_+ = 2t_c$) менее быстродействующие. Эти выводы еще в большей степени справедливы для второго и четвертого основных алгоритмов умножения, так как в случае их использования выигрыш в быстродействии от совмещения тактов суммирования и сдвигов дает незначительный эффект из-за того, что

$$t_c \ll 4,5t_+.$$

Вследствие этого наиболее эффективными являются блоки, реализующие умножение с младших разрядов множителя и сдвиг суммы частичных произведений.

Самым сильным логическим методом ускорения умножения, как и для двоичной системы счисления, является метод преобразования множителя. Применительно к десятичной системе этот метод состоит в представлении цифр, больших 5, с помощью отрицательных цифр. При этом 6 представляется как $\bar{1}\bar{4}$, 7 как $\bar{1}\bar{3}$, 8 как $\bar{1}\bar{2}$, 9 как $\bar{1}\bar{1}$. Например, $0,593497 = 1,4\bar{1}350\bar{3}$. Если при умножении на первое число потребовалось бы $5 + 9 + 3 + 4 + 9 + 7 = 37$ сложений, то при умножении на второе число требуется всего $1 + 4 + 1 + 3 + 5 + 0 + 3 = 17$ сложений-вычитаний. В общем случае среднее число суммирований на один разряд сокращается с 4,5 до 2,5. К оборудованию, изображенному на рис. 5.25, а, при этом добавляется комбинационная схема для проверки логических условий s_0, s_1, s_3 (имеющих тот же смысл, что и в п. 5.6), а также триггер признака φ . Предположим для определенности, что умножение выполняется по первому способу.

Тогда

$$\begin{aligned} s_0 &= \bar{\varphi} 0 \vee \varphi 9, & s_1 &= \bar{\varphi} 5 \vee \varphi 0 \vee 1 \vee 2 \vee 3 \vee 4, \\ s_3 &= \bar{\varphi} 9 \vee \varphi 5 \vee 6 \vee 7 \vee 8, & R_\varphi &= s_1, & S_\varphi &= s_3, \end{aligned}$$

где R_φ и S_φ — функции возбуждения триггера признака φ ; 0, 1, ..., 9 — очередные цифры множителя.

Пример 4. Пусть $X = 0,109$, $Y = 0,625$. Табл. 81 представляет собой цифровую диаграмму состояний узлов блока умножения, в котором используется рассматриваемый способ ускорения при указанных X и Y , где T_Φ — эдигер признака Φ ; S — выполняемая операция.

Таблица 81

RGX	T_Φ	S	x_n	RGZ	RGZ'	CTC	RGY
109	0	$-Y$	9	00,	000	00	0,627
			+1	99,373			
010	1	Сдв		99,937	300	01	то же
	0	$+Y$	0	00,564	300		
001		Сдв	1	00,056	430	10	»
		$+Y$	-1	00,683			
000	0	Сдв	0	00,068	343	11	»
		K					

Из-за высокой эффективности ускорения умножения с преобразованием множителя этот метод используют как отдельно, так и в сочетании с аппаратными методами ускорения умножения десятичных чисел. В принципе все рассмотренные аппаратные методы ускорения умножения двоичных чисел могут быть в той или иной степени переюжены на умножение десятичных чисел. Однако применять тот или иной аппаратный метод ускорения в десятичном множительном блоке целесообразно только в том случае, если он обеспечивает существенное окрашение числа суммирований, на что в десятичном блоке уходит $0 \div 90$ % времени. Один из таких методов состоит в подготовке чисел, кратных множимому. Идея метода заключается в том, что наряду регистром множителя RGY в операционный блок (рис. 5.25, б) вводится регистр RGY' , в котором хранится число $Y' = \alpha Y$, где $\alpha \in \{2, 3, 4, 5\}$.

Работа блока начинается с формирования числа αY , которое осуществляется путем суммирования α раз слов, записанных в RGY и RGY' . На эти регистры α раз поступают сигналы B_1RGY , B_1RGY' , C_1RGY' . Далее начинается умножение с использованием регистров RGY или RGY' в соответствии с табл. 82, где x_n — очередная цифра преобразованного множителя, S_1 и S_2 — операции, производимые в первом и втором такте сложения, B_11 и B_12 — микрооперации выдачи слов с RGZ по каналам 1 и 2; N — количество тактов суммирования при умножении на один десятичный разряд. Например, если $\alpha = -3$, то в первом такте осуществляется суммирование слов из RGZ и RGY' , а во втором такте — из RGZ и RGY , в результате чего сумме частичных произведений прибавляется число $-4Y + Y = -3Y$. Время выполнения операции умножения будет равно

$$t_y \geq \alpha t_+ + n(t_0 + H(\alpha)t_+),$$

где $H(\alpha)$ — среднее число суммирований при умножении на один разряд множителя. Нетрудно подсчитать, что

$$H(2) = H(3) = 1,5, H(4) = 1,4,$$

т. е. при $\alpha = 4$

$$t_y \geq 4t_+ + n(t_c + 1,4t_+).$$

Дальнейшего повышения быстродействия можно достичь путем подготовки двух чисел, кратных множимому. Однако эффективность такого способа ускорения быстро падает. Например, использование двух кратных типа $2Y$ и $4Y$ дает 1,2 суммирования на разряд, что всего на 0,2 меньше, чем при одном кратном типа $4Y$. Подготовку чисел, кратных множимому, можно не осуществлять, если между RGY и сумматором включить комбинационную схему КСУ (рис. 5.25, б), реализующую операцию, например, удвоения множимого (табл. 83,

Таблица 82

$\alpha = 4$				
x_n	S_1	S_2	Микрооперации на RGZ	N
-4	-4Y	+0	B_12	1
-3	-4Y	+Y	B_12, B_11	2
-2	-Y	-Y	B_11, B_11	2
-1	-Y	+0	B_11	1
0	+0	+0	—	0
1	+Y	+0	B_11	1
2	+Y	+Y	B_11, B_11	2
3	+4Y	-Y	B_12, B_11	2
4	+4Y	+0	B_12	1
5	+4Y	+Y	B_12, B_11	2

Таблица 83

x_i	$x_{i+1} < 5$	$x_{i+1} \geq 5$
	x'_i	x'_i
0	0	1
1	2	3
2	4	5
3	6	7
4	8	9
5	0	1
6	2	3
7	4	5
8	6	7
9	8	9

где x_i — i -я цифра множимого, а x'_i — i -я цифра удвоенного множимого). При этом отпадает необходимость в сдвигающем регистре RGY' (так как микрооперации B_1 и Π_1 одновременно можно осуществлять только на сдвигающем регистре) и в дополнительных цепях выдачи слов с RGZ . Уменьшается также число логических элементов во входных цепях сумматора. Однако по сложности КСУ равноценна сумматору, хотя ее быстродействие будет выше из-за отсутствия цепей переноса.

Повысить быстродействие десятичных блоков умножения можно также за счет использования комбинационных схем, реализующих таблицы произведений $u_{ij} = x_i y_j \pmod{10}$ и переносов w_{ij} при таком умножении. В табл. 84 в каждой клетке указаны значения u_{ij} (левый верхний угол) и w_{ij} (нижний правый угол). Блок умножения в этом случае должен содержать два сумматора: один для суммирования слов с выходов таблиц переносов и произведений с целью получения произведений вида Yx_i ; другой сумматор — для образования очередной суммы частичных произведений. Нетрудно подсчитать, что время умножения будет равно

$$t_y \geq n(t_c + 0,9t_+),$$

где 0,9 — число суммирований на один десятичный разряд множителя. Заметим, что в качестве комбинационных схем для u_{ij} и w_{ij} можно использовать постоянные запоминающие устройства.

		x_i									
		0	1	2	3	4	5	6	7	8	9
y_j	0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
	1	0/0	1/0	2/0	3/0	4/0	5/0	6/0	7/0	8/0	9/0
	2	0/0	2/0	4/0	6/0	8/0	0/1	2/1	4/1	6/1	8/1
	3	0/0	3/0	6/0	9/0	2/1	5/1	8/1	1/2	4/2	7/2
	4	0/0	4/0	8/0	2/1	6/1	0/2	4/2	2/2	8/3	6/3
	5	0/0	5/0	0/1	5/1	0/2	5/2	0/3	5/3	0/4	5/4
	6	0/0	6/0	2/1	8/1	4/2	0/3	6/3	2/4	8/4	4/5
	7	0/0	7/0	4/1	1/2	8/2	5/3	2/4	9/4	6/5	3/6
	8	0/0	8/0	6/1	4/2	2/3	0/4	8/4	6/5	4/5	2/7
	9	0/0	9/0	8/1	7/2	6/3	5/4	4/5	3/6	2/7	1/8

5.8. ОСНОВНЫЕ АЛГОРИТМЫ И СТРУКТУРЫ ОПЕРАЦИОННЫХ БЛОКОВ ДЛЯ ДЕЛЕНИЯ В ПРЯМЫХ КОДАХ

Операция деления чисел в программах встречается сравнительно редко (обычно частота деления γ_d не превышает 0,02), однако реализация ее по подпрограмме требует значительных затрат времени. Поэтому в большинстве современных ЭВМ деление реализуется специальным операционным блоком. Известно два основных алгоритма выполнения деления чисел, а именно: деление с восстановлением и без восстановления остатка. Реализация этих алгоритмов требует примерно одинаковых аппаратных затрат, однако при делении по первому алгоритму выполняется больше операций сложения и вычитания. Поэтому в ЭВМ используется, как правило, деление без восстановления остатка. При этом знаковые разряды и цифры мантисс обрабатываются отдельно. Знак результата определяется путем суммирования по модулю 2 цифр, записанных в знаковых разрядах.

Алгоритм деления чисел без восстановления остатка сводится к выполнению следующих действий.

1. Вычесть из делимого X делитель Y . Если результат $R_0 \geq 0$, то старшая цифра Z_0 частного, относящаяся к разряду целых, равна 1, а если $R_0 < 0$, то нулю. Полученная разность является первым остатком.

2. Удвоить остаток.

3. Если $R_i < 0$, то добавить, а если $R_i \geq 0$, то вычесть делитель. Если вновь полученный остаток $R_{i+1} \geq 0$, то очередная цифра частного $z_{i+1} = 1$, а если $R_{i+1} < 0$, то $z_{i+1} = 0$.

4. Повторить пункты 2 и 3 n раз.

При реализации этого алгоритма может оказаться, что $Z_0 = 1$. Это свидетельствует о переполнении разрядной сетки или нарушении нормализации (для машин с плавающей запятой). Если делитель был нормализован, то деление можно продолжать, так как результат

деления будет верным, но находящимся в пределах $1 \leq Z < 2$. Если же делитель ненормализован, то деление должно быть прекращено, так как возможна потеря старших разрядов остатка и результат будет ошибочным. Далее будем считать, что $\frac{X}{Y} < 1$.

Пункт 2 данного алгоритма можно заменить пунктом «Уменьшить в два раза делитель Y ». Наличие двух интерпретаций второго пункта дает два основных варианта построения блоков деления. В блоках, построенных по первому варианту, осуществляется сдвиг влево остатка при неподвижном делителе. В блоках же, построенных по второму

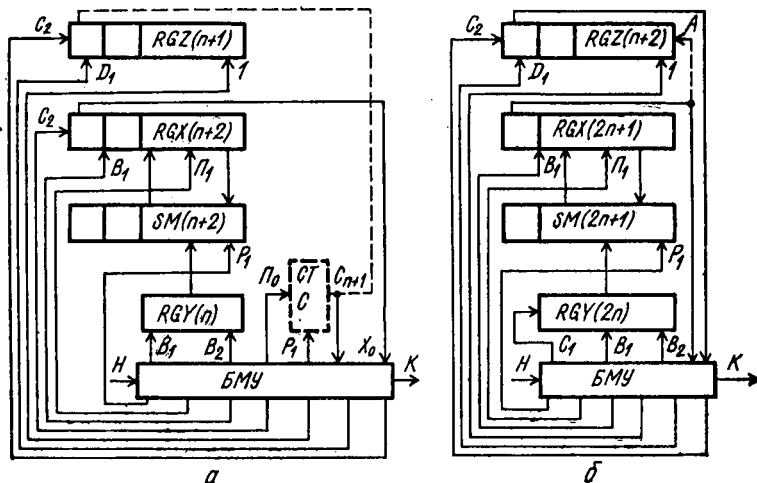


Рис. 5.26

варианту, остаток неподвижен, а взаимный сдвиг остатка и делителя осуществляется за счет сдвига последнего вправо. Как в первом, так и во втором случае блок деления должен содержать регистры RGX , RGY , RGZ и сумматор. Так как производится вычитание делителя Y из делимого X , то RGX и RGY должны быть соединены с сумматором SM , а выходы SM должны быть связаны со входами RGX . Составим микроалгоритм деления (МАД) чисел на указанном оборудовании.

$$MAD = H(P_0 CTC)(\downarrow^4 C_{n+1} \uparrow^1 B_1 RGX \quad X_0 \uparrow^2 B_2 RGY \quad P_1 SM n \uparrow^3 \downarrow^2 B_1 RGY)(\downarrow^3 P_1 RGX \quad D_1 RGZ)(P_1 CTC \quad C_2 RGX \quad C_2 RGZ \uparrow^4 \downarrow^1)K.$$

Здесь $D_1 RGZ$ — микрооперация формирования очередной цифры частного, равной инверсии знака остатка, то есть $z_i = \bar{X}_0$.

Структура блока, реализующего данный микроалгоритм, показана на рис. 5.26, а. Для формирования сигнала K необходимо использовать счетчик сдвигов с периодом $n + 1$. Поскольку регистр RGZ перед началом деления устанавливается в 0, то конец операции можно формировать и с помощью RGZ . Для этого RGZ снабжается дополнительным $n + 2$ -м разрядом и перед началом деления (например, по сигналу H) в младший разряд RGZ записывается 1. После $n + 1$ сдвигов

та 1 попадает в дополнительный $n + 2$ -й разряд, выход которого управляет формированием сигнала K (см. штриховую линию на ис. 5.26, а).

Пример 1. Пусть $Y = \frac{3}{4} = 0,11000$, $X = \frac{15}{32} = 0,01111$, при этом $Z = \frac{X}{Y} = \frac{5}{8} = 0,10100$. Цифровая диаграмма состояний узлов блока деления приведена табл. 85.

Таблица 85

Из цифровой диаграммы видно, что после четвертого вычитания в регистре RGX образовалось нулевое слово, после чего все цифры частного равны нулю. В дальнейшем необходимо производить только сдвиги, в то время как операции вычитания можно опустить.

Если цифры частного с равной вероятностью могут принимать значения 0 и 1, то пропуск такта вычитания позволит сократить время деления в среднем не более чем на величину t_+ . В общем же случае ускорение и стоимость блока деления определяются выражениями

$$t_d \geq (n + 1)(t_c + t_+), \quad C = \alpha n + \beta.$$

Рассмотрим теперь второй вариант построения блока деления, при котором остаток неподвижен (ис. 5.26, б). Отличие данного

RGZ	S	RGX	RGY
00,00000	Н —Y	00,01111	11000
00,00001		11,01000	то же
00,00010	Сдв Y	11,10111	
		11,01110	»
00,00101	Сдв —Y	00,11000	»
		00,00110	
00,01010	Сдв Y	00,01100	»
		11,01000	»
00,01010	Сдв Y	11,10100	
		11,01000	»
00,10101	Сдв —Y	00,11000	»
		00,00000	
01,01010	Сдв Y	11,01000	»
		10,10000	»
10,10100	Сдв K	00,11000	»
		11,01000	»
		10,10000	»

блока от ранее рассмотренного состоит в том, что регистры RGZ и RGY должны быть сдвигающими, RGX — двутактным; регистры RGX и RGY и сумматор должны содержать удвоенное число разрядов. При этом сдвиг осуществляется только в регистрах RGY и RGZ . Микроалгоритм работы данного блока аналогичен микроалгоритму работы блока, построенного по первому варианту. Поскольку результат вычитания (остаток) неподвижен, то можно совмещать операцию вычитания со сдвигом. Операция сдвига в RGZ выполняется только после образования знака остатка, т. е. после окончания процесса формирования сигнала A (рис. 5.26, б). Вследствие этого сдвиг можно осуществлять по сигналу $\Pi_1 RGX$ и совмещение операций будет не полным. Но так как обычно на сдвиг и прием слова требуется примерно одно и то же время, то длительность одного такта деления будет равна $\alpha t_+ < t_+ + t_c$, где $\alpha = \frac{t_b + t_s}{t_b + t_n}$ — коэффициент,

близкий к 1. Поэтому

$$t_d = (n + 1)\alpha t_+ < t_d = (n + 1)(t_+ + t_c).$$

Аппаратурные затраты на построение данного блока можно значительно уменьшить, если длину RGX , RGY и SM сократить до $n + l$ разрядов, где l определяется из условия

$$l \geq 1 + \log_2 (n - l - 1).$$

Такое количество разрядов необходимо для того, чтобы погрешность от отбрасывания младших разрядов остатка и деления не превысила половины веса младшего разряда результата.

Сравнивая первый и второй блоки деления по стоимости и быстродействию, можно сделать вывод, что по стоимости второй вариант уступает первому на 10—20 %, а по быстродействию на 15—25 % превосходит его. Следовательно, по эффективности первый и второй варианты блоков деления примерно равноценны. Следует отметить, что в блоках деления с фиксированной запятой должна быть предусмотрена возможность фиксации переполнения Q разрядной сетки. Осуществляют это путем формирования сигнала совпадения $\bar{X}_0$ с сигналом первого такта деления, который получают дешифруя нулевое состояние СТС или состояние 00—01 регистра RGZ .

В тех АЛУ, в состав которых входит быстродействующий множительный блок (например, реализующий аппаратный метод ускорения 2-го порядка), деление можно выполнять путем умножения делимого на обратную величину от делителя $V = \frac{1}{Y}$. Основная трудность при таком подходе к построению блока деления заключается в получении достаточно точного представления V . В связи с этим для вычисления V пользуются итерационными формулами. Пусть известно число V_i , представляющее V с относительной погрешностью ε_i ,

$$\varepsilon_i = \frac{V - V_i}{V} = 1 - \frac{V_i}{V} = 1 - YV_i.$$

Поэтому при достаточно малой погрешности ε_i можно считать, что

$$V \approx V_i (1 + \varepsilon_i) = V_i (2 - YV_i).$$

Это выражение используют для нахождения $i + 1$ -го приближения к величине V

$$V_{i+1} = V_i (2 - YV_i).$$

Относительная погрешность представления V_{i+1} величины V в этом случае составит

$$\frac{V - V_{i+1}}{V} = \frac{V - V_i (2 - YV_i)}{V} = \frac{V - V_i \left(2 - \frac{V_i}{V}\right)}{V} = \frac{(V - V_i)^2}{V^2} = \varepsilon_i^2.$$

Аналогично можно показать, что относительная погрешность $i + 2$ -го приближения равна ε_i^4 и т. д. Следовательно, итерационный процесс вычисления V будет сходящимся, если начальное приближение V_0 выбрано так, что относительная погрешность ε_0 удовлетворяет условию $|\varepsilon_0| < 1$. При $|\varepsilon_0| \ll 1$ с каждой новой итерацией число

верных разрядов V будет удваиваться, так как изменение погрешности характеризуется рядом $\varepsilon_0, \varepsilon_0^2, \varepsilon_0^4, \dots$. Поскольку

$$\varepsilon_0 = 1 - YV_0,$$

то условие $|\varepsilon_0| < 1$ можно записать как

$$0 < |1 - YV_0| < 1;$$

$$0 < YV_0 < 2;$$

$$0 < V_0 < \frac{2}{Y}.$$

Для двоичного положительного нормализованного делителя ($\frac{1}{2} \leq Y \leq 1$) последнее условие имеет вид

$$0 < V_0 < 2.$$

Отсюда следует, что в качестве V_0 можно было бы принять некоторое фиксированное число из указанного диапазона. Однако при этом для некоторых значений Y итерационный процесс был бы очень долгим. Поэтому в составе оборудования АЛУ целесообразно иметь ПЗУ для начальных приближений V_0 . Деление в этом случае сводится к выборке из ПЗУ величины V_0 , нескольким итерациям для определения V с достаточной точностью и окончательному умножению X на V . Необходимые последовательности управляющих сигналов формируются БМУ.

Помимо ПЗУ и оборудования, входящего в состав блока умножения, дополнительно необходимы также регистры для хранения YV_i и $2 - YV_i$. Быстродействие такого блока деления в 3—5 раз ниже быстродействия блока умножения. При использовании вместо приближенной формулы $V_{i+1} \approx V_i(1 + \varepsilon_i)$ других более точных формул (например, $V_{i+1} \approx V_i(1 + \varepsilon_i + \varepsilon_i^2)$), можно сократить число итераций, необходимых для получения V . Однако длительность каждой итерации несколько увеличивается.

5.9. МЕТОДЫ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ БЛОКОВ ДЕЛЕНИЯ

Методы ускорения деления по аналогии с ускорением умножения делятся на логические, аппаратные и комбинированные. Из известных логических методов наибольшее распространение получили следующие два метода. Первый из них применим в случае использования в блоке деления комбинационного сумматора. Если результат очередного вычитания отрицательный, то вместо приема остатка на регистр RGX осуществляется сдвиг, а очередной цифре частного приписывают значение 0. Заметим, что такой алгоритм деления иногда рассматривают как один из вариантов деления с восстановлением остатка, хотя никаких микроопераций, связанных с восстановлением, здесь не выполняется, поскольку предыдущий положительный остаток сохраняется на RGX . Структура блока, реализующего этот алгоритм, показана на рис. 5.27, где A — выход знакового разряда сумматора. Если

в основу построения блока положен алгоритм с неподвижным делителем, то время деления составит

$$t_d \geq \frac{1}{2} (n+1)(t_c + t_+ + \alpha t_+) = (n+1) \left(\frac{1}{2} t_c + \beta t_+ \right) < \\ < (n+1)(t_c + t_+),$$

где $\alpha = \frac{t_b + t_s}{t_b + t_n}$ — коэффициент, близкий к 1, а $\beta = \frac{1+\alpha}{2}$. При этом предполагается, что цифры 0 и 1 в частном появляются с одинаковыми вероятностями. Помимо повышения быстродействия такой алгоритм обеспечивает некоторое упрощение суммирующего блока за счет устранения цепей реализации микроопераций B_1 на регистре RGY .

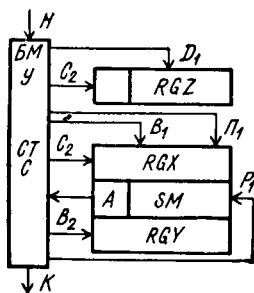


Рис. 5.27

Второй метод может быть использован как при неподвижном делителе, так и при неподвижном остатке. Однако наиболее легко он реализуем при нормализованном неподвижном делителе. В его основе лежит идея быстрого (без суммирований — вычитаний) получения групп цифр частного типа 00...0 или 11...1 (т. е. групп, состоящих из одинаковых цифр). Группы из $m-1$ нулей в частном может возникнуть, например, когда в m старших разрядах очередного положительного остатка будут нули. Действительно, если

$$R_i = 0,00 \dots 01 \dots,$$

$\underbrace{\hspace{1.5cm}}_m$

то

$$\frac{1}{2^m} > R_i > 0, \quad z_i = 1.$$

Для получения z_{i+1} сдвигаем R_i и вычитаем из него Y :

$$R_{i+1} = 2R_i - Y < \frac{1}{2^m} \cdot 2 - Y < \frac{1}{2^{m-1}} - \frac{1}{2},$$

то есть при $m > 1$ и нормализованном делителе Y

$$R_{i+1} < 0; \quad z_{i+1} = 0.$$

Далее

$$R_{i+2} = 2R_{i+1} + Y = 2^2R_i - 2Y + Y = 2^2R_i - Y < \frac{1}{2^{m-2}} - \frac{1}{2}.$$

Поэтому при $m > 2$

$$R_{i+2} < 0; \quad z_{i+2} = 0;$$

в $i+m-1$ -м такте получим

$$R_{i+m-1} = 2^{m-1}R_i - Y < \frac{1}{2} - \frac{1}{2} = 0; \quad z_{i+m-1} = 0.$$

В следующем такте остаток равен

$$R_{i+m} = 2^m R_i - Y < 1 - \frac{1}{2} = \frac{1}{2}.$$

В этом случае сделать какой-то вывод о значении z_{i+m} нельзя. Отсюда следует, что при

$$\frac{1}{2^m} > R_i > 0$$

очередные цифры частного имеют структуру $z_{i-1} \underbrace{100 \dots 0}_{m-1} z_{i+m}$. При

этом для вычисления R_{i+m} достаточно сдвинуть R_i на m разрядов влево и вычесть Y .

Аналогично может быть получена группа из $m-1$ единиц. Действительно, пусть очередной остаток имеет вид

$$R_i = 11, \underbrace{11 \dots 10}_m,$$

т. е.

$$-\frac{1}{2^m} < R_i < 0; \quad z_i = 0,$$

тогда

$$R_{i+1} = 2R_i + Y > -2\frac{1}{2^m} + \frac{1}{2}.$$

При $m > 1$ и нормализованном делителе Y получаем

$$R_{i+1} > 0; \quad z_{i+1} = 1.$$

Далее

$$R_{i+2} = 2R_{i+1} - Y = 4R_i + Y > -4\frac{1}{2^m} + \frac{1}{2}.$$

Отсюда при $m > 2$ следует $R_{i+2} > 0$, $z_{i+2} = 1$.

В $(i+m-1)$ -м такте остаток равен

$$R_{i+m-1} = 2^{m-1}R_i + Y > -\frac{1}{2} + \frac{1}{2} = 0; \quad z_{i+m-1} = 1,$$

а остаток R_{i+m}

$$R_{i+m} = 2^m R_i + Y > -1 + \frac{1}{2} = -\frac{1}{2}.$$

Следовательно, сделать вывод о значении z_{i+m} в общем случае нельзя.

Таким образом, при

$$-\frac{1}{2^m} < R_i < 0$$

очередные цифры частного будут иметь вид $z_{i-1} 011 \dots 1 z_{i+m}$. При этом для вычисления R_{i+m} достаточно сдвинуть R_i на m разрядов и прибавить Y .

Рассмотренные случаи не исчерпывают всех возможных ситуаций получения групп одинаковых цифр частного. Действительно, как видно из приведенных выше выкладок, группа единиц может возник-

в старших разрядах и, следовательно, должен быть зафиксирован второй случай. Правда, при сдвигах по нулям остатка (при быстром получении группы нулей частного) указанный случай все же может возникнуть, однако вероятность этого мала. Следовательно, вероятность появления третьего случая можно считать много меньше вероятности появления первого случая. Количественные оценки этих вероятностей подтверждают приведенный выше качественный вывод. Аналогичные выводы можно сделать для четвертого и второго случаев. Таким образом, алгоритм ускоренного деления сводится к выполнению следующих действий.

1. Если $A_0 = A_1 = 0$, то в соответствии с п. 3. алгоритма деления определить очередной остаток, очередную цифру частного и перейти к п. 2. Если $A_0 = 1$ и $A_1 = 0$, то $z_i = 0$ и перейти к п. 2. Если $A_0 = 0$ и $A_1 = 1$, то $z_i = 1$ и перейти к п. 2.

2. Удвоить остаток.

3. Пункты 1 и 2 повторить $n + 1$ раз.

Пример. 1. Пусть $X = \frac{39}{128}$, $Y = \frac{13}{16}$, $Z = \frac{X}{Y} = \frac{3}{8}$. Состояния узлов блока деления иллюстрируются табл. 86.

Таблица 86

RGZ	Такты	A_0	A_1	S	RGX	RGY
00,0000000 00,0000001 00,0000010	1	1 0	0 0	H Сдв —Y	00,0100111 00,1001110 11,0011000 11,1100110	$Y = 0,1101000$ $-Y = 1,0011000$ —
00,0000100 00,0001001	2 3	0 0	1 0	Сдв Сдв +Y	11,1001100 11,0011000 00,1101000 00,0000000	— —
00,0010011 00,0100110 00,1001100 01,0011000 10,0110000	4 5 6 7 8	1 1 1 1 1	0 0 0 0 0	Сдв Сдв Сдв Сдв К	00,0000000 00,0000000 00,0000000 00,0000000 00,0000000	— — — — —

Структура блока, работающего в соответствии с данным алгоритмом, показана на рис. 5.28, где КС — комбинационная схема, реализующая функции:

$$s_1 = X_0 \overline{(A_0 \vee A_1)}, \quad s_3 = \overline{X_0} \overline{(A_0 \vee A_1)}, \quad s_0 = \overline{s_1 \vee s_3},$$

(как и раньше, единичные значения функций s_0 , s_1 , s_3 соответствуют операциям $+0$, $+Y$, $-Y$). Очередные цифры частного в этом случае формируются по выражению

$$z_i = \overline{X_0} P \vee A_1 B,$$

где $P = s_1 \vee s_3 H$, $B = s_0 H$. Сравнивая структуры операционных блоков, на рис. 5.26 и 5.28, можно сделать вывод, что по аппаратным затратам эти блоки практически одинаковы. При оценке быстродействия рассматриваемого блока будем считать, что цифры 0 и 1 в разрядах остатка (делимого) могут встречаться одинаково часто, число анализируемых разрядов остатка равно трем, а формирование цифр частного совмещено со сдвигом. Тогда операции, выполняемые на очередном такте деления, определяются содержанием старших разрядов остат-

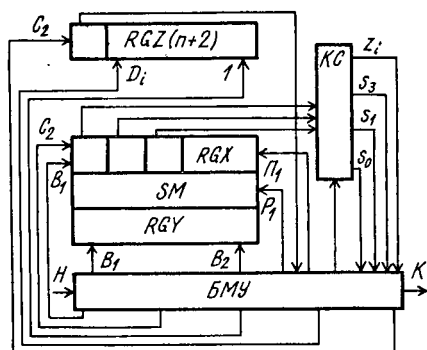


Рис. 5.28

ка. Если в этих разрядах будут комбинации 010, 011, 100, 101, то выполняется сдвиг и суммирование. Если же содержимое старших разрядов равно 000, 001, 110, 111, то следует выполнить сдвиг. Таким образом, суммирование выполняется только в 4 случаях из 8, т. е. число суммирований равно 1/2 на разряд, что является нижней оценкой для числа сложений. Это (как и при умножении) говорит о высокой эффективности логических способов ускорения деления.

Если $t_+ \approx t_c$, дальнейшего ускорения деления путем сокращения числа суммирований добиться не удастся, так как около 75 % времени деления затрачивается на сдвиги. Поэтому для ускорения деления целесообразно использовать дополнительные цепи сдвига. Так, например, если регистры RGX и RGZ снабдить цепями сдвига на 1 и 2 разряда, то в зависимости от результата очередного вычитания или сложения можно производить сдвиг на 1 или 2 разряда. Алго-

Таблица 87

i	A	S	B	C
0	0000	—0	0000	2
1	0001	—0	0001	1
2	0010	—Y	1110 1111 1111 0000	1 2 2 2
3	0011	—Y	1111 0000 0000 0001	2 2 2 1
4	1100	+Y	1110 1111 1111 0000	1 2 2 2
5	1101	+Y	1111 0000 0000 0001	2 2 2 1
6	1110	+0	1110	1
7	1111	+0	1111	2

ритмы деления для данного случая иллюстрирует табл. 87, где A — комбинация цифр в четырех старших разрядах регистра RGX , S — выполняемая операция, B — комбинация цифр в четырех старших разрядах регистра RGX после выполнения операции S , C — количество разрядов, на которое производится сдвиг. Схема блока, реализующего данный алгоритм, отличается от схемы на рис. 5.28 тем, что KC управляется четырьмя разрядами регистра RGX и помимо сигналов s_0, s_1, s_3, z_i вырабатывает два сигнала C_{21} и C_{22} (вместо сигнала C_2 в блоке на рис. 5.28).

$$C_{21} = C_2 G_1;$$

$$C_{22} = C_2 G_2,$$

где

$$G_2 = \bar{Z}_0 (\bar{X}_0 \bar{X}_0' \bar{x}_1 \bar{x}_2 \vee X_0 X_0' x_1 x_2),$$

$$G_1 = \bar{G}_2.$$

Цифры частного формируются в соответствии с выражениями

$$z_i = \bar{X}_0 P \vee A_1 B; z_{i+1} = A_1 (P \vee B).$$

Сигнал G_2 блокируется сигналом Z_0 , так как в случае появления маркерной единицы в знаковом разряде сдвиг можно производить только на один разряд (в противном случае на регистре RGZ будет удвоенное частное).

Пример 2. Составим таблицу состояний узлов рассматриваемого блока при значениях X и Y из предыдущего примера (табл. 88).

Таблица 88

RGZ	Такты	A_0	A_1	G_2	z_i	z_{i+1}	S	RGX	RGY
00,0000000 00,0000001 00,0000010 —	1	1 0 0	0 0 0	0 0 0	0 0 0	0 0 0	H Сдв1 — Y	00,0100111 00,1001110 11,0011000 11,1100110	$Y = 0,1101000$ $-Y = 1,0011000$ —
00,0001001 —	2	0 0	1 0	1 0	0 0	1 0	Сдв2 + Y	11,0011000 00,1101000 00,0000000	—
00,0100110 01,0011000 10,0110000	3 4 5	1 1 1	0 0 0	1 1 0	1 0 0	0 0 0	Сдв2 Сдв2 Сдв1 K	00,0000000 00,0000000 00,0000000 00,0000000	— — — —

Если при оценке быстродействия исходить из сделанных выше предположений, то это дает $\frac{2}{3}t_c$ на одну цифру частного. Таким образом, введение цепей сдвига на 1 и 2 разряда позволяет в 1,5 раза сократить количество сдвигов. Использование регистров с цепями сдвига на 3 и более разрядов позволяет еще уменьшить количество

сдвигов, однако эффективность этого метода быстро падает. Поэтому введение цепей сдвига более чем на три разряда нецелесообразно.

Сократить число суммирований можно за счет введения дополнительных сумматоров. Один из вариантов использования дополнительного сумматора состоит в подготовке с его помощью величин, кратных делителю. Сущность этого метода рассмотрим для случая, когда имеется только одна величина, кратная делителю, а именно $\frac{3}{2}Y$. Предположим, что в процессе деления очередной положительный остаток стал близок делителю $R_i \approx Y$. Это означает, что в очередном такте деления после сдвига будет производиться вычитание. При этом $i + 1$ -й остаток будет положительным, $z_{i+1} = 1$ и в $i + 2$ -м такте необходимо будет производить снова вычитание, т. е.

$$\begin{aligned} R_{i+1} &= 2R_i - Y > 0; \\ R_{i+2} &= 4R_i - 2Y - Y. \end{aligned}$$

По знаку R_{i+2} определяют z_{i+2} и знак операции в $i + 3$ -м такте деления. Последнее выражение можно представить в виде

$$R_{i+2} = 2 \left(2R_i - \frac{3}{2} Y \right)$$

и далее

$$R_{i+3} = 4 \left(2R_i - \frac{3}{2} Y \right) \pm Y.$$

Отсюда следует, что при наличии величины $3/2Y$ в данной ситуации можно два очередных вычитания заменить одним. Аналогичная ситуация возникает, если очередной отрицательный остаток близок к делителю. В этом случае два очередных сложения заменяются прибавлением величины $3/2Y$. При этом цифра $z_{i+1} = 0$, а z_{i+2} определяется по знаку соответствующего остатка.

Получать величину $3/2Y$ можно с помощью дополнительного сумматора и сдвигателя перед началом деления. В последнем случае необходим дополнительный регистр для запоминания $3/2Y$. Кроме того, блок деления должен иметь узел, определяющий, какие операции (± 0 , $\pm Y$ или $\pm 3/2Y$) необходимо производить при данной комбинации цифр остатка и делителя. Таким узлом может быть комбинационная схема, анализирующая, например, три старших разряда остатка и делителя и работающая в соответствии с табл. 89. Эта таблица не является единственно возможной. Так при комбинациях цифр 100 и 101, 101 и 110, 110 и 111 можно вместо операции $\pm Y$ осуществлять операцию $\pm 3/2Y$. Из всех возможных вариантов таблиц выбирают ту, которая обеспечивает наименьшее число суммирований-вычитаний.

Способ подготовки кратных делителя в сочетании с дополнительными цепями сдвига обеспечивает высокую скорость выполнения операции деления. В качестве иллюстрации рассмотрим блок деления одной цифровой ЭВМ, в котором использованы все наиболее сильные способы ускорения, а именно: быстрое получение групп нулей и единиц, введение дополнительных цепей сдвига и использование

Таблица 89

$y_1 y_2 y_3$	$x_1 x_2 x_3$				
	$0x_2x_3$	100	101	110	111
100	± 0	$\pm Y$	$\pm Y$	$\pm \frac{3}{2} Y$	$\pm \frac{3}{2} Y$
101	± 0	$\pm Y$	$\pm Y$	$\pm Y$	$\pm \frac{3}{2} Y$
110	± 0	$\pm Y$	$\pm Y$	$\pm Y$	$\pm Y$
111	± 0	$\pm Y$	$\pm Y$	$\pm Y$	$\pm Y$

Таблица 91

Выбран- ное крат- ное	Знак нового остатка	Цифры частного	
$\pm Y$	0 1	1 0	00000 11111
$+\frac{3}{2} Y$	0 1	01 00	0000 1111
$-\frac{3}{2} Y$	0 1	11 10	0000 1111
$+\frac{3}{4} Y$	0 1	101 100	000 111
$-\frac{3}{4} Y$	0 1	011 010	000 111

кратных делителя. В каждом такте деления выполняется одна из операций ± 0 , $\pm Y$, $\pm 3/2Y$, $\pm 3/4Y$. Делитель всегда нормализован. Если в старших разрядах очередного остатка записаны нули, то производится нормализация остатка (т. е. осуществляется быстрое получение групп одинаковых цифр частного). Для ускорения нормализации использованы дополнительные цепи сдвига на 2, 3, 4, 5, 6 разрядов. Для этого формируются сигналы

Таблица 90

Делитель 0,1у ₂ у ₃ у ₄ у ₅	Прямой ход x ₁ x ₂ x ₃			
у ₂ у ₃ у ₄ у ₅	100	101	110	111
0000				
0001			± $\frac{3}{2}$ Y	
0010				
0011				
0100				
0101				
0110		± Y		
0111				
1000				
1001				
1010				
1011	± $\frac{3}{4}$ Y			
1100				
1101				
1110				
1111				

$$\begin{aligned}
G_1 &= \bar{X}_0 \bar{x}_1 x_2 \vee X_0 x_1 \bar{x}_2; \quad G_2 = \bar{X}_0 \bar{x}_1 \bar{x}_2 x_3 \vee X_0 x_1 x_2 \bar{x}_3; \\
G_3 &= \bar{X}_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 \vee X_0 x_1 x_2 x_3 \bar{x}_4; \\
G_4 &= \bar{X}_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 x_5 \vee X_0 x_1 x_2 x_3 x_4 \bar{x}_5; \\
G_5 &= \bar{X}_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 x_6 \vee X_0 x_1 x_2 x_3 x_4 x_5 \bar{x}_6; \\
G_6 &= \bar{X}_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \bar{x}_6 x_7 \vee X_0 x_1 x_2 x_3 x_4 x_5 x_6 \bar{x}_7.
\end{aligned}$$

Сдвиг на 7 и более разрядов осуществляется за несколько тактов. Например, сдвиг на 15 разрядов осуществляется за три такта (в первом и втором — на 6, а в третьем — на 3 разряда). Однако двух- и более тактная нормализация является редким событием, вероятность которого равна всего 0,05. После нормализации выполняется одна из операций $\pm Y$, $\pm 3/2Y$ или $\pm 3/4Y$. Знак операции определяется знаком очередного остатка.

На выбор того или иного кратного накладывается два ограничения. Во-первых, выбор $3/2Y$ или $3/4Y$ возможен только в том случае, если заведомо известно, что две очередных операции будут иметь один и тот же знак. Во-вторых, выбор кратного должен обеспечивать наименьшее число суммирований-вычитаний. Для этого необходимо, чтобы очередной остаток был как можно ближе к нулю, например, заведомо меньше $1/4$. С учетом этих условий составляется таблица выбора кратных делителя.

В рассматриваемом случае выбор кратных осуществляется с учетом пяти цифр делителя и трех цифр остатка (табл. 90). Легко проверить, что после очередного суммирования-вычитания сдвиг осуществляется не менее чем на два разряда. Поэтому за один полный такт определяется не менее двух цифр частного. Цифры частного формируются в зависимости от выбранного кратного делителя и от знака нового остатка. Если выполнялась операция $\pm Y$ и $R_i \geq 0$, то это означает, что $z_i = 1$, а z_{i+1} и последующие цифры частного определяются по правилу быстрого получения групп одинаковых цифр и для данного случая равны 0. При $R_i < 0$ аналогично получаем $z_i = 0$, z_{i+1} , z_{i+2} , ... = 1. Если выбрано кратное $3/4Y$, то вне зависимости от X_0 предыдущий нормализованный остаток был отрицательным и по абсолютной величине больше Y , поэтому $z_i = 0$, а z_{i+1} определяется по знаку нового остатка. Аналогично можно показать, что при выборке $-3/2Y$ первая цифра частного равна 1, а вторая определяется знаком нового остатка.

При операциях типа $\pm 3/4Y$ за один цикл определяется три цифры частного. Пусть для определенности было выбрано — $3/4Y$. Это означает, что предыдущий нормализованный остаток был больше $1/2$ и меньше Y . Поэтому можно не вычитать Y из остатка, присвоить z_i значение 0, удвоить остаток и из него вычесть Y . Так как удвоенный остаток больше Y , то вновь полученный остаток будет положительным. Следовательно, $z_{i+1} = 1$ и после удвоения нового остатка необходимо вычесть Y . Описанные действия можно выполнить иначе, а именно: вместо удвоения остатка и первого вычитания Y можно из неудвоенного остатка вычесть $1/2Y$, а затем из вновь полученного остатка вычесть $1/4Y$. Эти два вычитания эквивалентны вычитанию $3/4Y$. Отсюда следует, что при выборе $-3/4Y$, $z_i = 0$, $z_{i+1} = 1$,

а z_{i+2} определяется по знаку нового остатка. Аналогично при выборе $+3/4Y$, $z_i = 1$, $z_{i+1} = 0$, а z_{i+2} определяется по знаку нового остатка.

Результаты приведенных выше рассуждений сведены в табл. 91 образования цифр частного. Вертикальная черта в последнем столбце таблицы отделяет цифры частного, полученные в результате полного цикла сложения—сдвига, от цифр частного, полученных при последующей нормализации остатка. При использовании кратного $\pm 3/4Y$ после сложения или вычитания сдвиг остатка на три разряда приводит к нарушению нормализации на один разряд влево. Это, во-первых, вызывает замедление выполнения деления из-за того, что группа одинаковых цифр частного будет формироваться полным циклом сложения—сдвига, а во-вторых, приводит к ненормализованному остатку. Поэтому в случае возникновения указанной ситуации сдвиг производят не на три, а на два разряда и осуществляют операцию $\pm Y$. Таким образом происходит как бы восстановление положительного (отрицательного) остатка, по знаку которого формируются цифры частного. При этом первая цифра инвертируется, так как при восстановлении остатка его знак всегда меняется на противоположный. Остальные же цифры частного формируются по знаку восстановленного остатка при нормализации.

Можно показать, что такая ситуация возникает, когда $R \approx \frac{1}{2}Y$. Отсюда следует другая возможность устранения указанных трудностей, которая заключается во введении кратного делителя, равного $\pm 1/2Y$.

Пример. 3. Пусть $X = \frac{329}{512}$, $Y = \frac{400}{512}$. Тогда процесс деления в соответствии с вышеизложенным можно проиллюстрировать табл. 92.

Таблица 92

RGZ	Такты	S	RGX	RGY
00,000000001	1	$\left\{ \begin{array}{l} H \\ -\frac{3}{4}Y \end{array} \right.$	$\begin{array}{r} 00,101001001 \\ 11,011010100 \\ \hline 00,000011101 \end{array}$	$\begin{array}{l} Y = 00,110010000 \\ -Y = 11,001110000 \end{array}$
00,000010110		Сдв 4	$\begin{array}{r} 00,111010000 \end{array}$	$\begin{array}{l} \frac{3}{4}Y = 00,100101100 \\ -\frac{3}{4}Y = 11,011010100 \end{array}$
00,001011010	2	$\left\{ \begin{array}{l} -Y \\ \text{Сдв 2} \end{array} \right.$	$\begin{array}{r} 11,001110000 \\ 00,001000000 \\ 00,100000000 \\ 11,011010100 \\ \hline 11,111010100 \end{array}$	—
01,011010010	3	$\left\{ \begin{array}{l} -\frac{3}{4}Y \\ \text{Сдв 3} \end{array} \right.$	$\begin{array}{r} 11,010100000 \\ 00,100101100 \\ \hline 11,111001100 \end{array}$	—
10,110100101	4	$\left\{ \begin{array}{l} -\frac{3}{4}Y \\ \text{Сдв 1} \\ K \end{array} \right.$	$\begin{array}{r} 11,110011000 \end{array}$	—

В последнем (четвертом) такте сдвиг осуществляется на один разряд, так как сдвиг на три разряда привел бы к неправильному размещению частного в RGZ . Из рассмотренного примера видно, что для получения 10 цифр частного потребовалось всего четыре такта. В среднем же данный метод обеспечивает получение 3,5 цифр частного за один такт, т. е. время деления составит

$$t_d \geq \frac{n+1}{3,5} (t_c + t_+).$$

Примерный состав оборудования, необходимого для реализации рассмотренного алгоритма деления, показан на рис. 5.29. Регистры RGZ и RGX имеют цепи сдвига на 1, 2, ..., 6 разрядов. Выдача на сумматор $+Y$ или $-Y$ осуществляется с регистра RGY , а слов $\pm 3/4Y$ или $\pm 3/2Y$ с регистра $RGZY$ через сдвигатель CDB .

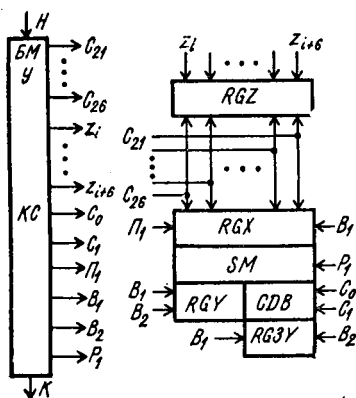


Рис. 5.29

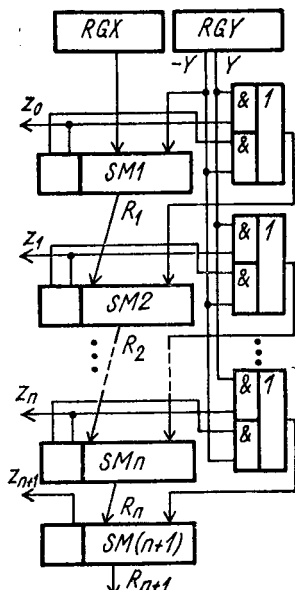


Рис. 5.30

В заключение рассмотрим аппаратный метод ускорения деления, в соответствии с которым для формирования каждого остатка используется отдельный сумматор, т. е. частное получают с помощью комбинационной схемы (рис. 5.30). Такая схема реализует алгоритм деления без восстановления остатка с неподвижным делителем, причем сдвиг остатков осуществляется в процессе передачи их с одного сумматора на другой. Знаковый разряд сумматора, формирующего остаток R_i , управляет передачей делителя Y (прямым или дополнительным кодом) на сумматор, формирующий остаток R_{i+1} . Если частное должно быть представлено n разрядами после запятой, то для построения такого блока деления требуется $n+1$ $n+1$ -разрядных сумматоров. В целом же аппаратные затраты на такой блок будут составлять

$$C = \alpha n^2 + \beta n + \gamma.$$

Однако деление будет здесь выполняться за один такт. Длительность этого такта должна быть не менее времени прохождения сигналов

в самой длинной цепи суммирующей части блока, состоящей из $2n + 1$ одноразрядных сумматоров. Поэтому для рассматриваемого блока деления

$$t_d \geq (2n + 1) t_s.$$

5.10. ДРУГИЕ АЛГОРИТМЫ И БЛОКИ ДЕЛЕНИЯ В ЦИФРОВЫХ ЭВМ

Рассмотренные ранее алгоритмы деления нетрудно приспособить для выполнения этой операции с дополнительными кодами. Если считать, что сдвигается остаток, то алгоритм деления будет следующий.

