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

Кафедра программного обеспечения информационных технологий

ОРГАНИЗАЦИЯ И ФУНКЦИОНИРОВАНИЕ ЭВМ

Методическое пособие для студентов специальности 1-40 01 01
«Программное обеспечение информационных технологий»
дневной формы обучения

В 3-х частях

Часть 3

А. Т. Пешков, А. С. Кобайло

СХЕМОТЕХНИЧЕСКИЕ ОСНОВЫ ЭВМ

Минск БГУИР 2009

УДК 004.27+004.3′144:621.3.049.75(076)
ББК 32.973.26-02я7
О-64

Р е ц е н з е н т – доцент кафедры информатики,
кандидат физико-математических наук С. И. Сиротко

О-64 **Организация** и функционирование ЭВМ : метод. пособие для студ. спец. 1-40 01 01 «Программное обеспечение информационных технологий» днев. формы обуч. В 3 ч. Ч 3 : Схемотехнические основы ЭВМ / А. Т. Пешков, А. С. Кобайло. – Минск : БГУИР, 2009. – 70 с. : ил.
ISBN 978-985-488-380-9 (ч. 3)

Предназначено для студентов специальности «Программное обеспечение информационных технологий». Излагается материал, связанный с схемотехническими вопросами построения ЭВМ. Пособие включает большое количество иллюстраций и таблиц.

Части 1-я и 2-я изданы в БГУИР 2004 и 2005 гг.

УДК 004.27+004.3′144:621.3.049.75(076)
ББК 32.973.26-02я7

ISBN 978-985-488-380-9 (ч. 3)
ISBN 985-444-601-8
ISBN 978-985-444-601-1

Содержание

1. УЗЛЫ ЭВМ

Узлы ЭВМ представляют собой совокупность нескольких логических схем и элементов памяти, формирующих выходные сигналы, соответствующие нескольким логическим функциям от входных сигналов.

Узлы ЭВМ можно подразделить на два типа:

– комбинационные;

– накапливающие.

Характерной особенностью узлов *комбинационного* типа является то, что их выходные сигналы определяются только действующими в данный момент входными сигналами (не зависят от «истории» входных сигналов).

Характерной особенностью узлов *накапливающего* типа является то, что их выходные сигналы определяются не только действующими в данный момент входными сигналами, но и тем, какие входные сигналы поступали на узел ранее, т. е. выходные сигналы зависят от «истории» входных сигналов. В накапливающих узлах свойство хранить историю обеспечивается памятью, представленной некоторой совокупностью запоминающих элементов.

1.1. Комбинационные узлы

К числу узлов комбинационного типа относятся следующие.

Дешифратор

На рис. 1.1, а приведена схема формирования выходных сигналов дешифратора для значений входных данных (4-разрядных кодов, соответствующих десятичным цифрам от 1 до 9). Каждый вход обозначен набором входных переменных и соответствующей ему десятичной цифрой.

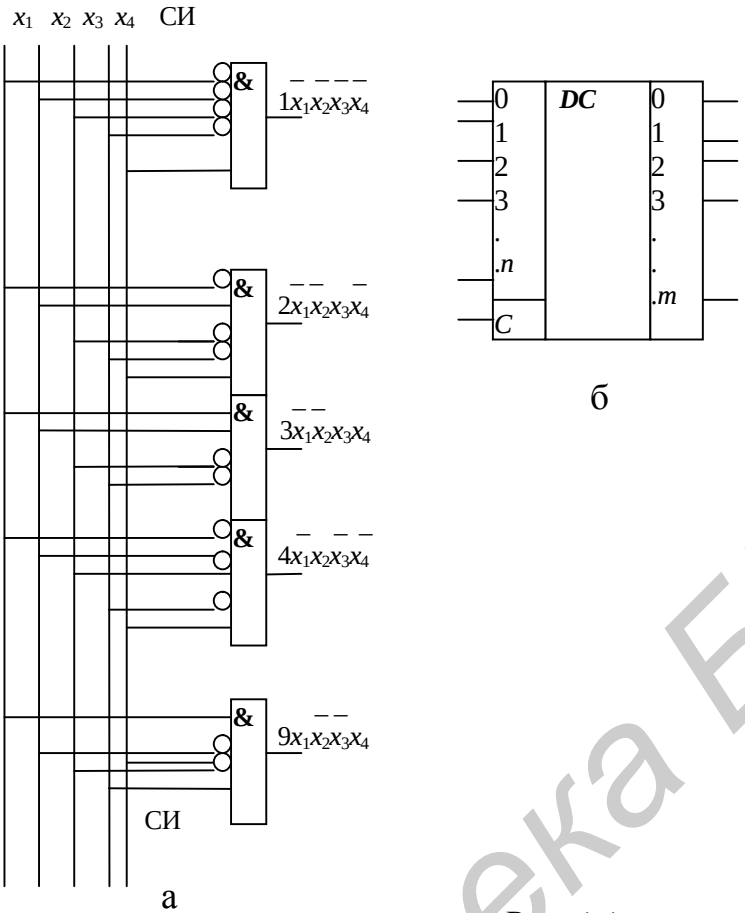
На вход дешифратора поступает n-разрядный код. В зависимости от его значения формируется сигнал на одном из m выходов. Значения n и m связаны соотношением $2^n \supseteq m$.

На рис. 1.1, б дано условное обозначение дешифратора. Вход C является входом синхронизации.

Шифратор

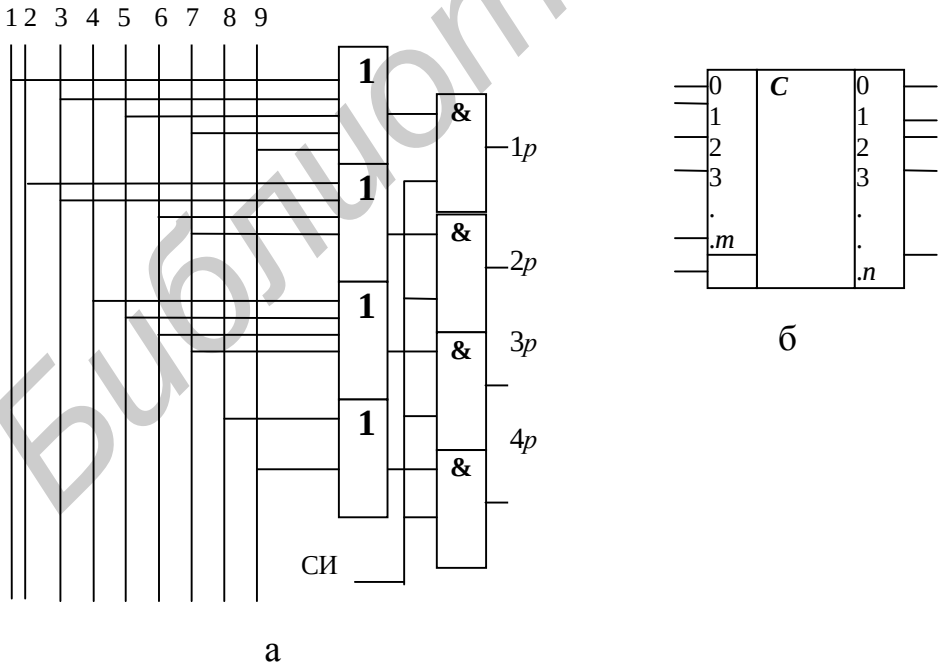
На рис. 1.2, а показана реализация шифратора (кодера), а на рис. 1.2, б – его условное обозначение. На вход шифратора поступает один из n сигналов. Вход C является входом синхронизации. Значения n и m связаны соотношением $2^m \supseteq n$.

Шифратор (см. рис. 1.1, а) формирует на четырех выходах код для одного из десяти входов, на котором в данное время имеет место единичный сигнал. Формируемый код соответствует в двоично-десятичной кодировке номеру входа с единичным сигналом. На рис. 1.2, а выходы шифратора обозначены номерами двоичных разрядов тетрады, отображающих 4-разрядный двоичный эквивалент десятичной цифры.



а б

Рис. 1.1



а б

Рис. 1.2

Сумматор по модулю 2

Сумматор по модулю 2 вырабатывает на своем входе сигнал логической единицы, если количество его входов с сигналом логической единицы нечетное.

На рис. 1.3, а приведена схема сумматора по модулю 2 на два входа, а на рис. 1.3, б – его условное обозначение. На рис. 1.3, в приведен сумматор по модулю 2 с восьмью входами, построенный по принципу каскадирования из двух входовых сумматоров по модулю 2.

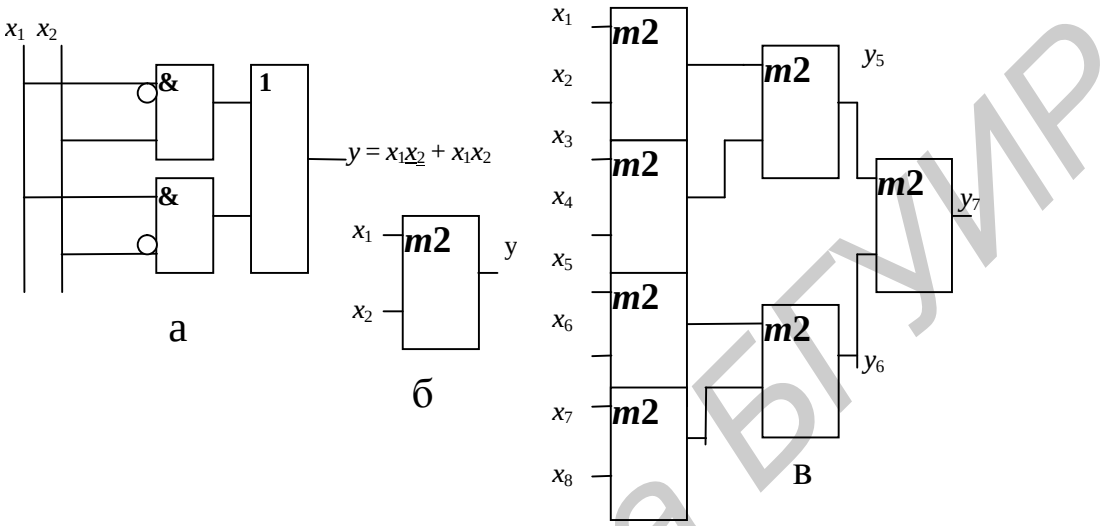


Рис. 1.3

Сигнал на выходе схемы (см. рис. 1.3, в) y_7 определяется логическим выражением

$$y = (y_5\bar{y}_6 + \bar{y}_5 y_6) = (y_1\bar{y}_2 + \bar{y}_1 y_2)\overline{(y_3\bar{y}_4 + \bar{y}_3 y_4)} + \overline{(y_1\bar{y}_2 + \bar{y}_1 y_2)}(y_3\bar{y}_4 + \bar{y}_3 y_4).$$

Мультиплексор

Мультиплексор реализует функцию подключения одного из нескольких входов к единственному выходу. На рис. 1.4, а приведена схема мультиплексора, а на рис. 1.4, б – его условное обозначение.

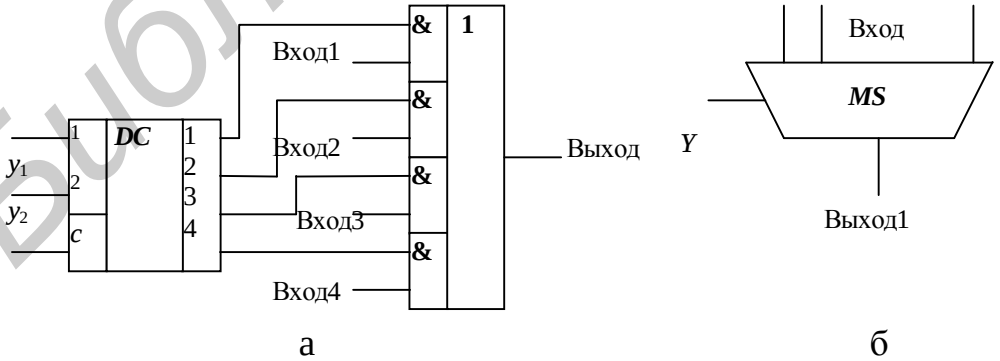


Рис. 1.4

Приведенная схема осуществляет коммутацию четырех *одноразрядных* входов на один *одноразрядный* выход. Входы y_1, y_2 определяют номер одного из четырех входов, который нужно логически соединить с выходом.

На рис. 1.5 приведена схема мультиплексора, обеспечивающая коммутацию четырех *n*-разрядных входов на один *n*-разрядный выход.

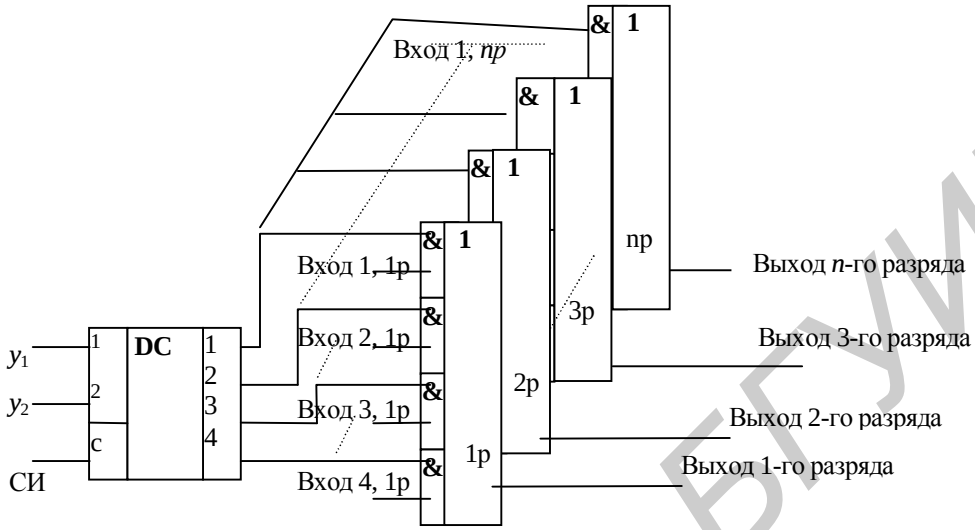


Рис. 1.5

Демультиплексор

Демультиплексор выполняет функцию логического подключения одного входного канала к одному из нескольких выходных каналов, т. е. его функция является обратной по сравнению с функцией, реализуемой мультиплексором. На рис. 1.6, а приведена схема демультиплексора, реализованного на дешифраторе.

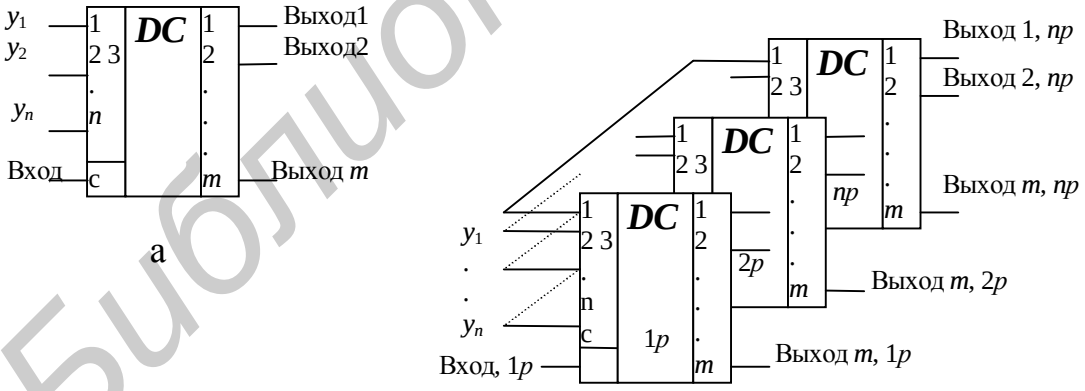


Рис. 1.6

На входы y_1, …, y_n подается код номера входа, к которому нужно логически подсоединить вход демультиплексора, которым является вход синхронизации дешифратора. Демультиплексор, приведенный на схеме, осуществляет коммутацию одного одноразрядного входа на один из четырех одноразрядных выходов.

На рис. 1.6, б приведена схема демультиплексора, обеспечивающего коммутацию n-разрядного входа на один из m n-разрядных выходов. Схема включает m дешифраторов (по числу разрядности входного и выходных каналов). Каждый дешифратор имеет по m выходов (по количеству выходов демультиплексора). Разрядами коммутируемого входа являются входы синхронизации соответствующих дешифраторов. Одноименные информационные входы дешифраторов объединены, на них подаются соответствующие разряды кода, определяющего номер выходного канала.

Сумматор

Одноразрядный *двоичный сумматор* обеспечивает сложение одноименных разрядов операндов с учетом переноса, поступающего от ближайшего младшего разряда. Сумматор вырабатывает значение соответствующего разряда суммы (S) и перенос (P), который должен быть учтен в соседнем старшем разряде. Синтез схемы, реализующей функции одноразрядного сумматора, можно выполнить на основании таблицы истинности (табл. 1.1).

Таблица 1.1

N	a	b	p	P	S
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

Исходя из реализуемой функции сумматор представляет собой логический узел с двумя выходами (выход суммы S и выход переноса P) и тремя входами: a – разряд первого операнда; b – разряд второго операнда; p – перенос из младшего разряда. На основании таблицы истинности можно записать логические выражения для формируемых суммы и переноса, которые будут иметь следующий вид:

$$P = \bar{a}bp + a\bar{b}p + ab\bar{p} + abp;$$

$$S = \bar{a}bp + a\bar{b}p + ab\bar{p} + abp.$$

Полученные функции удобно минимизировать с помощью карты Карно, так как количество переменных невелико. Карты Карно с представленными в них функциями S и P приведены на рис. 1.7.

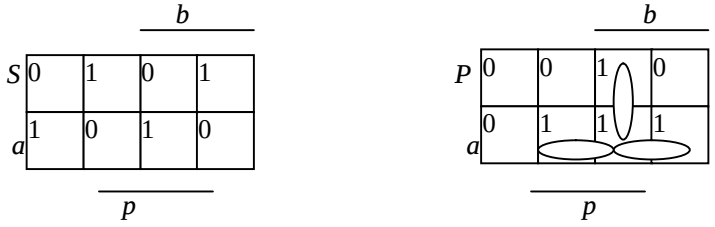


Рис. 1.7

На основании представления функции S на карте можно заключить, что логическое выражение для этой функции не минимизируется.

Минимизированная функция переноса с учетов введенных контуров имеет вид

$$P = ab + ap + bp.$$

Ввиду того что функции P и S формируются в одном и том же узле, при формировании S целесообразно использовать средства, примененные для реализации функции P. С этой целью рассмотрим функцию P как переменную для функции S. Тогда модифицированная функция S, зависящая теперь от четырех переменных a, b, p, P, будет записываться в карту Карно как функция четырех переменных.

На рис. 1.8 приведена карта Карно для такой функции S, а на рис. 1.9 представлены 4 контура, используемых для ее минимизации. В приведенной карте часть клеток, соответствующих наборам переменных, на которых функция не определена, заполнена отметкой «–», т. к. клетки соответствуют наборам, которые невозможны.

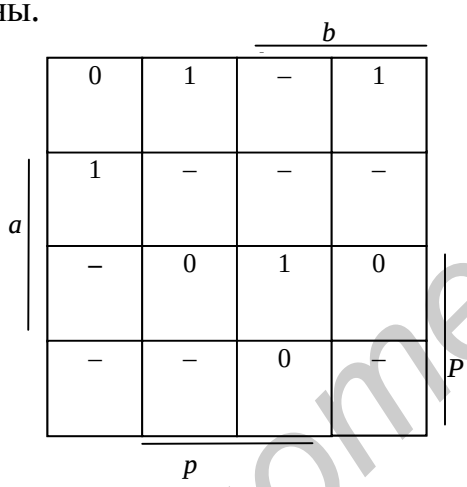


Рис. 1.8

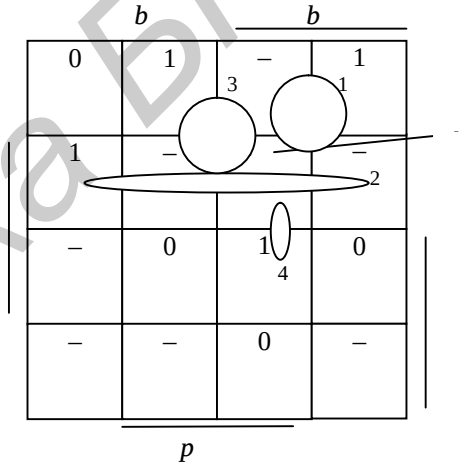


Рис. 1.9

При охвате контурами клетки с отметкой «–» можно включать в контур наряду с клетками, имеющими единичные значения. На основании четырех контуров можно составить минимизированное логическое выражение для функции S, которое имеет вид

$$S = abp + a\overline{P} + b\overline{P} + p\overline{P} = (a + b + p)P + abp.$$

Таким образом, определение функции S как функции четырех переменных позволило получить для ее представления более простое выражение, чем исходная СДНФ для этой функции.

На рис. 1.10 приведена схема одноразрядного двоичного сумматора, реализующая выведенные логические выражения для суммы S и переноса P.

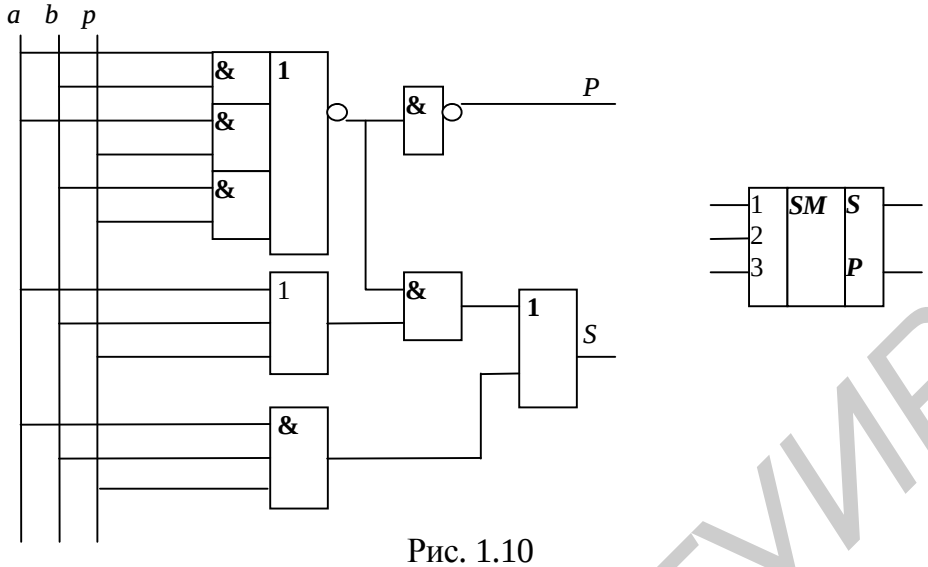


Рис. 1.10

Многоразрядный двоичный сумматор строится на основе одноразрядных сумматоров с введением соответствующих связей между разрядами. На рис. 1.11 приведена простейшая схема такого сумматора. На схеме показана только часть сумматора, относящаяся к некоторому i-му разряду и его соседям: $(i + 1)$-й соседний младший разряд и $(i - 1)$-й соседний старший разряд.

