Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

И.П. Кобяк

ПРОЦЕССОРЫ КОМПЬЮТЕРНЫХ СИСТЕМ. СИНТЕЗ ОПЕРАЦИОННЫХ АВТОМАТОВ

Методическое пособие

по курсовому и дипломному проектированию
по ТиП ЭВМ и СиФО ЭВМ для студентов специальности
40 02 01 «Вычислительные машины, системы и сети»
дневной формы обучения

Минск 2003

УДК 681.3 (075.8)
ББК 32.973-04 я 73
К 55

Р е ц е н з е н т :
доцент кафедры ЭВМ БГУИР, канд. техн. наук Ю.А. Луцик

Кобяк И.П.

К 55 Процессоры компьютерных систем. Синтез операционных автоматов: Метод. пособие по курсовому и дипломному проектированию по ТиП ЭВМ и СиФО ЭВМ для студентов специальности 40 02 01 «Вычислительные машины, системы и сети» дневной формы обучения / И.П. Кобяк. – Мн.: БГУИР, 2003. – 83 с.: ил.

ISBN 985-444-470-8.

Методическое пособие предназначено для ознакомления студентов с принципами проектирования операционных автоматов, а также с системными вопросами организации процессоров и блоков обработки данных. В пособии рассматриваются МПК К1804 и принципы использования комплекта при построении БОД, вопросы синтеза операционных автоматов и примеры применения методик для построения конкретных устройств. Представленные материалы могут быть использованы студентами для самостоятельного выполнения курсовых и дипломных проектов, а также для подготовки к экзамену по профилирующей специальности.

УДК 681.3 (075.8)
ББК 32.973-04 я 73

ISBN 985-444-470-8

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

1.1. Принцип действия ЭВМ

Классическая структура компьютера содержит в своем составе следующие блоки:

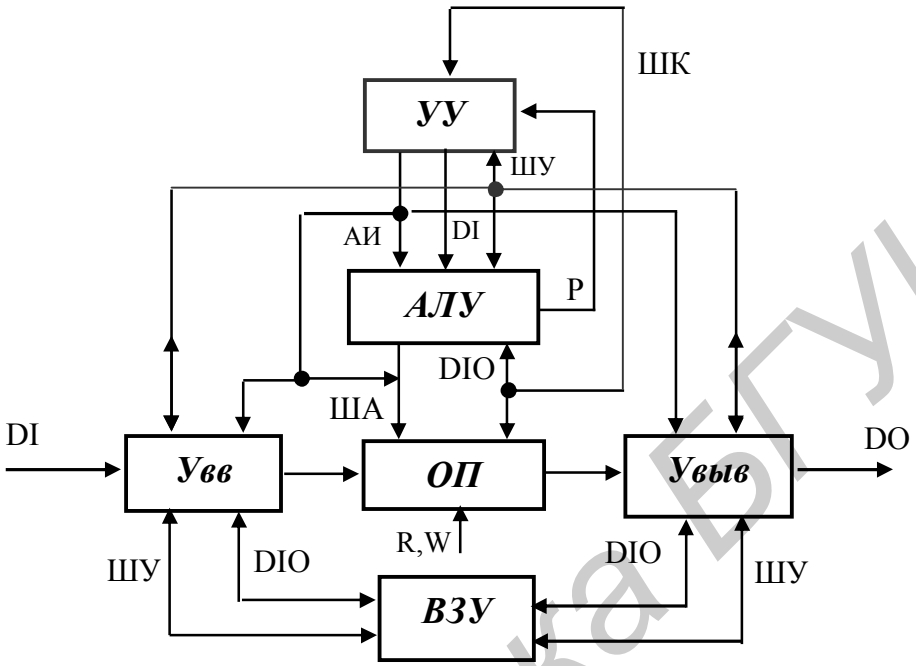

Рис. 1

Структурные компоненты приведенной системы имеют следующее функциональное назначение.

Арифметико-логическое устройство (АЛУ) предназначено для выполнения арифметических и логических операций над числами, представленными в формате с ПЗ или ФЗ. Кроме данных в АЛУ могут обрабатываться адреса (адресная информация), команды (например, преобразование форматов), признаки результатов и другая двоичная информация.

Устройство управления (УУ) предназначено для автоматического управления вычислительным процессом путем посылки всем блока компьютера сигналов, предписывающих те или иные действия. В частности, УУ всегда указывает на:

1) источники информации для АЛУ;

2) функцию, выполняемую АЛУ;

3) приемник результатов, полученных при обработке данных.

ОП и ВЗУ – это память ЭВМ – предназначена для хранения информации, поступающей в компьютер извне, информации, выводимой на внешние носители (печать, монитор и т. д.). Эти блоки также предназначены для хранения программ, результатов промежуточных расчетов и другой машинной информации.

Оперативная память (ОП) состоит из определенного число ячеек, каждая из которых предназначена для хранения машинного слова. Основными характеристиками ОП считают время обращения и емкость памяти. При этом под временем обращения понимают время, необходимое для записи или считывания единицы информации из любой ячейки.

Устройство ввода (Увв) обеспечивает считывание информации с внешних носителей и представление ее в форме электрических сигналов.

Устройство вывода (Увыв) преобразует кодовую информацию, поступающую из памяти или других блоков машины, в форму, необходимую для обмена с внешней средой.

На рис.1 изображены следующие шины: DIO – двунаправленная шина данных, ШУ – шина управления, ША – шина адреса, DI – входные данные, DO – выходные данные, АИ – адресная информация, R, W – сигналы чтения\записи.

Другой вариант схемы предполагает представление компьютера в виде трехшинной модели (рис. 2).

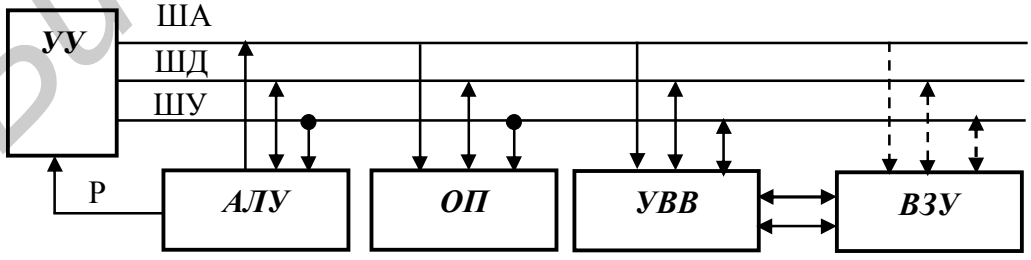

Рис. 2

Время обращения к ОП (или ОЗУ – оперативное запоминающее устройство) в задачах системотехники называют циклом обращения к памяти.

Непосредственно в вычислительном процессе участвует только ОП. Внешнее ЗУ (ВЗУ) служит для хранения больших массивов информации и обмена данными с ОП на различных этапах вычислений.

В общем случае для решения задачи на компьютере пользователь должен задать алгоритм ее решения в виде программы. При этом под алгоритмом понимают последовательность действий, которые необходимо выполнить (над данными) для получения решения задачи. В результате программа машины может интерпретироваться как алгоритм, представленный в терминах системы команд компьютера.

1.2. Неймановские принципы программного управления

Принципы Джона фон Неймана, предложенные в 1945 г., используются в качестве основных для построения большинства современных цифровых компьютеров.

Главная неймановская идея заключается в реализации на некоторых аппаратных носителях принципа хранимой программы, представленной двоичными кодами, различающимися по способу использования в вычислительном устройстве.

Рассмотрим 5 основных принципов организации вычислений, предложенных Джоном фон Нейманом, на начальных этапах создания вычислительной техники.

Первый принцип – использование двоичной системы счисления в качестве базовой. В соответствии с данным принципом обрабатываемая и управляющая информация кодируется двоичной системой счисления и разделяется на блоки, называемые машинными словами.

Использование двоичной системы определяется следующими преимуществами:

1. Числовая информация в ЭВМ отождествляет состояние физических элементов, применяемых в устройствах или блоках машины для хранения и преобразования данных. Причем простота управления переключением этих схем очевидна.

2. Выполнение арифметических и логических операций в компьютере базируется на известных законах булевой алгебры. Наработка этих законов и простота реализации вычислений в двоичной системе общеизвестны и ведут отсчет еще с позапрошлого века.

3. Система счисления с основанием, равным двум, близка к оптимальной.

4. В литературе имеются исследования, доказывающие, что двоичная система имеет быстродействие примерно на 26% большее, чем любая другая. И вообще, из всех систем счисления двоичная система самая быстродействующая.

5. Аппаратура и информация, использующие двоичное кодирование, имеют наибольшую помехозащищенность.

В общем случае, если задачу выбора системы счисления обосновать с точки зрения организации живой материи, то можно сделать аналогичный вывод. Это следует из того, что процесс деления клетки – основного элемента организованной материи – имеет явно выраженную двоичную зависимость. Причем мощность этого процесса столь велика, что определяет воспроизводство всего живого на Земле.

2-й принцип. Информационные слова, циркулирующие в компьютере, различаются по способу использования, но не способом кодирования, т.е. согласно принципу хранимой программы, команды закодированы и хранятся в памяти в виде кодов наравне с данными. Это позволяет каждую команду использовать в программе многократно и, кроме того, над командами, как и над данными, выполнять различные преобразования, т.е. при необходимости модифицировать команды.

3-й принцип. Машинные слова размещаются в ячейках памяти в виде двоичных кодов и идентифицируются номерами, называемыми адресами памяти.

Как правило, в качестве машинной памяти используются схемы ЗУ с произвольной выборкой. Создание более эффективных систем (например, ассоциативной памяти) позволяет реализовать безадресные обращения, что в общем случае свойственно памяти человека. Однако стоимость проектирования таких схем существенно велика, хотя при наличии отлаженных технологий процесс создания встроенных специальных ЗУ используется достаточно часто.

4-й принцип заключается в организации программного управления работой машины. Данный принцип предполагает, что алгоритм решения задачи состоит из отдельных команд, каждая из которых предписывает определенные действия. Команда включает в свой состав определенное число разрядов, кодирующих: 1) выполняемую операцию, 2) адреса источников операндов, 3) приемники результата и другую информацию. Например:

КОП	A1	A2	A3	…

5-й принцип – принцип последовательной отработки команд. В процессе выполнения программы порядок исполнения команд определяется алгоритмом решаемой задачи. Первой выполняется команда, заданная начальным адресом. Адрес очередной команды определяется в процессе выполнения текущей. Данный принцип нарушается, если компьютер имеет архитектуру, отличную от неймановской, т.е. имеется, например, возможность параллельного исполнения команд в процессоре с несколькими АЛУ и т.д. Таким образом, в процессе проектирования компьютеров, как правило, осуществляется синтез неймановских архитектур. Однако в спецсистемах, предназначенных для быстрых вычислений, при введении различных форм параллелизма возможно нарушение 3-го и 5-го принципов.

Нарушение других принципов, очевидно, приведет к созданию вычислительных устройств, функционирующих на других физических основах.

Основными характеристиками компьютеров считаются:

1. Производительность, т.е. скорость выполнения контрольных или измерительных смесей команд. В общем случае для определения производительности используют смеси двух типов, а именно синтетические (или полусинтетические) и профильные.

Синтетические смеси представляют собой отобранные специальным образом процедуры с наибольшим удельным весом команд, соответствующих данному классу задач.

Профильные смеси представляют собой набор фрагментов реальных программ, работающих в единой операционной среде. Они используются для определения производительности компьютеров на проблемно-ориентированных задачах.

2. Стоимость – это показатель, включающий в себя расходы на эксплуатацию и обслуживание компьютера.

На практике указанные две характеристики могут быть использованы для построения диаграмм вида стоимость/производительность. Эти диаграммы при покупке вычислительной техники позволяют выбрать оптимальный или квазиоптимальный для данного класса задач вариант системы. Следует, однако, помнить, что индекс стоимость/производительность не включают в свой состав расходы на покупку компьютера. Это обусловлено тем, что основное назначение данного показателя заключается в определении эффективности эксплуатации новой техники, а не ее покупки.

3. Число разрядов в машинном слове определяет точность представления данных и результатов (измеряется в битах).

4. Скорость выполнения основных видов команд, т.е. преобразований типа «регистр-регистр», «регистр-память» и т.д. (измеряется в оп/с).

5. Емкость оперативной памяти (измеряется в мегабайтах).

6. Скорость обмена между ядром ЭВМ и периферийным оборудованием, где под ядром понимают схемы процессора и памяти и их линии связей.

2. ПРОЦЕССОРЫ

Процессор занимает центральное место в структуре компьютера. Он предназначен для реализации процессов, связанных с обработкой цифровой информации, а также для управления взаимодействием узлов и блоков, входящих в состав вычислительного устройства. Укрупненно структура процессора может быть представлена в виде композиции операционного и управляющего автоматов (рис. 3).

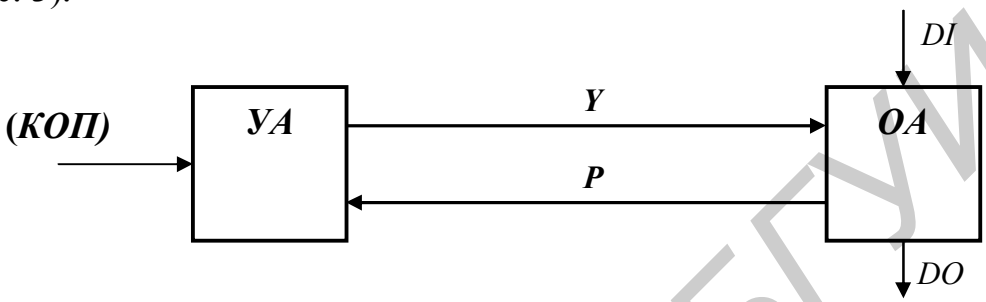

Рис. 3

При этом управляющий автомат может быть построен по схеме с жесткой логикой (автомат Мили или Мура) или спроектирован в виде микропрограммного УУ с хранимой логикой (схема Уилкса или ее модификации). Операционный автомат (ОА) используется для преобразования данных и включает в свой состав регистры, элементы управляющей логики, сумматоры, сдвигатели и другие функциональные узлы.

Вычислительный процесс инициируется кодом операции каждой выполняемой команды. Он преобразуется управляющим автоматом в последовательность сигналов, подаваемых на вход ОА, т.е. задает временную диаграмму пересылки и преобразования операндов.

По результатам выполненной команды ОА процессора формирует ряд признаков, используемых для ветвления исполняемого алгоритма.

2.1. Микропроцессорная секция К1804ВС1

Четырехразрядная микропроцессорная секция *К1804ВС1* предназначена для построения операционных блоков цифровых устройств с разрядностью,

кратной 4. На структурной схеме *МПС* условно выделяют четыре крупных блока (рис. 4): 1) *БВП* – блок внутренней памяти; 2) *АЛБ* – арифметико-логический блок; 3) блок регистра *Q*; 4) *БУ* – блок управления.

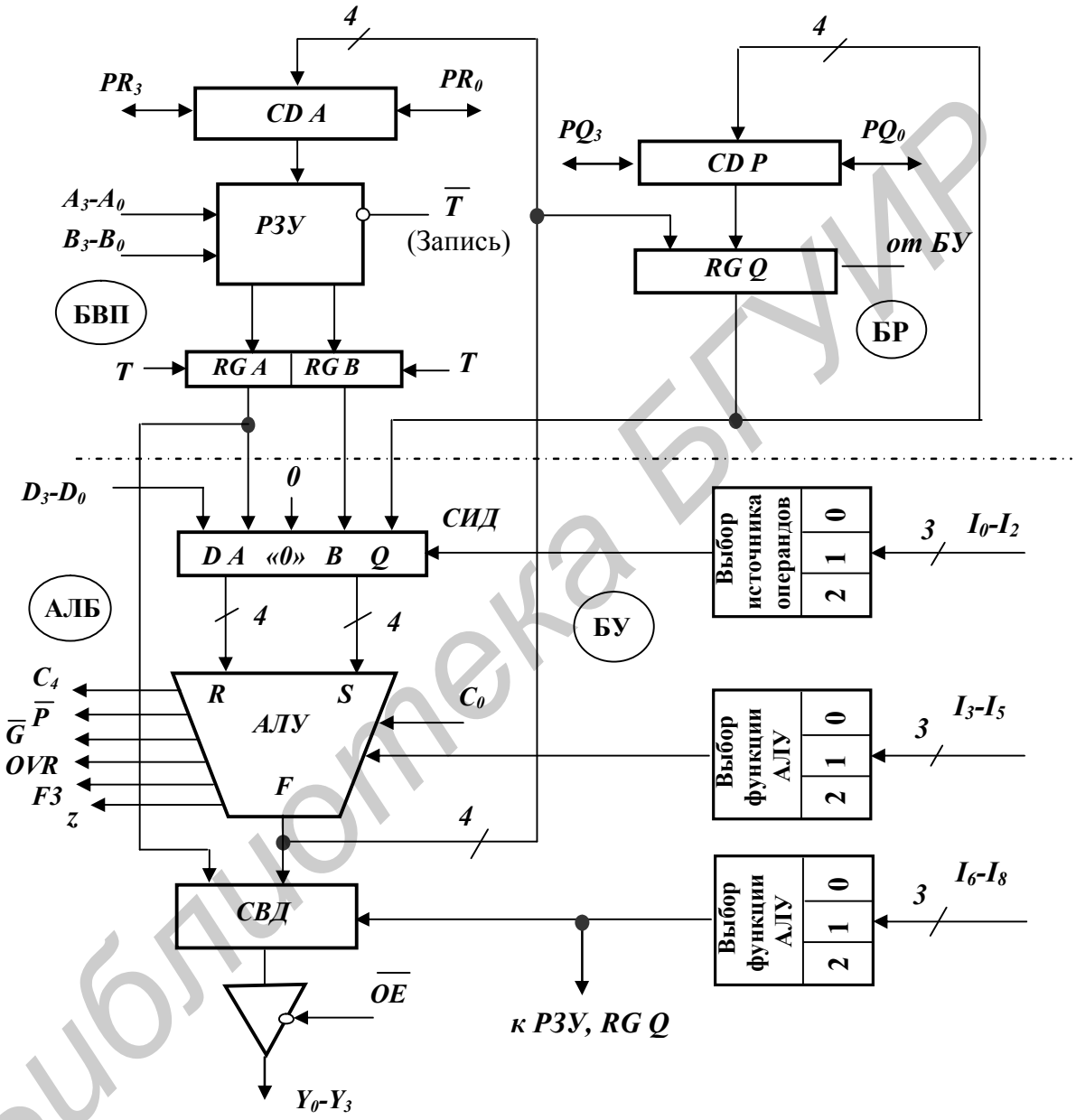

Рис. 4

Блок внутренней памяти содержит в своем составе регистровое *ЗУ* с двумя независимыми каналами выбора информации – канал адреса *A* и канал адреса *B*. На входе *РЗУ* включен сдвигатель *A (CD A)*, позволяющий записывать в *ЗУ*

информацию как без сдвига, так и со сдвигом вправо или влево на один разряд. Запись в *РЗУ* возможна только по адресу, указанному на линиях канала *B*.

Регистры *RG A* и *RG B,* установленные на выходах *РЗУ,* представляют собой 4-разрядные устройства с синхронной записью информации.

Выбор из *БВП* любого *РОН* в качестве источника информации осуществляется путем подачи на входы *A* и *B* адресной информации без программирования каких-либо других управляющих сигналов. Из *РЗУ* одновременно могут быть считаны два операнда. При этом, если на входах *A* и *B* установлены одинаковые адреса (выполняется обращение к одному и тому же *РОН*), то на обоих выходах *РЗУ* появляются идентичные данные.

Запись данных в *РЗУ* выполняется только по каналу *B,* при этом адрес по каналу *A* игнорируется. Моментом начала записи в *РЗУ* является момент перехода тактового импульса из состояния «1» в состояние «0».

Информация перед записью может быть сдвинута влево или вправо на один разряд. Эту операцию выполняет сдвигатель данных *CD A*, управляемый сигналами с дешифратора приемника результата. Схема *CD A* имеет следующую структуру:

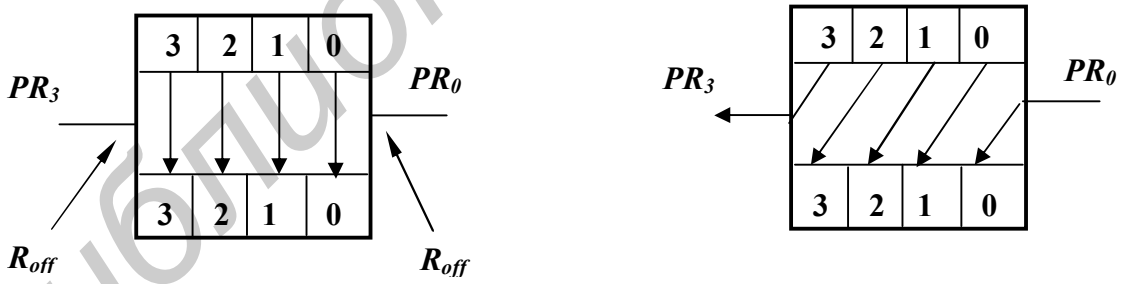

Рис. 5

Арифметико-логический блок включает в свой состав двухвходовое *АЛУ*, выполняющее 8 арифметических и логических операций и формирующее 4 признака результата:

C_4 – перенос из старшего разряда результата;

OVR – переполнение; *OVR* = 1, если $C_4 \oplus C_3 = 1$;

F_3 – знак числа или значения старшего разряда на выходе *АЛУ*;

z – признак нулевого результата.

С выхода *АЛУ* информация подается на первый вход селектора выходных данных (*СВД*). На второй вход селектора данные передаются прямо с выхода *RG A*, минуя *АЛУ*. С выхода *СВД* информация через управляемые усилители передается на выходную шину *МПС* – трехстабильную шину *Y*.

Управление работой *АЛБ* осуществляется с помощью табл. 1,2,3.

Таблица 1

Микрокод				Источник операндов *АЛУ*	
I_2	I_1	I_0	8-рич.	*R*	*S*
0	0	0	0	*A*	*Q*
0	0	1	1	*A*	*B*
0	1	0	2	0	*Q*
0	1	1	3	0	*B*
1	0	0	4	0	*A*
1	0	1	5	*D*	*A*
1	1	0	6	*D*	*Q*
1	1	1	7	*D*	0

Таблица 2

Микрокод				Операция
I_5	I_4	I_3	8-рич.	АЛУ
0	0	0	0	$R+S+C_0$
0	0	1	1	$S-R-1+C_0$
0	1	0	2	$R-S-1+C_0$
0	1	1	3	$R \vee S$
1	0	0	4	$R \wedge S$
1	0	1	5	$\overline{R} \wedge S$
1	1	0	6	$R \oplus S$
1	1	1	7	$\overline{R \oplus S}$

Селектор источников данных выбирает операнды для *R* и *S* входов *АЛУ*. Причем выбор источников операндов осуществляется сигналами микрокоманды $I_2 – I_0$ (см. табл. 1), приемника результата – сигналами $I_8 – I_6$ (см. табл. 3), а функции *АЛУ* – сигналами $I_5 – I_3$ (см. табл. 2).

Арифметические операции в *АЛУ* выполняются с учетом значения сигнала входного переноса C_0 и по правилам дополнительного кода при представлении отрицательных чисел. Это, в частности, означает, что если оба операнда нулевые и выполняется вычитание при $C_0 = 0$, на выходе устанавливается значение 1111, обозначающее в дополнительном коде число –1. При $C_0 = 1$ на выходе

Таблица 3

Микрокод				*РЗУ*		*RG Q*		Выход *Y*
I_8	I_7	I_6	*8-рич.*	Сдвиг	Загрузка	Сдвиг	Загрузка	
0	0	0	0	—	—	—	$F \to Q$	F
0	0	1	1	—	—	—	—	F
0	1	1	2	—	$F \to B$	—	—	A
0	1	1	3	—	$F \to B$	—	—	F
0	1	0	4	Вправо	$F/2 \to B$	Вправо	$Q/2 \to Q$	F
1	0	1	5	Вправо	$F/2 \to B$	—	—	F
0	0	1	6	Влево	$2F \to B$	Влево	$2Q \to Q$	F
1	1	1	7	Влево	$2F \to B$	—	—	F

АЛУ формируется значение 0000.

Выводы $\overline{P},\overline{G}$ *АЛУ* позволяют с помощью внешних схем организовать между секциями ускоренный перенос при комплексировании их в блок с разрядностью, кратной четырем.

Сигналы признаков, формируемые АЛУ, используются следующим образом.

Вывод F_3 – старший разряд *АЛУ* – может быть использован, например, для определения знака арифметической операции. При этом отсутствует необходимость отпирания трехстабильной выходной шины данных, что упрощает выполнение команд перехода в мультипроцессорных системах. При соединении нескольких МПС знаком является вывод F_3 старшей секции. Выводы F_3 остальных секций не используются.

Выход *z* выполнен по схеме с открытым коллектором, и при объединении нескольких МПС все выводы *z* соединяются в общей точке, подключаемой через резистор к источнику питания. Потенциал этой точки имеет высокий уровень, если все выходы *АЛУ* одновременно нулевые.

Блок регистра Q состоит из дополнительного регистра *RG Q* и сдвигателя *CD P*. Сдвигатель данных позволяет перезаписывать информацию в *RG Q* как

без сдвига, так и со сдвигом влево или вправо на один разряд. Запись данных в *RG Q* выполняется по положительному фронту сигнала синхроимпульса.

Блок управления формирует управляющие сигналы для остальных блоков *МПС*. Входами данной подсхемы является шина микрокоманды I_8–I_0, которую условно можно разделить на три части. Блок управления соответственно также может быть представлен в виде совокупности трех частей, причем первая – вырабатывает сигналы управления для *СИД*, вторая – используется для управления функцией *АЛУ*, а третья – для управления *СВД*, *RG Q, CD P, CD A*.

Инверсный сигнал $\overline{OE}$ предназначен для управления *Y*-выходами *МПС*. При $\overline{OE} = 0$ разрешается вывод информации через *Y*-выходы на *ШД*, если $\overline{OE} = 1$, выходная шина отключается (переводится в состояние R_{off}).

2.2. Микропроцессорная секция К1804ВС2

По отношению к предыдущей разработке процессорная секция *К1804ВС2* имеет ряд усовершенствований (рис. 6):

во-первых, *АЛУ* микропроцессора выполняет арифметические, логические и специальные функции;

во-вторых, сдвигатель данных *АЛУ CDA* выполняет как логические, так и арифметические сдвиги;

в-третьих, в микропроцессоре заложена возможность внешнего расширения *РЗУ* путем подсоединения любого числа дополнительных рабочих регистров, обращение к которым возможно в различных режимах адресации.

В целом микропроцессорная секция состоит из 4 основных блоков: блока внутренней памяти (*БВП*), арифметико-логического блока (*АЛБ*), блока рабочего регистра (*БР*) и блока управления (*БУ*).