1. По знаку X_0 делимого и знаку Y_0 делителя определить знак Z_0 частного. В качестве первого остатка взять делимое.

2. Если знаки очередного остатка и делителя совпадают, то из остатка вычесть делитель, а если знаки остатка и делителя разные, то к остатку прибавить делитель. Полученный результат является новым остатком.

3. Если знаки остатка и делителя совпадают, то $z_i = 1$. В противном случае $z_i = 0$.

4. Удвоить остаток.

5. Пункты 2—4 повторить $n + 1$ раз.

Пример 1. Пусть $X = \frac{15}{32}$, $Y = -\frac{3}{4}$. Тогда $Z = -\frac{5}{8}$, а состояния узлов блока, использующего дополнительные коды, иллюстрируются табл. 93, где MIZ_0 — маркерная единица, служащая для определения конца операции.

Таблица 93

RGZ	Такты	S	RGX	RGY
000,00000 000,00011 MIZ_0 000,00111	1 т	$\left\{ \begin{array}{l} H \\ +Y \end{array} \right.$	00,01111 11,01000 <u>11,10111</u> 11,01110 00,11000	$Y = 1,01000$ $-Y = 0,11000$
000,01110	2 т	$\left\{ \begin{array}{l} \text{Сдв} \\ -Y \end{array} \right.$	00,00110 00,01100 11,01000	—
000,11101	3 т	$\left\{ \begin{array}{l} \text{Сдв} \\ +Y \end{array} \right.$	11,10100 11,01000 00,11000	—
001,11010	4 т	$\left\{ \begin{array}{l} \text{Сдв} \\ +Y \end{array} \right.$	00,00000 00,00000 11,01000	—
011,10101	5 т	$\left\{ \begin{array}{l} \text{Сдв} \\ -Y \end{array} \right.$	11,01000 10,10000 00,11000	—
111,01011 <u>1</u>	6 т	$\left\{ \begin{array}{l} \text{Сдв} \\ \rightarrow K \end{array} \right.$	11,01000 10,10000 11,01011	—

ной разряд частного записывать 1, а во все последующие — нули. Блок деления, реализующий данный алгоритм, как по структуре, так и по аппаратным затратам и быстродействию подобен соответствующему блоку, рассмотренному в п. 5.8.

336

жит для формирования сигнала К. Сравнивая блок деления последовательного типа с блоком параллельного типа по быстродействию, аппаратным затратам и эффективности, можно сделать вывод, что аппаратные затраты уменьшаются не более чем в 10 раз, в то время как t_d увеличивается примерно в n раз. Поэтому коэффициент эффективности последовательных блоков всегда меньше, чем параллельных блоков.

Алгоритмы деления десятичных чисел в целом аналогичны алгоритмам деления двоичных чисел, хотя и имеют некоторые особенности. Одна из них состоит в том, что на каждый сдвиг приходится несколько сложений—вычитаний. Так же как и в двоичной системе счисления, здесь имеется два основных алгоритма деления, а именно деление с восстановлением остатка и деление без восстановления остатка. При этом сдвигать можно остаток или делитель. Легко подсчитать, что для реализации алгоритма деления с восстановлением остатка требуется

$$t_d \geq n(t_c + 6,5t_+) + 6,5t_+$$

единиц времени. Быстродействие блока, реализующего алгоритм деления без восстановления остатка, оценивается величиной

$$t_d \geq n(t_c + 5,5t_+) + 5,5t_+. \quad (5.12)$$

Из этих выражений следует, что алгоритм деления без восстановления остатка в десятичной системе обеспечивает рост быстродействия всего на 15—17 %.

Для реализации алгоритма деления без восстановления остатка необходимы регистры RGX , RGY и RGZ , сумматор SM и триггер $T3n$ для запоминания знака выполняемой в данном такте операции. Кроме того, необходимо иметь реверсивный счетчик CTz для подсчета числа суммирований—вычитаний, выполненных в данном такте, а также счетчик CTT для подсчета числа тактов. Первый счетчик необходим для формирования цифр частного, а второй — для фиксации конца операции. В качестве первого счетчика можно использовать последнюю декаду регистра RGZ , а конец деления формировать по появлению маркерной 1 в старшем разряде RGZ . Составим микроалгоритм деления на перечисленных узлах, считая, что единичное состояние триггера $T3n$ соответствует вычитанию делителя в данном такте. Тогда

$$MAD = H(\Pi_0 CTT S_1 T3n)(\downarrow^4 T \uparrow^1 \downarrow^2 R_1 CTz B_1 RGY B_1 RGX) \\ (\Pi_1 RGX)(X_0 \uparrow^2 \downarrow^5 C_{n+1} \uparrow^3 C_2 RGXC_2 RGZ T_1 T3n P_1 CTT \uparrow^4)(\downarrow^1 \downarrow^6 P_1 CTz \\ B_1 RGX B_2 RGY P_1 SMn)(\Pi_1 RGX)(X_0 \uparrow^5 \uparrow^6 \downarrow^3)K,$$

где $S_1 T3n$ — микрооперация установки триггера $T3n$ в единичное состояние, $R_1 CTz$ и $P_1 CTz$ — микрооперации вычитания и прибавления единицы к содержимому CTz , $T_1 T3n$ — микрооперация переключения триггера $T3n$ по входу T , $T \uparrow$ — логическое условие «состояние триггера $T3n$ ». Схема блока, реализующего рассмотренный микроалгоритм, приведена на рис. 5.32. Конец деления фиксируется по маркерной единице в старшем дополнительном разряде регистра

RGZ. Эта единица по сигналу H в начале деления заносится во второй (считая с младших) разряд *RGZ* и после $n + 1$ тактов сложения и n сдвигов прекращает деление. Младший десятичный разряд *RGZ* построен как реверсивный счетчик. При выполнении операции $+Y$ содержимое счетчика уменьшается на 1, а при операции $-Y$ и положительном частичном остатке увеличивается на 1. При сдвиге слова в *RGZ* по сигналу C_2 счетчик устанавливается в 0. В остальном работа такого блока подобна работе ранее рассмотренных двоичных блоков.

Будем считать, что для представления одной десятичной цифры использовано 4 двоичных разряда. Тогда стоимость данного блока может быть представлена как

$$C = 4n_{10}\alpha + \beta, \quad (5.13)$$

где n_{10} — число десятичных разрядов. Учитывая, что $n_{10} \approx 0,3n_2$, представим выражения (5.12) и (5.13) в виде

$$t_d \geq n_2 (0,3t_c + 1,65t_+) + 5,5t_+; \\ C = 1,2n_2\alpha + \beta.$$

Сравнивая эти выражения с соответствующими выражениями для двоичных блоков деления, можно сделать вывод, что десятичные блоки примерно в 1,2 раза сложнее и в 1,2 раза (если $t_+ \approx 2t_c$) менее быстродействующие, чем соответствующие двоичные блоки.

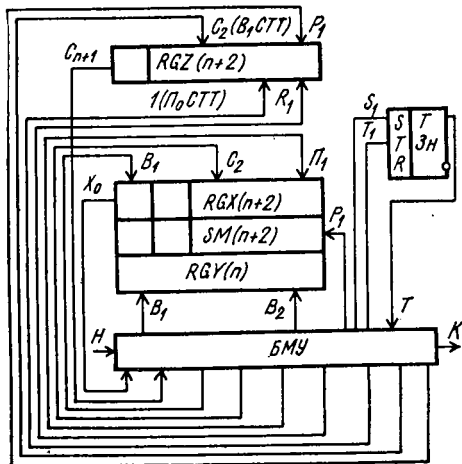


Рис. 5.32

При повышении быстродействия десятичных блоков деления следует прежде всего стремиться к уменьшению числа сложений—вычитаний. Наиболее простым и эффективным таким способом ускорения является использование кратных делителя. Например, наличие кратных типа Y и $3Y$ позволяет уменьшить число суммирований до 2,5 на разряд.

5.11. ОПЕРАЦИОННЫЕ БЛОКИ ДЛЯ РЕДКИХ И ВСПОМОГАТЕЛЬНЫХ ОПЕРАЦИЙ

Одной из сравнительно редко встречающихся в программах является операция извлечения корня (частота ее появления γ_V равна 0,002). Поэтому многие ЭВМ не имеют в составе системы команд операции извлечения корня. В таких ЭВМ корень извлекается по подпрограмме в соответствии, например, с итерационной формулой вида

$$Z_{n+1} = \frac{1}{2} \left(Z_n + \frac{X}{Z_n} \right), \quad (5.14)$$

где $Z_n \rightarrow \sqrt{X}$ при $n \rightarrow \infty$.

Однако анализ длительности выполнения этой операции по итерационным формулам показал, что значительную часть своего времени

машина тратит на извлечение корня. Действительно, для одного цикла итераций по формуле (5.14) требуется выполнить пять трехадресных команд (см. гл. 6), для чего необходимо $t_V = 5(t_{АЛУ} + \rho t_0)$ единиц времени. Здесь $t_{АЛУ}$ — время выполнения операций в АЛУ; t_0 — длительность цикла обращения к ЗУ; коэффициент ρ для трехадресной цифровой ЭВМ равен 4. Если для вычисления корня необходимо, например, 15 итераций, то это займет $t_V = 75(t_{АЛУ} + 4t_0)$ единиц времени. При $t_{АЛУ} \approx t_0$ получим $t_V = 375t_{АЛУ}$. Отсюда следует, что

$$\frac{\gamma_V t_V}{(1 - \gamma_V)(t_{АЛУ} + 4t_0) \gamma_V t_V} \cdot 100 \% \approx 13 \%$$

времени машина занята извлечением корня. Поэтому, с целью повышения быстродействия, во многих современных цифровых ЭВМ операция извлечения корня реализуется схемно, т. е. имеется специальный блок, реализующий эту операцию.

Так же как и деление, операцию извлечения корня чаще всего реализуют методом последовательных приближений, при котором за один цикл сложения-сдвига определяется одна цифра результата. Обозначим

$$Z = \sqrt{X}; \quad Z_i = 0, z_1 z_2 \dots z_i,$$

где X — операнд; Z — результат; Z_i — результат, определенный с точностью до i разрядов, причем $Z = Z_n$. Разность $X - Z_i^2$ обозначим как X'_i . С увеличением i величина Z_i^2 стремится к X , а X'_i — к нулю. Если вычислено i цифр результата, то можно записать, что

$$X_i = X'_i 2^i = (X - Z_i^2) 2^i.$$

Тогда при $z_{i+1} = 0$ имеем

$$X_{i+1} = (X - Z_{i+1}^2) 2^{i+1} = (X - (Z_i + 0 \cdot 2^{-i-1})^2) 2^{i+1} = 2X_i. \quad (5.15)$$

Если же $z_{i+1} = 1$, то

$$\begin{aligned} X_{i+1} &= (X - (Z_i + 1 \cdot 2^{-i-1})^2) 2^{i+1} = \\ &= (X - Z_i^2 - 2Z_i 2^{-i-1} - 2^{-2i-2}) 2^{i+1} = 2X_i - 2Z_i - 2^{-i-1} = \\ &= 2X_i - (2Z_i + 2^{-(i+1)}). \end{aligned} \quad (5.16)$$

Из определений Z_i и X'_i следует, что

$$Z_i^2 \leq X < (Z_i + 2^{-i})^2$$

или

$$0 \leq X'_i < 2Z_i 2^{-i} + 2^{-2i}. \quad (5.17)$$

Отсюда получим

$$0 \leq X_i < 2Z_i + 2^{-i}.$$

Аналогично имеем

$$0 \leq X_{i+1} < 2Z_{i+1} + 2^{-i-1}. \quad (5.18)$$

Очередную $(i+1)$ -ю цифру результата необходимо выбирать такой, чтобы выполнялось условие (5.18). Это можно осуществить,

формируя X_{i+1} по выражению (5.16) и анализируя его знак. Если $X_{i+1} > 0$ и условие (5.17) выполнялось, то и (5.18) будет выполняться. Если же $X_{i+1} < 0$, то это означает, что $z_{i+1} = 0$. При этом надо восстановить положительный остаток, а X_{i+1} формировать путем сдвига X_i , т. е. в соответствии с выражением (5.15). В начале операции $Z_0 = 0$, $X_0 = X$.

Один из возможных вариантов блока, реализующего такой алгоритм извлечения корня, состоит из сумматора SM , счетчика тактов $СТТ$, регистров операнда RGX и результата RGZ , а также вспомогательного регистра RGY , служащего для формирования числа 2^{-i} . В старший разряд этого регистра перед началом операции записывается 1, которая с каждым тактом сдвигается вправо. Микроалгоритм работы такого блока можно записать как

$$MAK = H (P_0 CTT) (\downarrow^3 C_n \downarrow^1 B_{12} RGX \downarrow^{B_{22}} RGZ \downarrow^{B_1} RGY) (P_1 RGX \downarrow^{D_1} RGZ \downarrow^{P_1} CTT) (X_0 \uparrow^2 \downarrow^4 C_1 RGY \downarrow^{C_2} RGX \uparrow^3 \downarrow^2 B_{12} RGX \downarrow^{B_{12}} RGZ \downarrow^{B_1} RGY) (P_1 RGX \uparrow^4) \downarrow^1 K.$$

Здесь B_{12} и B_{22} — микрооперации B_1 и B_2 , выполняемые со сдвигом передаваемого слова на один разряд влево, D_1 — микрооперация формирования цифры результата. Заметим, что регистр RGX должен быть двухтактным, так как должен одновременно реализовать микрооперации B_1 и P_1 .

Пример 1. Составим цифровую диаграмму состояний узлов блока, реализующего описанный алгоритм при $X = \frac{121}{256}$ (табл. 94).

Таблица 94

RGZ	Такты	S	RGX
00,00000000	1 т	H	00,01111001
		$\left\{ \begin{array}{l} \text{Сдв} \\ -(2Z_0 + 2^{-1}) \end{array} \right.$	00,11110010 11,10000000
00,1	2 т	$\left\{ \begin{array}{l} \text{Сдв} \\ -(2Z_1 + 2^{-2}) \end{array} \right.$	00,01110010 00,11100100 10,11000000
00,10		$\left\{ \begin{array}{l} \\ +(2Z_1 + 2^{-2}) \end{array} \right.$	11,10100100 01,01000000
		$\left\{ \begin{array}{l} \text{Сдв} \\ -(2Z_2 + 2^{-3}) \end{array} \right.$	00,11100100 01,11001000 10,11100000
00,101	4 т	$\left\{ \begin{array}{l} \text{Сдв} \\ -(2Z_3 + 2^{-4}) \end{array} \right.$	00,10101000 01,01010000 10,10110000
00,10110000			00,00000000 $\underbrace{\hspace{1cm}}_K$

Сигнал K можно формировать также с помощью регистра RGY , длина которого в таком случае должна быть равна $n + 1$, а единица

старший его разряд записывается сигналом H . Некоторого ускорения операции можно достичь, формируя сигнал K по нулевому содержимому RGX (табл. 94). Если считать, что цифры 0 и 1 в результате операции могут появляться одинаково часто, то

$$t_+ \geq n(t_c + 1,5t_+).$$

Значительно ускорить выполнение операции можно за счет использования алгоритма извлечения корня без восстановления остатка. В этом случае результат получается представленным в двоичной системе с цифрами $\{1, \bar{1}\}$. Пусть как и раньше

$$Z = \sqrt{X}, \\ Z_i = 1, z_1 z_2 \dots z_i, Z_{i+1} = Z_i + 2^{-i-1} z_{i+1},$$

де

$$z_1, z_2, \dots, z_i \in \{1, \bar{1}\}.$$

Обозначим

$$X_i = (X - Z_i^2) 2^{-n-1-2i}.$$

Уясним, каким образом связаны между собой X_i и X_{i+1} . Если $z_{i+1} = 1$, то

$$X_{i+1} = (X - (Z_i + 1 \cdot 2^{-i-1})^2) 2^{-n-1+2(i+1)} = 4X_i - (2Y_i + 2^{-n-1}),$$

де $Y_i = Z_i 2^{-n+i}$. Следовательно,

$$Y_{i+1} = Z_{i+1} 2^{-n+1+i} = 2Z_i 2^{-n+i} + 2^{-i-1-n+i+1} = 2Y_i + 2^{-n}.$$

Отсюда получим

$$X_{i+1} = 4X_i - Y_{i+1} + 2^{-n-1}.$$

Аналогично при $z_{i+1} = \bar{1}$ имеем

$$X_{i+1} = 4X_i + Y_{i+1} + 2^{-n-1}; \quad Y_{i+1} = 2Y_i - 2^{-n}.$$

В общем случае

$$\left. \begin{aligned} X_{i+1} &= 4X_i - z_{i+1} Y_i + 2^{-n-1}; \\ Y_{i+1} &= 2Y_i + z_{i+1} 2^{-n}. \end{aligned} \right\} \quad (5.19)$$

Цифры результата надо выбирать таким образом, чтобы при любом выполнялось условие

$$Z_i - 2^{-i} < \sqrt{X} < Z_i + 2^{-i}. \quad (5.20)$$

Тогда при $i = n$

$$Y_i = Y_n = Z_n 2^{-n-m} = Z_n \approx \sqrt{X}.$$

Условие (5.20) можно записать так:

$$\begin{aligned} (Z_i - 2^{-i})^2 &< X < (Z_i + 2^{-i})^2; \\ -2Z_i 2^{-i} + 2^{-2i} &< (X - Z_i^2) < 2Z_i 2^{-i} + 2^{-2i}; \\ -2Z_i 2^{-n+i-1} + 2^{-n-1} &< (X - Z_i^2) 2^{-n-1+2i} < 2Z_i 2^{-n+i-1} + 2^{-n+1}. \end{aligned}$$

Окончательно получим

$$-Y_i + 2^{-n-1} < X_i < Y_i + 2^{-n-1}. \quad (5.21)$$

Покажем, что если цифры z_i формировать по правилу «Знак $Z_i = \text{Знак } X_i$ », X_{i+1} и Y_{i+1} вычислять по выражениям (5.19) и если для какого-либо i выполняется условие (5.20) (или условие (5.21), что то же самое), то оно будет выполняться и для любого $j > i$. Пусть для определенности $X_i > 0$ и $z_i = 1$. Тогда из правой части (5.21) имеем

$$X_{i+1} = 4X_i - Y_{i+1} + 2^{-n-1} < 4(Y_i + 2^{-n-1}) - Y_{i+1} + 2^{-n-1}. \quad (5.22)$$

В данном случае $Y_{i+1} = 2Y_i + 2^{-n}$. Поэтому из (5.22) окончательно получим

$$X_{i+1} < 2(Y_{i+1} + 2^{-n} - 2^{-n}) - Y_{i+1} + 2^{-n-1}.$$

Аналогично можно доказать левую часть неравенства (5.21). При $i = 0$ имеем

$$Z_0 = 1; \quad X_0 = (X - 1)2^{-n-1}; \quad Y_0 = 2^{-n}.$$

Отсюда следует, что условие (5.21) при $i = 0$, следовательно, при любом i всегда выполняется, так как для этого необходимо, чтобы

$$-2^{-n} + 2^{-n-1} < (X - 1)2^{-n-1} < 2^{-n} + 2^{-n-1}$$

или

$$0 < X < 4,$$

что заведомо справедливо. Следовательно, выполнение операции сводится к n -кратному выполнению цикла

$$\begin{aligned} z_{i+1} &= \begin{cases} 1 & \text{при } X_i \geq 0, \\ \bar{1} & \text{при } X_i < 0; \end{cases} \\ Y_{i+1} &= 2Y_i + z_{i+1}2^{-n}; \\ X_{i+1} &= 4X_i - z_{i+1}Y_{i+1} + 2^{-n-1} \end{aligned}$$

при $i = 0, 1, \dots, n$; $X_0 = (X - 1)2^{-n-1}$; $Y_0 = 2^{-n}$; $Z_0 = 1$.

В соответствии с таким алгоритмом операции сложения и вычитания необходимо выполнять в двоичной системе с цифрами $\{1, \bar{1}\}$, что, в свою очередь, требует нетрадиционного сумматора и приводит к дополнительным затратам на его построение. Перевод чисел из системы с цифрами $\{1, \bar{1}\}$ в систему с цифрами $\{0, 1\}$ может быть осуществлен в процессе выполнения операции.

Для этого очередную цифру z_i результата необходимо записывать сразу в $n - 1$ -й разряд, в n -й разряд постоянно записывать 1, а старшую положительную цифру отбросить в первом же цикле путем разрыва цепи сдвига между n -м и $n - 1$ -м разрядами регистра RGZ .

Следовательно, выполнение операции $\sqrt{X}$ в системе счисления с цифрами $\{0, 1\}$ сводится к n -кратному выполнению цикла

$$z_i = \begin{cases} 1 & \text{при } X_i \geq 0; \\ 0 & \text{при } X_i < 0. \end{cases}$$

здесь замена индекса $i + 1$ на i при z соответствует сдвигу на один аэряд),

$$Y_{i+1} = 2Y_i + z_i 2^{-n+1},$$

замена показателя $-n$ на $-n + 1$ при 2 также соответствует сдвигу а один разряд),

$$X_{i+1} = \begin{cases} 4X_i - Y_{i+1} - 2^{-n} + 2^{-n-1} & \text{при } X_i \geq 0; \\ 4X_i + Y_{i+1} + 2^{-n} - 2^{-n-1} & \text{при } X_i < 0 \end{cases}$$

здесь 2^{-n} соответствует единице, постоянно записанной в n -й разряд).

Схема блока извлечения корня, работающего в соответствии с рассмотренным алгоритмом, показана на рис. 5.33. Слово с регистра RGX при сдвиге поступает в младшие разряды двукратного регистра RGY . Регистр RGZ имеет $n - 1$ разрядов. При выполнении операции $\pm Y$ в два младших разряда ($n + 1$ -й и n -й) сумматора поступают 1. Единица в $n + 1$ -м разряде соответствует прибавлению 2^{-n-1} при формировании X_{i+1} , а 1 в n -м разряде соответствует постоянно записанной 1 в этом разряде регистра RGZ . В начале операции по сигналу H все разряды регистра RGY устанавливаются в 1. При каждом сдвиге в последний $n - 1$ -й аэряд RGZ заносится очередная цифра результата. Конец операции формируется по счетчику сдвигов CTC . Микроалгоритм извлечения орня в рассматриваемом случае имеет вид

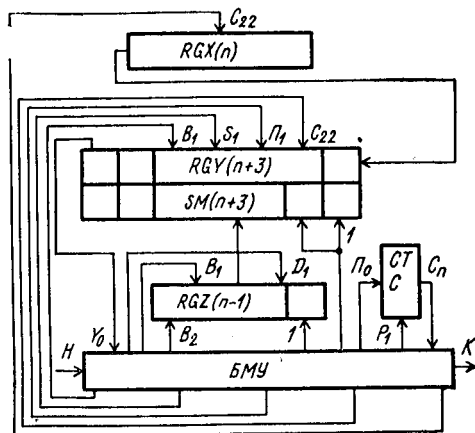


Рис. 5.33

$$MAK = H (P_0 CTC S_1 RGY) (\downarrow^4 C_n \uparrow^1 C_{22} RGX C_{22} RGY D_1 RGZ) (Y_0 \uparrow^2 B_2 RGZ \uparrow^3 \downarrow^2 B_1 RGZ \downarrow^3 B_1 RGY) (\bar{P}_1 RGY P_1 CTC \uparrow^4) \downarrow^1 K,$$

де $S_1 RGY$ — микрооперация установки в 1 всех разрядов RGY .

Пример 2. Составим цифровую диаграмму работы блока, реализующего писанный алгоритм для $X = \frac{121}{256}$ (табл. 95).

Быстродействие блока на рис. 2.79 определяется выражением

$$t_v = n(t_0 + t_+) - t_+.$$

На аналогичных идеях основаны способы извлечения корня в де-ятичной системе счисления.

Как известно, более 50 % всех цифровых ЭВМ работают в двоич-ной системе счисления. Однако исходные данные и результаты реше-ния задачи должны быть представлены в форме, удобной для их вос-приятия, т. е., как правило, в десятичной форме счисления.

RGZ	Такты	S	RGY	RGX	CTC
,0000000 1			00,00000000 0	01111001	0000
0 1		H	11,11111111 1	01111001	
	1 т	Сдв 2	11,11111110 1	11100100	0001
		+Y	00,00000001 1		
01 1			00,00000000 0		
	2 т	Сдв 2	00,00000001 1	10010000	0010
		-Y	11,11111101 1		
010 1			11,11111111 0		
	3 т	Сдв 2	11,11111101 0	01000000	0011
		+Y	00,00000101 1		
0101 1			00,00000010 1		
	4 т	Сдв 2	00,00001010 1	00000000	0100
		-Y	11,11110101 1		
01011 1			00,00000000 0		
	5 т	Сдв 2	00,00000000 0	00000000	0101
		-Y	11,11101001 1		
010110 1			11,11101001 1		
	6 т	Сдв 2	11,10100110 0	00000000	0110
		+Y	00,00101101 1		
,0101100 1			11,11010011 1		
	7 т	Сдв 2	11,01001110 0	00000000	0111
		+Y	00,01011001 1		
0,1011000 1			11,10100111 1		
		Сдв 2	10,10011110 0	00000000	1000
		K			

Известно две разновидности аппаратной реализации операции перевода. В первом случае блок преобразования кодов чисел (БПК) является частью АЛУ и использует его основное оборудование (регистры и сумматор). Во втором случае БПК выполняется обособленно и не используется никакое оборудование АЛУ. В этом случае БПК включается в работу во время обмена информацией между ЗУ и внешними устройствами. При этом АЛУ цифровой ЭВМ освобождается и появляется возможность совмещать решение одной задачи с подготовкой данных для ввода—вывода другой задачи. В п. 3.7 рассмотрены алгоритмы функционирования и вопросы проектирования БПК комбинационного типа. Здесь же рассмотрим БПК других типов.

Наибольшее распространение в ЦВТ нашел метод преобразования путем сдвига и коррекции. В основе этого метода лежат операции умножения и деления на 2, сводящиеся в двоичной системе к обычному сдвигу на один разряд влево или вправо. В десятичной системе кроме сдвига необходима коррекция результата. Рассмотрим сначала деление. Предположим, что десятичное число представлено двоично-десятичным кодом с весами 8, 4, 2, 1. Поэтому сдвиг двоичных цифр, представляющих данную десятичную цифру, на один разряд вправо уменьшает ее в два раза, если только в данную тетраду нет переноса из соседнего старшего десятичного разряда. Если же в старшей тетраде

сдвига было нечетное число, то при сдвиге вправо $1/2$ единицы гаршего разряда переносится в младший разряд и должна быть воспринята в нем как 5. Однако указанная 1 переноса попадает в старший разряд тетрады, вес которого равен 8, а не 5. Поэтому для получения правильного результата необходимо уменьшить содержимое той тетрады на $8 - 5 = 3$ единицы.

Таким образом, для деления десятичного числа на 2 необходимо двинуть двоично-десятичный код данного числа на один разряд вправо и вычесть 3 из тех тетрад, в старших разрядах которых после сдвига оказались 1.

Рассмотрим теперь удвоение десятичного числа, представленного кодом с избытком 3. Сдвиг влево на один двоичный разряд каждую десятичную цифру увеличивает в два раза. Следовательно, при сдвиге избыток кода возрастает до 6 и для получения кода с избытком 3 надо во всех тетрадах вычесть 3. Так как цифры больше 5 в коде с избытком 3 будут представляться двоичным кодом больше 8, то добавление в старшую тетраду при удвоении происходит путем сдвига (переноса) из старшего разряда данной тетрады в младший разряд следующей тетрады. При этом содержимое данной тетрады надо уменьшить на 1 (на 1 старшей тетрады). Но так как 1 старшего разряда данной тетрады перешла в младший разряд следующей тетрады, вес которого в двоичном коде равен 16, фактически произошло уменьшение содержимого данной тетрады на 16 единиц. Следовательно, для коррекции результата в данную тетраду надо добавить $16 - 10 = 6$ единиц. Две писанные коррекции при удвоении можно объединить в одну, а именно: если переноса из данной тетрады не было, то содержимое данной тетрады уменьшается на 3 единицы, если же перенос был — то увеличивается на $-3 + 6 = 3$ единицы. Таким образом, умножение и деление на 2 сводится к левому или правому сдвигу на один разряд и коррекции результата путем добавления или вычитания троек.

Рассмотрим теперь алгоритмы преобразования чисел путем сдвига коррекции. Дробные и целые части чисел преобразуются по-разному.

Преобразовать десятичную дробь в двоичную можно путем умножения на два и анализа 1 переноса из каждой тетрады.

Пример 3. Пусть $X_{10} = 0,6875$. Тогда $X_{(2,10)} = ,0110\ 1000\ 0111\ 0101$, в коде с избытком 3 $X_{(2,10)} = ,1001\ 1011\ 1010\ 1000$. Преобразуем этот двоично-десятичный код в двоичный (табл. 96).

В табл. 96 M — маркерная единица. Для реализации данного алгоритма необходимо иметь два сдвигающих регистра (двоичный десятичный) и цепи коррекций, которые по сложности примерно равны сумматору, однако значительно превосходят его по быстродействию из-за отсутствия цепей переноса между тетрадами. При этом регистры соединены так, что цифры со старшей тетрады десятичного регистра при сдвиге поступают в младший разряд двоичного регистра.

Преобразовать десятичное целое число можно путем деления его на 2 и анализа остатка.

Пример 4. Пусть $X_{10} = 75 = 0111\ 0101$. Тогда процесс преобразования (10) в двоичное число можно проиллюстрировать табл. 97.

Таблица 96

RGZ	RGX	S
0,0001M	,1001 1011 1010 1000	H
0,0011	,0011 0111 0101 0000	Сдв
	+11 +11 +11 +11	Кор
0,0110	0110 1010 1000 0011	Сдв
	,1101 0101 0000 0110	
	-11 +11 +11 -11	
0,1101	,1010 1000 0011 0011	Сдв
	,0101 0000 0110 0110	Кор
	+11 +11 -11 -11	
1,1011	,1000 0011 0011 0011	Сдв
	0000 0110 0110 0110	K

Таблица 98

RGX	S	RGZ
1,1011	H	,0000 0000 0000 0000
0,1101	Сдв	,1000 0000 0000 0000
	Кор	-11 -00 -00 -00
0,0110	Сдв	0101 0000 0000 0000
	Кор	1010 1000 0000 0000
		-11 -11 -00 -00
0,0011	Сдв	,0111 0101 0000 0000
	Кор	,0011 1010 1000 0000
		-00 -11 -11 -00
0,0001	Сдв	0011 0111 0101 0000
	Кор	1001 1011 1010 1000
		-11 -11 -11 -11
		0110 1000 0111 0101

Таблица 97

S	RGX	RGZ
H	0111 0101	M
Сдв	0011 1010	1000000,0
Кор	-00 -11	1100000,0
Сдв	0011 0111	
Кор	0001 1011	1110000,0
	-00 -11	
Сдв	0001 1000	
Кор	0000 1100	0111000,0
	-00 -11	
Сдв	0000 1001	
Кор	0000 0100	1011100,0
	-00 -00	
Сдв	0000 0100	
Кор	0000 0010	0101110,0
	-00 -00	
Сдв	0000 0001	
Кор	-00 -00	0010111,0
Сдв	0000 0001	
К←	0000 0000	1001011,1

Таблица 99

RGZ	S	RGX
0011 0011	H	1001011
0110 0111	Сдв	0010110
-11 -11	Кор	
0011 0100		
0110 1000	Сдв	0101100
-11 -11	Кор	
0011 0101		
0110 1010	-Сдв	1011000
-11 -11	Кор	
0011 0111		
0110 1111	Сдв	0110000
-11 -11	Кор	
0011 1100		
0111 1000	Сдв	1100000
-11 +11	Кор	
0100 1011		
1001 0111	Сдв	1000000
-11 +11	Кор	
0110 1010		
1101 0101	Сдв	0000000
-11 +11	Кор	
1010 1000		
0111 0101	K	
	(Код 8421)	

Оборудование, необходимое для реализации данного алгоритма, такое же, как и в предыдущем случае. Отличие состоит в соединениях: при сдвиге цифра младшего разряда RGX попадает в старший разряд RGZ.

Перевод чисел из двоичной системы в десятичную также осуществляется путем сдвига и коррекции. Эти операции являются обратными с отношением к ранее рассмотренным. Если в первом алгоритме изменить левый сдвиг правым, а умножение на 2 — делением, получим алгоритм перевода двоичной дроби в десятичную.

Пример 5. Переведем в десятичную систему число $X_{(2)} = 0,1011$ (табл. 98).

В этом случае сигнал K уже нельзя получить с помощью маркер-
-ой 1 в RGX . Для этого необходимо использовать либо специальный
-етчик сдвигов, либо схему, фиксирующую нулевое содержимое
 GX . Аналогично получаем алгоритм перевода целого числа из
-воичной системы в десятичную.

Пример 6. Составим цифровую диаграмму состояний узлов блока, реализующего этот алгоритм, если $X_{(2)} = 1001011$ (табл. 99). Конец операции здесь определяется так же, как и в предыдущем случае. Состав оборудования, необходимого для реализации двух последних алгоритмов, легко определить из табл. 98 и 99.

В заключение укажем, что рассмотренные выше операционные
-локи являются лишь частными примерами, и далеко не исчерпывают
-сего многообразия аппаратных средств ЦВТ, применяемых для ре-
-ализации редких и вспомогательных операций.

5.12. ОРГАНИЗАЦИЯ И СТРУКТУРЫ АРИФМЕТИКО-ЛОГИЧЕСКИХ УСТРОЙСТВ

Рассмотренные в предыдущих разделах блоки для выполнения от-
-дельных операций редко используются обособленно, чаще всего эти
-локи объединены в одном арифметико-логическом устройстве. Такое
-бъединение удастся осуществить благодаря тому, что практически
-се операционные блоки состоят из одних и тех же операционных
-злов, а именно: нескольких регистров и сумматора. Отличия про-
-вляются большей частью только в соединениях между операцион-
-ными узлами и последовательностях управляющих сигналов. Рас-
-смотрим более детально методику объединения операционных блоков
-АЛУ.

Как видно из изложенного, задача проектирования операционных
-блоков цифровых ЭВМ в общем случае состоит в построении схемы
-лока, обеспечивающего выполнение заданного набора операций за
-ремя, не превышающее заданного, при минимальных затратах обо-
-удования. Формальных методов решения этой задачи не существует,
-то объясняется большим числом и разнообразием алгоритмов выпол-
-нения операций, многообразием методов аппаратной реализации опера-
-ций, многокритериальностью целевой функции проектирования. По-
-тому проектирование АЛУ и структур операционных блоков цифро-
-ых ЭВМ, особенно на его начальных ступенях, в значительной мере
-сновывается на знаниях, опыте и интуиции разработчиков, т. е. имеет
-вристический характер. Тем не менее можно выделить следующие
-тапы проектирования:

- 1) выбор алгоритмов выполнения отдельных операций;
- 2) составление микроалгоритмов и построение схем операционных блоков для отдельных операций;
- 3) объединение схем операционных блоков отдельных операций;
- 4) объединение микроалгоритмов.

Алгоритмы выполнения отдельных операций разрабатываются, исходя из методов и правил вычислительной математики, машинной арифметики, способов представления информации в цифровых ЭВМ и других факторов, включая квалификацию проектировщиков. При этом элементарные действия, предусматриваемые алгоритмами, должны быть реализуемы на тех аппаратных средствах, которые предполагается использовать в дальнейшем. На этом этапе должны быть решены вопросы кодирования информации, выбраны принципы передачи информации, получены предварительные оценки аппаратных затрат и времени выполнения операций. Далее, зная возможности известных схемотехнических решений, надо определить основные операционные узлы, входящие в состав проектируемого АЛУ или блока. Для упрощения этой задачи предварительно строят схемы для выполнения отдельных операций.

Чтобы учесть возможность выполнения нескольких операций на одном и том же оборудовании, прежде других проектируют схемы для самых сложных операций. При этом целесообразно придерживаться следующего порядка. Вначале определяют набор операционных узлов, необходимых для хранения исходных данных, промежуточных и конечных результатов. Входы и выходы этих узлов соединяют со входами и выходами проектируемого блока. Затем определяют микрооперации, необходимые для пересылки информации, в проектируемый блок и из него. Для реализации элементарных действий, предусматриваемых микроалгоритмом, в схему дополнительно вводят информационные и управляющие шины, логические и запоминающие элементы, операционные узлы.

При выполнении третьего этапа проектирования — объединения отдельных схем операционных блоков — считают, что в АЛУ в течение некоторого промежутка времени может выполняться только одна операция из заданного набора операций. На основе схем, разработанных для отдельных операций, определяют набор операционных узлов, составляющих операционную часть проектируемого АЛУ. Длина операционных узлов (в разрядах) выбирается по максимальному значению, присущему одной из схем для отдельных операций. К узлам подключают входные и выходные шины, посредством которых проектируемый блок или АЛУ соединяется с другими. Затем определяют перечень микроопераций, используемых для ввода операндов и вывода результатов (полученную схему иногда называют «скелетом» блока или АЛУ). Далее объединяют наборы микроопераций отдельных операций. По микроалгоритмам отдельных операций определяют информационные связи между узлами. При выполнении указанных процедур удобно пользоваться таблицами.

Пример 1. Пусть необходимо выполнить объединение схем операционных блоков, реализующих операции сложения — вычитания, умножения и деления по следующим микроалгоритмам:

$$\begin{aligned}
MAC &= H(X_0 \uparrow^1 B_1 RGX \uparrow^2 \downarrow^1 B_2 RGX \downarrow^2 F_0 \uparrow^3 B_2 RGY \uparrow^4 \downarrow^3 B_1 RGY \downarrow^4) \\
&\quad (P_1 RGZ) (Z_0 \uparrow^5 B_1 RGZ \uparrow^6 \downarrow^5 B_2 RGZ \downarrow^6) K; \\
MAU &= H(P_0 CTC \downarrow^3 C_n \uparrow^1 \bar{x}_n \uparrow^2 B_1 RGY \downarrow^1 B_1 RGZ) (P_1 RGZ) (\downarrow^2 C_1 RGX \\
&\quad C_1 RGZ \downarrow^1 P_1 CTC \uparrow^3 \downarrow^1) K; \\
MAD &= H(P_0 CTC) (\downarrow^4 C_{n+1} \uparrow^1 B_1 RGX \downarrow^1 X_0 \uparrow^2 B_2 RGY \downarrow^1 P_1 SM n \uparrow^3 \downarrow^2 B_1 RGY) \\
&\quad (\downarrow^3 P_1 RGX \downarrow^1 D_1 RGZ) (P_1 CTC \downarrow^1 C_2 RGX \downarrow^2 C_2 RGZ \uparrow^4 \downarrow^1) K.
\end{aligned}$$

Первый из приведенных микроалгоритмов соответствует сложению—вычитанию в прямых кодах с использованием обратного кода для вычитания; второй микроалгоритм — умножению с младших разрядов множителя со сдвигом суммы частичных произведений вправо и пропуском такта сложения, когда очередная цифра множителя равна нулю; третий микроалгоритм — делению без восстановления остатка с неподвижным делителем. Заметим, что при выполнении последней операции в качестве регистра делимого (остатка) RGX целесообразно использовать тот регистр, который в предыдущих двух операциях использовался для хранения суммы, т. е. RGZ . Определим набор узлов, необходимых для реализации этих операций. Для этого оставим табл. 100, строки которой будем отмечать символами реализуемых операций (C , U , D), а столбцы — идентификаторами операционных узлов. В клетки такой таблицы впишем длину соответствующих узлов. Заполним таблицу, начиная с самой сложной операции — деления.

Таблица 100

Операции	Операционные узлы				
	CTC	RGX (RGZ для деления)	RGY	RGZ (RGX для деления)	SM
D	$\lceil \log_2(n+1) \rceil$	$n+1$	n	$n+2$	$n+2$
U	$\lceil \log_2 n \rceil$	n	n	$n+1$	$n+1$
C	—	$n+1$	$n+1$	$n+1$	$n+2$

Из табл. 100 следует, что для объединенного операционного блока требуется RGX длиной в $n+1$ разрядов, RGY длиной в $n+1$ разрядов, RGZ длиной в $n+2$ разрядов SM на $n+2$ разрядов и CTC на $\lceil \log_2(n+1) \rceil$ разрядов. При делении CTC должен иметь модуль $n+1$, а при умножении — n (в выполнении сложения — вычитания CTC не участвует). Для ввода и вывода информации в проектируемом блоке при сложении—вычитании и умножении необходимо, выполнить CTC микрооперации $P_1 RGX$, $P_1 RGY$, $P_0 RGZ$, $B_1 RGZ$, $B_2 RGZ$, а также микрооперации $P_1 RGZ$, $P_1 RGY$, $P_0 RGX$, $B_1 RGX$ — при делении. Таким образом, «скелет» объединенного операционного блока будет представлен схемой на рис. 5.34, а. Объединение микроопераций также удобно производить с помощью таблицы, строки которой обозначены идентификаторами операционных узлов, столбцы соответ-

ствуют отдельным микрооперациям из всего их множества, реализуемого операционным блоком, а клетки таблицы отмечают каким-либо символом (например, $\times$), указывающим операционный узел, выполняющий данную микрооперацию. Для рассматриваемого примера таким способом можно построить табл. 101. Следовательно, структура объединенного операционного блока (с учетом информационных связей между узлами) будет иметь вид, показанный на рис. 5.34, б.

Четвертый этап проектирования операционных блоков, состоящий в объединении микроалгоритмов, необходим для построения

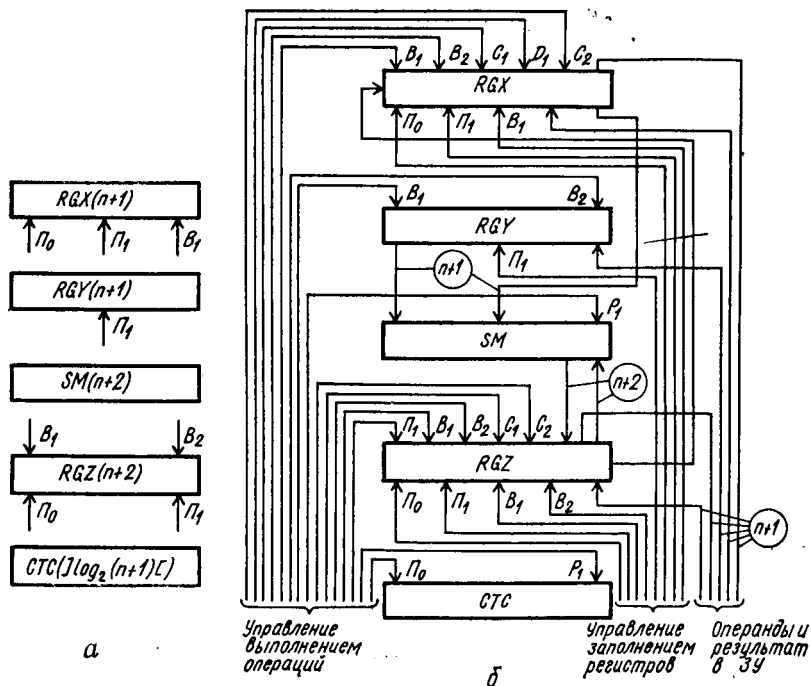


Рис. 5.34

блока управления операциями. Этот этап удобно выполнять, пользуясь представлением микроалгоритмов в графической форме. Простейший способ объединения микроалгоритмов заключается во введении в граф объединенного микроалгоритма после начальной вершины переключательной вида $G_1 \uparrow^1 G_2 \uparrow^2 G_3 \uparrow^3 \dots$, где G_1, G_2, G_3 — графы отдельных микроалгоритмов. Конечная вершина объединенного графа является общей для всех $G_1, G_2, G_3, \dots$. Однако при этом не учитывается то, что микроалгоритмы могут содержать одинаковые микрооперации и одинаковые их последовательности. Это обуславливает необходимость применения достаточно трудоемких методов минимизации числа состояний управляющего автомата (см. п. п. 2.5, 2.6).

Для уменьшения числа операторных (т. е. соответствующих микрооперациям) вершин в объединенном графе и, следовательно, для уменьшения памяти управляющего автомата можно пользоваться

Узлы	Микрооперации														
	B_1RGX	B_2RGX	B_3RGY	B_4RGY	P_1RGZ (для деления P_1RGX)	B_1RGZ (для деления B_1RGX)	B_2RGZ	P_6CTC	C_1RGX	C_1RGZ	P_1CTC	P_1SMn	(для деления D_1RGZ)	(для деления C_2RGX)	(для деления C_2RGZ)
RGX	×	×	—	—	—	—	—	—	×	—	—	—	×	—	×
RGY	—	—	×	×	—	—	—	—	—	—	—	—	—	—	—
SM	—	—	—	—	—	—	—	—	—	—	—	×	—	—	—
RGZ	—	—	—	—	×	×	×	—	—	×	—	—	—	×	—
CTC	—	—	—	—	—	—	—	×	—	—	×	—	—	—	—

следующим приемом. Если некоторая вершина A_i (включая начальную вершину) является общей для графов микроалгоритмов $G_r, G_{r+1}, G_{r+2}, \dots$, то в объединенном графе после вершины A_i следует ввести переключательную вершину вида $G_r \uparrow G_{r+1} \uparrow G_{r+2} \dots$. Выходы переключательной вершины, соответствующие выполнению условия G_j , необходимо подвести к тем операторным или условным вершинам, которые в графе G_i следуют за вершиной A_i . При этом может случиться так, что с одной и той же вершиной будут связаны несколько выходов переключательной вершины, которые в таком случае могут быть объединены в один выход. На рис. 5.35 показан объединенный граф трех приведенных выше микроалгоритмов (G_1 соответствует графу микроалгоритма сложения—вычитания; G_2 — умножения; G_3 — деления). Следует заметить, что такой способ объединения микроалгоритмов хотя и не увеличивает числа операторных вершин, однако усложняет комбинационную часть управляющего автомата. Кроме того, при большом числе микроалгоритмов такая процедура становится весьма громоздкой. Поэтому обычно все реализуемые на АЛУ операции разбивают на группы близких операций и объединение микроалгоритмов производят внутри таких групп.

В настоящее время отсутствует установившаяся классификация структур АЛУ. В цифровых ЭВМ первого и второго поколений широко использовались малорегистровые АЛУ как с фиксированной, так и с плавающей запятой. Отличительной особенностью таких АЛУ является наличие небольшого (3—5) числа регистров. На примере цифровых ЭВМ «Днепр» и «Киев» рассмотрим типовую структуру АЛУ с фиксированной запятой и накапливающим сумматором.

АЛУ этих машин имеют примерно одинаковую структуру и отличаются в основном только числом разрядов в регистрах и сумматоре

(рис. 5.36). Для цифровой ЭВМ «Киев» $n = 40$, $m = 5$, а для цифровой ЭВМ «Днепр» $n = 25$, $m = 5$.