Приведенная схема многоразрядного сумматора называется схемой сумматора с последовательным переносом. Схема очень простая. Сумматор обладает малым быстродействием из-за последовательного учета переноса, возникшего в младшем разряде, в непрерывной цепочке старших разрядов, имеющих значение поразрядной суммы, равное единице. Такие разряды называются разрядами, пропускающими перенос. В худшем случае перенос, возникший в младшем разряде, распространяется до самого старшего разряда формируемой суммы.

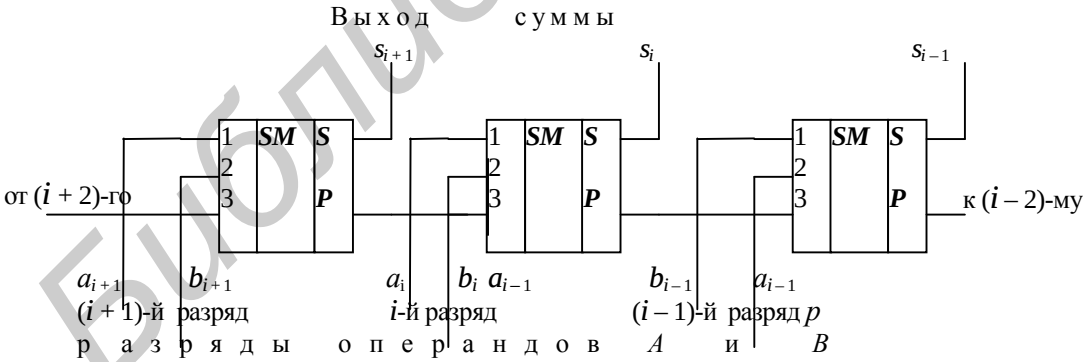


Рис. 1.11

На рис. 1.12 представлена схема сумматора со сквозным переносом. В этом сумматоре перенос, пришедший из младшего разряда на сумматор i-го разряда, поступает на третий вход этого сумматора и одновременно, если поразрядная сумма, сформированная в i-м сумматоре, равна единице, проходит на следующий $(i - 1)$-й сумматор.

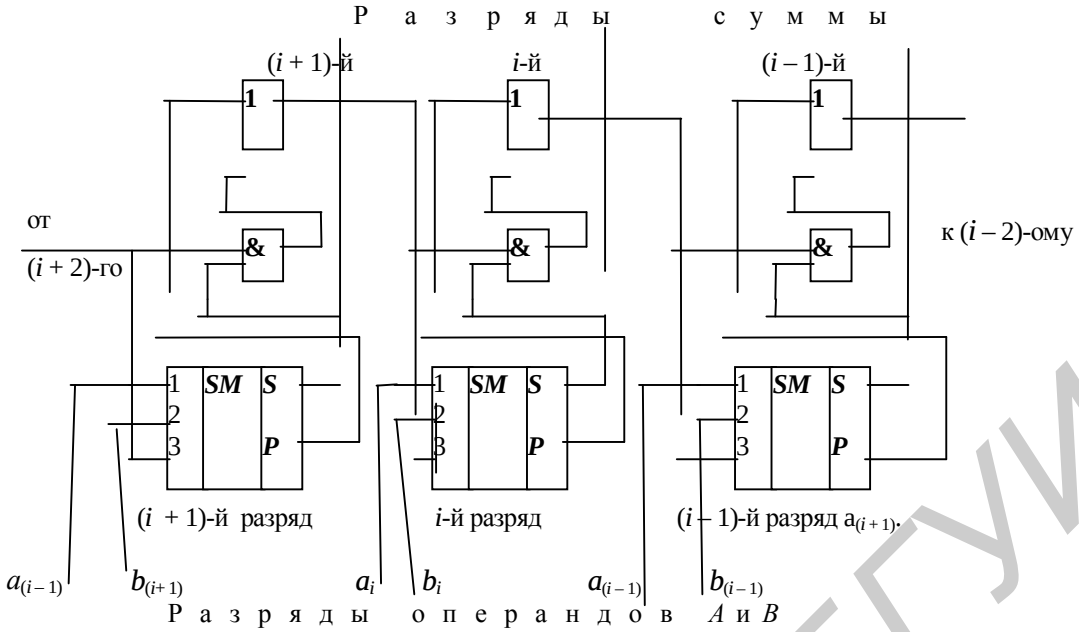


Рис. 1.12

Схема работает в два такта. На первом такте формируется поразрядная сумма и генерируются поразрядные переносы. На втором такте разрешается распространение переносов по разрядам. При этом выработка сигналов переноса на отдельных сумматорах блокируется.

Программируемая логическая матрица

Программируемая логическая матрица (ПЛМ) представляет собой комбинационный узел, обеспечивающий формирование нескольких функциональных зависимостей на основании заданных переменных. Вид функциональных зависимостей программируется.

Программируемая логическая матрица реализует функциональные зависимости в виде дизъюнкции простых конъюнкций. Структурная схема ПЛМ представлена на рис. 1.13.

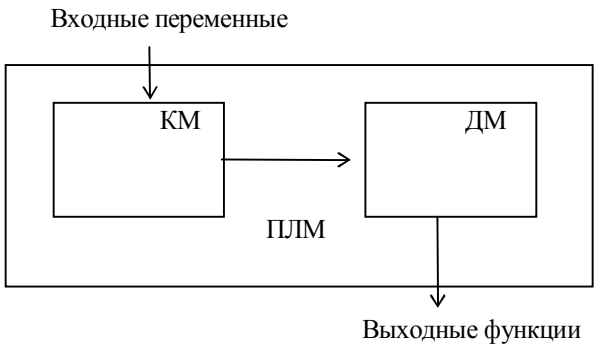


Рис. 1.13

В состав ПЛМ входят конъюнктивная (КМ) и дизъюнктивная (ДМ) матрицы. КМ формирует множество неповторяющихся конъюнкций, используемых во всех формируемых логических функциях. ДМ для каждой выходной функции формирует логическую сумму дизъюнкций соответствующих конъюнкций.

Пример ПЛМ приведен на рис. 1.14. На пересечении горизонтальных и вертикальных шин конъюнктивной матрицы, обозначенных кружком, располагаются цепочки, состоящие из диода (D) и легкоплавкой перемычки (ЛП). В дизъюнктивной матрице в кружках, обозначающих точку пересечения горизонтальных и вертикальных шин, располагаются цепочки, включающие транзистор (Т) и легкоплавкую перемычку. На рис. 1.14 горизонтальные шины конъюнктивной матрицы помечены логическими выражениями формируемых ими конъюнкций. На каждой вертикальной шине дизъюнктивной матрицы реализована логика ИЛИ.

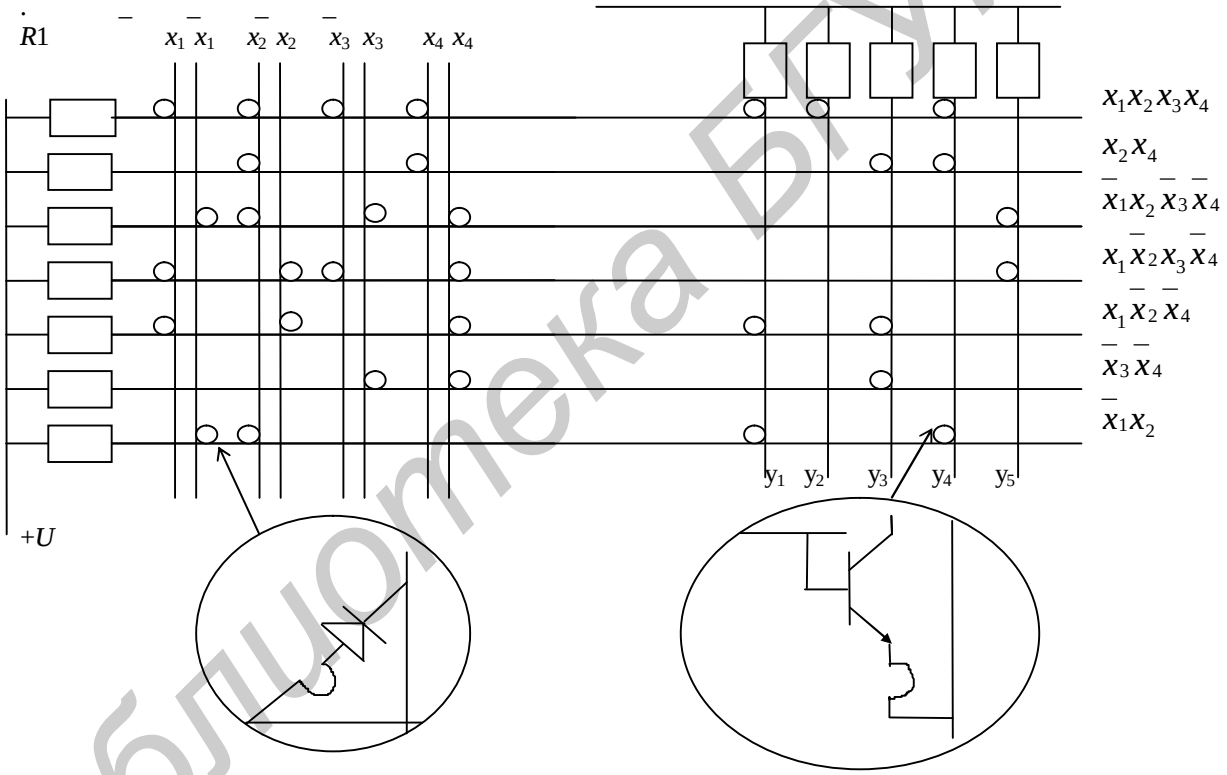


Рис. 1.14

Приведенная матрица реализует следующую логику для выходных функций:

$$y_1 = x_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_4 + x_1\bar{x}_2;\ y_2 = x_1x_2x_3x_4;$$

$$y_3 = x_2x_4 + x_1\bar{x}_2\bar{x}_4 + \bar{x}_3\bar{x}_4;$$

$$y_4 = x_1x_2x_3x_4 + x_2x_4 + \bar{x}_1x_2;\ y_5 = \bar{x}_1x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4.$$

Программирование ПЛМ выполняется следующим образом. При производстве одним из методов интегральной технологии создается заготовка ПЛМ, в которой на пересечениях горизонтальных и вертикальных шин дизъюнктивной мат-

рицы имеет место диодная цепочка, а на пересечениях горизонтальных и вертикальных шин конъюнктивной матрицы располагается транзисторная цепочка. Пользователь в зависимости от логики, которую он собирается реализовать, удаляет перемычки в «ненужных» цепочках. Удаление перемычки осуществляется посредством пропускания по соответствующей горизонтальной и вертикальной шинам мощного тока, который разогревает и испаряет соответствующую легкоплавкую перемычку.

1.2. Накапливающие узлы

Для накапливающего узла характерна зависимость выходных сигналов не только от входных, но и от состояния, в котором находился данный узел в момент воздействия входного сигнала. Это означает, что такие узлы могут хранить «историю» входных сигналов, т. е. узлы данного типа обладают памятью. К типовым накапливающим узлам, используемым в вычислительной технике, относятся следующие узлы.

Регистры

Основная функция регистра – хранение многоразрядного кода. Регистры реализуются на основе элемента типа «триггер».

На рис. 1.15 приведен регистр, построенный на основе *D*-триггера. Отдельные разряды устанавливаемого в регистр кода поступают на вход *D* соответствующих триггеров. Входной код воспринимается регистром только при подаче сигнала «прием кода» (ПК), который поступает на вход синхронизации каждого триггера. Независимо от того, какой код ранее находился в регистре, при наличии сигнала ПК в нем будет установлен код, который в данный момент присутствует на входе регистра.

При снятии сигнала ПК регистр хранит этот код до тех пор, пока не поступит очередной сигнал ПК. Таким образом, особенностью регистра на основе *D*-триггера является то, что он не требует предварительного сброса «старого» кода перед установкой в него «нового» кода.

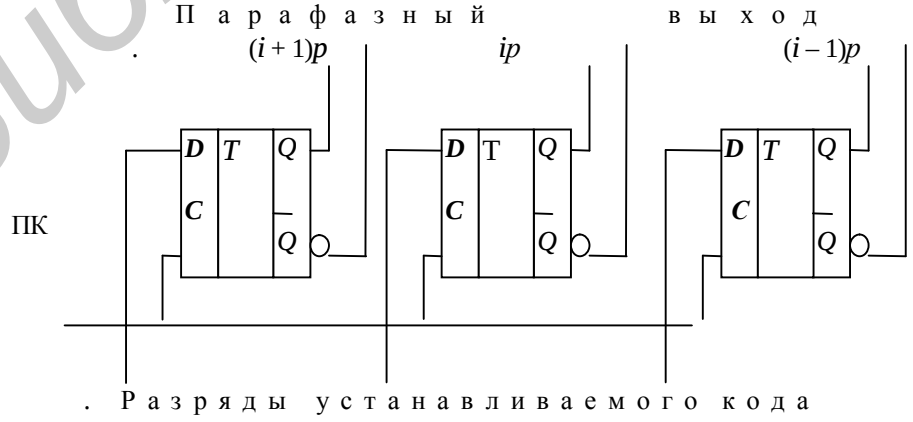


Рис. 1.15

Регистры могут выполнять функцию сдвига хранимого кода вправо или влево. Такие регистры называются сдвигающими. На рис. 1.16 приведена схема сдвигающего регистра.

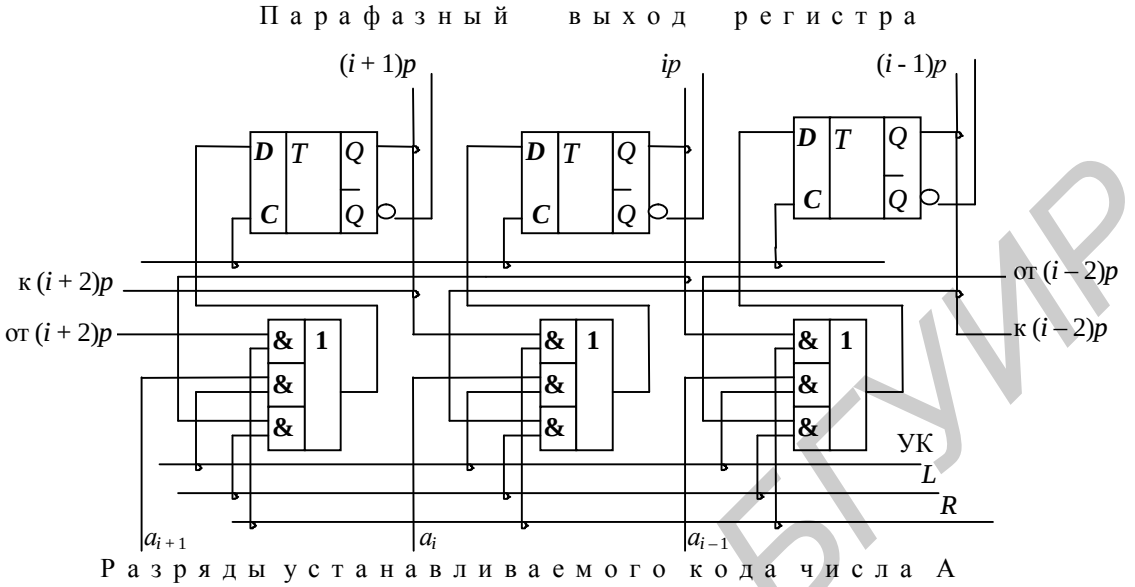


Рис. 1.16

Приведенный на рисунке регистр может выполнять следующие функции:

– прием кода (выполняется по сигналу ПК);
– сдвиг хранимого кода влево (выполняется по сигналу *L*);
– сдвиг хранимого кода вправо (выполняется по сигналу *R*).

На входе каждого разрядного триггера *T* используется логическая схема, которая обеспечивает подключение ко входу *D* некоторого *i*-го триггера или соответствующий *i*-й разряд входного кода (для установки в регистре кода по сигналу ПК), или выход единицы триггера ближайшего старшего (*i* – 1)-го разряда (для сдвига кода вправо, если есть сигнал *R*), или выход единицы триггера ближайшего младшего (*i* – 1)-го разряда (для сдвига кода влево, если есть сигнал *L*). Многоразрядный выход регистра представлен выходом единицы и выходом нуля каждого триггера. Это позволяет формировать парафазный выход регистра.

Счетчик

Счетчик в общем случае представляет собой типовой узел, который по каждому входному сигналу изменяет (увеличивает или уменьшает) хранимый в ней код на единицу.

Используются следующие виды счетчика:

– счетчики прямого счета;
– счетчики обратного счета;
– реверсивные счетчики.

Счетчик прямого счета в качестве входного сигнала использует сигнал +1. По каждому входному сигналу он увеличивают значение хранимого в нем кода на единицу. На рис. 1.17 приведена схема счетчика прямого счета, построенного на базе двухтактного Т-триггера. Счетчик имеет три разряда и может считать от 0 до 7. Приведенный счетчик можно рассматривать как сумматор по модулю 8 количества сигналов, поступающих на его вход.

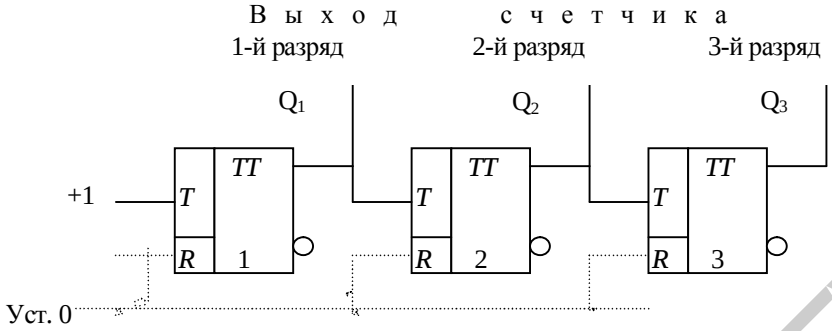


Рис. 1.17

На схеме приведен вход установки нуля счетчика «Уст. 0», который подключен ко входам установки нуля (вход *R*) каждого триггера. Сигнал по этому входу приводит рассматриваемый счетчик в состояние «0» (все составляющие его триггеры имеют в этом случае состояние «0»). Вход *R* в классическом варианте *T*-триггера отсутствует. Однако он часто вводится в реальный *T*-триггер, позволяя решить проблему задания начального значения.

На рис. 1.18 приведена временная диаграмма, поясняющая работу данного счетчика.

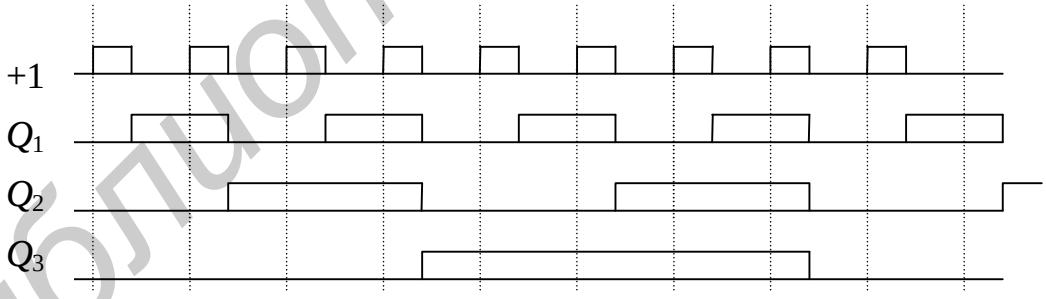


Рис. 1.18

На временной диаграмме в качестве начального состояния счетчика используется состояние «0», когда во всех триггерах имеет место 0. На счетчик поступает девять входных сигналов +1. Так как вход счетчика подключен непосредственно ко входу Т первого двухтактного Т-триггера, выходной сигнал Q_1 этого триггера при поступлении каждого входного сигнала будет изменяться на противоположный, причем смена сигнала Q_1 осуществляется по заднему фронту сигнала +1 (свойство двухтактного триггера).

Выходной сигнал второго триггера Q_2 будет изменяться по заднему фронту выходного сигнала первого триггера, являющегося входным сигналом для второго триг-

гера. Входным сигналом для третьего триггера является выходной сигнал второго триггера. Поэтому сигнал Q_3 будет изменяться по заднему фронту сигнала Q_2.

Счетчик обратного счета в качестве входного сигнала использует сигнал –1 и по каждому входному сигналу уменьшает значение хранимого в нем кода на единицу. На рис. 1.19 приведена схема счетчика обратного счета, построенная на базе двухтактного *T*-триггера. Счетчик имеет три разряда и может считать от 7 до 0. На схеме вход установки нуля «Уст. 0» подключен ко входам установки нуля *R* каждого триггера.

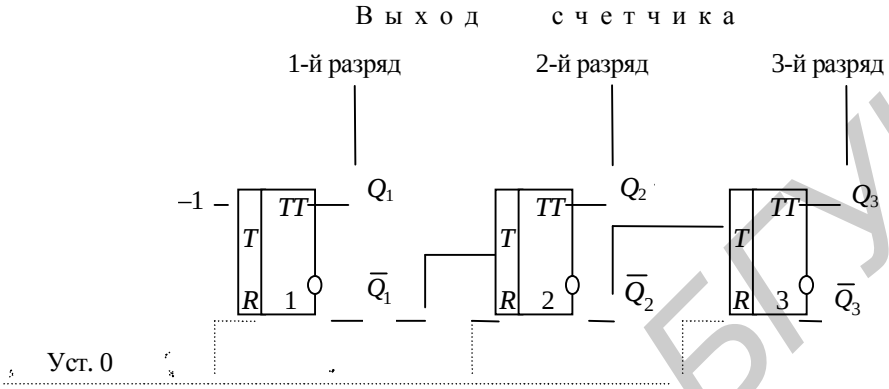


Рис. 1.19

На рис. 1.20 показана временная диаграмма, поясняющая работу данного счетчика. На временной диаграмме в качестве начального состояния приведено состояние «7», когда во всех триггерах имеет место 1. На счетчик поступает девять входных сигналов –1. Так как вход счетчика подключен непосредственно ко входу *T* первого двухтактного триггера, то выходной сигнал Q_1 этого триггера при поступлении каждого входного сигнала будет изменяться на противоположный, причем смена сигнала Q_1 осуществляется по заднему фронту сигнала –1 (свойство двухтактного триггера).