Блок внутренней памяти включает в свой состав шестнадцать 4-разрядных *РОН*, объединенных в *РЗУ*, а также регистр *A* и регистр *B* с трехстабильным выходом. Информация, размещаемая в регистрах, адресуется по каналам адреса $A_3 – A_0$ и $B_3 – B_0$ *РЗУ* соответственно.

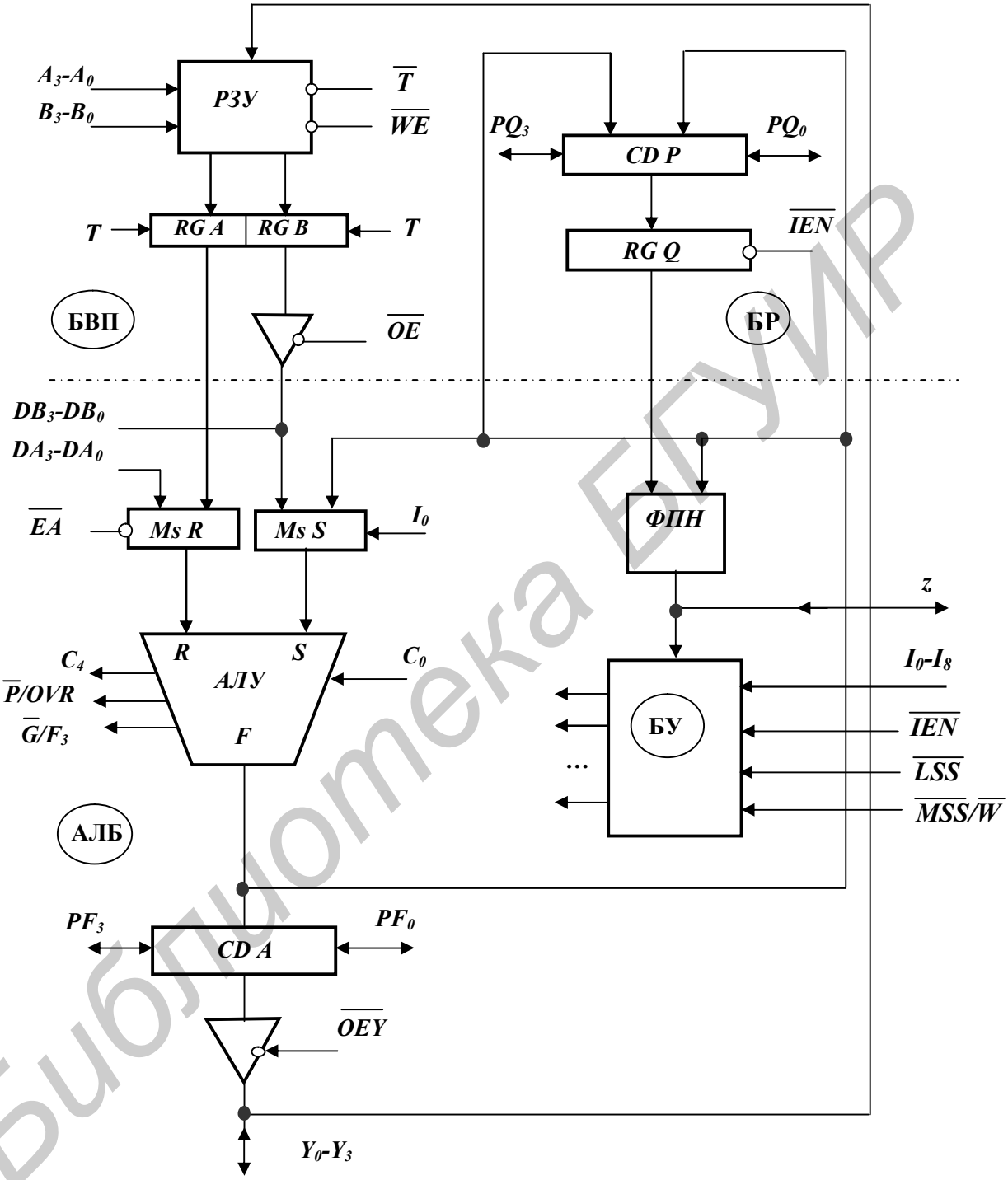

Рис. 6

Управление выходом регистра *B* осуществляется сигналом $\overline{OE}$, что позволяет передавать данные с выхода *БВП* на вход мультиплексора *MsS АЛУ* и на входы DB_3-DB_0 *МПС* при $\overline{OE}=0$, кроме того, при $\overline{OE}=1$ имеется возможность вводить информацию со входов DB_3-DB_0.

Каждый из *РОН*ов может быть выбран в качестве источника или приемника полученного результата. При этом информация с выходов *РЗУ* записывается в *RG A* или *RG B* при наличии уровня логической единицы на входе *T*, в частности, если *T=0*, то регистры находятся в режиме хранения.

Запись информации в *РЗУ* может производиться только по адресу *B*. При этом необходимо, чтобы на входах $\overline{WE}$ и $\overline{T}$ был установлен логический «*0*». Подача на входы $\overline{WE}$ или $\overline{T}$ единичного потенциала запрещает режим записи. Считывание информации из *РЗУ* может производиться одновременно по адресам *A* и *B*. Если на входах *A* и *B* установлены одинаковые адреса, то на выходы *РЗУ* считывается одинаковая информация.

МПС может функционировать в режиме двухадресной ($B=A+B$) и трехадресной ($Q=A+B$) обработки данных. В двухадресном режиме на входы A_3–A_0 подается адрес операнда *R*, а на входы B_3–B_0 – адрес операнда *S* и результата. В трехадресном режиме приемником результата служит дополнительный регистр *Q*, адресуемый в микрокоманде неявно.

Арифметико-логический блок состоит из двух мультиплексоров *MsS* и *MsR,* арифметико-логического устройства, сдвигателя данных *CDA* с трехстабильным выходом (рис. 7), формирователя признака нуля *ФПН*, используемого при выполнении арифметических, логических и специальных функций, а также при формировании сигналов состояния *МПС*.

Входные мультиплексоры осуществляют выбор источников операндов *R* и *S* под действием управляющих сигналов $\overline{EA}$, $\overline{OE}$, I_0 в соответствии с табл. 4.

Из таблицы следует, что, если I_0=1, то параллельно с выполнением операции в *АЛУ* возможен вывод данных из *РЗУ* по шине DB_3–DB_0 во внешнюю память.

Таблица 4

$\overline{EA}$	I_0	$\overline{OE_1}$	R	S
0	0	0	A	B
0	0	1	A	DB
0	1	0	A	Q
0	1	1	A	Q
1	0	0	DA	B
1	0	1	DA	DB
1	1	0	DA	Q
1	1	1	DA	Q

Арифметико-логическое устройство обеспечивает выполнение семи арифметических, девяти логических операций, а также девяти специальных функций. Выбор операции *АЛУ* осуществляется под действием поля регистра микрокоманды $I_8 - I_0$, причем если разряды $I_4 - I_0$ имеют значение 00000, то АЛУ выполняет специальные функции.

Выводы $\overline{P},\overline{G}$ *МПС* используются для организации ускоренного переноса в многоразрядных процессорах. Причем выходы $\overline{P},\overline{G}$ *Ст.МПС* не используются, а выходы F_3, OVR – наоборот, используются только у *Ст.МПС*. Этот факт позволяет объединить выводы $\overline{G}$ и F_3 , а также $\overline{P}$ и OVR и соответственно уменьшить общее число контактов интегральной микросхемы.

Данные с выхода *АЛУ* могут быть переданы на входы сдвигателя *CD P* регистра *Q* и на входы сдвигателя *CD A*, соединенного с выходами *F АЛУ*. Таким образом, выходная информация может записываться в *РЗУ* и выводиться на шину Y_3 - Y_0 как без сдвига, так и со сдвигом влево или вправо на один разряд.

Отличительной особенностью *CDA АЛУ* является процесс выполнения арифметических сдвигов. Схема сдвигателя для старшей *МПС* процессора при сдвиге влево или вправо на 1 разряд в этом случае имеет вид, показанный на рис. 7.

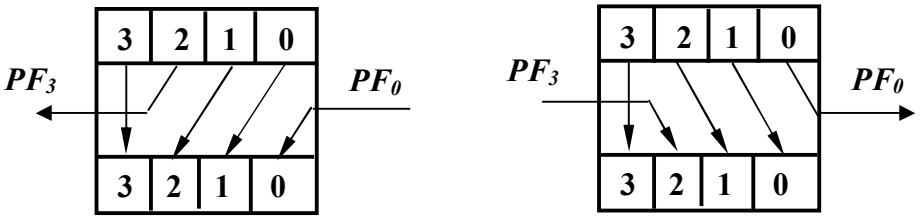

Рис. 7

Выполнение сдвиговых операций в процессоре *К1804ВС2* осуществляется под управлением сигналов микрокоманды $I_8 - I_5$, при этом сигнал $\overline{IEN} = 0$, сигналы микрокоманды $I_0 \vee I_1 \vee I_2 \vee I_3 \vee I_4 = 1$ (табл. 5).

Таблица 5

Код микрокоманды	Операция *АЛУ*	Выходные сигналы					
		z	C_4	$\overline{P}/OVR$		$\overline{G}/F3$	
$I_4I_3I_2I_1I_0$				*Ст. МПС*	*Мл. и Ср. МПС*	*Ст. МПС*	*Мл. и Ср. МПС*
00000	*Специальные функции*						
00001	$F_{АЛУ}$:=«1111»	*0*	*0*	*0*	*0*	F_3	$\overline{G}$
0001x *0010x* *0011x* *0100x* *0101x* *0110x* *0111x*	$S-R-1+C_0$ $R-S-1+C_0$ $R+S+C_0$ $S+C_0$ $\overline{S}+C_0$ $R+C_0$ $\overline{R}+C_0$	*z*	C_4	*OVR*	$\overline{P}$		
1000x	$F_{АЛУ}$:=«0000»	1	*0*	*0*	*0*		
1001x *1010x* *1011x* *1100x* *1101x* *1110x* *1111x*	$\overline{R} \wedge S$ $\overline{R} \oplus S$ $R \oplus S$ $\overline{R \wedge S}$ $\overline{R \vee S}$ $R \wedge S$ $R \vee S$	*z*	*0*	*0*	*0*		

Функционирование *МПС* при выполнении некоторых операций зависит от ее положения в системе, поэтому при соединении секций необходимо выполнить их настройку на заданное положение: *Ст.МПС*, *Ср.МПС, Мл.МПС*. Такая настройка выполняется при помощи выводов $\overline{LSS}$ и $\overline{MSS}/\overline{W}$. В младшей *МПС* на входе $\overline{LSS}$ устанавливается «0», при этом линия $\overline{MSS}/\overline{W}$ становится выходом $\overline{W}$, причем необходимо, чтобы $\overline{W}$=0 для каждого такта записи в РЗУ, в противном случае – $\overline{W}$=1. Поэтому, как правило, входы $\overline{WE}$ всех *МПС* соединяют с выходом $\overline{W}$ *Мл.МПС*, как показано на рис. 8.

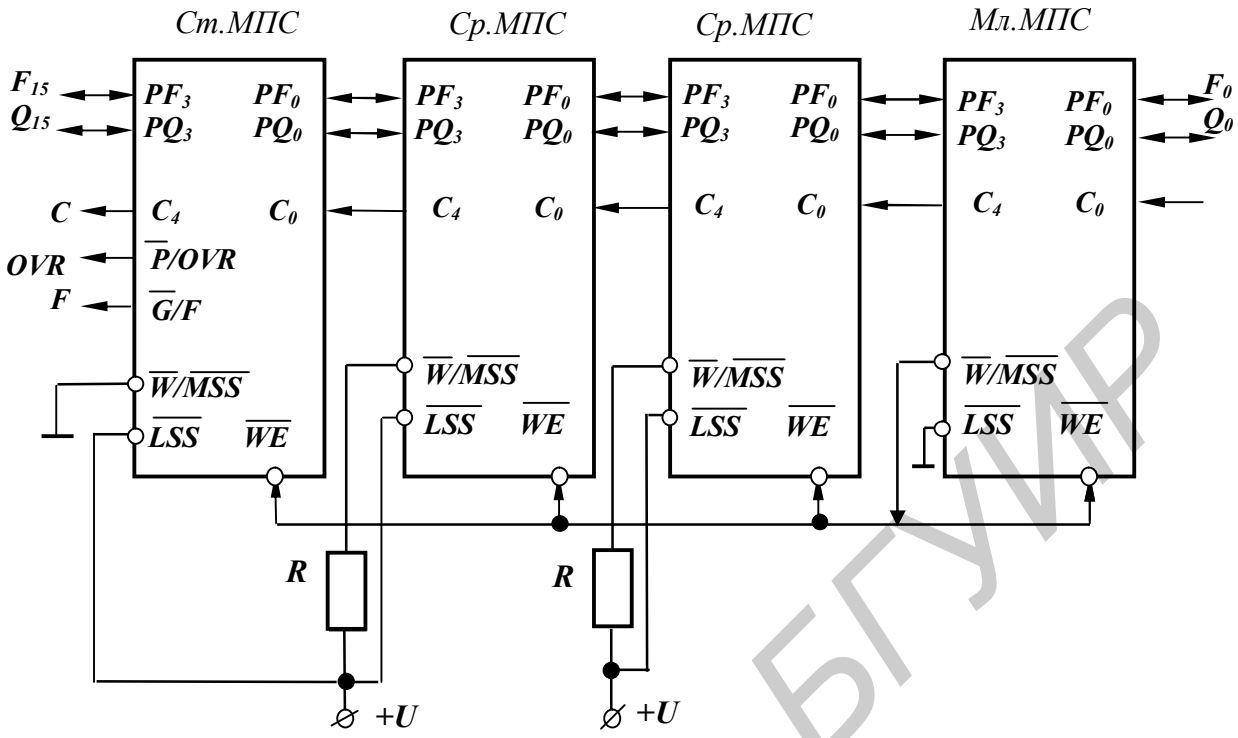

Рис. 8

В средней и старшей *МПС* на входе $\overline{LSS}$ устанавливается логическая «1». При этом линия $\overline{MSS}/\overline{W}$ становится входом секции $\overline{MSS}$. В старшей *МПС* на этот вход подается уровень нуля, а в средних – уровень логической «1». Это позволяет вывести в старшей *МПС* на выходы $\overline{P}/OVR$ и $\overline{G}/F_3$ сигналы *OVR* и F_3, а на выводы средних *МПС* – выходы ускоренного переноса $\overline{P}$, $\overline{G}$.

Расширение знака при вычислениях может быть выполнено на несколько разрядов за 1 такт, для чего используется предпоследняя микрокоманда (код $I_8 - I_5 = E$, табл. 6). Микрокоманда *E* передает знак с вывода PF_0 *МПС* на выводы PF_3, $Y_3 - Y_0$. Принцип расширения знака используется не для модификации результата, а для формирования данных. Например, если 16-разрядный процессор должен интерпретировать 8-разрядные данные как двоичные числа со знаком, то знаковый разряд расширяется на один байт. Однако данный подход требует реорганизации межсекционных цепей сдвига процессора.

Таблица 6

I_8–I_5	Функц. СДА	Функц. СДР	Y_3		Y_2		Y_1	Y_0	PF_3		PF_0	$\overline{W}$	RG Q	
			Ст. МПС	Ср.,Мл. МПС	Ст. МПС	Ср.,Мл МПС			Ст.МПС	Ср., Мл. МПС		LSS=0	PQ_3	PQ_0
0000	$F/2 \to Y$ Ар.С	$Q \to Q$	F_3	PF_3	PF_3	F_3	F_2	F_1	Вх.	Вх.	F_0	0	R_{off}	
0001	$F/2 \to Y$ Лог.С	$Q \to Q$	PF_3	PF_3	F_3	F_3	F_2	F_1	Вх.	Вх.	F_0	0	R_{off}	
0010	$F/2 \to Y$ Ар.С	$Q/2 \to Q$ Лог.С	F_3	PF_3	PF_3	F_3	F_2	F_1	Вх.	Вх.	F_0	0	Вх.	Q_0
0011	$F/2 \to Y$ Лог.С	$Q/2 \to Q$ Лог.С	PF_3	PF_3	F_3	F_3	F_2	F_1	Вх.	Вх.	F_0	0	Вх.	Q_0
0100	$F \to Y$	$Q \to Q$	F_3	F_3	F_2	F_2	F_1	F_0	Вх.	Вх.	P	0	R_{off}	
0101	$F \to Y$	$Q/2 \to Q$ Лог.С	F_3	F_3	F_2	F_2	F_1	F_0	Вх.	Вх.	P	1	Вх.	Q_0
0110	$F \to Y$	$F \to Q$	F_3	F_3	F_2	F_2	F_1	F_0	Вх.	Вх.	P	1	R_{off}	
0111	$F \to Y$	$F \to Q$	F_3	F_3	F_2	F_2	F_1	F_0	Вх.	Вх.	P	1	R_{off}	
1000	$2F \to Y$ Ар.С	$Q \to Q$	F_3	F_2	F_1	F_1	F_0	PF_0	F_2	F_3	Вх.	0	R_{off}	
1001	$2F \to Y$ Лог.С	$Q \to Q$	F_2	F_2	F_1	F_1	F_0	PF_0	F_3	F_3	Вх.	0	R_{off}	
1010	$2F \to Y$ Ар.С	$2Q \to Q$ Лог.С	F_3	F_2	F_1	F_1	F_0	PF_0	F_2	F_3	Вх.	0	Q_3	Вх.
1011	$2F \to Y$ Лог.С	$2Q \to Q$ Лог.С	F_2	F_2	F_1	F_1	F_0	PF_0	F_3	F_3	Вх.	0	Q_3	Вх.
1100	$F \to Y$	$Q \to Q$	F_3	F_3	F_2	F_2	F_1	F_0	F_3	F_3	R_{off}	1	R_{off}	
1101	$F \to Y$	$2Q \to Q$ Лог.С	F_3	F_3	F_2	F_2	F_1	F_0	F_3	F_3	R_{off}	1	Q_3	Вх.
1110	$PF_0 \to$ $Y_0\ Y_1 Y_2 Y_3$	$Q \to Q$	PF_0	PF_0	PF_0	PF_0	PF_0	PF_0	PF_0	PF_0	Вх.	0	R_{off}	
1111	$F \to Y$	$Q \to Q$	F_3	F_3	F_2	F_2	F_1	F_0	F_3	F_3	R_{off}	0	R_{off}	

Четыре микрокоманды (коды $I_8 - I_5 = 4, 5, 6, 7$) используются для формирования паритета на шине PF_0. Паритет представляет собой результат операции *M2* всех выходов *АЛУ* и сигнала, подаваемого на вход PF_3. Паритетная логика обладает способностью наращивания путем соединения вывода PF_3 предыдущей *МПС* с выводом PF_0 последующей. При этом уравнение паритета будет иметь следующий вид:

$$PF_{0\ Мл.МПС} = F_{15} \oplus F_{14} \oplus \ldots \oplus F_1 \oplus F_0 \oplus PF_{3\ Ст.МПС}.$$

Специальные функции, выполняемые *МПС*, определяются сигналами $I_8 - I_5$ при наличии нулевой комбинации на входах $I_4 - I_0$. Схема может выполнять девять специальных функций, которые перечислены в табл. 10. Семь кодовых комбинаций не используются и являются запрещенными при функционировании *МПС*.

1. Умножение без знака. Реализация данной функции предполагает выполнение базовых операций сдвига и сложения. Для умножения чисел с разрядностью, кратной четырем ($4n \times 4n$), требуется $4n$ тактов работы *МПС*. При выполнении умножения предполагается, что регистр R_0 *ОЗУ* предварительно был очищен и далее будет использоваться для размещения старших бит частичных произведений результата. Множимое записывается в регистр R_1, а множитель – в регистр R_2. Регистры R_0 и R_1 могут быть расположены в *РЗУ МПС* и адресованы по входам *A* и *B* соответственно, а также располагаться во внешней памяти процессорной секции. Во втором случае регистр R_1 используется как источник операнда *R* и соединяется с шиной $DA_0 - DA_3$, а регистр R_0 – как источник операнда *S* и как приемник результата и соединяется с шинами $DB_0 - DB_3$ и $Y_3 - Y_0$. После начальной установки множитель из R_2 пересылается в *RG Q*. После этого микрокоманда умножения без знака выполняется $4n$ раз.

Соединение выводов МПС при выполнении данной специальной функции показано на рис. 9.

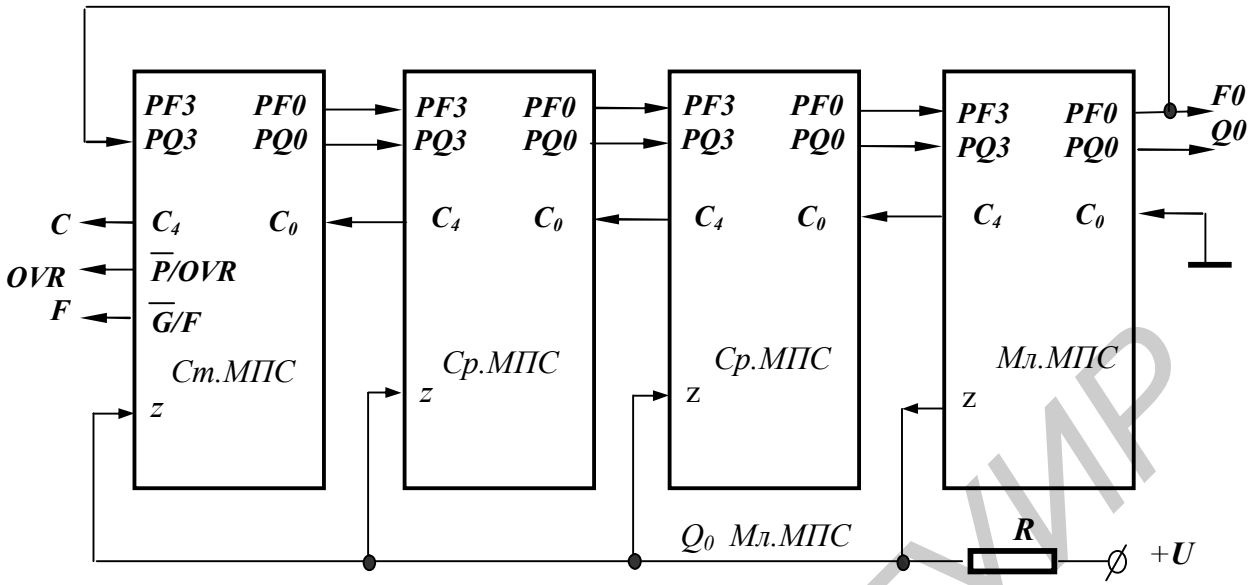

Рис. 9

Шина *z Мл.МПС* включена в режим выдачи информации. Сюда выдается младший разряд множителя из *RG Q (*Q_0 *Мл.МПС)*. Линии *z* остальных *МПС* являются входами. *АЛУ* реализует функцию $F=S+C_0$ при $z=0$ или $F=R+S+C_0$ при z=1. Таким образом, при $z= Q_0 = 1$ выполняется сложение множимого из *R1* с частичным произведением. При $z= Q_0 = 0$ сложение не выполняется ($C_0=0$).

По каждому положительному фронту синхросигнала содержимое выходов АЛУ сдвигается вправо, и полученное частичное произведение записывается в R_0 (рис. 10, 11). При этом сигнал C_4 *Ст.МПС* передается в старший разряд R_0, а

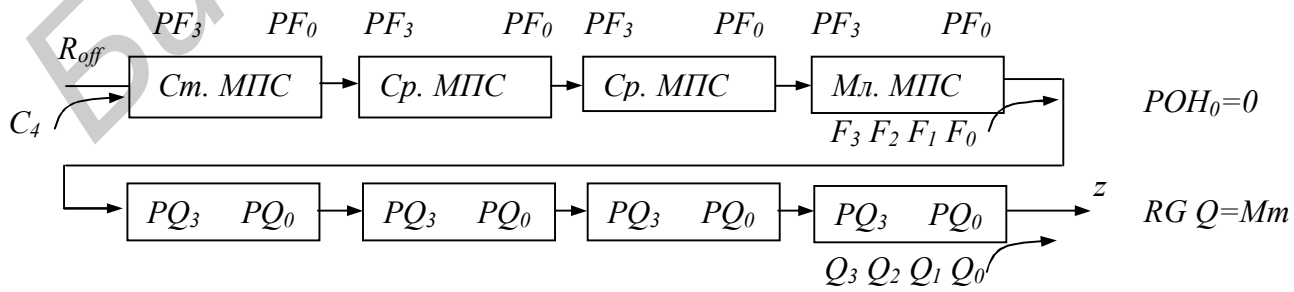

Рис. 10

младший разряд R_0 через выводы PF_0 *Мл.МПС* и PQ_3 *Ст.МПС* передается в регистр *Q,* одновременно выполняется сдвиг *RG Q* вправо.

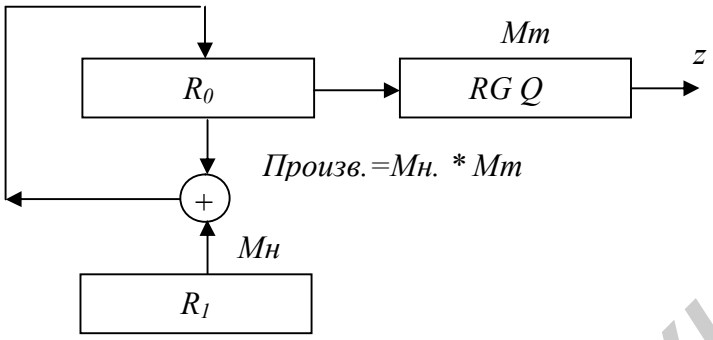

Рис. 11

При использовании типичного устройства управления для выполнения умножения без знака в *МПП* требуется хранить только две микрокоманды (табл.7), а общее время умножения составит $4n+1$ тактов.

Таблица 7

Адрес	Микрокоманда								Комментарий
	I_0	$I_4I_3I_2I_1$	$I_8I_7I_6I_5$	$\overline{OE_1}$	$\overline{OEY}$	A_3-A_0	B_3-B_0	C_0	
A_1	*X*	*0110* $R+S+C_0$	*0110* $F\rightarrow Y$ $F\rightarrow Q$	*X*	*X*	*0010* R_2	*XXXX*	*0*	$RG_2\rightarrow RG\ Q$ $(Mm\rightarrow RG\ Q)$
A_2	*0* *S=B*	*0000*	*0000* $R+S+C_0$	*0* *S=B*	*0* ½ *F* $\rightarrow Y$	*0001* R_1	*0000* *RG 0*	*0*	Умножение $R_0:=R_0+R_1$

2. Умножение в дополнительном коде. Данный алгоритм выполняется аналогично предыдущему, за исключением последнего такта, на котором производится коррекция результата. Начальная установка и первые $4n-1$ тактов умножения выполняются точно так же, как и при умножении без знака. Однако в данном случае в старший разряд сдвинутого частичного произведения в каждом такте записывается сумма $F_3 \oplus OVR$, а не C_4. Это обеспечивает передачу

требуемого бита в старший разряд частичного произведения при возникновении переноса.