АЛУ состоит из двух регистров $RG41$ и $RG42$, накапливающего сумматора SM , блока местного управления БМУ и блока обработки

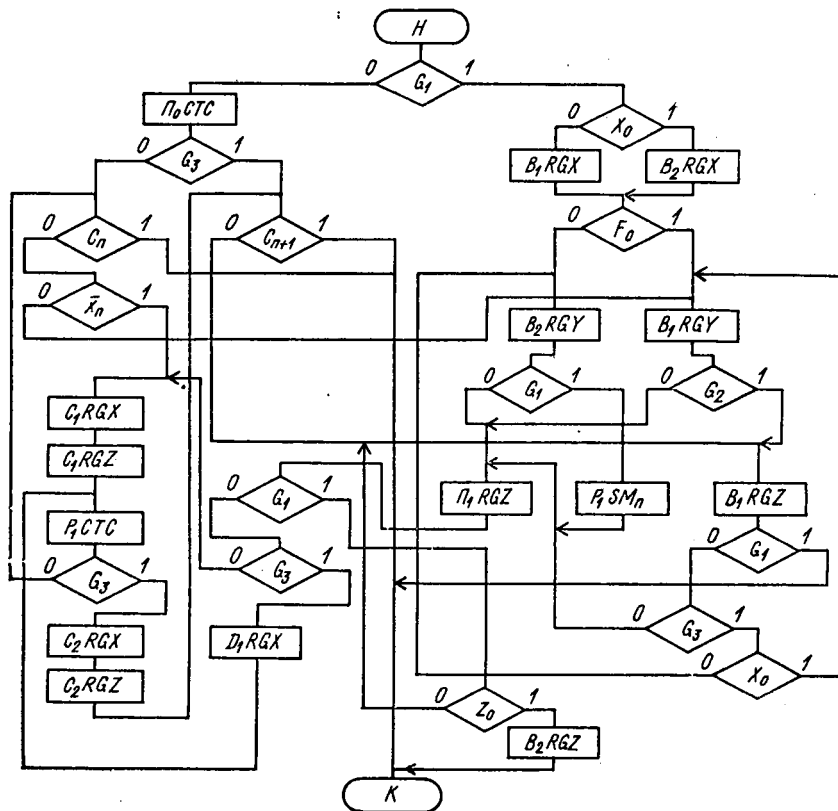


Рис. 5.35

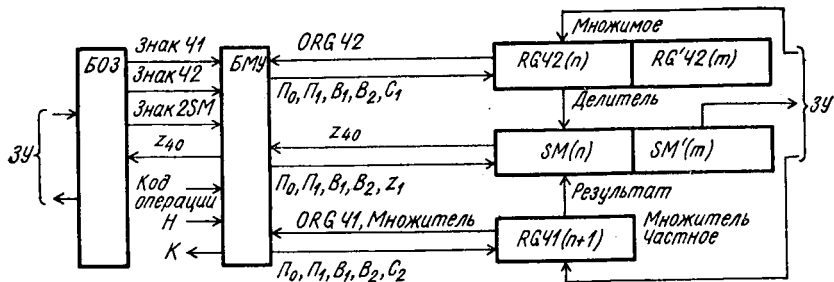


Рис. 5.36

знаков БОЗ. БОЗ имеет триггеры для запоминания знаков первого числа (Ч1) и второго числа (Ч2) и двух знаков результата. БМУ формирует сигналы Π_0 , Π_1 , B_1 , B_2 , C_1 , C_2 , а также сигнал циклического

переноса в младший разряд сумматора z_1 . Из регистров поступают сигналы нулевого их содержимого ($ORG41$ и $ORG42$) и цифры множителя (из $RG41$). Из сумматора в БМУ и далее в БОЗ поступает сигнал z_{40} переноса из старшего разряда сумматора.

Рассмотрим особенности выполнения отдельных операций. Операции начинаются после приема операндов на регистры $RG41$ и $RG42$. При выполнении умножения множитель записывается в $RG41$, а множимое — в $RG42$. Конец умножения определяется по сигналу $ORG41$ или $ORG42$. При делении делитель заносится в $RG42$, а делимое — в $RG41$. Перед началом деления делимое пересылается прямым кодом в сумматор, а на регистр $RG41$ поступает сигнал P_0 . Для определения конца деления используется регистр $RG41$, в младший разряд которого после сброса в нуль заносится единица. Деление прекращается, когда после очередного сдвига в БМУ из $RG41$ поступит указанная единица. В этот же регистр записываются цифры частного.

Результат любой операции через сумматор поступает в ЗУ машины. Округление результата осуществляется путем подачи единицы в старший разряд дополнительного сумматора с последующим отбрасыванием дополнительных разрядов.

Рассмотрим теперь структуру АЛУ с накапливающим сумматором и плавающей запятой. АЛУ

такого типа использовалось в машинах серии «БЭСМ», «Урал», «Раздан», «М-20» и других (рис. 5.37). АЛУ состоит из БМУ, АЛУ порядков и АЛУ мантисс. АЛУ порядков и мантисс в свою очередь состоят из двух регистров и сумматора (соответственно $RGПЧ1$, $RGПЧ2$, $СМП$, $RGЧ1$, $RGЧ2$ и $СММ$).

Числа из ЗУ поступают по шинам мантиссы и порядка и заносятся в один из регистров АЛУ. Результат операции выдается только из сумматора мантиссы и порядка по шинам результата. Если результат операции находится в одном из регистров, то сначала происходит пересылка результата на сумматор. Так как в большинстве случаев результат операции находится на сумматоре, то указанная пересылка осуществляется редко и время, затрачиваемое на нее, незначительно. В то же время отсутствие связи регистров с ЗУ позволяет сэкономить $2n$ элементов И и $2n$ усилителей, необходимых для пересылки результата из регистров АЛУ в ЗУ.

Из устройства управления (УУ) машины в БМУ поступают код операции и сигнал начала операции. Из БМУ в УУ поступают сигналы конца операций и некоторые признаки (например, переполнение разрядной сетки). АЛУ мантисс работает, как правило, с обратными кодами, а АЛУ порядков — с дополнительными кодами. Это связано с тем, что обычно в ЗУ мантиссы хранятся в прямых, а порядки — в дополнительных кодах. Поэтому использование в АЛУ дополнительных кодов мантисс приводило бы к увеличению времени суммирования мантисс.

В цифровой ЭВМ «Минск-32» предусмотрен режим работы как с фиксированной, так и с плавающей запятой. В этом случае АЛУ

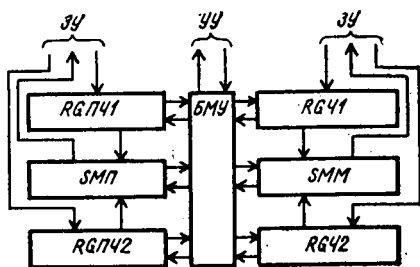


Рис. 5.37

мантисс и АЛУ порядков объединяются в одно АЛУ. В режиме с фиксированной запятой все разряды регистров и сумматора отведены для представления и обработки мантисс чисел. В случае же режима с плавающей запятой регистры и сумматор условно разбиваются на две части. Старшая большая группа разрядов (например, длиной в 30 бит) отведена для представления и обработки мантисс, а младшая меньшая (например, длиной в 7 разрядов) — для представления и обработки порядков.

С целью реализации операций десятичной арифметики АЛУ цифровой ЭВМ «Минск-32» содержит 9 микросумматоров — тетрадь из двоичных одноразрядных сумматоров. Это разбиение на 9 микросумматоров используется и для организации группового переноса при выполнении двоичных операций. Каждый микросумматор имеет цепи

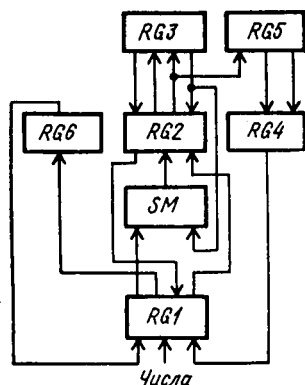


Рис. 5.38

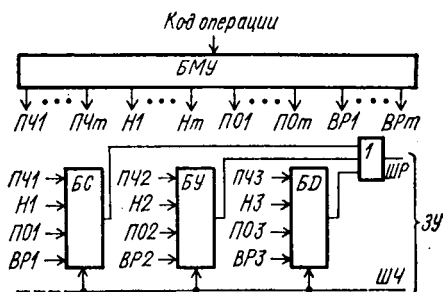


Рис. 5.39

для коррекции результата (используется код с весами 8, 4, 2, 1). В зависимости от режима работы микросумматоры коммутируются либо в 36-разрядный двоичный сумматор, либо в 9-разрядный десятичный сумматор.

Наличие накапливающего сумматора характерно для цифровых ЭВМ, в которых использована импульсно-потенциальная элементная база. В машинах с потенциальной элементной базой применяют, как правило, комбинационный сумматор. В качестве примера малорегистровой АЛУ с таким сумматором рассмотрим АЛУ цифровой ЭВМ «Днепр-21», содержащее шесть статических регистров (рис. 5.38). Регистры $RG1 \div RG5$ имеют длину в 32 разряда. Старшие разряды регистров $RG1 \div RG3$ являются знаковыми. С помощью $RG1$ производится прием слов из $3У$ и, при необходимости, их инвертирование. На этом же регистре хранится одно из слагаемых, множимое и делитель в процессе выполнения соответствующих операций. Входная логика регистра $RG2$ представляет собой комбинационный сумматор SM . На $RG2$ формируется сумма чисел из $RG1$ и $RG3$. Через $RG2$ также производится выдача результатов операций. Регистр $RG3$ осуществляет хранение второго слагаемого. При выполнении операций умножения и деления на $RG2$ формируются суммы частных произведений и остатки, сдвиг которых осуществляется при помощи $RG3$,

выполняющего в этом случае функции вспомогательного ряда триггеров в двухтактном сдвигающем регистре. Регистры $RG4$ и $RG5$ используются как один сдвигающий регистр, куда помещается множитель и записывается частное. Регистр $RG6$, длиной в девять разрядов, служит для хранения порядка результата при работе АЛУ в режиме с плавающей запятой. При обработке в рассматриваемом АЛУ чисел, разрядность которых превышает 32, используются регистры, получаемые из $RG2$ и $RG4$. Эти четыре регистра могут составить 64-разрядный сдвигающий регистр. Управление переключением регистров и передачами слов осуществляется БМУ.

В последнее время, в связи с все более широким внедрением больших интегральных схем, получили распространение так называемые многорегистровые АЛУ. В таких АЛУ число регистров значительно больше минимально необходимого их количества. В качестве примера многорегистрового АЛУ рассмотрим АЛУ блочного типа, которое состоит из отдельных блоков сложения (БС), умножения (БУ), деления (БД) и других, каждый из которых предназначен для выполнения одной или нескольких операций (рис. 5.39). Такие АЛУ обеспечивают широкие возможности для распараллеливания вычислений по одной программе, либо при выполнении нескольких программ. В блочном АЛУ на отдельные блоки из БМУ поступают сигналы «Прием числа» (ПЧ), «Начало операции» (НО), «Выдача результата» (ВР) и «Признак операции» (ПО). Сигнал ПО поступает только на те блоки, которые могут выполнять несколько операций. На БМУ из блоков АЛУ поступают сигналы К. АЛУ связано с ЗУ шинами чисел (ШЧ) и шинами результата (ШР).

Из-за больших аппаратурных затрат АЛУ такого типа не нашли широкого применения в универсальных цифровых ЭВМ 1-го и 2-го поколений. Область их использования ограничивалась специализированными устройствами, где требовалась большая скорость вычислений при малом наборе операций. Примером цифровой ЭВМ 3-го поколения, в которой использовано такое АЛУ, является СДС-6600. Времена выполнения операций в блоках сложения—вычитания, умножения и деления этой цифровой ЭВМ равны соответственно 0,4 мкс, 1 мкс и 2,9 мкс. При вычислении, например, выражения

$$f = \frac{a+b}{c} (a^2 + b^2 + c^2) = \frac{a+b}{c} d$$

без совмещения выполнения операций на блоках АЛУ потребовалось бы три суммирования, одно деление и четыре умножения, на что было бы затрачено 8,1 мкс. При совмещении операций в блочном АЛУ (рис. 5.40) для этого потребуется 4,4 мкс, что соответствует почти двукратному повышению производительности АЛУ. При полном совмещении работы m блоков производительность блочного АЛУ возра-

стает еще больше — в $\frac{\sum_{i=1}^m t_i}{\sum_{i=1}^m t_i \gamma_i} \leq m$ раз.

Таким образом, несмотря на значительные аппаратурные затраты АЛУ блочного типа характеризуются рядом важных преимуществ:

1) блочное построение АЛУ позволяет унифицировать отдельные блоки АЛУ, оптимизировать эти блоки, упростить методику и сократить сроки проектирования АЛУ и т. п.;

2) появляется возможность в широких пределах изменить состав операций, выполняемых АЛУ, без изменения устройства управления цифровой ЭВМ;

3) имеется возможность повышать быстродействие АЛУ не только за счет оптимизации отдельных блоков, но и за счет широкого совмещения операций (АЛУ может одновременно выполнять столько операций, сколько блоков имеется в его составе).

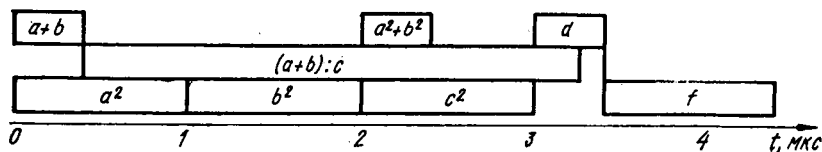


Рис. 5.40

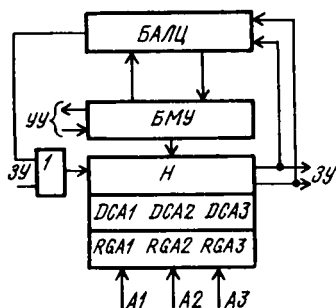


Рис. 5.41

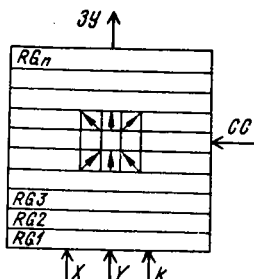


Рис. 5.42

Рассмотренные выше структуры малорегистровых и многорегистровых АЛУ относятся к классу АЛУ с закрепленными микрооперациями. Этот класс АЛУ характерен тем, что в регистрах АЛУ для реализации каждой микрооперации имеется специальная логическая схема, а АЛУ в целом имеет большое число идентичных схем, связанных с различными регистрами. При большом числе выполняемых операций (особенно в многорегистровых АЛУ) количество оборудования в АЛУ становится очень большим. В связи с этим в малых и средних цифровых ЭВМ для реализации АЛУ более рационально использовать принцип обобществления микроопераций. В таких АЛУ все регистры объединены в накопитель типа СОЗУ, а все арифметические и логические цепи регистров вынесены из накопителя и объединены в блок арифметических и логических цепей (БАЛЦ). В качестве операционных могут использоваться любые три регистра накопителя, выбор которых и подключение к числовым магистралям (каналам связи) 1, 2 и 3 (рис. 5.41) осуществляется путем задания адресов $A1$, $A2$ и $A3$ в регистрах $RGA2$, $RGA3$ и $RGA1$ и последующей их дешифрации дешифраторами $ДС1$, $ДС2$ и $ДС3$. Вследствие этого удастся значительно сократить количество всевозможных пересылок слов между регистрами

как по сравнению с АЛУ блочного типа, так и по сравнению с системой малорегистровое АЛУ — СОЗУ. Это способствует повышению быстродействия, так как пересылка по времени равноценная обращению к СОЗУ.

АЛУ подобной структуры получили название магистральных или АЛУ с сосредоточенной памятью и логикой. В магистральных АЛУ при большом числе операций удается несколько сократить аппаратные затраты по сравнению с рассмотренными выше типами АЛУ за счет исключения цепей одинакового назначения в различных регистрах. Такое исключение имеет и отрицательные последствия, которые прежде всего проявляются в невозможности совмещения во времени выполнения отдельных микроопераций. Однако указанные преимущества, гибкость и регулярность структуры таких АЛУ послужили причиной их широкого распространения (по такой блок-схеме построены АЛУ в ЕС ЭВМ).

Перспективной структурой АЛУ, ориентированной на использование больших интегральных схем, является АЛУ с так называемой конвейерной обработкой информации (рис. 5.42). АЛУ такого типа можно представить в виде двумерной матричной структуры размерностью $m \times n$, где m — число строк, равное количеству тактов, требуемых для полной или частичной обработки информации на АЛУ; n — число элементарных машин в строке, равное числу разрядов обрабатываемых слов. Под элементарной машиной понимают функциональный блок с внутренним управлением и внутренней памятью для обработки на каждом такте одного разряда n -разрядных слов. Сигналы от каждой элементарной машины i -й строки могут передаваться только в смежные элементарные машины $i + 1$ -й строки, т. е. количество возможных каналов распространения сигналов от каждой элементарной машины равно трем. Поэтому передача информации носит однонаправленный характер со сдвигом или без него. Включение того или иного канала связи при преобразовании информации или ее передаче обеспечивается внутренним управлением элементарной машины. Это управление осуществляется посредством набора триггеров, каждый из которых управляет комбинационной схемой для реализации элементарного одноктактного преобразования информации, соответствующего выполняемой операции. Таким образом, элементарная машина, находящаяся на пересечении i -й строки и j -го столбца матрицы, включает в себя некоторое множество триггеров, предназначенных для хранения и преобразования информации, и некоторое множество триггеров, управляющих преобразованием информации. Триггеры для хранения и преобразования информации любой i -й строки естественным образом могут быть объединены в регистры.

Информация (операнды и код k операции) может приниматься извне только на регистры нижней строки матрицы, т. е. строки, имеющей номер $i = 1$. Под воздействием синхронизирующих сигналов СС эта информация передается последовательно с i -й строки на $i + 1$ -ю строку. На каждом шаге передачи выполняются одноктактные элементарные преобразования в соответствии с алгоритмом преобразования так, чтобы через m тактов была выполнена заданная операция. К этому времени результат окажется на регистрах m -й строки,

откуда он может быть передан вновь на регистры первой строки или на внешние по отношению к АЛУ устройства.

Поскольку в такой структуре происходит одноплатная передача информации, то если в некоторый момент времени в регистрах i -й строки находится информация, относящаяся к данной задаче, то в регистрах $i - 1$ -й строки и $(i + 1)$ -й строки может находиться информация, относящаяся к другим задачам или к той же задаче, но не зависящая от содержимого i -й строки.

Таким образом, хотя любая операция из множества операций, реализуемых на АЛУ, выполняется за m и более тактов, но одновременно на АЛУ может выполняться до m операций, относящихся к различным, независимым друг от друга программам или частям одной программы. Поэтому производительность такого АЛУ при полной его загрузке будет весьма высокой.

Структурной разновидностью АЛУ являются также табличные АЛУ, особенности построения которых рассмотрены в п. 4.9. В настоящее время сфера применения табличных АЛУ быстро расширяется как в направлении универсальных цифровых ЭВМ, так и в направлении проблемно-ориентированных цифровых ЭВМ.

5.13. РАСЧЕТ ОСНОВНЫХ ПАРАМЕТРОВ АЛУ

В процессе проектирования АЛУ кроме определения их структурной организации необходимо решить еще целый ряд вопросов, а именно: сформировать список операций, реализуемых в АЛУ; рассчитать разрядную сетку АЛУ; выбрать алгоритмы выполнения операций; обосновать форму представления чисел; определить способ задания отрицательных чисел; выбрать систему счисления и положение запятой.

Рассмотрим факторы, влияющие на принятие решений по перечисленным вопросам. Обычно при проектировании АЛУ задана система команд разрабатываемой цифровой ЭВМ. Однако не все команды целесообразно реализовать аппаратным способом в АЛУ.

Ряд команд может быть реализован программно, например, по стандартной подпрограмме, записанной в ПЗУ. Решение данного вопроса так же, как и любого другого вопроса проектирования, должно опираться на технико-экономический анализ. Основная идея технико-экономического анализа в наиболее общем виде состоит в достижении заданной цели при наименьших затратах. Конкретные интерпретации этой идеи во многом зависят от поставленной при проектировании цели и нашли свое отражение в большом числе различных критериев эффективности. Наиболее известными такими критериями являются введенный ранее коэффициент эффективности, а также эффективное быстроедействие V цифровой ЭВМ и его цена q :

$$V = \frac{N(T)}{T}; \quad q = \frac{C(T)}{N(T)},$$

где T — технический ресурс цифровой ЭВМ (промежуток времени от начала эксплуатации цифровой ЭВМ до ее разрушения или другого предельного состояния, например до списания); $N(T)$ — количество

операций, выполненное за время T ; $C(T)$ — суммарные затраты на изготовление и эксплуатацию цифровой ЭВМ.

Обычно технический ресурс для цифровой ЭВМ равен 10 годам. Для цифровой ЭВМ «Киев» $C(T) = 1,2$ млн. руб. при $T = 10$ лет, а $q = 0,0004$ коп/опер., или 4 руб. за миллион операций.

Цена эффективного быстрогодействия является несомненно, более полным технико-экономическим показателем качества цифровой ЭВМ, однако трудности, связанные с получением зависимостей $C(T)$ и $N(T)$, часто не позволяют использовать этот критерий на этапе проектирования. Поэтому здесь и далее будем пользоваться уже известным критерием

$$\varepsilon = \frac{1}{Ct}.$$

Пусть известна система команд, вероятности γ_i их появления в программах и некоторое основное оборудование АЛУ. Аппаратная реализация i -й команды требует введения дополнительного оборудования, стоимость которого $\Delta C'_i$. Например, аппаратная реализация операции умножения по четвертому способу требует введения цепей сдвига в двух регистрах ($2a_0n$) и некоторого количества логических и запоминающих элементов (A)

$$\Delta C'_i = 2a_0n + A.$$

При этом емкость ПЗУ или ОЗУ может быть несколько уменьшена, так как отпадает необходимость в хранении программы i -й команды. Снижение емкости приведет к уменьшению стоимости памяти на $\Delta C''_i$. Следовательно, аппаратная реализация i -й команды требует увеличения стоимости аппаратуры на величину

$$\Delta C_i = \Delta C'_i - \Delta C''_i.$$

Экономия времени выполнения i -й команды при этом составит

$$\Delta t_i = (t'_i - t_i) \gamma_i,$$

где t'_i и t_i — времена выполнения i -й команды до и после введения аппаратных затрат $\Delta C'_i$.

Введение аппаратной реализации i -й команды будет целесообразным, если это приведет к увеличению коэффициента ε , т. е. когда

$$\Delta \varepsilon = \frac{\partial \varepsilon}{\partial C} \Delta C_i + \frac{\partial \varepsilon}{\partial t} \Delta t_i > 0;$$

$$-\frac{\Delta C_i}{iC^2} - \frac{\Delta t_i}{i^2C} > 0; \quad -\frac{\Delta t_i}{t} > \frac{\Delta C_i}{C},$$

или окончательно

$$\left| \frac{\Delta t_i}{t} \right| > \left| \frac{\Delta C_i}{C} \right|. \quad (5.23)$$

Следовательно, если относительное увеличение аппаратных затрат меньше относительного уменьшения среднего времени выполнения одной команды, то схемная реализация i -й команды целесообразна. При этом в выражении (5.23) параметры C_i и t_i относятся ко всей

цифровой ЭВМ (C_i — общая стоимость цифровой ЭВМ, а t_i — время выполнения одной команды с учетом обращения к памяти). Таким образом можно обосновать целесообразность аппаратной реализации всех заданных операций, то есть сформировать список операций, выполняемых в АЛУ.

Расчет разрядной сетки АЛУ состоит в определении количества разрядов, отводимых для представления чисел. Рассмотрим сначала случай, когда числа представлены в форме с фиксированной запятой.

Предположим, что известна точность, с которой необходимо получить результат, т. е. известно значение

$$\delta = \max \left| \frac{\Delta X}{X'} \right| = \max \left| \frac{X - X'}{X'} \right|,$$

где X' — точное значение результата; X — вычисленное значение.

Если для представления числа X используется n_0 разрядов, то погрешность в представлении этого числа, равная половине веса младшего разряда, должна быть меньше заданной точности

$$\frac{1}{2} k^{-n_0} \leq \delta,$$

где k — основание системы счисления. Отсюда следует

$$n_0 \geq \log_k \frac{1}{2\delta}.$$

Так как n_0 — целое, то окончательно имеем

$$n_0 = \left\lceil \log_k \frac{1}{2\delta} \right\rceil.$$

В процессе вычислений возникают погрешности (например, от округления). Поэтому для получения n_0 точных разрядов необходимо в промежуточных вычислениях удерживать дополнительно n_1 разрядов. Для оценки n_1 будем считать, что ошибки $x_1, x_2, \dots, x_m$, возникающие при выполнении m отдельных операций, являются независимыми случайными величинами, равномерно распределенными в интервале $[-\frac{1}{2}, \frac{1}{2}]$. Здесь и далее вычисления будем производить в единицах младшего дополнительного разряда. Будем также считать, что ошибка выполнения последовательности из m операций равна сумме отдельных ошибок. В этом случае, согласно центрально предельной теореме теории вероятностей, при $m \rightarrow \infty$ распределение суммарной ошибки стремится к нормальному. Практически уже при $m > 5$ суммарное распределение можно считать нормальным. Тогда среднее квадратичное отклонение суммарной ошибки x будет равно (в единицах младшего разряда)

$$\begin{aligned} \sigma &= \sqrt{\int_{-0,5}^{0,5} \int_{-0,5}^{0,5} \dots \int_{-0,5}^{0,5} (x_1^2 + x_2^2 + \dots + x_m^2) dx_1 dx_2 \dots dx_m} = \\ &= \sqrt{m \int_{-0,5}^{0,5} x_1^2 dx_1} = \sqrt{\frac{m}{12}}. \end{aligned}$$

Из теории вероятностей для центрированного нормального распределения случайной величины x (т. е. такого распределения x , когда математическое ожидание x равно нулю) известны следующие соотношения:

$$\begin{aligned}P(-\sigma < x < \sigma) &= 0,68; \\P(-2\sigma < x < 2\sigma) &= 0,95; \\P(-3\sigma < x < 3\sigma) &= 0,997.\end{aligned}$$

Таким образом, с вероятностью 0,95 распределение случайной величины x укладывается на участке 2σ в одну и в другую сторону от центра распределения. Следовательно, для того, чтобы суммарная ошибка с вероятностью 0,95 не превышала половины веса младшего разряда, необходимо выполнение условия

$$2\sigma \leq \frac{1}{2} k^{n_1}.$$

Учитывая, что $2\sigma = \sqrt{\frac{m}{3}}$, из этого условия получим

$$n_1 \geq \frac{1}{2} \log_k \frac{4m}{3} \approx \frac{1}{2} \log_k m$$

или

$$n_1 = \left\lceil \frac{1}{2} \log_k m \right\rceil.$$

Так как в машинах с фиксированной запятой разрядная сетка используется полностью только для чисел близких 1, а числа меньше $\frac{1}{2}$, $\frac{1}{4}$ и так далее используют разрядную сетку не полностью, то это приводит к потере значащих разрядов. Для компенсации указанной потери точности необходимо выделить дополнительно n_2 разрядов. Оценим величину n_2 . Пусть наибольшее и наименьшее (не равные нулю) числа, с которыми оперирует ЭВМ, будут равны $X_{\max}$ и $X_{\min}$. Тогда числу $X_{\max}$ в ЭВМ будут равны слова $0,11 \dots 11$, а число $X_{\min}$ можно записать в виде $\frac{X_{\min}}{1} \approx \frac{X_{\min}}{X_{\max}} = 0,00 \dots 01$. Следовательно, n_2 должно удовлетворять условию

$$k^{-n_2} \leq \frac{X_{\min}}{X_{\max}}.$$

Отсюда получим

$$n_2 = \left\lceil \log_k \frac{X_{\max}}{X_{\min}} \right\rceil.$$

Таким образом, для получения n точных разрядов необходимо, чтобы мантисса имела длину

$$n = \left\lceil \log_k \frac{1}{2\delta} \left[+ \right] \frac{1}{2} \log_k m \left[+ \right] \log_k \frac{X_{\max}}{X_{\min}} \right\rceil.$$

Рассмотрим теперь случай, когда числа представлены в форме с плавающей запятой. Для представления мантиссы необходимо

$n_0 + n_1$ разрядов, так как в этом случае все числа полностью используют разрядную сетку цифровой ЭВМ. Число разрядов n_3 для представления порядка выбирается таким, чтобы диапазон представляемых в цифровой ЭВМ чисел был больше диапазона чисел данной задачи:

$$\frac{1}{k^{-(k^{n_3}-1)}} \leq X_{\min}; \quad k^{(k^{n_3}-1)} \geq X_{\max}.$$

Пусть $N = \max \left(X_{\max}, \frac{1}{X_{\min}} \right)$,

тогда

$$N \leq k^{k^{n_3}-1},$$

где $k^{n_3} - 1$ — максимальный порядок числа. Поэтому

$$n_3 \geq \log_k (\log_k N + 1) \approx \log_k \log_k N$$

или окончательно

$$n_3 = \lceil \log_k \log_k N \rceil.$$

Стоимость и быстродействие АЛУ во многом зависят от правильного выбора алгоритмов выполнения отдельных операций. Выбор алгоритмов должен производиться таким образом, чтобы для аппаратной реализации операций максимально использовалось общее оборудование АЛУ. В конечном счете этот выбор сводится к перебору многих вариантов, который может быть выполнен по следующей схеме. Пусть

$$A_{i_1}, A_{i_2}, \dots, A_{i_{n_i}} \quad (i = 1, 2, \dots, m)$$

будут различные варианты алгоритмов выполнения i -й операции и их схемных реализаций. Тогда каждому набору

$$A_{1j_1}, A_{2j_2}, \dots, A_{mj_m}$$

будет соответствовать стоимость $C = C(j_1, j_2, \dots, j_m)$ и быстродействие $t = t(j_1, j_2, \dots, j_m)$ АЛУ. Имея эти величины для каждого варианта, можно найти произведения

$$\frac{1}{\varepsilon} = (C_0 + C)(\alpha t_0 + t),$$

где C_0 — стоимость остальных устройств цифровой ЭВМ; t_0 — время обращения к памяти; α — среднее число обращений к памяти, приходящееся на одну арифметическую операцию с учетом совмещения операций в АЛУ, ЗУ и УУ.

Далее по величинам $\frac{1}{\varepsilon}$ находят тот вариант набора алгоритмов, который обеспечивает наибольший коэффициент эффективности. При реализации описанного перебора имеются две трудности. Первая связана с необходимостью определить величины C и t для большого числа вариантов. Действительно, даже при $m = 10$ и $n_i = 2$ число вариантов равно 2^{10} . Частично число вариантов можно уменьшить за счет исключения явно нецелесообразных сочетаний алгоритмов, та-

ких как, например, первый метод умножения и второй метод деления. Вторая трудность связана с точным определением величины αt_0 , которая зависит от числа уровней ЗУ, частоты появления операций условного перехода, глубины совмещений операций в ЗУ, УУ и АЛУ и других факторов. При большой глубине совмещения операций $\alpha t_0 \ll t$. Поэтому можно считать $\alpha \approx 0$.

При выборе формы представления чисел обычно исходят из следующих соображений. Цифровая ЭВМ с фиксированной запятой проще по структуре, требует меньше аппаратных затрат и является более быстросействующей. Однако при использовании фиксированной запятой усложняется программирование, так как необходимо анализировать диапазоны изменения чисел с целью выбора масштабных коэффициентов. Исходя из этого фиксированная запятая используется в ЭВМ, ориентированных для решения узкого класса задач с использованием одних и тех же программ с малым диапазоном изменения чисел (например, предназначенных для обработки экономической информации). Плавающая запятая используется в универсальных цифровых ЭВМ, предназначенных для инженерных и научных расчетов, а также цифровых ЭВМ, ориентированных на широкий класс задач с большим диапазоном изменения чисел. В современных цифровых ЭВМ наметилась тенденция использования АЛУ, работающих как с плавающей, так и с фиксированной запятой.

Важным вопросом является также выбор кода для представления отрицательных чисел. Преимущества дополнительного кода, прежде всего, проявляются при реализации операции сложения—вычитания. При этом повышается быстросействие и сокращаются аппаратные затраты. Кроме того, результат операции всегда будет в дополнительном коде и нет необходимости его преобразования в стандартную форму, что особенно важно в регистровых АЛУ, где результат не отсылается в память, а остается в АЛУ. Однако при использовании дополнительных кодов несколько увеличивается время выполнения многотактных операций типа умножения и деления. Необходимы также дополнительные преобразования кодов при вводе и выводе информации. Количественный анализ указанных особенностей на базе статистических данных о классе подлежащих решению задач позволяет в каждом конкретном случае принять оптимальное решение о способе представления отрицательных чисел. Современной же тенденцией в решении этой задачи является использование дополнительных кодов.

При выборе основания k системы счисления для цифровых ЭВМ с фиксированной запятой рассматривают обычно два случая: $k = 10$ и $k = 2$.

Блоки для выполнения операций над десятичными числами в 1,2—1,4 раза являются более сложными и менее быстросействующими, чем соответствующие двоичные блоки. Однако в случае использования двоичной системы счисления необходимо при вводе—выводе информации осуществлять двоично-десятичные преобразования, для чего необходимо время и дополнительное оборудование. Это частично компенсирует указанные недостатки десятичной системы. Окончательное решение по данному вопросу принимается путем анализа количества

двоично-десятичных преобразований в типовых задачах, которые будут решаться на проектируемой цифровой ЭВМ.

Для машин с плавающей запятой представляют интерес системы счисления с основаниями 2^m ($m = 2, 3, \dots$), так как в таких цифровых ЭВМ увеличение основания системы счисления приводит к повышению быстродействия за счет уменьшения числа сдвигов при выравнивании порядков и нормализации результата. Поэтому целесообразно повышать основание системы счисления. Однако с повышением основания уменьшается точность представления чисел из-за неполного использования разрядной сетки. Например, двоичное число $2^7 \cdot 0,110111$ в системе с основанием 8 при том же количестве разрядов будет иметь вид $8^3 \cdot 0,001\ 101$. Частично указанную потерю точности можно компенсировать за счет уменьшения числа разрядов, отводимых для представления порядка, так как в системе с основанием k^m для представления порядка требуется меньше разрядов. Действительно,

$$\begin{aligned} n_3(k^m) &= \lceil \log_{k^m} \log_{k^m} N \rceil = \left\lceil \log_{k^m} \frac{\log_k N}{m} \right\rceil = \\ &= \left\lceil \frac{\log_k \log_k N - \log_k m}{m} \right\rceil. \end{aligned}$$

Но так как для представления одной цифры из системы с основанием k^m в системе с основанием k требуется m k -х разрядов, то

$$n_3(k^m) = m \left\lceil \frac{\log_k \log_k N - \log_k m}{m} \right\rceil = \lceil n_3(k) - \log_k m \rceil,$$

т. е. при $m = k = 2$ для представления порядка требуется на один двоичный разряд меньше, чем в системе с основанием 2. Освободившегося разряда вполне достаточно для компенсации указанной погрешности. Таким образом, в машине с плавающей запятой переход от двоичной системы счисления к системе с основанием 4 позволяет повысить быстродействие и увеличить точность представления чисел, мантиссы которых больше $1/2$. Дальнейшее повышение основания системы повышает быстродействие АЛУ, однако точность представления чисел уменьшается.

Выбор положения запятой представляет особый интерес для цифровой ЭВМ с плавающей запятой. Возможно два варианта, а именно: запятую можно ставить перед старшим разрядом и после него. Положение запятой влияет на вероятность появления ненормализованного результата. Например, если запятая фиксирована перед старшим разрядом мантиссы, то при умножении возможно нарушение нормализации на один разряд вправо. Так как операнды X и Y нормализованы, то $\frac{1}{k} \leq X, Y < 1$. Тогда вероятность P нарушения нормализации будет равна

$$P = W \left\{ XY < \frac{1}{k} \right\} = \int_{\frac{1}{k}}^1 dX \int_{\frac{1}{k}}^{1/kX} q(X) P(Y) dY,$$

где $q(X)$ и $P(Y)$ — плотности распределения мантисс X и Y , а $W\left\{XY < \frac{1}{k}\right\}$ — вероятность события, состоящего в том, что произведение XY меньше $1/k$. При равномерном распределении мантисс имеем

$$W\left\{XY < \frac{1}{k}\right\} = \frac{k \ln k - (k-1)}{(k-1)^2},$$

что при $k = 2$ дает

$$W\left\{XY < \frac{1}{2}\right\} = \frac{2 \ln 2 - 1}{1} = 0,39.$$

Если же запятая стоит после старшего разряда, то

$$1 \leq X, Y < k$$

и

$$W\{YX > k\} = \int_1^k dX \int_{k/X}^k q(X) P(Y) dY = \frac{k(k-1-\ln k)}{(k-1)^2},$$

что при $k = 2$ дает

$$W\{XY > k\} = 0,61.$$

Таким образом, если стремиться к уменьшению нарушения нормализации при умножении, то фиксация запятой перед старшим разрядом в случае равномерного распределения мантисс дает лучшие результаты. Следует однако отметить, что бывают случаи, когда нарушение нормализации допустимо, так как при этом повышается быстродействие. Примером является умножение по первому способу, при котором нормализацию можно осуществлять путем блокировки последнего сдвига, что сокращает время умножения на t_c .

УСТРОЙСТВА УПРАВЛЕНИЯ ЦИФРОВЫМИ ЭВМ И ОРГАНИЗАЦИЯ ПРОЦЕССОРОВ

6.1. СТРУКТУРА И КОДИРОВАНИЕ КОМАНД ЦИФРОВЫХ ЭВМ. ФУНКЦИИ УСТРОЙСТВА УПРАВЛЕНИЯ

Процесс вычислений на ЭВМ сводится к выполнению ряда элементарных действий. Информация о том, какое элементарное действие должно выполняться машиной в данный момент, где взять исходные данные, куда записать результат и к выполнению какого элементарного действия перейти в следующем такте, называется командой. Совокупность команд, последовательное выполнение которых приводит к решению данной задачи, называется программой. Команды кодируются двоичными цифрами и хранятся в ЗУ машины.

Из определения команды следует, что машинное слово, которым представлена команда, можно разбить на две части: операционную и адресную. В операционной части указывается код операции, которую выполняет машина при выполнении данной команды. В адресной части должна содержаться информация об адресах исходных чисел, адресе результата и адресе следующей команды. В зависимости от того, какое количество адресов фактически указывается в адресной части команды, различают четырех-, трех-, двух-, одно- и нульадресные команды. Структура четырехадресной команды приведена на рис. 6.1, а. Здесь k — код операции; $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ — соответственно адреса первого и второго чисел, адрес результата и адрес следующей команды. Команды всегда можно расположить в такой последовательности, что адрес следующей команды будет на 1 больше адреса предыдущей. Следовательно, адрес следующей команды можно формировать с помощью счетчика, в который заносят адрес первой команды и после выполнения очередной команды прибавляют 1. Такой счетчик получил название счетчика адреса, а способ формирования адреса следующей команды с его помощью назван естественным, в отличие от принудительного способа, когда адрес следующей команды задается в адресной части предыдущей. Таким образом, отпадает необходимость фактически указывать в команде адрес следующей команды. Принудительное следование команд не повышает производительности процессоров, но создает некоторые удобства при программировании. Однако для представления команд требуются слова большой длины и, следовательно, дополнительное оборудование. Поэтому четырехадресные команды и принудительный способ формирования их последовательности не получили распространения.

Трехадресная команда (рис. 6.1, б) означает следующее: взять числа из ячеек α_1 и α_2 , выполнить над ними операцию, код которой k , результат записать в ячейку α_3 и перейти к выполнению команды, записанной в следующей по порядку ячейке ЗУ.

В некоторых машинах ячейка α_3 фиксирована, т. е. для всех команд одна и та же. В этом случае ее в явном виде в команде не указывают. Команда превращается в двухадресную (рис. 6.1 а) и означает следующее: взять числа из ячеек α_1 и α_2 , выполнить над ними операцию k , результат записать в ячейку α_3 и перейти к выполнению следующей команды. Бывают и другие типы двухадресных команд. Например, результат можно записать в одну из ячеек α_1 или α_2 .

В одноадресных командах фиксирована также и ячейка α_2 (в качестве ячейки α_2 может выступать ячейка α_3). Команда (рис. 6.1, б) в этом случае означает: взять числа из ячеек α_1 и α_2 , выполнить над ними операцию, код которой k , результат записать в ячейку α_2 и перейти к выполнению следующей команды. В качестве ячейки α_2 используется чаще всего либо накапливающий сумматор, либо один из регистров АЛУ.

При наличии в ЭВМ стекового СОЗУ используются команды переменной длины, а именно: одноадресные и нулеадресные. Одноадресные команды служат только для обмена информацией между стековым СОЗУ и адресным ОЗУ. Все остальные команды содержат только код операции. Операнды при этом извлекаются из СОЗУ, а результат заносится в СОЗУ.

Наличие стекового СОЗУ не является необходимым условием применения команд переменной длины. Длина команды может быть указана специальным признаком в коде операции и изменяться как за счет адресности (трех-, двух- или одноадресные), так и за счет длины отдельных частей команды. Например, выделенные 16 ячеек памяти могут быть пронумерованы четырьмя разрядами и снабжены маркерным разрядом, единица в котором указывает, что следующие за ней четыре разряда являются адресом одной из 16 выделенных ячеек, а ноль — на то, что за маркером расположен полный адрес, т. е. номер ячейки из любой невыделенной части памяти.

Структура команд и прежде всего их адресность оказывает существенное влияние на такие важные характеристики ЭВМ как производительность, экономичность, емкость памяти, сложность программного обеспечения и другие показатели. Для приближенной оценки влияния адресности на указанные характеристики рассмотрим программы вычисления функции

$$F = \frac{A + X(B + CX)}{D}. \quad (6.1)$$

Предположим, что исходные данные A, B, C, D и X записаны в ячейках $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ и α_5 соответственно. Тогда программа для четырехадресной машины будет представлена в табл. 102.

Здесь принято, что цикл памяти равен t_0 , а времена выполнения коротких (типа сложения) и длинных (типа умножения) операций в АЛУ соответственно равны t_1 и t_2 . Программа и данные занимают по пять ячеек памяти или (бит)

$$N_4 = 5(N_K + 4N_A + N_C),$$

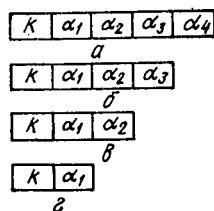


Рис. 6.1

Таблица 102

Адреса команд	Программа					Пояснения	
	Код операции	Адреса				Операция	Время
j_1	$\times$	α_5	α_3	α_3	j_2	CX	$4t_0 + t_2$
j_2	$+$	α_2	α_3	α_3	j_3	$B + CX$	$4t_0 + t_1$
j_3	$\times$	α_5	α_3	α_3	j_4	$X(B + CX)$	$4t_0 + t_2$
j_4	$+$	α_1	α_3	α_3	j_5	$A + X(B + CX)$	$4t_0 + t_1$
j_5	$\div$	α_3	α_4	α_3	j_6	F	$4t_0 + t_2$

где N_K , N_A и N_C — соответственно длина кода операции, адреса и слова. Суммарное время вычислений по рассматриваемой программе составит

$$T_4 = 20t_0 + 2t_1 + 3t_2 + t_3(N_4 + N_c),$$

где t_3 — приведенное к одному биту время ввода (N_4 бит программы и исходных данных) и вывода (N_c бит результата).

Таблица 103

$j+1$	$\times$	α_5	α_3	α_3	CX	$4t_0 + t_2$
$j+2$	$+$	α_2	α_3	α_3	$B + CX$	$4t_0 + t_1$
$j+3$	$\times$	α_5	α_3	α_2	$X(B + CX)$	$4t_0 + t_2$
$j+4$	$+$	α_1	α_3	α_3	$A + X(B + CX)$	$4t_0 + t_1$
$j+5$	$\div$	α_3	α_4	α_3	F	$4t_0 + t_2$

Программа тех же вычислений для трехадресной цифровой ЭВМ представлена в табл. 103. Здесь команды располагаются в ячейках $j+1, j+2, \dots, j+5$, адреса которых формируются с помощью счетчика команд. Соответствующие подсчеты дают

$$N_3 = 5(N_K + 3N_A + N_C);$$

$$T_3 = 20t_0 + 2t_1 + 3t_2 + t_3(N_3 + N_c).$$

Программа для двухадресной ЭВМ представлена в табл. 104. Здесь предполагается, что результат операции остается в регистре P АЛУ, вследствие чего добавляется одна команда ($\rightarrow$) пересылки содержимого этого регистра в ячейку α_3 , по длительности равная короткой операции. Объем памяти и затраты времени для такой программы равны

$$N_2 = 6(N_K + 2N_A) + 5N_C;$$

$$T_2 = 13t_0 + 3t_1 + 3t_2 + t_3(N_2 + N_c).$$

Программа для одноадресной машины представлена в табл. 105. Здесь используются команды, после выполнения которых результат остается в регистре P АЛУ. Вследствие этого первая команда программы — подготовительная и заключается в пересылке содержимого ячейки α_5 в P . Для такой программы

$$N_1 = 7(N_K + N_A) + 5N_C;$$

$$T_1 = 14t_0 + 4t_1 + 3t_2 + t_3(N_1 + N_C).$$

$j+1$	$\times$	α_5	α_3	CX	$3t_0 + t_2$
$j+2$	$+$	α_2	P	$B + CX$	$2t_0 + t_1$
$j+3$	$\times$	α_5	P	$X(B + CX)$	$2t_0 + t_2$
$j+4$	$+$	α_1	P	$A + X(B + CX)$	$2t_0 + t_1$
$j+5$	$\div$	P	α_4	F	$2t_0 + t_2$
$j+6$	$\rightarrow$		α_3	$P \rightarrow \alpha_3$	$2t_0 + t_1$

Программа для цифровой ЭВМ со стековым СОЗУ представлена в табл. 106. По команде $j+1 (\rightarrow)$ происходит пересылка числа из ячейки α_5 в регистр $RG1$ СОЗУ; команда $j+2 (\leftrightarrow)$ осуществляет дублирование информации в $RG1$ и $RG2$ и т. д.; по команде $j+12 (\leftarrow)$ выполняется передача содержимого $RG1$ в ячейку α_3 . В этом случае

Таблица 105

$j+1$	$\rightarrow$	α_5	$\alpha_5 \rightarrow P$	$2t_0 + t_1$
$j+2$	$\times$	α_3	CX	$2t_0 + t_2$
$j+3$	$+$	α_2	$B + CX$	$2t_0 + t_1$
$j+4$	$\times$	α_5	$X(B + CX)$	$2t_0 + t_2$
$j+5$	$+$	α_1	$A + X(B + CX)$	$2t_0 + t_1$
$j+6$	$\div$	α_4	F	$2t_0 + t_2$
$j+7$	$\rightarrow$	α_3	$P \rightarrow \alpha_3$	$2t_0 + t_1$

$$N_0 = 12N_K + 6N_A + 5N_C;$$

$$T_0 = 18t_0 + 9t_1 + 3t_2 + t_3N_0.$$

В табл. 107 и 108 приведены сводные данные по объему памяти и времени выполнения программ в предположении, что $N_A \approx (2 \div 3) N_K$, $N_C \approx (4 \div 5) N_K$; $t_0 \approx t_1$, $t_2 \approx (2 \div 10) t_0$; $t_3 \approx (10^2 - 10^3) \frac{t_0}{N_K}$. Увеличенное число обращений к памяти в трех- и четырехадресных ЭВМ обусловлено записью в ЗУ результатов операций по третьему адресу даже в тех случаях, когда этот результат является операндом для

следующей команды. Однако в трех- и четырехадресных командах легко предусмотреть средства, исключающие такие обращения, например, путем указания в адресной части команд номеров регистров АЛУ, из которых можно брать операнды и в которые можно записывать результаты операций. Если вместо α_3 по третьему адресу команд $(j+1) - (j+4)$, второму адресу команд $(j+2) -$

Таблица 106

Адреса команд	Программа	Регистры СОЗУ			Время	Длина команды
		1	2	3		
$j+1$	$\rightarrow$	α_5	X		$2t_0 + t_1$	$N_k + N_A$
$j+2$	$\leftrightarrow$		X		$t_0 + t_1$	N_k
$j+3$	$\rightarrow$	α_3	C	X	$2t_0 + t_1$	$N_k + N_A$
$j+4$	$\times$		CX		$t_0 + t_2$	N_k
$j+5$	$\rightarrow$	α_2	B	CX	$2t_0 + t_1$	$N_k + N_A$
$j+6$	$+$		$B + CX$	X	$t_0 + t_1$	N_k
$j+7$	$\times$		$X(B + CX)$		$t_0 + t_2$	N_k
$j+8$	$\rightarrow$	α_1	A	$X(B + CX)$	$2t_0 + t_1$	$N_k + N_A$
$j+9$	$+$		$A + X(B + CX)$		$t_0 + t_1$	N_k
$j+10$	$\rightarrow$	α_4	D	$A + X(B + CX)$	$2t_0 + t_1$	$N_k + N_A$
$j+11$	$\div$		F		$t_0 + t_2$	N_k
$j+12$	$\leftarrow$	α_3			$2t_0 + t_1$	$N_k + N_A$

Таблица 107

	N_K	N_A	N_C	$\sum (N_R)$
N_0	12	6	5	44—55
N_1	7	7	5	41—53
N_2	6	12	5	50—77
N_3	5	15	5	55—75
N_4	6	20	5	65—90

Таблица 108

	t_0	t_1	t_2	$\sum (t_0)$	$t_3 N_i (t_0 \times 10^3)$
T_0	18	9	3	33—57	4,4—55
T_1	14	4	3	24—47	4,1—53
T_2	13	3	3	22—46	5,0—77
T_3	20	2	3	28—52	5,5—75
T_3'	12	2	3	20—44	5,5—75
T_4	20	2	3	28—52	6,5—90

— $(j + 4)$ и первому адресу команды $j + 5$ (табл. 103) указать номер P регистра в АЛУ, то число обращений к памяти можно сократить до 12, что и отражено в строке T'_3 табл. 108.