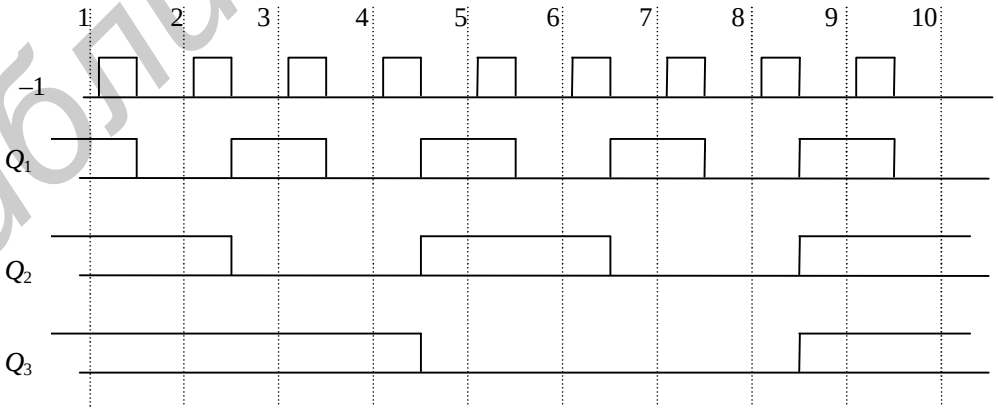


Рис. 1.20

Выходной сигнал Q_2 второго триггера будет изменяться по заднему фронту выходного сигнала выхода «0» первого триггера, являющегося ин-

версией выхода Q_1, т. е. выходной сигнал будет изменяться по переднему фронту сигнала Q_1.

Входным сигналом для третьего триггера является выходной сигнал выхода нуля второго триггера. Выход нуля второго триггера является инверсией выхода единицы этого же триггера. Поэтому сигнал Q_3 будет изменяться по переднему фронту сигнала Q_2.

Реверсивный двоичный счетчик в зависимости от управляющих сигналов может работать в режиме прямого или обратного счета. На рис. 1.21 приведен пример реализации такого счетчика. На схеме показаны управляющие сигналы:

P«+» – сигнал установки режима прямого счета;

P«–» – сигнал установки режима обратного счета;

Уст. 0 – сигнал установки нулевого кода в счетчике;

1 – сигнал модификации значения кода в счетчике на единицу.

Логические схемы на входах второго и третьего разрядов в зависимости от управляющих сигналов подключают ко входу *T* соответствующего триггера или выхода единицы, или выхода нуля триггера предыдущего младшего разряда.

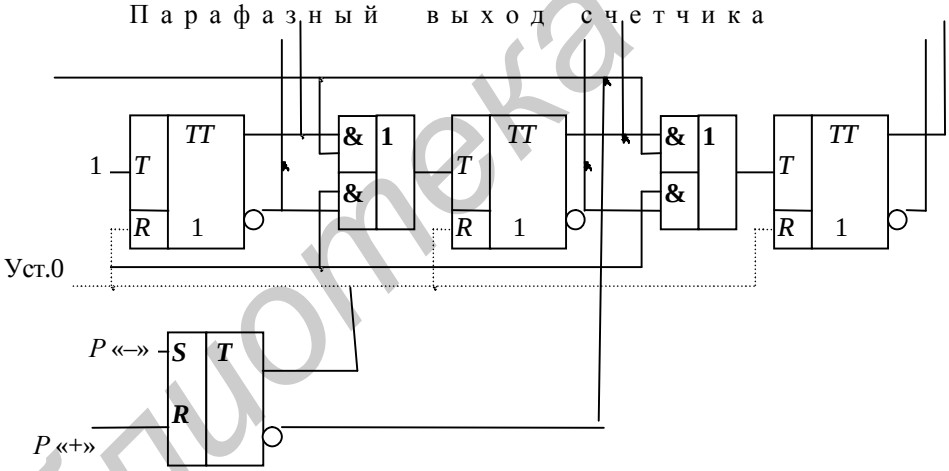


Рис. 1.21

Двоично-десятичный счетчик в системе 8, 4, 2, 1 отличается от ранее рассмотренных двоичных счетчиков тем, что каждая цифра представления десятичного числа, фиксируемая в четырех разрядах двоичного кода, может изменяться в диапазоне от 0 до 9 (а не до 16, как это имело бы место, если четыре разряда отражали бы код двоичного числа).

Основу двоично-десятичного счетчика представляет 4-разрядный счетчик десятичной цифры, который в свою очередь строится на основе четырех триггеров.

Двоично-десятичный счетчик на одну десятичную цифру может быть построен с использованием триггера любого из рассмотренных типов. Если счетчик строить на основе *T*-триггера, то задать его работу можно с помощью табл. 1.2.

В таблице заданы переходы из всех возможных начальных состояний, определяемых четырьмя разрядами $Q_1(t)Q_2(t)Q_3(t)Q_4(t)$, в конечные состояния $(Q_1(t+1)Q_2(t+1)Q_3(t+1)Q_4(t+1))$ при подаче на вход счетчика сигнала +1. В столбцах q_{T1}, q_{T2}, q_{T3}, q_{T4}, единицами отмечены случаи, когда нужно подавать сигналы на вход соответствующего триггера для формирования кода нового состояния счетчика. Информация из этих столбцов позволяет формировать логические выражения для сигналов, подаваемых на входы триггеров двоично-десятичного счетчика.

Таблица 1.2

+1	$Q_1(t)$	$Q_2(t)$	$Q_3(t)$	$Q_4(t)$	$Q_1(t+1)$	$Q_2(t+1)$	$Q_3(t+1)$	$Q_4(t+1)$	q_{T1}	q_{T2}	q_{T3}	q_{T4}
1	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	1	1
1	0	0	1	0	0	0	1	1	0	0	0	1
1	0	0	1	1	0	1	0	0	0	1	1	1
1	0	1	0	0	0	1	0	1	0	0	0	1
1	0	1	0	1	0	1	1	0	0	0	1	1
1	0	1	1	0	0	1	1	1	0	0	0	1
1	0	1	1	1	1	0	0	0	1	1	1	1
1	1	0	0	0	1	0	0	1	0	0	0	1
1	1	0	0	1	0	0	0	0	1	0	0	1

На основании приведенной таблицы можно записать функции q_{T1}, q_{T2}, q_{T3}, q_{T4} следующим образом:

$$q_{T1} = \overline{Q}_1 Q_2 Q_3 Q_4 + Q_1 \overline{Q}_2 \overline{Q}_3 Q_4;\ q_{T2} = \overline{Q}_1 Q_2 Q_3 Q_4 + \overline{Q}_1 \overline{Q}_2 Q_3 Q_4;$$

$$q_{T3} = \overline{Q}_1 \overline{Q}_2 \overline{Q}_3 Q_4 + \overline{Q}_1 \overline{Q}_2 Q_3 Q_4 + \overline{Q}_1 Q_2 \overline{Q}_3 Q_4 + \overline{Q}_1 Q_2 Q_3 Q_4.$$

$q_{T4} = 1$, т. е. эта функция не зависит от начального состояния. При подаче каждого сигнала +1 на счетчик необходимо формировать сигнал на вход четвертого триггера (триггера младшего разряда счетчика).

Для нахождения минимальных выражений для рассматриваемых функций используем карты Карно.

На рис. 1.22 приведены записи в карту Карно функций q_{T1}, q_{T2} и q_{T3}. На рис. 1.23 приведены те же самые карты с введенными на них контурами.

На основания карт (см. рис. 1.23) можно записать минимизированные выражения для функций q_{T1}, q_{T2}, q_{T3} и q_{T4}:

$$q_{T1} = Q_2 Q_3 Q_4 + Q_1 Q_4;\ q_{T2} = Q_3 Q_4;\ q_{T3} = \overline{Q}_1 Q_4;\ q_4 = 1.$$

Полученные логические выражения позволяют синтезировать логические схемы формирования сигналов изменения текущего состояния всех че-

тырех триггеров, входящих в состав двоично-десятичного счетчика, и, следовательно, и весь счетчик.

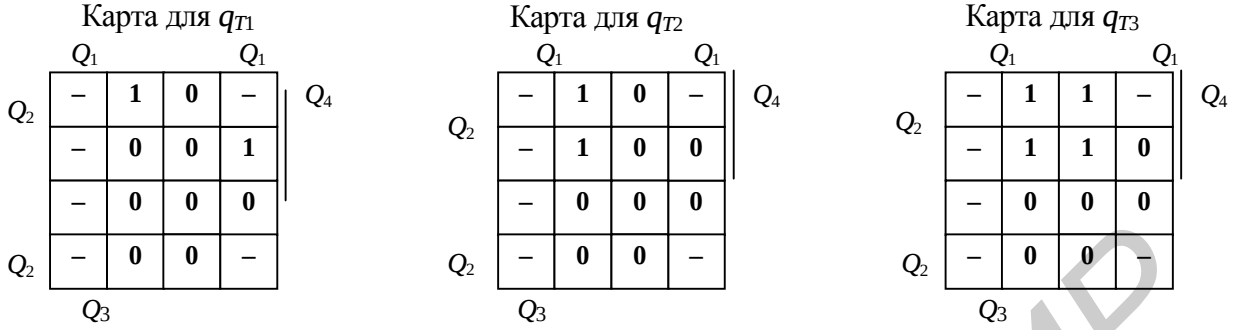


Рис. 1.22

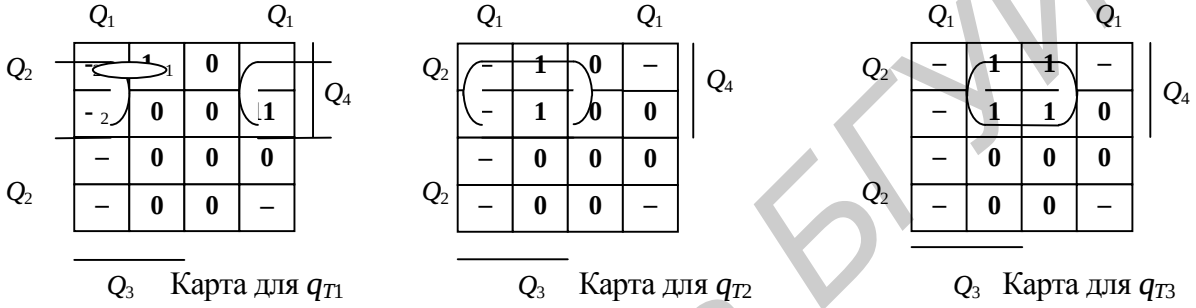


Рис. 1.23

Схема синтезированного двоично-десятичного счетчика на базе *T*-триггера приведена на рис. 1.24.

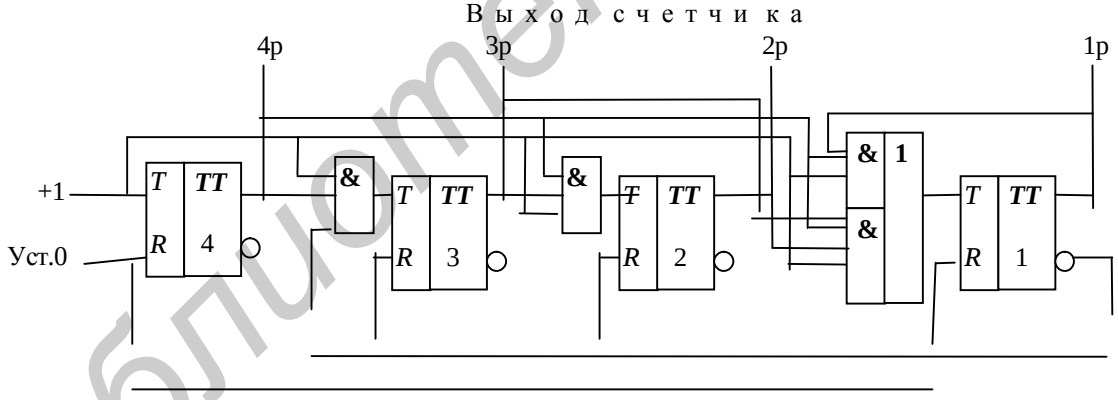


Рис. 1.24

1.3. Элементы теории цифровых автоматов

1.3.1. Основные определения

Цифровой автомат представляет собой объект, определяемый следующими характеристиками:

$\{A, W, Z, \delta, \lambda, a_н\}$,

где $A = \{a_1, a_2, ..., a_n\}$ – множество состояний цифрового автомата;

$W = \{\varpi_1, \varpi_2, ..., \varpi_m\}$ – множество выходных сигналов цифрового автомата;

$Z = \{z_1, z_2, ..., z_n\}$ – множество входных сигналов цифрового автомата;

δ – функция перехода цифрового автомата в новое состояние $a(t + 1)$ в зависимости от его текущего состояния $a(t)$ и действующего на его входе сигнала $z(t)$: $a(t + 1) = \delta\,(a(t), z(t))$;

λ – функция выработки выходного сигнала $\varpi(t + 1)$ в зависимости от текущего состояния $a(t)$ цифрового автомата и действующего на его входе сигнала $z(t)$: $\varpi\,(t + 1) = \lambda\,(a(t), z(t))$;

$a_н \in A$ – начальное состояние цифрового автомата.

Если рассматривать множество входных сигналов Z как множество букв входного алфавита, а множество выходных сигналов W – как множество букв выходного алфавита, то цифровой автомат можно рассматривать как преобразователь слов, т. е. входному слову, состоящему из S букв входного алфавита, он ставит в соответствие выходное слово, состоящее из S букв выходного алфавита. В этом случае эквивалентность двух цифровых автоматов можно определить следующим образом: два цифровых автомата являются эквивалентными, если они, имея одинаковые множества входных и выходных сигналов, являются одинаковыми преобразователями слов, т. е. при любом начальном состоянии цифровые автоматы для любого входного слова формируют одинаковые выходные слова.

Цифровой автомат называется *конечным*, если используемые для его задания множества конечны.

Цифровой автомат является *полностью определенным*, если каждой паре $a(t)$, $z(t)$ ставится в соответствие пара $a(t + 1)$, $\varpi(t + 1)$. В противном случае цифровой автомат называется *частично определенным* или просто частичным.

Практически все накапливающие узлы ЭВМ можно рассматривать как цифровые автоматы. Поэтому хорошо разработанная теория цифровых автоматов находит широкое применение для синтеза и анализа сложных схем в вычислительной технике.

Существуют два типа цифровых автоматов: автомат Мили и автомат Мура.

Автомат Мили определяют как автомат общего типа, т. е. его выходной сигнал и новое состояние являются функцией текущего состояния и действующего на входе сигнала.

Что же касается *автомата Мура*, то его выходной сигнал прямо не зависит от входного сигнала и определяется состоянием автомата, т. е. работа цифрового автомата типа Мура задается в виде уравнений

$a(t + 1) = \delta(a(t), z(t))$;

$\varpi(t + 1) = \lambda(a(t))$.

Зависимость выходного сигнала от входного в автомате Мура проявляется косвенно, т. е. через зависимость состояния от входного сигнала.

Цифровой автомат может быть задан с помощью таблиц или посредством графа.

Задание цифрового автомата с помощью таблицы

Автомат Мили задается с помощью двух таблиц:

– таблицы переходов (рис. 1.25, а), определяющей правило формирования нового состояния на основании текущего состояния и действующего входного сигнала;

– таблицы выходов (рис. 1.25, б), определяющей правило формирования выходного сигнала на основании текущего состояния и действующего входного сигнала.

	a_1	a_2	a_3	a_4
z_1	a_3	a_1	a_2	a_4
z_2	a_4	a_3	a_1	a_1
z_3	a_4	a_4	a_4	a_2

а

	a_1	a_2	a_3	a_4
z_1	w_1	w_1	w_3	w_4
z_2	w_3	w_4	w_5	w_2
z_3	w_5	w_2	w_2	w_1

б

Рис. 1.25

Приведенный цифровой автомат имеет множество входных сигналов $Z = \{z_1, z_2, z_3\}$, множество состояний $A = \{a_1, a_2, a_3, a_4\}$ и множество выходных сигналов $W = \{w_1, w_2, w_3, w_4, w_5\}$.

Таблицы переходов и выходных сигналов имеют одинаковую размерность и одинаковую разметку столбцов и строк, что позволяет объединять их в одну таблицу. На рис. 1.26, а приведена объединенная таблица, соответствующая таблицам на рис. 1.25.

Таблицы переходов и выходов задают следующую дисциплину переходов и формирования выходного сигнала: если автомат находится в текущий момент в состоянии a_j и получает по входу сигнал z_i, то он переходит в новое состояние, которое записано в клетке таблицы переходов, расположенной на пересечении j-го столбца и i-й строки, и вырабатывает выходной сигнал, который определен значением в этой клетке.

Например, если автомат находится в состоянии a_4, а на вход поступает сигнал z_2, то он переходит в состояние a_1 (см. рис. 1.25, а) и вырабатывает выходной сигнал w_2 (см. рис. 1.25, б).

Работа данного автомата как преобразователя слов иллюстрируется следующим образом.

Предположим, что на вход автомата, находящегося в начальном состоянии $a_н = a_3$, поступает входное слово в виде последовательности букв z_1, z_1, z_3, z_2, z_2, z_1, тогда цифровой автомат будет переходить в последовательность состояний (см. рис. 1.25, а) и вырабатывать последовательность выходных сигналов (см. рис. 1.25, б), имеющих вид

$a(t)$: a_3, a_2, a_1, a_4, a_1, a_4;
$z(t)$: z_1, z_1, z_3, z_2, z_2, z_1;
$a(t+1)$: a_2, a_1, a_4, a_1, a_4, a_4;
$\varpi(t+1)$: ϖ_3, ϖ_1, ϖ_5, ϖ_2, ϖ_3, ϖ_4.

В приведенных последовательностях приняты следующие обозначения:

– $a(t)$ – текущее состояние цифрового автомата;

– $z(t)$ – входной сигнал, действующий на текущем такте работы цифрового автомата;

– $a(t+1)$, $\varpi(t+1)$ – новое состояние цифрового автомата и вырабатываемая им буква выходного алфавита соответственно.

Отработка входного слова из шести букв осуществляется за шесть тактов входного автомата. Новое состояние на такте является одновременно текущим состоянием на следующем такте.

Из приведенных последовательностей видно, что в ответ на поступившее входное слово z_1, z_1, z_3, z_2, z_2, z_1, цифровой автомат, находившийся в исходном состоянии a_3, выдает выходное слово ϖ_3, ϖ_1, ϖ_5, ϖ_2, ϖ_3, ϖ_4, содержащее столько же букв, что и входное слово, и в конце преобразования оказывается в состоянии a_4. Очевидно, что при другом исходном состоянии, отличным от a_3, цифровой автомат выработает выходное слово, отличное от ϖ_3, ϖ_1, ϖ_5, ϖ_2, ϖ_3,ϖ_4. В этом можно убедиться, взяв другое начального состояние, и для того же входного слова найти выходное слово, которое сформирует цифровой автомат.

Автомат Мура задается с помощью одной таблицы, в которой определяются правило формирования нового состояния на основании текущего состояния и действующего входного сигнала и связь выходного сигнала с состоянием цифрового автомата. На рис. 1.26, б приведен пример табличного задания автомата Мура.

Таблица переходов, так же как и в случае автомата Мили, задает следующую дисциплину переходов: если автомат находится в текущий момент в состоянии a_j и получает по входу сигнал z_i, то он переходит в новое состояние, которое записано в клетке таблицы переходов, расположенной на пересечении j-го столбца и i-й строки, и вырабатывает выходной сигнал, который определяется новым состоянием цифрового автомата. На рис. 1.26, б в самой верхней строке таблицы перечислены выходные сигналы в их взаимосвязи с состояниями, перечисленными во второй строке. Например, если автомат находится в состоянии a_4, а на вход поступает сигнал z_2, то он переходит в состояние a_3, которому соответствует выходной сигнал ϖ_1.

	a_1	a_2	a_3	a_4
z_1	a_3 / ϖ_1	a_1 / ϖ_1	a_2 / ϖ_3	a_4 / ϖ_4
z_2	a_4 / ϖ_3	a_3 / ϖ_4	a_1 / ϖ_5	a_1 / ϖ_2
z_3	a_4 / ϖ_5	a_4 / ϖ_2	a_4 / ϖ_2	a_2 / ϖ_1

а

	ϖ_2	ϖ_3	ϖ_1	ϖ_1
	a_1	a_2	a_3	a_4
z_1	a_2	a_1	a_1	a_4
z_2	a_3	a_4	a_1	a_3
z_3	a_4	a_3	a_4	a_2

б

Рис. 1.26
а – автомат Мили; б – автомат Мура

Работа данного автомата как преобразователя слов иллюстрируется следующим образом.

Предположим, что на вход автомата, находящегося в начальном состоянии $a_н = a_2$, поступает входное слово в виде последовательности букв z_1, z_3, z_3, z_2, z_3, z_1, тогда цифровой автомат будет переходить в последовательность состояний и вырабатывать последовательность выходных сигналов, имеющих вид (см. рис. 1.26, б):

$a(t)$: a_2, a_1, a_4, a_2, a_4, a_2;
$z(t)$: z_1, z_3, z_3, z_2, z_3, z_1;
$a(t + 1)$: a_1, a_4, a_2, a_4, a_2, a_1;
$\varpi(t + 1)$: ϖ_2, ϖ_1, ϖ_3, ϖ_1, ϖ_3, ϖ_2.

В приведенных последовательностях приняты следующие обозначения:

$a(t)$ – текущее состояние цифрового автомата;

$z(t)$ – входной сигнал, действующий на текущем такте работы цифрового автомата;

$a(t + 1)$, $\varpi(t + 1)$ – новое состояние цифрового автомата и вырабатываемый выходной сигнал соответственно.

Отработка входного слова из шести букв осуществляется за шесть тактов работы входного автомата. Новое состояние на текущем такте является одновременно текущим состоянием на следующем такте.

Из приведенных последовательностей видно, что в ответ на поступившее входное слово z_1, z_3, z_3, z_2, z_3, z_1 цифровой автомат, находившийся в исходном состоянии a_2, выдает выходное слово ϖ_2, ϖ_1, ϖ_3, ϖ_1, ϖ_3, ϖ_2, содержащее столько же букв, что и входное слово, и в конце преобразования оказывается в состоянии a_1. Очевидно, что при другом исходном состоянии цифровой автомат в общем случае выработает выходное слово, отличное от ϖ_2, ϖ_1, ϖ_3, ϖ_1, ϖ_3, ϖ_2.

Задание цифрового автомата с помощью графа

Цифровой автомат может быть задан графом, количество вершин которого равно количеству состояний цифрового автомата. Вершины графа соединены дугами, указывающими возможный переход из одного состояния в другое, помеченными входными сигналами, при которых имеет место представляемый дугой переход.

В автоматах Мили дуги также помечаются выходными сигналами, которые вырабатываются цифровым автоматом при переходе. В автоматах Мура выходными сигналами помечаются вершины графа. На рис. 1.27 представлен граф, соответствующий автомату Мили, заданному таблицей (см. рис. 1.26, а), а на рис. 1.28 приведен граф автомата Мура, заданного таблицей (см. рис. 1.26, б).