На *4n*-м такте знаковый разряд множителя находится на шине *z Мл.МПС*. В это время необходимо подать микрокоманду последнего такта умножения в дополнительном коде или выполнить коррекцию результата.

Объединение МПС при выполнении умножения в дополнительном коде представлено на рис. 12.

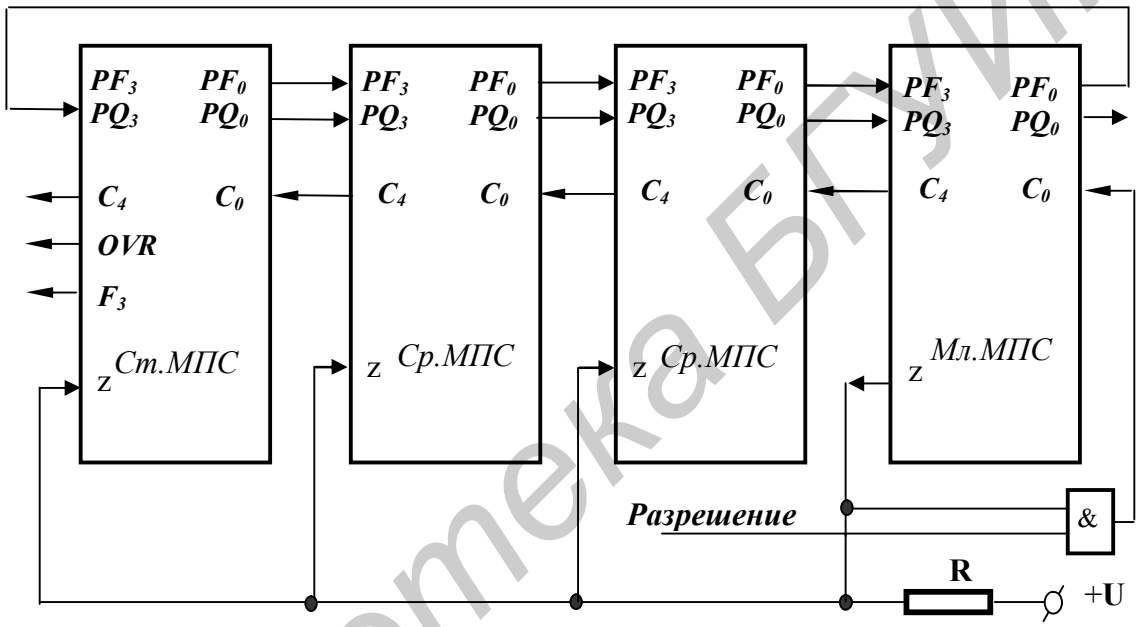

Рис. 12

При умножении на входе C_0 процессора должен присутствовать уровень нуля до последнего такта. На последнем такте на данном входе должен присутствовать сигнал, соответствующий логическому уровню линии *z*. В связи с этим на входе C_0 устанавливают элемент 2И с входными сигналами *z* и разрешение. Сигнал «Разрешение» устанавливается в единичное состояние только на последнем такте умножения. В остальных тактах линия *z* используется точно так же, как и в предыдущей команде.

На последнем такте умножения в дополнительном коде АЛУ реализует функцию $F=S+C_0$ при $z=0$ и или $F=S–R–1+C_0$ при z=1. Таким образом, если множитель положительный, то процесс умножения на этом такте заканчивает-

ся, а если множитель отрицательный, то множимое вычитается из полученного частичного произведения, после чего умножение заканчивается.

Пример. Пусть *Мн=7*, *Мт=-5*. Тогда в результате умножения получим произведение $[-Пр]_д$=1.1011101. В соответствии с рис. 11 процесс формирования дополнительного кода результата будет иметь вид:

$$
\begin{array}{lll}
 & 00000000 & RG\,Sm \\
1\ шаг: & 01110000 & +Мн \\
\hline
 & 00111000 & RG\,Sm + Мн,\ RG\,Sm \cdot 2; \\
2\ шаг: & 01110000 & +Мн \\
\hline
 & 01010100 & RG\,Sm + Мн,\ RG\,Sm \cdot 2; \\
3\ шаг: & 00000000 & +0 \\
\hline
 & 00101010 & RG\ \ Sm\ \ +Мн,\ \ RG\ \ Sm \cdot 2; \\
4\ шаг: & 01110000 & +Мн \\
\hline
 & 01001101 & RG\,Sm + Мн,\ RG\,Sm \cdot 2; \\
5\ шаг: & 10010000 & [-Мн]_д - коррекция \\
\hline
 & 11011101 & RG\,Sm - Мн = [-Пр]_д .
\end{array}
$$

3. Нормализация чисел обычной и двойной длины. Операция нормализации чисел в дополнительном коде выполняется путем сдвига числа в сторону старших разрядов до тех пор, пока два старших бита не будут иметь различные значения. При этом знак нормализуемого операнда фиксируется в старшем разряде.

Пример

0.000001011000111	–	0.101100011100000
1.110101101111101	–	1.010110111110100
0.000000000000000	–	не нормализуется
1.111111111111111	–	1.000000000000000

Нормализация выполняется для чисел как обычной, так и двойной длины. При нормализации чисел обычной длины используется только регистр *Q*, а при нормализации чисел двойной длины необходимо использовать еще

АЛУ и блок внутренней памяти. В первом случае операнд помещается в регистр Q и по положительному фронту тактового сигнала выполняется его сдвиг в сторону старших разрядов. Режим сдвига сохраняется до получения в двух старших битах регистра Q различных значений (0.1… или 1.0…), при этом через вывод PQ_0 *МПС* в младшие разряды регистра Q вдвигаются нулевые значения. Появление «1» на выходе C_4 *Ст.МПС* означает окончание операции нормализации; это обусловлено тем, что в *Ст.МПС* значение переноса формируется по закону $C_4 = Q_3 \oplus Q_2$. Кроме того, по признаку *OVR* можно определить предпоследний такт нормализации, так как $OVR = Q_2 \oplus Q_1$, т.е. опережает C_4 на один такт.

Чтобы не выполнять нормализацию нулевого операнда, необходимо идентифицировать соответствующую ситуацию установкой какого-либо признака. С этой целью используется схема ФПН, на выходе *z* которой устанавливается «1», если все выходы регистра Q – нулевые. При появлении единицы на выходе C_4 операция нормализации прекращается, а на выходе F_3 *Ст.МПС* устанавливается знак числа, хранящегося в *RG Q* ($Q_3 \rightarrow F_3$).

При выполнении операции нормализации обычной длины можно подсчитать количество тактов, необходимое для выполнения этой операции. С этой целью на выходе C_0 *МПС* устанавливается логическая единица, при этом регистр *РЗУ*, адресованный по каналу *B*, становится счетчиком тактов. В данном случае *АЛУ* выполняет функцию $F=S+C_0$.

Соединение выводов МПС при выполнении операции нормализации чисел обычной длины имеет вид, показанный на рис. 13.

При выполнении сдвигов счетчик тактов может быть организован во внешней памяти. В данном случае содержимое счетчика передается на *S* вход *АЛУ* через внешнюю шину $DB_3 – DB_0$, а результат выводится на шину $Y_3 – Y_0$. Если выполняется нормализация чисел двойной длины, то старшие разряды числа находятся в ячейке *РЗУ*, адресуемой по *B*, а младшие – в *RG Q* . Вывод PQ_3 старшей *МПС* должен быть соединен с выводом PF_0 младшей *МПС*. На выводе

PQ_0 младшей МПС (так же, как и при выполнении нормализации чисел обычной длины) устанавливается ноль.

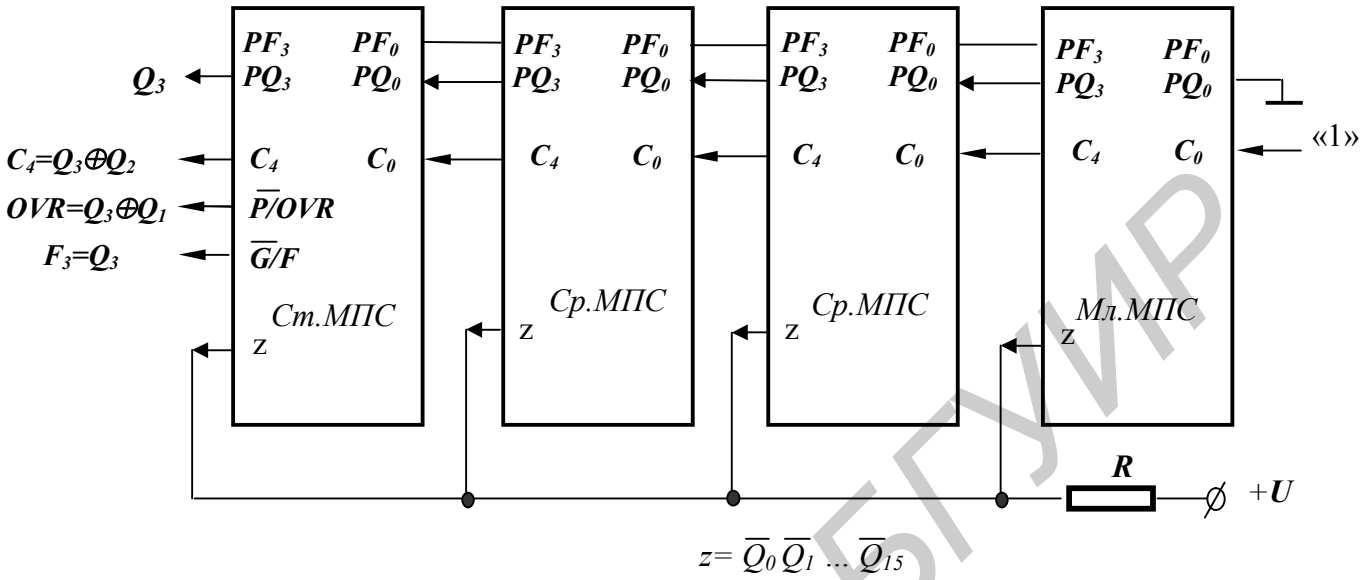

Рис. 13

Для хранения старших разрядов нормализуемого числа можно также использовать и внешнюю память, при этом источником операнда служит шина DB_3–DB_0, а сдвинутое значение передается на шину $Y_3 – Y_0$. Так как при нормализации чисел двойной длины *АЛУ* участвует в выполнении операции и реализует функцию $F=S+C_0$, то на выходе C_0 необходимо установить уровень нуля. Для подсчета числа сдвигов в этом случае можно применять внешнюю логику, а сигналы C_4, OVR и z используются так же, как и при нормализации чисел обычной длины, причем $C_{4Ст.МПС} = F_3 \oplus F_2$ этой же секции, $OVR_{Ст.МПС} = F_3 \oplus F_1$ тоже старшей *МПС,* значение

$$z = \overline{F}_{n-1}\overline{F}_{n-2}\ldots\overline{F}_1\overline{F}_0\overline{Q}_{n-1}\overline{Q}_{n-2}\ldots\overline{Q}_1\overline{Q}_0,$$

где n – разрядность нормализуемого числа.

Соединение выводов *МПС* при нормализации чисел двойной длины показано на рис. 14.

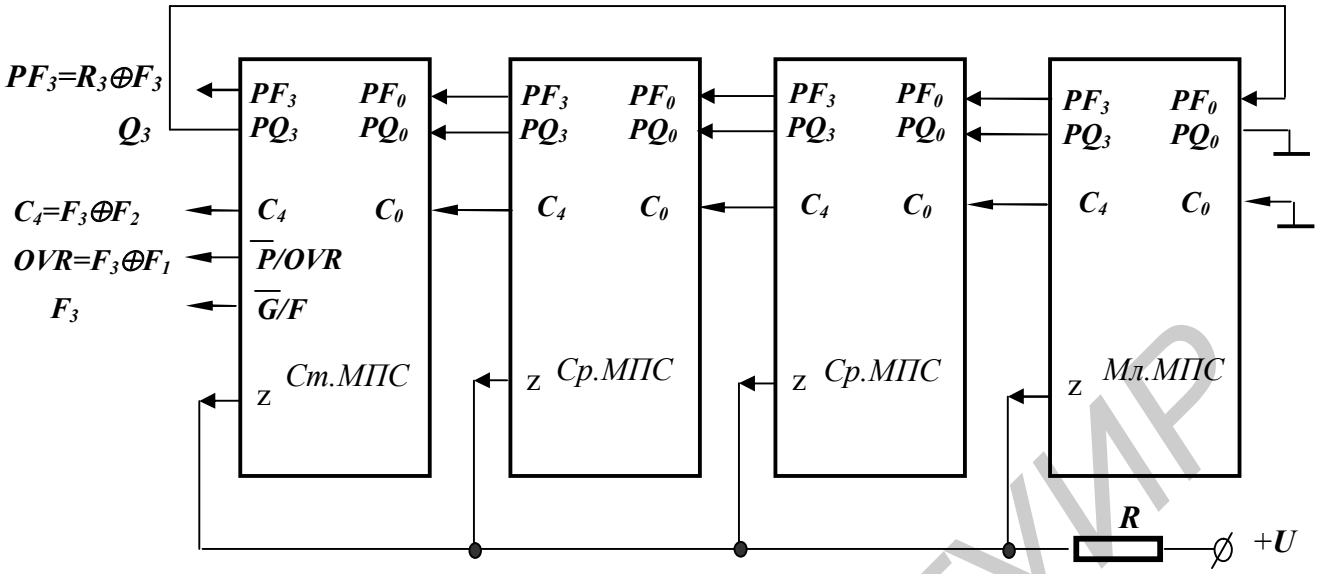

Рис. 14

Приведенное на рис. 14 обозначение R_3 указывает на знаковый разряд операнда, подаваемого на *R-вход* АЛУ.

Код микрокоманды для данной специальной функции $I_8 - I_5$ =1010 (для операнда двойной длины) и $I_8 - I_5$ =0111 (для операнда одинарной длины).

4. Преобразование «число со знаком – дополнительный код» (I_8–I_5 = 0101). В процессе отработки данной специальной функции положительные числа не модифицируются, а отрицательные – преобразуются в дополнительный код от текущего кода операнда.

В ходе преобразования исходное число подается на вход *S АЛУ* (например, из регистра *RGB БВП,* из *RGQ* или с внешней шины $DB_3 - DB_0$), а его знак – разряд S_3 *Ст.МПС* – на шину *z* и определяет, над каким числом (отрицательным или положительным) выполняются действия. Вход C_0 *Мл.МПС* соединяется с шиной *z*. АЛУ реализует следующий алгоритм вычислений:

$F=S+C_0$ при $z=0$,

$F = \overline{S} + C_0$ при z=1 (образование дополнительного кода).

При преобразовании отрицательного числа на шине *z Ст.МПС* появляется потенциал логической единицы, что позволяет в процессе преобразования к

обратному коду прибавить единицу, так как $z=C_0$.

Соединение выводов МПС для выполнения функции кодопреобразования имеет следующий вид:

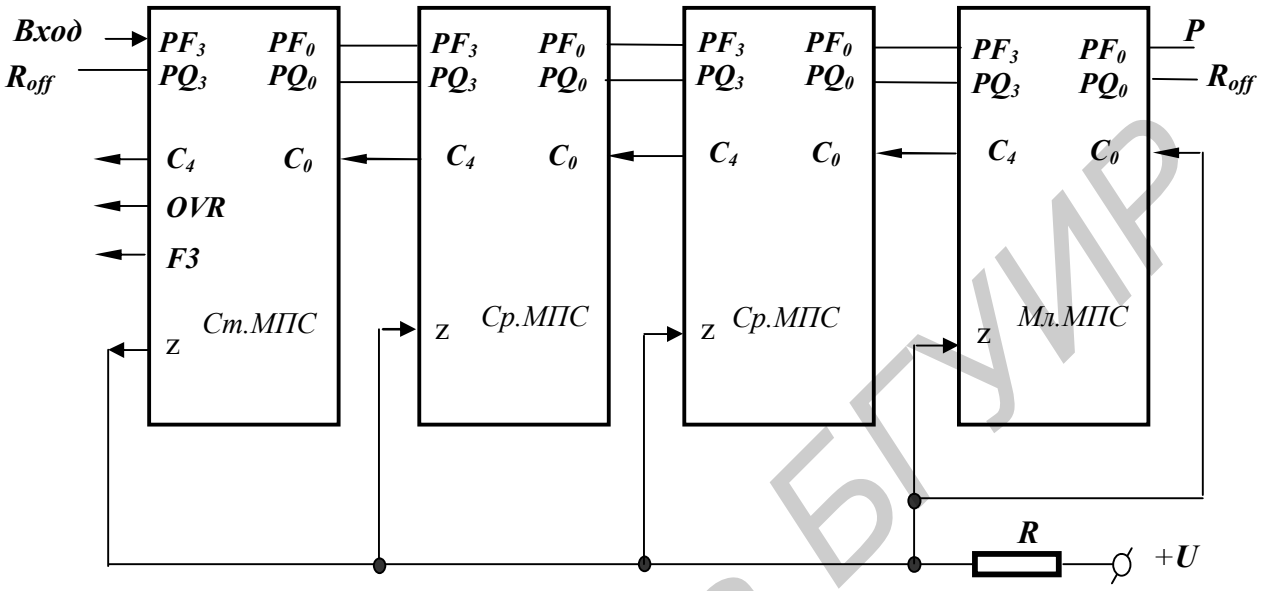

Рис. 15

5. Инкрементирование операнда на 1 или 2. Увеличение числа на единицу или двойку выполняется за один такт. Модифицируемый операнд подается на вход *S АЛУ* (например, из регистра *RGB БВП,* из *RGQ* или с внешней шины $DB_3 – DB_0$), после чего выполняется функция $F=S+C_0+1$, что при $C_0=1$ предполагает увеличение операнда на 2, а при $C_0=0$ – на единицу. При выполнении данного преобразования используется стандартное включение *МПС*.

*6. Деление в дополнительном коде***.** Данная специальная функция выполняется в соответствии с микрокомандой, представленной в табл. 8.

Таблица 8

Адрес	Микрокоманда								Комментарий
	I_0	$I_4I_3I_2I_1$	$I_8I_7I_6I_5$	$\overline{EA}$	$\overline{OEY}$	$A_3 – A_0$	$B_3 – B_0$	C_0	
A_1	0 S=B	0000	1100 S+R+C_0 z=0 S-R-1+C_0 z=1	0 R=A	X	0000 R_0 (Дт)	0001 R_1 (Дм.Ст. часть)	z	Деление в доп. коде

Соединение выводов МПС при выполнении операции деления в дополнительном коде показано на рис. 16; рис. 17 поясняет расположение операндов в регистрах и размещение частного.

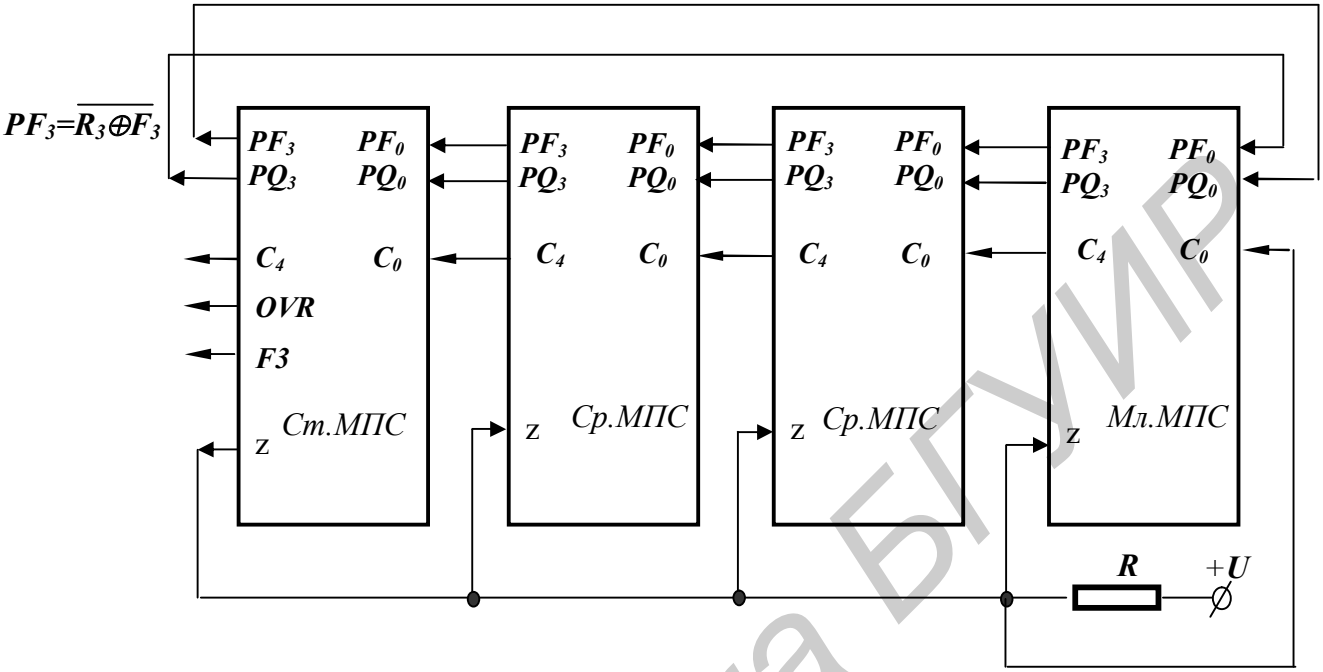

Рис. 16

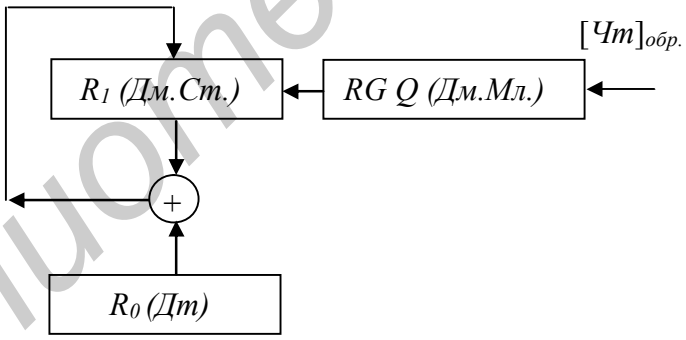

Рис. 17

При выполнении данной специальной функции следует помнить, что используемый алгоритм деления формирует частное в обратном коде. Таким образом, при отрицательном частном необходимо выполнить коррекцию результата с помощью специальной микрокоманды (табл. 9). Данная микрокоманда фактически прибавляет единицу в младший разряд частного, если результат деления отрицательный. При положительном частном коррекция не выполняется.

Таблица 9

Адрес	Микрокоманда								Коммента-рий
	I_0	$I_4I_3I_2I_1$	$I_8I_7I_6I_5$	$\overline{EA}$	$\overline{OEY}$	A_3-A_0	B_3-B_0	C_0	
A2	*1* $S=Q$	*0100* $S+C_0$	*0110* $F\rightarrow Q$	*0* $R=A$	*X*	*0000*	*0001*	*1*	*Деление в доп. коде (коррекция)*

Сводная таблица специальных функций может быть представлена в следующем виде:

Таблица 10

Мк. код.	Функция АЛУ	PF_3		PF_0	PQ_3	PQ_0	$\overline{W}$	C_4	$\overline{P}$ / OVR		$\overline{G}$ / F_3		z		
Bx I_8-I_5		*Ст.МПС*	*Мл.Ср.МПС*						*Ст.МПС*	*Ст.МПС*	*Ст.МПС*	*Мл.Ср.МПС*	*Ст.МПС*	*Ср.МПС*	*Мл.МПС*
1. Умножение без знака															
0000	$F=S+C_0$ *при* $z=0$, $F=S+R+C_0$ *при* $z=1$	R_{off}	*Bx*	F_0	*Bx*	Q_0	*0*	C_4	*OVR*	$\overline{P}$	F_3	$\overline{G}$	*Bx*	*Bx*	Q_0
2. Умножение в дополнительном коде															
0010	$F=S+C_0$ *при* $z=0$, $F=S+R+C_0$ *при* $z=1$	R_{off}	*Bx*	F_0	*Bx*	Q_0	*0*	C_4	*OVR*	$\overline{P}$	F_3	$\overline{G}$	*Bx*	*Bx*	Q_0
3. Инкрементирование															
0100	$F=S+1+C_0$	*Bx*	*Bx*	*P*	R_{off}	R_{off}	*0*	C_4	*OVR*	$\overline{P}$	F_3	$\overline{G}$	*z*	*z*	*z*
4. Преобразование «число со знаком – дополнительный код»															
0101	$F=S+C_0$ *при* $z=0$, $F=\neg S+C_0$ *при* $z=1$	*Bx*	*Bx*	*P*	R_{off}	R_{off}	*0*	C_4	*OVR*	$\overline{P}$	*	$\overline{G}$	S_3	*Bx*	*Bx*

Окончание табл. 10

Мк. код	Функция АЛУ	PF_3		PF_0	PQ_3	PQ_0		C_4	$\overline{P}$ / OVR		$\overline{G}$ / F_3		Z		
$I_8 - I_5$		*Ст.МПС*	*Мл.Ср.МПС*						*Ст.МПС*	*Ст.МПС*	*Ст.МПС*	*Мл.Ср.МП*	*Ст.МПС*	*Ср.МПС*	*Мл.МПС*
5. Последний цикл умножения в дополнительном коде															
0110	$F=S+C_0$ *при* $z=0$, $F=S-R-1+C_0$ *при* $z=1$	R_{off}	*Вх*	F_0	R_{off}	Q_0	*0*	C_4	*OVR*	$\overline{P}$	F_3	$\overline{G}$	*Вх*	*Вх*	Q_0
6. Нормализация обычной длины															
0111	$F=S+C_0$	F_3	F_3	R_{off}	Q_3	*Вх*	*0*	**	$Q_2 \oplus Q_1$	$\overline{P}$	Q_3	$\overline{G}$	***		
7. Нормализация двойной длины															
1010	$F=S+C_0$	$R_3 \oplus F_3$	F_3	*Вх*	Q_3	*Вх*	*0*	****	$F_2 \oplus F_1$	$\overline{P}$	F_3	$\overline{G}$	*****		
8. Деление в дополнительном коде															
1100	$F=S+R+C_0$ *при* $z=0$, $F=S-R-1+C_0$ *при* $z=1$	$\neg(R_3 \oplus F_3)$	F_3	*Вх*	Q_3	*Вх*	*0*	C_4	*OVR*	$\overline{P}$	F_3	$\overline{G}$	******	*Вх*	*Вх*
9. Деление в дополнительном коде (коррекция)															
1110	$F=S+C_0$ *при* $z=0$, $F=S-R-1+C_0$ *при* $z=1$	F_3	F_3	R_{off}	Q_3	*Вх*	*0*	C_4	*OVR*	$\overline{P}$	F_3	$\overline{G}$	******	*Вх*	*Вх*

* F_3, *если* $z=0$; $S_3 \oplus F_3$, *если* $z=1$.
** $Q_3 \oplus Q_2$ – *для Ст.МПС,* C_4 – *для остальных МПС.*
*** *z регистра Q.*
**** $F_3 \oplus F_2$ – *для Ст.МПС,* C_4 – *для остальных МПС.*
***** *z выходов АЛУ и регистра Q.*
****** $\overline{R_3 \oplus F_3}$.