Сравнение трех- и четырехадресных команд показывает, что наличие четвертого адреса не дает выигрыша в быстродействии машины, однако увеличивает объем памяти, занимаемый программой, что, при отсутствии совмещения ввода — вывода с вычислениями, приводит к уменьшению производительности ЭВМ. Производительность уменьшается также и из-за увеличения объема пересылаемой между уровнями памяти информации. Однако эти выводы справедливы только для ЭВМ, имеющих ОЗУ с М-поиском. Если же в качестве ОЗУ используется ЗУ с В-поиском, то четырехадресные команды позволяют существенно повысить быстродействие ЭВМ за счет оптимального рас-

ОТ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Чтение						P			P					P			P			
	j_1	α_5	α_3			j_2	α_2		j_3	α_5'			j_4	α_1		j_5	α_4			
Оп				$\times$				$+$			$\times$				$+$			$\div$		
Запись					P			P				P			P					α_3'
$\Phi A_3 \cup O$	1	6	7			2	8		3	6			4	9		5	10			6
НПЗ	1					2			3				4			5				6
$\Phi A_4 \cup O$	1	2	3			6	7		9	2/10			13	14		16	17			3/20
НП4	1									2										3

Рис. 6.2

положения информации в памяти. Это подтверждается диаграммой работы трех- и четырехадресной ЭВМ (рис. 6.2) при вычислении выражения (6.1), построенной в предположении, что $t_2 = 2t_1$. Все действия ЭВМ, имеющей ЗУ с В-поиском, разбиты на три группы: чтение, операция (On) и запись. В первом относительно такте (ОТ) производится чтение команды по адресу j_1 ; во втором ОТ — чтение числа по адресу α_5 ; в третьем ОТ — по адресу α_3 . В четвертом и пятом ОТ выполняется операция умножения ($\times$), результат которой записывается в регистр P АЛУ. Заканчиваются вычисления в 20 ОТ записью результата в ЗУ по адресу α_3 .

При переводе условных адресов j_i и α_i в фактические необходимо учитывать, что в трехадресной ЭВМ соседние команды должны быть расположены в соседних ячейках. Следовательно, если $j_1 = 1$, то $j_2 = 2$, $j_3 = 3$, $j_4 = 4$ и $j_5 = 5$. Числовую же информацию можно располагать произвольно в свободных ячейках ЗУ. При кодировании адресов, показанном в табл. 109, фактические адреса трехадресной программы (ΦA_3), участвующие в соответствующих операциях, приведены в строке $\Phi A_3 \cup O$ диаграммы на рис. 6.2. В ЭВМ, имеющей ОЗУ с В-поиском, фактические адреса являются номерами тактов работы машины. Следовательно, выборку команды по ΦA_1 можно осуществить только во втором периоде работы ЗУ. Завершаются же вычисления в шестом периоде (строка НПЗ диаграммы) записью результата по адресу α_3 . Указанные ограничения на расположение команд в четырехадресных ЭВМ отсутствуют. Вследствие этого при кодирова-

нии ΦA_4 , показанном в табл. 109, для реализации программы достаточно три периода. Если же при вводе чисел содержимое ячейки α_5 задублировать в ячейке α'_5 и присвоить этим ячейкам фактические адреса 1 и 10, а результат записать по адресу $\alpha'_3 = 20$, то все вычисления можно завершить за один период, что в шесть раз быстрее, чем в трехадресных ЭВМ. Следует однако заметить, что ЗУ с В-поиском в качестве ОЗУ в настоящее время практически не применяются, вследствие чего и четырехадресная система команд не получила распространения.

Таблица 109

	i_1	i_2	i_3	i_4	i_5	α_1	α_2	α_3	α_4	α_5	α'_5	α'_3
ΦA_3	1	2	3	4	5	9	8	7	10	6		
ΦA_4	1	6	9	13	16	14	7	3	17	2	10	20

Сравнение объема памяти, необходимого для решения задачи на нуль-, одно-, двух-, четырехадресных ЭВМ, показывает, что он минимален для одноадресных машин. В общем случае

$$N_i = L_i(N_k + iN_A) + MN_c, \quad (i = 1, 2, 3),$$

где L_i — число команд в i -адресной программе; M — количество исходных данных (в словах). В свою очередь

$$L_i = L_3 \gamma_i \quad (i = 1, 2),$$

где γ_i — коэффициент, характеризующий относительное удлинение программы за счет команд пересылки информации.

Так как любая трехадресная команда эквивалентна трем одноадресным, то $\gamma_i \leq 3$. Величины γ_i могут быть определены из статистически установленного на очень большом числе алгоритмов факта, что в 80 случаях из 100 результат данной операции является операндом для следующей. Отсюда следует, что только 20 % команд являются трехадресными. Если результаты выполнения последовательности двухадресных команд являются операндами для следующих команд, то такие команды превращаются в одноадресные (один из операндов берется из регистра АЛУ и там же остается результат). Поэтому частоты появления двухадресных и одноадресных команд будут равны соответственно $0,2 \cdot 0,8 = 0,16$ и $0,8 \cdot 0,8 = 0,64$. Отсюда получим

$$\begin{aligned} \gamma_2 &= 2 \cdot 0,2 + 1 \cdot 0,8 = 1,2; \\ \gamma_1 &= 3 \cdot 0,2 + 2 \cdot 0,16 + 1 \cdot 0,64 = 1,56. \end{aligned}$$

Следовательно,

$$\begin{aligned} \frac{N_1 - N_2}{L_3} &= \gamma_1(N_k + N_A) - \gamma_2(N_k + 2N_A) - (\gamma_1 - \gamma_2)N_k - \\ &\quad - (2\gamma_2 - \gamma_1)N_A = 0, \quad 36N_k - 0,84N_A < 0; \\ \frac{N_1 - N_3}{L_3} &= (\gamma_1 - \gamma_3)N_k - (3\gamma_3 - \gamma_1)N_A = 0,56N_k - 1,44N_A < 0, \end{aligned}$$

как для существующих ЭВМ $N_k = (0,15 \dots 0,5) N_A$. Знаки последних неравенств не изменяются даже при значительном увеличении. Отсюда следует, что наиболее компактными будут программы для одноадресных ЭВМ. При этом

$$N_1 < N_2 < N_3.$$

Сравнивая по быстродействию ЭВМ с различной адресностью, можно заметить значительное влияние на T_i времени, необходимого для ввода и вывода информации. При большом количестве операций ввода и вывода решающее влияние на быстродействие оказывает N_i следовательно, наибольшим быстродействием будет обладать одноадресная ЭВМ. При малом числе операций ввода—вывода и значительном объеме вычислительных операций наибольшим быстродей-

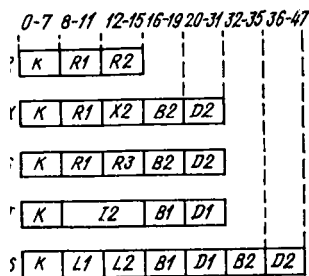


Рис. 6.3

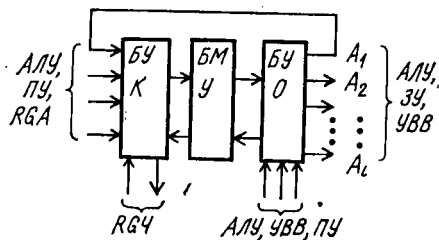


Рис. 6.4

ствием будет обладать трехадресная ЭВМ, в программах которой отсутствуют невычислительные операции пересылок.

Выше при подсчетах T_i предполагалось, что время выборки команды из памяти не зависит от ее длины. Но одноадресная команда более чем в два раза короче трехадресной. Следовательно, за время выборки одной трехадресной команды можно выбрать несколько одноадресных. Учет этого фактора корректирует приведенные выше выводы в сторону малоадресных команд.

Существенным является то, что ни одна из рассмотренных систем команд не обеспечивает для всех классов задач наибольшего быстродействия. Вследствие этого в современных ЭВМ часто используют команды с переменной адресностью, в которых число адресов, указываемых в адресной части, зависит от длины (формата) команды. Например, в ЕС ЭВМ наименьшей адресуемой единицей информации является байт. Длина команд в ЭВМ этой серии может быть равна одному, четырем или шести байтам. При этом два байта называют полусловом. Команды длиной в полуслово не позволяют обращаться к ОЗУ, предназначены для выполнения операций над содержимым регистров ИУ. В командах длиной в два полуслова указывается адрес одного регистра из ОЗУ. В командах из трех полуслов указывается 2 адреса памяти. Форматы команд ЕС ЭВМ представлены на рис. 6.3. Каждый формат команды обозначается своим форматным кодом: RR, RX, SI, SS.

Форматный код *RR* обозначает операцию типа «регистр—регистр» с операндами, находящимися в регистрах АЛУ, адреса которых соответственно *R1* и *R2*. Результат операции при этом записывается в первому адресу *R1*.

Коды *RX* и *RS* обозначают операцию типа «регистр — память» при выполнении которой один из операндов находится в регистре с адресом *R1*, а другой в ОЗУ. При этом в командах *RX* исполнительный адрес формируется как сумма содержимого базового регистра (его адрес *B2*), содержимого индексного регистра (*X2*) и величины смещения *D2*. Команды же типа *RS* означают операции, выполняемые над тремя операндами, причем первый и третий операнды находятся в регистрах *R1* и *R3*, а адрес второго формируется сложением содержимого базового регистра с адресом *B2* и величины смещения *D2* указанной в самой команде.

Команды типа *SI* означают операцию «память — непосредственный операнд». В этом случае адрес первого операнда формируется сложением содержимого регистра базового адреса *B1* и смещения *D1*. Второй же операнд указывается непосредственно в команде в поле *I1*.

При выполнении команд типа *SS* адрес первого операнда формируется так же, как в команде типа *SI*, а адрес второго — так же, как в командах типа *RX* и *RS*. В полях *L1* и *L2* указывают длину соответственно первого и второго операндов.

В каждом формате первый байт полуслова содержит код операции *k*, а формат команды определяется первыми двумя разрядами этого кода: 00 соответствует командам типа *RR*; 01 — командам *RX*; 10 — командам *RS* или *SI*; 11 — командам *SS*. Как известно, универсальные ЭВМ включают в свой состав ЗУ, АЛУ, устройство управления (УУ) и внешние устройства. Рассмотрим взаимодействие перечисленных устройств машины в процессе выполнения какой-либо программы вычислений. Для определенности предположим, что машина трех адресная. С помощью клавиатуры на пульте управления набирается команда ввода и адрес первой команды программы. При нажатии кнопки «Пуск» указанная информация поступает в УУ. Это устройство выдает на устройство ввода необходимую последовательность управляющих сигналов, в результате чего происходит выполнение команды ввода — запись программы и исходных данных в ЗУ. После окончания ввода УУ посылает в ЗУ адрес первой команды. Извлеченная по этому адресу команда поступает в УУ и разбивается на две части: операционную и адресную. Адреса операндов из УУ пересылаются в ЗУ. Выбранные по этим адресам операнды поступают в АЛУ, а из УУ в АЛУ поступает серия управляющих сигналов, необходимых для выполнения заданной операции. После окончания операции результат записывается в ЗУ по адресу, указанному в команде, а УУ формирует адрес следующей команды. Время, в течение которого выполняется одна команда, называют циклом машины. Из изложенного следует, что в течение одного цикла ЭВМ выполняет следующие действия:

- 1) считывает очередную команду из ЗУ;
- 2) по информации, содержащейся в команде, формирует адрес операндов и результаты операции;

3) по сформированным адресам считывает операнды из ЗУ и передает их в АЛУ;

4) выполняет операцию, код которой указан в считанной из ЗУ команде;

5) по сформированному ранее адресу записывает в ЗУ результат операции;

6) формирует адрес следующей команды.

При этом основными функциями устройства управления являются:

1) выборка очередной команды из ЗУ;

2) хранение команды в течение цикла;

3) преобразование адресной части команды (формирование адреса, выборка по ним из ЗУ операндов и запись в ЗУ результата);

4) преобразование операционной части команды (выработка необходимой для выполнения данной операции последовательности управляющих сигналов);

5) формирование адреса следующей команды.

Рассмотрим теперь состав, классификацию и структуру аппаратных средств, реализующих перечисленные функции. Блок, осуществляющий хранение очередной команды, а также формирование адресной части команды называют блоком управления командами (БУК). Необходимыми узлами БУК являются регистр команд *RGK*, служащий для приема команд и хранения их в течение цикла, и счетчик команд *СТК*, формирующий адрес следующей команды. Считывание из ЗУ очередной команды, преобразование адресной и операционной частей команд осуществляется блоком управления операциями (БУО). Для выработки синхронизирующих сигналов и согласования работы БУК и БУО служит блок центрального управления (БЦУ). Предварительно, структура УУ имеет вид, показанный на рис. 6.4. УО вырабатывает сигналы $A_1, A_2, \dots, A_n$, управляющие работой АЛУ, внешних устройств и БУК. На БУО поступают сигналы с АЛУ, внешних устройств, БУК и пульта управления ПУ. Эти сигналы характеризуют состояние соответствующих устройств. БУК связан с регистрами числа (*RGЧ*) и адреса (*RGА*) ЗУ, а также с ПУ АЛУ.

Кроме перечисленных выше основных функций по управлению числительным процессом УУ выполняет также ряд вспомогательных: пуск и останов ЭВМ, контроль работы машины, обеспечение автоматического и потактного режимов работы и др.

Классифицируют УУ ЭВМ по различным признакам. Так, например, различают универсальные и специализированные (с «жесткой программой») УУ. Первые позволяют выполнять любые программы, написанные как последовательность команд, с учетом лишь ограничений, связанных с емкостью памяти машины, длиной разрядной сетки, быстродействием ЭВМ. Вторые работают по фиксированным программам, изменить которые можно лишь путем соответствующих перемutations или перепаек в УУ.

В зависимости от того, постоянна или переменна длительность цикла работы ЭВМ, УУ делят на синхронные и асинхронные. Иногда деление УУ на синхронные и асинхронные производят, ориентируясь на длительность цикла, а на длительность такта микрооперации.

При этом для синхронных УУ частота работы задающего генератора определяется либо по длительности самой длинной микрооперации либо же по длительности наиболее часто встречающейся микрооперации (для «длинных» микроопераций в этом случае требуется несколько тактов). Для синхронных УУ длительность такта зависит от времени выполнения каждой очередной микрооперации.

УУ разделяют также на центральные (централизованные) и смешанные. В центральных УУ все управляющие сигналы, необходимые для выполнения любой из операций, входящих в набор команд машины, вырабатываются непосредственно в УУ. Такие УУ часто являются асинхронными и применяются в одноадресных ЭВМ с наибольшим набором микроопераций. При смешанном управлении УУ выдает только основные управляющие сигналы, обычно являющиеся сигналами расшифровки кодов операций (например: «Сложение», «Вычитание», «Умножение» и т. д.). Эти сигналы поступают на блоки местного управления, которые осуществляют непосредственное управление выполнением команд программы.

6.2. КЛАССИФИКАЦИЯ БЛОКОВ УПРАВЛЕНИЯ ОПЕРАЦИЯМИ. СПОСОБЫ ПОСТРОЕНИЯ СХЕМНЫХ БУО

Как и ранее, последовательность микроопераций, выполнении которых приводит к выполнению команды, назовем микроалгоритмом данной команды. В зависимости от способа реализации микроалгоритмов в БУО различают схемные и микропрограммные БУО. Сущности и особенности микропрограммного управления операциями рассмотрены в п. 6.3. При схемном управлении БУО состоит из отдельных схем, каждая из которых управляет одной операцией. При этом часть узлов может быть общей для всех схем. Существуют также БУО со смешанным управлением, когда одна часть микроопераций реализуется схемным способом, а другая часть — микропрограммным.

При построении БУО по схемному принципу в качестве основных узлов могут использоваться элементы задержки (ЭЗ), логические узлы типа конечных автоматов, распределители сигналов на сдвигающих регистрах или счетчиках с дешифраторами. В соответствии с этим различают БУО на ЭЗ, БУО на основе управляющих конечных автоматов, БУО на распределителях сигналов. Следует указать, что классификацию БУО можно провести и по другим признакам. Однако установившейся классификации БУО к настоящему времени не существует, что часто затрудняет пользование технической литературой.

Исследование различных способов построения БУО будем для определенности производить на примере трехадресной ЭВМ, реализующей аппаратным способом операции сложения, вычитания, умножения и деления в прямых кодах с фиксированной запятой. Обобщенная функциональная схема этой ЭВМ приведена на рис. 6.5. Для упрощения записей здесь использована следующая, менее наглядная по сравнению с рассмотренной в гл. 5, но более краткая система условных обозначений микроопераций (МО) и логических условий (ЛУ):

A_{11} и A_{12} — ЛУ, выполнению которых соответствует наличие 1 в знаковых разрядах регистров $RG1$ и $RG2$ в АЛУ;

A_n — ЛУ «слово в счетчике тактов СТТ меньше n » (n — длина ерандов);

B_{11} и B_{21} — МО выдачи содержимого *RGI* прямым и инверсным дом;

$B_{12} - B_{22}$ — МО выдачи содержимого *RG2* прямым и инверсным дом;

B_{30} — МО передачи адреса со счетчика команд СТК в регистр адреса RGA ЗУ;

B_{31}, B_{32}, B_{33} — МО передачи первого, второго и третьего адресов регистра команд RGK в регистр RGA ЗУ;

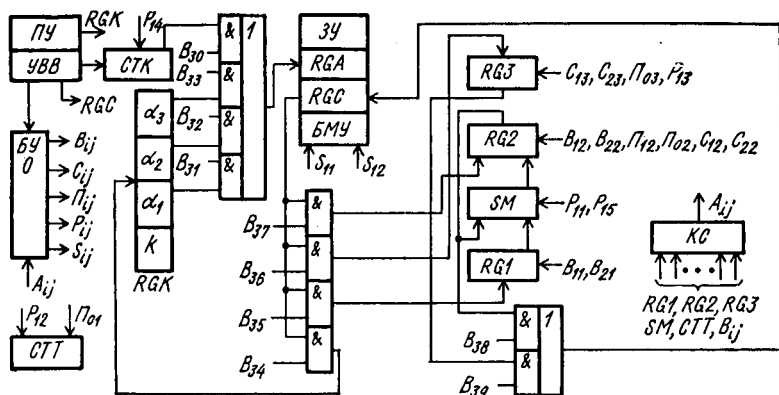


Рис. 6.5

B_{38}, B_{39} — МО передачи слов из $RG2$ и $RG3$ в $RG4$;

$C_{12}, C_{13}, C_{22}, C_{23}$ — МО сдвига на 1 разряд вправо (C_1) и влево (C_2) на регистрах $RG2$ и $RG3$;

 P_{12} — МО приема слов на $RG2$;

P_{01} и P_{02} — МО установки в нуль СТТ и $RG2$;

P_{03} — МО начальной установки $RG3$ перед выполнением деления;

P_{11} и P_{12} — МО прибавления 1 соответственно на сумматоре SM СТТ;

P_{14} — МО прибавления 1 к содержимому СТК;

P_{13} — МО записи 1 в младший разряд $RG3$;

P_{15} — МО замыкания цепи циклического переноса на SM;

S_{11} и S_{12} — МО начала чтения и записи в ЗУ;

H и K — соответственно начало и конец машинного цикла.

С учетом принятых обозначений микроалгоритмы указанных операций будут иметь вид:

$$MAC = H(B_{30}S_{11})(B_{34})(B_{31}S_{11})(B_{35})(B_{32}S_{11})(B_{37})(A_{11}\uparrow^1B_{11}\uparrow^2B_{21}\downarrow^3A_{12}\uparrow^4B_{12}\uparrow^5B_{22}\downarrow^6P_{15})(\Pi_{12})(A_{12}\uparrow^7B_{12}\uparrow^8B_{22}\downarrow^9B_{38}B_{33}P_{14})K;$$

$MAB = H (B_{30}S_{11}) (B_{34}) (B_{31}S_{11}) (B_{35}) (B_{32}S_{11}) (B_{37}) (A_{11} \uparrow {}^1B_{11} \uparrow {}^2 \downarrow {}^1B_{21} \downarrow {}^2A_{12} \uparrow {}^3B_{22} \uparrow {}^4 \downarrow {}^3B_{12} \downarrow {}^4P_{15}) (\Pi_{12}) (A_{11} \uparrow {}^5B_{12} \uparrow {}^6 \downarrow {}^5B_{22} \downarrow {}^6B_{38}S_{12}B_{33}P_{14}) K;$

$MAU = H (B_{30}S_{11}) (B_{34}) (B_{31}S_{11}) (B_{35}) (B_{32}S_{11}) (B_{36}B_{37} \Pi_{02}) (\downarrow {}^2A_{02} \uparrow {}^1B_{11}B_{12}) (\Pi_{12}) (\downarrow {}^1C_{12}P_{12}C_{13}) (A_n \uparrow {}^2) (B_{12}B_{33}B_{38}P_{14}S_{12}) K;$

$MAD = H (B_{30}S_{11}) (B_{34}) (B_{31}S_{11}) (B_{37}) (B_{32}S_{11}) (B_{35}\Pi_{03}) (\downarrow {}^4B_{12}A_{12} \uparrow {}^1B_{21}P_{11} \uparrow {}^2 \downarrow {}^1B_{11}) (\downarrow {}^2\Pi_{12}) (C_{14}C_{15}A_{12} \uparrow {}^3P_{13}) (\downarrow {}^3A_{01} \uparrow {}^4) (B_{39}S_{12}B_{33}P_{14}) K.$

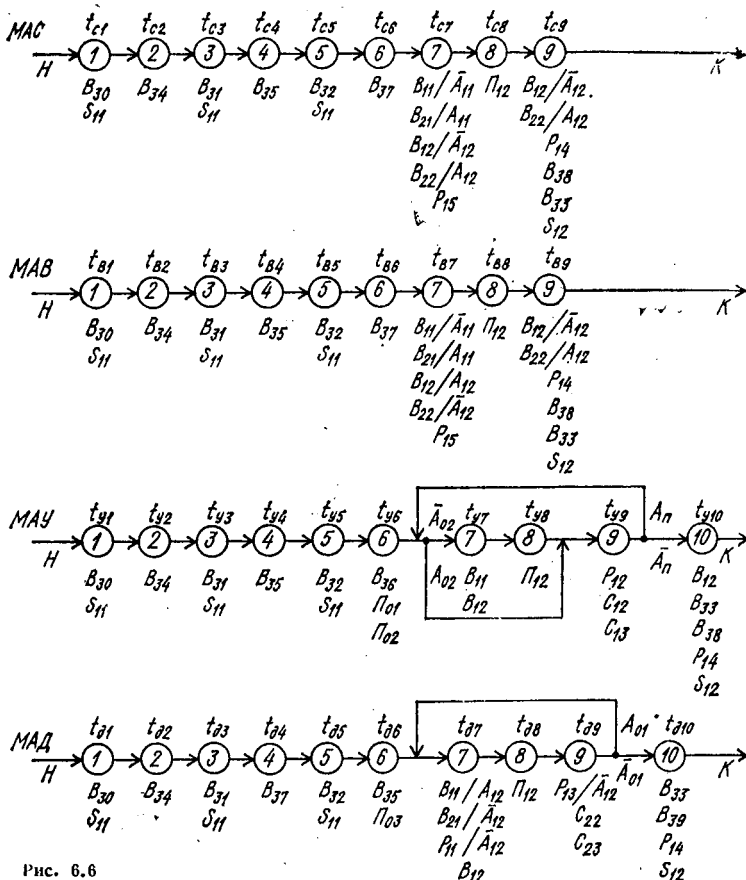


Рис. 6.6

Здесь предполагается, что передачи слов осуществляются парафазными кодами и предварительная установка в нуль регистров не требуется. Скобки, как и в гл. 5, означают, что МО и ЛУ в скобках могут выполняться и проверяться одновременно. Кроме того, проверка ЛУ A_{02} и A_n при переходе по стрелке 2 в МАУ и A_{12} и A_{01} при переходе по стрелке 4 в МАД также может осуществляться одновременно.

Для построения БУО, реализующего приведенные микроалгоритмы, необходимо установить однозначное соответствие между символами

и микроалгоритмов с одной стороны и элементами БУО, связями между ними и сигналами на их выходах с другой стороны. При этом необходимо знать не только последовательность управляющих сигналов, задаваемую последовательностью символов в микроалгоритмах, и продолжительность действия этих сигналов, определяемую временем выполнения соответствующих микроопераций. Всю эту информацию удобно задавать с помощью графов микроалгоритмов (рис. 6.6). Вершины графа соответствуют состояниям, находясь в которых, БУО

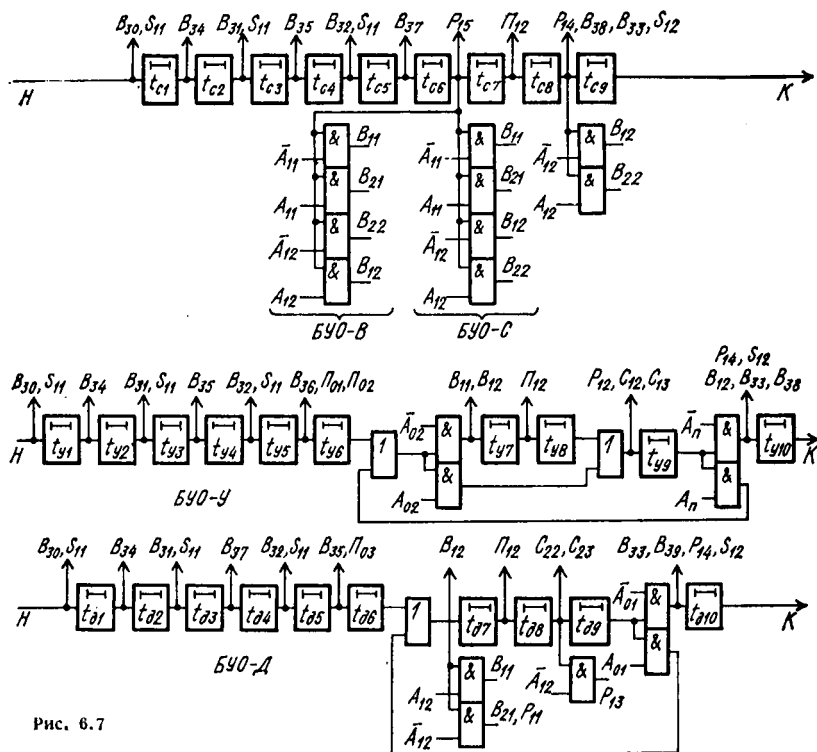


Рис. 6.7

дает сигналы выполнения совместимых МО. Возле вершин указаны имена выполнения соответствующих множеств совместимых МО, символы и условия, при которых они выполняются. Например, при сложении БУО в состоянии 7 выдает сигнал B_{11} при условии $\bar{A}_{11}$.

Непосредственно по графу микроалгоритма можно построить БУО ЭЗ. При этом вершинам графа будут соответствовать элементы задержки, разветвлениям ребер графа — проверка ЛУ элементами И, объединениям ребер — элементы ИЛИ. Структура БУО на ЭЗ показана на рис. 6.7. Поскольку схемы блоков для управления сложением вычитанием отличаются очень мало, то на рис. 6.7 приведена лишь схема первого блока и указаны отличия между ними (БУО—С БУО—В). Рассмотренные примеры реализации БУО позволяют оценить необходимые аппаратные затраты (условную стоимость).

$$C_1 = \alpha h + \beta (k + l),$$

где α — стоимость ЭЗ; β — стоимость одного логического элемента h — количество вершин графа микроалгоритма; k — количество проверяемых ЛУ; l — количество объединяемых ребер на графе микроалгоритма.

По рассмотренным выше схемам можно строить БУО не только на пассивных ЭЗ, но и при использовании в качестве элементов задержки сдвигающих регистров и счетчиков. Период следования синхронизи-

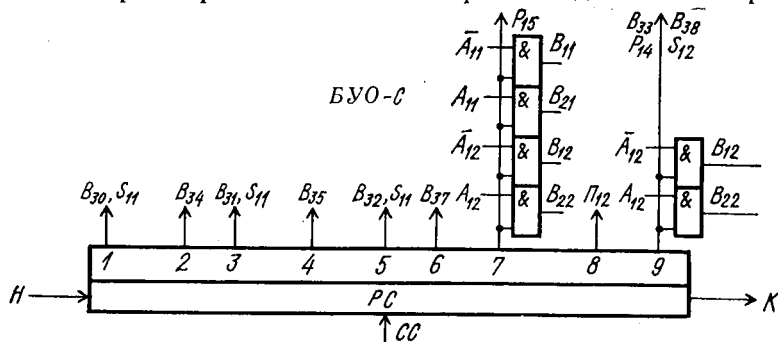


Рис. 6.8

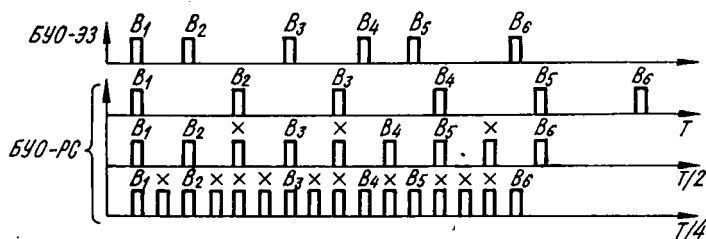


Рис. 6.9

рующих сигналов (СС) для таких элементов задержки определяется как

$$T \leq \min t_i,$$

где t_i — длительность i -й МО.

Удобным методом построения БУО для нециклических микроалгоритмов является использование распределителей сигналов (РС) т. е. узлов, осуществляющих выдачу последовательности сигналов после действия стартового сигнала H (рис. 6.8). Чаще всего РС выполняют на основе сдвигающего регистра, кольцевого счетчика или счетчика с дешифратором. Если РС должен формировать n управляющих сигналов, то счетчик в схеме РС имеет период, равный $n + 1$ причем последнее состояние счетчика запрещает подачу синхронизирующих сигналов СС. Для правильной работы БУО на основе РС период СС должен удовлетворять условию

$$T \geq \max t_i. \quad (6.2)$$

Это снижает быстродействие БУО и делает его синхронным, так как все МО будут выполняться за одно и то же время. Преимуществом

такой схемы БУО является простота переключения ее для реализации других микроалгоритмов. Повысить же быстродействие БУО на РС можно путем увеличения частоты СС в целое число раз и пропуска тактов при выполнении «длинных» МО. На рис. 6.9 приведены временные диаграммы работы БУО на ЭЗ и на РС с периодом синхронизирующих сигналов $T; \frac{T}{4}; \frac{T}{4}$, где знаком X отмечены пропускаемые

СС. Из этого рисунка видно, что при достаточно малом периоде СС быстродействие БУО на РС может быть таким же, как и у БУО на ЭЗ.

Например, при $T = \min t_i$ БУО на РС по существу будет представлять собой описанный выше БУО, в котором в качестве ЭЗ используются регистры или счетчики. Однако увеличение частоты СС приводит к росту периода РС, что равноценно увеличению разрядности регистра или счетчика — дешифратора в РС и влечет за собой увеличение аппаратных затрат. Условная стоимость БУО на РС равна

$$C_2 = \gamma h + \beta k,$$

где γ — стоимость одного выхода РС.

Построение БУО на РС является одним из вариантов реализации БУО как управляющего конечного автомата.

При построении управляющего автомата считают, что его входной алфавит совпадает с множеством сигналов логических условий, выгодной алфавит — с множеством сигналов выполнения отдельных микроопераций, а алфавит состояний и функции переходов задаются графом автомата. Граф микроалгоритма составлен в содержательных обозначениях (микрооперациях и логических условиях) и не может быть непосредственно использован для синтеза управляющего автомата. Чтобы на основе содержательного графа микроалгоритма получить граф управляющего автомата, прибегают к так называемой отметке графа микроалгоритма, правила выполнения которой различны для автоматов Мили и Мура. Заметим, что на практике управляющие автоматы обычно строят как автоматы Мура. Поэтому вначале кратко изложим правила и особенности отметки для автоматов Мили, а затем, более подробно — для автоматов Мура.

Для получения отмеченного графа автомата Мили поступают следующим образом:

1) символом начального состояния Z_0 автомата отмечают вход первой вершины графа, следующей за начальной, а также вход конечной вершины;

2) входы вершин (операторных или условных), следующих за операторными вершинами, отмечают символами состояний $Z_1; Z_2; \dots$;

3) входы двух различных вершин, кроме начальной и конечной, не могут быть отмечены одинаковыми символами;

4) вход вершин может отмечаться только одним символом.

Переход по отмеченному графу из отметки Z_s в соседнюю отметку Z_r при отсутствии между ними условных вершин вызывается синхронизирующими сигналами. Выходные сигналы при таком переходе определяются символами микроопераций, обозначающими операторные

вершины. При наличии между Z_s и Z_r условных вершин A_{ij} ($j = 1, 2, \dots, p$) переход из Z_s в Z_r вызывается конъюнкцией сигналов вида

$$T\tilde{A}_{i_1}\tilde{A}_{i_2} \dots \tilde{A}_{i_p},$$

где T — синхронизирующий сигнал; $\tilde{A}_{ij} = A_{ij}$ — если в переходе используется выход условной вершины A_{ij} , отмеченный единицей, и $\tilde{A}_{ij} = \bar{A}_{ij}$, если в переходе используется выход A_{ij} , отмеченный нулем. Выходные сигналы при этом определяются так же, как и в предыдущем случае. Кроме того, необходимо учитывать, что все переходы из состояния Z_0 вызываются сигналом H и что при всех переходах в состояние Z_0 формируется сигнал K .

При построении управляющего автомата Мура правила отметки графа микроалгоритма состоят в следующем:

- 1) символом Z_0 отмечают начальную и конечную вершины графа;
- 2) каждую операторную вершину отмечают символами $Z_1, Z_2, \dots$;
- 3) две различные операторные вершины не могут быть отмечены одинаковыми символами.

Сигналы, вызывающие переходы из одного состояния в другое, определяются так же, как и для автомата Мили. Согласно приведенным правилам каждая операторная вершина графа микроалгоритма представляется одним состоянием автомата. При этом предполагается, что период T следования синхронизирующих сигналов удовлетворяет условию (6.2). Однако в зависимости от конкретного соотношения периода СС и времени выполнения отдельных микроопераций одной операторной вершине графа микроалгоритма может соответствовать несколько состояний автомата. Согласование времен выполнения МО с периодом СС осуществляется путем введения «пустых» тактов и «пустых» состояний аналогично тому, как это делается в БУО на РС.

Далее будем считать, что период СС определен условием (6.2) и в качестве элементарных автоматов используются синхронные RS — триггеры с внутренней задержкой (т. е. переключающиеся по заднему фронту СС). Тогда графу микроалгоритма умножения рис. 6.6) будет соответствовать граф автомата на рис. 6.10, а. Начальным состоянием автомата является состояние 0. Переход из начального состояния осуществляется под воздействием сигнала H , а переходы из состояний 1, 2, 3, 4 — по СС. Из состояния 5 автомат переходит в состояния 6 или 8 по сигналам A_{02} или $\bar{A}_{02}$ синхронно с СС. Если схема подключения регистра $RG3$ (рис. 6.5) не позволяет совмещать во времени прием слова и анализ LVA_{02} , то между состояниями 5 и 6 должно быть промежуточное состояние для анализа A_{02} . Аналогичная ситуация может возникнуть и при проверке ЛУ $A_{02}\bar{A}_n$, $A_{02}A_n$ и $\bar{A}_n$. Граф автомата при этом имеет вид, показанный на рис. 6.10, б. Однако в этом случае состояния 6 и 10 в соответствующие такты работы автомата будут «пустыми», что снижает быстродействие БУО. Этот недостаток устранен в автомате, граф которого приведен на рис. 6.10, в.

После составления графа автомата необходимо выполнить кодирование его состояний. От способа кодирования в значительной степени зависит надежность и сложность автомата. Понижение надежности при

неоптимальном кодировании состояний обусловлено возможностью возникновения гонок (см. пп. 2.6, 2.10, 3.1). Соседнее кодирование, как метод устранения гонок часто связано с необходимостью введения

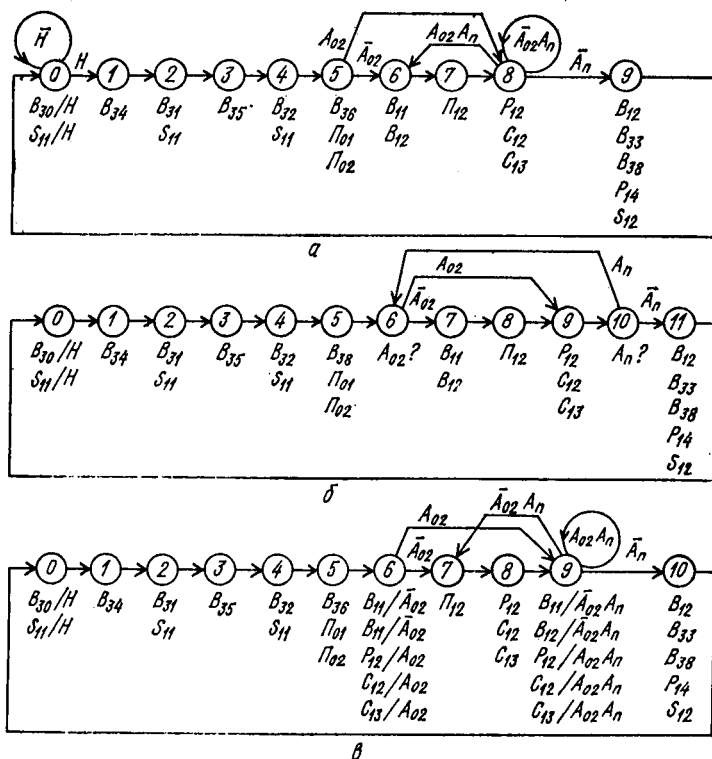


Рис. 6.10

дополнительных состояний (например, в автоматах, графы которых содержат циклы с нечетным числом состояний). При относительно небольшом числе состояний автомата соседнее кодирование удобно осуществлять с помощью диаграмм Вейча, для чего соседним (на графе) состояниям присваивают коды соседних клеток на диаграмме Вейча. Пример кодирования состояний автомата, заданного графом на рис. 6.10, а, приведен на рис. 6.11. Здесь все состояния, кроме 6 и 8, закодированы соседними комбинациями. Кроме того, неиспользованные для кодирования (избыточные) комбинации расположены так, что большинство используемых кодовых комбинаций являются соседними к ним. Это обеспечивает дополнительные возможности для упрощения комбинационной части автомата. Однако нельзя считать, что рассматриваемое кодирование состояний является оптимальным, так как алгоритмы такого кодирования не разработаны (за исключением тривиального перебора всех возможных вариантов кодирования).

		y				
x	8	5	6	7	z	
		4				
		3				
	9	2	1	0		
		t				

Рис. 6.11

Таблица 110

Состояние	Код состояния	Состояния, в которые переходит автомат в $(s+1)$ -й момент		
0	0000	0000/ $\bar{H}$	0001/ H	
1	0001	0101		
2	0101	0111		
3	0111	1111		
4	1111	1101		
5	1101	1001/ $\bar{A}_{02}$	1100/ A_{02}	
6	1001	1000		
7	1000	1100		
8	1100	1001/ $\bar{A}_{02}A_n$	1100/ $A_{02}A_n$	0100/ $\bar{A}_n$
9	0100	0000		

Для определения функций возбуждения триггеров составим кодированную таблицу переходов (табл. 110), из которой следует, что переключение триггера x из состояния 1 в состояние 0 будет происходить при переходе автомата из состояния 8 в состояние 9 при условии $\bar{A}_n$. Следовательно, на наборе 1100 функция возбуждения R этого триггера (обозначаемая далее как R_x) должна быть равна $\bar{A}_n$. На наборах 1111, 1101, 1001, 1000 состояние триггера в s -й и $(s+1)$ -й моменты времени равно 1. Следовательно, на этих наборах $R_x = 0$. На наборах 0000, 0001, 0101, 0100 состояние триггера в s -й и $(s+1)$ -й моменты времени равно 0. Поэтому функция R_x может здесь принимать как значение 0, так и 1. На избыточных наборах R_x может принимать любые значения, так как при правильной работе автомата эти состояния не могут возникать. Аналогично можно определить остальные функции возбуждения. При раздельной минимизации этих функций получим (τ — синхронизирующие сигналы).

$$R_x = y\bar{t}\bar{A}_n\tau; S_x = z\tau; R_y = (y\bar{t}\bar{A}_{02}A_n \vee x\bar{z}\bar{t}\bar{A}_{02} \vee \bar{x}\bar{t})\tau;$$

$$S_y = (\bar{x}\bar{t} \vee x\bar{y}\bar{t})\tau, R_z = x\tau; S_z = \bar{x}y\bar{t}\tau;$$

$$R_t = (x\bar{z}\bar{t}A_{02} \vee \bar{x}y)\tau; S_t = (\bar{x}y\bar{H} \vee x\bar{y}\bar{t}\bar{A}_{02}A_n)\tau.$$

Дешифрируя состояния автомата с учетом избыточных кодовых комбинаций, получим функции выходов:

$$F_0 = \bar{x}y\bar{t}\tau; F_1 = \bar{x}y\bar{t}\tau; F_2 = \bar{x}y\bar{z}t\tau; F_3 = x\bar{z}\bar{t}\tau;$$

$$F_4 = xz\tau; F_5 = x\bar{y}z\bar{t}\tau; F_6 = \bar{x}y\bar{t}\tau; F_7 = \bar{x}y\bar{t}\tau;$$

$$F_8 = x\bar{y}\bar{t}\tau; F_9 = \bar{x}y\bar{t}\tau.$$

Используя введенные ранее обозначения для сигналов микроопераций, можно записать, что

$$B_{30} = F_0H; B_{31} = F_2; B_{32} = F_4; B_{33} = B_{38} = P_{14} = S_{12} = F_9;$$

$$B_{34} = F_1; B_{35} = F_3; B_{36} = \Pi_{01} = \Pi_{02} = F_5; B_{11} = F_6; \Pi_{12} = F_7;$$

$$P_{12} = C_{12} = C_{13} = F_8; S_{11} = F_0 \vee F_2 \vee F_4; B_{12} = F_6 \vee F_9.$$

Функциональная схема автомата, реализующего микроалгоритм умножения, приведена на рис. 6.12. Заметим, что схему следует дополнить цепями установки автомата в исходное состояние.

БУО ЭВМ может быть построен на основе схем, реализующих управление отдельными операциями ($СУ_i$), как показано на рис. 6.13.

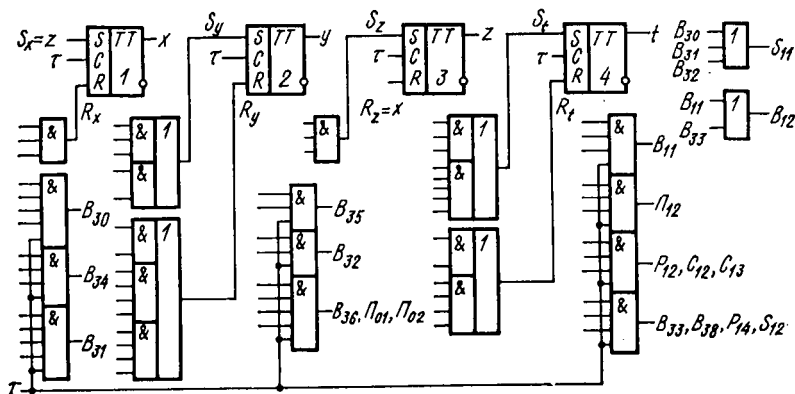


Рис. 6.12

Недостатком такой организации БУО является дублирование реализации ряда общих для всех микроалгоритмов МО. К таким МО относятся МО формирования адреса следующей команды, записи результатов в ЗУ, чтения из ЗУ операндов и команд. Устранить этот недостаток позволяет БУО, где общие для всех операций подготовительные МО реализуются узлом подготовки к следующей команде УПК (штриховая линия на рис. 6.13). Дальнейшего упрощения БУО можно достичь путем объединения не только подготовительных МО, но и всех однотипных МО. Такое объединение удобно производить на стадии подготовки микроалгоритмов по их графам. Объединению могут быть подвергнуты эквивалентные (т. е. совпадающие по составу и времени выполнения МО) участки микроалгоритмов. Пример объединения трех эквивалентных участков микроалгоритмов показан на рис. 6.14, а.

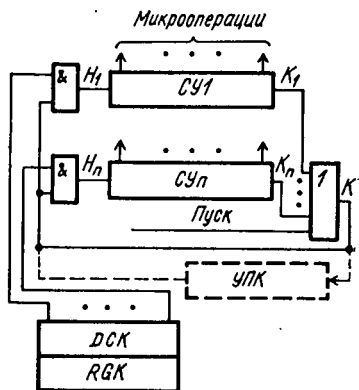


Рис. 6.13

Объединение микроалгоритмов всегда сопровождается уменьшением числа состояний БУО, что приводит к уменьшению числа запоминающих элементов. В то же время процесс объединения сопровождается усложнением комбинационных схем, входящих в БУО. Например, при объединении четырех микроалгоритмов, показанных на рис. 6.14, б, число состояний уменьшилось на 9, что при общем числе состояний, не меньшем 16, может и не приводить к уменьшению числа запоминающих элементов (например, было 28, стало 19, но как в пер-

вом, так и во втором случаях число триггеров равно 5). Однако во втором случае появилась необходимость в сигналах $M_1 \vee M_2$ и $M_3 \vee M_4$, для получения которых необходимо два логических элемента

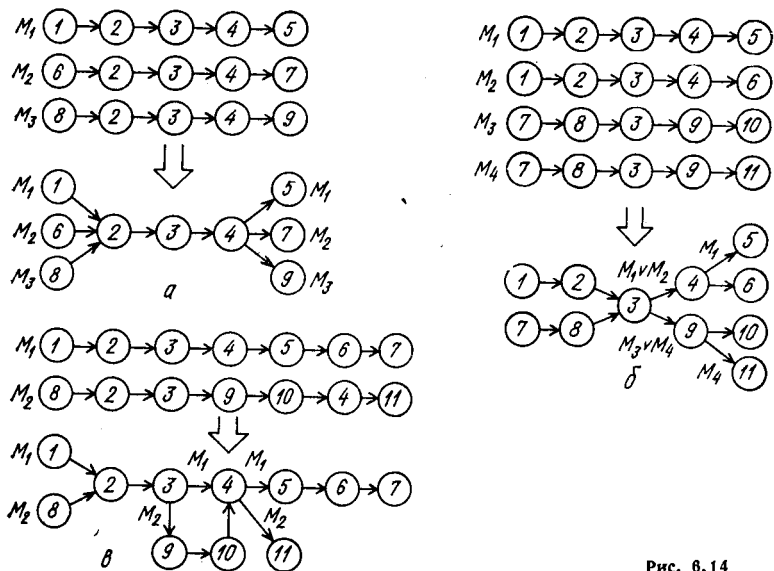


Рис. 6.14

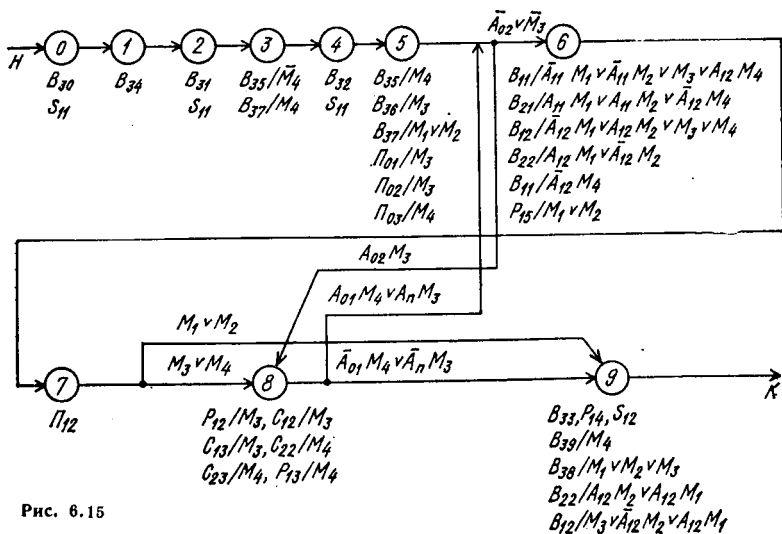


Рис. 6.15

ИЛИ. Вместе с тем, уменьшение числа состояний в рассматриваемом случае увеличивает число избыточных наборов, что открывает дополнительные возможности для упрощения комбинационных схем. Следовательно, объединение микроалгоритмов сложным образом влияет на аппаратные затраты и целесообразность такого объединения

требует обоснования в каждом конкретном случае. Дело осложняется тем, что отсутствуют алгоритмы оптимального объединения микроалгоритмов, кроме тривиального и громоздкого перебора всех вариантов.