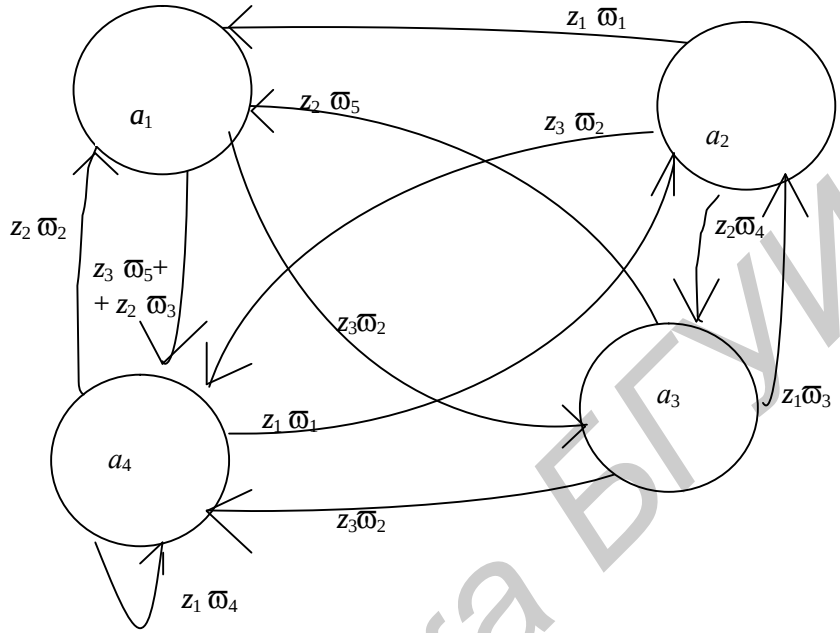


Рис. 1.27

Стрелки на дугах показывают, в какое состояние осуществляется переход. В случае если из одной вершины в другую имеют место переходы по нескольким входным сигналам, то такой переход обозначается одной дугой, на которой через знак логической суммы перечисляются все входные сигналы.

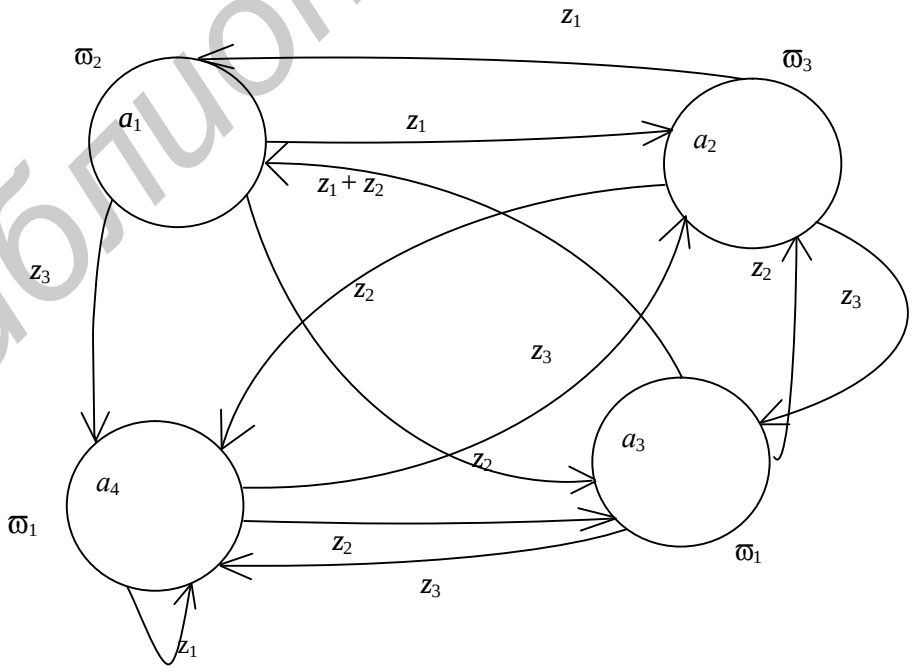


Рис. 1.28

На графе автомата Мили переход из вершины a_1 в вершину a_4 имеет место при входном сигнале z_3 с выработкой выходного сигнала ϖ_5 и при входном сигнале z_2 с выработкой выходного сигнала ϖ_3. Поэтому стрелка перехода из вершины a_1 в вершину a_4 помечена логическим условием $z_3\varpi_5 + z_2\varpi_3$.

1.3.2. Переход от одной формы задания автомата к другой

Любой цифровой автомат можно представить в виде автомата Мура или в виде автомата Мили.

Переход от автомата Мура к автомату Мили

Процедуру рассматриваемого перехода иллюстрирует следующий пример.

Имеется цифровой автомат Мура, заданный в виде таблицы, приведенной на рис. 1.29, а.

	ϖ_1	ϖ_2	ϖ_2	ϖ_4	ϖ_3
	A_1	A_2	A_3	A_4	A_5
Z_1	A_3	A_3	A_2	A_2	A_5
Z_2	A_5	A_5	A_5	A_5	A_4
Z_3	A_1	A_1	A_4	A_4	A_2

а

	C_1	C_2	C_3
Z_1	C_2 / ϖ_2	C_1 / ϖ_2	C_3 / ϖ_3
Z_2	C_3 / ϖ_3	C_3 / ϖ_3	C_2 / ϖ_4
Z_3	C_1 / ϖ_1	C_2 / ϖ_4	C_1 / ϖ_2

б

	ϖ_2	ϖ_2	ϖ_3	ϖ_4	ϖ_1
	B_1	B_2	B_3	B_4	B_5
Z_1	B_2	B_1	B_3	B_2	B_1
Z_2	B_3	B_3	B_4	B_3	B_3
Z_3	B_4	B_5	B_2	B_4	B_5

в

Рис. 1.29

Найти задание автомата Мили, эквивалентного заданному автомату Мура.

Решение

Из множества состояний автомата Мура составим подмножества, каждое из которых включает состояния, имеющие одинаковые переходы. Каждому полученному подмножеству поставим в соответствие состояние C_i искомого автомата Мили:

$\{A_1, A_2\} - C_1$;

$\{A_3, A_4\} - C_2$;

$\{A_5\} - C_3$.

Таким образом, искомый автомат Мили будет иметь три состояния, что позволяет разметить столбцы и строки таблицы задания искомого автомата Мили. Таблица имеет вид, приведенный на рис. 1.29, б.

Заполнение клеток таблицы выполняется следующим образом.

Для того чтобы найти переход из C_1 при поступлении Z_1, рассмотрим переходы, которые имеют место для любого из состояний автомата Мура, объединенных в подмножество, обозначенное C_1. Из элементов $\{A_1, A_2\}$ возьмем, например, A_2.

В таблице автомата Мура для этого состояния при входном сигнале Z_1 имеет место переход в состояние A_3, которое входит в подмножество, обозначенное как C_2. Кроме того, состоянию A_3, как видно из той же таблицы, соответствует выходной сигнал ϖ_2. Из этого следует, что в клетку, соответствующую переходу из C_1 по входному сигналу Z_1, нужно поставить C_2 и указать его в качестве вырабатываемого выходного сигнала ϖ_2.

Рассмотрим заполнение клетки перехода из состояния C_3 при входном сигнале Z_2. Из подмножества $\{A_5,\}$, соответствующего C_3, берем состояние автомата Мура A_5. В таблице автомата Мура для этого состояния при входном сигнале Z_2 имеет место переход в состояние A_4, которое входит в подмножество, обозначенное как C_2. Кроме того, состоянию A_4, как видно из той же таблице, соответствует выходной сигнал ϖ_4. Из этого следует, что в клетку, соответствующую переходу из C_3 по входному сигналу Z_2, нужно поставить C_2 и указать его в качестве вырабатываемого выходного сигнала ϖ_4. Аналогичным образом заполняются все клетки приведенной таблицы.

Переход от автомата Мили к автомату Мура

Процедуру данного перехода иллюстрирует следующий пример.

Имеется цифровой автомат Мили, заданный в виде таблицы (см. рис. 1.29, б). Найти задание автомата Мили, эквивалентного заданному автомату Мура.

Решение

Составим множество неповторяющихся пар $C_i\varpi_j$. Каждому элементу этого множества поставим в соответствие одно из состояний искомого автомата Мура:

$C_2\varpi_2 - B_1$;

$C_1\varpi_2 - B_2$;

$C_3\varpi_3 - B_3$;

$C_2\varpi_4 - B_4$;

$C_1\varpi_1 - B_5$.

Для пяти состояний и трех входов таблица задания искомого автомата Мура будет иметь вид, приведенный на рис. 1.29, в. В столбцах таблицы отмечено пять найденных состояний автомата и выходные сигналы (см. пары $C_i\varpi_j$).

Заполнение клеток переходов таблицы осуществлено следующим образом.

Переход из B_1 по входному сигналу Z_1 будет соответствовать переходу из C_2 по сигналу Z_1. Из таблицы на рис. 1.29, б, видно, что при текущем состоянии C_2 по сигналу Z_1 осуществляется переход в состояние C_1 и вырабатывается выходной сигнал ϖ_2, что соответствует состоянию B_2. Поэтому в клетку, указывающую на переход из B_1 по входному сигналу Z_1, записывается B_2.

Аналогично заполняются все остальные клетки таблицы.

Полученный автомат Мили должен быть эквивалентен автомату Мили, заданному в таблице на рис. 1.29, б, и автомату Мура, заданному в таблице на рис. 1.29, а. Для проверки эквивалентности полученных автоматов возьмем в каче-

стве начального состояния для автомата Мура, заданного в таблице на рис. 1.29, а, $A_н = A_3$, а в качестве входного слова – $Z_2\ Z_1\ Z_1\ Z_3\ Z_2$.

Основываясь на таблице, приведенной на рис. 1.29, б, составим для выбранного входного слова последовательность состояний и формируемых выходных сигналов. Эта последовательность будет иметь вид:

$A(t) – A_3\ A_5\ A_5\ A_5\ A_2$;

$X(t) – Z_2\ Z_1\ Z_1\ Z_3\ Z_2$;

$A(t + 1) – A_5\ A_5\ A_5\ A_2\ A_5$;

$\omega\ (t + 1) – \varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$.

Таким образом, рассмотренный автомат на заданное входное слово выработал выходное слово $\varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$ и оказался в состоянии A_5. Найдем реакцию на то же входное слово автомата Мили, представленного в таблице на рис. 1.29, б.

На основании данной таблицы составим для заданного входного слова последовательность состояний и формируемых выходных сигналов. В качестве исходного состояния возьмем C_2, так как оно отражает подмножество состояний $\{A_3, A_3\}$, в которое входит начальное состояние ранее рассмотренного цифрового автомата Мура.

Эта последовательность будет иметь вид:

$C(t) – C_2\ C_3\ C_3\ C_3\ C_1$;

$X(t) – Z_2\ Z_1\ Z_1\ Z_3\ Z_2$;

$C(t + 1) – C_3\ C_3\ C_3\ C_1\ C_3$;

$\omega(t + 1) – \varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$.

Таким образом, рассмотренный автомат на заданное входное слово выработал выходное слово $\varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$ и оказался в состоянии C_1.

На основании таблицы автомата Мура, приведенной на рис. 1.29, в, составим для заданного входного слова последовательность состояний и формируемых выходных сигналов. В качестве исходного состояния используем B_1, которое отражает пару $C_2\varpi_2$, где C_2 используется в качестве исходного состояния ранее рассмотренного цифрового автомата Мили (с таким же успехом можно взять в качестве начального состояние B_4).

Эта последовательность будет иметь вид:

$B(t) – B_1\ B_3\ B_3\ B_3\ B_2$;

$X(t) – Z_2\ Z_1\ Z_1\ Z_3\ Z_2$;

$C(t + 1) – B_3\ B_3\ B_3\ B_2\ B_3$;

$\omega(t + 1) – \varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$.

Таким образом, рассмотренный автомат на заданное входное слово выработал выходное слово $\varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$ и оказался в состоянии C_2. Следовательно, все три рассмотренных автомата при соответствующих начальных состояниях одинаково преобразовали входное слово $Z_2\ Z_1\ Z_1\ Z_3\ Z_2$ в выходное слово $\varpi_3\ \varpi_3\ \varpi_3\ \varpi_2\ \varpi_3$. Отметим, что после преобразования заданного входного слова все три автомата перешли в состояния A_5, C_3, B_3 соответственно.

1.3.3. Синтез цифрового автомата

С точки зрения синтеза цифровой автомат удобно представить в виде структурной схемы, приведенной на рис. 1.30.

Цифровой автомат в общем случае состоит из памяти и логической части. Память хранит информацию о предыстории цифрового автомата, вырабатывая сигнал, характеризующий текущее состояние автомата. Логическая часть на основании входного и поступающего из памяти сигналов вырабатывает выходной сигнал. Кроме того, используя входной сигнал и сигнал состояния, логическая часть вырабатывает сигнал управления памятью, обеспечивающий переход из текущего в новое состояние.

Рис. 1.30

Память реализуется на триггерах, количество которых должно позволять запоминать любое состояние из множества состояний цифрового автомата, а сигналы ее управления должны соответствовать оговоренным при задании цифрового автомата переходам и типу используемого триггера для построения памяти.

Построение логической части предполагает представление входных и выходных сигналов цифрового автомата на языке алгебры логики, т. е. в виде логических переменных и функций.

Учитывая вышеизложенное, можно выполнить синтез цифрового автомата следующим образом.

1. Кодирование входных сигналов в виде набора логических переменных.
2. Кодирование выходных сигналов в виде набора логических функций.
3. Кодирование состояний цифрового автомата.
4. Формирование кодированной таблицы переходов и выходов.
5. Выбор типа запоминающего элемента.
6. Составление логических выражений для логических функций, использованных для кодировки выходных сигналов.
7. Составление логических выражений для сигналов управления памятью.
8. Синтез логических схем для сформированных логических выражений.
9. Формирование выходных сигналов цифрового автомата на основании кодирующих их функций.

Рассмотрим реализацию перечисленных действий на конкретном примере.

Пример

Синтезировать цифровой автомат, заданный в виде таблиц, приведенных на рис. 1.31.

	A_1	A_2	A_3	A_4
z_1	A_3	A_3	A_4	–
z_2	A_3	–	A_2	A_1
z_3	A_2	A_1	A_1	A_3

а

	A_1	A_2	A_3	A_4
z_1	ω_3	ω_2	ω_1	–
z_2	ω_1	–	ω_3	ω_2
z_3	ω_4	ω_1	ω_4	ω_3

б

Рис. 1.31

Память автомата построить на T-триггере.

При синтезе логических выражений использовать базис И, ИЛИ, НЕ.

Решение

Кодирование состояний выполним через набор логических переменных Q. Множество состояний включает четыре элемента. Поэтому для представления каждой из них достаточно использовать комбинации из двух переменных Q_1 и Q_2. Кодировка состояний представлена таблицей на рис. 1.32, а.

Кодирование входных сигналов выполним через набор логических переменных x. Множество входных сигналов включает три элемента. Поэтому для представления каждого из них достаточно использовать комбинации из двух переменных x_1 и x_2. Кодировка входных переменных представлена таблицей на рис. 1.32, б.

Кодирование выходных сигналов выполним через набор логических функций y. Множество выходных сигналов включает четыре элемента. Поэтому для представления каждой из них достаточно использовать комбинации из двух переменных y_1 и y_2. Кодировка выходных переменных представлена таблицей на рис. 1.32, в.

	Q_1	Q_2
A_1	0	1
A_2	1	0
A_3	1	1
A_4	0	0

а

	x_1	x_2
z_1	0	1
z_2	1	0
z_3	1	1

б

	y_1	y_2
ω_1	0	1
ω_2	1	0
ω_3	1	1
ω_4	0	0

в

Рис. 1.32

Таблицы переходов (см. рис. 1.31, а) и выходных сигналов (см. рис. 1.31, б) после замены переменных исходного задания цифрового автомата на их кодированные значения будут иметь вид, представленный на рис. 1.33, а и рис. 1.33, б соответственно.

В таблице переходов на рис. 1.33, а клетки заполнены двухразрядным кодом, первый разряд которого отображает значение переменной Q_1, а второй – пе-

ременной Q_2 (1 – переменная имеет прямое значение, 0 – переменная имеет обратное значение).

В таблице выходов на рис. 1.33, б клетки заполнены двухразрядным кодом, первый разряд которого отображает значение переменной y_1, а второй – переменной y_2 (1 – переменная имеет прямое значение, 0 – переменная имеет обратное значение).

	$\overline{Q}_1 Q_2$	$Q_1\overline{Q}_2$	$Q_1 Q_2$	$\overline{Q}_1\overline{Q}_2$
$\overline{x}_1 x_2$	11	11	00	–
$x_1 \overline{x}_2$	11	–	10	01
$x_1 x_2$	10	01	01	11

а

	$\overline{Q}_1 Q_2$	$Q_1 \overline{Q}_2$	$Q_1 Q_2$	$\overline{Q}_1\overline{Q}_2$
$\overline{x}_1 x_2$	11	10	01	–
$x_1 \overline{x}_2$	01	–	11	10
$x_1 x_2$	00	01	00	11

б

Рис. 1.33

Логические выражения для разрядов кода выходных сигналов составляются на основе таблицы на рис. 1.33, б. Эти выражения имеют следующий вид:

$$y_1 = \overline{Q}_1 Q_2 \overline{x}_1 x_2 + Q_1\overline{Q}_2\overline{x}_1 x_2 + Q_1 Q_2 x_1\overline{x}_2 + \overline{Q}_1\overline{Q}_2 x_1\overline{x}_2 + \overline{Q}_1\overline{Q}_2 x_1 x_2;$$

$$y_2 = \overline{Q}_1 Q_2 \overline{x}_1 x_2 + \overline{Q}_1 Q_2 x_1\overline{x}_2 + Q_1\overline{Q}_2 x_1 x_2 + Q_1 Q_2 \overline{x}_1 x_2 + Q_1 Q_2 x_1\overline{x}_2 + \overline{Q}_1\overline{Q}_2 x_1 x_2.$$

Приведенные выражения составляются следующим образом.

При формировании y_1 в кодированной таблице выходных сигналов на рис. 1.33, б выбираются все случаи, когда y_1 имеет единичное значение, и для каждого из них формируется конъюнкция, отражающая начальное состояние, и входные сигналы, при которых функция y_1 имеет единичное значение. Например, первая конъюнкция в выражении для y_1 имеет вид $\overline{Q}_1 Q_2 \overline{x}_1 x_2$ и соответствует случаю, когда начальное состояние имеет значения $\overline{Q}_1 Q_2$, а на вход поступает сигнал $\overline{x}_1 x_2$. В этой клетке переменная y_1 (первый разряд двухразрядного кода выходного сигнала) имеет единичное значение.

Пятая конъюнкция выражения для y_1 имеет вид $\overline{Q}_1\overline{Q}_2 x_1 x_2$ и соответствует клетке, расположенной на пересечении столбца $\overline{Q}_1\overline{Q}_2$ и строки $x_1 x_2$, где функция y_1, согласно таблице на рис. 1.33, б, имеет единичное значение.

Выражение для y_2 формируется аналогично, но рассматриваются значения второго разряда двухразрядного кода, соответствующего выходному сигналу цифрового автомата.

Логические выражения для функций управления памятью составляются на основе таблицы на рис. 1.33, а с учетом особенностей используемого элемента памяти. В рассматриваемом случае элементом памяти является *T*-триггер, основная особенность которого заключается в том, что по каждому входному сигналу триггер изменяет свое состояние на противоположное. Поэтому сигнал на вход *T*-триггера надо подавать только в тех случаях, когда новое его состояние отличается от текущего. Логические выражения для сигналов управления памятью в рассматриваемом случае имеют вид

$$q_{T1} = \overline{Q}_1 Q_2 \overline{x}_1 x_2 + \overline{Q}_1 Q_2 x_1 \overline{x}_2 + \overline{Q}_1 Q_2 x_1 x_2 + Q_1 \overline{Q}_2 x_1 x_2 + Q_1 Q_2 \overline{x}_1 x_2 + Q_1 Q_2 x_1 x_2 + \overline{Q}_1 \overline{Q}_2 x_1 x_2;$$

$$q_{T2} = \overline{Q}_1 Q_2 x_1 x_2 + Q_1 \overline{Q}_2 \overline{x}_1 x_2 + Q_1 \overline{Q}_2 x_1 x_2 + Q_1 Q_2 \overline{x}_1 x_2 + Q_1 Q_2 x_1 \overline{x}_2 +$$
$$+ \overline{Q}_1 \overline{Q}_2 x_1 \overline{x}_2 + \overline{Q}_1 \overline{Q}_2 x_1 x_2.$$

Приведенные выражения составляются следующим образом.

В выражении для функции сигнала *T*-входа первого разряда q_{T1} в общей дизъюнктивной форме используются конъюнкции, отражающие все случаи, когда значение первого разряда кода нового состояния противоположно его начальному состоянию. Например, первая конъюнкция в выражении для q_{T1} имеет вид $\overline{Q}_1 Q_2 \overline{x}_1 x_2$ и соответствует случаю, когда начальное состояние имеет значения $\overline{Q}_1 Q_2$ (01), и на вход поступает сигнал $\overline{x}_1 x_2$, а код нового состояния $Q_1 Q_2$ имеет в первом разряде значение, отличное от значения первого разряда кода начального состояния.

Пятая конъюнкция выражения для q_{T1} имеет вид $Q_1 Q_2 \overline{x}_1 x_2$ и соответствует клетке, расположенной на пересечении столбца $Q_1 Q_2$ и строки $\overline{x}_1 x_2$ где первый разряд кода нового состояния автомата, согласно таблице на рис. 1.33, а, имеет значение $\overline{Q}_1$, противоположное значению, которое имеет место в первом разряде кода начального состояния цифрового автомата.

Выражение для q_{T2} формируется аналогично, но рассматриваются изменения значения второго разряда двухразрядного кода состояния после учета воздействия входного сигнала.

Для более компактного представления полученных логических выражений и обозначений на формируемой схеме цифрового автомата введем десятичную кодировку конъюнкций, используемых в полученных логических выражениях. Каждая конъюнкция представляет набор одних и тех же переменных Q_1, Q_2, x_1, x_2, поэтому эти конъюнкций можно рассматривать как четырехразрядный двоичный код и кодировать десятичными эквивалентами этого двоичного кода. Следовательно, ранее полученные выражения можно представить в следующей компактной форме:

$y_1 = 5 + 9 + 14 + 2 + 3$; $y_2 = 5 + 6 + 11 + 13 + 14 + 3$;

$q_{T1} = 5 + 6 + 7 + 11 + 13 + 15 + 3$; $q_{T2} = 7 + 9 + 11 + 13 + 14 + 2 + 3$.

Таким образом, множество неповторяющихся конъюнкций в кодированной форме, которые используются во всех сформированных логических выражениях, имеет вид {5, 9, 14, 2, 3, 6, 11, 13, 7,15}.

Логическая схема, реализующая сформированные логические выражения для выходных сигналов и сигналов управления памятью цифрового автомата и осуществляющая кодировку входных и декодировку выходных сигналов, имеет вид (рис. 1.34).