2.3. Построение операционных устройств на базе МПС К1804ВС1

Требуемая разрядность операционного устройства обеспечивается объединением необходимого числа микропроцессорных секций. Например, если разрядность обрабатываемых в компьютере слов *D* равна 12, то схема вычислительного блока для данного частного случая будет иметь вид

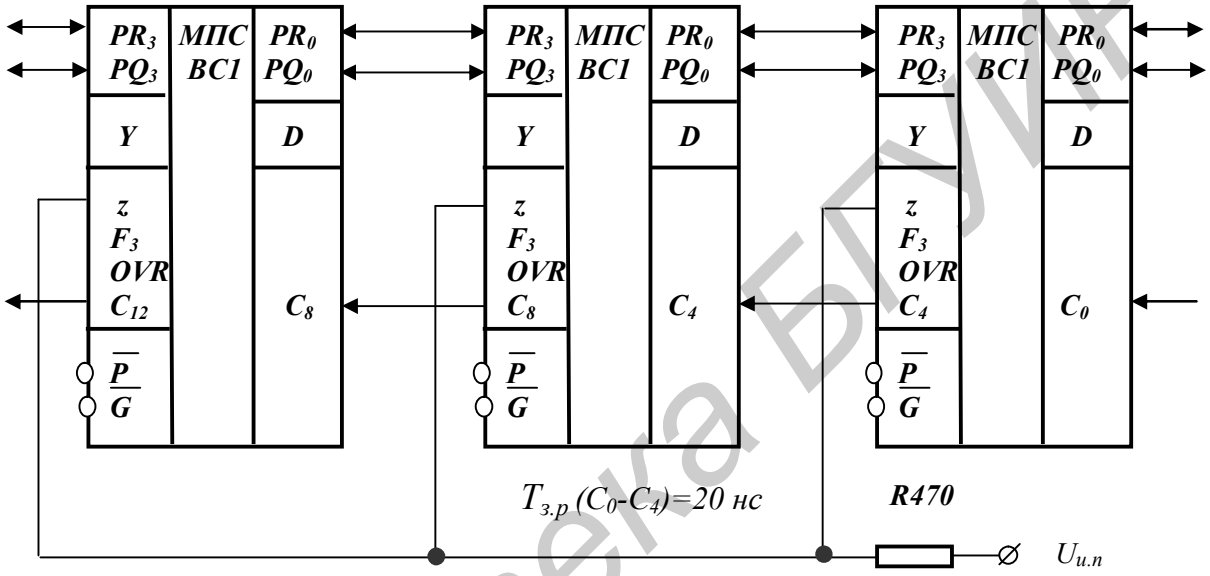

Рис. 18

Приведенное соединение *МПС* обеспечивает последовательное распространение переноса от младших к старшим разрядам. В связи с этим при синтезе многоразрядных процессоров приходится *решать задачу* уменьшения времени задержки распространения переноса с целью повышения скорости выполнения арифметических операций. Для решения данной задачи при проектировании применяют схемы ускоренного переноса (*СУП*) *К1804ВР1* (рис. 19).

Выходы $\overline{P},\overline{G}$ схемы ускоренного переноса используются для каскадного соединения нескольких *ИМС К1804ВР1*. В этом случае схема, приведенная на рис. 19, рассматривается как единый блок (рис. 20).

Рассмотрим схему распространения переноса в процессоре без блока СУП, а также при наличии данного быстродействующего модуля.

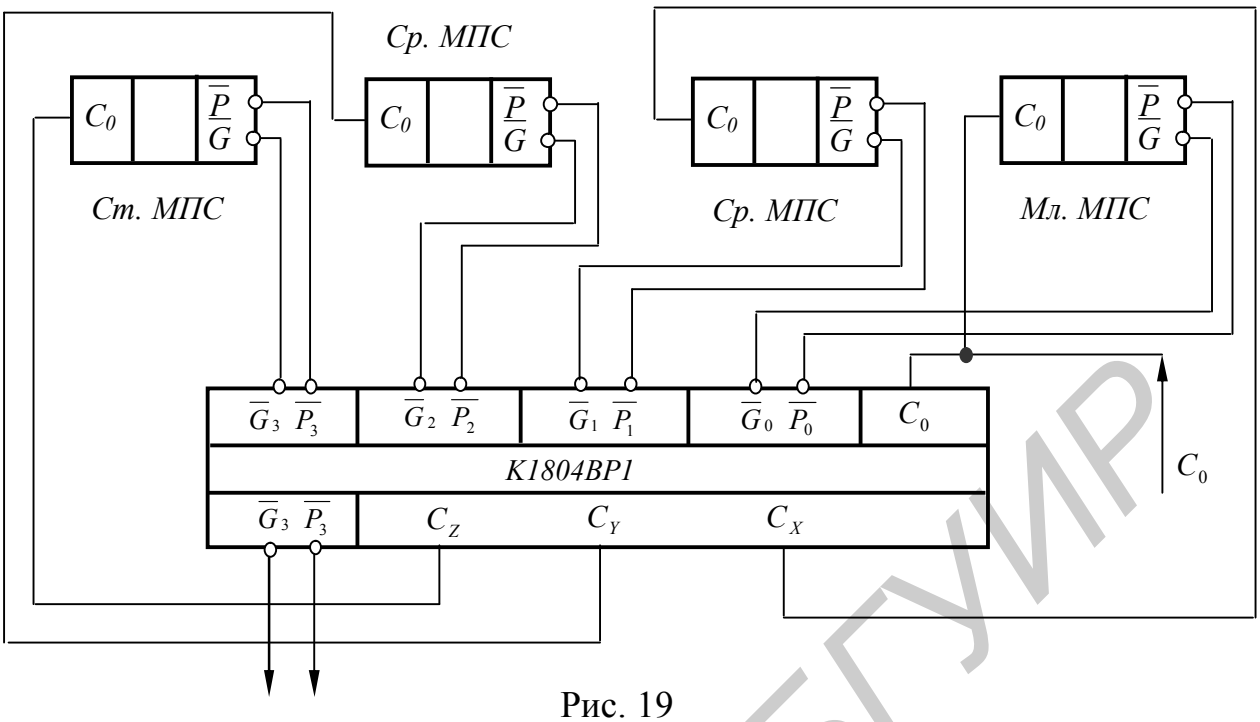

Рис. 19

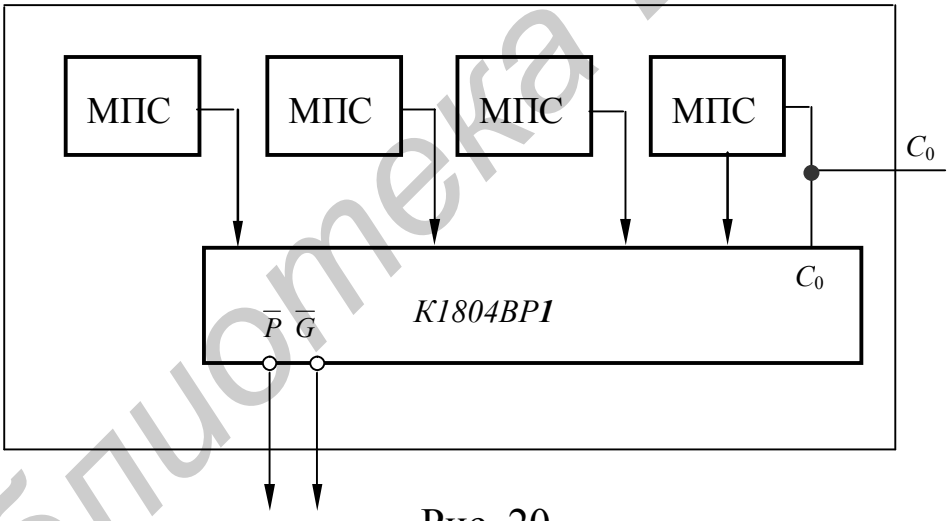

Рис. 20

Для первой *МПС* в блоке время задержки распространения определяется разницей во времени между моментом появления переноса C_4 на выходе *МПС* и моментом подачи двоичного кода на адресные входы *РЗУ*. Это время составляет величину примерно *70* нс. Для остальных секций время задержки распространения определяется равным $t_{з.р}$ $(C_0 - C_4)=20$ нс. Таким образом, в схеме, приведенной на рис. 19, $t_{з.р}$ переноса C_{12} на выходе *3*-й микросхемы составит $70 + 20 + 20 = 110$ нс.

В схеме с *СУП* аналогичный выход обозначен как C_z. Его время задержки определяется задержкой появления сигналов $\overline{P}$, $\overline{G}$ *МПС* относительно времени подачи адресов на входы *РЗУ* и задержкой в *СУП*, равной *20 нс* (общая задержка для всех *ИМС*). Итого: $t_{з.р}$= *79 нс.*

Другая задача, решаемая при объединении *МПС*, состоит в построении цепей передачи информации при организации сдвигов. Принципиально она может быть решена путем непосредственного соединения выходов PR_3 PQ_3 младших МПС со входами PR_0 PQ_0 старших секций. Однако в операционном устройстве остаются свободными соответствующие выходы младшей и старшей секций. Кроме того, решение ряда задач в процессоре может потребовать выполнения различных типов сдвигов. Каждая разновидность таких операций определяет различное соединение оставшихся свободными выводов или подачу на их входы определенной информации. Как правило, требуемые коммутации в БОД (блоке обработки данных) осуществляются с помощью специальной схемы – узла сдвига данных (рис. 21).

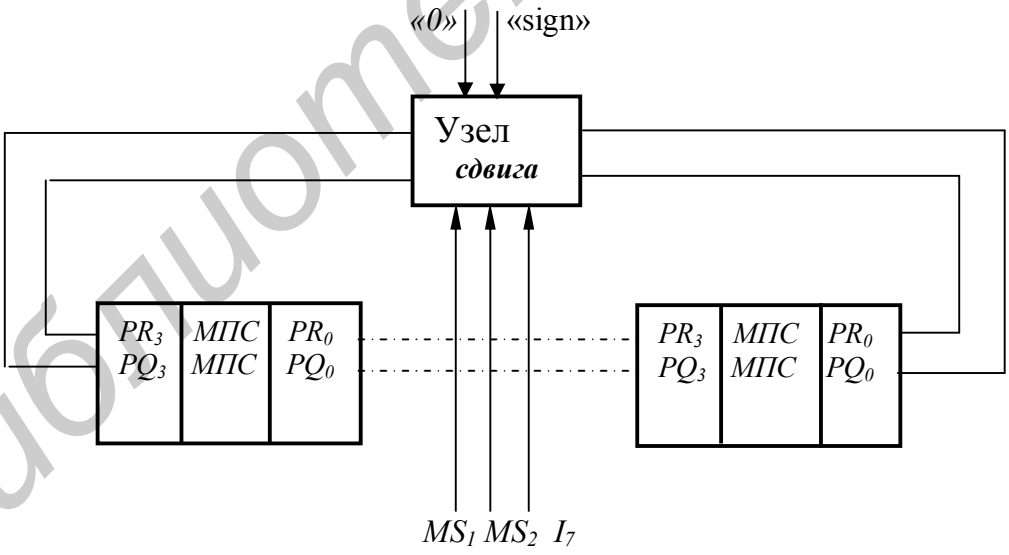

Рис. 21

На практике узел сдвигов может быть реализован с использованием мультиплексоров, управление которыми осуществляется сигналами MS_1 MS_2, а также сигналом микрокоманды I_7, который определяет направление сдвига. В

общем случае эти сигналы вносятся в структуру микрокоманды в специально отведенные разряды.

Третья задача, решаемая при построении операционных устройств, – это формирование и обработка слова состояния процессора, т.е. признаков, предназначенных для выполнения условных переходов.

Слово состояния операционного устройства включает в себя сигналы PR_3 PQ_3 на выходах старшей *МПС*, PR_0 PQ_0 на выходах младшей *МПС*, признаки z, F_3, OVR, C_4. Для хранения признаков в компьютерах используется регистр состояния *RG C* или регистр флагов (рис. 22).

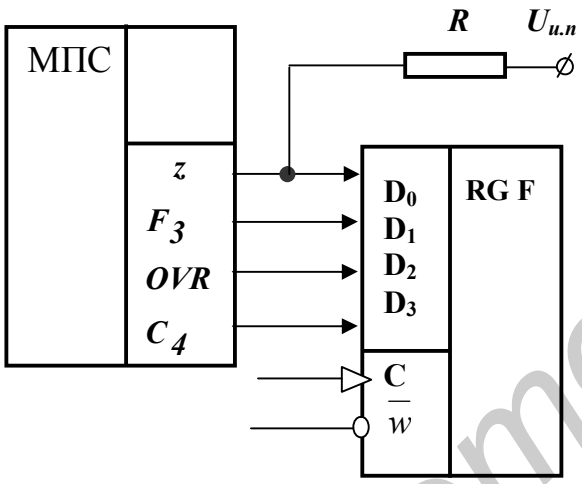

Рис. 22

Запись признаков в регистр осуществляется по синхросигналу *C* на соответствующем входе *RG F*. Сигнал на входе $\overline{w}$ используется для запрещения записи и организации ветвления в микропрограмме по нескольким условиям. При этом переход по каждому флагу осуществляется, как правило, с помощью отдельной микрокоманды. В связи с этим в течение всего времени ветвления по состоянию *RG F* (от *n*-й микрокоманды) вновь формируемые признаки результатов не запоминаются.

Для ускорения процесса ветвления и усложнения условия перехода используются специальные аппаратные средства, реализованные в виде *БИС К1804ВР2*. Данная схема предназначена для выполнения микроопераций сдвига и обработки слова состояния процессора. Схема соединения *СУСС* (схема управления состояниями и сдвигами) и *МПС* имеет вид, показанный на рис. 23.

БИС К1804ВР2 осуществляет следующие функции:

1) формирует сигналы входного переноса в МПС и схему ускоренного переноса;

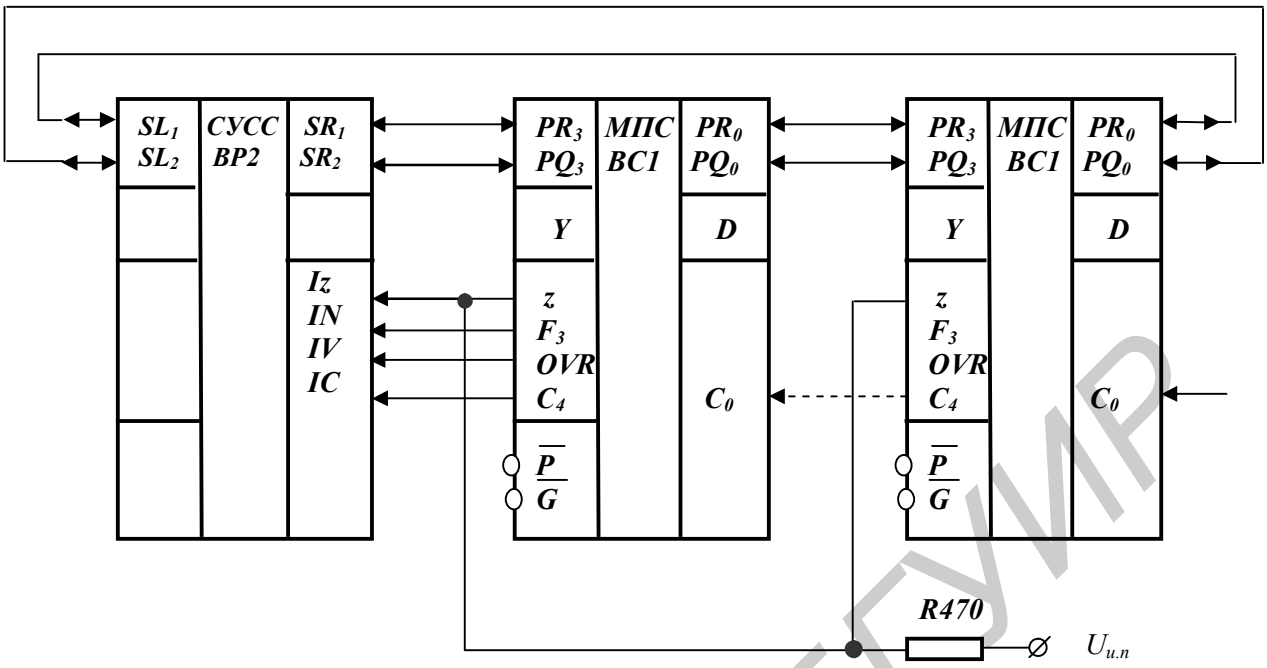

Рис. 23

2) выполняет арифметические, логические и циклические сдвиги чисел обычной и двойной длины;

3) осуществляет преобразование как целого *ССП,* так и отдельных бит любого из 2 регистров состояния (*RG* F_1 и *RG* F_2), входящих в состав *СУСС*;

4) выполняет проверку за один такт одной из 16 различных комбинаций условий, поступающих из внутренних регистров состояния или из *МПС*.

2.4. Микропрограммирование МПС

Определение: элементарная функциональная операция, выполняемая за один тактовый интервал времени и приводимая в действие одним управляющим сигналом, называется микрооперацией. Соответственно совокупность микроопераций, выполняемых параллельно во времени, называют микрокомандой.

Выполнение всех действий в процессоре осуществляется в виде отработки множества команд (процессоры с фиксированной архитектурой) или микроко-

манд. В частности, микропроцессорная секция *К1804ВС1* функционирует под действием микрокоманд, хранящихся в управляющей памяти. При этом всю последовательность микрокоманд, предназначенных для реализации некоторого преобразования, называют микропрограммой.

Стандартная структура микрокоманды обычно включает в свой состав две основные части: операционную и адресную. Адресная часть используется при формировании адреса следующей микрокоманды. При этом в состав компьютера включается специальный блок (*БМУ*), который позволяет выполнять различные действия над адресами.

Управление процессом преобразования данных осуществляется операционной частью микрокоманды. В этом случае микрооперации кодируются двоичными кодами и для кодирования используют три основных способа:

1. *Вертикальное кодирование* или *микропрограммирование.* Данный метод предполагает, что всей совокупности микроопераций присваиваются коды, образующие ряд целых чисел без знака. При отработке таких микрокоманд специальный дешифратор преобразует код микрокоманды в один управляющий сигнал, который и является сигналом микрооперации. Недостаток способа заключается: 1) в необходимости построения достаточно сложного дешифратора; 2) в увеличении длины микропрограммы (в одной микрокоманде содержится только одна микрооперация); 3) в отсутствии наглядности функционального назначения микрокоманды.

Число разрядов в операционной части при вертикальном микропрограммировании определяется по формуле

$$n_{о.ч} = \text{int} \log_2 m ,$$

где *int* – это ближайшее большее целое число; *m* – количество микроопераций.

2. *Горизонтальное микропрограммирование*. При данном способе каждому разряду операционной части микрокоманды ставится в соответствие определенная микрооперация. Наличие в разряде единицы говорит о том, что некото-

рая микрооперация выполняется, причем выполнение действий под управлением i разряда не зависит от состояния других j разрядов. Длина операционной части при горизонтальном микропрограммировании составляет число m, т.е. соответствует количеству микроопераций.

Достоинства метода: 1) возможность выполнения одновременно любого числа функционально совместимых микроопераций; 2) отсутствие схем дешифрации. Недостатки: 1) большая длина микрокоманды, достигающая в отдельных случаях сотен бит; 2) большая разрядность *ЗУ* микрокоманд или большая длительность выборки управляющих слов при малой разрядности *ЗУ Мк*.

3. *Смешанный способ микропрограммирования*. В данном случае сочетаются горизонтальный и вертикальный способы кодирования, при этом множество микроопераций m разбивается на k подмножеств, а в составе микрокоманды каждому подмножеству выделяется свое поле:

$$m = \bigcup_{i=1}^{k} m_i .$$

Соответственно каждое поле имеет свое функциональное назначение, а микрооперации внутри подмножества (поля) кодируются вертикально.

Длина операционной части микрокоманды (рис. 24) при смешанном микропрограммировании определяется величиной

$$n_{o.ч} = \sum_{i=1}^{k} \operatorname{int} \log_2 m_i ,$$

где k – число функциональных групп; m_i – число микроопераций в группе.

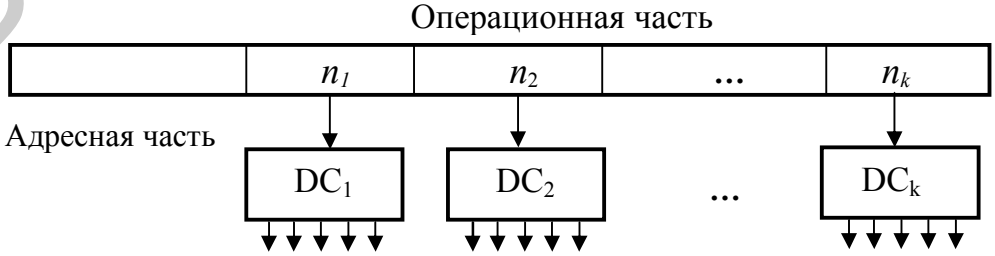

Рис. 24

Смешанный способ микропрограммирования применяется наиболее часто. Однако в данном случае требуется решать задачу разбиения множества всех микроопераций на непересекающиеся подмножества, т.е. микрооперации подмножества m_i не должны включаться в подмножество m_j и наоборот. Например: *МПС 1804ВС1* реализует 24 микрооперации, которые разбиты на 3 группы по 8 микроопераций. Каждая группа имеет свое функциональное назначение, принципиально отличное от преобразований, кодируемых в другой группе.

Теперь, с учетом методов кодирования микроопераций, рассмотрим процесс формирования структуры микрокоманды, достаточной для управления *БИС К1804ВС1*.

Итак, на первом этапе в состав микрокоманды включаются три поля для сигналов управления $I_0 - I_8$, что позволяет получить следующую структуру:

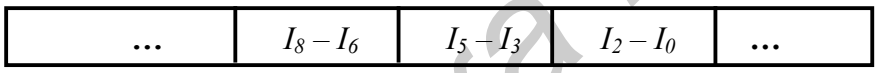

Рис. 25

Далее, будем учитывать наличие в архитектуре процессора POH_i, назначение которых состоит в хранении операндов или другой информации, используемых в вычислительном процессе. В общем случае номера регистров указываются явно, хотя допускается и неявное указание при адресации, например, регистра *Q*. Используя явный принцип адресации регистров, на данном этапе получаем следующую структуру *RG Mk*:

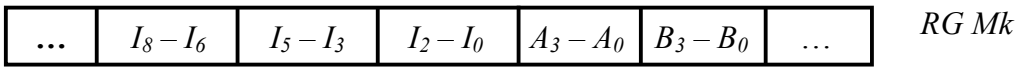

Рис. 26

Приведенная структура содержит основную информацию, необходимую для выполнения действий в *МПС*. Однако, как правило, в состав *RG Mk* включают ряд дополнительных сигналов или полей, необходимых для управления

другими структурными компонентами компьютера. К ним относят сигналы управления памятью, блоком микропрограммного управления, системой ввода-вывода, схемами внутреннего интерфейса.

Реализация адресной части микрокоманды зависит от используемого способа преобразования адресов и имеющегося комплекта аппаратных средств. Основные поля в этой части – это поле управления адресом следующей микрокоманды $P_{i-1} – P_0$, и поле адреса следующей микрокоманды $BR_{j-1} – BR_0$.

Поле управление адресом определяет источник адреса в *БМУ*, который подключается к адресной шине, если выполнено некоторое заданное условие или безусловно. При выполнении безусловного перехода адрес микрокоманды, как правило, извлекается из разрядов поля $BR_{j-1} – BR_0$, где программируется предварительно.

Заполнение полей любой микрокоманды производится в соответствии с ориентацией ее на выполнение определенных действий. При этом множество микроопераций *МПС* допускает многовариантность кодовых наборов при выполнении одной и той же логической или арифметической операции.

Для примера рассмотрим методику составления микрокоманды для управления данными в системе, использующей архитектуру процессорного комплекта *К1804ВС1*.

Пусть требуется выполнить увеличение содержимого регистра общего назначения с номером *0000*, входящего в состав *МПС*. Структуру микрокоманды, достаточную для управления данной системой, будем считать априорно заданной в виде

RG Mk

BR_3-BR_0	P_3-P_0	*Ms2* $I_8 – I_6$	*Ms1* $I_2 – I_0$	C_0 $I_5 – I_3$	$A_3 – A_0$	$B_3 – B_0$	$D_3 – D_0$

Рис. 27

Здесь сигналы *Ms2*, *Ms1* – это биты управления сдвигами (см. рис. 21); $D_3 – D_0$ – непосредственный операнд.

Пусть в начальный момент времени в регистре с адресом *0000* храниться нулевая константа. Тогда по табл. 2 (управление функцией *АЛУ)* находим микрокод $I_5 – I_3=000$, определяющий операцию суммирования вида $R+S+C_0$, где *R* и *S* – входы *АЛУ МПС*. Полагая значение C_0 равным «1», определим содержимое третьей тетрады *RG Mk* в виде (рис. 28).