Рассмотренные примеры не исчерпывают всех вариантов объединения микроалгоритмов. Еще одна возможность показана на рис. 6.14, в.

Объединенный граф четырех введенных ранее микроалгоритмов показан на рис. 6.15, где M_1 соответствует МАС; M_2 — МАВ; M_3 —

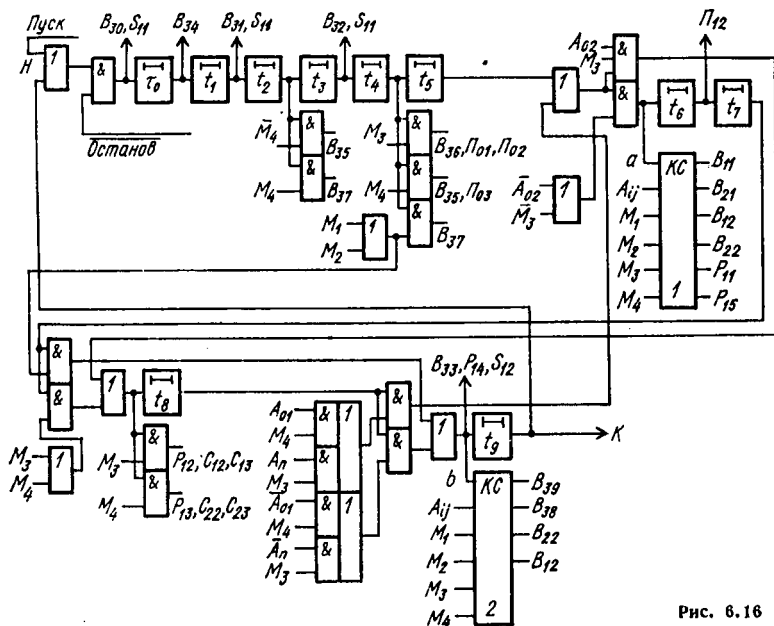


Рис. 6.16

МАУ; M_4 — МАД. По этому объединенному графу построен БУО на ЭЗ (рис. 6.16).

Комбинационные схемы КС1 и КС2 реализуют следующие системы функций:

$$\begin{aligned} B_{11} &= a (\bar{A}_{11}M_1 \vee \bar{A}_{11}M_2 \vee M_3 \vee A_{12}M_4); \\ B_{21} &= a (A_{11}M_1 \vee A_{11}M_2 \vee \bar{A}_{12}M_4); \\ B_{12} &= a (\bar{A}_{12}M_1 \vee A_{12}M_2 \vee M_3 \vee M_4); \\ B_{22} &= a (A_{12}M_1 \vee \bar{A}_{12}M_2); \\ P_{11} &= a \bar{A}_{12}M_4; \\ P_{15} &= a (M_1 \vee M_2); \\ B_{39} &= b M_4; \\ B_{38} &= b (M_1 \vee M_2 \vee M_3); \\ B_{22} &= b (A_{12}M_2 \vee A_{12}M_1); \\ B_{12} &= b (M_3 \vee A_{12}M_2 \vee A_{12}M_1). \end{aligned}$$

Рассмотренные БУО относятся к классу асинхронных.

В синхронных БУО на основе РС (рис. 6.17), общее для всех операций оборудование составляет генератор синхронизирующих сигналов (ГСС), счетчик (СТС), дешифратор (ДСС), регистр кода операции (RGk) и дешифратор кода операции (DCk). Специфичное же для отдельных операций оборудование объединено в узел распределения сигналов (УРС). С помощью ГСС, СТС и ДСС генерируется распределенная во времени и в пространстве серия сигналов q_i . В зависимости от выполняемой операции, определяемой сигналом на выходе DCk, УРС распределяет сигналы q_i по шинам микроопераций. Принцип построения УРС рассмотрим на примере следующих четырех нециклических микроалгоритмов:

$$M_1 = HB_1B_3B_1B_3B_4K; M_2 = HB_2B_1\bar{A}_2 \uparrow B_4 \downarrow K; \\ M_3 = HB_1B_3B_1B_4A_1 \uparrow B_1 \downarrow K; M_4 = HB_4B_1B_2B_3K.$$

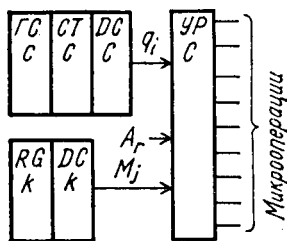


Рис. 6.17

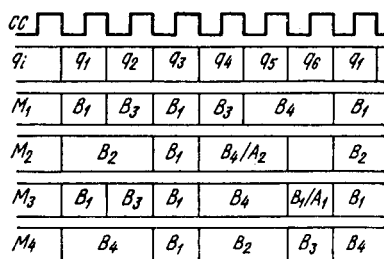


Рис. 6.18

Будем считать, что $t_1 = 1$; $t_2 = 2$; $t_3 = 1$; $t_4 = 2$ — единицы времени и, кроме того, совмещать микрооперации нельзя. Выберем период CC равным 1. Тогда непосредственно по микроалгоритмам можно построить их временные диаграммы работы (рис. 6.18). Для сигналов микроопераций получаем следующую систему функций:

$$B_1 = q_3M_1 \vee q_3M_2 \vee q_1M_1 \vee q_3M_3 \vee q_6A_1M_3 \vee q_3M_4; \\ B_2 = q_1M_2 \vee q_2M_2 \vee q_4M_4 \vee q_5M_4; \\ B_3 = q_2M_1 \vee q_4M_1 \vee q_2M_3 \vee q_6M_4; \\ B_4 = q_5M_1 \vee q_6M_1 \vee q_4M_2A_2 \vee q_5M_2A_2 \vee q_4M_3 \vee q_5M_3 \vee \\ \vee q_1M_4 \vee q_2M_4.$$

Реализовать эти функции в УРС удобно с помощью матрицы элементов И с последующей сборкой их выходов на элементах ИЛИ (рис. 6.19). УРС может быть минимизирован по числу элементов И следующим образом. Если сигнал q_i используется для формирования одних и тех же сигналов B_r в строках матрицы, число p которых больше $\frac{m}{2}$ (m — число микроалгоритмов или число строк), то вместо конъюнкций q_iM_j ($j \in \{1, 2, \dots, p\}$) целесообразно формировать конъюнкции $q_i\bar{M}_j$ ($j \in \{1, 2, \dots, p\}$), число которых равно $m - p$. Кроме того, значительного упрощения можно достичь за счет вынесения за скобки q_i . В предельном случае, когда $p = m$, вообще нет необходимости формирования указанных конъюнкций, а шина q_i не-

посредственно подключается к выходу УРС. В рассматриваемом примере по q_3 во всех микроалгоритмах выполняется микрооперация B_1 . Поэтому элементы И в узлах q_3 можно опустить (на рис. 6.19 эти узлы отмечены знаком $\times$). Кроме того, микроалгоритм M_2 завершается по q_3 или q_5 , поэтому элемент И в узле (q_6, M_2) также можно опустить. Еще одна возможность упрощения УРС заключается в том, что

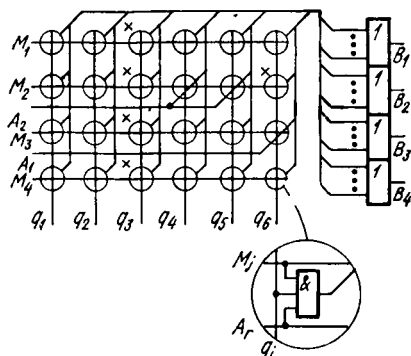


Рис. 6.19

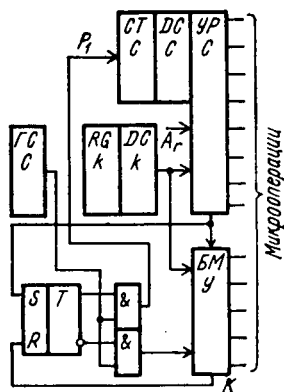


Рис. 6.20

CC	1	2	3	4	5	6	7	8	9
q_i	1	2	3	4	5	6	7	8	9
M_1	B_{30} S_{11}	B_{34}	B_{31} S_{11}	B_{35}	B_{32} S_{11}	B_{37}	$B_{11}/\bar{A}_{11}$ $B_{21}/\bar{A}_{11}$ $B_{12}/\bar{A}_{12}$ $B_{22}/\bar{A}_{12}$	Π_{12}	P_{14}, S_{12} B_{33}, B_{38} $B_{12}/\bar{A}_{12}$ $B_{22}/\bar{A}_{12}$
M_2	B_{30} S_{11}	B_{34}	B_{31} S_{11}	B_{35}	B_{32} S_{11}	B_{37}	$B_{11}/\bar{A}_{11}$ $B_{21}/\bar{A}_{11}$ $B_{12}/\bar{A}_{12}$ $B_{22}/\bar{A}_{12}$	Π_{12}	P_{14}, S_{12} B_{35}, B_{38} $B_{12}/\bar{A}_{12}$ $B_{22}/\bar{A}_{12}$
M_3	B_{30} S_{11}	B_{34}	B_{31} S_{11}	B_{35}	B_{38} S_{11}	P_{12}, Π_{01} B_{36}	R	S/K	P_{14}, S_{12} B_{33}, B_{38} B_{12}
M_4	B_{30} S_{11}	B_{34}	B_{31} S_{11}	B_{37}	B_{32} S_{11}	B_{35} Π_{03}	R	S/K	P_{14}, S_{12} B_{33}, B_{39}

Рис. 6.21

если сигналы микроопераций не должны иметь единичного значения на протяжении всего времени их выполнения, то некоторые из них можно не формировать. В УРС на рис. 6.19 при указанном допущении можно не формировать произведения $q_2 M_2$, $q_2 M_4$, $q_5 M_2$, $q_5 M_3$, $q_5 M_4$, $q_6 M_1$.

Кроме синхронных и асинхронных БУО в ряде случаев используются комбинированные синхронно-асинхронные БУО (рис. 6.20). Здесь все операции разбивают на две группы, а именно: малотактные (сложение, вычитание и т. п.) и многотактные. Все малотактные операции и подготовительные микрооперации многотактных операций

чаем



$$\begin{aligned} & \vee q_7 A_{12} M_2 \vee q_9 \bar{A}_{12} (M_1 \vee M_2) \vee q_9 M_3, B_{22} = q_7 A_{12} M_1 \vee \\ & \vee q_7 \bar{A}_{12} M_2 \vee q_9 A_{12} (M_1 \vee M_2), \Pi_{12} = q_8 (M_1 \vee M_2), \Pi_{01} = \\ & = q_6 M_3, \Pi_{02} = q_6 M_3, \\ & \Pi_{03} = q_6 M_4, P_{15} = q_7 (M_1 \vee M_2), P_{14} = q_9, S_{11} = q_1 \vee q_3 \vee q_5, \\ & S_{12} = q_9, R = q_7 (M_3 \vee M_4), S = q_8 K (M_3 \vee M_4). \end{aligned}$$

В заключение укажем, что простота и относительная регулярность синхронно-асинхронных БУО послужили причинами их широкого распространения в ЭВМ 1-го и 2-го поколений.

За один такт работы цифровой ЭВМ может выполняться несколько микроопераций на различных узлах и операционных блоках. При микропрограммном управлении операциями каждой микрооперации, или некоторой их совокупности ставится в соответствие слово, называемое микрокомандой. Следовательно, для каждой операции можно составить микропрограмму, состоящую из совокупности микрокоманд. Саму процедуру составления микропрограмм называют микропрограммированием. Микропрограммы записывают в ЗУ подобно тому, как записывают обычные программы. Управление операциями в этом

случае осуществляется путем последовательного считывания и расшифровки микрокоманд. Иногда такое управление еще называют «хранимой логикой», чем подчеркивается, что алгоритм управления реализуется не в виде специальной схемы, а в виде закодированной информации, записанной в ЗУ.

Идея микропрограммного управления операциями была предложена еще во время проявления первых цифровых ЭВМ. Однако в цифровых ЭВМ первого и второго поколений этот принцип не получил распространения. Объясняется это с одной стороны тем, что микропрограммирование рассматривалось раньше как один из способов повышения гибкости программирования и каждый программист должен был строить из микрокоманд тот набор операций, который ему был нужен. Это требовало высокой квалификации программистов и затрудняло массовое применение микропрограммных цифровых ЭВМ. Сейчас же в связи с разработкой языков программирования более высокого уровня, чем программирование в машинных адресах, микропрограммирование вышло из ведения программистов и превратилось в метод построения системы управления цифровых ЭВМ, удобный для инженера-разработчика. Нынешний пользователь цифровой ЭВМ может и не знать о микропрограммной основе управления вычислительным процессом.

Вторая причина задержки реализации микропрограммного принципа управления заключалась в отсутствии в прошлом достаточно быстродействующих ЗУ невысокой стоимости, пригодных для этих целей. Микроэлектроника позволила создать такие ЗУ, и многие цифровые ЭВМ третьего и четвертого поколений строятся с микропрограммным управлением.

Устройства микропрограммного управления могут строиться на основе ОЗУ и ПЗУ (иногда микропрограммные БУО на основе ОЗУ называют динамическими, а на основе ПЗУ — статическими). В первом случае набор микрокоманд можно менять как в процессе производства, так и в процессе эксплуатации цифровых ЭВМ. Использование же ПЗУ предопределяет неизменность микропрограмм в процессе эксплуатации цифровых ЭВМ. Чаще всего микропрограммные БУО строятся по схеме на рис. 6.23. Такой БУО состоит из ПЗУ

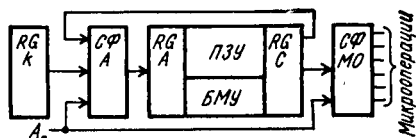


Рис. 6.23

микропрограмм, схемы формирования сигналов микроопераций (СФМО) и схемы формирования адреса следующей микрокоманды (СФА). После каждого обращения к ПЗУ в регистре слов РГС будет находиться очередная микрокоманда, одна часть которой используется для формирования адреса следующей микрокоманды (с помощью СФА), а другая — для генерации сигналов выполнения отдельных микроопераций (с помощью СФМО).

В зависимости от принятого способа кодирования микроопераций различают два варианта организации микропрограммного управления: горизонтальное и вертикальное микропрограммирование. При горизонтальном микропрограммировании для каждой микрооперации выделяется один разряд в микрокоманде. Для рассматриваемой цифровой ЭВМ будем считать, что микрооперациям $B_{30} — B_{39}$ соответствуют разряды 0—9 микрокоманды, микрооперациям $B_{11} — B_{22}$ соответствуют разряды 10—13, соответствие же между остальными микрооперациями и разрядами микрокоманды следующее:

$P_{12} - 14, P_{11} - 15, P_{12} - 16, P_{01} - 17, P_0 - 18, C_{12} - 19,$
 $C_{13} - 20, C_{22} - 21, C_{23} - 22, P_{03} - 23, P_{13} - 24, P_{15} - 25,$
 $P_{14} - 26, S_{11} - 27, S_{12} - 28.$

При таком кодировании все одновременно выполняемые микрооперации определяются единицами в соответствующих разрядах одной микрокоманды. Код операции, реализуемой с микропрограммным управлением, задает адрес первой микрокоманды в микропрограмме. Адреса следующих микрокоманд определяются по принципу принудительной адресации, вследствие чего микрокоманда состоит из

Таблица 111

МПС			МПВ			МПУ			МПД						
α_0	B_{30}, S_{11}	α_1	β_0	B_{30}, S_{11}	β_1	γ_0	B_{30}, S_{11}	γ_1	μ_0	B_{30}, S_{11}	μ_1				
α_1	B_{34}	α_2	β_1	B_{34}	β_2	γ_1	B_{34}	γ_2	μ_1	B_{34}	μ_2				
α_2	B_{31}, S_{11}	α_3	β_2	B_{31}, S_{11}	β_3	γ_2	B_{31}, S_{11}	γ_3	μ_2	B_{31}, S_{11}	μ_3				
α_3	B_{35}	α_4	β_3	B_{35}	β_4	γ_3	B_{35}	γ_4	μ_3	B_{37}	μ_4				
α_4	B_{32}, S_{11}	α_5	β_4	B_{32}, S_{11}	β_5	γ_4	B_{32}, S_{11}	γ_5	μ_4	B_{32}, S_{11}	μ_5				
α_5	B_{37}	α_6	β_5	B_{37}	β_6	γ_5	$\Pi_{01}, \Pi_{02}, B_{36}$	γ_6	μ_5	B_{35}, Π_{03}	μ_6				
α_6	$B_{11}/\bar{A}_{11}$	α_7	β_6	$B_{11}/\bar{A}_{11}$	β_7	γ_6	B_{11}, B_{12}	γ_7	μ_6	B_{11}/A_{12}	μ_7				
	B_{21}/A_{11}			γ_7		Π_{12}	γ_8	$B_{21}\bar{A}_{12}$							
	$B_{12}/\bar{A}_{12}$ B_{22}/A_{12} P_{15}			B_{12}/A_{12} $B_{22}/\bar{A}_{12}$ P_{15}		γ_8	P_{12} C_{12} C_{13}	$\gamma_6/\bar{A}_{02}A_n$ $\gamma_8/A_{02}A_n$ $\gamma_9/\bar{A}_n$		$P_{11}/\bar{A}_{12}$ B_{12}					
α_7	Π_{12}	α_8	β_7	Π_{12}	β_8	γ_9	B_{38}, B_{38} P_{14}, S_{11}	?	μ_8	Π_{12}	μ_8				
α_8	$B_{12}/\bar{A}_{12}$ B_{22}/A_{12}	?	β_8	$B_{12}/\bar{A}_{12}$ B_{22}/A_{12}	?					C_{22}, C_{23} $P_{13}/\bar{A}_{12}$					
	P_{11}, S_{12} B_{33}, B_{38}			P_{14}, S_{12} B_{33}, B_{38}						B_{33}, B_{39} P_{14}, S_{12}					
							B_{13}		μ_9						

двух частей — микрооперационной и адресной. Набор возможных микрокоманд в таком БУО составляет 2^n , где n — число различных микроопераций или рядов для представления микрокоманды. В современных цифровых ЭВМ величина n может достигать нескольких сотен. Такое многообразие сочетаний выполняемых микроопераций позволяет создать широкий набор операций и строить наиболее эффективные программы. Однако значительное

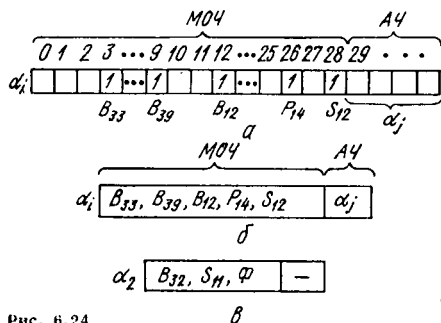


Рис. 6.24

число возможных операций оказывается лишенным смысла.

Для рассматриваемого набора микроопераций структура микрокоманды показана на рис. 6.24, а. В микрооперационной части (МОЧ) микрокоманды единицы расположены на позициях 3, 9, 12, 26, 28. Это означает, что при выборке из ПЗУ по адресу α_i этой микрокоманды должны одновременно формироваться сигналы микроопераций $B_{33}, B_{39}, B_{12}, P_{14}, S_{12}$. В адресной части (АЧ) микрокоманды записан адрес α_j следующей микрокоманды. Учитывая, что в МОЧ число единиц, как правило, невелико, запись микропрограмм можно производить более компактно путем указания символов позиций в МОЧ, содержащих единицы. На рис. 6.24, б так записана микрокоманда с рис. 6.24, а. С использованием приведенных обозначений микропрограммы операций сложения (МПС), вычитания (МПВ), умножения (МПУ) и деления (МПД) можно представить в табл. 111. На основе микропрограмм отдельных операций составляется микропрограмма работы цифровой ЭВМ (табл. 112), в микрокомандах которой МОЧ

Таблица 112

МПР		
α_0	B_{30}, S_{11}	α_1
α_1	B_{34}	α_2
	B_{31}, S_{11}	α_3/M_1
α_2		β_3/M_2
		γ_3/M_3
		μ_3/M_4
α_3	B_{35}	α_4
$\vdots$	МПС	$\vdots$
α_8		α_0
β_3	B_{35}	β_4
$\vdots$	МПВ	$\vdots$
β_8		α_0
γ_3	B_{35}	γ_4
$\vdots$	МПУ	$\vdots$
γ_9		α_0
μ_3	B_{37}	μ_4
$\vdots$	МПД	$\vdots$
μ_9		α_0

условно опущена. Так как микропрограммы всех операций содержат одинаковые участки из трех начальных микрокоманд, то при объединении их в микропрограмму работы (МПР) ЭВМ указанный участок составлен только в МПС. Вследствие этого машинный цикл всегда начинается с выполнения этого участка. После выполнения общей для всех операций микрокоманды B_{31} , S_{11} , записанной в ячейке α_2 , производится анализ кода операции (M_1 , M_2 , M_3 или M_4) и переход на выполнение соответствующей микропрограммы. Последняя микрокоманда любой из них передает управление микрокоманде из ячейки α_0 , тем самым начиная новый машинный цикл.

Сложность СФА и СФМО существенно зависит от способа размещения микропрограмм в ПЗУ и от количества проверяемых условий. За счет увеличения количества микрокоманд и некоторого усложнения СФА можно вообще исключить СФМО и формировать сигналы микроопераций непосредственно на выходе ЗУ микропрограмм. Например, часть МПС, хранящуюся в ячейках α_5 — α_8 , можно преобразовать так, как показано в табл. 113. Вследствие этого все сигналы микроопераций формируются независимо от каких-либо условий. Однако при этом увеличивается длина микропрограмм и возникает необходимость в проверке условий при формировании адресов (микрокоманды из ячеек α_5 и α_{10}).

Таблица 113

α_5	B_{37}	$\alpha_8/\bar{A}_{11}A_{12}, \alpha_9/\bar{A}_{11}\bar{A}_{12},$ $\alpha_6/A_{11}A_{12}, \alpha_7/A_{12}\bar{A}_{12}$
α_6	B_{21}, B_{22}, P_{15}	α_{10}
α_7	B_{21}, B_{12}, P_{15}	α_{10}
α_8	B_{11}, B_{22}, P_{15}	α_{10}
α_9	B_{11}, B_{12}, P_{15}	α_{10}
α_{10}	P_{12}	$\alpha_{11}/\bar{A}_{12}, \alpha_{12}/A_{12}$
α_{11}	$B_{12}, P_{14}, S_{12}, B_{33}, B_{38}$	α_0
α_{12}	$B_{22}, P_{14}, S_{12}, B_{33}, B_{38}$	α_0

Количество проверяемых условий можно несколько сократить за счет введения новых микроопераций. Если, например, ввести микрооперацию формирования адреса следующей микрокоманды путем передачи слова с RGk в RGA ЗУ микропрограмм (обозначим ее как

Φ), в качестве кодов операций M_1, M_2, M_3 и M_4 принять соответственно 00000, 01000, 10000 и 11000, а в ячейке ЗУ микропрограмм с такими адресами записать первые микрокоманды МПС, МПВ, МПУ и МПД (т. е. $\alpha_3 = 00000$, $\beta_3 = 01000$, $\gamma_3 = 10000$, $\mu_3 = 11000$), в микрокоманде из ячейки α_2 условия формирования адреса можно исключить. Сама микрокоманда примет вид, показанный на рис. 6.24, в.

В адресной части этой микрокоманды может быть указан любой адрес, так как в формировании адреса следующей микрокоманды он участия не принимает. Пять разрядов для представления кода операции выбраны с учетом того, что для хранения МПР в данном случае достаточно 32 ячейки ЗУ микропрограмм и, следовательно, его RGA будет содержать 5 разрядов. Но для кодирования 4 операций достаточно двух разрядов. Поэтому три младших разряда из кода операции

можно исключить и дополнять ими код операции только при формировании адреса первой микрокоманды соответствующей микропрограммы по сигналу Φ , осуществляющему передачу слова из RGA в RGA.

Размещение микрокоманд в ЗУ также влияет на сложность СФА. Один из способов уменьшения сложности СФА состоит в соседнем

Таблица 114

$\mu_3 = 11000$	B_{37}	11001
$\mu_4 = 11001$	$\vdots$	11010
$\mu_5 = 11010$	$\vdots$	11011
$\mu_6 = 11011$	$\vdots$	11100
$\mu_7 = 11100$	$\vdots$	11101
$\mu_8 = 11101$	$\vdots$	11111/ $\bar{A}_{01}$ 11011/ A_{01}
$\mu_9 = 11111$	$\vdots$	α_0

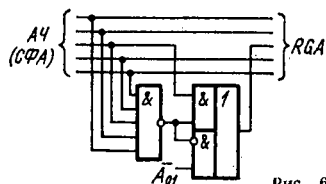


Рис. 6.25

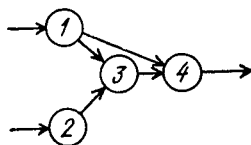


Рис. 6.26

кодировании адресов, по которым записываются ветви микропрограмм. Например, в МПД ветвление происходит в микрокоманде из ячейки μ_8 , после выполнения которой осуществляется переход к микрокоманде из ячейки μ_6 или μ_9 . Если μ_8 и μ_9 будут соседними кодовыми комбинациями (табл. 114), то при формировании адреса необходимо изменить только один разряд в зависимости от условия A_{01} , что, как правило, схемно реализовать проще, чем преобразование двух и более разрядов адреса (рис. 6.25). При таком формировании адреса в АЧ той микрокоманды, по которой осуществляется ветвление, указывается адрес первой микрокоманды ветви, переход к которой про-

исходит только от одной микрокоманды (в данном случае — 11111). Если в каждой части микропрограммы возможен переход от двух или более микрокоманд, то указанным приемом воспользоваться нельзя, так как может возникнуть ложное ветвление. Примером такой ситуации является микропрограмма, составленная по графу на рис. 6.26. В первой микрокоманде указать адрес третьей микрокоманды нельзя, так как будет возникать ложное ветвление при переходе к ней от второй микрокоманды. По этой же причине нельзя указывать и адрес четвертой микрокоманды, так как будет ложное ветвление при переходе к четвертой микрокоманде от третьей. В этом случае необходимо сформировать признак перехода по условию. Таким признаком может быть единица в специально выделенном разряде микрокоманды или адрес микрокоманды, после выполнения которой осуществляется ветвление. В последнем случае необходимо дешифровать адрес микрокоманды, осуществляющей ветвление, и, следовательно, выход RGA должен быть подключен ко входам СФА.

Как видно из рассмотренных примеров, основным преимуществом горизонтального микропрограммирования является повышенное быстродействие как за счет простоты или вообще отсутствия СФМО и возможности одновременной генерации произвольного числа сигналов микроопераций, так и за счет быстрого формирования адреса следующей микрокоманды. Однако при горизонтальном микропрограммировании длина МОЧ должна быть не меньше максимального числа несовместимых микроопераций, то есть требуются «длинные» форматы микрокоманд и ячейки ЗУ, что приводит к значительным затратам оборудования. Кроме того, лишь небольшое число разрядов в МОЧ будет содержать единицы и, следовательно, ЗУ будет использоваться неэффективно.

Сократить длину микрокоманд позволяет применение вертикального микропрограммирования, при котором каждая микрооперация кодируется $\lceil \log_2 n \rceil$ — разрядным кодом, где n — общее число микроопераций. Такое кодирование накладывает ограничение на методы выполнения операций, а именно: не должно быть операций, требующих одновременного выполнения ряда микроопераций. В тех случаях, когда это ограничение выполнить нельзя, следует вводить сложные микрооперации, состоящие из совокупности простых. Например, при применении комбинационного сумматора раздельная выдача слагаемых с регистров недопустима. В этом случае следует ввести сложную микрооперацию «Выдача слов с регистров 1 и 2», состоящую из двух простых «Выдача слова с регистра 1» и «Выдача слова с регистра 2». Однако такой прием усложняет СФМО как за счет появления дополнительных входов у дешифратора кода микроопераций, так и за счет последующей сборки на элементах ИЛИ шин сложных микроопераций и их распределения по цепям исходного множества микроопераций (рис. 6.27, а).

Вследствие недостатков, присущих горизонтальному и вертикальному способам микропрограммирования, они не получили широкого распространения в чистом виде и используются только в тех случаях, где специфика наборов микроопераций и микроалгоритмов сглаживает указанные недостатки. Например, горизонтальное микропрограм-

мирование может быть применено в специализированных или универсальных ЦЭВМ с небольшим количеством микроопераций.

Устранить в значительной степени недостатки рассмотренных способов микропрограммирования позволяет комбинированное (смешанное) микропрограммирование, при котором все множество микроопераций разбивают на непересекающиеся подмножества. Каждое подмножество имеет свое поле в микрокоманде, в котором указывается

Таблица 115

МПС						
	V_1	V_2	V_3	V_4	V_5	АЧ
α_0	B_{30}	0	0	S_{11}	0	α_1
α_1	B_{34}	0	0	0	0	α_2
α_2	B_{31}	0	0	S_{11}	0	α_3
α_3	B_{35}	0	0	0	0	α_4
α_4	B_{32}	0	0	S_{11}	0	α_5
α_5	0	0	0	B_{37}	0	α_6
α_6	0	0	0	0	0	α_7
α_7	0	Π_{12}	0	0	0	α_8
α_8	B_{33}	B_{38}	$B_{12}/\bar{A}_{12}$ B_{22}/A_{12}	P_{14}	S_{12}	α_0

$\lceil \log_2 (m + 1) \rceil$ — разрядный номер микрооперации (m_i — число микроопераций в данном

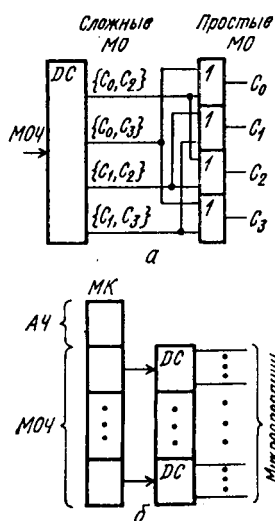


Рис. 6 .27

подмножестве, а +1 означает, что в каждом поле должна быть предусмотрена кодовая комбинация, при которой

ни одна из микроопераций данного подмножества не выполняется). В этом случае в одной микрокоманде можно совмещать произвольное количество микроопераций, но при условии, что взято не более одной микрооперации из каждого подмножества. При комбинированном микропрограммировании СФМО будет состоять из набора дешифраторов, каждый из которых дешифрирует свое поле микрокоманды (рис. 6.27, б). Если число подмножеств микроопераций взять равным максимальному числу микроопераций, выполняемых одновременно по одной микрокоманде, и в каждое из подмножеств включить несовместимые (никогда одновременно не выполняемые) микрооперации, то длина микропрограммы комбинированного формата будет равна

длине соответствующей микропрограммы горизонтального формата. Быстродействие же БУО с комбинированным микропрограммированием по сравнению с БУО горизонтального типа уменьшится только на время прохождения сигналов микроопераций через дешифратор поля микрокоманды (рис. 6.27, б).

Для рассмотренных ранее МПС, МПВ, МПУ и МПД число подмножеств равно пяти, так как в микрокомандах из ячеек α_8 и β_8 пять микроопераций. Пять непересекающихся подмножеств микроопераций будут иметь вид

$$\begin{aligned} V_1 &= \{B_{30}, B_{31}, B_{32}, B_{33}, B_{34}, B_{35}, B_{36}\}; \\ V_2 &= \{B_{38}, B_{39}, B_{11}, B_{21}, P_{12}, P_{12}, P_{01}\}; \\ V_3 &= \{B_{12}, B_{22}, P_{11}, P_{02}, C_{12}, C_{23}, P_{03}\}; \\ V_4 &= \{B_{37}, C_{13}, C_{22}, P_{13}, P_{15}, P_{14}, S_{11}\}; \\ V_5 &= \{S_{12}\}. \end{aligned}$$

При таком разбиении на подмножества каждая микрокоманда будет содержать в МОЧ четыре поля по 3 разряда и одно поле длиной в 1 разряд. Структура и количество микрокоманд в микропрограмме при этом не изменится. Например, МПС будет представлена табл. 115, в которой 0 означает, что в данной микрокоманде по данному полю сигнал микрооперации не формируется. Нулю соответствует кодовая комбинация, которая дешифратором поля не декодируется. Дешифраторы полей, таким образом, будут неполными.

Рассмотренные примеры микропрограммных БУО относятся к классу синхронных, так как все микрооперации здесь выполняются за одно и то же время в течение одного такта. Однако при значительном разбросе времени выполнения отдельных микроопераций это приводит к существенному замедлению вычислительного процесса. Устранить этот недостаток можно путем применения асинхронных БУО, в которых переменное время выполнения микроопераций вводится либо за счет пустых микрокоманд, либо за счет дополнительного поля в микрокоманде, где указывается количество периодов СС, необходимых для данной микрокоманды. В первом случае период СС выбирают равным длительности самой быстрой микрокоманды. Медленные микрокоманды выполняют за несколько периодов СС путем включения в микропрограмму соответствующего числа пустых микрокоманд. Однако при этом возрастает требуемая емкость ЗУ за счет удлинения микропрограмм. Во втором случае содержимое поля, задающего длительность микрокоманды, записывается в таймер (преобразователь «код — время»), выполненный, например, на основе вычитающего счетчика. Сигнал заема со старшего разряда этого счетчика служит сигналом выборки следующей микрокоманды. При этом длина микропрограмм остается неизменной, однако увеличивается длина микрокоманд и, следовательно, емкость ЗУ. Кроме того, возрастает сложность разрядных цепей и схем управления ЗУ микропрограмм.

Все рассмотренные выше микрокоманды относятся к классу однофазных в том смысле, что выполнение всех микроопераций данной микрокоманды совмещено во времени. Несколько уплотнить информацию в ЗУ микропрограмм и повысить быстродействие БУО за счет одновременной выборки более информативных микрокоманд позволя-

ет применение многофазных микрокоманд, в которых выполнение отдельных микроопераций разнесено во времени. Простейшую двух-фазную микрокоманду можно получить путем объединения двух одно-фазных микрокоманд (рис. 6.28), при котором МОЧ i -й и $i + 1$ -й микрооперации с одной АЧ образуют новую микрокоманду, выполняемую за два такта. В общем случае k -фазная команда позволяет сократить емкость ЗУ на $k - 1$ адресных частей и за одно обращение к памяти выбирать k однофазных команд. При этом длина МОЧ увеличивается в k раз. Однако во многих случаях удастся ограничить рост МОЧ за счет разнесения по разным фазам несовместимых микроопераций. Например, если в МПС микрооперации распределить по фазам f_1 и f_2 следующим образом:

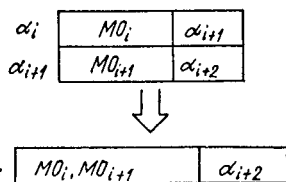


Рис. 6.28

$$\{B_{30}, B_{31}, B_{32}, B_{33}, B_{38}, B_{11}, B_{21}, B_{12}, B_{22}, P_{15}, P_{14}, S_{11}, S_{12}\} \rightarrow f_1,$$

$$\{B_{34}, B_{35}, B_{37}, \Pi_{12}\} \rightarrow f_2,$$

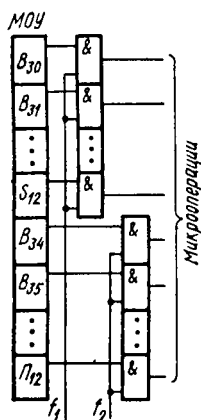


Рис. 6.29

МР		
МК0	0	МОЧ
МК1	1	$\bar{A}$ АЧ

Рис. 6.30

Таблица 116

МПС			
	f_1	f_2	АЧ
α_0	$B_{30}; S_{11}$	B_{34}	α_3
α_2	$B_{31}; S_{11}$	B_{35}	α_4
α_4	$B_{32}; S_{11}$	B_{37}	α_6
α_6	$B_{11}/\bar{A}_{11};$ $B_{21}/\bar{A}_{11};$ $B_{12}/\bar{A}_{12};$ $B_{22}/\bar{A}_{12};$ P_{15}	Π_{12}	α_8
α_8	$B_{33}; B_{38};$ $B_{12}/\bar{A}_{12};$ $B_{22}/\bar{A}_{12};$ $P_{14}; S_{12}$		α_0

то длина МПС сократится почти в два раза (табл. 116), а количество разрядов в микрокоманде останется неизменным. Схемные отличия БУО будут состоять в том, что в СФМО сигналы микроопераций синхронизируются двухфазными СС (рис. 6.29).

Кроме принудительной адресации в микропрограммных БУО используют и естественную адресацию, позволяющую сократить длину микрокоманд за счет отсутствия АЧ. При такой адресации микрокоманды располагают в ЗУ так, чтобы адрес следующей микрокоманды был на единицу больше адреса данной. Вследствие этого адрес следующей микрокоманды формируется либо на *RGA*, выполненном как счетчик, либо на специальном счетчике адреса микрокоманд. Для реализации ветвлений вводят микрокоманды условного и безусловного переходов (МКП), отличающиеся от микрокоманд операционного типа (МКО), например, единицей в специальном маркерном разряде МР (рис. 6.30). В МКП кроме АЧ имеется поле, в котором размещается

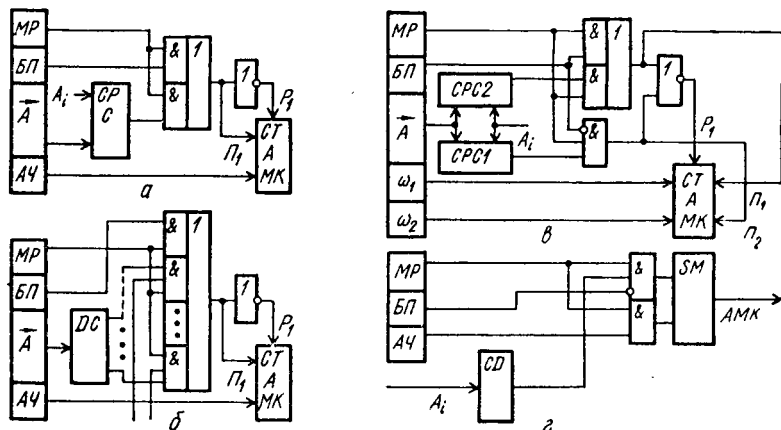


Рис. 6.31

вектор значений логических условий A или, в простейшем случае, номер логического условия, по которому осуществляется ветвление. Отличить микрокоманду условного перехода от микрокоманды безусловного перехода (БП) можно либо с помощью специального разряда в поле A , либо путем введения кодовой комбинации в поле A , которая никогда не встречается при условных переходах. Схемная реализация этих идей показана на рис. 6.31. В первой схеме переход к микрокоманде, адрес которой задан в АЧ микрокоманды перехода, осуществляется по совпадению на схеме равенства слов (CPC) логических условий A_i с их значениями, указанными в микрокоманде. Во второй схеме такой переход происходит путем опроса $\bar{A}_i$ (взятого с отрицанием или без него), номер которого указан в поле $\bar{A}$ и дешифрован с помощью дешифратора ДС. Если условие перехода не выполняется, то осуществляется переход к следующей микрокоманде путем прибавления единицы на счетчике адреса микрокоманд (СТАМК).

Описанные приемы удобно применять при переходе к двум возможным продолжениям микропрограммы. Однако в ряде случаев требуется осуществлять ветвление по трем и более направлениям. Например, в МПУ после выполнения микрокоманды из ячейки $\gamma_{(8)}$ можно переходить к сдвигу, сложению или на конец операции. При реализации таких

ветвлений в АЧ микрокоманды должно быть указано не менее двух адресов микрокоманд, к которым возможен переход (рис. 6.31, а). Сложность СФА растет при этом пропорционально числу направлений ветвления как за счет СРС, так и за счет удлинения АЧ. При большом числе направлений ветвления и большой емкости ЗУ микропрограмм адресная часть МКП может не поместиться в формате микрокоманды. Сократить длину АЧ как при естественном, так и при принудительном способе формирования адреса следующей микрокоманды можно за счет использования не прямой адресации (присоединенной или относительной), при которой в АЧ указывают только младшие разряды адреса. В качестве старших разрядов может быть использован код операции. Если же микропрограммы отдельных операций значительно отличаются по количеству занимаемых ячеек ЗУ, то, с целью более полного использования памяти, должна быть предусмотрена микрооперация замены слова в регистре базового адреса при выполнении длинных микропрограмм. В качестве такого регистра может быть использован *RGk*.

При относительной адресации микрокоманд, реализуемой с помощью специального сумматора адреса, появляются возможности упрощения комбинационной части СФА при трех и более направлениях ветвления. Так как количество возможных кодовых комбинаций, создаваемых сигналами логических условий, не меньше максимального числа направлений ветвления, то просуммировав такую кодовую комбинацию с АЧ, получим адрес первой микрокоманды соответствующей ветви (рис. 6.31, б). Такой метод позволяет либо вообще отказаться от поля логических условий в микрокоманде перехода, либо использовать его для маскирования логических условий, несущественных для данной микрокоманды, однако накладывает жесткие ограничения на размещение микропрограмм. Несколько смягчить эти ограничения можно путем использования шифратора СД. В простейшем случае СД осуществляет перекоммутацию или сдвиг на i разрядов вектора логических условий, увеличивая тем самым объем ЗУ для одной ветви микропрограммы в 2^i раз. Следует отметить, что многонаправленное ветвление может быть реализовано путем набора микропрограмм перехода двунаправленным ветвлением, но при этом увеличивается длина микропрограмм и, следовательно, снижается быстродействие цифровой ЭВМ.

Из рассмотренных выше примеров видно, что методы микропрограммирования и соответствующие им структуры БУО могут быть классифицированы по многим признакам, а именно: по типу используемого ЗУ — динамические (на основе активного ЗУ) и статические (на основе ПЗУ); по способу кодирования микроопераций — горизонтальные, вертикальные и смешанные; по длительности выполнения отдельных микроопераций — синхронные и асинхронные; по числу групп микроопераций, задаваемых одной микрокомандой, выполнение которых может быть совмещено во времени — однофазные и многофазные; по способу формирования адреса следующей микрокоманды — с принудительной и естественной адресацией, с прямой и не прямой адресацией; по способу управления ветвлениями микропрограмм — с использованием схем равенства слов и дешифраторов

логических условий и др. Сочетания этих признаков в конкретных БУО определяют все многообразие вариантов их построения (заметим, что только с учетом перечисленных признаков возможны 192 варианта микропрограммных БУО). Выбор методов и средств, используемых при микропрограммировании, осуществляется на основе технико-экономического анализа проектируемой цифровой ЭВМ, состава команд, реализуемых аппаратным способом, а также их микропрограмм.

Сравнивая между собой схемные и микропрограммные БУО, можно сделать вывод, что последние уступают схемным БУО по быстродействию примерно на время обращения к ЗУ, т. е. периоды генерации сигналов микроопераций при схемном T_c и микропрограммном T_m управления связаны соотношением

$$T_m \approx t_0 + T_c,$$

где t_0 — время обращения к ЗУ. При небольшом числе операций в цифровой ЭВМ схемные БУО превосходят микропрограммные и по экономичности. Однако с ростом количества реализуемых микроалгоритмов и увеличением их сложности аппаратные затраты на схемные БУО растут быстрее, чем на микропрограммные БУО. Это объясняется тем, что увеличение сложности микроалгоритмов в схемных БУО приводит к введению новых логических и запоминающих элементов, а в микропрограммных БУО — к увеличению емкости памяти, то есть к увеличению количества значительно более простых элементов (особенно при использовании ПЗУ). Следовательно, существует такая сложность S_0 системы микроалгоритмов, после которой схемные БУО являются менее экономичными, чем микропрограммные. Значение S_0 оценивается примерно в 200 вершин графов микроалгоритмов и существенно зависит от соотношения стоимостей элементов в схемных и микропрограммных БУО.

Важнейшим преимуществом микропрограммного управления является его высокая гибкость, заключающаяся в том, что проектирование БУО, введение новых команд, внесение изменений в микроалгоритмы сводится к написанию новых или изменению существующих микропрограмм, с последующей их записью в ЗУ. При этом микропрограммное управление позволяет формировать систему команд цифровой ЭВМ, так, что она в максимальной степени будет учитывать специфику решаемых задач. Изменение состава команд не требует изменения состава технических средств машины. Кроме того, при микропрограммном управлении возможен программный доступ к любому узлу цифровой ЭВМ, что упрощает наладку и диагностику неисправностей таких цифровых ЭВМ. Структура БУО в этом случае проста и однородна и не зависит от характера решаемых задач. Интегральная технология производства основных узлов микропрограммных БУО, способствующая снижению их стоимости, а также то, что стоимость УУ составляет примерно 10 % от стоимости цифровой ЭВМ, обуславливают избыточную компенсацию недостатков микропрограммных БУО при малой сложности микроалгоритмов их преимуществами, связанными с гибкостью. Это привело к тому, что в цифровых ЭВМ третьего поколения основным способом реализации функций

управления стало микропрограммирование. Исключение составляют быстродействующие цифровые ЭВМ, преобладающим требованием при проектировании которых является высокая скорость выполнения операций. Со схемным управлением, например, построены цифровые ЭВМ ЕС-1050 и ЕС-1060, характеризующиеся высокой производительностью. Во всех цифровых ЭВМ малой и средней производительности серии ЕС (ЕС-1010, ЕС-1020, ЕС-1022 и др.) применен микропрограммный принцип управления.

В качестве примера рассмотрим микропрограммный БУО цифровой ЭВМ ЕС-1020, которая является одной из первых и наиболее распространенных машин третьего поколения.

Таблица 117

РМК	Поле	РМК	Поле
0—4	С	37—41	УО
5—8	А	42, 43	М
9—12	Ф	44	К
13—17	В	45—48	КС
18—20	ДВ	49—52	КМ
21—27	УП	53—58	АП
28—32	У	59—61	З
33—36	У1	62, 63	К

Микропрограммный БУО в цифровой ЭВМ ЕС-1020 (рис. 6.32) построен на основе ПЗУ емкостью 8К 64-разрядных слов. Отдельные

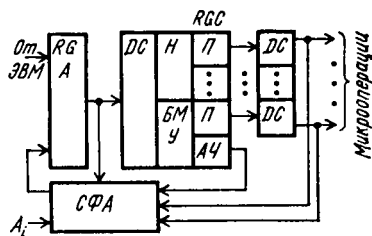


Рис. 6.32

поля микрокоманд (табл. 117) являются независимыми группами разрядов (РМК), которые управляют дешифраторами микроопераций. За один машинный такт выполняется только одна микрооперация из данного поля. Поля микрокоманд могут быть разбиты на 5 групп по назначению микроопераций: управление работой АЛУ, управление работой ЗУ, управление занесением информации на триггеры и регистры цифровой ЭВМ, формирование адреса следующей микрокоманды и организация контроля. Управление работой АЛУ осуществляют поля А, Ф, В и ДВ. Поля А и В определяют источники информации, подаваемой на регистры RG1 и RG2 АЛУ (всего регистров в АЛУ три). Источником информации может быть один из регистров цифровой ЭВМ или содержимое группы разрядов (от 44 до 52) данной микрокоманды. Поле Ф (функция) определяет микрооперацию, выполняемую в АЛУ. Поле ДВ (деформация В) определяет способ передачи слов с RG2 на обрабатывающие схемы АЛУ.

Поля управления памятью (УП) определяют регистр, содержимое которого заносится в RGA ОЗУ (разряды 21 — 23), операцию (чтение, запись, стирание) и тип памяти (основная, локальная, мультимплексная память и память блока защиты). Два последних поля содержат по два разряда.

Управление занесением информации на триггеры и регистры осуществляют поля У (установ) и С. Поле У задает номер триггера и тип установа (в 0 или в 1). Поле С определяет регистр, в который заносится информация с RG3 АЛУ.