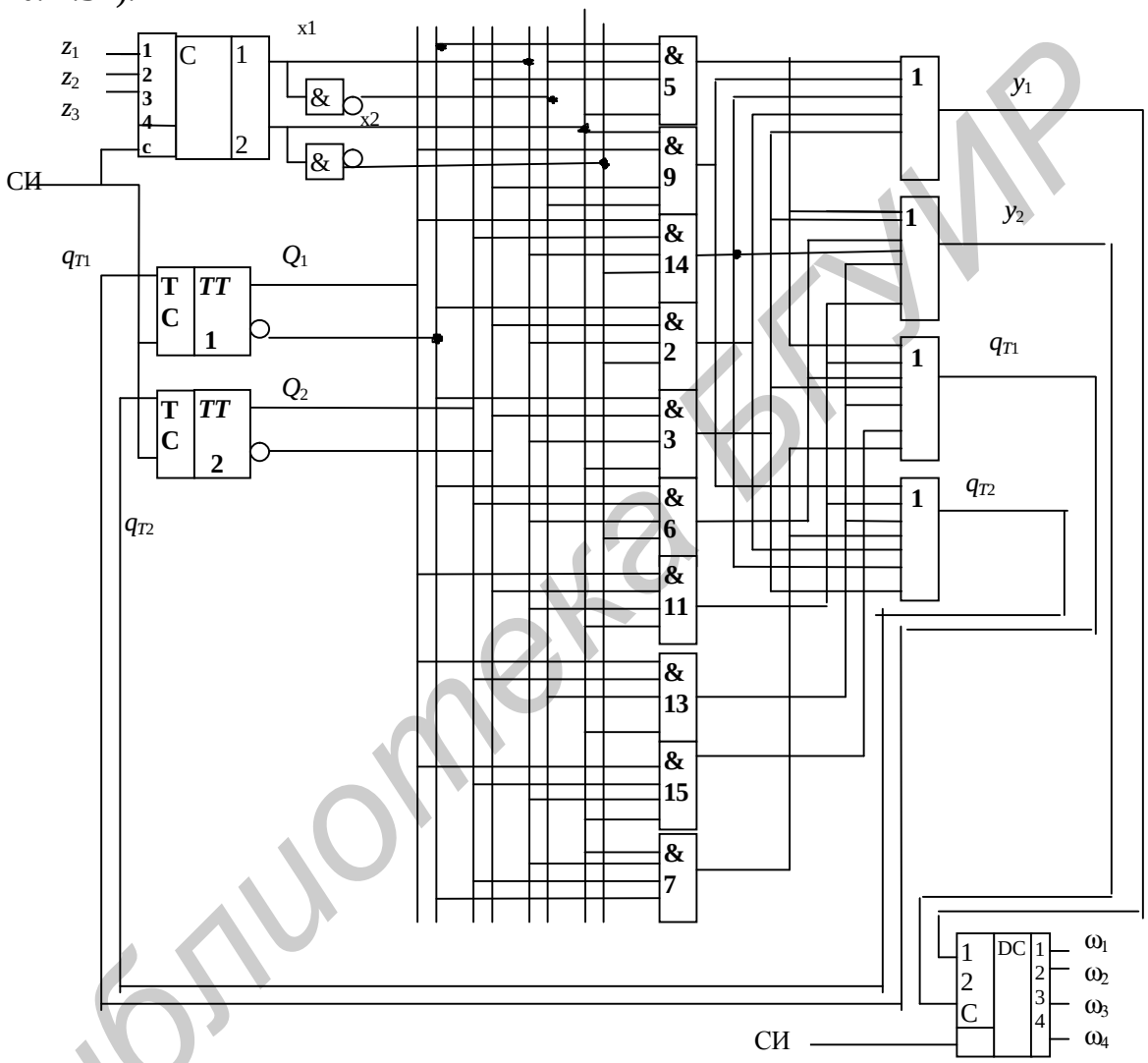


Рис. 1.34

Память цифрового автомата реализована на двух *T*-триггерах, формирующих парафазные выходные сигналы $Q_i, \overline{Q}_i$, используемые для кодировки состояний A_1, A_2, A_3, A_4. Кодер формирует сигналы $x_1, \overline{x}_1; x_2, \overline{x}_2$, кодирующие входные сигналы z_1 , z_2, z_3. Схемы «И» обозначены десятичными числами, соответствующими кодам конъюнкций, которые они формируют. Выходы логических схем «ИЛИ» обозначены логическими функциями, формируемыми этими логиче-

скими схемами. Выходные сигналы цифрового автомата формируются с помощью декодера.

2. УСТРОЙСТВА ЭВМ

2.1. Арифметико-логическое устройство ЭВМ

Арифметико-логическое устройство ЭВМ (АЛУ) служит для выполнения логических и арифметических операций.

Структурная схема АЛУ представлена на рис. 2.1.

АЛУ можно разделить на два блока:

– управляющий блок (Упр. АЛУ);

– операционный блок.

Операционный блок состоит из следующих типовых узлов:

– регистры (*R*), служащие для хранения операндов и результатов;

– сумматор (*SM*), служащий для выполнения операции суммирования многоразрядных кодов;

– операционные узлы (ОУ), служащие для выполнения логических операций;

– мультиплексор (*MS*);

– счетчик (Сч), обеспечивающий подсчет тактов длинных операций;

– регистр флажков (*RF*), служащий для фиксации особой информации, характеризующей полученный результат.

Для передачи информации между отдельными узлами используются шины Ш 1 – Ш 3. Шина Ш 3 обеспечивает также связь с запоминающими устройствами (ЗУ) ЭВМ.

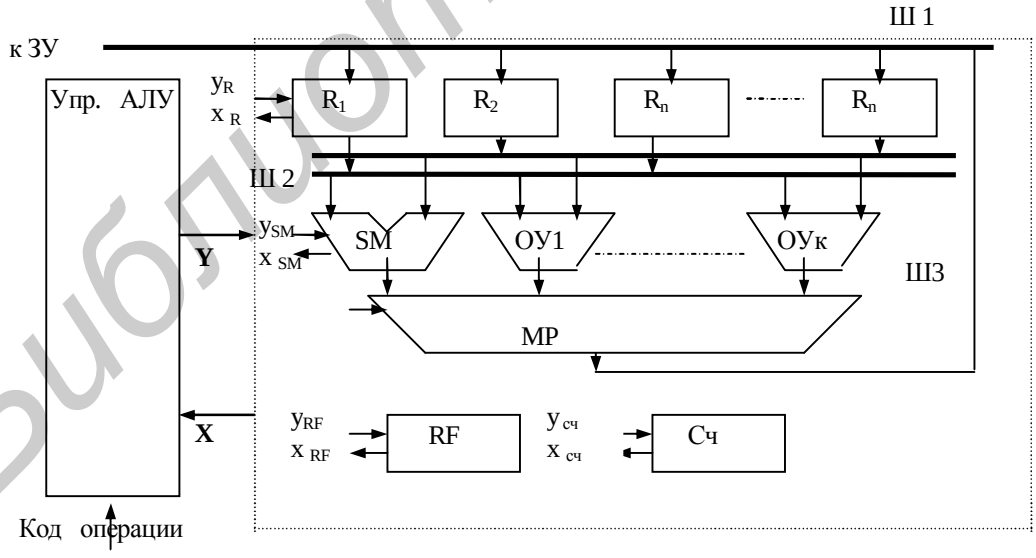


Рис. 2.1

Управляющий блок осуществляет выработку множества управляющих сигналов *Y*, обеспечивающих выполнение элементарных операций (микроопераций) типовыми узлами операционного блока.

При работе управляющая часть АЛУ использует код заданной операции (например сложение, умножение, вычитание и т. п.), а также информацию о состоянии операционного блока, представленную в виде множества X признаков, формируемых типовыми узлами.

К признакам, вырабатываемым регистром и посылаемым в управляющую часть, относятся:

– «ноль регистра» ($R\{0...n\} = 0$) характеризует состояние, при котором во всех разрядах регистра имеет место нулевое значение;

– «ноль знака» ($R\{зн\} = 0$) – в знаковом разряде регистра находится значение 0;

– «единица старшего разряда» ($R\{1\} = 1$) – в старшем разряде регистра находится значение единица;

– «единица младшего разряда» ($R\{n\} = 1$) – в младшем разряде регистра находится значение единица.

К микрооперациям, которые может выполнять регистр при поступлении соответствующего управляющего сигнала y_i, относятся:

– прием кода;

– выдача прямого кода;

– выдача инверсного кода;

– установка единицы в некотором разряде регистра;

– обнуление знакового разряда;

– сдвиг кода влево;

– сдвиг кода вправо;

– обнуление регистра (во все разряды регистра устанавливается нулевое значение).

К признакам, вырабатываемым счетчиком и посылаемым в управляющую часть, относятся:

– «ноль счетчика» («0» Сч) – характеризует состояние, при котором во всех разрядах регистра имеет место нулевое значение;

– «переполнение счетчика» – при поступлении очередного счетного сигнала счетчик переходит от максимального значения к значению «0».

Счетчик может выполнять следующие операции, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– установка нуля в счетчике;

– установка в счетчике некоторого начального значения;

– установка режима счета (обратный или прямой счет);

– изменение находящегося в счетчике текущего значения на единицу.

К признакам, вырабатываемым сумматором и посылаемым в управляющую часть, относятся:

– признак нулевого результата;

– признак единичных значений во всех разрядах результата;

– признак единицы в первом знаковом разряде результата;
– признак единицы во втором знаковом разряде результата;
– признак переноса из старшего разряда сумматора;
– признак наличия в тетраде значения, большего 9;
– признак межтетрадного переноса.

Каждому из перечисленных состояний может соответствовать отдельный разряд (флажок) в регистре флажков.

Сумматор может выполнять следующие микрооперации, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– прием кода двух операндов на свои входы;
– формирование поразрядной суммы операндов, поступающих на его входы;
– генерирование поразрядного переноса;
– распространение переносов через разряды поразрядной суммы, пропускающие перенос;
– прибавление единицы в младший разряд;
– прибавление корректирующих кодов в тетрады при сложении двоично-десятичных кодов.

Выполнение любой арифметической операции в АЛУ реализуется за счет выполнения определенной последовательности микроопераций в узлах операционной части АЛУ. Такие последовательности образуют алгоритм выполнения операций на уровне микроопераций. Удобной формой представления алгоритма выполнения операций является граф-схема алгоритма (ГСА).

2.2. Граф-схема алгоритма выполнения операции

Рассмотрим пример граф-схемы алгоритма для случая выполнения операции деления.

Исходные данные:

– делимое и делитель представлены в двоичном коде как правильная дробь с фиксированной точкой;
– метод деления – без восстановления остатка;
– используется дополнительный код;
– делимое помещается в регистре $R1$, делитель – в $R2$, результат формируется в регистре $R3$; формируемый в начале операции знак частного хранится в регистре $R4$;
– делимое располагается в ЗУ по адресу «$a1$», делитель – «$a2$», результат помещается по адресу «$a1$».

Граф-схема алгоритма приведена на рис. 2.2.

Вершины приведенной ГСА пронумерованы. Вершина 1 обеспечивает приведение узлов операционной части АЛУ в исходное состояние, например установка используемых регистров в нулевое состояние.

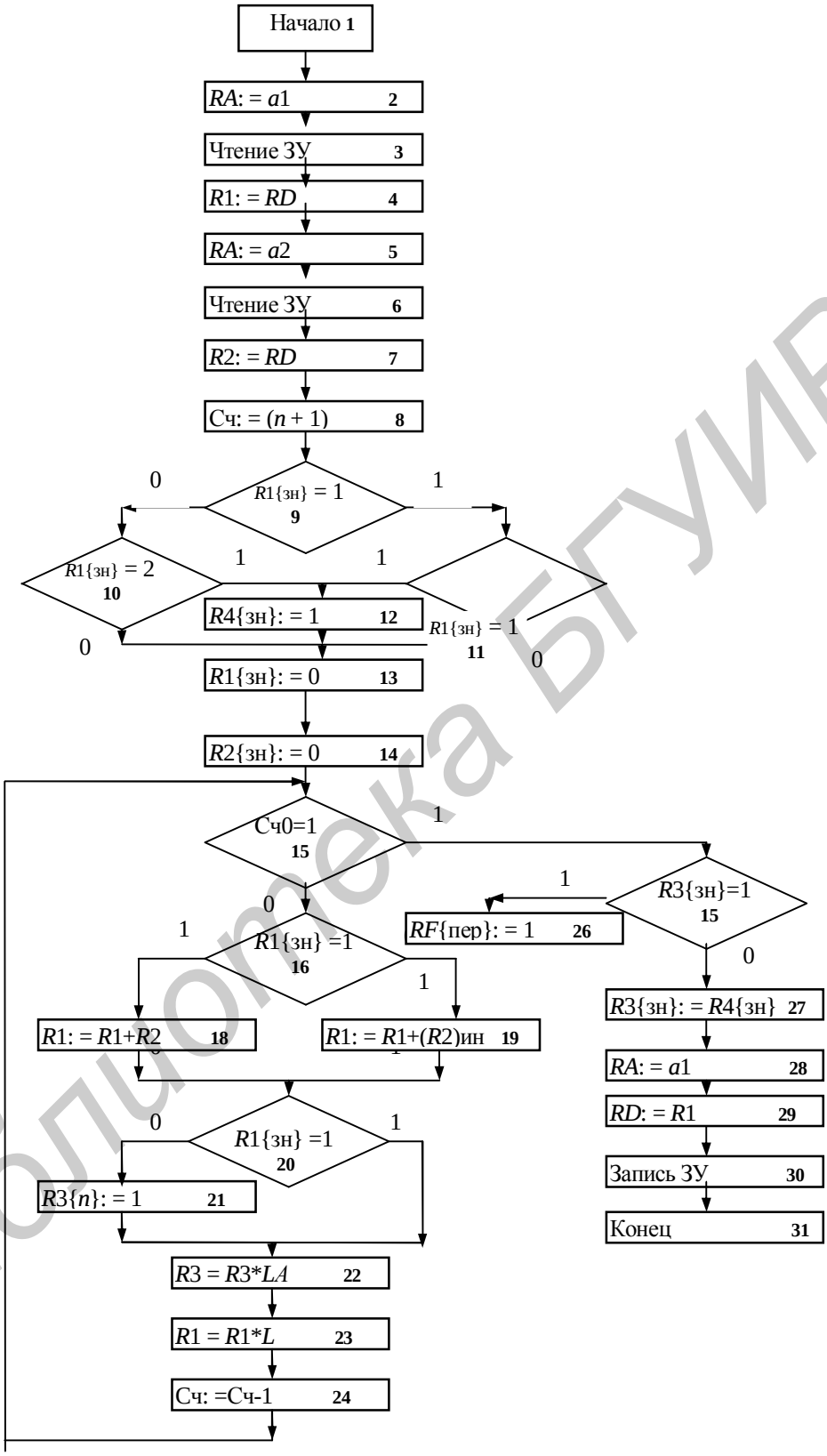


Рис. 2.2

Вершины 2 – 7 обеспечивают чтение из запоминающего устройства по адресам, устанавливаемым в регистре *RA* (регистре адреса ЗУ), делимого и делителя и размещение их в соответствующих регистрах операционной части

АЛУ. *RD* представляет собой регистр данных ЗУ, обеспечивающий кратковременное хранение информации после ее чтения или перед ее записью в ЗУ. Вершина 8 предназначена для записи в счетчик количества тактов выполнения операции деления. Здесь *n* – разрядность поля модуля представления чисел.

Вершины 9 – 14 осуществляют определение знака частного, запись его в знаковый разряд *R*4 и обнуление знаков делимого и делителя для поиска абсолютного значения частного.

Вершины 16 – 24 представляют действия, выполняемые на одном такте:

– определяют необходимость выполнения очередного такта (если в начале такта в счетчике ноль, то это означает, что необходимое количество тактов уже выполнено и осуществляется переход к завершению операции деления);

– в зависимости от знака остатка (на первом такте это знак делимого) осуществляется прибавление к остатку или вычитание из остатка абсолютного значения делителя;

– если знак результата положительный, то в младший разряд *R*3 устанавливается единица;

– выполняется арифметический сдвиг влево полученного результата в *R*1 (вершина 22, где сокращение *LA* означает арифметический сдвиг влево);

– выполняется сдвиг влево формируемого частного в *R*3;

– уменьшается на единицу значение в счетчике.

При завершении операции деления, которое осуществляется по обнаружению нулевого значения в счетчике при реализации вершины 16, выполняются следующие действия:

– если в знаковом разряде *R*3{зн} находится «1», то это означает, что найденное частное имеет ненулевую целую часть и в регистре флагов *RF* в разряд *RF*{пер}, отведенный для фиксации переполнения, устанавливается «1» (действия при обнаружении переполнения зависят от того, являются ли делимое и делитель собственно операндами, представленными числами с фиксированной точкой, или мантиссами операндов, представленных с плавающей точкой; эти действия на ГСА не отражены);

– при отсутствии переполнения знаковому разряду *R*3 приписывается значение, соответствующее значению в знаковом разряде *R*4;

– частное из регистра записывается в память по адресу «*a*1».

Приведенная ГСА деления не предусматривает округления результата.

На рассматриваемой ГСА проверяемые условия и выполняемые микрооперации описаны сокращенно. Такая форма ГСА называется *содержательной*. Она удобна для понимания сущности выполняемых действий в алгоритме. Помимо содержательной, используется также *кодированная* форма представления ГСА, в которой микрооперация и проверяемое условие записаны в виде условных обозначений y_i и x_j соответственно. Кодированная форма, реализующая рассматриваемый алгоритм, удобна при синтезе устройства управления.

На рис. 2.3 приведена кодированная ГСА выполнения операции умножения дробных чисел в форме с фиксированной точкой.

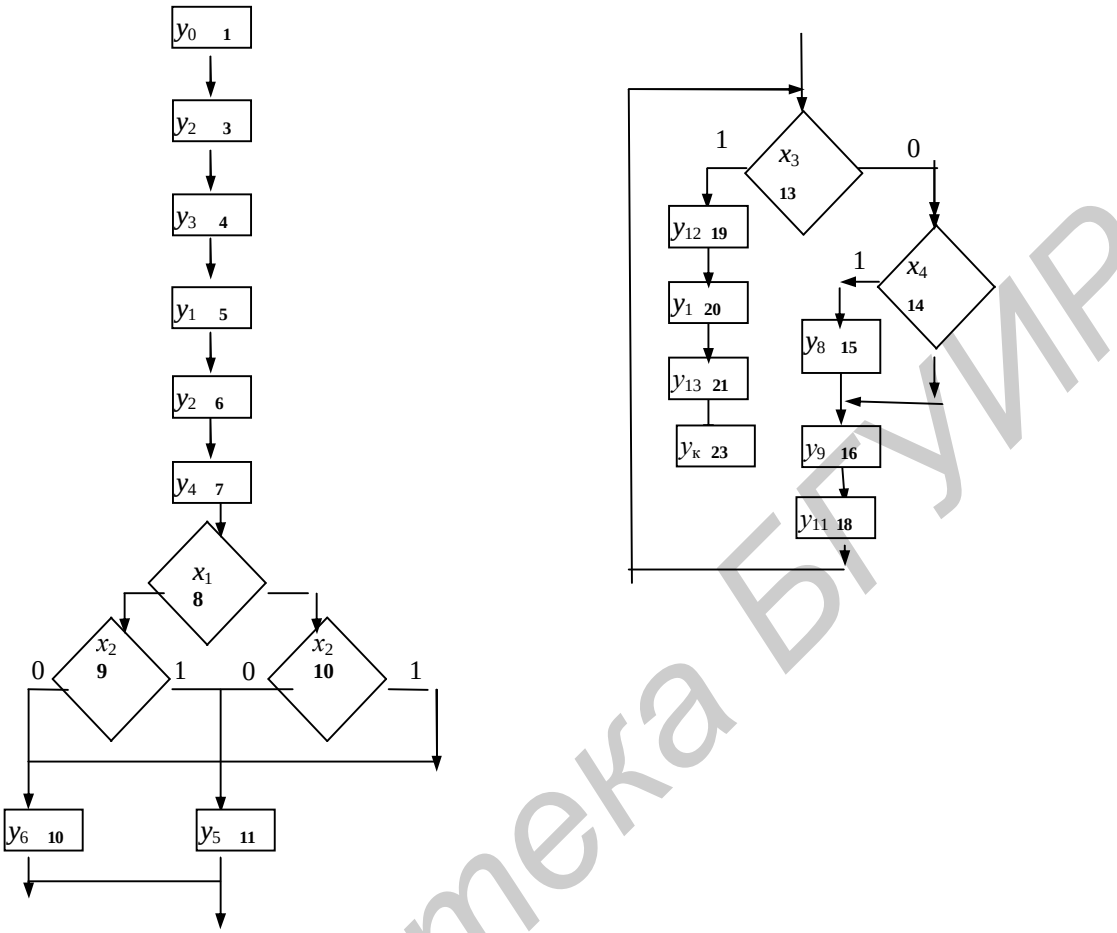


Рис. 2.3

На схеме принята следующая кодировка условий x_i и микроопераций y_j:

– y_0 – обнуление используемых регистров $R1$, $R2$, $R3$, $R4$ и счетчика;

– y_1 – прием кода адреса в *RA* (регистр адреса ЗУ);

– y_2 – чтение в ЗУ;

– y_3 – прием кода из *RD* в $R2$;

– y_4 – прием кода из *RD* в $R3$;

– x_1 – признак знака $R2$;

– x_2 – признак знака $R3$;

– y_5 и y_6 – соответственно установка нуля и единицы в знаковом разряде регистра $R4\{зн\}$;

– y_7 – установка кода «n» (количество разрядов модуля множителя) в счетчике;

– x_3 – признак нуля счетчика;

– y_{12} и y_1 – соответственно запись в $R2\{зн\}$ значения из $R4\{зн\}$ и прием кода адреса в RA;

– x_4 – признак единичного значения младшего разряда регистра R3;

– y_8 – суммирование содержимого $R1$ и $R2$ и запись полученной суммы в $R1$;

– y_9 – сдвиг вправо содержимого регистров $R3$ и $R1$;

– y_{11} – уменьшение содержимого счетчика на «1»;

– y_{13} – запись в ЗУ содержимого $R1$;

– $y_к$ – микрооперация завершения операции умножения.

ГСА построена для случая использования умножения начиная с младших разрядов множителя со сдвигом промежуточного результата. Множимое размещается в $R2$, множитель размещается в регистре $R3$, в регистре $R1$ размещается промежуточный результат, в знаковом разряде регистра $R4$ размещается знак формируемого произведения. Для подсчета тактов используется счетчик. Перед выполнением операции операнды размещаются в памяти. После выполнения операции результат записывается также в память. Микрооперации, выполняемые в различных узлах и не зависящие друг от друга, могут осуществляться одновременно и объединяться в одной операционной вершине. На приведенной кодированной граф-схеме алгоритма такими операциями являются y_0, y_1; y_3, y_1; y_5, y_6, y_7; y_{12} , y_1; y_9, y_{11}.

2.3. Построение блока управления

Устройство управления вообще и блок управления АЛУ в частности могут строиться по принципу микропрограммирования (программируемая логика) или по принципу с жесткой логикой (аппаратный принцип). В любом случае удобной формой задания поведения управляемого объекта является кодированная ГСА.

При использовании микропрограммного принципа выработка необходимой последовательности сигналов управления имеющимся объектом выполняется за счет реализации микропрограммы, разработанной в соответствии с заданной ГСА. При использовании аппаратного принципа блок управления строится в виде цифрового автомата, который вырабатывает последовательность выходных сигналов, используемых для управления объектом.