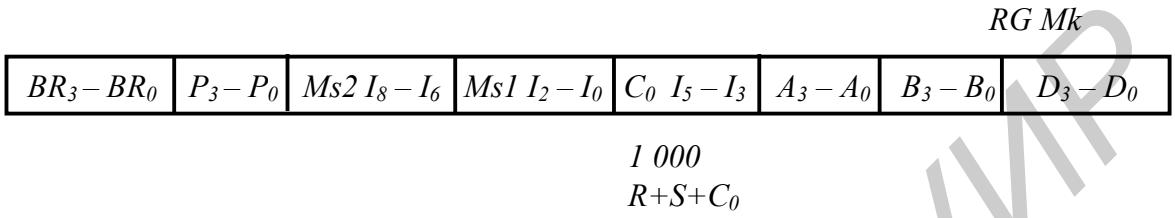

Рис. 28

Далее определим источники информации на входах *R* и *S АЛУ*. Для простоты положим один из операндов, например по входу *R*, равным нулю. Тогда из табл. 1 (управление данными) выберем микрокод $I_2 – I_0=011$, соответствующий подаче на входы *АЛУ* содержимого *РОН* по адресу из $B_3 – B_0$ и технологически «зашитой» константы «*0*». Таким образом, в структуре *RG Mk* на данный момент будут определены 2 поля:

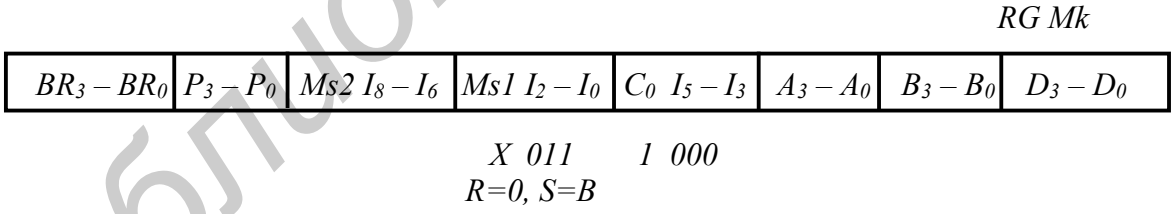

Рис. 29

Выберем в качестве приемника результата тот же *РОН*, который является источником операнда. Адрес этого регистра указываем в поле *B* и определяем запись результата в *РЗУ* по форме $F \rightarrow B$. Это действие в *МПС* производится под управлением микрокода $I_8 – I_6=011$ (см. табл. 3). В общем случае номер регистра, адресуемый в *RG Mk,* может быть любой в диапазоне $0–F_{16}(15_{10})$. В частном

случае при POH_i=0000 поле $B_3 - B_0$ будет иметь значение 0000, а структура микрокоманды вид

							RG Mk
$BR_3 - BR_0$	$P_3 - P_0$	$Ms2\ I_8 - I_6$	$Ms1\ I_2 - I_0$	$C_0\ I_5 - I_3$	$A_3 - A_0$	$B_3 - B_0$	$D_3 - D_0$
Адресная часть		*X 011* $F \rightarrow B$	*X 011* $R=0, S=B$	*1 000* $R+S+C_0$	*XXXX*	*0000* RG_0	*XXXX*

Рис. 30

Адресные линии $A_3 - A_0$ в нашем примере не используются, поэтому здесь может быть указан произвольный код *XXXX*. Программирование адресной части требует специальных знаний и может быть рассмотрено после изучения темы «Устройства управления».

2.5. Процессоры с фиксированной архитектурой

Деление структуры процессора на управляющий и операционный автоматы позволяет синтезировать исполнительную часть схемы в соответствии с некоторыми общими правилами.

Исходные данные для построения ОА формируются на основе выполняемых проектируемым процессором функций:

1) множество входных слов $DI=\{DI_1, \ldots, DI_i\}$, представляющих собой исходные операнды решаемой задачи;

2) множество выходных слов $DO=\{DO_1, \ldots, DO_j\}$, представляющих собой результаты операций;

3) множество внутренних слов $R=\{R_1, \ldots, R_p\}$, используемых для представления промежуточных результатов в процессе выполнения операций;

4) множество микрокоманд $Y=\{y_1, \ldots, y_m\}$, инициирующих преобразования над словами информации вида

$$y_q)\ R_u := \varphi_q\{R_t, \ldots, R_s\},$$

где φ_q – вычисляемая функция.

5) множество логических условий $X=\{x_1, \ldots, x_l\}$, где $x_w = ш_z\{R_t, \ldots, R_s\}$, $z=1, 2, \ldots, l$, а $ш_z$ – булева функция.

Таким образом, функция операционного автомата считается заданной, если определены множества *DI, DO, R, Y, X* и алгоритмы преобразования данных.

Правило синтеза канонической структуры операционного автомата заключается в выполнении следующих действий:

1. Словам *R,* определяемым алгоритмом в качестве внутренних, ставятся в соответствие регистры *DL* длиной, равной длине слов. Если слово *DL* разделяется на поля, то в соответствующем регистре выделяются подрегистры, ориентированные на хранение содержимого подполей.

2. Словам *DI*, определяемым алгоритмом в качестве входных, ставятся в соответствие входы структурной схемы *OA*. Каждый вход соединяется с соответствующим регистром входной шиной.

3. Словам *DO*, которые определены как выходные, ставятся в соответствие выходы структурной схемы *OA*. Каждый выход соединяется с соответствующим регистром выходной шиной.

4. Каждой микрокоманде y_q, которая определяется оператором присваивания $R_u := \varphi_q\{R_t, \ldots, R_s\}$, ставится в соответствие комбинационная схема φ_q, причем входы этой схемы подключаются к регистрам $R_t, \ldots, R_s$, а выходы – соединяются управляемой шиной с регистром R_u.

5. Каждому логическому условию $x_w = ш_z\{R_t, \ldots, R_s\}$ ставится в соответствие комбинационная схема, входы которой соединяются с регистрами, а выходы отмечаются сигналами x_w и используются управляющим автоматом для ветвления программы. В некоторых случаях комбинационная схема может отсутствовать, если сигнал x_w является выходом одиночного разряда регистра.

2.6. Типовые решающие узлы ОА

Микрооперация передачи информации между регистрами встречается в компьютерных системах наиболее часто. Схемотехнически она реализуется с

помощью группы логических элементов, управляемых от схемы *УА*. На практике межрегистровые связи организуются с учетом реальных требований к быстродействию и сокращению затрат на реализацию компьютерных устройств.

Наиболее часто встречаются следующие схемы передачи:

1. Прямая передача данных (рис. 31, 32).
2. Передача данных со сдвигом влево или вправо на один разряд (рис. 33, 34).
3. Передача данных с инверсией (образование обратного кода), со сдвигом на два разряда и т.д.

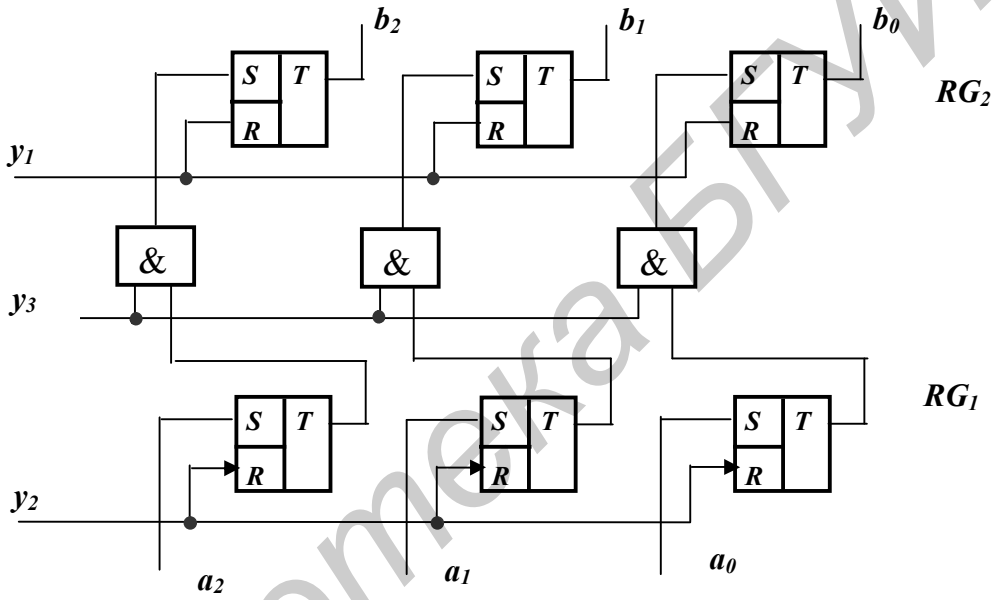

Рис. 31

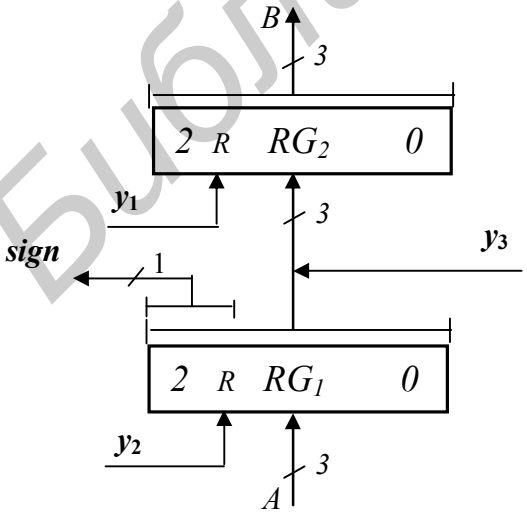

Рис. 32

На структурных схемах межрегистровая передача изображается в виде шины передачи информации, управляемой сигналом микрооперации (см. рис. 32). В приведенной схеме реализуются микрооперации:

$y_1)\ RG_2[2,0]:=0$
$y_2)\ RG_1[2,0]:=0$
$y_3)\ RG_2[2,0]:=RG_1[2,0]$.

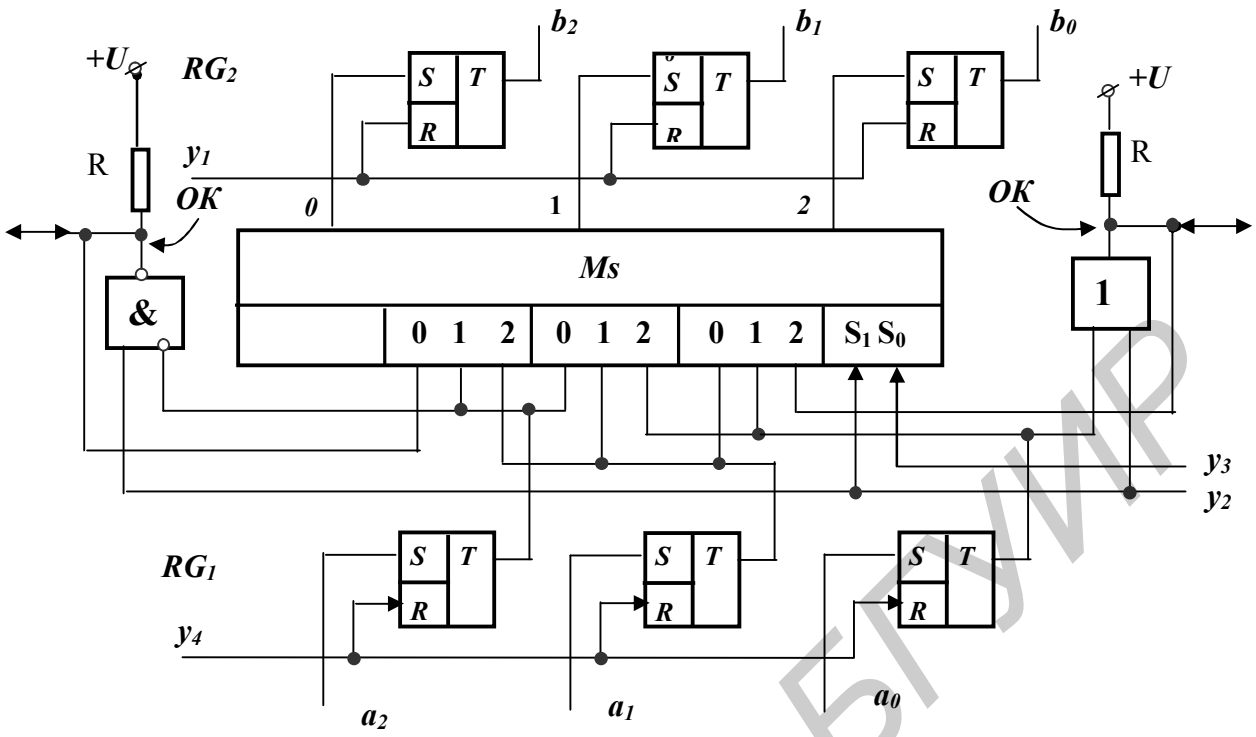

Рис. 33

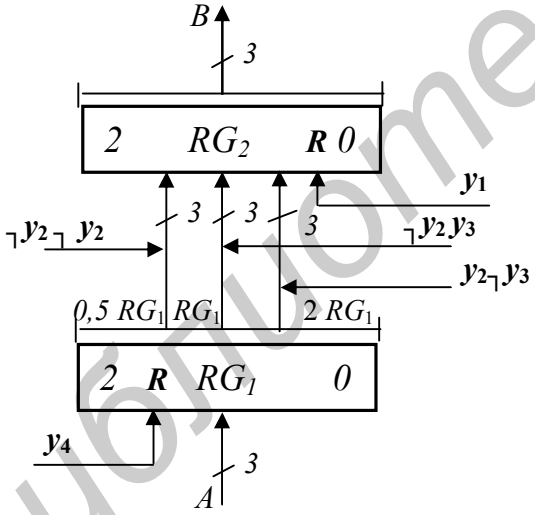

Рис. 34

В схеме (см. рис. 33, 34) реализуются следующие механизмы сдвига (*ОК* – открытый коллектор):

$\bar{y}_2 y_3$) $RG_2[2,0] := RG_1[2,0]$;

$\bar{y}_2 \bar{y}_3$) $RG_2[2,0] := R1(0.RG_1[2,0])$;

$y_2 \bar{y}_3$) $RG_2[2,0] := L1(RG_1[2,0].0)$;

$y_2 y_3$) – запрещено.

$\bar{y}_i = \neg y_i$

4. Для обмена данными в цифровых устройствах используют информационные каналы, которые называют шинами. В общем случае информация может передаваться по шине в виде последовательности бит по одной линии связи или же в виде слов по нескольким линиям. Соответственно и шины подразделяют на параллельные и последовательные.

При построении схем часто используют шинные формирователи (ШФ), позволяющие осуществить двунаправленную передачу данных. Одна из реализаций ШФ имеет вид, показанный на рис. 35.

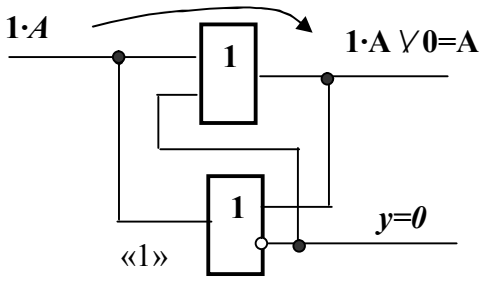

Рис. 35

В приведенной схеме выходы элементов коммутирующей логики должны быть выполнены по схеме с открытым коллектором. Тогда при *y=0* на выходе нижнего элемента будет сформирована логическая единица. Потенциал логической единицы, умноженный на передаваемую информацию монтажно, дизъюнктивно складывается с сигналом *y=0* на верхнем логическом элементе. В итоге в правую сторону по двунаправленной шине передается неискаженная информация.

Другой способ организации двунаправленной шины предполагает использование трехстабильных управляемых усилителей. Схема шинного формирователя при этом имеет следующий вид:

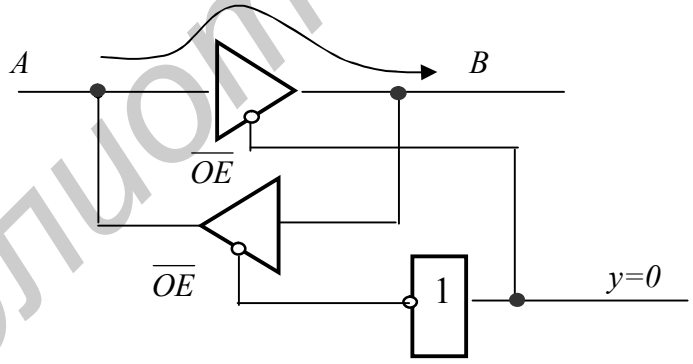

Рис. 36

5. Накапливающий и двоично-десятичный сумматоры. Накапливающий сумматор предназначен для хранения информационного слова и выполнения микрооперации сложения:

$$RGCм[n\text{-}1.0] := RGCм[n\text{-}1.0] + RGA[n\text{-}1.0] + C_0 \ .$$

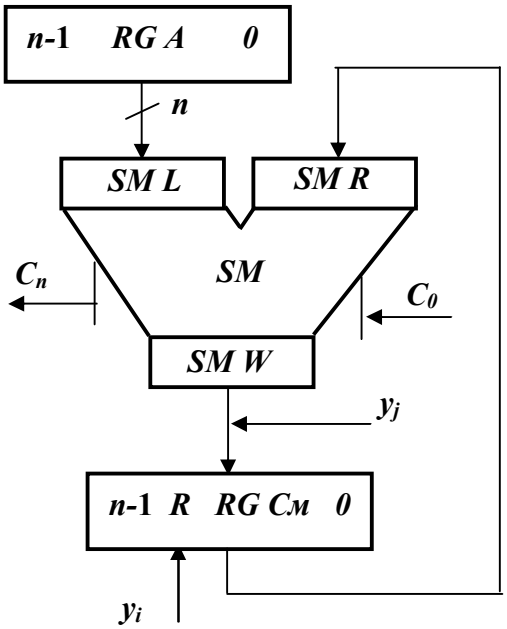

Рис. 37

Приведенная на рис. 37 схема предполагает распространение переноса последовательно от младших к старшим разрядам. Это определяет низкое быстродействие устройства и соответственно малые затраты аппаратуры.

Большее быстродействие может быть достигнуто в схемах, реализующих ускоренный перенос. Сюда относят устройства с групповым, параллельным и другими видами распространения переноса. Однако высокое быстродействие схемы предполагает увеличение сложности и снижение надежности изделия.

Ряд алгоритмов, реализуемых в схемах *ОА*, выполняет операции над числами, представленными в двоично-десятичном коде. При построении таких устройств следует помнить, что одно 4-разрядное двоичное слово позволяет представить десятичные числа в диапазоне от 0_{10} до 15_{10}. При двоично-десятичном кодировании из *16* используется только *10* комбинаций. Поэтому, если при суммировании получилось одно из значений в диапазоне $10_{10} - 15_{10}$, то оно будет содержать псевдотетраду, коррекция которой осуществляется добавлением числа $6_{10}=0110_2$. Построение схемы двоично-десятичного сумматора осуществляется с учетом следующих соображений. Так, появление псевдотетрады в сумме $S`=A+B+C_0$ предполагает формирование кодовых комбинаций: 1010, 1011, 1100, 1101, 1110, 1111. При склеивании соответствующие конъюнкции приводят к ДНФ вида $11xx \vee 1x1x$. Далее следует учесть, что появление сигнала переноса C_4 также свидетельствует о двоично-десятичном переполнении. Таким образом, применение схемы ДНФ, регистрирующей указанные

комбинации и сигнал переноса C_4, позволяет сформировать структурную схему, автоматически корректирующую результат суммирования:

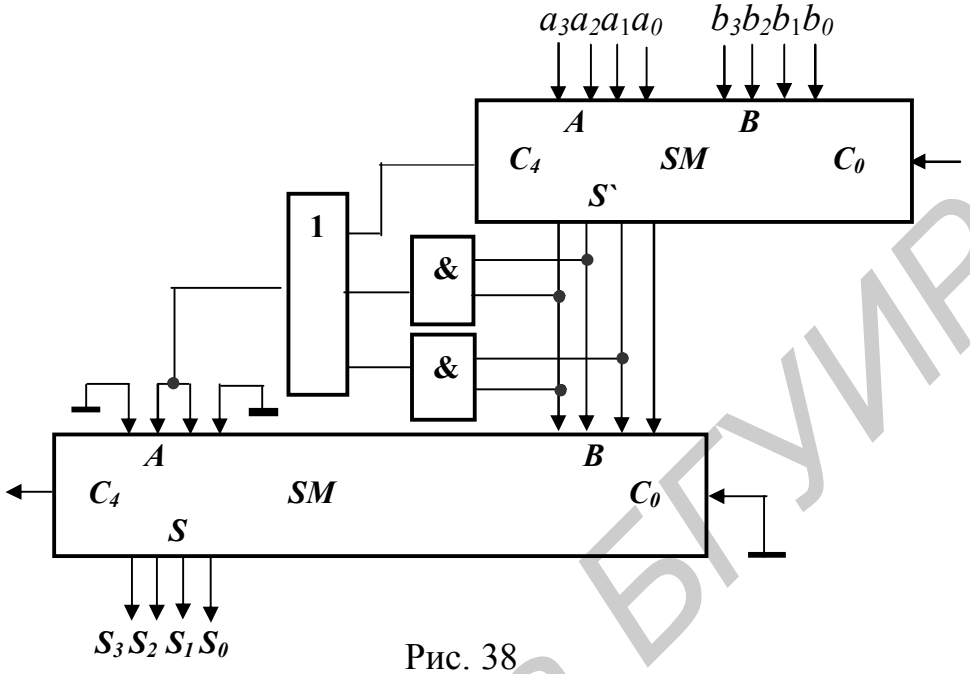

Рис. 38

2.7. Синтез ОА с элементами управляющей логики

Выполним синтез устройства, предназначенного для умножения двух чисел по методу Мак-Сорли. Сущность метода заключается в умножении чисел с одновременным анализом двух разрядов множителя (*Мт*), начиная с его старших разрядов. Алгоритм умножения по данному методу задается табл. 11:

Таблица 11

Мл. разряд пред. пары	Пара разрядов *Мт*	Знак дейст.	Кратн. *Мн.*
0	00	*нет*	0
0	01	+	*2 Мн.*
0	10	+	*2 Мн.*
0	11	+	*4 Мн.*
1	00	–	*4 Мн.*
1	01	–	*2 Мн.*
1	10	–	*2 Мн.*
1	11	*нет*	0

Выполним умножение операндов, выбрав в качестве множимого и множителя значения: $Мн = 51_{10} = 110011_2$, $Мт = 54_{10} = 110110_2$.

Таблица 12

00000000000000 110011 110110	*См* *Мн* *Мт*
00001100110000 0110	*См+4Мн, См* $\cdot 2^2$ *Мт*$\cdot 2^2$
11111110011010 00001011001010 00101100101000 10	$[-2Мн]_д$ $См+[-2Мн]_д$ $См\cdot 2^2$ $Мт\cdot 2^2$
11111110011010 00101011000010 10101100001000 00	$[-2Мн]_д$ $См+[-2Мн]_д$ $См\cdot 2^2$ $Мт\cdot 2^2$
101011000010	*Произведение*

В приведенном примере использован алгоритм умножения со сдвигом множителя и частичных произведений влево:

$Мн = 51_{10}$
$Мт = 54_{10}$
———————
$Произв. = 2754_{10}$

Анализ метода умножения позволяет заключить следующее:

1) регистр множителя должен иметь $n+1=7$ разрядов, предназначенных для хранения операнда и младшего разряда предыдущей пары;

2) регистр множимого должен иметь длину, равную длине регистра суммы, так как суммирование отрицательного множимого и сумматора требует дополнения *RG Мн* в старших разрядах единичными битами, а длина сумматора в рамках данного алгоритма равна $2n+2=14$ разрядам;

3) управляющая логика комбинационного типа подключается к старшим разрядам регистра множителя *RG Мт*[*6,4*] и формирует все требуемые табл. 11 импульсы управления: y_v – вычитание, y_s – сложение, y_m – формирование *4Мн*;

4) формирование требуемых кодов и кратности *Мн* будем выполнять с помощью логики инвертирования и мультиплексирования.

Формирование сигналов управления на выходе блока «Управляющей логики» может быть записано с использованием булевых равенств на основании таблицы умножения:

$$y_v = (\bar{x}_1\bar{x}_2 \vee \bar{x}_1 x_2 \vee x_1\bar{x}_2)T_\partial = x_1 T_\partial \vee \bar{x}_2 T_\partial, \qquad x_1 = RGMm\,[5],$$
$$y_s = (\bar{x}_1 x_2 \vee x_1\bar{x}_2 \vee x_1 x_2)\overline{T}_\partial = \bar{x}_1\overline{T}_\partial \vee x_2\overline{T}_\partial, \qquad x_2 = RGMm\,[4],$$
$$y_m = \bar{x}_1\bar{x}_2 T_\partial \vee x_1 x_2 \overline{T}_\partial.$$

В целом схема, иллюстрирующая логику преобразования операндов, показана на рис. 39.

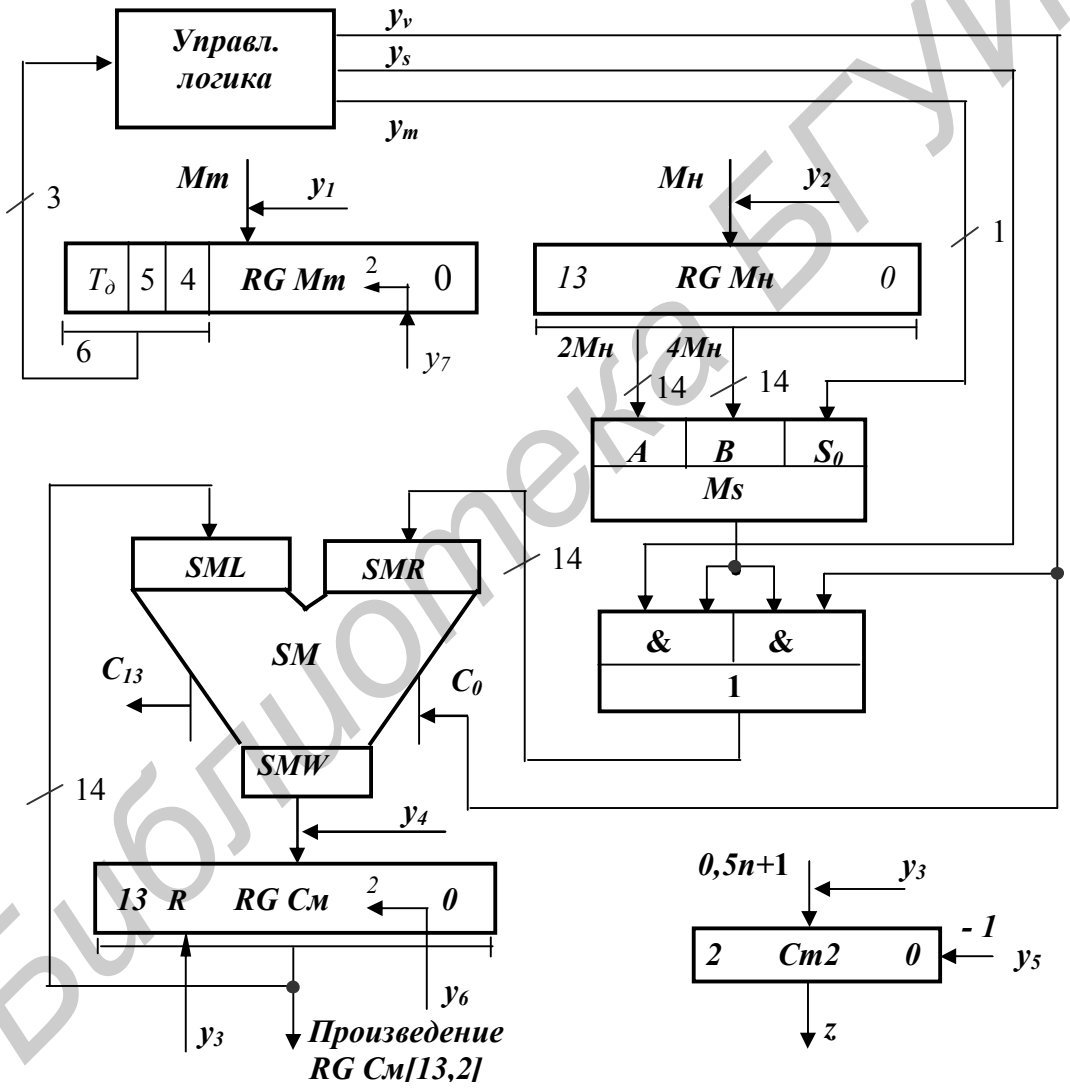

Рис. 39

Процесс умножения чисел в приведенном устройстве интерпретирует микропрограмма:

1) $y_1: RG\ Mm[5,0] := Mm,\ T_0 := 0,$
   $y_2: RG\ Mн[13,0] := Mн,$
   $y_3: RG\ Cм[13,0] := 00...0,\ Cm2\ [2,0] := \frac{n}{2} + 1 = 4,$
2) $y_4: RG\ Cм[13,0] := RG\ Cм[13,0] + L2(RG\ Mн[13,0].00),$
   $y_5: Cm2\ [2,0] := Cm2\ [2,0] - 1,$
   $y_6: RG\ Cм[13,0] := L2(RG\ Cм[13,0].00),$
   $y_7: RG\ Mm[6,0] := L2(RG\ Mm[6,0].00),$
3) *выполнение микрокоманд п. 2, если* $z = 0$, *продолжить при* $z = 1$,
4) *конец.*

Построенный операционный автомат формирует произведение операндов без анализа сигналов-признаков схемой управляющего устройства. Эту часть функций берет на себя внутренняя «управляющая логика», которая и формирует требуемые сигналы. Однако, если все функции управления возложить на автомат Мили, Мура или микропрограммное УУ, то алгоритм умножения следует вначале представить с помощью ГСА, после этого, используя стандартные методики синтеза или эвристический подход (для схем малой сложности), построить операционное устройство *I*-, *M*- или *IM*-типа.