Формированием адреса следующей микрокоманды управляют поля $Y1$, $Y0$, M , KC , KM и AP при участии некоторых сигналов микроопераций с ряда полей $МОЧ$. С помощью полей $Y1$ и $Y0$ осуществляется формирование двух младших разрядов (№ 0 и № 1) регистра RGA путем проверки логических условий, номера которых заданы в соответствующих полях, вследствие чего реализуется ветвление по четырем направлениям. При слове 00000 в поле $Y0$ (0000 в поле $Y1$) и слове 00001 в $Y0$ (0001 в $Y1$) происходит установка нулевого (первого по $Y1$) разряда RGA в 0 или 1, что и приводит к выполнению безусловного перехода. Поле M и разряды 57, 58 поля AP задают способ формирования адреса следующей микрокоманды из содержимого полей KC , KM и AP (по принципу принудительной адресации). Например, при $M = 00$ разряды микрокоманды 45—48 и 53—58 записываются в разряды 2 — 11 RGA , разряды 0 и 1 RGA формируются полями $Y0$ и $Y1$, а 12-й разряд RGA устанавливается в 0 или 1 соответствующими микрооперациями из поля $У$. При $M = 11$ и нулевом содержимом 57 и 58 разрядов осуществляется ветвление по 32 направлениям путем установки нулевого разряда в соответствующее состояние полем $Y0$ и передачи на RGA четырехразрядного слова с RGS , полученного в результате обработки значений логических условий в АЛУ.

Для организации контроля используется поле K , состоящее из разрядов 44, 62 и 63. Ряд разрядов (50 — 61) являются запасными (3). Связь между выходом RGA ПЗУ и входом $СФА$ (рис. 6.32) используется при прерываниях для запоминания адреса текущей микропрограммы, к которой необходимо вернуться после завершения работы микропрограммы, обрабатывающей прерывания.

Таким образом, в БУО цифровой ЭВМ ЕС-1020 использован комбинированный (вертикально-горизонтальный) формат однофазных микрокоманд с принудительным способом формирования адреса и присоединенной адресацией памяти микропрограмм с ветвлениями по 32 направлениям путем преобразования вектора значений логических условий в АЛУ. Аналогично построен БУО ЭВМ ЕС-1022.

В качестве второго примера рассмотрим БУО цифровой ЭВМ «Наири 3», особенностями которой являются: развитая система команд (всего 128); наличие реализованных на микропрограммном уровне эмуляторов (т. е. программных и технических средств, позволяющих данной цифровой ЭВМ выполнять программы, написанные для другой цифровой ЭВМ) машин «Наири-2», «Минск-22», «Минск-22М», что позволяет использовать накопленное программное обеспечение этих машин; перевод ряда традиционных функций аппаратно-программного обеспечения на уровень микропрограмм; возможность составления новых микропрограмм для оптимальной реализации алгоритмов каждого пользователя, что обеспечивает значительно более полное использование вычислительных ресурсов с выигрышем в быстродействии в 10—20 раз по сравнению с программной реализацией.

Реализация этих особенностей на микропрограммном уровне с традиционной структурой БУО и принудительной адресацией потребовала бы ПЗУ емкостью примерно 128К 270-разрядных слов (т. е. около $4 \cdot 10^7$ бит) и привела бы не только к большим затратам аппа-

[illegible]

На линейных участках микропрограмм в адресных полях указаны адреса ячеек ПЗУ M , в которых хранятся МОЧ очередных четырех микрокоманд. С помощью счетчика тактов СТТ по модулю 4, дешифратора тактов ДСТ и коммутатора K эти адреса поочередно передаются в ПЗУ M , где по ним выбираются МОЧ микрокоманд горизонтального типа. Выбранная МОЧ генерирует сигналы микроопераций, которые без каких-либо промежуточных усилителей и формирователей воздействуют на узлы цифровых ЭВМ. После четырех обращений к ПЗУ M и при отсутствии запрета от CP перенос со счетчика СТТ производит прибавление 1 к счетчику СТАМК и генерирует сигнал начала обращения (H) к ПЗУ A . Благодаря такой организации работы БУО за одно обращение к медленному ПЗУ A происходит четыре обращения к быстродействующему ПЗУ M , в течение которых выбираются и выполняются четыре микрокоманды. Так как в ПЗУ M 512 ячеек, а с ПЗУ A передается 8-разрядный адрес, то возникает необходимость в формировании девятого разряда адреса. Для этого введены микрооперации $У0$ и $У1$. Это удлиняет ячейки ПЗУ M на два разряда, однако сокращает длину ячеек ПЗУ A на 4 разряда, что дает значительную экономию общей емкости ЗУ микропрограмм.

Кроме естественной адресации СР с помощью комбинационной схемы *КС1* обеспечивают также различные режимы переходов (условные, безусловные переходы с возвратом, переход на ускоренное формирование сигналов микроопераций). При условном переходе (УП) адрес микрооперации опроса (МОО) логических условий *A* размещается на первой позиции в регистре слов ПЗУ *A*. Каждое логическое условие опрашивается своей МОО с помощью комбинационной схемы *КС2*, результат опроса фиксируется на триггере условного перехода ТУП. Адрес перехода располагают на полях 2 и 3 в *RGC*. Если условие не выполняется, то управление передается микрокоманде, адрес которой указан в поле 4 *RGC*. При выполнении условия адрес перехода передается на СТАМК.

При безусловном переходе (БП) адрес перехода размещается на полях 3 и 4 и из них передается на СТАМК.

Переход с возвратом (ПВ) позволяет многократно использовать в различных микропрограммах одинаковые последовательности микрокоманд, что существенно сжимает управляющую информацию. При ПВ в первых двух полях указывается адрес перехода *A1*, а в последних двух полях — адрес *A2* микрокоманды, после выполнения которой должен произойти возврат к исходной микропрограмме. По сигналу начала ПВ увеличенный на 1 адрес из СТАМК передается на регистр возврата *RGB*, адрес *A1* фиксируется на СТАМК, а адрес *A2* — на регистре *RGA2*. После окончания указанных передач БУО будет работать на новом участке микропрограммы до тех пор, пока адрес на СТАМК не станет равным *A2*. В этом случае схема равенства слов СРС обеспечит подключение *RGB* в СТАМК, вследствие чего произойдет возврат к исходной микропрограмме. При всех указанных переходах передача адреса в ПЗУ *M* блокируется, вследствие чего теряется несколько тактов ПЗУ *M*. Потери времени становятся особенно заметными в микропрограммах, содержащих значительное число проверок логических условий. Для устранения этого недостатка предусмотрен режим ускоренного выполнения микрокоманд, при котором адрес очередной микрокоманды формируется в шифраторе ПЗУ *M* при помощи восьми специально выделенных разрядов, образующих адресную часть микрокоманды. При появлении в СР кодовой комбинации, обеспечивающей переход к ускоренному выполнению (УВ) микрокоманд, триггер ТУВ устанавливается в 1, подача СС на ПЗУ *A* прекращается и адреса микрокоманд считываются с адресной части шифратора. Возврат к обычному выполнению микрокоманд осуществляется специальной микрооперацией, устанавливающей триггер ТУВ в 0.

Приведенные примеры еще раз свидетельствуют о большом разнообразии реализаций микропрограммирования, которое стало основным методом управления вычислительным процессом в цифровых ЭВМ.

6.4. СПОСОБЫ ПОСТРОЕНИЯ БЛОКОВ УПРАВЛЕНИЯ КОМАНДАМИ

В зависимости от того, каким образом формируется адрес следующей команды, все команды в цифровых ЭВМ можно разбить на три типа. К командам первого типа относятся такие, которые не нару-

шают естественного порядка выполнения команд (например, трех-адресная команда на рис. 6.1, б). После выполнения такой команды, записанной в i -ю ячейку ЗУ, переходят к выполнению команды, находящейся в ячейке $i + 1$. Однако в некоторых случаях необходимо нарушить естественный порядок выполнения команд. Для этого существуют команды второго типа, т. е. команды безусловного перехода. Такая команда означает: перейти к выполнению команды, адрес которой указан в данной команде. В трехадресной команде безусловного перехода первых два адреса не используются. Часто естественный порядок следования команд необходимо нарушить только при выполнении некоторого условия. Для этого используются команды третьего типа — команды условного перехода. Такая команда озна-

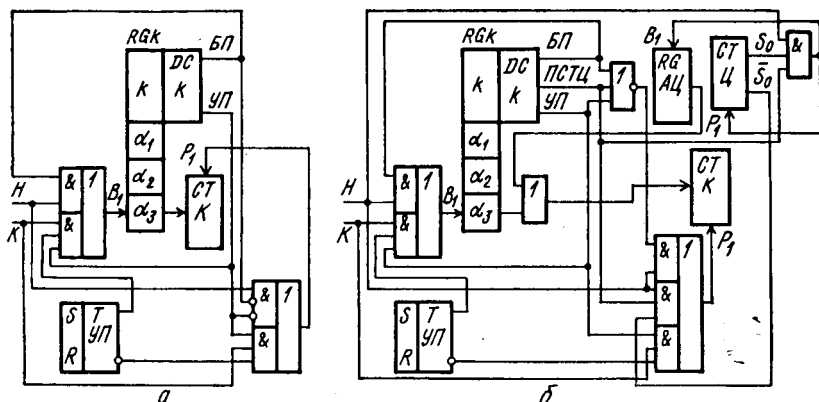


Рис. 6.34

чает: взять числа из ячеек α_1 и α_2 , а затем проверить, выполняется ли условие, код которого k . Если условие выполняется, то перейти к выполнению команды, записанной в ячейке α_3 . Если же условие не выполняется, то перейти к выполнению команды, записанной в следующей $i + 1$ -й ячейке.

Рассмотрим простейшую схему, осуществляющую формирование адреса следующей команды (рис. 6.34, а). В схеме имеется регистр команд (RGK), счетчик команд (СТК), дешифратор операций (ДСК), триггер условия перехода (ТУП) и ряд элементов И и ИЛИ. Если выполняется команда безусловного (БП) или условного (УП) перехода и условие перехода выполнено (триггер ТУП находится в единичном состоянии), то адрес следующей команды формируется путем записи в СТК адреса, указанного в выполняемой команде. В остальных случаях адрес следующей команды формируется путем добавления 1 к содержимому СТК. При выполнении команды типа УП адрес следующей команды формируется по сигналу конца операции (К), так как этот адрес зависит от того, выполнено или нет условие перехода. В остальных случаях формирование адреса следующей команды можно совместить с выполнением команды, то есть осуществлять формирование по сигналу начала операции (Н).

При решении задач на цифровых ЭВМ очень часто алгоритм решения может быть представлен в виде ряда повторяющихся серий операций. Такие вычислительные процессы называются циклическими, а повторяющиеся в них участки — циклами. Возврат к первой команде цикла можно осуществлять с помощью команды условного перехода. Однако, при этом удлиняется цикл, что при коротких циклах значительно замедляет их выполнение. Поэтому были разработаны другие более эффективные способы выполнения циклических программ. Один из них состоит в использовании счетчика циклов (СТЦ). При подготовке перехода к циклической программе в СТЦ записывается начальное значение — n , где n — число циклов. После выполнения очередного цикла к содержимому счетчика прибавляется 1 и проверяется знак S_0 суммы, равной $r - n$, где r — количество выполнен-

Таблица 118

j	$\times$	α_1	α_2	ω
$j+1$	Π	α_1	α_2	ω
$j+2$	$+$	α_1	α_2	α_1
$j+3$	$\leq$	α_1	α_3	j
$j+4$	ОСТ	—	—	—

Таблица 119

j	ПЦ	—	—	α_4
$j+1$	$\times$	α_1	α_1	ω
$j+2$	Π	—	—	ω
$j+3$	$+(ПСТЦ)$	α_1	α_2	α_1
$j+4$	ОСТ	—	—	—

ных циклов. При отрицательном знаке суммы (т. е. $S_0 = 1$) осуществляется переход к началу цикла, а при положительном знаке ($S_0 = 0$) — к команде, записанной после циклического участка программы. Последняя команда цикла определяется по специальному признаку. Например, в операционной части команды выделяется специальный разряд, 1 в котором указывает на конец цикла. В некоторых машинах в конце цикла ставится специальная команда перехода по знаку счетчика циклов (ПСТЦ).

Схема формирования адреса следующей команды с использованием СТЦ приведена на рис. 6.34, б. От предыдущей схемы она отличается наличием СТЦ и регистра адреса цикла $RGAC$. Этот регистр служит для запоминания адреса команды цикла. Если в цифровой ЭВМ имеется команда ПСТЦ, то адрес первой команды цикла указывается в адресной части команды перехода, а в качестве $RGAC$ может использоваться RGk .

Оценим эффективность введения СТЦ и $RGAC$ на примере вычисления таблицы квадратов чисел $(a + ic)^2$, где $i = 0, 1, \dots, n$. Будем считать, что число a находится в ячейке с адресом α_1 ; число c — в ячейке α_2 ; число $a + nc$ — в ячейке α_3 . Тогда для трехадресной цифровой ЭВМ без счетчика циклов программу вычислений можно задать табл. 118, в которой символы $\times$, Π , $+$, $\leq$ и ОСТ соответствуют командам умножения, печати, сложения, сравнения и останова машины.

Для вычислений по этой программе таблицы квадратов чисел необходимо выполнить $n + 1$ цикл, в каждом из которых 4 операции,

и 13 раз обратиться к ЗУ. Таким образом, время выполнения этой программы составит

$$t_1 = (n + 1) (t_x + t_n + t_+ + t_< + 13t_0),$$

где t_x , t_n , t_+ , $t_<$ — длительность соответствующих операций.

При использовании СТЦ и RGAЦ программу вычислений можно представить в табл. 119, где ПЦ — команда подготовки цикла, по которой в СТЦ из ячейки α_4 записывается $-n$, а + (ПСТЦ) — команда сложения с переходом к следующей команде по знаку содержимого СТЦ. Для выполнения одного цикла в этом случае необходимо время

$$t_x + t_n + t_{+(ПСТЦ)} + 10t_0,$$

а для выполнения $n + 1$ циклов

$$t_2 = (n + 1)(t_x + t_n + t_{+(ПСТЦ)} + 10t_0) + t_{ПЦ} + 2t_0.$$

Следовательно, время выполнения циклической программы во втором случае сократится на

$$t_1 - t_2 \approx (n + 1) (t_< + 3t_0),$$

т. е. на время выполнения одной трехадресной команды, умноженное на число циклов. Определим, как при этом повышается производительность цифровой ЭВМ. Пусть n_1 — среднее число команд нециклических участков программ, приходящееся на один цикл, а n_2 — среднее число команд в цикле. Тогда время выполнения программы на цифровой ЭВМ без СТЦ будет равно

$$t_{n_1} = n_1(t_{АЛУ} + \mu t_0) + n_2(t_{АЛУ} + \mu t_0) = (n_1 + n_2)(t_{АЛУ} + \mu t_0),$$

где $t_{АЛУ}$ — средняя длительность операций на АЛУ; μ — среднее число обращений к ЗУ при выполнении одной команды. Для цифровой ЭВМ со СТЦ и с RGAЦ получим

$$\begin{aligned} t_{n_2} &= n_1(t_{АЛУ} + \mu t_0) + (n_2 - \eta)(t_{АЛУ} + \mu t_0) = \\ &= (n_1 + n_2 - \eta)(t_{АЛУ} + \mu t_0), \end{aligned}$$

где η — среднее число команд, на которое уменьшается длина цикла. Тогда относительное сокращение времени выполнения программ составит

$$\frac{t_{n_1} - t_{n_2}}{t_{n_1}} = \frac{\eta}{n_1 + n_2}$$

при $1 \leq \eta \leq 2$; $n_1 \approx n_2 = 3 - 10$ дает 5—33 %. При этом аппаратные затраты увеличиваются на один счетчик и на один регистр. Так как эти затраты составляют значительно меньшую долю, чем 5—33 % от общих затрат на оборудование цифровой ЭВМ, то введение СТЦ является целесообразным и эффективным способом повышения производительности цифровой ЭВМ. Заметим, что для подтверждения этого вывода необходимо иметь более точные данные о величинах η , n_1 , n_2 , которые могут быть получены путем анализа программ, подлежащих выполнению на проектируемой цифровой ЭВМ.

Адреса, по которым происходит обращение к ЗУ, могут указываться непосредственно в адресной части команды, либо же формироваться с помощью схем адресации путем суммирования адреса, записанного в команде, со словом, записанным в специальном индексном регистре. В каждой команде имеется признак, указывающий на то, каким способом получают исполнительные адреса. Особенно эффективным способом является применение индексных регистров в циклических программах с переадресацией (переадресацией называется операция автоматического изменения адресов операндов). Пусть, например, необходимо вычислить сумму

$$S_n = \sum_{i=1}^n a_i.$$

Таблица 120

j	+	$\alpha+1$	$\alpha+2$	$\alpha+1$
$j+1$	+	β_1	β_2	β_1
$j+2$	+	j	γ	j
$j+3$	$\leq$	β_1	β_3	j
$j+4$	ОСТ	—	—	—

Таблица 121

j	ПЦ + (И)	—	—	β_3
$j+1$	+	$\alpha+1$	$\alpha+2$	$\alpha+1$
$j+2$	КЦ	—	—	$j+1$
$j+3$	ОСТ	—	—	—

Числа a_i находятся в ячейках с адресами $\alpha + i$, а в ячейках β_1 и β_2 записаны 1, в ячейке β_3 записано число n , а в ячейке γ хранится команда, имеющая во всех своих частях нули, за исключением второго адреса, в младшем разряде которого записана 1. Программа для вычисления S_n в цифровой ЭВМ без индексных регистров и СТЦ представлена табл. 120. В этой программе команда $j+1$ совместно с ячейкой β_1 выполняет функции счетчика циклов. Команда $j+2$ осуществляет переадресацию (ко второму адресу команды j прибавляется 1). Возврат к первой команде цикла осуществляется командой условного перехода при неравенстве чисел в ячейках β_1 и β_3 .

Использование счетчика циклов и индексного регистра делает ненужными команды $j+1$, $j+2$ и $j+3$ (табл. 121). При выполнении $j+1$ -й команды + (И) второй адрес определяется путем суммирования адреса, указанного в команде, с содержимым индексного регистра, а при выполнении $i+2$ -й команды КЦ (конец цикла) к содержимому индексного регистра и к слову на счетчике циклов прибавляется 1. Переход к началу цикла осуществляется этой же командой по знаку счетчика циклов. Из изложенного видно, что общее число команд в программе вычисления суммы S_n сократилось на 1, а число команд в цикле сократилось на 2. Количество обращений к ЗУ при выполнении одного цикла сократилось с 15 до 5, вследствие чего время выполнения одного цикла уменьшилось почти в три раза. При наличии регистра адреса цикла команду $j+2$ из программы можно исключить, возложив ее функции на команду $j+1$.

Таким образом, использование индексных регистров и СТЦ позволяет устранить команды переадресации и программную реализацию счетчика циклов, а также совместить переход к началу цикла

с выполнением последней команды цикла (т. е. устранить команду условного перехода). Отсюда без дополнительного количественного анализа видна высокая эффективность и целесообразность введения индексных регистров.

Еще одним эффективным способом выполнения циклических участков программ является введение так называемых групповых операций. Групповая операция в цифровой ЭВМ задается одной командой, но выполняется многократно над множеством операндов. Возможная схема управления групповой операцией в трехадресной машине приведена на рис. 6.35. Подготовка такой операции осуществляется специальной командой начала групповой операции (НГО), в результате выполнения которой в счетчик групповой операции (СТГО) заносится со знаком минус количество N_i циклов в групповой операции, триггер групповой операции (ТГО) устанавливается в 1 и в ре-

Таблица 122

i	НГО	γ_1	γ_2	—
$i+1$	+	$\alpha+1$	$\alpha+2$	$\alpha+1$
$i+2$	ОСТ	—	—	—

гистр групповой операции (RGFO) заносится слово A_i , определяющее, какие адреса должны изменяться при выполнении групповой операции. После этого в RGk записывается команда групповой операции.

Адресная часть RGk выполняется в виде трех счетчиков. Ввод в RGk новой команды (ВНК) осуществляется путем изменения на 1 адресов α_1 , α_2 и α_3 . Конец групповой операции формируется СТГО. Сигнал переноса p с этого счетчика устанавливается ТГО в нуль и добавляет 1 к содержимому СТК, в результате чего осуществляется переход к следующей команде. Таким образом, введение групповой операции, так же как и введение индексных регистров и счетчика циклов, позволяет устранить команды переадресации, программную реализацию счетчика циклов и команду условного перехода к началу цикла.

Для оценки эффективности введения групповой операции рассмотрим программу вычисления S_n (табл. 120). При вычислениях по этой программе необходимо $15n + 1$ раз обратиться к ЗУ и выполнить $4n + 1$ команду. С использованием же групповой операции программу для S_n можно записать так, как показано в табл. 122 (в ячейке γ_1 записано число $N_i = -n$, а в ячейке $\gamma_2 - A_i = 010$). Вычисления по этой программе требуют $3n + 4$ обращений к ЗУ и выполнения $n + 2$ команд.

В общем случае время выполнения групповой операции на цифровой ЭВМ, не имеющей соответствующей команды равно

$$t_1 = n\lambda(t_{АЛУ} + \mu t_0).$$

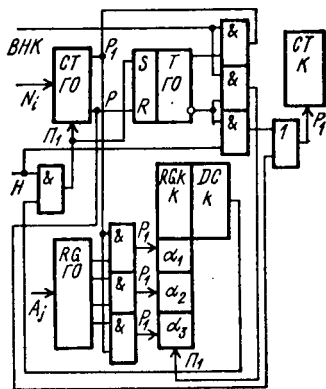


Рис. 6.35

где λ — среднее число команд, необходимых для реализации групповой операции. Обычно $\lambda = 4$, так как необходимо выполнить саму операцию, осуществить переадресацию, прибавить 1 к программно реализуемому СТС и перейти к началу цикла. При наличии команды групповой операции время ее выполнения будет равно

$$t_2 = (n + 1)(t_{A\mathcal{L}Y} + \mu t_0) \approx n(t_{A\mathcal{L}Y} + \mu t_0).$$

Если W — частота появления групповой операции, то производительность цифровой ЭВМ возрастает от введения команды групповой операции в

$$\frac{(1 - W) (t_{A/Y} + \mu t_0) + W \lambda (t_{A/Y} + \mu t_0)}{(1 - W) (t_{A/Y} + \mu t_0) + W (t_{A/Y} + \mu t_0)}$$

раз (даже при $W = 0,1$ это отношение равно 1,3).

Очень часто в программах имеются отдельные участки, которые встречаются во многих программах. Такие участки получили назва-

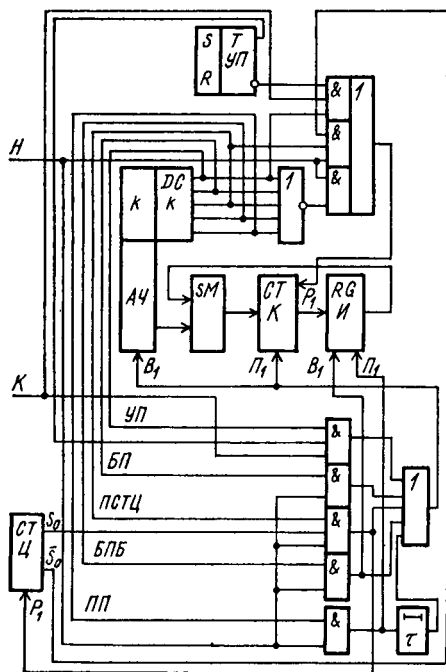


Рис. 6.36

ние стандартных подпрограмм. Во многих машинах имеется целый набор стандартных подпрограмм, записанных в ПЗУ. В процессе решения задач к этим подпрограммам приходится обращаться и тем самым нарушать естественный порядок следования команд. Переход от основной программы обычно осуществляется командой безусловного перехода, в которой указывается адрес первой команды подпрограммы. При обратном переходе от подпрограммы к основной программе в команде перехода должен быть указан адрес первой команды того участка программы, к которому осуществляется переход. Этот адрес на 1 больше адреса команды, осуществившей переход к подпрограмме. Наиболее просто этот переход осуществляется при наличии индексных регистров. В один из индексных регистров записывается адрес команды,

осуществившей переход к подпрограмме, а в адресной части последней команды подпрограммы записывается 1 и признак переадресации по индексному регистру. Адрес команды, следующей за подпрограммой, формируется сложением адресной части последней команды подпрограммы и содержимого индексного регистра.

Составим теперь более полный вариант схемы БУК (рис. 6.36). Дешифратор операций (ДСК) формирует сигналы: a — переход по

счетчику циклов (ПСТЦ); b — условный переход (УП); v — безусловный переход с заданием в команде исполнительного адреса (БП); z — безусловный переход с заданием в команде базисного адреса (БПБ), например, для возврата к основной программе; d — переход к подпрограмме (ПП), т. е. БП, перед выполнением которого осуществляется запись содержимого счетчика команд (СТК) в индексный регистр (RGI). При переходе от подпрограммы на RGI подается сигнал B_1 и этот же сигнал поступает на RGk . В результате в СТК запишется адрес первой команды соответствующего участка основной программы.

6.5. ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ОДНОПРОЦЕССОРНЫХ ЦИФРОВЫХ ЭВМ

Для достижения наибольшей производительности однопроцессорных ЦЭВМ необходимо организовать такое управление командами, при котором минимизируются простои процессора. Рассмотрим некоторые возможности совершенствования структуры БУК, направленные на достижение указанной цели.

Один из эффективных методов повышения производительности ЦЭВМ состоит во введении операций над словами переменной длины. При работе с такими операциями естественным является введение переменных форматов и для команд, что способствует уплотнению информации, сокращает время, необходимое для пересылки ее между уровнями памяти и создает возможность выборки нескольких команд за один цикл памяти. При трехадресном формате в командах, задающих одноместные операции X^2 , $\sqrt{X}$, $\frac{1}{X}$, $\sin X$ и других используется только два адреса, а в команде безусловного перехода — один адрес. В этих случаях длину команд можно сократить за счет исключения неиспользуемых частей адресов. Например, при длине условного адреса в 12 бит и 64 операциях, реализуемых командами, получим следующие наборы длины команд:

3 адреса — $6 + 3 \cdot 12 = 42$ бит,

2 адреса — $6 + 2 \cdot 12 = 30$ бит,

1 адрес — $6 + 1 \cdot 12 = 18$ бит.

Следовательно, при длине ячейки ЗУ, равной 64 разрядам, за одно обращение к памяти можно выбирать либо одну трехадресную команду, либо 2 двухадресные команды, либо три одноадресные команды. При 64-разрядной ячейке ЗУ, которое реализовано, например, цифровой ЭВМ ЕС-1050, за одно обращение к памяти можно выбирать 4 команды типа RR , 2 команды типа RX и одну полную и половину следующей команды типа SS .

При выборке слов переменной длины из памяти необходимо указать адрес первого символа слова и длину слова. Объем указываемой в адресе информации при выборке команд переменной длины можно сократить за счет того, что код операции часто содержит информацию о длине команд. Например, в машинах серии ЕС два старших разряда кода операции являются условным обозначением формата команды и ее длины. Поэтому для выборки команд переменной длины доста-

вается (ЗР) по третьему адресу адресной части команды. Для этого формируется сигнал B_{33} передачи третьего адреса в RGA ЗУ. В это же время в БУК формируется адрес следующей команды (ФАСК). На этом цикл заканчивается и по сигналу B_{30} начинается следующий цикл.

Временные диаграммы работы цифровой ЭВМ при выполнении команд условного и безусловного переходов показаны на рис. 6.38, б и 6.38, в.

Для одно- и двухадресных цифровых ЭВМ временные диаграммы несколько отличаются от рассмотренных выше. На рис. 6.38, г приведена диаграмма работы одноадресной цифровой ЭВМ при выпол-

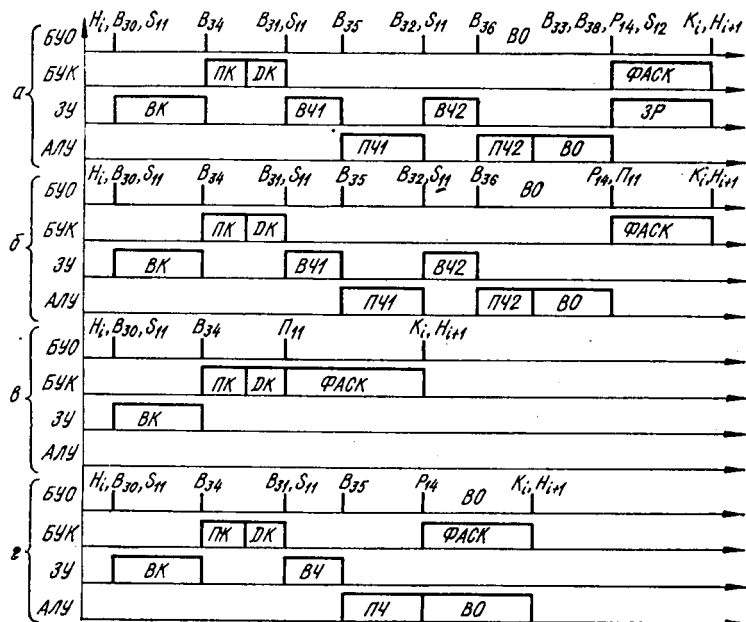


Рис. 6.38

нении команды первого типа. Так как результат операции в этом случае остается в регистре АЛУ, то цикл здесь оканчивается после выполнения операции в АЛУ. Из диаграмм видно, что цифровая ЭВМ имеет резерв времени для повышения ее производительности путем совмещения операций, выполняемых отдельными устройствами.

В цифровой ЭВМ, построенной по классической схеме, отдельные устройства (АЛУ, ЗУ, УУ, внешние устройства) работают последовательно друг за другом. Это приводит к тому, что эти устройства заняты полезной работой лишь малую часть времени. Путем некоторого усложнения машины можно добиться параллельной работы всех устройств, что существенно повышает производительность машины в целом. Сущность такого усложнения состоит в следующем.

ЗУ команд должно быть отделено от ЗУ чисел с тем, чтобы в процессе исполнения команды информация распространялась в одном направлении. Это достигается путем секционирования ЗУ и снабжения

каждой секции своими регистрами адреса и числа. Например, типовой состав оборудования ЗУ в цифровой ЭВМ «Стреч» включает 6 блоков. Первые 4 блока предназначены в основном для хранения чисел. Ячейки в них пронумерованы так, что любые 4 последовательных (соседних) адреса относятся к различным блокам. При этом номер блока задается двумя соседними разрядами адреса. Пятый и шестой блоки предназначены для хранения программ.

Ячейки здесь нумеруются так, что две соседних относятся к разным блокам, причем номер первой ячейки пятого блока равен $N+1$, где N — номер последней ячейки первой части ЗУ. Размещение информации в таком ЗУ показано в табл. 123.

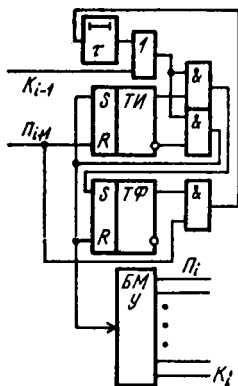


Рис. 6.39

Таблица 123

Блоки ЗУ					
1	2	3	4	5	6
0	1	2	3	$N+1$	$N+2$
4	5	6	7	$N+3$	$N+4$
⋮	⋮	⋮	⋮	⋮	⋮
$N-3$	$N-2$	$N-1$	N	⋯	⋯
ЗУ чисел				ЗУ программ	

Кроме секционирования ЗУ, параллельная работа всех устройств цифровой ЭВМ требует наличия в каждом из них входного и выходного буферных регистров. Каждое устройство также должно иметь блок местного управления со своим регистром кода операции, триггером-индикатором (ТИ) занятости устройства и триггером-фиксатором (ТФ) запроса к устройству. Схема, иллюстрирующая взаимодействие перечисленного дополнительного оборудования, приведена на рис. 6.39. Здесь i -е устройство включается в работу сигналом K_{i-1} , вырабатываемым предыдущим ($i-1$)-м устройством. Если i -е устройство свободно (триггер ТИ в 0), то оно включается в работу немедленно. При этом ТИ устанавливается в 1, а БМУ выдает сигнал приема слов от $i-1$ -го устройства P_i , по которому из $i-1$ -го устройства передаются операнды и код операции. Этим же сигналом триггер ТИ $i-1$ -го устройства устанавливается в 0. Далее происходит выработка необходимой последовательности управляющих сигналов БМУ и выполнение операции. После ее завершения вырабатывается сигнал K_i , поступающий на $i+1$ -е устройство. Если в момент прихода запроса K_{i-1} , i -е устройство занято (триггер ТИ в 1), то запрос фиксируется триггером ТФ и после освобождения i -го устройства сигналом P_{i+1} оно вновь запускается в работу. Принцип работы таких машин можно пояснить табл. 124, где показано, выполнение и обработка каких команд происходит в различных устройствах ма-

Номер цикла	Счетчик команд	ЗУ команд	АЛУ команд	ЗУ чисел	АЛУ чисел
m	Команда j	Команда $j-1$	Команда $j-2$	Команда $j-3$	Команда $j-4$
$m+1$	Команда $j+1$	Команда j	Команда $j-1$	Команда $j-2$	Команда $j-3$
$m+2$	Команда $j+2$	Команда $j+1$	Команда j	Команда $j-1$	Команда $j-2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m+l$	Команда $j+l$	Команда $j+l-1$	Команда $j+l-2$	Команда $j+l-3$	Команда $j+l-4$

шины. Термин «АЛУ команд» объединяет все узлы БУК, участвующие в формировании исполнительных адресов. Временная диаграмма работы цифровой ЭВМ при таком совмещении операций показана на рис. 6.40, а. Здесь промежутки времени ΦA_i соответствуют формированию адреса i -й команды; ПК — приему и преобразованию адрес-

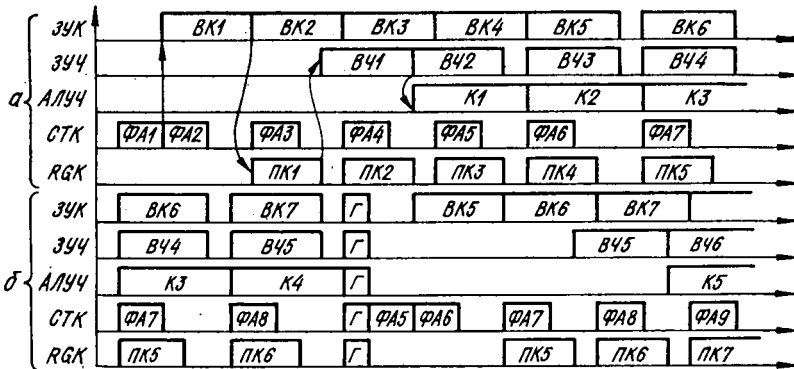


Рис. 6.40

ной части i -й команды на RGK в БУК; K_i — выполнению i -й команды в АЛУ чисел, а стрелками указана последовательность работы отдельных блоков и устройств при выполнении команды K_1 . Некоторые затруднения возникают при появлении команд условного перехода, так как формирование адреса следующей команды можно начинать только после выполнения указанной операции. Однако и в этом случае за счет некоторого усложнения БУК можно совмещать работу отдельных устройств. Можно считать, например, что один из вариантов продолжения программы более вероятен (например, продолжение цикла всегда более вероятно, чем его окончание), поэтому не дожидаясь проверки условия перехода можно выполнять более вероятную ветвь программы. В случае, если после выполнения команды условного перехода возникла необходимость изменить порядок следования

команд, то ранее сформированный адрес следующей команды гасится (Г), а управление передается вновь сформированной команде. Эта ситуация иллюстрирована диаграммой на рис. 6.40, б, где показано, как команда К4 изменяет естественное следование команд и повторно формируется адрес команды К5.

Для осуществления этого БУК должен содержать два счетчика команд и иметь возможность передавать управление с одного из них на другой (рис. 6.41). На линейных участках программы счетчики команд СТК1 и СТК2 работают синхронно, например, с целью взаимного контроля. Сигнал начала операции условного перехода (НУП) передает управление СТК1, куда записывается адрес из *RGK*, т. е. условие перехода априори считают выполненным. При этом на СТК2 формируется и хранится адрес того участка программы, к которому

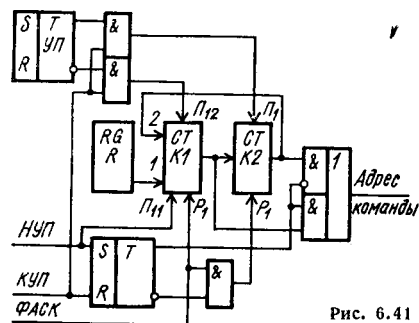


Рис. 6.41

необходимо перейти в случае, если условие перехода не будет выполнено. Сигнал конца операции условного перехода (КУП) опрашивает состояние триггера ТУП, уравнивает содержимое СТК1 и СТК2 путем передачи информации из СТК1 в СТК2 при единичном состоянии ТУП, а также передает управление СТК2. Аналогично может быть построена схема при предположении, что более вероятным является событие, состоящее в невыполнении условия перехода.

Для оценки эффективности совмещения операций предположим, что времена считывания команд и чисел, преобразования команд, выполнения коротких операций в АЛУ равны друг другу и равны τ . Тогда время выполнения многотактной операции t_m в АЛУ чисел можно представить как $t_m = n\tau$. Пусть W — частота появления многотактной операции. В этом случае среднее время выполнения одной команды в цифровой ЭВМ без совмещения операций составит

$$t_1 = t_{зук} + t_{алук} + t_{зуч} + Wt_m + (1 - W)\tau = (4 + (n - 1)W)\tau.$$

Как видно из диаграмм на рис. 6.40, время прохождения команды через все устройства в цифровой ЭВМ с совмещением операций такое же, как и в цифровой ЭВМ без совмещения операций. Однако за счет параллельной работы отдельных устройств через каждые $t_{\max}$ единиц времени, где $t_{\max}$ — время выполнения одной операции в наиболее медленном устройстве, будет завершаться выполнение одной команды. Следовательно, среднее время выполнения одной команды t_2 в такой цифровой ЭВМ составит

$$t_2 = (t_{зук} + t_{алук} + t_{зуч} + \tau)V + t_mW + \tau(1 - W - V) = (1 + 3V + (n - 1)W)\tau,$$

где V — частота появления команд, изменяющих естественный порядок их выполнения (предполагаем, что все команды такого типа одно-

тактные). Поэтому

$$v = \frac{t_1}{t_2} = \frac{4 + (n-1)W}{1 + 3V + (n-1)W},$$

при $V = 0$, $n = 1$, $v = v_{\max} = 4$, а при $V = 1$ или $n = \infty$ $v = v_{\min} = 1$, т. е. $1 \leq v \leq 4$. Следовательно, совместная параллельная работа четырех устройств цифровой ЭВМ при благоприятных условиях позволяет в 4 раза повысить производительность. В общем случае при параллельной работе m устройств производительность может возрасти в m раз. Цикл цифровой ЭВМ при этом разбивается на m частей и каждая часть цикла выполняется отдельным устройством. Приемами, позволяющими увеличить число m параллельно и независимо работающих устройств, являются расслоение памяти, разделение памяти на ЗУК и ЗУЧ, применение АЛУ блочного типа и другие.

Рассмотренный способ совмещения операций позволяет значительно сократить простои отдельных устройств. Простои будут минимальны, если все устройства будут иметь одинаковую производительность. В процессоре это условие нетрудно выполнить, однако согласовать работу устройств ввода и вывода с процессором из-за большой разницы в быстродействии (соответственно $10 - 10^2$ и $10^5 - 10^7$ операций в секунду) практически невозможно. Для частичной компенсации этого недостатка современные цифровые ЭВМ снабжены несколькими (до 10 и более) устройствами ввода-вывода, которые могут работать параллельно и независимо.

Однако встречаются программы, у которых на несколько вычислительных операций приходится одна операция обращения к устройствам ввода-вывода. Производительность машины при работе с такими программами падает с $10^5 - 10^7$ до $10 - 10^2$ операций в секунду. Для устранения этого недостатка вводят операцию прерывания совместно с мультипрограммным режимом работы цифровой ЭВМ (такой режим, при котором машина ведет вычисления одновременно по нескольким программам). При этом, как правило, осуществляется совмещение операций на трех уровнях, а именно:

- 1) параллельная работа различных блоков и устройств цифровой ЭВМ в процессе выполнения одной команды;
- 2) совмещение подготовки и выполнения нескольких последовательных команд одной программы;
- 3) совмещение работ различных устройств цифровой ЭВМ по разным и независимым программам.

Для достижения такого совмещения необходим комплекс программных и аппаратных средств. Программные средства при этом обычно обеспечивают планирование очередности выполнения программ в зависимости от их важности (приоритета), распределяют память и координируют выполнение отдельных программ во времени, контролируют ход вычислений и обнаруживают неисправности.

С помощью же аппаратных средств обеспечивается автономная работа отдельных устройств цифровой ЭВМ, прерывание выполнения текущей программы, запоминание информации о прерывании для последующего возврата к прерванной программе, выбор другой программы и переключение машины на ее выполнение, ограждение со-

вместно выполняемых программ от влияния их друг на друга, доступ к цифровой ЭВМ большого числа пользователей.

При работе цифровой ЭВМ в мультипрограммном режиме все устройства машины имеют свои блоки местного управления, позволяющие им автономно выполнять команды после получения из ЗУ соответствующей информации. Занятость каждого из устройств фиксируется специальным триггером-индикатором. Кроме того, в ЗУ хранится специальная служебная программа-диспетчер (супервизор, монитор). Эта программа просматривает все имеющиеся в ЗУ программы и определяет оптимальную последовательность их выполнения.

Если, например, требуется АЛУ свободно, то ему передается номер первой команды программы, подлежащей выполнению. Вслед за этим АЛУ включается в работу. Если устройство занято, то поступающие запросы образуют очередь к устройству.

При необходимости обращения к внешним устройствам выполнение программы на АЛУ прерывается и ей передается управление соответствующими внешними устройствами. В это же время программа-диспетчер готовит АЛУ к выполнению второй программы. В составе программы-диспетчера имеются команды передачи в стандартные ячейки ЗУ содержимого регистров АЛУ, индексных регистров, счетчика циклов, триггера условия перехода и другой информации,

необходимой для возврата в дальнейшем к прерванной программе. После запоминания указанной информации программа-диспетчер передает управление следующей программе, т. е. формирует адрес первой команды программы.

В процессе совместной работы АЛУ и внешних устройств необходимо обращаться к ОЗУ. При этом запросы к ОЗУ могут совпадать. Поэтому в устройстве управления ЗУ имеется специальная схема приоритета, которая определяет последовательность обслуживания запросов и переключает соответствующие каналы (рис. 6.42). Запросы $31, 32, \dots, 3n$ от отдельных устройств поступают на регистр приоритета $RGП$, где каждому устройству соответствует свой триггер. Формирование сигналов исполнения запроса $ИЗ\ i$ осуществляется элементами И, управляемыми триггерами $RGП$. Сигналы $ИЗ\ i$ устанавливают триггер управления $ТУ$ в 1, фиксируя тем самым состояния триггеров на время выполнения запроса. Занумерованы эти триггеры в порядке убывания приоритета. Сигналы конца операции обращения i -го устройства к ОЗУ ($КО\ i$) устанавливают $ТУ$ и триггеры $RGП$ в «0», освобождая тем самым ОЗУ. Наибольший приоритет имеют наиболее медленные устройства. С увеличением быстродействия приоритет устройства убывает. Если бы наиболее быстродействующее

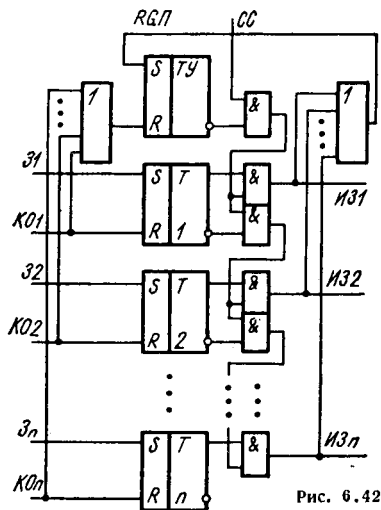


Рис. 6.42

устройство обладало наибольшим приоритетом, то могли бы возникать ситуации, при которых медленные устройства простаивали из-за невозможности обращения к ОЗУ, так как ОЗУ все время обслуживало быстрые устройства. После выполнения запроса некоторого устройства следует период его автономной работы, во время которого ОЗУ может быть связано с другими устройствами.

Для уменьшения вероятности наложения запросов к ОЗУ его можно секционировать и организовать одновременный доступ к отдельным секциям. В этом случае совпадением запросов будет считаться одновременное обращение нескольких устройств не к ОЗУ, а к одной и той же секции ОЗУ.

Существует большое число причин, вызывающих необходимость прерывания выполнения программ. Эти причины можно разделить

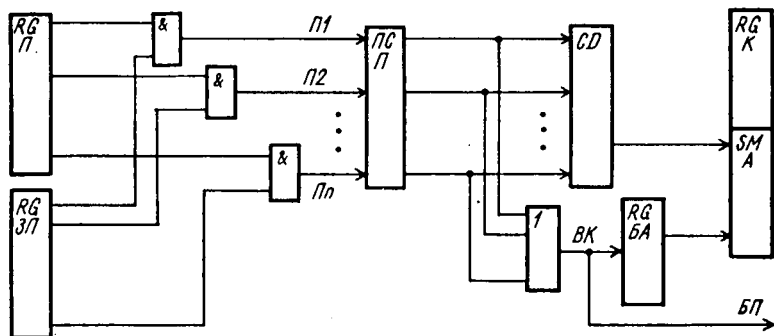


Рис. 6.43

на две группы. Первая группа объединяет причины, возникающие в ходе выполнения программ: переполнение разрядной сетки, обнаружение ошибки схемами контроля, необходимость обращения к внешним устройствам или к программе-диспетчеру, появление заведомо неправильных команд, адресов, операндов и т. п. Необходимость прерывания по таким причинам может возникнуть в любой момент времени. При наличии сигнала о таком прерывании должна быть включена программа, устраняющая причину прерывания.

Вторая группа объединяет причины, не зависящие от выполняемых программ: запросы от внешних устройств, поступление информации от другой цифровой ЭВМ, команды с пульта управления и др. При мультипрограммной организации работы цифровой ЭВМ прерывание, приводящее к включению новой программы, должно сопровождаться запоминанием информации, дающей возможность вернуться к продолжению прерванной программы.

Наличие или отсутствие каждой из возможных причин прерывания фиксируется на одном из триггеров регистра прерывания *RGП* (рис. 6.43). Допустимость или недопустимость прерывания по данной причине определяется словом (маской) в регистре защиты от прерывания *RGЗП* (регистре масок). Каждой прерывающей программе соответствует своя маска. Вызов маски из памяти и ее засылка в *RGЗП* производится в начале выполнения программы. Маска может формироваться программным путем в процессе выполнения самой

программы и свидетельствовать, например, о получении результатов, готовых к выдаче. Сигналы прерывания P_i получают с помощью элементов И при наличии причины прерывания и отсутствия запрета на прерывание по данной причине. Для выборки одного из нескольких появившихся одновременно сигналов P_i имеется приоритетная схема прерывания ПСП. Сигнал с i -го выхода ПСП должен сформировать команду безусловного перехода БП к программе-диспетчеру. Для этого необходимо передать содержимое счетчика команд в некоторую стандартную ячейку и сформировать адрес первой команды i -го участка программы-диспетчера. Указанный адрес получается путем суммирования на сумматоре адреса SMA базисного адреса прерывания, находящегося в регистре $RGBA$ с номером источника прерывания, который формируется шифратором CD .

Информацию о состоянии прерванной программы называют словом состояния программ (ССП). ССП хранятся в фиксированных ячейках ЗУ. В простейшем случае ССП должно содержать маску прерываний, код причины прерывания, признаки состояний блоков процессора в момент прерывания, содержимое счетчика команд, указывающее адрес следующей невыполненной команды прерванной программы. После записи в память старого ССП (т. е. ССП прерванной программы) на блоки процессора поступает новое ССП, характеризующее состояние прерывающей программы. Если независимо от причины прерывания для хранения ССП отводятся одни и те же ячейки ЗУ, то исключается возможность прерывания самой прерывающей программы без потери информации, необходимой для возвращения к ее выполнению. В таком случае для правильной работы цифровой ЭВМ в соответствующий разряд регистра $RG3P$ необходимо записать нуль. Сигнал перехода к продолжению прерванной программы должен вырабатываться программным путем в конце работы прерванной программы. Для этого в машине должна быть специальная команда.