2.3.1. Аппаратный принцип построения блока управления

Блок управления аппаратного принципа может строиться на базе цифрового автомата Мили или Мура. Наличие большого количества входных сигналов и состояний цифрового автомата, а также значительного числа пар «входной сигнал – состояние» делает затруднительным использование классического подхода к синтезу цифрового автомата, рассмотренного в подразделе «Элементы теории цифровых автоматов». Поэтому в данном случае синтез цифрового автомата, реализующего данный блок управления, осуществляется по несколько отличному принципу и включает следующие этапы:

– построение графа выбранного типа цифрового автомата на основе ГСА функционирования имеющегося объекта;

– составление объединенной кодированной таблицы переходов и выходов цифрового автомата;

– составления логических выражений для сигналов управления памятью и выходных сигналов цифрового автомата;

– синтез логических схем на основании полученных логических выражений в заданном логическом базисе.

Построение блока управления на базе автомата Мура

Рассмотрим построение блока управления на базе автомата Мура для объекта, работа которого задается ГСА, приведенной на рис. 2.4.

Формирование графа автомата Мура, соответствующего ГСА выполнения операции в управляемом объекте, происходит следующим образом:

– объединяются операционные вершины ГСА, имеющие однозначную связь по входу и выходу, при условии, что результат выполнения микрооперации в предыдущей вершине не используется при выполнении микрооперации в последующей вершине;

– устраняются замкнутые пути из одной логической вершины ГСА в другую, минуя операторные вершины, посредством введения в этот путь пустой операторной вершины;

– каждой операторной вершине ГСА ставится в соответствие вершина графа автомата Мура.

Используя указанные правила, ГСА на рис. 2.4 преобразуется в граф автомата Мура, приведенный на рис. 2.5. Вершинами графа A_i являются операционные вершины исходной ГСА. Данный граф формируется из исходной ГСА при условии, что микрооперации $У_4$ и $У_{11}$ допускают одновременное выполнение; через $У_п$ обозначена фиктивная (пустая, т. е. «ничего не делать») микрооперация. Микрооперация $У_0$ выполняется при инициализации цифрового автомата.

Объединенная кодированная таблица переходов и выходов цифрового автомата строится за счет нахождения всех существующих путей из каждой вершины имеющегося графа в ближайшую другую вершину с указанием условий, при которых имеет место данный путь, и вырабатываемых выходных сигналов, которые в автомате Мура однозначно определяются конечным состоянием (конечной вершиной пути):

$$A_i\{x_s^{\sigma s}, x_p^{\sigma p} \dots x_f^{\sigma f}, y_n(A_J), \dots y_m(A_J)\}\ A_J,$$

где A_i, A_J – начальная и конечная вершина пути соответственно; $x_s^{\sigma s}, x_p^{\sigma p} \dots x_f^{\sigma f}$ – условия, через которые проходит путь из A_i в A_J; $y_n(A_J), \dots y_m(A_J)$ – выходные сигналы, однозначно зависящие от состояния A_J.

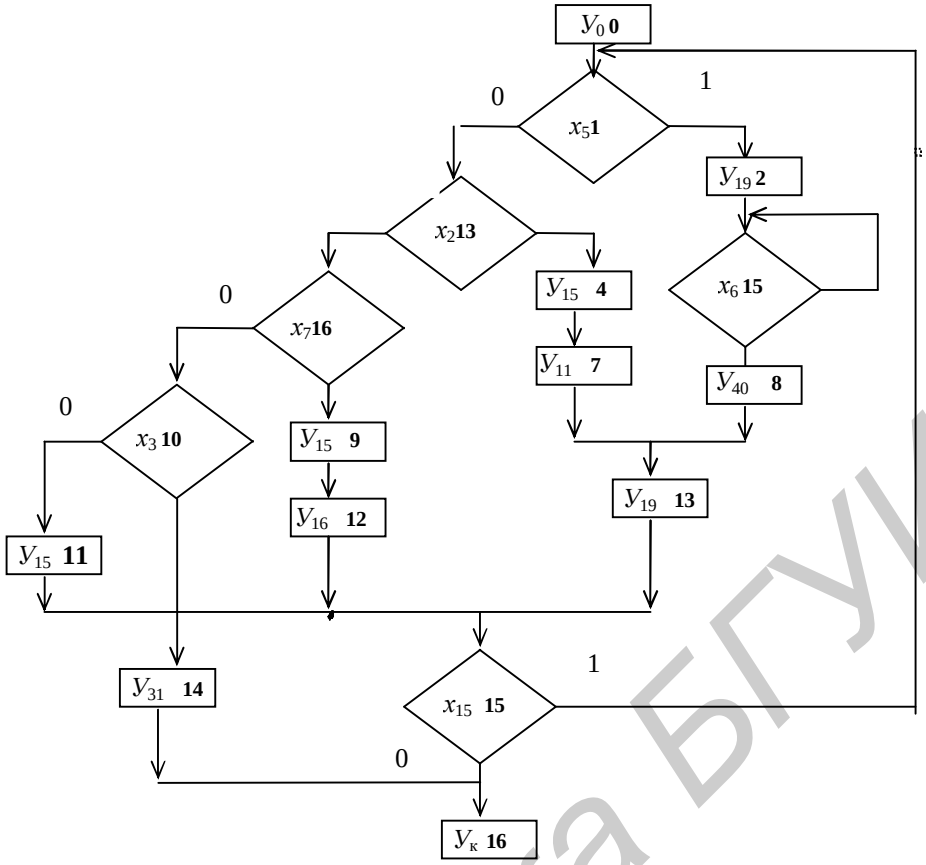


Рис. 2.4

Объединенная кодированная таблица переходов и выходов цифрового автомата составляется на основе всех возможных путей из всех вершин графа автомата. В табл. 2.1 приведена объединенная кодированная таблица переходов и выходов для графа автомата Мура.

При формировании таблицы использовалась кодировка состояний цифрового автомата двоичными эквивалентами их индексов, а разряды четырехбитового кода состояния обозначены как $Q_1Q_2Q_3Q_4$. В качестве элемента памяти использован D-триггер, имеющий вход q_{D1}.

В таблице каждому путю из вершин графа автомата соответствует одна строка, что позволяет кодировать код начала пути (четырехбитовый код в графе 3) и условие (подмножество условий, лежащих на этом пути, в графе 6) единым числом – номером соответствующей строки в таблице.

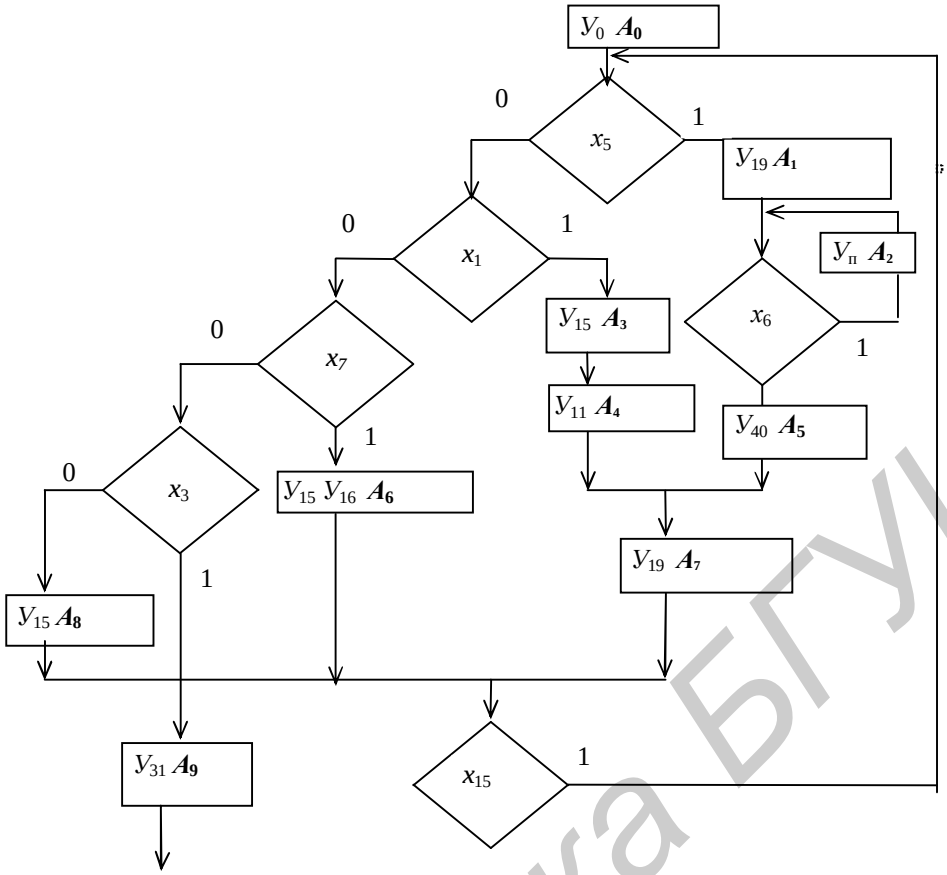


Рис. 2.5

Таблица 2.1

Начало пути		Конец пути		Логическое условие	Выходной сигнал	Управление памятью			
$A_н$	код $A_н$ Q_1 Q_2 Q_3 Q_4	$A_к$	код $A_к$ Q_1 Q_2 Q_3 Q_4			q_{D1}	q_{D2}	q_{D3}	q_{D4}
1	2	3	4	5	6	7	8	9	10
A_0	0000	A_8	1000	$\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7$	Y_{15}	1	0	0	0
		A_9	1001	$\overline{x}_1x_3\overline{x}_5\overline{x}_7$	Y_{31}	1	0	0	1
		A_6	0110	$\overline{x}_1\overline{x}_5x_7$	Y_{15} Y_{16}	0	1	1	0
		A_3	0011	$x_1\overline{x}_5$	Y_{15}	0	0	1	1
		A_1	0001	x_5	Y_{19}	0	0	0	1
A_1	0001	A_5	0101	$\overline{x}_6$	Y_{40}	0	1	0	1
		A_2	0010	x_6	–	0	0	1	0
A_2	0010	A_5	0101	$\overline{x}_6$	Y_{40}	0	1	0	1
		A_2	0010	x_6	–	0	0	1	0
A_3	0011	A_4	0100	1	Y_{11}	0	1	0	0
A_4	0100	A_7	0111	1	Y_{19}	0	1	1	1

Окончание табл. 2.1

1	2	3	4	5	6	7	8	9	10
A_5	0101	A_7	0111	1	y_{19}	0	1	1	1
A_6	0110	A_{10}	1010	$\overline{x}_{15}$	$y_к$	1	0	1	0
		A_8	1000	$\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7x_{15}$	y_{15}	1	0	0	0
		A_9	1001	$\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15}$	y_{31}	1	0	0	1
		A_6	0110	$\overline{x}_1\overline{x}_5x_7x_{15}$	$y_{15}\ y_{16}$	0	1	1	0
		A_3	0011	$x_1\overline{x}_5x_{15}$	y_{15}	0	0	1	1
		A_1	0001	x_5x_{15}	y_{19}	0	0	0	1
A_7	0111	A_{10}	1010	$\overline{x}_{15}$	$y_к$	1	0	1	0
		A_8	1000	$\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7x_{15}$	y_{15}	1	0	0	0
		A_9	1001	$\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15}$	y_{31}	1	0	0	1
		A_6	0110	$\overline{x}_1\overline{x}_5x_7x_{15}$	$y_{15}\ y_{16}$	0	1	1	0
		A_3	0011	$x_1\overline{x}_5x_{15}$	y_{15}	0	0	1	1
		A_1	0001	x_5x_{15}	y_{19}	0	0	0	1
A_8	1000	A_{10}	1010	$\overline{x}_{15}$	$y_к$	1	0	1	0
		A_8	1000	$\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7x_{15}$	y_{15}	1	0	0	0
		A_9	1001	$\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15}$	y_{31}	1	0	0	1
		A_6	0110	$\overline{x}_1\overline{x}_5x_7x_{15}$	$y_{15}\ y_{16}$	0	1	1	0
		A_3	0011	$x_1\overline{x}_5x_{15}$	y_{15}	0	0	1	1
		A_1	0001	x_5x_{15}	y_{19}	0	0	0	1
A_9	1001	A_{10}	1010	1	$y_к$	1	0	1	0

На основании составленной таблицы логические выражения для выходных сигналов и сигналов управления памятью имеют вид:

$$y_{11} = \overline{Q}_1Q_2\overline{Q}_3\overline{Q}_4;$$

$$y_{15} = Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4 + \overline{Q}_1Q_2Q_3\overline{Q}_4 + \overline{Q}_1\overline{Q}_2Q_3Q_4;$$

$$y_{16} = \overline{Q}_1Q_2Q_3\overline{Q}_4;$$

$$y_{19} = \overline{Q}_1Q_2Q_3Q_4 + \overline{Q}_1\overline{Q_2Q_3}Q_4;$$

$$y_{31} = Q_1\overline{Q}_2\overline{Q_3}Q_4;$$

$$y_{40} = \overline{Q}_1Q_2\overline{Q}_3Q_4;$$

$$y_k = Q_1Q_2Q_3Q_4;$$

$$q_{D1} = Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} + Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} + Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} +$$
$$+Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} + Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} + Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} +$$
$$+Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15} + Q_1Q_2Q_3Q_4x_1x_3x_5x_7x_{15};$$
$$q_{D1} = \overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_1\overline{x}_3\overline{x_5}\,\overline{x}_7\overline{x}_{15} + \overline{Q_1}\overline{Q}_2\overline{Q_3}\overline{Q}_4\overline{x}_1x_3\overline{x_5}\,\overline{x_7} + \overline{Q_1}Q_2Q_3\overline{Q_4}\overline{x}_{15} +$$
$$+\overline{Q}_1Q_2Q_3\overline{Q_4}\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7x_{15} + \overline{Q}_1Q_2Q_3\overline{Q}_4\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15} + Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_{15} +$$
$$+Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7x_{15} + Q_1\overline{Q}_2\overline{Q}_3\overline{Q_4}\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15} +$$
$$+\overline{Q}_1Q_2Q_3Q_4\overline{x}_{15} + \overline{Q_1}Q_2Q_3Q_4\overline{x}_1\overline{x_3}\,\overline{x_5}\,\overline{x}_7x_{15} + \overline{Q}_1Q_2Q_3Q_4\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15} +$$
$$+Q_1\overline{Q}_2\overline{Q}_3Q_4\ (1,\ 2,\ 13,\ 14,\ 15,\ 19,\ 20,\ 21,\ 25,\ 26,\ 27,\ 31);$$
$$q_{D2} = \overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_1\overline{x}_5x_7 + \overline{Q}_1\overline{Q}_2\overline{Q}_3Q_4\overline{x}_6 + \overline{Q}_1\overline{Q_2}Q_3\overline{Q_4}\overline{x}_6 + \overline{Q}_1\overline{Q}_2Q_3Q_4 + \overline{Q}_1Q_2\overline{Q_3}\,\overline{Q}_4 +$$
$$+\overline{Q}_1Q_2\overline{Q_3}Q_4 + \overline{Q}_1Q_2Q_3\overline{Q}_4\overline{x}_1\overline{x}_5x_7x_{15} + \overline{Q_1}Q_2Q_3Q_4\overline{x}_1\overline{x}_5x_7x_{15} + Q_1\overline{Q}_2\overline{Q_3}\,\overline{Q}_4\overline{x}_1\overline{x}_5x_7x_{15}$$
$$(3,\ 4,\ 7,\ 9,\ 11,\ 12,\ 16,\ 22,\ 23,\ 25,\ 28,\ 29,\ 13,\ 31);$$
$$q_{D3} = \overline{Q_1}\overline{Q_2Q_3}\,\overline{Q}_4\overline{x}_1\overline{x}_5x_7 + \overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4x_1\overline{x}_5 + \overline{Q}_1\overline{Q_2}\,\overline{Q_3}Q_4\overline{x}_6 + +\overline{Q}_1\overline{Q}_2Q_3\overline{Q}_4x_6 +$$
$$+\overline{Q}_1Q_2\overline{Q}_3\overline{Q}_4 + \overline{Q}_1Q_2\overline{Q}_3Q_4 + \overline{Q_1}Q_2Q_3\overline{Q}_4\overline{x}_{15} + \overline{Q_1}Q_2Q_3\overline{Q}_4\overline{x}_1\overline{x}_5x_7x_{15} +$$
$$+\overline{Q}_1Q_2Q_3Q_4x_1\overline{x}_5x_{15} + Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_{15} +$$
$$+Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_1\overline{x}_5x_7x_{15} + Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4x_1\overline{x}_5x_{15} + \overline{Q}_1Q_2Q_3\overline{Q}_4 +$$
$$+Q\ \overline{Q}_2\overline{Q}_3Q_4\,(3,\ 4,\ 7,\ 9,\ 11,\ 12,\ 16,\ 17,\ 19,\ 22,\ 23,\ 25,\ 28,\ 29,\ 13,\ 31);$$
$$q_{D4} = \overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_1\overline{x}_5x_7 + \overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4x_1\overline{x}_5 + \overline{Q}_1\overline{Q}_2\overline{Q}_3Q_4\overline{x}_6 + \overline{Q}_1\overline{Q}_2Q_3\overline{Q}_4x_6 +$$
$$+\overline{Q}_1\overline{Q}_2Q_3\overline{Q}_4\overline{x}_6 + \overline{Q}_1Q_2\overline{Q}_3\overline{Q}_4 + \overline{Q}_1Q_2\overline{Q}_3Q_4 + \overline{Q}_1Q_2Q_3\overline{Q}_4\overline{x}_1\overline{x}_5\overline{x}_7x_{15} +$$
$$+\overline{Q}_1Q_2Q_3\overline{Q}_4x_1\overline{x}_5x_{15} + \overline{Q}_1Q_2Q_3\overline{Q}_4x_5x_{15} +$$
$$+\overline{Q}_1Q_2Q_3Q_4\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15} + \overline{Q}_1Q_2Q_3Q_4x_1\overline{x}_5x_{15} + \overline{Q}_1Q_2Q_3Q_4x_5x_{15} +$$
$$+Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15} + Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4x_1\overline{x}_5x_{15} + Q_1\overline{Q}_2\overline{Q}_3\overline{Q}_4x_5x_{15}$$
$$(2,\ 4,\ 5,\ 6,\ 8,\ 11,\ 12,\ 15,\ 17,\ 18,\ 21,\ 23,\ 24,\ 27,\ 29,\ 30).$$

После записи дизъюнктивной логической функции для сигналов управления разрядами памяти в скобках приведен перечень кодов используемых в этом выражении конъюнкций (номеров строк).

Построение логической схемы цифрового автомата на базе ПЛМ будет рассмотрено на примере построения автомата Мили.

Построение блока управления на базе автомата Мили

Рассмотрим построение блока управления на базе автомата Мили для объекта, работа которого описывается ГСА, приведенной на рис. 2.4.

Формирование графа автомата Мили, соответствующего ГСА выполнения операции в управляемом объекте, осуществляется следующим образом:

– объединяются операционные вершины ГСА, имеющие однозначную связь по входу и выходу, при условии, что результат выполнения микрооперации в предыдущей вершине не используется при выполнении микрооперации в последующей вершине;

– устраняются замкнутые пути из одной логической вершины ГСА в другую логическую вершину, минуя операторные вершины, посредством введения в этот путь пустой операторной вершины;

– во множество вершин графа автомата Мили включают начальную и конечную вершины ГСА;

– кроме того, в качестве вершин графа автомата рассматриваются выходы операционных вершин ГСА (если выходы операционных вершин сходятся, то они рассматриваются как одна вершина графа цифрового автомата).

С учетом указанных правил ГСА на рис. 2.4 преобразуется в граф автомата Мили, приведенный на рис. 2.6.

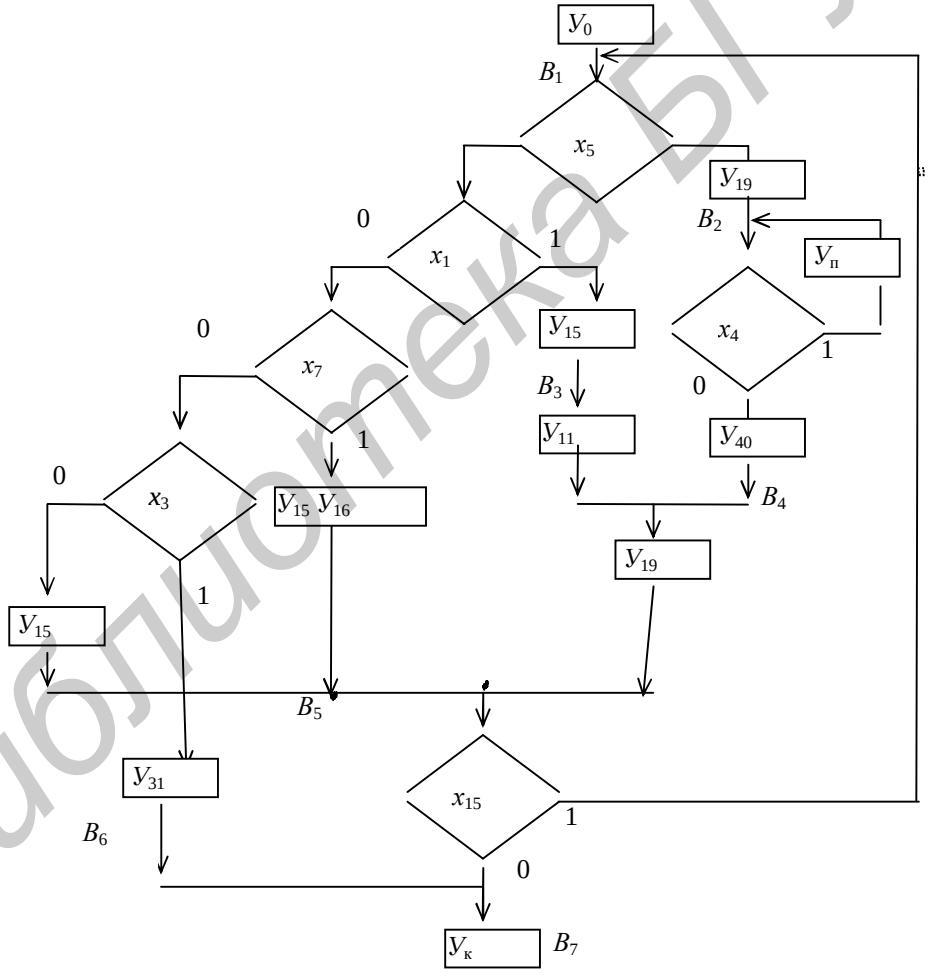


Рис. 2.6

Данный граф формируется из исходной ГСА при условии, что допускается одновременное выполнение микроопераций Y_4 и Y_{11}; через $Y_п$ обозначена фиктив-

ная вершина, содержащая пустую микрооперацию. Микрооперация Y_0 выполняется при инициализации цифрового автомата.