Итак, в соответствии с таблицей умножения и выполненным примером составим *ГСА* для алгоритма умножения по методу Мак-Сорли (рис. 40). Схема операционного автомата, соответствующего данной *ГСА,* будет иметь вид, показанный на рис. 41.

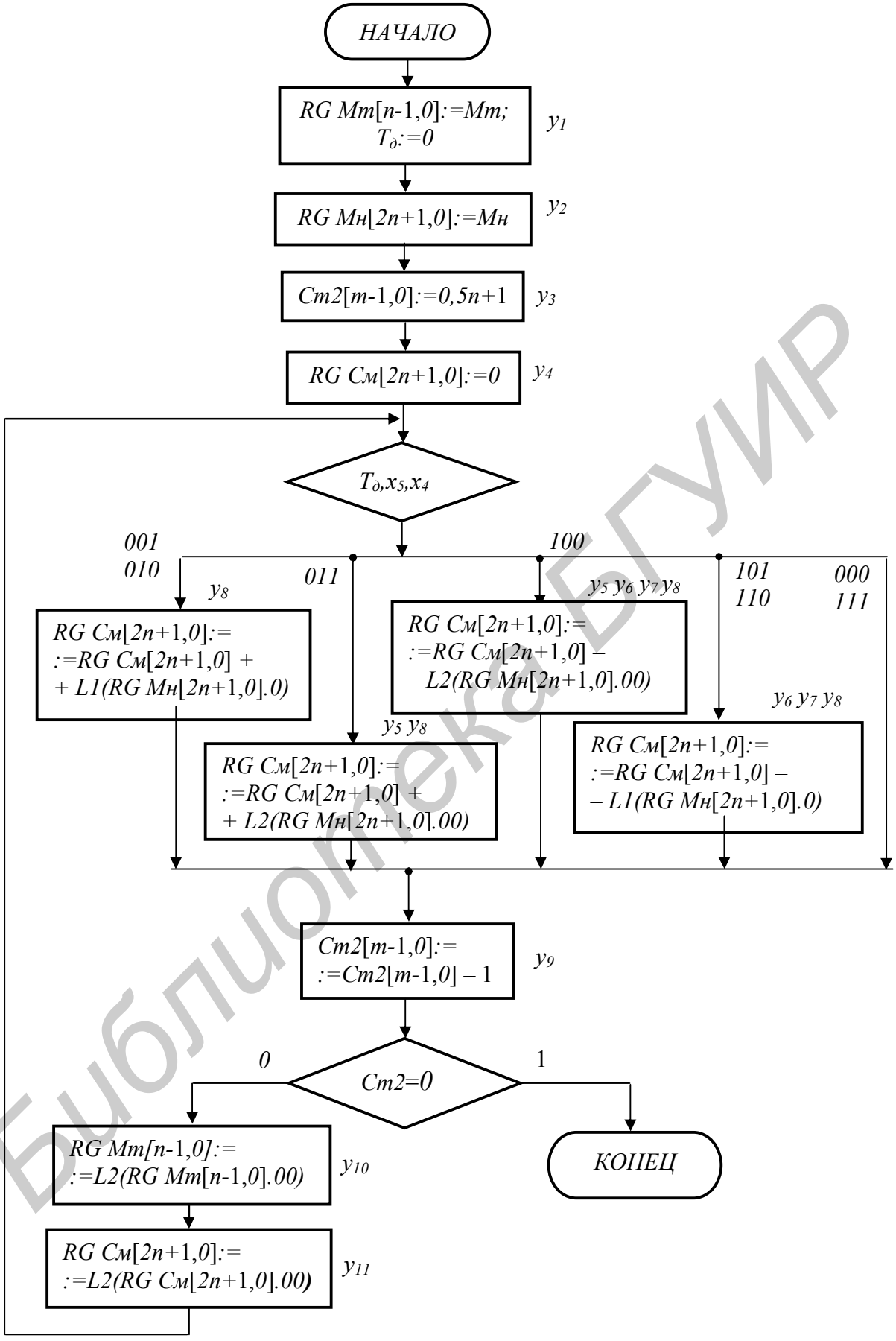

Рис. 40

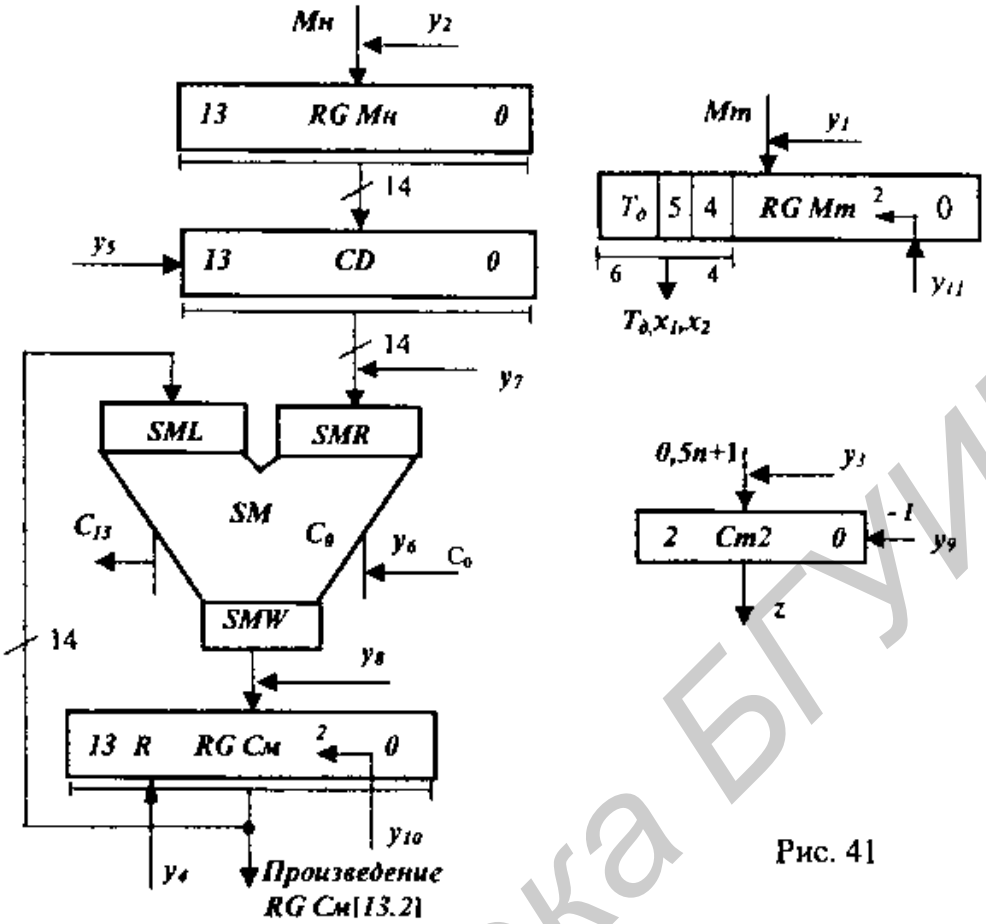

Рис. 41

2.8. Синтез *M*-автоматов

Как правило, процедура синтеза *M*-автоматов сводится к проектированию управляемых источников операндов, построению схемы ЛУ комбинационного типа, проектированию схем коммутации результатов вычислений в регистры *ОА*. Обобщенная структура *M*-автомата имеет вид

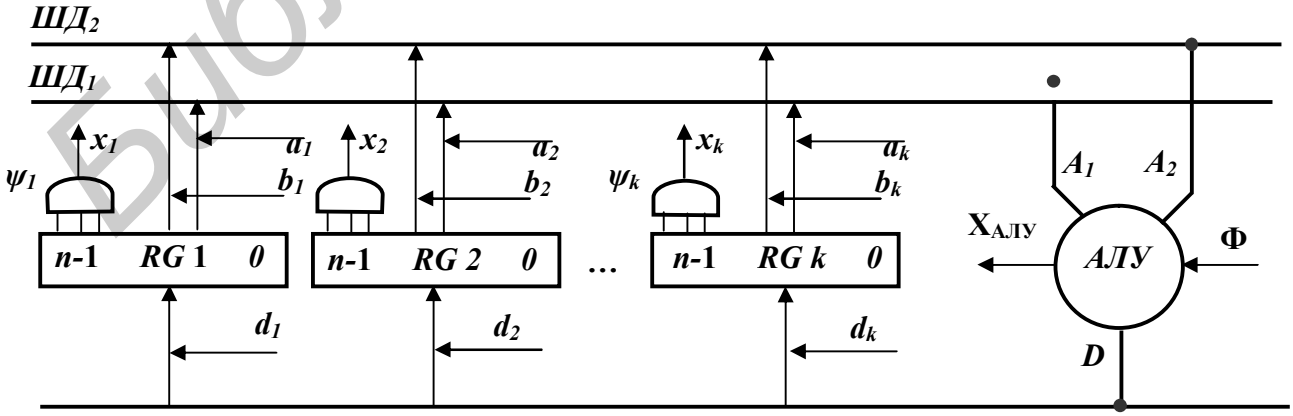

Рис. 42

В общем случае устройства данного типа имеют минимальную аппаратную сложность, однако производительность при этом также оказывается минимальной и равна одной микрокоманде за такт. В отличие от автоматов других типов *М*-автомат содержит одну обобщенную схему АЛУ, которая позволяет выполнять все микрооперации из множества *Ф*.

Логические условия в автоматах данного типа вычисляются таким же образом, что и в любой канонической структуре, т.е. соответствующие схемы ψ подключаются к выходам регистров, и, кроме того, схема АЛУ помимо функций обработки данных выполняет дополнительно функции по формированию логических условий.

Рассмотрим этапы синтеза *М*-автомата на примере построения схемы или устройства умножения по методу Мак-Сорли, т.е. умножение на *2* разряда одновременно начиная с младших разрядов *Мт*.

Итак, пусть задан алгоритм умножения (табл. 13), а процесс умножения иллюстрирует табл. 14. И пусть исходные значения операндов *Мн* и *Мт* равны:

$$\frac{Мн=110110_2=54_{10}\,,\quad Мт=100011_2=35_{10}}{Мн\cdot Мт=1890_{10}}\,.$$

Таблица 13

Пара разр. Мт	Допол. ед. из пред. пары	Доп. ед. в след. пару	Знак дейст.	Кратн. Мн.
00	0	0	*нет*	0
01	0	0	+	1
10	0	0	+	2
11	0	1	–	1
00	1	0	+	1
01	1	0	+	2
10	1	1	–	1
11	1	1	*нет*	0

Таблица 14

000000000000 000000110110 100011	*См* *Мн* *Мт*
111111001010 000011011000 001000	$См+[Мн]_Д$ $Мн\cdot 2^2$ $Мт\cdot 2^{-2}$
000010100010 001101100000 000010	*См+Мн* $Мн\cdot 2^2$ $Мт\cdot 2^{-2}$
011101100010 110110000000 000000	*См+2Мн* $Мн\cdot 2^2$ $Мт\cdot 2^{-2}$
011101100010	*=1890*

В соответствии с табл. 13 и примером умножения (табл.14) построим граф-схему алгоритма (рис. 43). При этом этапы синтеза *М*-автомата будут состоять в следующем.

1-й этап – распределение регистров по шинам.

В структуре *М*-автомата для передачи операндов используются две шины – *ШД*$_1$ и *ШД*$_2$. Они соединяются со входами *АЛУ* A_1 и A_2 соответственно. Регистры устройства $RG_1,...,RG_k$ подключаются к шинам через управляемые выходы. Управление осуществляется с помощью сигналов $a_1,...,a_k$ и $b_1,...,b_k$.

Таким образом, на первом этапе проектирования предполагается разделение множества источников операндов *АЛУ* на два подмножества

$$RA_1 = \{RG_1,...,RG_i\} \quad \text{и} \quad RA_1 = \{RG_j,...,RG_k\}.$$

Эти подмножества должны удовлетворять условиям:

1) если регистры RG_i и RG_j являются операндами одной микрокоманды, то они включаются в различные подмножества;

2) каждое слово RG_s должно принадлежать хотя бы одному из подмножеств RA_1 или RA_2 и $RA_1 \cup RA_2 = R$;

3) подмножества RA_1 и RA_2 формируются таким образом, чтобы суммарные затраты аппаратуры в схемах коммутации операндов были бы минимальными.

Для определения элементов, принадлежащих RA_1 и RA_2, построим табл.15 микрокоманд ГСА и выполним распределение регистров автомата по шинам. В результате множество регистров разделится на подмножества:

$$RA_1 = \{RG\,Cм\} \text{ и } RA_2 = \{RG\,Mн, RG\,Mm, RG\,Cm2\}.$$

2-й этап синтеза М-автомата – определение форматов слов на ШД.

Сформированные подмножества регистров совместим по младшим разрядам. Тогда количество линий в *ШД*$_1$ и *ШД*$_2$ будет определяться максимальным числом разрядов в слове из RA_1 или RA_2 соответственно.

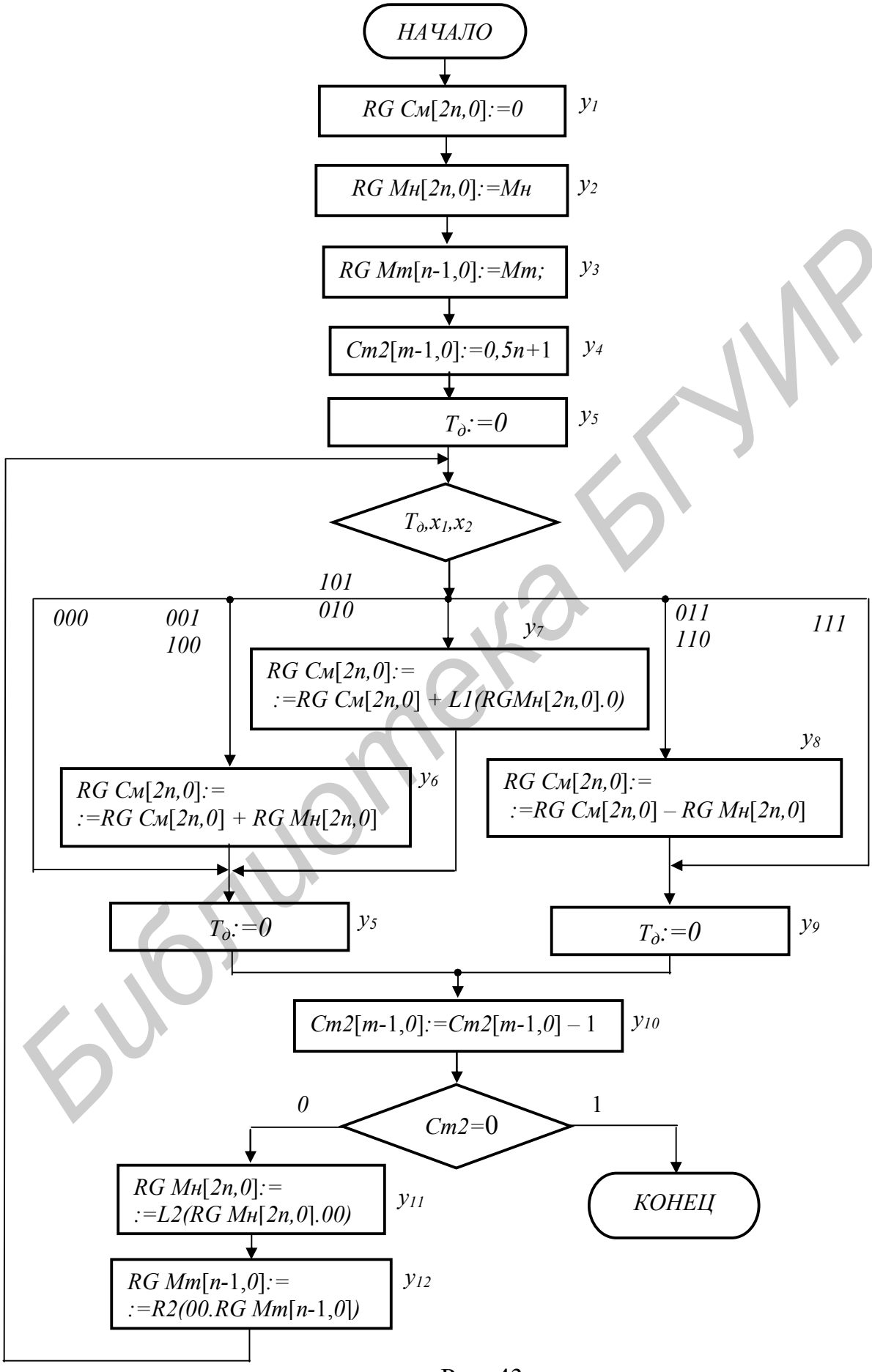

Рис. 43

Таблица 15

y_m	Микрокоманда	$ШД_1$	$ШД_2$
y_1	*RG См* [*12,0*]:= *0*	–	–
y_2	*RG Мн* [*12,0*]:= *Мн*	–	*Мн*
y_3	*RG Mm* [*5,0*]:= *Mm*	–	*Mm*
y_4	*RG Cm2* [*2,0*]:= *0,5n*+1=*4*	–	*0100*
y_5	$T_д$[*0*]:=*0*	–	–
y_6	*RG См* [*12,0*]:= *RG См* [*12,0*]+ *RG Мн* [*12,0*]	*RG См* [*12,0*]	*RG Мн* [*12,0*]
y_7	*RG См* [*12,0*]:= *RG См* [*12,0*]+*L1(RG Мн* [*12,0*].*0)*	*RG См* [*12,0*]	*RG Мн* [*12,0*]
y_8	*RG См* [*12,0*]:= *RG См* [*12,0*]+ ┐ *RG Мн* [*12,0*]+1	*RG См* [*12,0*]	*RG Мн* [*12,0*]
y_9	$T_д$[*0*]:=1	–	–
y_{10}	*RG Cm2* [*2,0*]:= *RG Cm2* [*2,0*] – 1	–	*RG Cm2* [*2,0*]
y_{11}	*RG Мн* [*12,0*]:= *L2(RG Мн* [*12,0*].*00)*	–	*RG Мн* [*12,0*]
y_{12}	*RG Mm* [*5,0*]:= *R2(00.RG Mm* [*5,0*]*)*	–	*RG Mm* [*5,0*]

Для рассматриваемого примера получим следующие форматы слов:

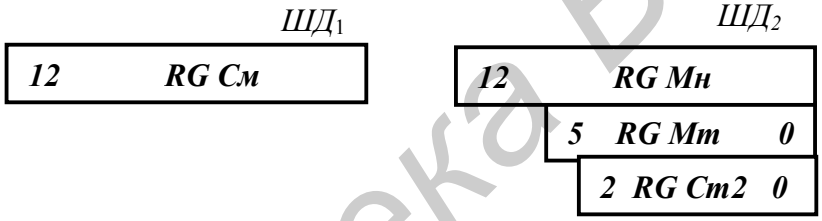

Рис. 44

Таким образом, каждая из шин будет содержать по 13 информационных линий. Очевидно, если число разрядов у одного из регистров оказывается меньше максимального (например в счетчике циклов), то осуществляется совмещение регистров по младшим битам, а при передаче информации по шинам – заполнение свободных разрядов константами 0 и 1.

3 этап синтеза – анализ преобразований в АЛУ.

На данном этапе микрокомандам *ГСА* ставятся в соответствие операторы присваивания *АЛУ*. Эти операторы характеризуют действия, которые выполняются непосредственно над входной информацией, а не источники или приемники данных. При этом используется ряд соглашений:

1) на входы A_1 и A_2 *АЛУ* поступают только прямые коды операндов;

2) неопределенный набор входных управляющих сигналов *АЛУ*, а следовательно, и неопределенное преобразование в схеме порождают на его выходе нулевое значение;

3) младшие разряды выходов *АЛУ* соответствуют младшим разрядам приемников результата.

Учитывая данные допущения, составим таблицу соответствия микрокоманд y_m *ГСА* умножения и операторов φ_s проектируемого *АЛУ*.

Таблица 16

y_m	Операторы АЛУ	Прием.	Обоз.
	$D:=\varphi_s(A_1,A_2)$	*RG*	φ_s
y_1	$D:=00\ldots0$	*RG См*	(φ_1)
y_2	$D:=A_2\,[12,0]$	*RG Мн*	φ_2
y_3	$D:=A_2\,[12,0]$	*RG Мт*	φ_2
y_4	$D:=A_2\,[12,0]$	*RG Ст2*	φ_2
y_5	$D:=00\ldots0,\ T_д:=0$	$T_д$	φ_1
y_6	$D:=A_1\,[12,0]+A_2\,[12,0]$	*RG См*	φ_3
y_7	$D:=A_1\,[12,0]+L1\,(A_2\,[12,0].0)$	*RG См*	φ_4
y_8	$D:=A_1\,[12,0]+\neg A_2\,[12,0].0+1$	*RG См*	φ_5
y_9	$D:=00\ldots01,\ (T_д:=1)$	$T_д$	φ_6
y_{10}	$D:=A_2\,[12,0]+111\ldots1$	*RG Ст2*	φ_7
y_{11}	$D:=L2\,(A_2\,[12,0].00)$	*RG Мн*	φ_8
y_{12}	$D:=R2\,(00.A_2[12,0])$	*RG Мт*	φ_9

Примечание к табл. 16. Вместо микрокоманды y_1 может быть использована запрещенная комбинация сигналов *АЛУ*. Это дает возможность сформировать нулевой операнд на выходе, а соответствующий сигнал управления (y_1) удалить из *ГСА*. Оператор *АЛУ* φ_1 тогда может использоваться для сброса $T_Д := 0$.

На этом же этапе синтеза могут быть построены таблицы для выбора операндов *АЛУ* (или управления операндами) и приемников результата.

Присвоим регистрам проектируемого устройства номера от 1 до 4:

$$RG\,См = RG_1,\ RG\,Мн = RG_2,\ RG\,Мт = RG_3,\ RG\,Ст2 = RG_4.$$

Тогда сигналы управления операндами и результатами вычислений будут иметь состав, указанный в табл. 17, 18.

4-й этап синтеза М-автомата – кодирование микроопераций наборами управляющих сигналов.

На данном этапе микрокоманды $y_m \subset \{Y\}$ исходной *ГСА* заменяется наборами управляющих сигналов a_i, b_j, d_k, φ_s (табл. 19).

Таблица 17

Источник операндов		*Сигнал* управления	
$ШД_1$	$ШД_2$	a_i	b_j
RGCм	–	a_1	–
–	*RGMн*	–	b_2
–	*RGMm*	–	b_3
–	*RG Cm2*	–	b_4

Таблица 18

Приемник результата	*Сигнал* управления
$D \rightarrow RG_k$	d_k
D→RGCм	d_1
D→RGMн	d_2
D→RGMm	d_3
D→RG Cm2	d_4
$D \rightarrow T_o$	d_5

Таблица 19

y_m	a_i	b_j	d_k	φ_s
y_1	–	–	d_1	φ_1
y_2	–	–	d_2	φ_2
y_3	–	–	d_3	φ_2
y_4	–	–	d_4	φ_2
y_5	–	–	d_5	φ_1
y_6	a_1	b_2	d_1	φ_3
y_7	a_1	b_2	d_1	φ_4
y_8	a_1	b_2	d_1	φ_5
y_9	–	–	d_5	φ_6
y_{10}	–	b_4	d_4	φ_7
y_{11}	–	b_2	d_2	φ_8
y_{12}	–	b_3	d_3	φ_9

Приведенная в табл. 19 информация может быть использована далее для перекодирования исходной ГСА и построения управляющего автомата.

5-й этап синтеза – составление подмножеств (или классов) эквивалентных операторов.

С целью оптимизации затрат аппаратуры в комбинационном *АЛУ* и формализации процесса проектирования ОА множество операторов $\{\varphi_s\}$ разбивается на классы k_1, …, k_j эквивалентных операторов. При этом под эквивалентны-

ми понимают операторы, реализующие в арифметико-логическом устройстве однотипные действия.

Пусть класс k_1 образуют операторы, реализующие функцию установки регистров проектируемого устройства. Список подмножества преобразований, входящих в данный класс, будет иметь вид

$$k_1 = \begin{Bmatrix} \varphi_1 \\ \varphi_6 \end{Bmatrix} = \begin{Bmatrix} D := 00...00 \\ D := 00...01 \end{Bmatrix}.$$

В класс k_2 будем включать операторы, которые характеризуют действия, соответствующие сложению чисел в дополнительных кодах

$$k_2 = \begin{Bmatrix} \varphi_2 \\ \varphi_3 \\ \varphi_5 \\ \varphi_7 \end{Bmatrix} = \begin{Bmatrix} D := 0 + A_2[12,0] + 0 \\ D := A_1[12,0] + A_2[12,0] + 0 \\ D := A_1[12,0] + \overline{A_2[12,0]} + 1 \\ D := 0 + A_2[12,0] + 11...1 \end{Bmatrix}.$$

В остальные классы включаем по одному оператору, так как они характеризуют различные типы действий:

$$k_3 = \{\varphi_4\} = \{D := A_1[12,0] + L1(A_2[12,0].0)\},$$
$$k_4 = \{\varphi_8\} = \{D := L2(A_1[12,0].00)\},$$
$$k_5 = \{\varphi_9\} = \{D := R2(00.A_2[12,0])\}.$$

6-й этап синтеза – построение обобщенных операторов.