При мультипрограммной работе цифровой ЭВМ возможно взаимное влияние независимых программ из-за ошибок в них, особенно при отладке новых программ, когда в памяти машины уже имеются отлаженные программы. Ошибки могут появиться как за счет неправильного указания адресов в некоторых командах, так и за счет неисправностей в самой машине. Появление ошибки может привести к такой ситуации, при которой одна программа будет записывать информацию в рабочие ячейки другой программы. В результате этого вычисления как по первой, так и по второй программе окажутся неверными.

Для обеспечения сохранности информации, относящейся к одной программе, от возможного разрушающего действия информации, относящейся к другим программам, в современных цифровых ЭВМ имеется совокупность программных и аппаратных средств, получившая название системы защиты памяти. Такая система должна защищать управляющие программы от рабочих программ; участки программ, общие для некоторой их совокупности; участки памяти, временно занимаемые другими программами; так называемые «личные массивы» информации от случайного или преднамеренного использования другими программами и т. п.

Простейший способ организации системы защиты памяти состоит во включении в состав УУ двух регистров, на которые перед началом работы некоторой программы заносятся два граничных адреса участка ЗУ, разрешенного для считывания команд и записи новой информации, получаемой в процессе выполнения программы. Переход к другой программе должен сопровождаться обновлением этих адресов. При каждом обращении к ЗУ все адреса сравниваются с заданными граничными адресами. Выход адреса в команде за указанные пределы приводит к прерыванию выполнения данной программы или к останову машины. Однако этот способ защиты памяти может быть применен лишь тогда, когда вся информация, относящаяся к данной программе, записана в ЗУ в виде единого массива.

Другим способом защиты памяти является пометка отдельных участков памяти специальными признаками, разрешающими запись в них новой информации. Вся память разбивается на l участков длиной в 2^l ячеек. Каждому участку соответствует один триггер l -разрядного регистра. В этот регистр при выполнении любой из программ записывается определенное слово, обновляемое при переходе к другой программе. Выполняемой программе доступны для записи только те участки памяти, которые отмечены признаками. Попытка записи в не отмеченный участок памяти приводит к прерыванию программы или к останову машины. Этот способ защиты памяти требует меньших аппаратных затрат по сравнению с предыдущим способом и является более быстродействующим. Однако число слов в заданном массиве редко бывает кратным величине 2^l , что может приводить к нерациональному использованию некоторого объема памяти.

В цифровых ЭВМ ЕС-1020 для целей защиты памяти ЗУ разделено на блоки по 2048 байт. Каждый такой блок помечается не одноразрядным признаком, а пятиразрядным словом, называемым ключом памяти. Ключ памяти при каждом обращении к ЗУ сравнивается с ключом программы, представляющим собой также пятиразрядное слово. При этом один из разрядов ключа памяти характеризует режим работы системы защиты памяти: защита по записи или защита по чтению. В режиме защиты по записи нарушением защиты памяти считается несоответствие ключа программы ключу памяти блока, в который должна быть записана информация. В режиме защиты по чтению нарушение защиты памяти происходит при несоответствии ключей программы и памяти даже в том случае, когда обращение производилось с целью чтения информации без ее изменения. При работе цифровой ЭВМ в мультипрограммном режиме ключ памяти входит в состав ССП. Любое нарушение защиты памяти приводит к прерыванию программ, а содержимое запрещенного блока памяти остается без изменения.

Современная цифровая ЭВМ, работающая в мультипрограммном режиме, в сущности представляет собой совокупность взаимодействующих модулей и часто называется вычислительной системой. При этом под модулем понимается автономное, логически завершенное и конструктивно оформленное устройство, которое выполняет самостоятельные функции в вычислительном процессе, например: модуль АЛУ, модуль ОЗУ и т. п. В такой ситуации с целью повышения про-

изводительности цифровой ЭВМ совершенно естественным является включение в число модулей независимых процессоров, взаимодействующих через ОЗУ или через специальный канал связи. Такой подход к организации вычислительного процессора привел к созданию мультипроцессорных вычислительных систем, способных параллельно выполнять несколько независимых программ или несколько независимых ветвей одной программы. Возможности такой организации вычислительного процесса имеются при решении многих задач. Наличие нескольких процессоров, модульный принцип построения и возможность быстрой перестройки структуры позволяют создавать вычислительные системы, характеризующиеся высокой надежностью и широкими функциональными возможностями.

Принцип мультипрограммной обработки информации в мультипроцессорных системах получил свое дальнейшее развитие. В таких системах совмещается не только выполнение одной программы с другой, но также выполнение любых независимых последовательностей команд как в одной программе, так и в разных программах. Вместе с тем мультипроцессорные системы отличаются от мультипрограммных, в памяти которых хранится несколько программ, но в каждый момент времени вычисления ведутся лишь по одной программе. Основная цель мультипрограммирования состоит в наиболее полном использовании всего оборудования. Достигается эта цель лучше всего при большом количестве программ в машине, когда, несмотря на прерывания и некоторое увеличение времени выполнения каждой отдельной программы, время выполнения совокупности программ будет меньше времени их выполнения на машине без совмещения операций. Мультипроцессорная же система позволяет уменьшить и время выполнения каждой отдельной программы и время выполнения большого числа программ.

Для обеспечения высокой производительности мультипрограммных цифровых ЭВМ необходима параллельная работа процессора и внешних устройств. Вместе с тем число типов внешних устройств современных цифровых ЭВМ расширилось настолько, что их набор и технические характеристики стали оказывать значительное влияние на эффективность машины в целом. Эти причины обусловили целесообразность выделения из состава процессора схем управления внешними устройствами. Кроме того, алгоритмы управления внешними устройствами часто не зависят от типа устройства и являются общими для достаточно большого их числа. Это делает возможным построение специальных унифицированных устройств управления вводом—выводом, обеспечивающих обмен информацией между памятью машины и внешними устройствами.

Эти устройства получили название каналов ввода—вывода (КВВ) или просто каналов (другие названия: устройства обмена, коммуникационные процессоры, процессоры ввода—вывода).

В машине КВВ соединяются с ОЗУ посредством унифицированной системы связей, называемой интерфейсом ОЗУ (рис. 6.44). Передача информации между КВВ и устройствами ввода—вывода (УВВ) также происходит через унифицированную систему связей — интерфейс ввода—вывода. Чаще всего в цифровых ЭВМ используется передача

информации через интерфейс ВВ по одному байту в каждом такте. Практически интерфейс представляет собой многоконтактное разъемное соединение, содержащее совокупность унифицированных шин и электронных схем, служащих для передачи информации из процессора в КВВ (из КВВ в УВВ) и обратно, а также для управления процессом передачи информации.

При выполнении команд ввода—вывода КВВ устанавливает связь через интерфейс с заданным внешним устройством. Участок ОЗУ, участвующий в операции ввода—вывода, задается каналу специальным управляющим словом путем указания адреса α начальной ячейки и размера β массива данных (например, числа слов или байтов в массиве). Далее КВВ обеспечивает формирование адресов α , $\alpha + 1$, $\alpha + 2$, ..., запись (чтение) информации по этим адресам, преобразование слов, поступающих с УВВ, в слова требуемого формата. Возможны ситуации, когда массив информации, обрабаты-

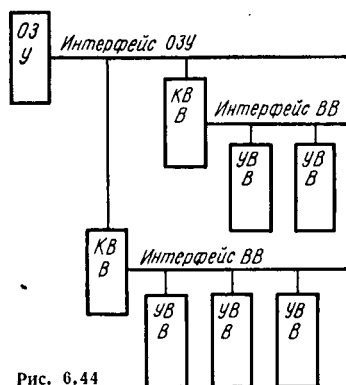


Рис. 6.44

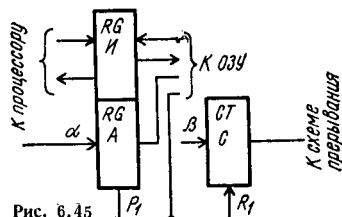


Рис. 6.45

ваемый КВВ, находится в произвольно расположенных участках ОЗУ. В этом случае каждой команде ввода—вывода соответствует последовательность управляющих слов. Помимо перечисленного выше, КВВ может выполнять также функции сортировки данных, что позволяет уменьшить избыточность информации, получаемой от УВВ, и более рационально использовать ОЗУ.

Обычно выделяют два класса внешних устройств: быстродействующие УВВ со скоростью передачи данных около 10^5 байтов за секунду (например, ВЗУ на магнитной ленте) и медленнодействующие УВВ, у которых скорость передачи имеет порядок 10^3 байтов за секунду (например, устройства цифровой печати, устройства для ввода с перфокарт и перфолент). В соответствии с этим различают селекторные каналы, работающие с быстродействующими УВВ, и мультимплексные каналы, обслуживающие медленные УВВ.

Селекторный канал работает в так называемом монопольном режиме, когда с ОЗУ связывается только одно внешнее устройство, а обращение к остальным устройствам становится невозможным. Если селекторный канал подключен к некоторому УВВ и в это время процессор выдает команду обращения к другому УВВ, то выполнение этой команды задерживается до окончания предыдущей операции в КВВ. Основными узлами простейшего селекторного канала (рис. 6.45) являются информационный регистр *RGИ*, регистр адреса

RGА и счетчик слов *СТС*. После каждого обращения к памяти содержимое *RGА* увеличивается на 1, а содержимое *СТС* уменьшается на 1, в результате чего подготавливается очередная операция обмена информацией между *УВВ* и *ОЗУ*. Управляющее слово в селекторном канале формируется путем преобразования самого сигнала запроса *УВВ*.

Помимо перечисленных основных узлов селекторный канал содержит также ряд вспомогательных, обеспечивающих необходимую последовательность работы основных узлов, выбор нужного *УВВ*, защиту памяти, преобразование слов одной длины в слова другой длины, формирование сигналов прерывания и т. п. Для увеличения числа операций, одновременно выполняемых на быстродействующих *УВВ*, цифровая ЭВМ может иметь несколько селекторных *КВВ* (например, в цифровой ЭВМ ЕС-1050 может быть до шести селекторных каналов, каждый из которых допускает подключение до 256 внешних устройств).

Мультиплексный канал позволяет обслуживать несколько параллельно работающих внешних устройств. Низкое быстродействие этих устройств обуславливает формирование запросов к *КВВ* с частотой порядка 10^3 Гц. При этом *КВВ* обслуживает *УВВ* за несколько десятков микросекунд, а число выполняемых запросов в секунду может составлять десятки тысяч. Работа мультиплексного канала организуется таким образом, что обмен информацией между *ОЗУ* и *УВВ* происходит в короткие промежутки времени («сеансы связи»), периодически формируемые каналом для всех *УВВ*. Мультиплексные каналы могут работать и в монопольном режиме — обслуживать одно *УВВ*. Часть оборудования канала, связанную с одним *УВВ* и необходимую для выполнения операции *ВВ*, называют подканалом. Подканал представляет собой область в *ЗУ* канала, где хранятся управляющие слова и другая информация, используемая для организации ввода—вывода.

Состав основных узлов мультиплексного канала такой же, как и селекторного, однако здесь при наличии запроса с *i*-го *УВВ* управляющее слово выбирается из *i*-го подканала и записывается в *СТС* и *RGА*. Далее следует обмен некоторым фиксированным количеством информации (чаще всего одним байтом), изменение на единицу частей α и β управляющего слова и его запись в подканал. Управляющие слова в подканалах могут задаваться как *УВВ*, так и процессором.

Цифровые ЭВМ обычно содержат один мультиплексный канал, однако число подканалов в нем может быть достаточно большим (например, 192 подканала в цифровой ЭВМ ЕС-1050). В машинах малой и средней производительности функции селекторного и мультиплексного каналов иногда выполняет основной процессор. Такие *КВВ* называют встроенными, в отличие от рассмотренных выше автономных каналов.

6.6. МИКРОПРОЦЕССОРЫ, МИКРОПРОЦЕССОРНЫЕ НАБОРЫ И МИКРО-ЭВМ

В номенклатуре изделий электронной промышленности, выпускаемых в настоящее время и предназначенных для использования в ЦВТ, наряду с БИС полупроводниковой памяти важное место занимают микропроцессоры.

Микропроцессор (МП) представляет собой автономное функционально законченное устройство, состоящее из одной или нескольких программно-управляемых БИС, включающее все средства, необходимые для обработки информации и управления данными, и рассчитанное на совместную работу с устройствами памяти и ввода — вывода информации.

Основной причиной появления МП (1971 г.) является преодоление противоречия между стремлением к повышению степени интеграции БИС и сокращением числа их типов. Быстрый прогресс технологии, с одной стороны, позволил создать как запоминающее, так и обрабатывающее БИС с высоким уровнем интеграции (10^4 и более компонент на кристалле). С другой стороны, увеличение уровня интеграции привело к снижению универсальности БИС. Вследствие этого появилось большое число разнотипных БИС, предназначенных для выполнения конкретных функций по обработке информации. Высокая стоимость и длительный срок разработки каждого типа БИС обусловили стремление к сокращению количества типов БИС. Сокращение числа типов БИС потребовало перехода от специализированных к универсальным БИС. В результате появились многофункциональные, а затем и универсальные БИС с программным управлением, ставшие основой построения МП. Вследствие этого МП превратился в универсальную БИС, заменившую специализированные БИС и пригодную для реализации самых различных функций обработки информации. Из-за своей универсальности МП могут производиться большими сериями, что определяет их низкую стоимость.

Создание МП привело к широкому внедрению универсальных вычислительных средств в те отрасли техники, где любые другие вычислительные средства, созданные ранее, не могли быть использованы. По своей структурной и функциональной организации МП аналогичны процессорам, реализующим программную обработку информации. Главными же отличительными признаками МП являются выполнение всех их структурных единиц в виде БИС, небольшая длина операндов, относительно небольшая емкость ЗУ, хранение микропрограмм в ПЗУ и пр. Идея создания универсального процессора малой вычислительной мощности для выполнения самых различных, но сравнительно простых программ, специализируемого на каждый конкретный вид работ путем изменения микропрограмм (т. е. путем замены ПЗУ), возникла значительно раньше появления МП. Еще в начале 60-х годов эта идея была реализована в отечественных управляющих машинах УМ-1НХ, УМШН, «Днепр-1». С помощью современной технологии удалось «сжать» процессоры таких цифровых ЭВМ, занимавшие площадь в несколько квадратных метров, до размеров нескольких стандартных микросхем.

В отличие от специализированных БИС, которые выпускались большими партиями или вообще были заказными изделиями, МП по существу стали первыми крупносерийными БИС общего назначения. Это объясняется следующими их качествами:

1. При проектировании цифровых устройств на основе МП оказывается возможным значительное сокращение производственных затрат, так как МП представляет собой стандартное средство обработки информации.

2. При использовании МП сокращаются материальные, трудовые и временные затраты на проектирование цифровых систем.

3. Применение МП расширяет возможности ЦВТ, при этом цифровые системы могут приобретать новые свойства без изменения состава аппаратных средств.

Различают универсальные и специализированные МП. Последние оптимизированы по различным параметрам на конкретные применения. Специализированные МП целесообразны при больших сериях производства. Применяют МП совместно с микропроцессорными наборами (МН), которые представляют собой совокупности совместимых БИС, разработанных для построения различных микропроцессорных средств обработки информации. Обычно в МН входят ОЗУ, ПЗУ, ГПЗУ, БИС микропрограммного управления и БИС ввода — вывода или интерфейса внешних устройств. На основе МН строят микро-ЭВМ, которые представляют собой конструктивно завершенные вычислительные устройства, оформленные в виде автономного прибора со

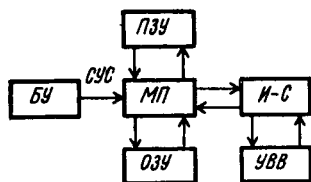


Рис. 6.46

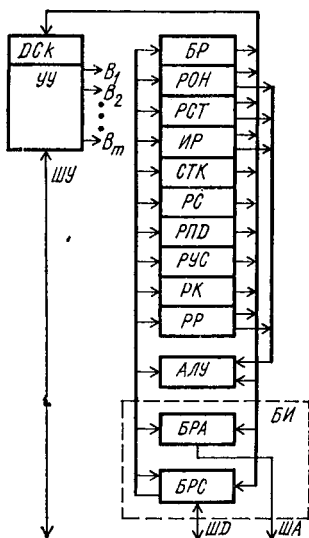


Рис. 6.47

своим источником питания, интерфейсом ввода — вывода и комплексом программного обеспечения. Обобщенная структурная схема микро-ЭВМ показана на рис. 6.46 и состоит из МП, ПЗУ, ОЗУ, БИС интерфейса (И—С) ввода — вывода (УВВ) и блока управления (БУ), генерирующего синхронизирующие и управляющие сигналы (СУС). Кроме микро-ЭВМ на основе МН строят микроконтроллеры (или просто контроллеры), представляющие собой устройства логического управления.

В последнее время получают распространение мультипроцессорные системы, в которых используется более одного микропроцессора с целью обеспечения параллельной обработки информации.

Для производства микропроцессоров используют различные технологии: p — канальную МОП (p — МОП); n — канальную МОП (n — МОП); на дополняющих (комплементарных) МОП транзисторах (КМОП); кремний на сапфире (КС); биполярную ТТЛ; ТТЛ с диодами Шоттки (ТТЛШ); инжекционную (ИИЛ, I^2L); биполярную эмиттерно-связанную (ТЛЭС, ЭСЛ); высокоуровневую МОП (ВМОП). Из перечисленных технологий минимальную стоимость и минимальное бы-

стродействие обеспечивает p — МОП технология, а максимальную стоимость и высокое быстродействие — ЭСЛ технология. Комплементарная МОП технология обеспечивает минимальную потребляемую мощность и среднее быстродействие. Одной из наиболее перспективных является ВМОП технология, позволяющая достичь задержки в 2 нс на каскад и высокой плотности размещения компонент (до 30000 на кристалле в 30 мм^2). В настоящее время промышленностью выпускается более 200 типов МП, обобщенные характеристики которых приведены в табл. 128.

Таблица 128

Характеристики МП	Минимальное значение	Типовое значение	Максимальное значение
Число кристаллов	1	1	7
Число выводов	16	40	64
Число регистров общего назначения	2	5	64
Число регистров АЛУ	1	1	8
Число индексных регистров	0	1	2
Число команд	16	60	150
Формат команд, байт	1	2	3
Разрядность слов, бит	2	8	16
Возможность адресации, байт	256	64K	48M
Число уровней стекового ЗУ	2	8	ОЗУ
Число уровней прерывания	1	4	16
Время сложения, мкс	1	6	62
Тактовая частота, МГц	0,2	2	10
Число фаз тактирующих сигналов	1	2	4
Число источников питания	1	2	3

Со времени появления первых МП прогресс в технологии БИС привел к созданию уже четырех поколений МП, важнейшие отличительные признаки которых отражены в табл. 129. Следует отметить, что понятие поколений МП носит более условный характер, чем понятие поколений цифровых ЭВМ и в меньшей степени отражает кон-

Таблица 129

Поколение	Технология	Управление	Время выполнения операций, мкс	Степень интеграции, компонент	Длина слова, бит	Структура	Примеры МП фирмы Intel
1	p —МОП	Схемное	5—50	10^3	4	Однокристалльные	8008
2	n —МОП, КМОП	Схемное	1—5	$5 \cdot 10^3$	8	Однокристалльные	8080
3	ТТЛШ, ИИЛ	Микропрограммное	0,5—1	10^3	2—4	Многочристалльные	3000
4	n —МОП, ВМОП	Схемное	0,5—3	$30 \cdot 10^3$	16	Однокристалльные	8086

структивно-технологические особенности их изготовления. Каждое последующее поколение цифровых ЭВМ по основным технико-экономическим показателям превосходит предыдущие поколения и вытесняет их. Микропроцессоры же разных поколений применяются совместно и не вытесняют, а взаимно дополняют друг друга. Тенденция перехода от низших поколений МП к высшим хотя и существует, однако выражена менее сильно, чем в области универсальных цифровых ЭВМ.

Рассмотрим теперь архитектуру МП и микро-ЭВМ, т. е. структуру отдельных устройств, их взаимосвязь, систему команд и взаимодействие между аппаратными средствами и программами обработки информации. Архитектура МП во многом повторяет архитектуру обычных процессоров. Однако в отличие от обычных процессоров, у которых операции могут выполняться параллельно, МП реализуют операции последовательно-параллельно по байтам (в этом их основной недостаток и причина сравнительно низкого быстродействия).

Структура МП должна удовлетворять трем основным требованиям: быть функционально гибкой, обеспечивать достаточно высокие быстродействие и допускать недорогую технологическую реализацию. Высокая функциональная гибкость МП, необходимая для создания эффективного программного обеспечения, достигается микропрограммным управлением, за счет побайтовой обработки и адресации данных, развитой системы прерываний и большего числа внутренних регистров. Различают однокристалльные и многокристалльные (модульные) МП. Структура однокристалльного МП показана на рис. 6.47. Такой МП содержит арифметико-логическое устройство (АЛУ), устройство управления (УУ), блок внутренних регистров (БР) и блок интерфейса (БИ). АЛУ, как правило, состоит из двоичного сумматора и нескольких регистров и аппаратным способом выполняет простейшие операции (сложение, вычитание, пересылку, основные логические операции и сдвиг). Сложные операции выполняются по микропрограммам. Блок внутренних регистров (БР) образует внутреннюю память МП и содержит специальные регистры и регистры общего назначения (РОН). Регистры БР связаны с другими частями МП общими шинами и во многом определяют вычислительные возможности МП. Все РОН доступны выполняемой программе и часто помимо основных функций (хранение операндов) выполняют функции специальных регистров, например регистров стека (РСТ), индексных регистров (ИР). Существует тенденция увеличения числа РОН, обусловленная тем, что увеличение объема внутренней памяти позволяет повысить производительность МП за счет сокращения числа обращений к ОЗУ. Специальные регистры выполняют функции счетчика команд (СТК) и регистров команд (РК), результата (РР), указателя стека (РУС), прямого доступа к памяти (РПД) и состояния (РС). РС фиксирует информацию о результатах выполнения последней обработанной команды. Например, является результат нулевым или имеет положительный или отрицательный знак, имеется ли переполнение или запрос на прерывание. Эта информация используется для реализации условных переходов по программе. Стек выполняется на совокупности внутренних регистров МП и входит в его состав, либо же организуется на некоторой

выделенной области ОЗУ. В последнем случае обращение и адресация к стеку производится через РУС. Стек на регистрах более быстродействующий, но из-за малого числа регистров в МП он имеет небольшую емкость. В РУС хранится адрес первой свободной ячейки в стеке. При записи слова в стек значение адреса в РУС увеличивается на единицу, а при чтении уменьшается на единицу. Помимо этого, имеется возможность прочитать без разрушения содержимое любой ячейки стека без изменения слова в РУС.

Устройство управления (УУ) вырабатывает серии внутренних сигналов управления $B_1, B_2, \dots, B_m$ с помощью дешифратора кода операции ДСк. Блок интерфейса — аппаратные средства, обеспечивающие сопряжение между МП, ЗУ и периферийными устройствами (внешний интерфейс), а также между частями МП (внутренний интерфейс). Внутренняя шина данных МП представляет собой многопроводную магистраль, связывающую все узлы внутри БИС МП. Обычно разрядность внутренней шины данных (8 разрядов) совпадает с разрядностью внешней шины данных (ШД), по которой передаются операнды и команды, и определяет разрядность слов, с которыми оперирует микропроцессор. Из-за ограниченного числа внешних выводов (особенно в однокристальных МП) шина данных часто работает через специальную буферную схему мультиплексора в режиме двунаправленной передачи. При этом противоположные направления обмена данными между МП и другими устройствами разделены во времени.

Шина адреса (ША) микропроцессора чаще всего бывает 16-разрядная, чего достаточно для прямой адресации периферийного оборудования и внешней памяти емкостью 64 К байт. Так как внутренняя шина данных в большинстве случаев имеет меньшую разрядность (чаще всего 8 разрядов), то 16-разрядный адрес формируется из нескольких слов. Двунаправленная шина управления (ШУ) обычно содержит 8 разрядов, по которым передаются управляющие сигналы о состоянии внешних устройств, запроса на прерывание от внешних устройств к МП и разрешения на прерывание от МП к внешним устройствам, обращения к ОЗУ и маркеры, сопровождающие информацию, передаваемую по ШД.

Многокристалльный МП (рис. 6.48) выполняется, как правило, на основе БИС двух типов: управляющей (БИСУ) и операционной (БИСО). Операционные БИС содержат АЛУ и РОН с разрядностью в 2 или 4 бита и снабжены схемами и выводами, позволяющими объединить их в операционную часть МП произвольной разрядности (рис. 6.48). Управляющая БИС содержит ряд общих для МП регистров и ЗУ микропрограмм (ЗУМ), из которого считываются параллельно на все БИСО сигналы микрокоманд МК. Наличие нескольких типов БИС является недостатком многокристалльных МП, но нарастающая разрядность позволяет повысить точность и скорость обработки информации. Этому же способствует меньшая степень интегра-

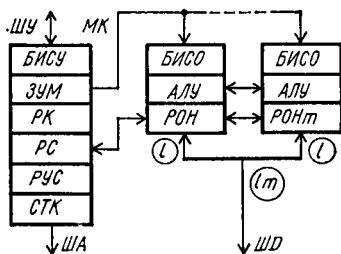


Рис. 6.48

ции отдельных БИС, что позволяет применять технологию, обеспечивающую более высокое быстродействие БИС. Кроме того, набор микрокоманд можно менять заменой только одной БИСУ или изменением содержимого ЗУМ (в случае использования ППЗУ). Вместе с тем менее жесткие требования к количеству внешних выводов позволяют использовать независимые ША, ШУ и ШД, что также способствует повышению быстродействия МП, позволяет исключить мультиплексор и упростить структуру МП.

Наиболее типичная система команд МП содержит набор из 70 — 100 трех-, двух-, одно- и безадресных команд. Однако ограничения на число выводов и сложность УУ не позволяют в полной мере использовать преимущества трех и двухадресных команд. Вследствие этого в МП наибольшее распространение получили команды, содержащие один адрес операнда или несколько коротких адресов операндов, хранящихся в РОН. Все команды любого микропроцессора можно условно разбить на арифметические, логические и вспомогательные. В свою очередь вспомогательные команды состоят из команд пересылки, ветвления, обращения к подпрограммам и управления вводом и выводом.

Арифметические команды МП состоят, как правило, из операций сложения, вычитания, добавления или вычитания 1 и сдвига. Логические команды выполняют поразрядные операции дизъюнкции, конъюнкции, неравнозначности и сравнения. Команды пересылки обеспечивают двусторонний обмен информацией между внутренними регистрами МП или регистрами и ОЗУ. Команды ветвления реализуют условные и безусловные переходы в программе. Команды обращения к подпрограммам реализуют вход в подпрограмму и последующий возврат к основной программе. Команды управления вводом — выводом реализуют операции ввода — вывода информации из внешней памяти и периферийных устройств.

В МП применяют прямую, непрямую, непосредственную и неявную адресации. Сущность первых двух способов адресации соответствует прямой и непрямой (присоединенной, относительной, косвенной) адресации в больших универсальных цифровых ЭВМ.

При непосредственной адресации содержимое адресной части команды является операндом. Команды в таком случае не обращаются в ОЗУ за операндами, что обеспечивает экономию памяти и увеличивает скорость обработки информации.

При неявной адресации адрес в команде явно не указывается, но код операции автоматически задает обращение к местам хранения данных, требуемых для выполнения операции. Например, для выполнения арифметических операций с неявной адресацией данные берут из РР.

Обмен данными в микро-ЭВМ происходит между МП, ОЗУ и УВВ. Обмен информацией между МП и ОЗУ часто осуществляется через интерфейс, расположенный на кристалле МП. Обмен между МП и УВВ осуществляется либо через специализированную БИС сопряжения, ориентированную на некоторый класс УВВ, либо через универсальный программируемый интерфейс (программируемый контроллер УВВ), обеспечивающий подключение к МП практически любого

УВВ. В наиболее общем случае программируемые контроллеры выполняют следующие функции:

- 1) управляют процессом передачи данных между МП и УВВ;
- 2) осуществляют преобразование форматов данных;
- 3) дешифрируют адреса УВВ, посылаемые МП;
- 4) передают сигналы готовности УВВ к обмену данными с МП;
- 5) дешифрируют коды операций, посылаемые МП, и осуществляют их выполнение.

Заметим, что обмен данными между ОЗУ и УВВ может осуществляться и через МП, однако это снижает быстродействие микроЭВМ. Поэтому в ряде МН имеется БИС, реализующие канал прямого доступа УВВ к памяти, с помощью которой обмен данными осуществляется без участия МП.

Устройства управления однокристальных МП первых поколений (например, моделей 8008 и 8080 фирмы Intel) строились исключительно по схемному способу. Однако повышение степени интеграции и требования гибкости структуры МП привели к широкому применению микропрограммного управления не только в многокристальных, но и в однокристальных МП третьего поколения. Различают три способа размещения ЗУ микропрограмм (ЗУМ) по отношению к МП, а именно: внутреннее, внешнее и комбинированное.

При внутреннем размещении ЗУМ находится на одном кристалле с МП и с точки зрения пользователя такое управление ничем не отличается от схемного. В этом случае МП реализует фиксированное множество микропрограмм, соответствующее заданной системе команд. Достигнутая степень интеграции лимитирует количество аппаратуры в МП и позволяет реализовать систему команд ограниченной мощности. Быстродействие МП с таким размещением ЗУМ наибольшее, однако замена микропрограмм, а следовательно, и системы команд здесь невозможна. Исключение составляет случай применения в качестве ЗУМ двусторонней (активной) памяти. Однако емкость такой памяти, как правило, недостаточная для хранения развитой системы микропрограмм. Кроме того, трудности создания полупроводниковой памяти, сохраняющей информацию при отключении питания, существенно сужают область применения МП с двусторонним ЗУМ.

При внешнем размещении ЗУМ находится вне МП и может быть выполнено в виде ПЗУ, ППЗУ или совмещено с ОЗУ. Микрокоманды из ЗУМ в этом случае поступают по ШД в регистр микрокоманд. Поэтому разрядность микрокоманд определяется разрядностью ШД. Быстродействие МП ограничено длительностью цикла внешнего ЗУМ и меньше, чем при внутреннем размещении ЗУМ. Однако внешнее размещение ЗУМ позволяет не только разработчику, но и пользователю изменять и расширять систему команд, приспособлявая ее к специфике конкретного применения.

При комбинированном размещении ЗУМ имеется как внутренняя, так и внешняя микропрограммная память, благодаря чему такой вариант объединяет в себе преимущества обоих способов: от внутреннего ЗУМ — быстродействие, от внешнего ЗУМ — гибкость. Однако при реализации такого размещения ЗУМ возникают трудности, обусловленные малым количеством шин, связывающих ЗУМ и МП. Один

из путей преодоления этих трудностей заключается в использовании двухуровневого микропрограммирования, при котором во внешнем ЗУМ хранятся только адреса микрокоманд, записанных во внутреннем ЗУМ. При этом адресоваться могут не только отдельные микрокоманды, но и микроподпрограммы или микропрограммы базовой системы команд. Одним из эффективных способов реализации такого микропрограммирования является применение механизма блокировки, позволяющего значительно расширить набор исходных микрокоманд.

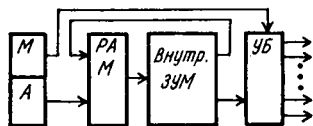


Рис. 6.49

При этом внешняя микрокоманда (т. е. хранящаяся во внешнем ЗУМ) будет иметь структуру вида А, М, где А — адрес внутренней микрокоманды (хранящейся во внутреннем ЗУМ), М — маска, указывающая группу микроопераций, которую надо блокировать при выполнении данной микрокоманды. Упрощенная схема, реализующая двухуровневый принцип микропрограммирования, показана на рис. 6.49. Здесь внешняя микрокоманда разбивается на две части: А и М. Часть А поступает на регистр адреса внутренней микрокоманды РАМ, вследствие чего из внутреннего ЗУМ выбирается внутренняя микрокоманда, часть управляющих сигналов которой блокируется узлом блокировки (УБ), управляемым маской М. Для уменьшения

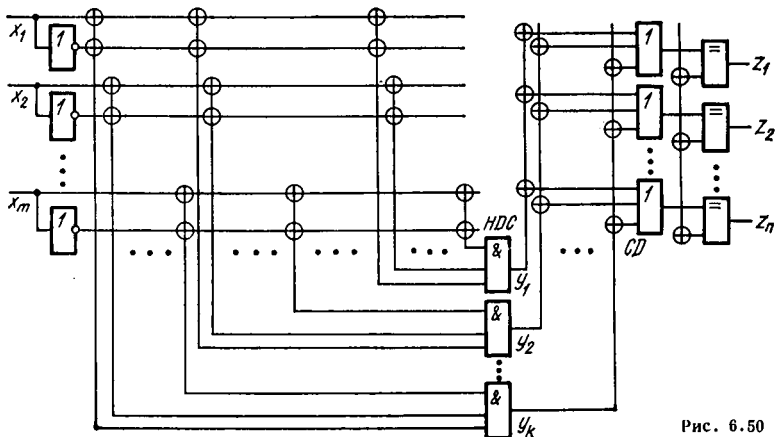


Рис. 6.50

длины адреса А можно использовать один из методов не прямой адресации, например, присоединенную или относительную. Однако в этом случае необходимо иметь внешнюю микрокоманду записи слова в регистр базового адреса микрокоманды.

Другой особенностью управления МП является широкое применение программируемых логических матриц (ПЛМ) вместо ПЗУ. Логическая структура ПЛМ (рис. 6.50) во многом подобна структуре ПЗУ и состоит из неполного дешифратора НДС (в ПЗУ дешифратор полный) и шифратора СД. В узлах пересечения шин, отмеченных кружками, находятся плавкие перемычки, которые при записи информации в ПЛМ могут пережигаться. Сигналы с шифратора поступают на вы-

ход ПЛМ через элементы равнозначности, позволяющие инвертировать выходные сигналы. В одной из выпускаемых промышленностью ПЛМ, например $m = 16$, $k = 48$, $n = 8$. Программируемые логические матрицы в устройствах управления МП применяются для преобразования микрокоманд или адресов, в качестве ЗУМ, либо для выполнения всех указанных функций одновременно. В последнем случае микропрограммное ЗУ становится предельно простым, состоящим из одной или нескольких параллельно работающих ПЛМ (рис. 6.51). Регистр Pk выбирает нужную совокупность микрокоманд, отдельные микрокоманды из которой выбираются содержимым регистра адреса микрокоманд (РАМК) и логическими условиями (ЛУ). Адрес микрокоманды может задавать внешняя МК или же он может быть сформирован с помощью ПЛМ за счет обратной связи, которая на рис. 6.51 указана штриховой линией. Применение вместо ПЛМ ПЗУ, все адреса которого дешифрируются, потребовало бы технически недостижимой емкости на одном кристалле. Для рассмотренного примера необходимо было бы иметь ПЗУ емкостью $8 \cdot 2^{16}$ байт вместо ПЛМ емкостью 48 8-разрядных слов.

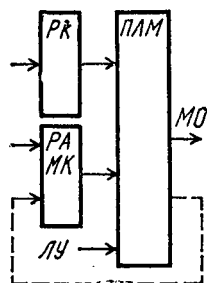


Рис. 6.51

6.7. СТРУКТУРЫ ОДНОКРИСТАЛЛЬНЫХ И МНОГОКРИСТАЛЛЬНЫХ МИКРОПРОЦЕССОРОВ

Особенности структур однокристалльных МП рассмотрим на примере первого и наиболее популярного универсального МП второго поколения типа 8080 фирмы Intel. Этот МП представляет собой параллельный 8-разрядный МП с фиксированной системой команд, реализованный на 5000 n — МОП транзисторах, размещенных на кристалле размерами $4,2 \times 4,8$, заключенном в корпус с 40 выводами. Частота тактовых сигналов для МП равна 2 МГц.

Микропроцессор состоит из БР, АЛУ и УУ (рис. 6.52). Внутренняя шина данных (ВШД) связывает между собой все указанные части МП (все информационные шины и регистры имеют длину в 8 разрядов, за исключением специально оговоренных случаев). АЛУ работает в дополнительном коде и выполняет операции сложения, сдвиг влево и вправо, поразрядные логические операции конъюнкции, дизъюнкции, отрицание дизъюнкции и равнозначности, определяет знак результата. Состоит АЛУ из трех арифметических регистров ($P1$, $P2$ и PP), регистра состояния (PC) для фиксации различных признаков (переполнения, нуля на сумматоре, десятичной арифметики), десятичного корректора ($ДК$), необходимого для обработки двоично-десятичных чисел, преобразователя кодов ($ПК$), сумматора ($СМ$).

Блок регистров состоит из мультиплексора (M) чтения и записи; восьми регистров (B , C , D , E , L , M , W , Z), которые могут использоваться индивидуально как 8-разрядными или парами как 16-разрядные при необходимости получения удвоенной точности вычислений, счетчика команд ($СТК$) и 16-разрядного регистра — указателя стека ($РУС$), которые позволяют непосредственно адресовать сте-

ковую память емкостью 64 К байт. Обмен данными и адресами производится через соответствующие буферные схемы БСД и БСА. Регистр изменения адреса (РИА), снабженный соответствующими цепями, позволяет прибавлять или вычитать из адреса заданную константу. Назначение выводов БИС МП и направления передачи информации по ним указаны в табл. 130, где запись «Из МП, в МП» означает двунаправленную цепь (шину). Шина адреса (ША) обеспечивает адресацию памяти емкостью 64 К байт или коммутацию 256 устройств ввода и 256 устройств вывода. Шина данных (ШД) обеспечивает двусторонний обмен информацией между МП и другими устройствами (например, ОЗУ или УВВ).

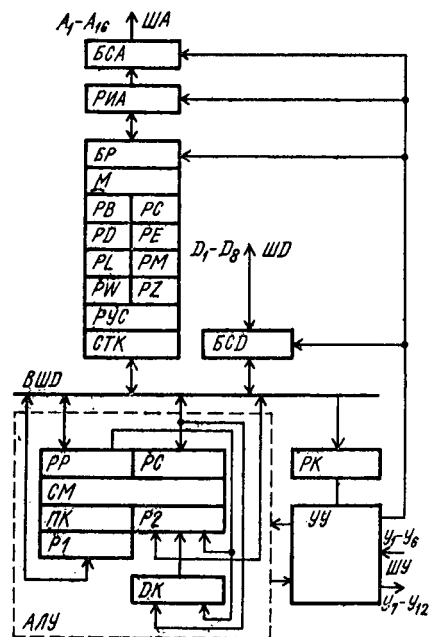


Рис. 6.52

Шина управления (ШУ) имеет шесть входных ($y_1 — y_6$) и шесть выходных ($y_7 — y_{12}$) линий, которые служат для передачи следующих сигналов: y_1 и y_2 — последовательности синхронизирующих сигналов $C1$ и $C2$, не пересекающиеся во времени; y_3 — сигнал сброса, очищающий счетчик команд БТК и обеспечивающий выполнение программы, начиная с нулевой ячейки памяти; y_4 — сигнал готовности, поступающий от внешних устройств и свидетельствующий о готовности данных для ввода в МП. Если после обращения МП к внешним устройствам этот сигнал не появляется, МП переходит в режим ожидания, что подтверждается появлением сигнала на выходе ожидания y_8 . Линия y_5 служит для сигналов запроса шин от внешних устройств для

прямого (минуя МП) выхода на внешние шины адреса и шины данных. Если сигнал «Запрос шин» принят микропроцессором, то об этом свидетельствует выходной сигнал y_{12} «Подтверждение запроса шин». Перед этим МП заканчивает выполнение текущего машинного цикла и приостанавливает дальнейшие действия до тех пор, пока не исчезнет сигнал запроса шин. Сигнал по входу y_6 «Запрос прерывания» — сигнал от какого-либо внешнего устройства на прерывание работы МП и обслуживание данного внешнего устройства. Этот сигнал воздействует на триггер разрешения прерывания и, если триггер установлен в состояние разрешения, МП переходит к выполнению программы обработки прерывания. Об установке триггера в состояние разрешения свидетельствует сигнал y_9 «Разрешение прерывания». Сигнал на выходе y_7 указывает начало машинного цикла и называется сигналом синхронизации. Во время действия этого сигнала информация о режиме работы МП по ШД пересылается в регистр со-

№ вывода	Направление передачи	Назначение цепей	№ вывода	Направление передачи	Назначение цепей
1	Из МП	Разряд A_{11} шины адреса	21	Из МП	Подтверждение запроса шин (y_{12})
2	—	Земля	22	В МП	Синхронизирующие сигналы C_1 (y_1)
3	Из МП,	Разряд D_5 шины данных	23	»	Готовность (y_4)
4	в МП	Разряд D_6 шины данных	24	Из МП	Ожидание (y_8)
5	То же	Разряд D_7 шины данных	25	То же	Разряд A_1 шины адреса
6	»	Разряд D_8 шины данных	26	»	Разряд A_2 шины адреса
7	»	Разряд D_4 шины данных	27	»	Разряд A_3 шины адреса
8	»	Разряд D_3 шины данных	28	—	+12 В
9	»	Разряд D_2 шины данных	29	Из МП	Разряд A_4 шины адреса
10	»	Разряд D_1 шины данных	30	»	Разряд A_5 шины адреса
11	—	—5В	31	»	Разряд A_6 шины адреса
12	В МП	Сброс (y_3)	32	»	Разряд A_7 шины адреса
13	»	Запрос шин (y_5)	33	»	Разряд A_8 шины адреса
14	»	Запрос прерывания (y_6)	34	»	Разряд A_9 шины адреса
15	»	Синхронизирующие сигналы C_2 (y_2)	35	»	Разряд A_{10} шины адреса
16	Из МП	Разрешение прерывания (y_9)	36	»	Разряд A_{16} шины адреса
17	То же	Разрешение ввода данных (y_{10})	37	»	Разряд A_{13} шины адреса
18	»	Запись-чтение (y_{11})	38	»	Разряд A_{14} шины адреса
19	»	Синхронизация (y_7)	39	»	Разряд A_{15} шины адреса
20	—	+5В	40	»	Разряд A_{12} шины адреса

стояния. Таких режимов восемь, а именно: 1 — «Считывание памяти», при котором данные из памяти поступают на ШД; 2 — «Запись во внешнее устройство», при котором на ШД присутствуют данные для их записи в память или во внешние устройства ввода — вывода; 3 — «Вводной цикл», при котором на ША установлен адрес устройства ввода; 4 — «Выводной цикл», при котором на ША установлен адрес устройства вывода; 5 — «Стек», при котором на ША формируется адрес стека; 6 — «М/», при котором в текущем цикле выбирается первый байт команды; 7 — «Подтверждение прерывания», который устанавливается вслед за приемом от внешних устройств входного

сигнала запроса на прерывание; 8 — «Подтверждение останова», при котором выполняется команда «Останов».

Выходной сигнал y_{10} указывает на то, что МП готов принять информацию от внешних устройств на ШД. Сигнал y_{11} запись-чтение — выходной сигнал, необходимый для записи данных в память или для управления вводом — выводом.

Формат команд рассматриваемого МП переменный и может состоять из 1, 2 или 3 байт. Большинство команд однобайтовые. Двухбайтовые команды служат для непосредственной адресации и работы с устройствами ввода — вывода. Трехбайтовыми являются, например, команды переходов. Для выполнения команды требуется от одного до пяти машинных циклов, причем каждый машинный цикл в процессе выполнения команды длится от 3 до 5 тактов, длительность которых определяется тактовой частотой и составляет 0,5 мкс. Цикл «M1» продолжается 4 или 5 тактов ($T1$ — $T5$) и служит для выборки кода операции. В такте $T1$ осуществляется пересылка содержимого СТК в основную память (ОЗУ) через регистр адреса и ША для чтения следующей команды. При этом выходной сигнал «Синхронизация» (на выходе y_7), указывающий начало каждого машинного цикла, принимает единичное значение, а на ШД формируется слово, определяющее «Вводный цикл» и означающее, что на ША установлен адрес устройства ввода. В течение такта $T2$ осуществляется проверка условия готовности данных для ввода из МП во внешнюю память или от внешних устройств, проверка запроса шин для управления прямым доступом к памяти внешних устройств и проверка подтверждения о выполнении команды «Останов». Если сигнал готовности равен 1, то произойдет переход МП к выполнению такта $T3$. Если же сигнал готовности в течение времени $T2$ отсутствует (равен нулю), то МП переходит в состояние ожидания и остается в этом состоянии до тех пор, пока сигнал готовности не примет единичное значение. Наличие входного сигнала y_4 «Готовность» обеспечивает синхронизацию МП с внешней памятью или устройством ввода — вывода с любым временем доступа. Кроме того, управляя этим сигналом, можно обеспечить выполнение всей программы в пошаговом режиме, например, при отладке микро-ЭВМ.

В течение такта $T3$ осуществляется передача выбранной команды из основной внешней памяти (ОЗУ) по ШД в регистр команд. Выполнение команды начинается в такте $T4$. Короткие операции выполняются за один такт. Для завершения длинной операции может понадобиться такт $T5$ или даже от одного до трех дополнительных циклов $M2$ — $M5$. Данные для ввода — вывода между машинными циклами хранятся в 8-разрядном буферном регистре данных.

Микропроцессор 8080 входит в состав микропроцессорного набора MCS-80, который включает микросхемы ОЗУ, ПЗУ, ППЗУ, интерфейса, управления, тактового питания и другого назначения (всего около 60 БИС) и по мере разработки пополняется БИС с улучшенными технико-экономическими показателями и новым функциональным назначением. Пример построения микро-ЭВМ из компонент микропроцессорного набора MCS-80 показан на рис. 6.53. Эта микро-ЭВМ состоит из БИС МП (8080), сопряжения (8228), ПЗУ (8316), ОЗУ (8101),

интерфейса (8212) и управления прерыванием (8214), интерфейса связи (8251), интерфейса устройств ввода — вывода (8255), тактового генератора (8224).

На основе микропроцессорного набора MCS-80 разработан и серийно выпускается более современный набор MCS-85, работающий от одного источника питания — 5В. МП 8085 во многом похож на МП 8080 и совместим с ним программно и по системе шин, но работает на частоте 3 МГц и обеспечивает вдвое большее быстродействие, чем МП 8080, выполняя при этом собственно функции микропроцессора, а также синхрогенератора, системного контроллера или контроллера прерывания и канала последовательного ввода — вывода, т. е. функции трех БИС из указанных ранее микросхем набора MCS-80.

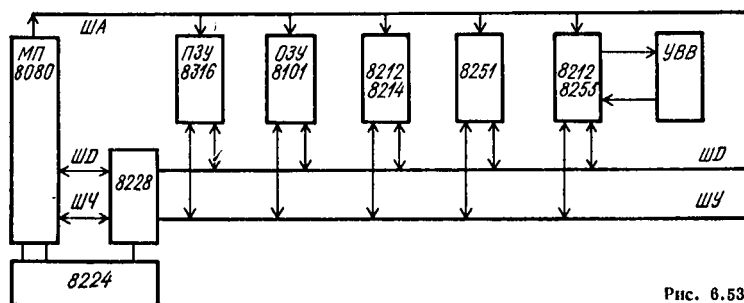


Рис. 6.53

Как правило, многокристалльные МН строятся по биполярной технологии, что повышает более чем на порядок их быстродействие по сравнению с однокристалльными МП. Однако при этом уменьшается степень интеграции на кристалле и растет энергопотребление. Относительно невысокий уровень интеграции биполярных схем пока не позволяет получить однокристалльные МП. Особенности многокристалльных МП рассмотрим на примере МП 3000 фирмы Intel, который является быстродействующим универсальным МП с микропрограммным управлением, изготавливаемым по ТТЛШ — технологии.

Микропроцессорный набор серии 3000 состоит из запоминающих, процессорных и интерфейсных БИС и предназначен для построения быстродействующих контроллеров с частотой выдачи управляющих сигналов до 10 МГц, микро-ЭВМ со временем выполнения операций типа регистр — регистр около 1 мкс и других микропроцессорных систем. МН типа 3000 совместим со всеми сериями ТТЛ интегральных схем и имеет напряжение питания 5В.