На основании графа автомата Мили объединенная кодированная таблица переходов и выходов цифрового автомата строится в результате нахождения всех существующих путей из каждой вершины графа в другую вершину с указанием условий, при которых имеет место данный путь. Кроме того, для всех путей находятся выходные сигналы, которые определяются микрооперациями, указанными в операционной вершине, через которую проходит данный путь:

$$A_i\{x_s^{\sigma s}, x_p^{\sigma p} \ldots x_f^{\sigma f}, y_n, \ldots y_n\}A_J,$$

где A_i, A_J – начальная и конечная вершина пути соответственно;

$x_s^{\sigma s}, x_p^{\sigma p} \ldots x_f^{\sigma f}$ – условия, через которые проходит рассматриваемый путь из одной вершины графа автомата Мура в другую;

$y_n, \ldots, y_n$ – выходные сигналы автомата, определяемые операционной вершиной, через которую проходит путь.

Например, для вершины B_2 имеют место два пути $B_2\{\overline{x}_6, Y_{40}\}B_4$; $B_2\{x_6, Y_{\text{п}}\}B_2$.

В табл. 2.2 приведена объединенная кодированная таблица переходов, в которой отражены все существующие пути рассматриваемого автомата Мили.

Таблица 2.2

Начало пути		Конец пути		Логическое условие	Выходной сигнал	Управление памятью		
$B(t)$	код $B(t)$	$B(t+1)$	код $A(t+1)$			q_{D1}	q_{D2}	q_{D3}
1	2	3	4	5	6	7	8	9
B_1	001	B_5	101	$\overline{x}_1x_3\overline{x}_5\overline{x}_7$	Y_{15}	1	0	1
		B_6	110	$\overline{x}_1x_3\overline{x}_5\overline{x}_7$	Y_{31}	1	1	0
		B_5	101	$\overline{x}_1\overline{x}_5x_7$	$Y_{15}\ Y_{16}$	1	0	1
		B_3	011	x_1x_5	Y_{15}	0	1	1
		B_2	010	x_5	Y_{19}	0	1	0
B_2	010	B_4	100	$\overline{x}_6$	Y_{40}	1	0	0
		B_2	100	x_6	–	0	1	0
B_3	011	B_4	100	1	Y_{11}	1	0	0
B_4	100	B_5	101	1	Y_{19}	1	0	1
B_5	101	B_7	111	$\overline{x}_{15}$	$Y_{\text{к}}$	1	1	1
		B_5	101	$\overline{x}_1\overline{x}_3\overline{x}_5\overline{x}_7x_{15}$	Y_{15}	1	0	1
		B_6	110	$\overline{x}_1x_3\overline{x}_5\overline{x}_7x_{15}$	Y_{31}	1	1	0
		B_5	101	$\overline{x}_1\overline{x}_5x_7x_{15}$	$Y_{15}\ Y_{16}$	1	0	1
		B_3	011	$x_1\overline{x}_5x_{15}$	Y_{15}	0	1	1
		B_2	010	x_5x_{15}	Y_{19}	0	1	0
B_6	110	B_7	111	1	$Y_{\text{к}}$	1	1	1

При формировании этой таблицы использовалась кодировка состояний цифрового автомата двоичными эквивалентами их индексов. В качестве элемента памяти использован *D*-триггер.

На основании составленной таблицы записываются логические выражения для выходных сигналов и сигналов управления памятью.

$$y_{11} = \overline{Q}_1 Q_2 Q_3 (8);$$

$$y_{15} = \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_3 \overline{x}_5 \overline{x}_7 + \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_5 x_7 + \overline{Q}_1 \overline{Q}_2 Q_3 x_1 \overline{\overline{x}}_5 + \overline{Q}_1 Q_2 \overline{Q}_3 \overline{x}_1 \overline{x}_3 \overline{x}_5 \overline{x}_7 x_{15} +$$
$$+ \overline{Q}_1 Q_2 \overline{Q}_3 \overline{x}_1 \overline{x}_5 x_7 x_{15} + \overline{Q}_1 Q_2 \overline{Q}_3 x_1 \overline{x}_5 x_{15} (1,\ 3,\ 4,\ 11,\ 13,\ 14);$$

$$y_{16} = \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_5 x_7 + \overline{Q}_1 Q_2 \overline{Q}_3 \overline{x}_1 \overline{x}_5 x_7 x_{15} (3,\ 13);$$

$$y_{19} = \overline{Q}_1 Q_2 Q_3 x_5 + Q_1 Q_2 Q_3 + Q_1 Q_2 Q_3 x_5 x_{15} (5,\ 9,\ 15);$$

$$y_{31} = \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 x_3 x_5 x_7 + Q_1 \overline{Q}_2 Q_3 \overline{x}_1 x_3 \overline{x}_5 \overline{x}_7 x_{15} (2,\ 12)$$

$$y_{40} = \overline{Q}_1 Q_2 \overline{Q}_3 x_6 (6);$$

$$y_k = Q_1 \overline{Q}_2 Q_3 \overline{x}_{15} + Q_1 Q_2 \overline{Q}_3 (10,\ 16);$$

$$q_{D1} = \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_3 \overline{x}_5 \overline{x}_7 + \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 x_3 \overline{x}_5 \overline{x}_7 + \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_3 x_7 + \overline{Q}_1 Q_2 \overline{Q}_3 \overline{x}_6 +$$
$$+ \overline{Q}_1 Q_2 Q_3 + Q_1 \overline{Q}_2 \overline{Q}_3 + Q_1 \overline{\overline{Q}}_2 Q_3 \overline{x}_{15} + Q_1 \overline{\overline{Q}}_2 Q_3 \overline{x}_1 \overline{x}_3 \overline{x}_5 \overline{x}_7 x_{15} + Q_1 Q_2 Q_3 \overline{x}_1 x_3 \overline{x}_5 \overline{x}_7 x_{15} +$$
$$+ Q_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_5 x_7 x_{15} + Q_1 Q_2 \overline{Q}_3 (1,\ 2,\ 3,\ 6,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 16);$$

$$q_{D2} = \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 x_3 \overline{x}_5 \overline{x}_7 + \overline{Q}_1 \overline{Q}_2 Q_3 x_1 \overline{x}_5 + \overline{Q}_1 \overline{Q}_2 Q_3 x_5 + \overline{Q}_1 Q_2 \overline{Q}_3 x_4 + Q_1 \overline{Q}_2 Q_3 \overline{x}_{15} +$$
$$+ Q_1 \overline{\overline{Q}}_2 Q_3 \overline{x}_{15} + Q_1 \overline{\overline{Q}}_2 Q_3 \overline{x}_1 x_3 \overline{x}_5 \overline{x}_7 x_{15} + Q_1 \overline{Q}_2 Q_3 x_1 \overline{x}_5 x_{15} + Q_1 \overline{Q}_2 Q_3 x_5 x_{15} + Q_1 Q_2 \overline{Q}_3$$
$$(2,\ 4,\ 5,\ 7,\ 10,\ 12,\ 14,\ 15,\ 16);$$

$$q_{D3} = \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_3 \overline{x}_5 \overline{x}_7 + \overline{Q}_1 \overline{Q}_2 Q_3 \overline{x}_1 \overline{x}_5 x_7 + \overline{Q}_1 \overline{Q}_2 Q_3 x_1 \overline{x}_5 + Q_1 \overline{Q}_2 \overline{Q}_3 + \overline{Q}_1 Q_2 Q_3 +$$
$$+ Q_1 \overline{\overline{Q}}_2 Q_3 \overline{x}_{15} + Q_1 \overline{\overline{Q}}_2 Q_3 \overline{x}_1 \overline{x}_3 \overline{x}_5 \overline{x}_7 x_{15} + Q_1 Q_2 Q_3 \overline{x}_1 \overline{x}_5 x_7 x_{15} + Q_1 \overline{Q}_2 Q_3 x_1 \overline{x}_5 x_{15} +$$
$$+ Q_1 Q_2 \overline{Q}_3 (1,\ 3,\ 4,\ 9,\ 10,\ 11,\ 13,\ 14,\ 16).$$

После записи дизъюнктивной логической функций для выходных сигналов и сигналов управления разрядами памяти в скобках приведен перечень кодов используемых в этом выражении конъюнкций. В качестве этих кодов использованы номера строк таблицы, в которых отражается соответствующий путь.

На рис. 2.7 приведена логическая схема, реализующая цифровой автомат, заданный графом на рис. 2.6

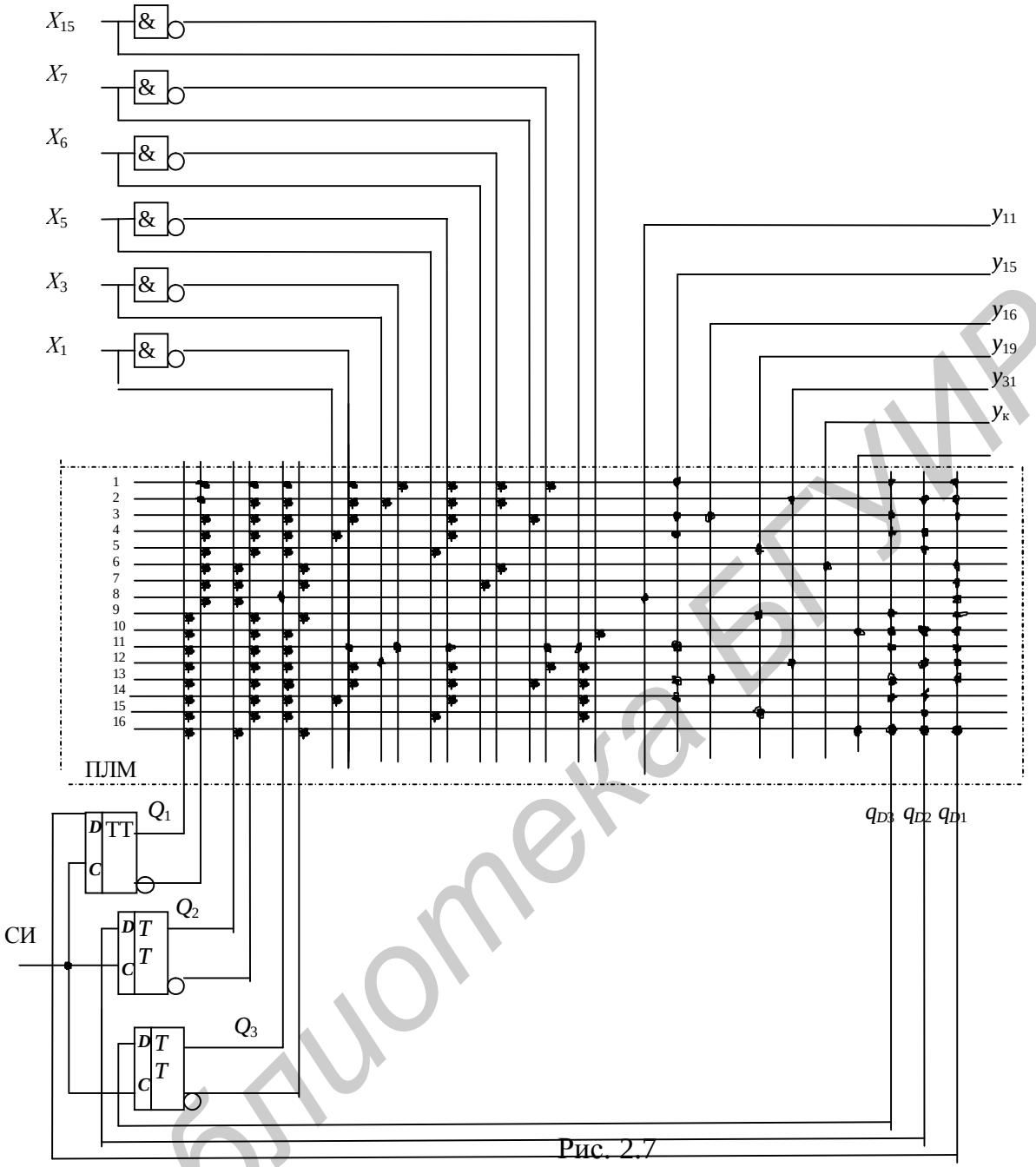


Рис. 2.7

На вход схемы поступают проверяемые условия x_1, x_3, x_5, x_6, x_7, x_{15}. Обратные значения этих условий формируются с помощью шести схем НЕ.

На схеме каждый выход конъюнктивной части ПЛМ (горизонтальные линии) помечен кодом конъюнкции (номером строки в таблице), формируемой на этом выходе. Выходом схемы является множество сигналов микроопераций y_{11}, y_{15}, y_{16}, y_{31}, y_{40}, $y_к$.

Выбор типа цифрового автомата (Мили или Мура) зависит от реализуемой ГСА. В тех случаях, когда имеет место много сходящихся ветвей вычислительного процесса, предпочтение следует отдавать автомату Мили, так как у него будет меньше вершин-состояний, чем у эквивалентного автомата Мура (сравните

табл. 2.1 и табл. 2.2). В тех случаях, когда в ГСА имеет место большое количество длинных линейных участков, предпочтение следует отдавать автомату Мура, так как у него проще логика формирования выходных сигналов.

2.3.2. Микропрограммный принцип построения блока управления

При микропрограммном принципе построения блока управления алгоритм выполнения операции осуществляется за счет особой программы, состоящей из отдельных команд, реализующих требуемую последовательность микроопераций. Такие команды получили название «микрокоманда», а совокупность микрокоманд – микропрограмма. Микропрограмма хранится в памяти ЭВМ.

При представлении алгоритма операции в виде ГСА выполняемая микрокоманда должна обеспечить выработку сигналов соответствующих микроопераций и переход к следующей микрокоманде, в том числе и при ветвлении вычислительного процесса. Таким образом, в формате микрокоманды, в принципе, необходимо иметь несколько полей:

– поле микроопераций (*У*), используемое для задания одной или нескольких микроопераций;

– поле условий (*X*), в котором задаются проверяемые условия, влияющие на ветвление вычислительного процесса;

– поле адреса (*А*), в котором необходимо задавать информацию, определяющую следующую микрокоманду при возможном ветвлении (по крайней мере по двум направлениям) для продолжения вычислительного процесса.

Задание всей перечисленной информации в едином формате микрокоманды затруднительно. Как правило, используют два вида, а следовательно, и два формата микрокоманд: операционные и перехода.

Операционная микрокоманда используется для реализации операторных вершин ГСА. Ее основная задача – задание выполняемой микрооперации. Формат операционной микрокоманды приведен на рис. 2.8, а. Микрокоманда включает два поля: поле типа микрокоманды (*Т*) и поле микрооперации (*У*).

Поле типа микрокоманды должно идентифицировать один из возможных типов микрокоманд. Так как используется всего два типа микрокоманды, то длина этого поля составляет один разряд.

Поле микрооперации задает, как правило, в кодированной форме подлежащую выполнению микрооперацию; его разрядность определяется множеством всех микроопераций, которые могут выполняться в управляемом устройстве. Если допустимая длина микрокоманды достаточно велика, то в одной операционной микрокоманде может задаваться более одной микрооперации (рис. 2.8, б).

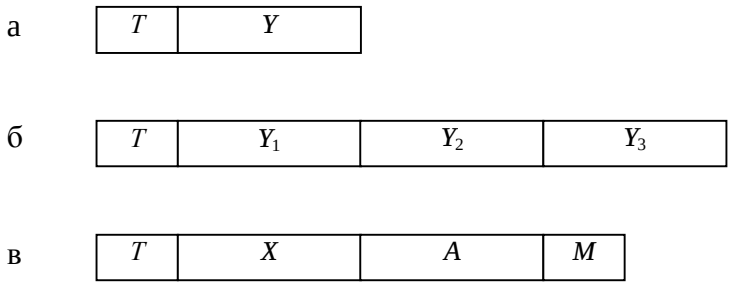


Рис. 2.8

После выполнения операционной микрокоманды выбирается микрокоманда, расположенная в следующем адресе ЗУ (после адреса расположения текущей микрокоманды).

Микрокоманда перехода в основном используется для организации ветвления на основании результата проверки некоторого условия (признака). Поэтому в ней необходимо задавать код проверяемого условия и информацию об адресах двух возможных ветвей продолжения выполнения алгоритма. Возможный формат микрокоманды перехода приведен на рис. 2.8, в. Приведенный формат включает следующие поля:

- поле типа микрокоманды (*T*);
- поле условия (*X*);
- поле адреса (*A*);
- поле модификатора дисциплины перехода (*M*).

Поле типа микрокоманды аналогично одноименному полю операционной микрокоманды.

Поле условия используется для задания кода условия, которое необходимо проверить при реализации данной микрокоманды. Его длина определяется общим количеством условий.

Поле адреса используется для задания местоположения в памяти адресов первых микрокоманд двух возможных ветвей продолжения процесса. При этом в качестве начальной микрокоманды одной из возможных ветвей продолжения используется микрокоманда, расположенная по адресу, следующему в памяти за адресом текущей выполняемой микрокоманды (A_T), а адрес начальной микрокоманды другой ветви задается в самой микрокоманде. Возможны две дисциплины перехода к следующей (A_c) микрокоманде по результату проверки заданного условия:

1) $A_c = \begin{cases} A_T + 1, & если\ x_i = 1, \\ A, & если\ x_i = 0; \end{cases}$

2) $A_c = \begin{cases} A_T + 1, & если\ x_i = 0, \\ A, & если\ x_i = 1; \end{cases}$

где A_T – адрес текущей выполняемой команды.

При реализации алгоритма в одних случаях удобнее использовать первую, в других вторую дисциплину перехода. Наличие модификатора дисциплины перехода (поле *M*) позволяет в текущей команде перехода использовать дисциплину, наиболее удобную для организации данного ветвления. Однако это поле необязательно.

На рис. 2.9 приведена структурная схема устройства управления, построенного на микропрограммном принципе. Она включает:

– запоминающее устройство: хранятся все микропрограммы управления объектом, на который посылается множество сигналов запуска микроопераций $Y = \{y_1, y_2, ..., y_m\}$;

– $X = \{x_1, x_2, ..., x_n\}$ – множество признаков, вырабатываемых управляемым объектом;

– регистр адреса: задается адрес микрокоманды, которую нужно прочитать и выполнить;

– формирователь начального адреса (ФНА) микропрограммы, выполненной при реализации операции в управляемом объекте, код которой (коп) поступает на вход ФНА;

– регистр микрокоманды, в котором фиксируется код микрокоманды, прочитанной из запоминающего устройства (текущая выполняемая микрокоманда);

– адрес перехода;

– дешифратор микрооперации *DC*1;

– дешифратор кода проверяемого устройства *DC*2;

– СИ – синхроимпульсы, по которым выполняются микрокоманды.

При единичном значении в нулевом разряде (операционная микрокоманда) в разрядах с 1-го по *k*-й регистра микрокоманды находится код команды, подлежащей выполнению. При нулевом значении в нулевом разряде (микрокоманда перехода) в разрядах с 1-го по *p*-й регистра команды находится код проверяемого условия, в разрядах от (*p* + 1)-го до *k*-го располагается адрес следующей микрокоманды.

*DC*1 осуществляет декодировку разрядов (1 – *k*) текущей микрокоманды (если это операционная команда). В данном случае на выходе элемента И(1) по сигналу СИ будет разрешена работа *DC*1.

*DC*2 осуществляет декодировку кода проверяемого условия, расположенного в разрядах (1 – *p*) текущей микрокоманды (если это команда перехода). В этом случае на выходе элемента И(2) по сигналу СИ будет выработан сигнал, разрешающий работу *DC*2. Логический элемент И(4) вырабатывает сигнал, поступающий на вход «+1» регистра адреса, который увеличивает значение, содержащееся в нем, на «+1». Сигнал «+1» вырабатывается по сигналу СИ тогда, когда имеет место единица в нулевом разряде регистра микрокоманды или имеет место «1» на выходе логического элемента И-ИЛИ 7.

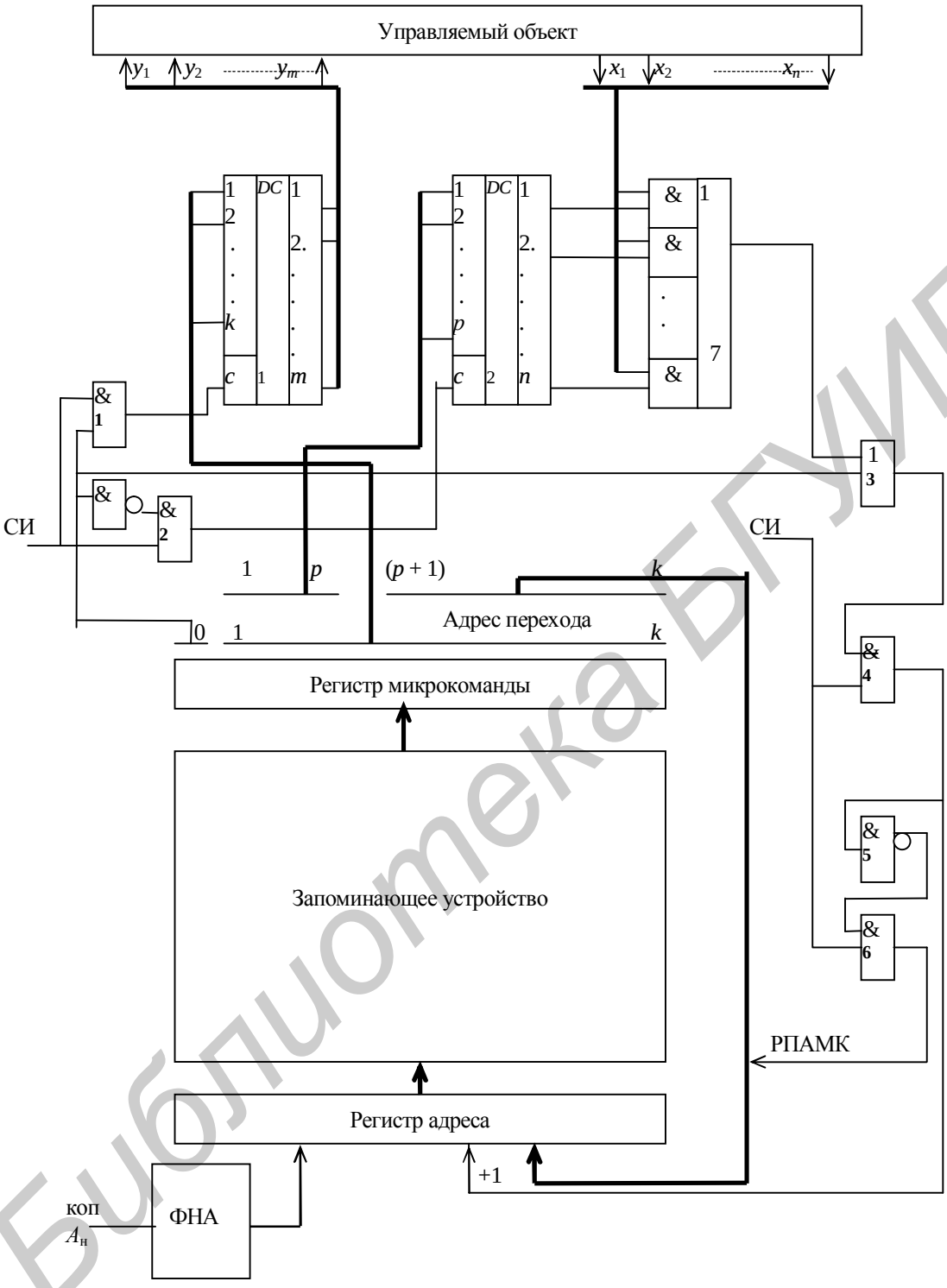


Рис. 2.9

Если сигнал «+1» не вырабатывается, то по сигналу СИ элемент И(4) вырабатает сигнал разрешения передачи адреса из микрокоманды (РПАМК), обеспечивающий передачу в регистр адреса кода поля *A* из регистра микрокоманды (разряды от (*p* + 1)-го до *k*-го).