На данном этапе для каждого класса эквивалентных операторов k_1, …, k_j строится обобщенный оператор вида

$$D := B_1 + B_2 + B_3,$$

где B_i – операнды, подаваемые на входы сумматора и вход переноса C_0.

Рассмотрим класс k_1. Операторы, включенные в данный класс, предназначены для сброса регистра суммы и управления T_∂. На практике оказывается удобным преобразовать структурную схему таким образом, чтобы T_∂ управлялся сигналами операторов, а функция сброса *RG См* реализовывалась при отсут-

ствии других сигналов φ_s. При такой реализации схемы сигнал управления d_5 должен быть удален из табл. 18. Если же триггер дополнительный будет реализован в виде однобитного регистра, то сигнал d_5 сохраняется, а T_∂ рассматривается как элемент общего поля регистров.

Рассмотрим класс k_2, предварительно преобразуя оператор φ_7 к виду

$$\varphi_7 : \; D := 11...1 + A_2[12,0] + 0 .$$

После этого обобщенный оператор для класса k_2 может быть представлен суммой

$$D := B_1 + B_2 + B_3,$$

где $$B_1 = \begin{cases} 0, \text{ если } \varphi_2 = 1, \\ A_1[12,0], \text{ если } \varphi_3, \varphi_5 = 1, \\ [-1]_\partial = 11...1, \text{ если } \varphi_7 = 1; \end{cases} \qquad B_2 = \begin{cases} A_2[12,0], \text{ если } \varphi_2, \varphi_3, \varphi_7 = 1, \\ \neg A_2[12,0], \text{ если } \varphi_5 = 1; \end{cases}$$

$$B_3 = \begin{cases} 0, \text{ если } \varphi_2, \varphi_3, \varphi_7 = 1, \\ 1, \text{ если } \varphi_5 = 1. \end{cases}.$$

Для упрощения схемы сдвига в *АЛУ* можно объединить классы k_3 и k_4 в класс k_6, предварительно преобразовав оператор φ_8 к виду

$$k_6 = k_3 \bigcup k_4 = \begin{Bmatrix} \varphi_4 \\ \varphi_8 \end{Bmatrix} = \begin{Bmatrix} D := A_1[12,0] + L1(A_2[12,0].0) \\ D := L1(A_2[12,0].0) + L1(A_2[12,0].0) \end{Bmatrix}.$$

Обобщенный оператор для данного класса будет иметь вид

$$D := B_1 + B_2 + 0,$$

где $$B_1 = \begin{cases} L1(A_2[12,0].0), \text{ если } \varphi_8 = 1, \\ A_1[12,0].0), \text{ если } \varphi_4 = 1; \end{cases} \qquad B_2 = L1(A_2[12,0].0), \text{ если } \varphi_4, \; \varphi_8 = 1.$$

Операторы класса k_5 не изменяются, при этом

$$D := B_2 = R_2(00.A_2[12,0]) \text{ при } \varphi_9 = 1.$$

7-й этап синтеза – построение структурных схем для реализации обобщенных операторов.

Аппаратная реализация класса k_1 очевидна и может быть представлена обычным триггером *RS*-типа.

В классе k_2 операцию сложения следует реализовать с помощью схемы комбинационного сумматора с двумя входами для слагаемых B_1 и B_2 и входом переноса C_0 для оператора φ_5. Для формирования дополнительного кода на одном из входов модуля устанавливается блок инверсии (рис.45).

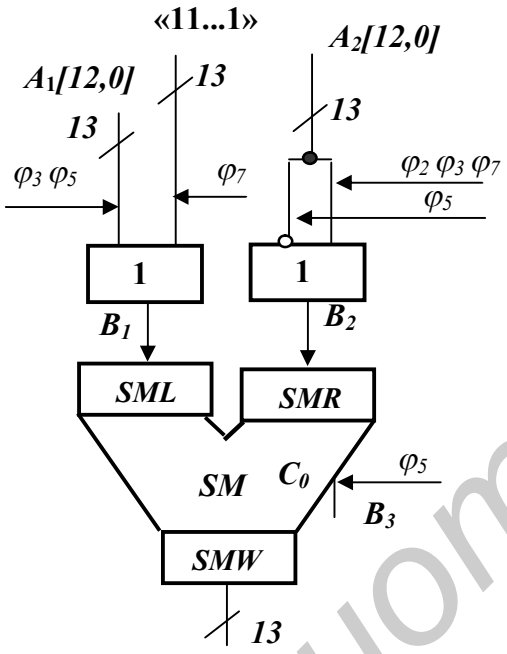

Рис. 45

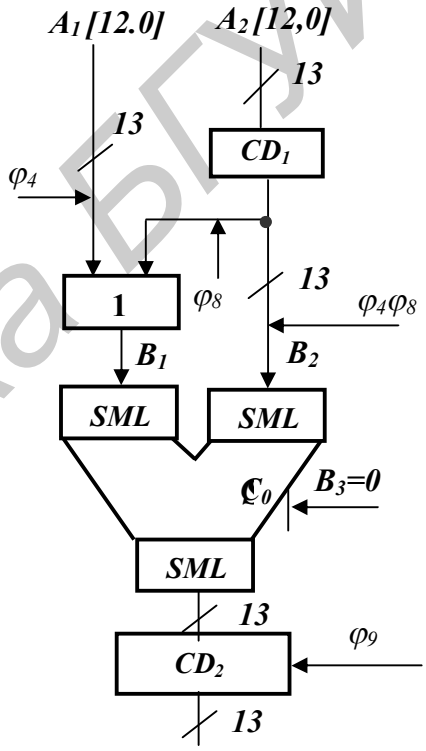

Рис. 46

Построим схему для операторов класса k_6. Она будет содержать модуль комбинационного сумматора и схему сдвигателя CD_1 (рис. 46). Здесь же для выполнения действий класса k_5 на выходе сумматора установим схему сдвигателя CD_2 на *2* разряда вправо за такт. Управление сдвигами в *CD2* осуществляется с помощью сигнала управления φ_9.

8-й этап синтеза – формирование списка логических условий.

На данном этапе составляется таблица, содержащая строку «перечень усло-

вий» и строку указателей на структурные компоненты *ОА* – источники этих условий (табл. 20). В процессе автоматизированного проектирования процессоров информация о признаках вводится в компьютер и используется для синтеза соответствующих комбинационных схем.

Таблица 20

Перечень условий	x_0	x_1	*N (signMн)*	…
Источник	*RG Mm*[*14*]	*RG Mm*[*13*]	*RG Mн*[*14*]	…

9-й этап синтеза – построение обобщенной схемы операционного устройства. На данном этапе объединяют (методом суперпозиции) все подсхемы, реализующие обобщенные операторы, в общую схему *АЛУ*. При необходимости элементы коммутации в подсхемах заменяются схемами мультиплексирования или устройствами с тремя состояниями на выходе (рис. 47).

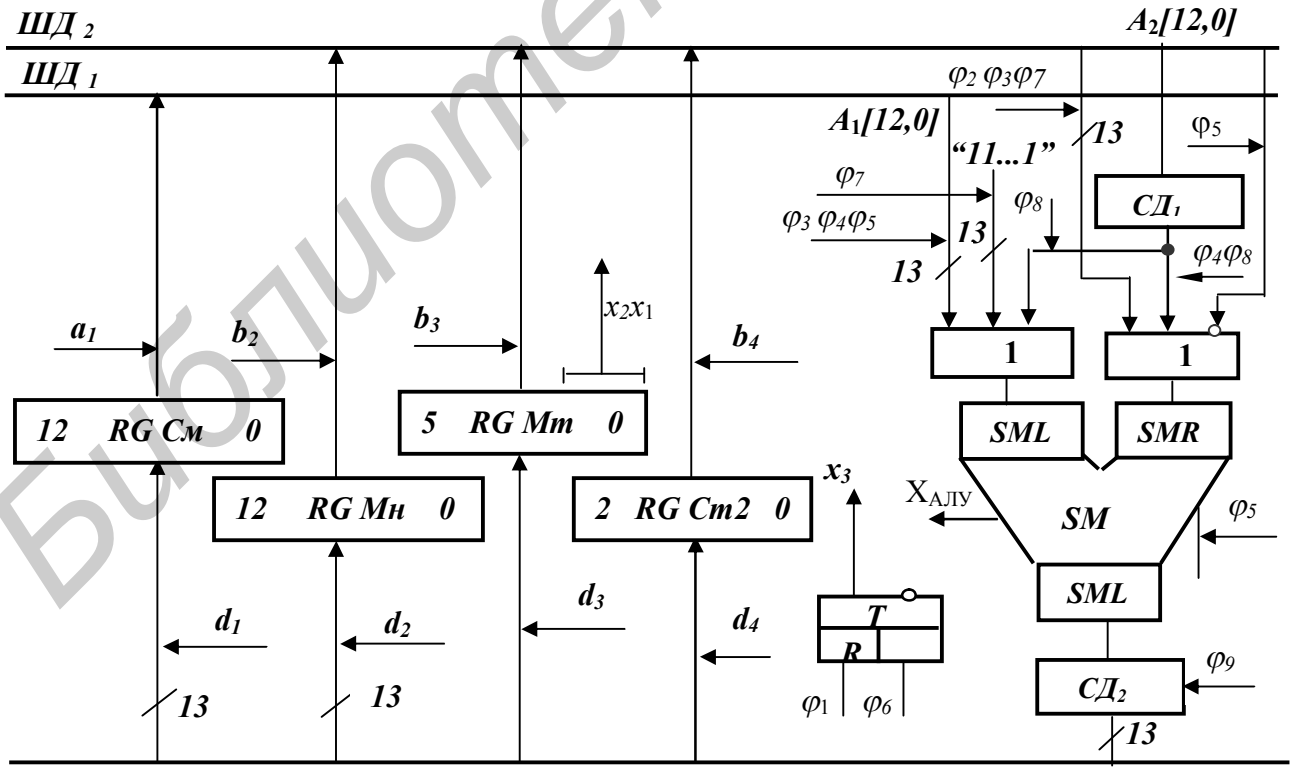

Рис. 47

M-автомат реализует в каждом такте одну микрокоманду граф-схемы алгоритма. Поэтому все совмещенные операторные вершины при синтезе устройства управления должны быть разделены с учетом потактовой работы операционного автомата.

2.9. Синтез *I*-автоматов

Структура операционного автомата *I*-типа условно может быть разделена на три составные части:

1) регистры памяти;
2) комбинационные схемы, реализующие функции АЛУ;
3) схемы, формирующие признаки результата.

Для получения максимальной производительности структура *I*-автомата не должна вносить ограничений на совместимость микрокоманд, т.е. структура должна обеспечить возможность одновременного выполнения всех функционально совместимых преобразований исходной граф-схемы алгоритма. Такой подход к вычислениям требует разделения комбинационной схемы *АЛУ* на подсхемы, выполняющие действия вида

$$RG_i := Ц_i(RG_1,...,RG_k), \quad i = 1,2,...,n.$$

Поставленному условию удовлетворяет класс устройств, у которых каждый из регистров RG_i обслуживается своей схемой *АЛУ*. Причем каждое $АЛУ_i$ должно выполнять только те операторы $Ф_i$, которые определяют содержимое регистра с номером *i*. Что касается максимального числа информационных шин *I*-автомата, то оно определяется возможностью распараллеливания вычислений, задаваемых исходным алгоритмом.

Каноническая структура автомата *I*-типа имеет вид, показанный на рис. 48.

На практике число используемых информационных шин у схем данного вида, как правило, невелико и составляет величину не более 2–3. В противном случае затраты на проектирование и построение блока *АЛУ* существенно возрастают.

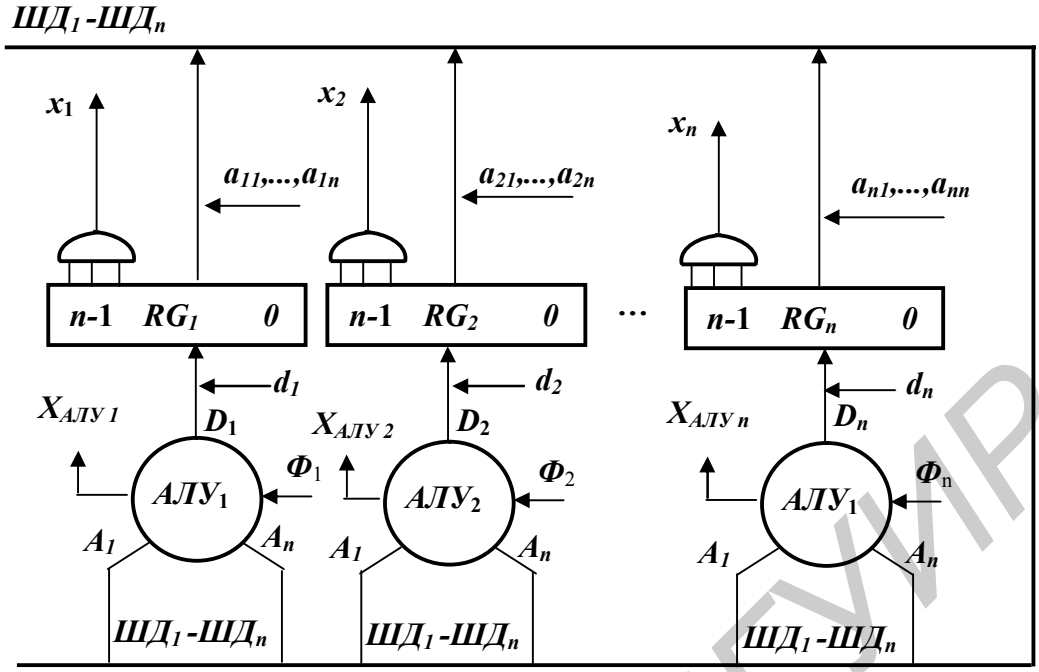

Рис. 48

Операционный автомат *I*-типа (укрупненно) синтезируется в соответствии со следующими этапами:

1-й этап. Множество микрокоманд, определяемых операторными вершинами *ГСА*, разбивается на подмножества $Y_1, \ldots, Y_n$, которые реализуют преобразование над словами:

$$RG_i := Ц_i(RG_k, \ldots, RG_j).$$

2-й этап. На подмножествах Y_i выделяются классы эквивалентных операторов для $АЛУ_i - K_{ij}$, причем в каждый класс j подмножества i включаются микрокоманды эквивалентного функционального назначения для регистра RG_i.

3-й этап. Для каждого класса j, содержащего не менее двух эквивалентных операторов, строится обобщенный оператор вида

$$D_{ij} := B_1 + B_2 + B_3.$$

4-й этап. Выполняется построение $АЛУ_i$ путем суперпозиции всех подсхем, соответствующих обобщенным операторам D_{ij}.

Рассмотрим теперь методику синтеза *I*-автомата на конкретном примере. Пусть требуется построить устройство, позволяющее выполнять умножение чисел на 2 разряда одновременно, начиная со старших разрядов *Мт*. Для решения задачи используем *ГСА* (рис. 40) со следующим списком микрокоманд:

$$y_1)\ RG\,Мт[5,0] := Мт, \qquad y_5)\ RG\ См[13,0] := RG\ См[13,0] + 2Мн,$$
$$y_2)\ RG\,Мн[13,0] := Мн, \qquad y_6)\ RG\ См[13,0] := RG\ См[13,0] + 4Мн,$$
$$y_3)\ См2[2,0] := 0{,}5n + 1, \qquad y_7)\ RG\ См[13,0] := RG\ См[13,0] - 2Мн,$$
$$y_4)\ RG\ См[13,0] := 0, \qquad y_8)\ RG\ См[13,0] := RG\ См[13,0] - 4Мн,$$

$$y_9)\ \ См2[2,0] := См2[2,0] - 1,$$
$$y_{10})\ RG\ См[13,0] := RG\ См[13,0] \times 2^2.$$
$$y_{11})\ RG\ Мт[5,0] := RG\ Мт[5,0] \times 2^2.$$

1-й этап синтеза I-автомата. Декомпозиция списка микрокоманд на 4 подмножества (в соответствии с числом регистров):

$$RG\ \ Мн: \ \ Y_1 = \{y_2\},$$
$$RG\ \ Мт: \ \ Y_2 = \{y_1, y_2\},$$
$$RG\ \ См: \ \ Y_3 = \{y_4, y_5, y_6, y_7, y_8, y_{10}\},$$
$$См2: \ \ Y_4 = \{y_3, y_9\}.$$

2-й этап. Распределение регистров по шинам.

В связи с малой трудоемкостью решаемой задачи выберем двухшинную организацию устройства и выполним распределение регистров по шинам в соответствии с табл. 21. В итоге множество регистров разделится на два подмножества:

$$ШД_1 = \{RG\ \ См\}, \quad ШД_2 = \{RGМн, RGМт, См2\}.$$

При выполнении данного этапа будем учитывать, что использование в схеме числа шин, большего, чем две, предполагает временное совмещение микрокоманд различных типов, например, при выполнении бинарных преобразований следует осуществлять пересылку данных с использованием линий $ШД_1$ и

ШД₂, а по шине *ШД*$_3$ (при необходимости) передавать декрементируемое значение счетчика (или сдвигаемый операнд).

Таблица 21

y_m	Содержание микрокоманды	*ШД₁*	*ШД₂*
y_1	*RG Мт[6,0]:= Мт*	–	–
y_2	*RG Мн[13,0]:= Мн*	–	–
y_3	*Ст2[2,0]:= 0,5n+1=4*	–	–
y_4	*RG См[13,0]:= 00…0*	–	–
y_5	*RG См[13,0]:= RG См[13,0]+ L1(RG Мн[13,0].0)*	*RG См[13,0]*	*RG Мн[13,0]*
y_6	*RG См[13,0]:= RG См[13,0]+ L2(RG Мн[13,0].00)*	*RG См[13,0]*	*RG Мн[13,0]*
y_7	*RG См[13,0]:= RG См[13,0]+ ┐L1(RG Мн[13,0].0)+1*	*RG См[13,0]*	*RG Мн[13,0]*
y_8	*RG См[13,0]:= RG См[13,0]+ ┐L2(RG Мн[13,0].00)+1*	*RG См[13,0]*	*RG Мн[13,0]*
y_9	*Ст2[2,0]:= Ст2[2.0] –1*	–	*Ст2[2,0]*
y_{10}	*RG См[13,0]:= L2(RG См[13,0].00)*	*RG См[13.0]*	–
y_{11}	*RG Мт[6,0]:= L2(RG Мт[6,0].00)*	–	*RG Мт[6,0]*

3-й этап. Определение форматов слов на *ШД*. Сущность этапа ничем не отличается от аналогичного шага синтеза *М*-автомата.

4-й этап. Анализ преобразований в *АЛУ*$_i$.

Таблица 22

y_m	Операторы АЛУ$_i$	Прием.	Обоз.
	$D_i:=\varphi_s(A_1,A_2)$	*RG*	φ_s
y_2	$D_1:= A_1[13,0]$	*RG Мн*	φ_2
y_1	$D_2:= A_1[6,0]$	*RG Мт*	φ_1
y_{11}	$D_2:= L2(A_1[6,0].00)$	*RG Мт*	φ_{11}
y_4	$D_3:= 00…0$	*RG См*	φ_4
y_5	$D_3:= A_1[13,0]+ A_2[13,0]$	*RG См*	φ_5
y_6	$D_3:= A_1[13,0]+L1(A_2[13,0].0)$	*RG См*	φ_6
y_7	$D_3:= A_1[12,0]+ \neg A_2[12,0]+1$	*RG См*	φ_7
y_8	$D_3:= A_1[12,0]+ \neg L1(A_2[13,0].0)+1$	*RG См*	φ_8
y_{10}	$D_3:= L1(A_1[13,0].0)$	*RG См*	φ_{10}
y_3	$D_4:= A_1[2,0]$	*Ст2*	φ_3
y_9	$D_4:= A_1[2,0]+111$	*Ст2*	φ_9

Табл. 22 состоит из подтаблиц, число которых равно числу регистров, а следовательно, и числу проектируемых *АЛУ*. При составлении данных под-

множеств учитывался также тот факт, что процесс выполнения сдвигов на два разряда *L2(A2[13,0].00)* в *RG См* реализуется с использованием технологической, «косой» передачи данных в *RG См* плюс аппаратный сдвиг также на один разряд. Выполнение сдвига в *RG Мт – L2(A1[6,0].00)* предполагается осуществлять только за счет технологической коммутации, сигнал же управления параллельной ветвью при этом должен быть равен нулю. На этом же этапе составляются таблицы управления источниками операндов и выбора приемников результатов для слов, пересылаемых с выходов *АЛУ* (см. табл. 23 и 24).

Для построения таблиц присвоим регистрам *ОА* следующие номера:

$$RG\ Мн:=RG_1;\quad RG\ Мт:=RG_2;\quad RG\ См:=RG_3;\quad См2:=RG_4.$$

Таблица 23

Источник операндов		*Сигнал* управления
$ШД_1$	$ШД_2$	a_{ij}
RGСм	–	a_{31}
–	*RGМн*	a_{12}
–	*RGМт*	a_{22}
–	*См2*	a_{42}

Таблица 24

Приемник результата	*Сигнал* управления
$D_i \rightarrow RG_k$	d_k
$D_3 \rightarrow RG\ См$	d_3
$D_1 \rightarrow RG\ Мн$	d_1
$D_2 \rightarrow RG\ Мт$	d_2
$D_4 \rightarrow См2$	d_4

5-й этап. Кодирование микрокоманд наборами управляющих сигналов (см. методику синтеза *М*-автоматов и табл. 25)

Таблица 25

y_m	a_{i1}	a_{i2}	d_k	φ_s
y_1	–	–	d_2	φ_1
y_2	–	–	d_1	φ_2
y_3	–	–	d_4	φ_3
y_4	–	–	d_3	φ_4
y_5	a_{31}	a_{12}	d_3	φ_5
y_6	a_{31}	a_{12}	d_3	φ_6
y_7	a_{31}	a_{12}	d_3	φ_7
y_8	a_{31}	a_{12}	d_3	φ_8
y_9	–	a_{42}	d_4	φ_9
y_{10}	a_{31}	–	d_3	φ_{10}
y_{11}	–	a_{22}	d_2	φ_{11}

Этап построения эквивалентных операторов в данном примере может быть пропущен ввиду малой сложности.

6-й этап. Составление списка обобщенных операторов для схем $АЛУ_i$.

$$D_1 := A_1[13,0], \quad если\ \varphi_2 = 1; \qquad D_2 := \begin{cases} A_1[6,0], & если\ \varphi_1 = 1; \\ L1(A_1[6,0].0), & если\ \varphi_{11} = 1; \end{cases}$$

$$D_3 := B_1 + B_2 + B_3,$$

$$B_1 = \begin{cases} A_1[13,0], & если\ \varphi_5,\ \varphi_6, \\ \varphi_7,\ \varphi_8 = 1, & \\ L1(A_1[13,0].0), & если\ \varphi_{10} = 1; \end{cases} \quad B_2 = \begin{cases} A_2[13,0], & если\ \varphi_5 = 1, \\ L1(A_2[13,0].0), & если\ \varphi_6 = 1, \\ \neg A_2[13,0], & если\ \varphi_7 = 1, \\ \neg L1(A_2[13,0].0), & если\ \varphi_8 = 1; \end{cases} \quad B_3 = \begin{cases} 0, & если\ \varphi_5, \varphi_6, \varphi_{10} = 1, \\ 1, & если\ \varphi_7, \varphi_8 = 1. \end{cases}$$

$$D_4 := B_1 + B_2,$$

$$B_1 = A_1[2.0], \quad если \quad \varphi_3, \varphi_9 = 1; \qquad B_2 = 111, \quad если \quad \varphi_9 = 1.$$

7-й этап. Формирование списка логических условий.

На данном этапе составляется таблица, содержащая строку «перечень условий» и строку указателей на структурные компоненты ОА – источники этих условий. Сущность этапа ничем не отличается от методики, рассмотренной выше при синтезе *М*-автомата.

8-й этап. Построение схем $АЛУ_i$ на основе обобщенных операторов.

На рис. 49 приведена реализация $АЛУ_1$ и $АЛУ_2$. В первой схеме вычислитель практически отсутствует и состоит только из схемы коммутации *Мн* в регистр. Сигнал управления записью в *RG Мн* при этом может быть интерпретирован как микрооперация умножения на единицу и заменен на оператор φ_2.

Второе *АЛУ* состоит из сдвигателя, предназначенного для сдвига множителя на два разряда за счет технологической «косой» передачи. Особенностью $АЛУ_1$ и $АЛУ_2$ является то, что обе схемы имеют один и тот же источник данных. В связи с этим, невзирая на подключение одного и другого *АЛУ* к $ШД_2$, каждый из входов *АЛУ* имеет первый порядковый номер.

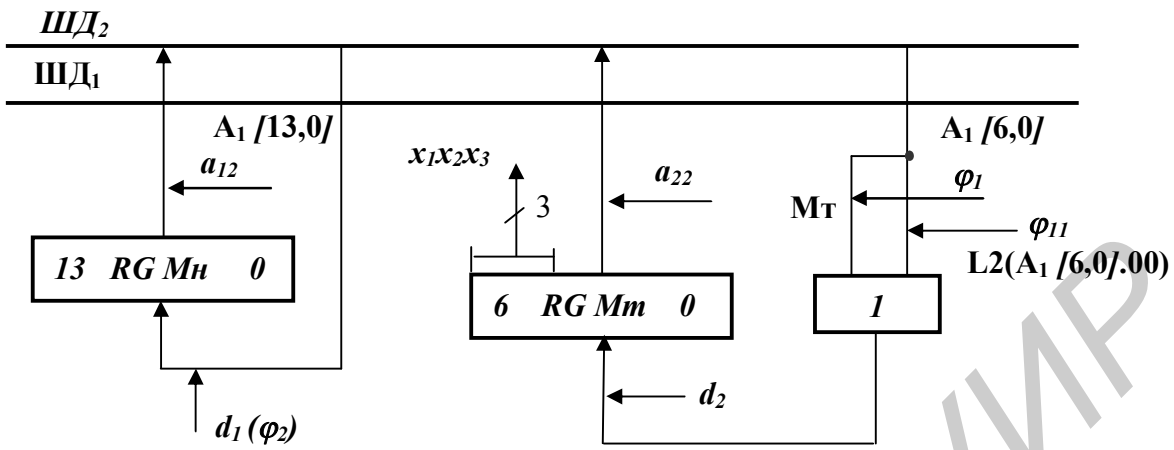

Рис. 49

Схема, приведенная на рис. 50, реализует схемотехнику *АЛУ3*. Здесь микрооперация сдвига регистра *RG См* также осуществляется за счет «косой» (технологической) передачи операнда на вход логического элемента *2ИЛИ*. Сдвиг множимого на один разряд выполняется аналогично, а в случае необходимости – аппаратно (дополнительно) еще на один разряд влево.