В состав МН входят: блок микропрограммного управления (БМУ), центральный процессорный элемент (ЦПЭ), схема ускоренного переноса (СУП), многорежимный буферный регистр (МБР), блок приоритетного прерывания (БПП), шинный формирователь (ШФ), шинный формирователь с инверсией (ШФИ).

БМУ размещен в корпусе с 40 выводами и состоит (рис. 6.54) из буферной схемы (БС), регистров команд (РК), признаков (РП) и адреса микрокоманд (РАМК), а также схемы определения адреса следующей микрокоманды (СОА), на которую поступают управляющие сигналы

с ЗУ микрокоманд и адресная часть микрокоманд (АЧМК). БМУ управляет выработкой заданной последовательности адресов МК и ЗУ микропрограмм (ЗУМ). Адрес очередной МК в ЗУМ задается 9-разрядным словом, формируемым БМУ, вследствие чего общее число адресуемых МК может достигать 512. Адрес очередной МК формируется

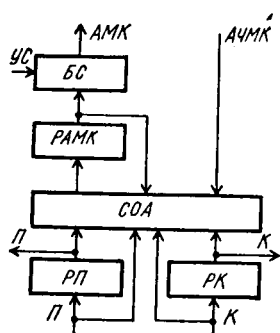


Рис. 6.54

на основе 7-разрядного слова, поступающего из ЗУМ в БМУ, текущего адреса МК и слов из РК и РП. Минимальная длина микрокоманды составляет 18 разрядов, из них семь разрядов предназначены для управления адресом следующей микрокоманды, семь разрядов для управления микрооперациями на ЦПЭ и четыре разряда для управления логической схемой признаков. Формат микрокоманд может быть увеличен для реализации, например, таких функций, как управление внешней памятью.

ЦПЭ размещен в корпусе с 28 выводами и организован по схеме на рис. 6.55, а, а его условное обозначение показано на рис. 6.55, б.

Центральный процессорный элемент содержит двухразрядное АЛУ, которое под управлением дешифратора микрокоманд ДСМ выполняет 40 логических и арифметических операций, буферные схемы адреса (БСА) и данных (БСД), регистр результата (РР), мультиплексоры МА и МВ и одиннадцать двухразрядных регистров общего назначения

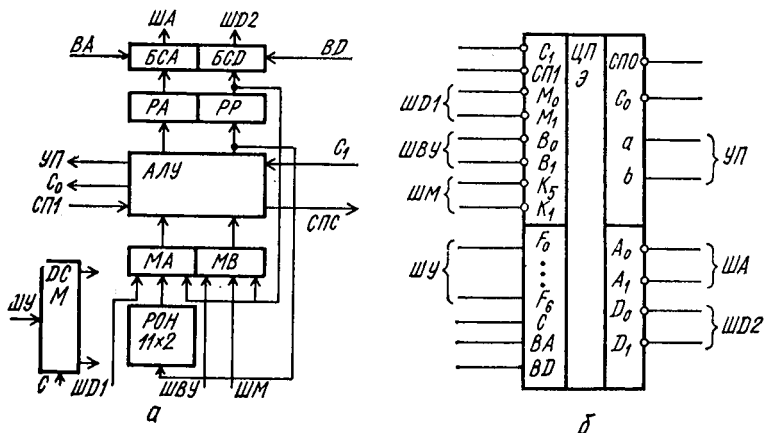


Рис. 6.55

(РОН). ЦПЭ имеет систему двухразрядных шин для ввода данных в ЦПЭ из ОЗУ (ШД1) и внешних устройств (ШВУ), для занесения маски микрокоманды или константы (ШМ), для выдачи содержимого РА (ША) и РР (ШД2), для выдачи сигнала ускоренного переноса (УП). Семиразрядная входная шина ШУ служит для управления (через ДСМ) выполнением в АЛУ микроопераций. Сигналы на вход ШУ подаются с выхода ЗУ микропрограмм. Одноразрядные шины

ЦПЭ используются как входная и выходная шины переноса C_0 и C_1 , входная и выходная шины сдвига слов вправо СП1 и СП0, входные шины разрешения выдачи адреса (ВА), данных (ВД) и шины синхронизации C , питания и земли.

Ввиду достаточного числа выводов при малой разрядности удалось разделить все шины не только по функциональному назначению (ША, ШД и ШУ), но и по направлениям (входные и выходные шины). Такое разделение шин позволяет устранить временное мультиплексирование и внешние регистровые схемы, что повышает быстродействие МП и сокращает количество вспомогательного оборудования.

Вследствие того что в ЦПЭ содержатся отдельные логические схемы разрешения на выдачу адресов и данных, возможно, при необходимости, объединение их входов и выходов и образование двухнаправленных шин.

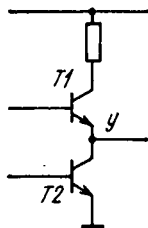
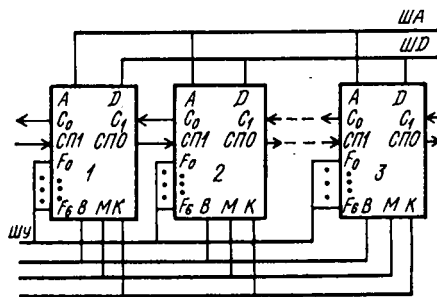


Рис. 6.56



цепи самого старшего переноса, перевода его в третье состояние. Условное обозначение и распределение выводов СУП приведены на рис. 6.58. Работает такая схема следующим образом. Сумма двухразрядных чисел без учета переноса в младший разряд может быть равна одному из 7 значений, указанных в табл. 131. При значениях суммы 000, 001 и 010 перенос c_{n+1} в следующий ЦПЭ возникнуть не может.

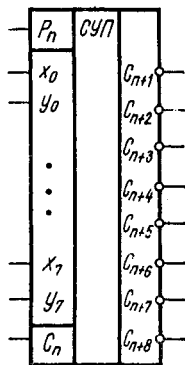


Рис. 6.58

Таблица 131

Сумма	a	b
000	0	0
001	0	0
010	0	0
011	1	0
100	1	1
101	1	1
110	1	1

Таблица 132

Сумма	a	b
00000	0	0
00001	0	0
...	..	..
01110	0	0
01111	1	0
10000	1	1
...	..	..
11110	1	1

При значении суммы 011 перенос c_{n+1} равен единице только при наличии переноса c_n . При суммах 100, 101, 110 перенос c_{n+1} возникает всегда. Если указанные три случая закодировать переменными a и b так, как показано в таблице, то

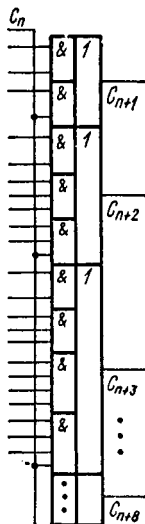


Рис. 6.59

$$\begin{aligned}
 c_{n+1} &= a_n b_n \vee b_n c_n; \\
 c_{n+2} &= a_{n+1} b_{n+1} \vee b_{n+1} c_{n+1} = \\
 &= a_{n+1} b_{n+1} \vee b_{n+1} a_n b_n \vee b_{n+1} b_n c_n; \\
 c_{n+3} &= a_{n+2} b_{n+2} \vee b_{n+2} c_{n+2} = a_{n+2} b_{n+2} \vee b_{n+2} a_{n+1} b_{n+1} \vee \\
 &\vee b_{n+2} b_{n+1} a_n b_n \vee b_{n+2} b_{n+1} b_n c_n.
 \end{aligned}$$

В соответствии с этими выражениями и строится логическая структура СУП (рис. 6.59). Схема подключения СУП и ЦПЭ показана на рис. 6.60. Интересной особенностью СУП является ее независимость от разрядности ЦПЭ. Например, для 4-разрядного ЦПЭ правило формирования сигналов a и b приведено в табл. 132, из которой следует, что при значениях суммы от 0 до 14 переноса из данного ЦПЭ не будет. Если же сумма равна 15, то перенос из данного ЦПЭ возникает только тогда, когда будет перенос в данный ЦПЭ. Если же сумма больше 15, то перенос всегда возникает. Указанные три случая можно закодировать переменными a и b так же, как и в таблице 131, что приведет к выражениям

и способам соединения, тождественным ранее рассмотренным.

Многорежимный буферный регистр (МБР) является универсальным 8-разрядным регистром с выходами, имеющими три состояния. МБР могут быть использованы для реализации простых регистров, буферных регистров со стробированием и двунаправленных шинных формирователей. Условное обозначение МБР приведено на рис. 6.61, a .

МБР состоит из восьми информационных Д-триггеров, восьми выходных буферных логических элементов (Э) с тремя устойчивыми состояниями, отдельного Д-триггера для формирования запроса на прерывание (ЗП) и схемы управления режимами работы регистра (рис. 6.61, б). Сигнал на шине BP («выбор режима») определяет операцию (МО), выполняемую на регистре, а именно, прием (Π) и выдачу (B) слова:

$$\Pi = C (\overline{BP}) \vee (\overline{BK1}) (BK2) (BP),$$

$$B = BP \vee (BK1) (BK2),$$

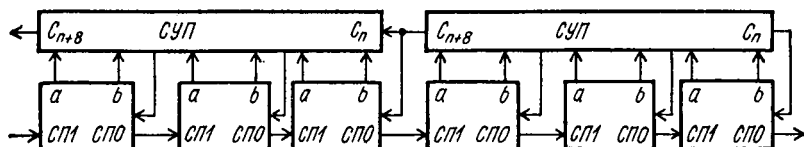


Рис. 6.60

Считают, что МБР находится в состоянии прерывания, когда выходу $\overline{ЗП}$ соответствует логический «0», что обеспечивается сигналом $1ТЗП=0$ или сигналом выбора кристалла $(\overline{BK1} BK2=1)$. Установка ТЗП в состояние 0 (Запрос) осуществляется стробом C . Снятие сигнала запроса осуществляется при сбросе в нуль МБР сигналом $R=0$ при снятом сигнале выбора кристалла путем установки ТЗП в состояние «1» (запроса нет). Примеры применения МБР показаны на рис. 6.62. В режиме входного буфера (рис. 6.62, а) МБР фиксирует поступающую на выходы D_i информацию и по сигналу B транслирует ее на выходы Q_i . В режиме выходного буфера (рис. 6.62, б) МБР обеспечивает прием информации по сигналу Π , формирование сигнала готовности информации на выходе ЗП и установку в нуль (УО) МБР. В режиме двунаправленной передачи информации (рис. 6.62, в) сигнал A задает направление трансляции слов. При $A=0$ выход МБР2 переходит в высокоомное состояние, вследствие чего информация будет передаваться от X через МБР1 к Y , а при $A=1$ — от Y через МБР2 к X .

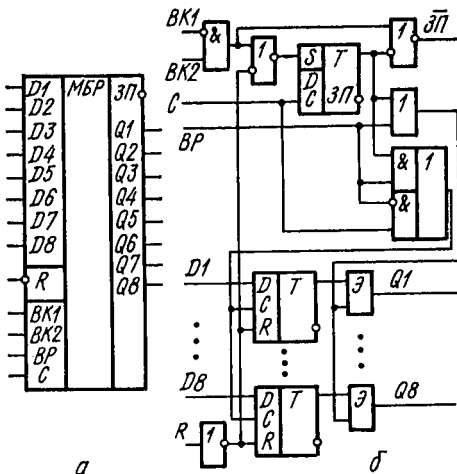


Рис. 6.61

Блок приоритетного прерывания (БПП) предназначен для приема сигналов запроса прерывания от различных устройств, входящих в состав микро-ЭВМ, и выработки сигнала о необходимости перехода к микропрограмме обработки прерывания. По сигналу с БПП на шине ад-

ресу БМУ устанавливается адрес первой микрокоманды этой микроподпрограммы. Размещен БПП в корпусе с 24 выводами и выполняет функции приема и хранения сигналов запроса на прерывание с 8 направлений, формирования кода уровня запроса на прерывание, хранения кода уровня прерывания, которое обрабатывается процессором, выдачи процессору сигнала о запросе на прерывание более высокого уровня по сравнению с обрабатываемым. БПП (рис. 6.63) состоит из регистра запросов на прерывания (РЗП), где каждому из направлений прерыва-

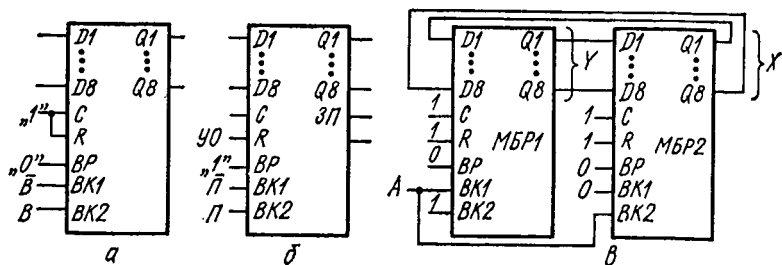


Рис. 6.62

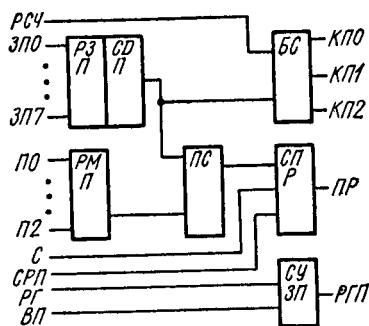


Рис. 6.63

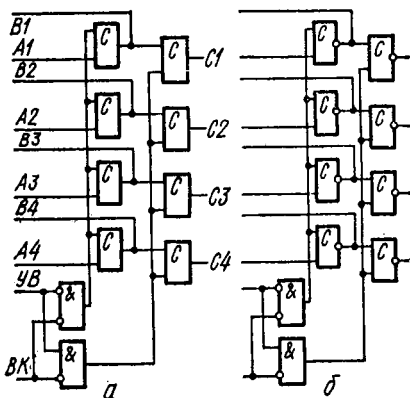


Рис. 6.64

ния соответствует отдельный триггер. Запросам отдельных устройств соответствуют сигналы ЗП0—ЗП7. Регистр маскирования прерывания (РМП) хранит код уровня прерывания, обрабатываемого процессором. Информация на этот регистр поступает по линиям ПО—П2. Шифратор приоритета преобразует пространственный код причины прерывания в трехразрядный двоичный код, который с помощью приоритетной схемы ПС сравнивается с состоянием РМП. Если приоритет пришедшего запроса выше приоритета выполняемой программы (или прерывания) и есть сигнал разрешения прерывания (СРП), то на выходе ПР схемы формирования сигналов прерывания (СПР) устанавливается нулевой сигнал, указывающий процессору о необходимости прерывания. Код уровня очередного запроса на прерывание может быть выдан из БПП через буферную схему по линиям КПО—КП2 по сигналу разрешения РС4. В момент выдачи сигналов ПР осуществляется блокировка дру-

их запросов на прерывание, которая снимается после записи в РМП пода уровня нового прерывания. Помимо перечисленных узлов в БПП ходит также схема для увеличения числа направлений приема запросов на прерывания (СУЗП).

Шинные формирователи (ШФ и ШФИ) обеспечивают развязку входов и выходов устройств, работающих на двунаправленную шину. Размещены эти формирователи в корпусе с 16 выводами и обеспечивают коммутацию 4-разрядных машинных слов по одному из двух направлений (рис. 6.64). Состоят эти БИС из набора повторителей (ШФ) или инверторов (ШФИ), объединенных общим управлением. Структура шинных формирователей допускает параллельное их использование. Для включения ШФ и ШФИ в работу необходимо подать нулевой сигнал по входу выбора кристалла ВК. Этим обеспечивается возможность управления БИС по входу УВ (управление выдачей информации). Данные могут поступать по

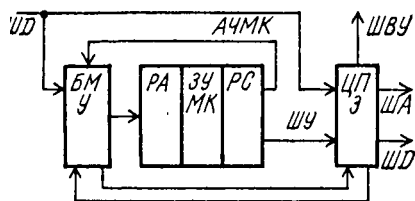


Рис. 6.65

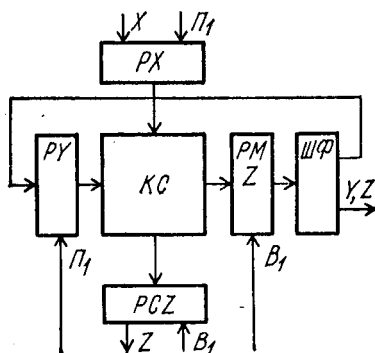


Рис. 6.66

пинам А или В, а выдаваться по шинам В или С. При нулевом сигнале на входе УВ информация со входов В передается на выходы С, а при единичном — со входов А на выходы В. В ШФИ направление передачи информации такое же, только осуществляются сами передачи с инверсией информационных сигналов.

Основными БИС для построения МП являются БМУ, ЦПЭ и одна из микросхем ПЗУ. На базе трех типов БИС может быть реализован простейший 2-разрядный МП по схеме на рис. 6.65. Для построения 6-разрядного МП необходимо использовать 8 отдельных 2-разрядных ЦПЭ, БМУ и стандартное ПЗУ микропрограмм на 512 18-разрядных слов. Такая структура МП обеспечивает минимальные функциональные возможности и быстродействие. При использовании других БИС из МН 3000 удастся почти в два раза увеличить быстродействие МП и значительно расширить функциональные возможности. Так, например, время сложения 16-разрядных чисел может быть уменьшено с 300 нс до 125 нс. Это достигается за счет введения СУП и МБР текущей микрокоманды, позволяющих совмещать циклы выборки и выполнения микрокоманды. Скорость выборки микрокоманд при этом повышается более чем, в два раза (с $3 \cdot 10^6$ до $8 \cdot 10^6$ микрокоманд за секунду).

В связи с постоянным совершенствованием технологии и повышением уровня интеграции микропроцессорных биполярных БИС в на-

стоящее время наметилась тенденция перехода от секционированных МП к 8- и 16-разрядным МП и даже микроконтроллерам. В соответствии с перспективными планами к 1985 году промышленностью будет освоен выпуск 32-разрядных МП.

Рассмотренные однокристалльные и многокристалльные МП предназначены для широкого применения не только в ЦВТ, но и в других отраслях техники, связанных с получением и обработкой информации, и являются в этом смысле универсальными. Однако универсальные МП оказываются малоэффективными при создании на их основе систем для выполнения сложных математических вычислений, так как МП, по существу, являются устройствами логической обработки информации, а не арифметическими процессорами. Сложные арифметические операции универсальные МП выполняют программным путем и затрачивают на них много времени. Например, один из наиболее популярных МП 8080 программным способом реализует операцию умножения двух 16-разрядных слов (с плавающей запятой) за 7 мс, извлечение квадратного корня осуществляет за 77 мс, а вычисление синуса — за 118 мс. Для многих задач указанное время выполнения сложных арифметических операций нельзя считать приемлемым. Уменьшить время выполнения подобных операций в МП можно путем использования более современной и быстродействующей элементной базы, что несколько усложняет технологию производства МП, удорожает их, снижает их помехоустойчивость. Другой путь повышения производительности МП состоит в реализации сложных операций не микропрограммным, а схемным способом. Наконец, третий путь состоит в создании специализированных МП, ориентированных на эффективное выполнение данного класса операций.

Специализированные МП редко используются автономно. Чаще они подключаются к универсальному МП в качестве дополнительного устройства, управляются им и увеличивают вычислительную мощность и возможности микропроцессорной системы. В соответствии с этим различают специализированные МП (СМП) для арифметических вычислений, управления УВВ, сложных логических преобразований, ассоциативного поиска, специализированных вычислений (типа преобразования Фурье) и других целей. Примером СМП является БИС для аппаратного умножения, содержащая комбинационную схему (КС) для умножения 16-разрядных чисел (рис. 6.66) и четыре регистра PX , PY , PCZ , PMZ соответственно для сомножителей X и Y , старших и младших разрядов произведения. Кроме того, в состав СМП входит шинный формирователь ШФ. Шина для ввода Y с целью сокращения числа выводов СМП объединена с шиной младших разрядов произведения через ШФ. Собственно умножение чисел здесь осуществляется за 200 мкс, что в несколько тысяч раз быстрее, чем умножение в универсальном МП по подпрограмме. Однако фактически быстродействие повысится примерно в 10 раз из-за того, что СМП должен выполнять и ряд вспомогательных микроопераций, например, выборку и дешифрацию команды, считывание и передачу операндов.

Для создания высокоэффективных информационно-поисковых систем разработаны ассоциативные СМП, основу которых составляет

ассоциативное ЗУ на 16 16-разрядных слов. Широкое распространение получили периферийные СМП, предназначенные для управления устройствами ввода — вывода. Отличительной особенностью СМП является то, что они разрабатываются с ориентацией на конкретные применения и их логическая структура, аппаратная реализация и система команд наиболее полно соответствуют характеру решаемых задач.

Отечественной промышленностью выпускаются МП различных областей применения. Наиболее низкую стоимость обеспечивает применение серии 536 с 8-разрядным ЦПЭ, построенным по *p*-МОП технологии. Недостатком серии является низкое быстродействие ($t_{ц} = 10$ мкс). По *n*-МОП технологии выпускаются однокристалльные МП (серия 580 с 8-разрядным МП) и однокристалльные микро-ЭВМ (серия 1801 с 16-разрядной микро-ЭВМ) с наиболее высокой степенью интеграции, достигающей 300 тыс. компонент в одном корпусе. Недостатками серий является ограниченная разрядность и среднее быстродействие ($t_{ц} = 2$ мкс). По ЭСЛ — технологии выпускаются сверхбыстродействующие ($t_{ц} = 30$ нс) БИС серии 1800 с 4-разрядным ЦПЭ для высокопроизводительных ЭВМ серии ЕС. Недостатками серии является большое энергопотребление и низкая степень интеграции. По ТТЛШ технологии выпускаются быстродействующие ($t_{ц} = 100$ нс) МП с наиболее широкими функциональными возможностями. Дополняющие друг друга серии 589, 1802 и 1804 этого типа содержат более 20 БИС, среди которых имеются матрицы умножения 16×16 разрядов, ассоциативные ЗУ и программируемые логические матрицы. Недостатками серии являются высокое энергопотребление и низкая степень интеграции. По КМОП технологии выпускаются МП 587 и 588 серий с рекордно низким энергопотреблением. Недостатком серии является невысокое быстродействие ($t_{ц} = 5$ мкс). По ИИЛ технологии выпускаются серии 582, 583 и 584, которые уступают *p*-МОП сериям по стоимости, *n*-МОП — по степени интеграции, ЭСЛ — сериям по быстродействию, КМОП — по потреблению энергии. В то же время ИИЛ серии не имеют и недостатков, присущих перечисленным технологиям. Это делает эффективным их применение в тех областях, где не требуется рекордных отдельных показателей, а необходимы средние показатели по многим характеристикам БИС.

Рассмотрим ЦПЭ 583ВС1, построенный по схеме многокристального МП и состоящий из АЛУ, ЗУ в виде 16 регистров общего назначения (РОН), УУ, реализованного на основе комбинационной схемы, и вспомогательных регистров результата (РР), управления (РУ), состояния (РС), микрокоманды (РМ) и двух буферных регистров (*РБ1* и *РБ2*). Регистры РМ и РС состоят из 16 и 2 разрядов, а остальные — из восьми. Кроме описанных регистров других запоминающих элементов в микросхеме (МС) нет. Обмен информацией осуществляется по двум двунаправленным шинам данных (*ШД1* и *ШД2*), шине МК (*ШМ*) и шине управления, включающей в себя ряд одно- и двунаправленных выводов, служащих для ввода или вывода управляющих сигналов, а также для каскадирования МС при увеличении разрядности, кратной 8 (табл. 133).

Номера выводов	Наименование и назначение выводов МС 583BC1
1—4	Входы синхросигналов приема ($S1$) и выполнения ($S2$) микрокоманды и выдачи результата на ШД1 ($S31$) и ШД2 ($S32$)
5—7, 35—47	Шина микрокоманд (ШМ)
8—23	Шины данных ШД1 и ШД2
24	Шина нулевого потенциала
25, 48	Гальванически связанные выводы, служащие для подключения источника питания
26 и 33	Правый (R) и левый (L) двунаправленные выводы сдвигателя АУ
27	Вход C сигнала переноса сумматора АУ
28	Выход признака нуля результата (признак Z)
29	Многофункциональный выход, служащий для индикации переполнения разрядной сетки или переноса из старшего разряда сумматора АУ
30, 31	Выходы сигналов распространения P и образования G группового переноса
32	Многофункциональный выход A , индицирующий состояние знакового разряда или разряда, выходящего при сдвигах за пределы разрядной сетки (выдвигаемого разряда)
34	Вход K , задающий позицию МС

МП работает с числами, представленными в дополнительном коде. Для указания знака отводится один старший разряд, который отделяется от мантиссы фиксированной запятой. Выделяется знаковый разряд сигналом K на входе 34. При $K = 1$ все разряды чисел в данной МС обрабатываются как разряды мантиссы. При $K = 0$ старший разряд обрабатывается как знаковый, а все остальные — как разряды мантиссы. Отличие в обработке знакового разряда проявляется при переполнении разрядной сетки и сдвигах. Под переполнением понимается выход результата за пределы диапазона $[-1, 1]$. Это возможно при сложении на сумматоре чисел одного знака, сумма которых по модулю больше 1. Переполнение разрядной сетки сопровождается появлением сигнала $q = 1$ (выход 29), который формируется по правилу

$$q = \bar{K} (q_0 \bar{q}_1 \vee \bar{q}_0 q_1) \vee K q_0,$$

где q_0 — перенос из знакового разряда; q_1 — перенос из старшего разряда мантиссы в знаковый разряд. Так как переполнение связано с ошибочным изменением знака, то МП восстанавливает его.

Другое отличие в обработке знакового разряда и разрядов мантиссы проявляется при сдвигах. При левом арифметическом сдвиге (ЛАС) в старшей МС ($K = 0$) знаковый разряд отключается от остальных разрядов и в сдвиге не участвуют, в младший разряд младшей МС ($K = 1$) записывается 0, а старший разряд мантиссы старшей МС ($K = 0$), выходящий за пределы разрядной сетки после завершения сдвига (выдвигаемый разряд), выдается на выход A . При правом арифметическом сдвиге (ПАС) цифра в знаковом разряде не изменяется, в старший разряд мантиссы записывается

цифра из знакового разряда, а на выход A выдается цифра из младшего (выдвигаемого) разряда. Для формирования признака A и выполнения сдвигов входы и выходы МС должны быть соответствующим образом соединены, а именно: вывод R старшей МС соединяется с выводом L соседней младшей МС; вывод L самой старшей МС ($K=0$) соединяется с выводом R самой младшей МС, выводы A этих двух МС объединяются в одну точку с целью выделения выдвигаемого при сдвиге разряда.

При выполнении операций в МС формируются три признака — q , A и Z , которые присутствуют на соответствующих выходах либо до приема следующей МК, либо до изменения кода на ШД1 или ШД2.

Управляет работой МС ВС1 16-разрядная МК, состоящая из четырех полей — $П1$, $П2$, I и J . Пятиразрядное поле $П1$ (разряды $m_0 — m_4$) задает одну из 32 операций, выполняемых МП (табл. 134).

Таблица 134

$m_2 \ m_3 \ m_4$	$m_0=m_1=0$	$m_0=\bar{m}_1=0$	$m_0=\bar{m}_1=1$	$m_0=m_1=1$
	Арифметические операции $f_i - C_i$	Логические операции	Операции счета и сложения со сдвигом, загрузка РС и пустая операция	
0 0 0	1	ab	ЛЛС ($A + C$)	ЛЦС ($B + C$)
0 0 1	$\bar{a} + b$	$\bar{a}\bar{b}$	ПЛС ($A + C$)	ПЦС ($B + C$)
0 1 0	$a + \bar{b}$	$a\bar{b}$	ЛЦС ($A + C$)	ЛАС ($B + C$)
0 1 1	$a + b$	$a \vee \bar{b}$	ПЦС ($A + C$)	ПАС ($B + C$)
1 0 0	$\bar{a}$	$\bar{a} \vee b$	ЛАС ($A + C$)	ЛЛС ($A + B + C$)
1 0 1	$\bar{a}$	$\bar{a} \vee \bar{b}$	ПАС ($A + C$)	ПЛС ($A + B + C$)
1 1 0	b	$\bar{a}\bar{b} \vee a\bar{b}$	ЛЛС ($B + C$)	Загрузка РС
1 1 1	$\bar{b}$	$ab \vee \bar{a}\bar{b}$	ПЛС ($B + C$)	Нет операции

Арифметические операции позволяют производить записи кодов 1, 0 и операндов A , $\bar{A}$, B и $\bar{B}$, поступающих с ШД1 и ШД2. Суммирование прямых кодов и инверсных кодов позволяют реализовать операции сложения и вычитания типа $A + B$, $A - B$, $B - A$. Логические операции выполняются при блокировке переносов между разрядами. Операции сдвига можно совмещать при $C=1$ с операцией счета. При выполнении рассмотренных операций всегда формируется признак Z . Формирование остальных признаков зависит от выполняемой операции и определяется табл. 135, в которой 0 означает, что признак не формируется.

Поле $П2$ определяет принятый в данном МК способ адресации операндов и результата (табл. 136). Например, в МК типа RR операнды A и B извлекаются из РОН по адресам I и J , а результат P заносится в ЗУ РОН по первому адресу. В команде RL1 операнд A поступает с PP, а B — с ШД2. Результат заносится в РОН по адресу I . Адреса I и J РОН формируются с помощью полей I и J РМ или РУ в зависимости от содержимого РС. Первый разряд РС

Таблица 135

$m_0 \ m_1 \ m_2 \ m_3 \ m_4$	А		q	
	K=0	K=1	K=0	K=1
0 0 0 0 0	Знак	0	0	Перенос
0 0 X X 1 0 0 X 1 X 0 0 1 X X	Знак	0	Переполнение	Перенос
0 1 X X X	0	0	0	0
1 0 X X 0 1 X 0 X 0 1 X X 0 0	Выдвигаемый разряд	0	Переполнение	Перенос
1 0 X X 1 1 X 0 X 1 1 X X 0 1	0	Выдвигаемый разряд	Переполнение	Перенос
1 1 1 1 X	Зависит от предыдущей МК			

Таблица 136

Поле $P2$			Тип операции	$P = A (A \cup) B$		
m_5	m_6	m_7		P	A	B
0	0	0	RR	PI	PI	PJ
0	0	1	RL	PI	PI	$\text{ШД}2$
0	1	0	$RR1$	PI	PP	PJ
0	1	1	$RL1$	PI	PP	$\text{ШД}2$
1	0	0	$RR2$	PP	PI	PJ
1	0	1	LL	PP	$\text{ШД}1$	$\text{ШД}2$
1	1	0	$RR3$	PP	PP	PJ
1	1	1	$RL2$	PP	$\text{ШД}1$	PJ

управляет формированием адреса *I*, а второй — *J*. По 0 в разряде РС соответствующий ему адрес извлекается из РМ, а по 1 — из первых или последних разрядов РУ, в котором фиксируются и хранятся соответствующие поля предыдущей МК. Запись же кода в РС производится МК загрузки (*П1* = 11110), в которой после поля *П1* указывается два подлежащих записи в РС бита, а остальные поля не используются.

Каждый цикл работы МП начинается с приема МК по нарастающей *S1*. Для этого на ШМ надо выставить МК за 50 нс и удерживать ее в течение 100 нс. По этому же перепаду в РС и РУ записываются коды, подготовленные БУ при выполнении предыдущей МК. В дальнейшем состояние ШМ не влияет на работу МП. Принятая на РМ МК дешифрируется за 150 нс. Далее по спаду сигнала *S2*

происходит запись информации с РОН или РР на *РБ1* и *РБ2*. При выполнении команд типа *RL*, *RL1*, *RL2*, и *LL* в АУ поступают коды непосредственно с *ШД1* или *ШД2*. Коды с *РБ1*, *РБ2*, *ШД1* или *ШД2* поступают на АУ и за 450 нс происходит выполнение операции, указанной в поле *П1* МК. По нарастанию *S2* полученный в АУ результат заносится в РОН или РР. Выдача информации в *ШД1* или *ШД2* может быть произведена только с *РБ1* или *РБ2* по сигналам *S31*, *S32*, которые никакими временными соотношениями сигналами *S1* и *S2* не связаны и могут подаваться в произвольные моменты времени. Так как результат данной МК фиксируется в РР или РОН, а не в *РБ1* или *РБ2*, то указанный результат может быть выдан на *ШД1* или *ШД2* только при выполнении следующей МК. При выполнении МК типа *RL*, *RL1*, *RL2* и *LL* данные на *ШД1* и *ШД2* должны быть установлены за 400 нс до фиксации результата операции сигналом *S2* и удерживаться на них в течение 100 нс после фиксации результата. Управляя подачей сигналов *S1* и *S2*, можно расширить функциональные возможности МС и кроме режима нормальной работы (подаются *S1* и *S2*) получить еще три других режима. При запрете *S1* новая МК не принимается, а исполняется предыдущая. При запрете *S2* новая МК принимается, но содержимое регистров *РБ1* и *РБ2* не изменяется и результат операции на регистрах МС не фиксируется. При запрете *S1* и *S2* состояние регистров МС не изменяется. При всех описанных режимах признаки *A*, *q* и *Z* могут изменяться либо в зависимости от вновь принятой МК, либо от изменений кодов на *ШД1* или *ШД2*.

СПИСОК ЛИТЕРАТУРЫ

1. Авах Ю. А., Фатин В. К. Одно-сторонние запоминающие устройства.— М.: Энергия, 1981.— 192 с.
2. Алексенко А. Г. Основы микросхемотехники.— М.: Сов. радио, 1977.— 405 с.
3. Алферова З. В. Теория алгоритмов.— М.: Статистика, 1973.— 164 с.
4. Атовмян И. Г. Устройство управления и организация работы ЦВМ.— М.: МИФИ, 1971.— 118 с.
5. Байков В. Д., Смоллов В. Б. Аппаратная реализация элементарных функций в ЦВМ.— Л.: Изд-во ЛГУ, 1975.— 96 с.
6. Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы.— М.: Радио и связь, 1981.— 328 с.
7. Баранов С. И. Синтез микропрограммных автоматов.— Л.: Энергия, 1979.— 232 с.
8. Березенко А. И., Корягин Л. Н., Назарян А. Р. Микропроцессорные комплекты повышенного быстродействия.— М.: Радио и связь, 1981.— 168 с.
9. Брик Е. А. Техника ПЗУ.— М.: Сов. радио, 1973.— 240 с.
10. Букреев И. Н., Мансуров Б. М., Горячев В. И. Микроэлектронные схемы цифровых устройств.— М.: Сов. радио.— 1975.— 368 с.
11. Гаврилов Ю. В., Пучко А. Н. Арифметические устройства быстродействующих ЭЦВМ.— М.: Сов. радио, 1970.— 280 с.
12. Глушков В. М. Синтез цифровых автоматов, М.: Физматгиз, 1962.— 458 с.
13. Глушков В. М. и др. Рекурсивные машины и вычислительная техника.— Киев: Ин-т кибернетики, 1974.— 14 с.— (Препринт).
14. Дроздов Е. А., Комарницкий В. А., Пятибратов А. П. Электронные вычислительные машины Единой Системы.— М. Машиностроение, 1976.— 672 с.
15. Евреинов Е. В. Однородные вычислительные системы, структуры и среды.— М.: Радио и связь, 1981.— 208 с.
16. Единая система ЭВМ. / Под ред. А. М. Ларионова.— М.: Статистика, 1974 г.— 135 с.
17. Журавлев Ю. П. Системное проектирование управляющих ЦВМ.— М.: Сов. радио, 1974.— 368 с.
18. Каган Б. М. Электронные вычислительные машины и системы.— М.: Энергия, 1979.— 528 с.
19. Каляев А. В. Однородные коммутационные регистровые структуры.— М.: Сов. радио, 1978.— 336 с.
20. Карцев М. А. Арифметика цифровых машин.— М.: Наука, 1969.— 575 с.
21. Карцев М. А., Брик В. А. Вычислительные системы и синхронная арифметика.— М.: Радио и связь, 1981.— 360 с.
22. Корнейчук В. И. Запоминающие устройства ЦВМ.— Киев: Техника, 1976.— 168 с.
23. Королев Л. Н. Структуры ЭВМ и их математическое обеспечение.— М.: Наука, 1978.— 351 с.
24. Крайзмер Л. П. Устройства хранения дискретной информации.— Л.: Энергия, 1969.— 312 с.
25. Лысков Б. Г. Арифметические и логические основы цифровых автоматов.— Минск: Высшая школа, 1980.— 336 с.
26. Майоров С. А., Новиков Г. И. Принципы организации цифровых машин.— Л.: Машиностроение. 1974.— 431 с.
27. Майоров С. А., Крутовских С. А., Смирнов А. А. ЭВМ. Справочник по конструированию.— М.: Сов. радио, 1975.— 504 с.
28. Малиновский Б. Н., Брюхович Е. И., Денисенко Е. Л. и др. Справочник по цифровой вычислительной технике (процессоры и память).— Киев: Техника, 1979.— 366 с.
29. Малиновский Б. Н., Яковлев Ю. С., Маковенко Е. Т. Анализ и синтез мини-ЭВМ.— Киев: Наук. думка, 1976.— 174 с.
30. Микропроцессорные комплекты интегральных схем: состав и структура; справочник /Под ред. А. А. Васенко-

- ва, В. А. Шахнова.— М.: Радио и связь, 1982.— 192 с.
31. *Наумов Б. Н. и др.* Малые ЭВМ и их применение.— М.: Статистика, 1980.— 231 с.
32. *Оранский А. М.* Аппаратные методы в цифровой вычислительной технике.— Минск: Изд-во Белорус. ун-та, 1977.— 208 с.
33. *Основы построения технических средств ЕС ЭВМ на интегральных микросхемах.* /Под ред. В. Н. Файзулаева.— М.: Радио и связь, 1981.— 288 с.
34. *Полупроводниковые запоминающие устройства и их применение.* /Под ред. А. Ю. Гордонова.— М.: Радио и связь, 1981.— 344 с.
35. *Поспелов Д. А.* Арифметические основы вычислительных машин дискретного действия.— М.: Высш. школа, 1970.— 307 с.
36. *Поспелов Д. А.* Логические методы анализа и синтеза схем.— М.—Л.: Энергоиздат, 1981.— 88 с.
37. *Потемкин И. С.* Автоматизация синтеза функциональных схем.— М.: Энергоиздат, 1981.— 88 с.
38. *Прангишвили И. В.* Микропроцессоры и микро-ЭВМ.— М.: Энергия, 1979.— 232 с.
39. *Преснухин Н. Н., Нестеров П. В.* Цифровые вычислительные машины.— М.: Высш. школа, 1974.— 415 с.
40. *Пржиалковский В. В., Ломов Ю. С.* Технические и программные средства единой системы ЭВМ (ЕС ЭВМ-2).— М.: Статистика, 1980.— 232 с.
41. *Проектирование цифровых вычислительных машин.* /Под ред. С. А. Майорова.— М.: Высш. школа, 1972.— 344 с.
42. *Пухов Г. Е., Евдокимов В. Ф., Синьков М. В.* Разрядно-аналоговые вычислительные системы.— М.: Сов. радио, 1978.— 255 с.
43. *Пятлин О. А., Овсищев П. И., Лазер И. М. и др.* Проектирование микроэлектронных цифровых устройств.— М.: Сов. радио, 1977.— 272 с.
44. *Рабинович З. Л.* Элементарные операции в вычислительных машинах.— Киев: Техніка, 1966.— 304 с.
45. *Рабинович З. Л., Раманаускас В. А.* Типовые операции в вычислительных машинах.— Киев: Техніка, 1980.— 264 с.
46. *Рыжков В. А., Сергеев Н. П., Раков Б. М.* Внешние ЗУ на магнитном носителе.— М.: Энергия, 1978.— 224 с.
47. *Савельев А. Я.* Арифметические и логические основы цифровых автоматов.— М.: Высш. школа, 1980.— 255 с.
48. *Самофалов К. Г., Корнейчук В. И., Романкевич А. М., Тарасенко В. П.* Цифровые многозначные элементы и структуры.— Киев: Вища школа, 1974.— 168 с.
49. *Самофалов К. Г., Корнейчук В. И., Тарасенко В. П.* Электронные цифровые вычислительные машины.— Киев: Вища школа, 1976.— 480 с.
50. *Самофалов К. Г., Корнейчук В. И., Городний А. В.* Структурно-логические методы повышения надежности ЗУ.— М.: Машиностроение, 1976.— 112 с.
51. *Самофалов К. Г., Кухарчук А. Г., Луцкий Г. М.* Структуры ЦВМ четвертого поколения.— Киев: Техніка, 1972.— 254 с.
52. *Самофалов К. Г., Луцкий Г. М.* Структуры и организация функционирования ЭВМ и систем.— Киев: Вища школа, 1976.— 391 с.
53. *Смолов В. Б.* Функциональные преобразователи информации.— Л.: Энергоиздат, 1981.— 248 с.
54. *Соловьев Г. Н.* Арифметические устройства ЭВМ.— М.: Энергия, 1978.— 177 с.
55. *Старос Ф. Г., Крайзмер Л. П.* Полупроводниковые интегральные запоминающие устройства.— М.—Л.: Энергия, 1973.— 112 с.
56. *Стахов А. П.* Введение в алгоритмическую теорию измерения.— М.: Сов. радио, 1977.— 287 с.
57. *Тарабрин Б. В., Якубовский С. В., Барканов Н. А. и др.* Справочник по интегральным микросхемам.— М.: Энергия, 1980.— 816 с.
58. *Хассон С.* Микропрограммное управление: В 2-х т. М.: Мир, 1973.— Т. 1.— 224 с.; т. 2.— 477 с.
59. *Шамаев Ю. М., Ненашев А. П., Дзегленок И. И.* Основы построения цифровых вычислительных машин 3-го поколения.— М.: МЭИ, 1977.— 80 с.
60. *Шигин А. Г., Дерюгин А. А.* Цифровые вычислительные машины (память ЦВМ).— М.: Энергия, 1975.— 536 с.
61. *Шоломов Л. А.* Основы теории дискретных логических и вычислительных устройств.— М.: Наука, 1980.— 400 с.
62. *Яблонский С. В.* Введение в дискретную математику. М.: Наука, 1979.— 272 с.

ОГЛАВЛЕНИЕ

Предисловие	3

Глава 1

ИНФОРМАЦИОННЫЕ ОСНОВЫ ЦИФРОВОЙ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

1.1. Краткие исторические сведения. Информация. Единицы информации	5
1.2. Основные понятия теории алгоритмов	8
1.3. Принципы построения и классификация цифровых ЭВМ	12
1.4. Системы счисления и способы перевода чисел из одной системы счисления в другую	17
1.5. Представление отрицательных чисел в цифровых ЭВМ	26
1.6. Формы представления чисел в цифровых ЭВМ	30
1.7. Двоично-кодированные и другие системы счисления	33

Глава 2

СХЕМОТЕХНИЧЕСКИЕ ОСНОВЫ ЦИФРОВОЙ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

2.1. Основы теории переключательных функций	39
2.2. Канонические формы представления переключательных функций	47
2.3. Минимизация переключательных функций	52
2.4. Синтез комбинационных схем	56
2.5. Задачи анализа и синтеза конечных автоматов	64
2.6. Элементарные автоматы. Методы структурного синтеза автоматов	71
2.7. Системы элементов цифровых ЭВМ. Их классификация и основные характеристики	80
2.8. Интегральные системы элементов ЦВТ	83
2.9. Триггеры в схемах ЦВТ	94

Глава 3

ОПЕРАЦИОННЫЕ УЗЛЫ ЦИФРОВЫХ ЭВМ И ОСНОВЫ ИХ ПРОЕКТИРОВАНИЯ

3.1. Общие вопросы построения операционных узлов цифровых ЭВМ	113
3.2. Дешифраторы	116
3.3. Схемы сравнения	123
3.4. Шифраторы и сдвигатели	129
3.5. Комбинационные сумматоры	136
3.6. Организация цепей переноса в параллельных сумматорах	145
3.7. Преобразователи кодов	151
3.8. Другие операционные узлы комбинационного типа	160
3.9. Регистры	164
3.10. Счетчики	171
3.11. Кольцевые счетчики и накапливающие сумматоры	182

Глава 4

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА ЦИФРОВЫХ ЭВМ

4.1. Назначение, классификация и основные параметры запоминающих устройств	190
4.2. Структурные схемы адресных ЗУ с М-поиском	192
4.3. Структурные схемы ЗУ с В-поиском	200
4.4. Накопители ЗУ на ферритовых сердечниках	204
4.5. Накопители ЗУ на тонких магнитных пленках	212
4.6. Полупроводниковые накопители ЗУ	217
4.7. Накопители на приборах с зарядовыми связями и магнитных доменных элементах	222
4.8. Постоянные и полупостоянные запоминающие устройства	221
4.9. Табличные и таблично-алгоритмические вычислительные устройства на основе ПЗУ	23
4.10. Ассоциативные запоминающие устройства	24
4.11. Обработка информации в ЗУ	24
4.12. Принципы построения и разновидности ЗУ с магнитной записью	25
4.13. Структура памяти цифровых ЭВМ. Системы адресации	25
4.14. Структуры СОЗУ и ОЗУ	26
4.15. Выборка в ЗУ слов переменной длины	26
4.16. Некоторые структурные способы повышения быстродействия ЗУ	266
4.17. Перспективные для ЦВТ системы памяти	271

Глава 5

АРИФМЕТИКО-ЛОГИЧЕСКИЕ УСТРОЙСТВА И ОПЕРАЦИОННЫЕ БЛОКИ ЦИФРОВЫХ ЭВМ

5.1. Назначение, классификация, состав и основные характеристики арифметико-логических устройств	274
5.2. Блоки для сложения и вычитания чисел в прямых и дополнительных кодах	277
5.3. Блоки для сложения и вычитания чисел с плавающей запятой	282
5.4. Основные алгоритмы и структуры операционных блоков для умножения в прямых кодах	288
5.5. Методы повышения быстродействия блоков умножения. Логические способы ускорения умножения	295
5.6. Аппаратные методы ускорения умножения	300
5.7. Другие алгоритмы и блоки умножения в цифровых ЭВМ	309
5.8. Основные алгоритмы и структуры операционных блоков для деления в прямых кодах	319
5.9. Методы повышения быстродействия блоков деления	323

5.10. Другие алгоритмы и блоки деления в цифровых ЭВМ	335
5.11. Операционные блоки для редких и вспомогательных операций	338
5.12. Организация и структуры арифметико-логических устройств	347
5.13. Расчет основных параметров АЛУ	358

Глава 6

УСТРОЙСТВА УПРАВЛЕНИЯ ЦИФРОВЫМИ ЭВМ И ОРГАНИЗАЦИЯ ПРОЦЕССОРОВ

6.1. Структура и кодирование команд цифровых ЭВМ. Функции устройства управления	366

6.2. Классификация блоков управления операциями. Способы построения схемных БУО	376
6.3. Микропрограммные блоки управления операциями	390
6.4. Способы построения блоков управления командами	406
6.5. Повышение производительности однопроцессорных цифровых ЭВМ	413
6.6. Микропроцессоры, микропроцессорные наборы и микро-ЭВМ	426
6.7. Структуры однокристалльных и многокристалльных микропроцессоров	435
Список литературы	452

Константин Григорьевич Самофалов,
Виктор Иванович Корнейчук,
Владимир Петрович Тарасенко

ЦИФРОВЫЕ ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ

ТЕОРИЯ
И ПРОЕКТИРОВАНИЕ

ИЗДАНИЕ ВТОРОЕ,
ПЕРЕРАБОТАННОЕ И ДОПОЛНЕННОЕ

Редактор Ж. Г. Давиденко
Переплет художника Г. М. Балюна
Художественный редактор С. П. Духленко
Технический редактор И. И. Каткова
Корректор Н. В. Волкова

Информ. бланк № 6301

Сдано в набор 20.08.82. Подп. в печать 30.06.83.
БФ 02692. Формат 60×90^{1/16}. Бумага типогр. № 3.
Лит. гарн. Вис. печать. 28,5 печ. л. 28,5 кр.-отт.
31,42 уч.-изд. л. Тираж 9000 экз. Изд. № 6038.
Зак. № 2-461. Цена 1 р. 50 к.

Головное издательство издательского объединения
«Вища школа», 252054, Киев-54, ул. Гоголевская, 7

Книжная фабрика им. М. В. Фрунзе, 310057, Харь-
ков-57, Донец-Захаржевского, 6/8.