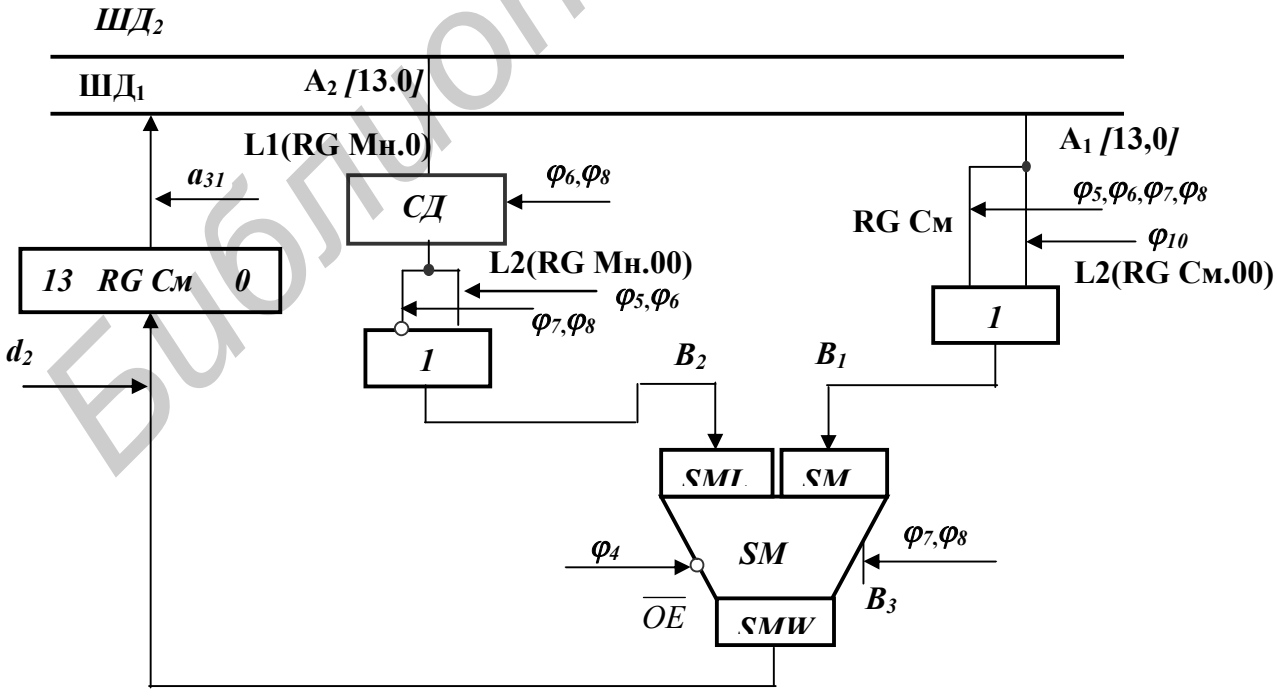

Рис. 50

Рис. 51 поясняет принцип функционирования *АЛУ*$_4$. В данной схеме декрементирование содержимого регистра *Cm2* осуществляется путем сложения текущего наполнения счетчика и дополнительного кода числа «–1».

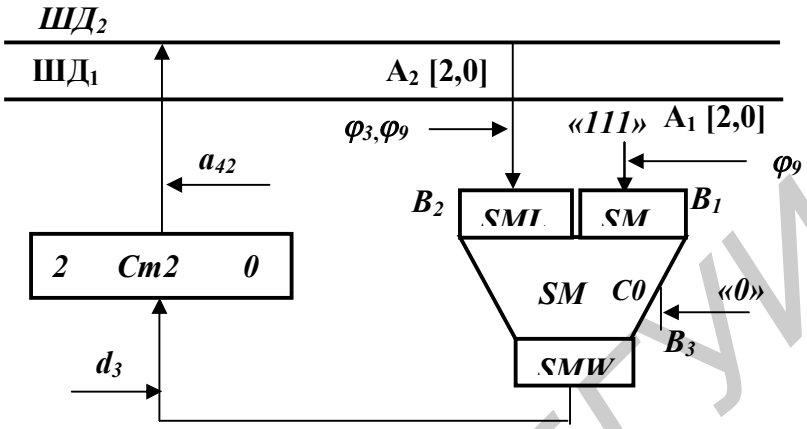

Рис. 51

2.10. Структурная организация и синтез *IM*-автоматов

Рассмотренные выше два класса *ОА* обладают диаметрально противоположными свойствами: *I*-автоматам присуща максимальная производительность при наибольшей сложности схемы, *M*-автоматам – минимальная производительность при наименьших затратах аппаратуры. Следует ожидать, что между этими двумя классами структур будут существовать варианты *ОА,* обладающие промежуточными свойствами. Эти структуры при проектировании выделяют в особый класс – класс так называемых *IM*-автоматов.

IM-автоматы – это операционные устройства, структурная организация которых вносит определенные ограничения на совместимость микрокоманд, однако позволяет выполнить за такт более одного преобразования функциональной микропрограммы.

Структура *IM*-автомата (укрупненно) порождается следующим образом.

Во-первых, аналогично, как и для *M*-автомата, определяются все операторы, реализуемые в комбинационных схемах *АЛУ*.

Во-вторых, определяются классы операторов, эквивалентных в смысле выполняемых ими функций.

В-третьих, в состав *ОА* включаются комбинационные схемы *АЛУ*, реализующие определенные совокупности эквивалентных или единый, обобщенный оператор.

Таким образом, *IM*-автоматы можно рассматривать как композицию из *k* *M*-автоматов, взаимодействующих с общим полем регистров. Исходя из этого, синтез автоматов среднего класса сводится к разбиению всего множества микрокоманд на *k* подмножеств и синтезу *k* *M*-автоматов, реализующих функции данных подмножеств.

Проведенный анализ операций в ОА показывает, что большинство вычислительных алгоритмов в компьютерной технике базируется на отработке следующих трех групп основных операций:

1) сложение в дополнительном или обратном коде;
2) сдвиг операнда на *i* разрядов;
3) логическое преобразование.

В связи с этим при построении *IM*-автоматов структура устройства, как правило, будет содержать комбинационные схемы $АЛУ_1$ и $АЛУ_2$, реализующие функции преобразования соответственно двух и одного операнда:

$$D_1 := \Phi_1\left(A_1, A_2\right), \quad D_2 := \Phi_2\left(A_3\right).$$

Обобщенная структурная схема рассматриваемого автомата представлена на рис. 52. В данной схеме a_i, b_j, c_k – это микрооперации выбора операндов; $Ф_1$, $Ф_2$ – подмножества операторов, выполняемых в схемах $АЛУ_1$, $АЛУ_2$; d_q, e_u – сигналы управления приемниками результатов, сформированных на выходах $АЛУ_1$ или $АЛУ_2$. Очевидно, что максимальная производительность *IM*-автомата определяется числом комбинационных *АЛУ* в проектируемой структуре. Как правило, данный параметр не превышает двух микрокоманд за такт машинного

времени. Этапы синтеза автомата *IM*-типа во многом повторяют этапы синтеза рассмотренных выше *M-, I*-структур.

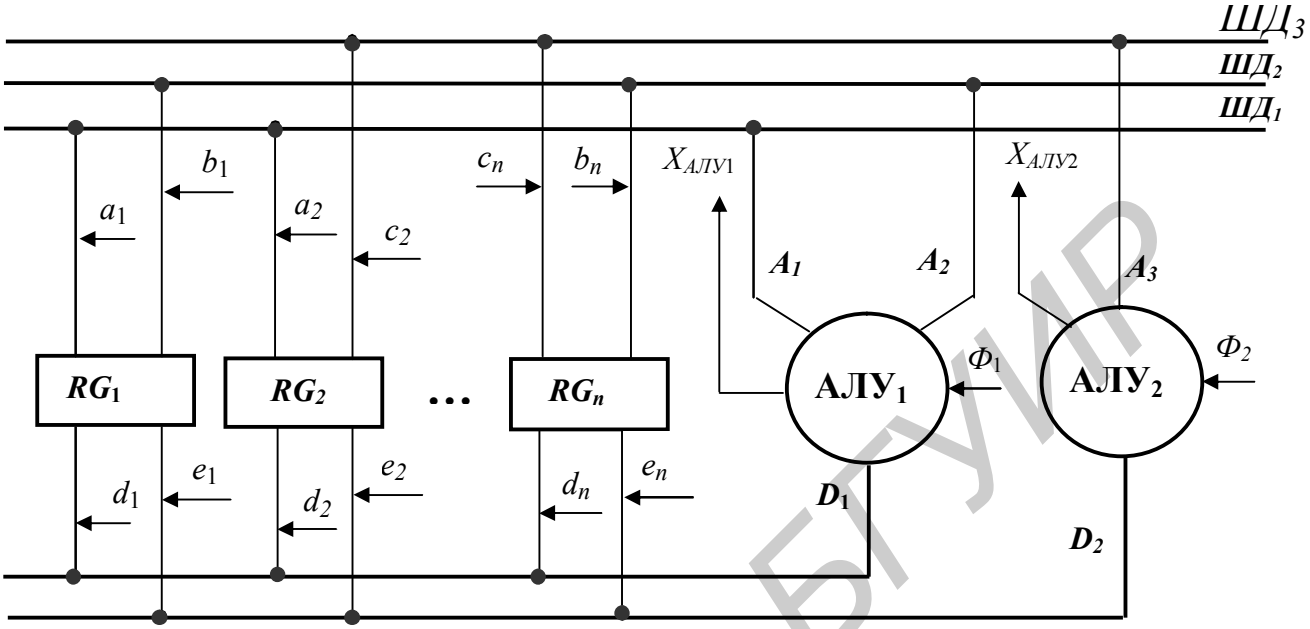

Рис. 52

Первый этап заключается в составлении списка микрокоманд, выполняемых в *ОА*. На данном этапе строится *ГСА,* и все операторные вершины алгоритма отмечаются сигналами y_i. После этого формируется список, в который включаются преобразования, указанные в операторных вершинах.

На втором этапе в зависимости от типа выполняемой микрокоманды и модифицируемого числа операндов множество микрокоманд Y разбивается на два подмножества. В первое Y_1 включаются бинарные преобразования, а регистры-источники операндов данного подмножества закрепляются за шинами $ШД_1$ и $ШД_2$. Во второе подмножество Y_2 включаются унарные преобразования, а регистры этого подмножества закрепляются за $ШД_3$ (табл. 26, 27).

Таблица 26

	y_m	*Микрокоманда*	$ШД_1$	$ШД_2$
Y_1	y_5	*RG См* [*14,0*]:= *RG См* [*14,0*]+ *RG Мн* [*14,0*]	*RG См*	*RG Мн*
		. . . . .		

Таблица 27

	y_m	*Микрокоманда*	$ШД_3$
Y_2	y_{10}	*RG См* [*14,0*]*:= L*1*(RG См* [*14,0*]*.0)*	*RG См*

В соответствии с принятой концепцией проектирования, т.е. с учетом разбиения множества Y на два подмножества, в исходную ГСА могут быть внесены коррективы, направленные на совмещение выполняемых действий.

На третьем этапе определяется разрядность слов, передаваемых на входы A_1 и A_2 $АЛУ_1$ и вход A_3 $АЛУ_2$. При этом принимается соглашение о совмещении регистров по младшим разрядам и выборе разрядности шин *ОА* равной разрядности регистра максимальной длины.

На четвертом этапе составляется список операторов, реализуемых схемами $АЛУ_1$ и $АЛУ_2$, а также составляются таблицы для выбора источников и приемников результата:

Таблица 28

	Микро-команда	Преобразование слов в $АЛУ_1$	
	Обознач.		
	команда		оператор
Y_1	y_4	$D_1 := A_1[14,0] + A_2[14,0]$	φ_8
		. . . . .	

Таблица 29

	Микро-команда	Преобразование слов в $АЛУ_2$	
	Обознач.		
	команда		оператор
Y_2	y_{12}	$D_2 := L2(A_3[14,0].00)$	φ_{19}
		. . . . .	

Таблицы для выбора источников операндов и управления приемниками результата составляются с учетом номеров регистров, предварительно закрепленных за структурными элементами схемы. Пусть *RG Сч* имеет обозначение RG_1, *RG Мт* – RG_2, *RG Мн* – RG_3, *RG См* – RG_4, тогда таблицы управления операндами могут быть представлены в виде:

Таблица 30

Источник операндов			Сигнал управления		
$ШД_1$	$ШД_2$	$ШД_3$	a_i	b_j	c_k
RGСм	*RGMн*	–	a_4	b_3	–
			.....		

Таблица 31

Приемник результата	Сигнал управлен.	
RG_n	d_q	e_u
RG_4	d_4	-
...		

На пятом этапе составляется таблица кодирования микрокоманд сигналами управления:

$$y_m = f(a_i, b_j, c_k, d_q, e_u, \varphi_s).$$

Таблица 32

Микро-операция	Сигнал управления					
y_m	a_i	b_j	c_k	d_q	e_u	φ_s
...						

На шестом этапе определяются классы эквивалентных операторов, что позволяет в ряде случаев оптимизировать схемы коммутации операндов.

На седьмом этапе для подмножеств эквивалентных операторов строятся обобщенные операторы, содержащие слагаемые B_1–B_3. Здесь же формируются подсхемы для реализации операторов D_{ij}:

$$D_{i1} := B_1+B_2+B_3; \quad D_{i2} := B_1.$$

Управление операндами *B* осуществляется с помощью сигналов φ_s – операторов *АЛУ*.

Восьмой этап состоит в формировании списка логических условий. При этом составляется таблица, содержащая строку «перечень условий» и строку указателей на структурные компоненты ОА – источники этих условий.

Например

Таблица 33

Перечень условий	x_0	x_1	*N (signMn)*	...
Источник RG Mm[n]	*RG Mm[14]*	*RG Mm[13]*	*RG Mn[14]*	...

На девятом этапе строится структурная схема *IM*-автомата путем суперпозиции подсхем, соответствующих обобщенным операторам $АЛУ_1$ и $АЛУ_2$.

Десятый этап синтеза (при необходимости) будет состоять в коррекции ГСА с учетом всех изменений, полученных в ходе проектирования вычислительного устройства. В целом же на каждом из этапов возможен возврат на предыдущие этапы для внесения коррективов, обусловленных общим ходом синтеза схемы.

2.11. Использование *ЗУ* в структуре операционного устройства и класс *S*-автоматов

В некоторых операционных устройствах преобразования выполняются над большим числом внутренних слов. Примером такой системы является процессор ввода-вывода, используемый как периферийный по отношению к центральному. Для снижения стоимости устройств данного вида регистровая память *ОА* заменяется оперативным *ЗУ*, которое обеспечивает в текущий момент времени доступ к любой из ячеек. Операционные автоматы данного вида получили название *S*-автоматов. Типичная структура *S*-автомата показана на рис. 53.

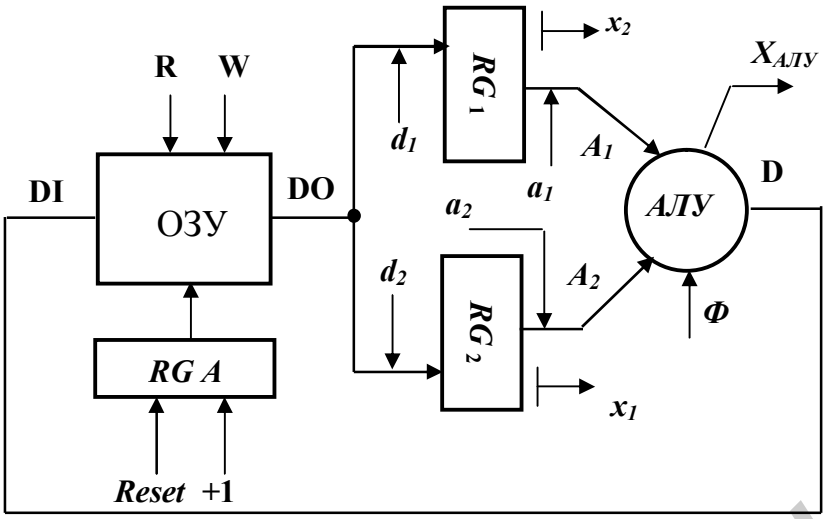

Рис. 53

ОЗУ операционного устройства обеспечивает хранение $N=2^m$ n-разрядных слов по адресам *0, 1, 2, ..., 2^m–1*. Обращение к ОЗУ инициируется сигналами чтения: $DO := ОЗУ[A]$ и записи: $ОЗУ[A] := D[АЛУ]$, где A – адрес памяти. Регистры RG_1 и RG_2 используются для хранения двух операндов, участвующих в исполняемых микрокомандах.

Цикл функционирования автомата разделяется на последовательность четырех действий:

1) чтение из *ОЗУ* первого операнда: $RG_1[n-1,0] := ОЗУ[A], \quad MEM \;\; R/\overline{W} = 1$;

2) чтение из *ОЗУ* второго операнда: $RG_2[n-1,0] := ОЗУ[A], \quad MEM \;\; R/\overline{W} = 1$;

3) выполнение преобразования в *АЛУ*: $D[АЛУ] := \varphi_m\left(RG_1, RG_2\right)$;

4) запись результата в память: $ОЗУ[A] := D[АЛУ]$.

Комбинационная схема *АЛУ* *S*-автомата строится так же, как и у *M*-автомата, а быстродействие определяется длительностью цикла обращения к памяти и временем задержки в арифметико-логическом устройстве.

С целью повышения производительности на некоторых операциях в структуру *S*-автомата иногда включают дополнительные регистры и связи. При этом результаты промежуточных вычислений могут размещаться во вновь введенных регистрах взамен существенно более медленнодействующего *ОЗУ*.

Пример. Пусть требуется построить *ОА*, предназначенный для изменения порядка следования элементов некоторого массива на противоположный.

Для решения поставленной задачи построим граф-схему вычислений вида рис. 54.

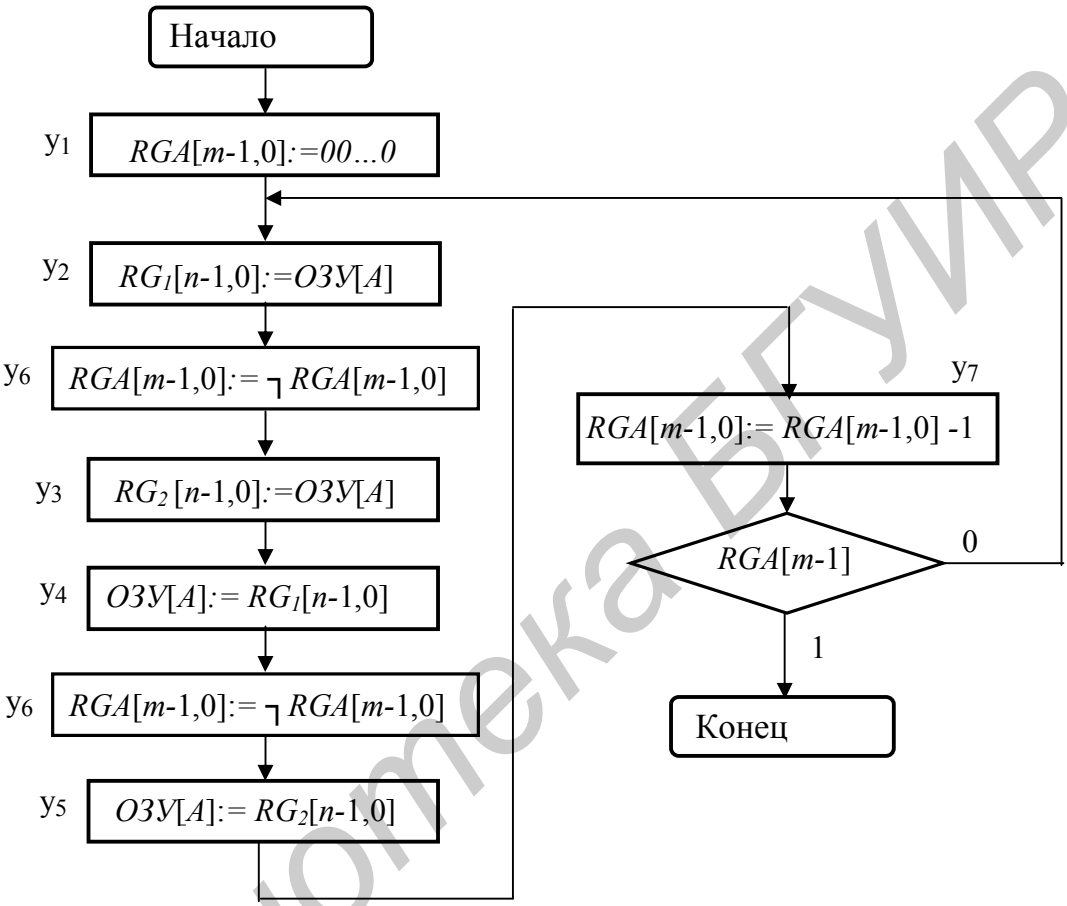

Рис. 54

По данной *ГСА* выполним синтез S-автомата в соответствии с алгоритмом проектирования *автомата М*-типа. При этом, учитывая малую сложность процесса проектирования, построим унифицированную табл. 34, присвоив регистру адреса третий номер: $RGA[m\text{-}1,0] = RG_3$.

Схема автомата, формируемая в соответствии с таблицей синтеза, может быть представлена в виде рис. 55.

В задачах большей сложности при синтезе *ОА* необходимо использовать (или повторить) все принципиальные моменты этапов проектирования, соответствующие ветствующие формированию структур *М*-типа.

Таблица 34

y_m	Микрокоманда	A_1	A_2	$D:=\varphi(A_1A_2)$	$D \rightarrow RG_k$	φ_m, a_i, b_j
y_1	$RGA[m\text{-}1,0]:=00\ldots0$	–	–	$D:=00\ldots0$	RGA	d_3
y_2	$RG_1[n\text{-}1,0]:=ОЗУ[A]$	–	–	–	–	R, d_1
y_3	$RG_2[n\text{-}1,0]:=ОЗУ[A]$	–	–	–	–	R, d_2
y_4	$ОЗУ[A]:=RG_1[n\text{-}1,0]$	RG_1	–	$D:=A_1[n\text{-}1,0]$	$ОЗУ[A]$	a_1, φ_4, W
y_5	$ОЗУ[A]:=RG_2[n\text{-}1,0]$	–	RG_2	$D:=A_2[n\text{-}1,0]$	$ОЗУ[A]$	a_2, φ_5, W
y_6	$RGA[m\text{-}1,0]:=\neg RGA[m\text{-}1,0]$	RGA	–	$D:=\neg A_1[n\text{-}1,0]$	RGA	a_3, φ_6, d_3
y_7	$RGA[m\text{-}1,0]:=RGA[m\text{-}1,0]+1$	RGA	–	$D:=A_1[n\text{-}1,0]+1$	RGA	a_3, φ_7, d_3

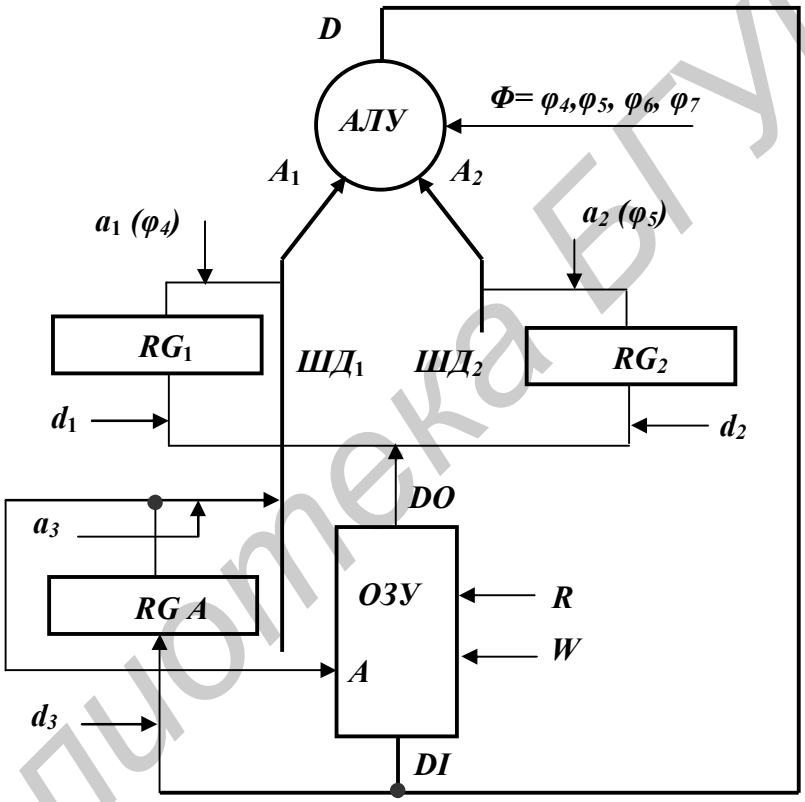

Рис. 55

Рассмотренные методики синтеза операционных автоматов в чистом виде, как правило, не применяются. Основные идеи, изложенные в данном методическом пособии, используются для автоматизированного проектирования микропроцессоров, микроконтроллеров и других БИС, ориентированных на сложные вычисления по разветвленным алгоритмам.

ЛИТЕРАТУРА

1. Майоров С.А., Новиков Г.И. Структура электронных вычислительных машин. – Л.: Машиностроение, 1979. –384 с.

2. Проектирование цифровых систем на комплектах микропрограммируемых БИС // С.С. Булгаков и др.; Под ред. В.Г. Колесникова. – М.: Радио и связь, 1984. –265 с.

Св. план 2003, поз. 41

Учебное издание

Кобяк Игорь Петрович

ПРОЦЕССОРЫ КОМПЬЮТЕРНЫХ СИСТЕМ. СИНТЕЗ ОПЕРАЦИОННЫХ АВТОМАТОВ

Методическое пособие
по курсовому и дипломному проектированию
по ТиП ЭВМ и СиФО ЭВМ для студентов специальности
40 02 01 «Вычислительные машины, системы и сети»
дневной формы обучения

Редактор Н.А. Бебель
Корректор Е.Н. Батурчик

Подписано в печать 26.02.2003. Формат 60x84 1/16. Бумага офсетная.
Печать ризографическая. Гарнитура «Таймс». Усл.печ. л. 5,0.
Уч.-изд. л. 4,1. Тираж 75 экз. Заказ 692.

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Лицензия ЛП № 156 от 30.12.2002.
Лицензия ЛП № 509 от 03.08.2001.
220013, Минск, П. Бровки, 6.