Элемент И-ИЛИ 7 по числу проверяемых условий имеет n двухвходовых схем И, вырабатываемых в управляемом объекте. На первый вход каждого элемента И подается сигнал, значение которого соответствует одному из проверяемых признаков, а на второй вход – соответствующий выход $DC2$. Таким образом, на выходе элемента И-ИЛИ будет иметь место сигнал «1» только тогда, когда проверяемый признак имеет единичное значение, т. е. когда выполняется микрокоманда перехода, в разрядах $(1 – p)$ которой присутствует код i, и управляемый объект на своем выходе x_i имеет единичное значение.

Схема устройства микропрограммного управления (см. рис. 2.9) приведена для случая, когда операционная микрокоманда и команда перехода имеют одинаковую длину; в каждом адресе запоминающего устройства полностью помещается одна микрокоманда, в операционной микрокоманде задается для выполнения только одна микрооперация. В микрокоманде перехода не используется поле M (поле модификатора дисциплины перехода).

Формирование микропрограммы по заданной ГСА рассмотрим на конкретном примере.

Пример: составить микропрограмму для ГСА, приведенной на рис. 2.10.

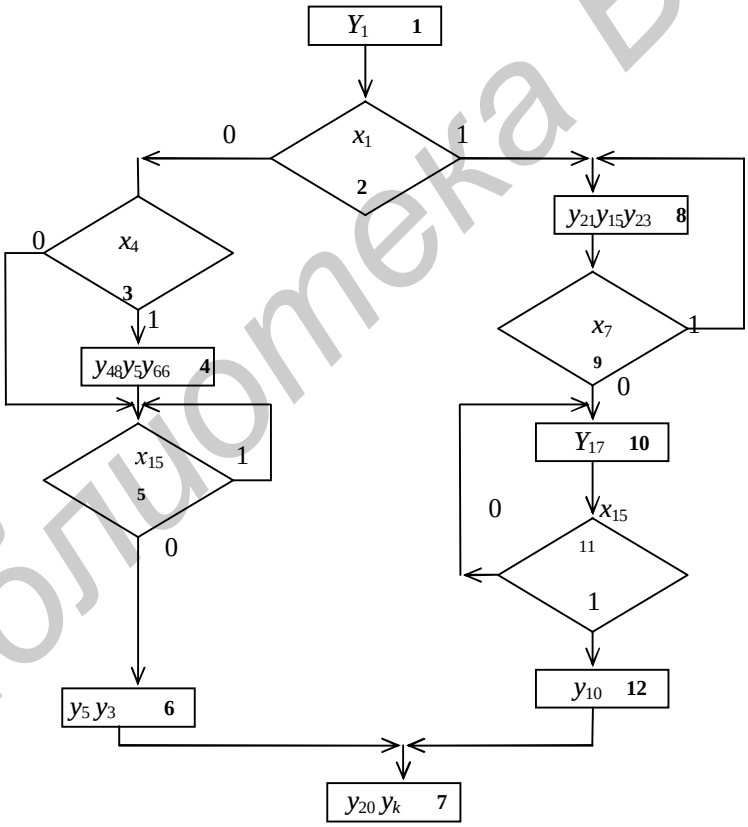


Рис. 2.10

Объект управления характеризуется следующими параметрами:

– множество проверяемых условий $X = \{x_1, x_2, ..., x_{15}\}$;

– множество выполняемых микроопераций $Y = \{y_1, y_2, ..., y_{120}, y_к\}$ ($y_к$ – микрооперация, обозначающая последнюю микрокоманду микропрограммы);

– емкость памяти для записи микропрограмм $V_{зу}$ = 2 Кбайт (1 Кбайт = = 2^{10} байт);

– длина ячейки памяти L = 16 бит;

– начальный адрес размещения составляемой микропрограммы в памяти $A_н$ = 530.

Решение

Из характеристик управляемого объекта следует:

– длина кода для кодирования микроопераций равна семи (т. е. $к$ = 7), так как количество выполняемых в управляемом объекте микроопераций равно 120;

– длина кода для кодирования условий равна четырем (т. е. p = 4), так как количество проверяемых условий в управляемом объекте равно 15;

– длина кода адреса равна десяти, так как количество адресов в памяти при длине адресуемой ячейки 16 бит (т. е. два байта) равно 1024.

Таким образом, формат микрокоманды для данного управляемого объекта имеет вид, приведенный на рис. 2.11.

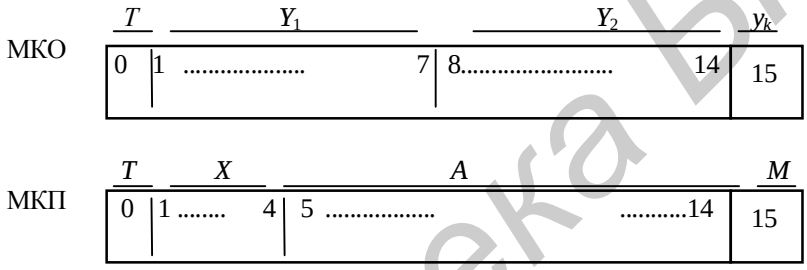


Рис. 2.11

Формат операционной микрокоманды (МКО) имеет длину 16 бит и включает:

– поле типа микрокоманды (M), имеющее длину в один бит и занимающее нулевой разряд микрокоманды; в этом поле для данного типа микрокоманды записано значение «1»;

– поле первой микрооперации (Y_1), которое занимает разряды с 1-го по 7-й;

– поле второй микрооперации (Y_2), которое занимает разряды с 8-го по 14-й;

– поле микрооперации $y_к$, которое используется только в последней микрокоманде.

Таким образом, данный формат позволяет задавать при необходимости в одной микрокоманде две микрооперации.

Формат микрокоманды перехода (МКП) имеет длину 16 бит и включает:

– поле типа микрокоманды (T), имеющее длину в один бит и занимающее нулевой разряд микрокоманды; в этом поле для данного типа микрокоманды записано значение «0»;

– поле проверяемого условия (X), которое занимает разряды с 1-го по 4-й;

– поле адреса (A), которое занимает разряды с 8-го по 14-й;

– поле модификатора дисциплины перехода (*M*) занимает 15-й разряд микрокоманды; при *M* = 1 используется первая, при *M* = 0 – вторая дисциплина перехода.

При составлении микропрограммы с помощью микрокоманд необходимо реализовать все вершины, имеющиеся в ГСА, и обеспечить необходимые ветвления процесса.

Микропрограмма для ГСА (см. рис. 2.10) приведена в форме табл. 2.3.

Таблица 2.3

№ строки	№ вершины	Адрес расположения микрокоманды в ЗУ	Код микрокоманды	Примечания
1	2	3	4	5
2	1	1000010010	1. 0000001 . 0000000. 0	
3	2	1000010011	0 . 0001. 1000011011. 1	3
4	8	1000010100	1. 0010101. 0001111. 0	
5	8’	1000010101	1. 0010111. 0000000. 0	
6	9	1000010110	0. 0111. 1000010100. 0	8
7	10	1000010111	1. 0010001. 0000000. 0	
8	11	1000011000	0. 1111. 1000010111. 1	10
9	12	1000011001	1. 0001010. 0000000. 0	
10	7	1000011010	1. 0010100. 0000000. 1	
11	3	1000011011	0. 0100. 1000011110.1	5
12	4	1000011100	1. 0110000. 0000101. 0	
13	4’	1000011101	1. 1000010. 0000000. 0	
14	5	1000011110	0. 1111. 1000011110. 0	5
15	6	1000011111	1. 0000101. 0000011. 0	
16	–	1000100000	0. 0000. 1000011010. 1	7

В приведенной таблице:

– в первой графе фиксируется номер строки;

– во второй графе приводится номер вершины, реализуемой микрокомандой этой строки;

– в третьей графе указан адрес расположения данной микрокоманды (в двоичном коде) в запоминающем устройстве;

– в четвертой графе располагается код микрокоманд;

– в пятой графе указаны номера вершин – ссылки, адрес микрокоманды в ЗУ которых должен быть указан в данной команде перехода.

В приведенной микропрограмме кодировка микроопераций и проверяемых условий осуществлена по их индексам в двоичном коде. Подчеркнутые коды адресов в микрокомандах перехода заполняются после записи последней строки формируемой микропрограммы, используя коды, соответствующие вершинам – ссылкам, указанные в графе «Адрес микрокоманды».

Микрокоманда во второй строке реализует первую вершину ГСА и поэтому записывается по адресу в ЗУ, соответствующему начальному адресу $A_н$,

равному заданному начальному адресу 530 (в графе «Адрес расположения микрокоманды в ЗУ» в первой строчке записан двоичный эквивалент десятичного числа 530). Данная микрокоманда реализует операторную вершину, поэтому в поле *T* кода микрокоманды имеет место «1». В реализуемой вершине задается одна микрооперация y_1, поэтому в поле *Y*1 записан двоичный семибитовый эквивалент ее индекса «1», в поле *Y*2 записан двоичный эквивалент «0», а в поле $y_к$ записан 0, так как данная микропрограмма реализует не последнюю вершину ГСА.

Микрокоманда третьей строки реализует вторую вершину ГСА. Она в любом случае выполняется вслед за микрокомандой, реализующей вершину номер 1, поэтому записывается в ЗУ по адресу на единицу большему, чем адрес расположения в ЗУ микрокоманды, реализующей вершину номер 1. Данная микрокоманда реализует условную вершину исходного графа, поэтому в данной микрокоманде поле *T* имеет значение 0. В поле *X* данной микрокоманды записан двоичный эквивалент индекса проверяемого в реализуемой вершине условия x_1. При выполнении данного ветвления используется первая дисциплина перехода, поэтому в поле *M* данной микрокоманды установлена 1. При первой дисциплине перехода в поле *A* микрокоманды должен быть установлен адрес расположения в памяти микрокоманды, реализующей вершину 3 ГСА, расположенную по выходу «0» данной условной вершины. Поэтому в графе «Примечания» записан номер этой вершины – 3, а в следующем адресе располагается микрокоманда, реализующая вершину 8, расположенную по выходу «1» данной вершины 2. Поле адреса *A* в данной микрокоманде перехода и во всех других микрокомандах перехода первоначально не заполняется. Оно заполняется после того, как будут записаны в память все микрокоманды формируемой микропрограммы.

При реализации вершины 8 необходимо задать три микрооперации, что нельзя сделать с помощью одной операционной микрокоманды принятого формата. Поэтому данная вершина реализуется с помощью двух микрокоманд (строки 4 и 5). Аналогичный случай имеет место при реализации вершины 4 (строки 12 и 13).

В строке 10 представлена микрокоманда, реализующая последнюю вершину ГСА, поэтому в ее коде в поле y_k установлена единица. В следующем адресе ЗУ размещается микрокоманда, соответствующая вершине начала одной из еще не реализованной ветви ГСА (в данном случае это вершина 3).

После записи микрокоманды, реализующей вершину 6, необходимо расположить по следующему адресу в ЗУ микрокоманду, реализующую вершину 7. Однако вершина 7 уже представлена в микрокоманде (строка 10). Поэтому в следующем адресе (строка 16) записывается команда безусловного перехода к микрокоманде, реализующей вершину 7. Команда безусловного перехода реализована на базе микрокоманды перехода с первой дисциплиной перехода при задании для проверки кода несуществующего условия (в данном случае в качестве такого кода использован код «0000»).

Таким образом, любое управление может быть реализовано на микропрограммном или аппаратном принципе.

При аппаратном принципе сложность устройства управления напрямую зависит от сложности алгоритма управления объектом. Кроме того, любое изменение алгоритма управления связано с необходимостью изменения аппаратной части синтезируемого устройства.

При микропрограммном принципе сложность устройства управления в основном влияет на объем микропрограммы, реализующей данный алгоритм. Изменение алгоритма управления в этом случае, как правило, связано с изменением текста микропрограммы в памяти без изменения аппаратной части синтезируемого устройства управления. Однако микропрограммный принцип связан с большими затратами времени, так как при реализации каждой вершины ГСА необходимо обращаться к памяти для извлечения очередной микрокоманды.

2.4. Процессор

Процессор представляет собой важнейшее устройство ЭВМ, которое используется для непосредственной реализации программы и управления взаимодействием всех других компонентов ЭВМ. Обобщенная структурная схема процессора приведена на рис. 2.12.

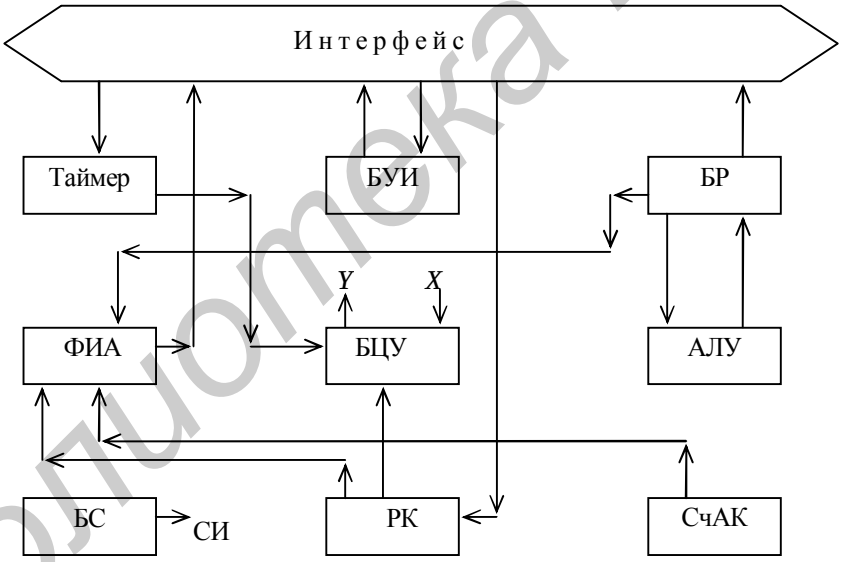


Рис. 2.12

Схема включает:

– интерфейс – средства связи с другими компонентами ЭВМ;

– таймер – средство для измерения временных интервалов;

– БУИ – блок управления интерфейсом, обеспечивающий управление обменом информации через интерфейс;

– БР – блок регистров;

– ФИА – формирователь исполнительного адреса;

– БЦУ – блок центрального управления;

– АЛУ – арифметико-логическое устройство;
– БС – блок синхронизации;
– РК – регистр текущей команды выполняемой программы;
– СчАК – счетчик адреса команды;

Блок синхронизации формирует серии синхросигналов, различающихся по частоте и фазе. На рис. 2.13 приведен пример временной диаграммы синхросигналов, вырабатываемых блоком синхронизации.

СС – синхросигналы основной частоты. Частота этих сигналов может измеряться десятками, сотнями мегагерц. СС1 – синхросигналы той же основной частоты, но сдвинутые по фазе на полпериода. СС2, СС3 – синхросигналы с частотой в два раза меньшей, чем частота сигналов СС; они различаются длительностью.

СС4, СС5, СС6, СС7 – синхросигналы с частотой в четыре раза меньшей основной; между собой они различаются сдвигам по фазе на одну четвертую своего периода.

Синхросигналы различной частоты, как правило, формируются за счет деления основной частоты с помощью счетчиков, при этом сдвиг по фазе можно обеспечить установкой различных начальных значений в этих счетчиках.

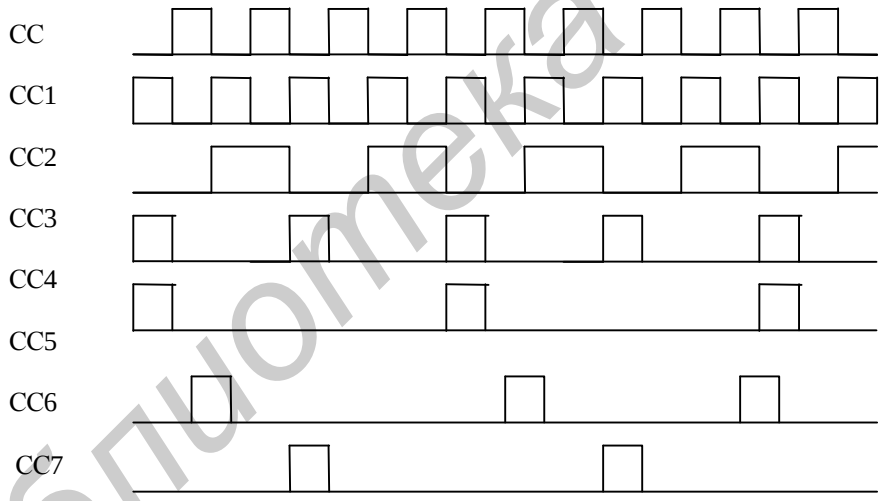


Рис. 2.13

Счетчик адреса команды представляет собой счетчик, в котором устанавливается адрес текущей выполняемой команды, увеличенный на 1, если длина команды равна разрядности ячейки памяти, или в общем случае на L, где L – длина текущей выполняемой команды. Таким образом, в счетчике заблаговременно формируется адрес следующей команды, если при выполнении текущей команды не будет организовано ветвление.

Таймер представляет собой один или несколько счетчиков, которые можно использовать для хранометрирования. Например, можно измерять затраты времени на выполнение некоторой процедуры – в начале измерения в счетчик записывается нулевое значение, и в течение всего измеряемого временного интервала

на его счетный вход «+1» подаются синхросигналы с известным периодом; по завершении процедуры в счетчике будет находиться число, соответствующее периоду времени, в течение которого на его вход поступали сигналы «+1». С помощью таймера можно определять моменты истечения заданного интервала времени (например периода регенерации запоминающего устройства). В этом случае в счетчик устанавливается число, соответствующее заданному периоду времени, и на его вход «–1» начинают подавать синхросигналы. Обнуление счетчика соответствует моменту истечения заданного периода.

Блок центрального управления в зависимости от поступающего на него из регистра команды кода выполняемой операции (коп) и множества сигналов *X*, характеризующих состояние других блоков процессора, вырабатывает множество сигналов *Y*, управляющих работой всех других блоков. Как правило, блок центрального управления строится на микропрограммном принципе.

Регистр команды используется для хранения кода выполняемой команды, и его организация зависит от форматов используемых команд. При реализации алгоритма обработки информации с помощью программы текущая выполняемая команда, в общем случае, должна нести информацию о двух операндах, местоположении результата и следующей команды. Задать всю эту информацию в едином формате команды затруднительно, тем более, что для обеспечения ветвления необходимо иметь информацию о двух возможных путях продолжения обработки. Поэтому так же, как и в случае микропрограммирования, используется по крайней мере два типа команд: операционные команды и команды перехода.

Команды перехода для организации ветвления в явной форме задают адрес одного из двух возможных продолжений. В качестве второго продолжения используется команда, расположенная в следующем после текущей выполняемой команды адресе памяти.

В операционных командах при неявном задании адреса следующей команды требуется определить только адреса двух операндов и результата. Однако трехадресные команды практически не используют из-за слишком большой длины. На практике применяют одноадресные или двухадресные команды. В двухадресных командах в явной форме задаются адреса двух операндов, а для размещения результата используется адрес одного из операндов. В одноадресных командах второй адрес определяется самим типом команды (как правило, в этом случае в качестве второго адреса используется один из регистров блока регистров).

Адреса в команде могут задавать или локацию в памяти, или один из регистров блока регистров. В этой связи двухадресные команды могут быть следующих типов:

– «память – память» – оба адреса команды относятся к памяти;

– «память – регистр» – один адрес команды относится к памяти, другой представляет собой номер регистра в блоке регистров;

– «регистр – регистр» – оба адреса в команде задают номера регистров.

Номер регистра имеет меньшую длину, чем адрес в памяти, поэтому меньшую длину из приведенных типов имеет команда «регистр – регистр».

На рис. 2.14 приведен формат команды, который включает три поля:

– коп – поле, определяющее код выполняемой команды;

– модификатор (м) – поле, которое уточняет операцию; это поле может задавать тип адресации и регистры, используемые по умолчанию, длину команды;

– адрес – поле, в котором определяются адреса операндов и результата.

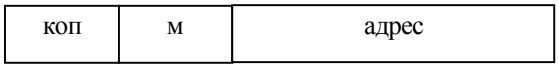


Рис. 2.14

В командах используются следующие типы задания операндов:

– непосредственное задание операнда;
– прямая адресация;
– косвенная адресация;
– базовая адресация;
– индексная адресация.

Непосредственное задание операнда предполагает размещение операнда непосредственно в самой команде. В этом случае при выполнении команды не нужно тратить время для извлечения операнда из памяти. Применяется только тогда, когда операндом является константа.

Прямая адресация – это самый очевидный способ адресации; в этом случае в команде находится код адреса размещения операнда в памяти. При такой адресации затруднительно решать проблему перемещаемости программы в памяти.

Косвенная адресация характеризуется тем, что в команде записывается адрес размещения в памяти адреса операнда. При таком способе адресации для получения операнда необходимо дважды обратиться к памяти:

– при первом обращении, используя адрес, заданный в команде, из памяти читается адрес операнда;

– при втором обращении, используя адрес, полученный при первом чтении, из памяти читается сам операнд.

Данный способ адресации требует больших затрат времени для получения операнда, но при такой адресации облегчается решение задачи перемещаемости программы в памяти – после расположения программы в ячейку памяти, на которую указывает адрес в команде, записывается адрес текущего положения операнда в памяти.

Базовая адресация характеризуется тем, что исполнительный адрес, т. е. адрес, который непосредственно передается в память для записи или чтения, формируется за счет сложения двух компонент – базы и смещения. На рис. 2.15 приведена схема, иллюстрирующая этот принцип.

Поля *B* и *D* в адресной части команды определяют базу и смещение формируемого исполнительного адреса соответственно. В приведенном примере в

поле *B* записан номер *i* регистра в регистровой памяти. Исполнительный адрес $A_И$ для обращения к одной из ячеек памяти формируется за счет суммирования содержимого заданного регистра *i* и содержимого поля *D*.

Адрес памяти $A_Б$ соответствует значению в базовом регистра, а отклонение (смещение) исполнительного адреса от $A_Б$ составляет *S*, которое равно значению кода смещения в поле *D* команды. Номер регистра, где хранится база, может задаваться неявно: данной команде по умолчанию может быть приписан один из регистров, куда предварительно помещается значение базы.

Данный вид адресации удобен для перемещаемости программ в памяти и однотипной обработки элементов матрицы.

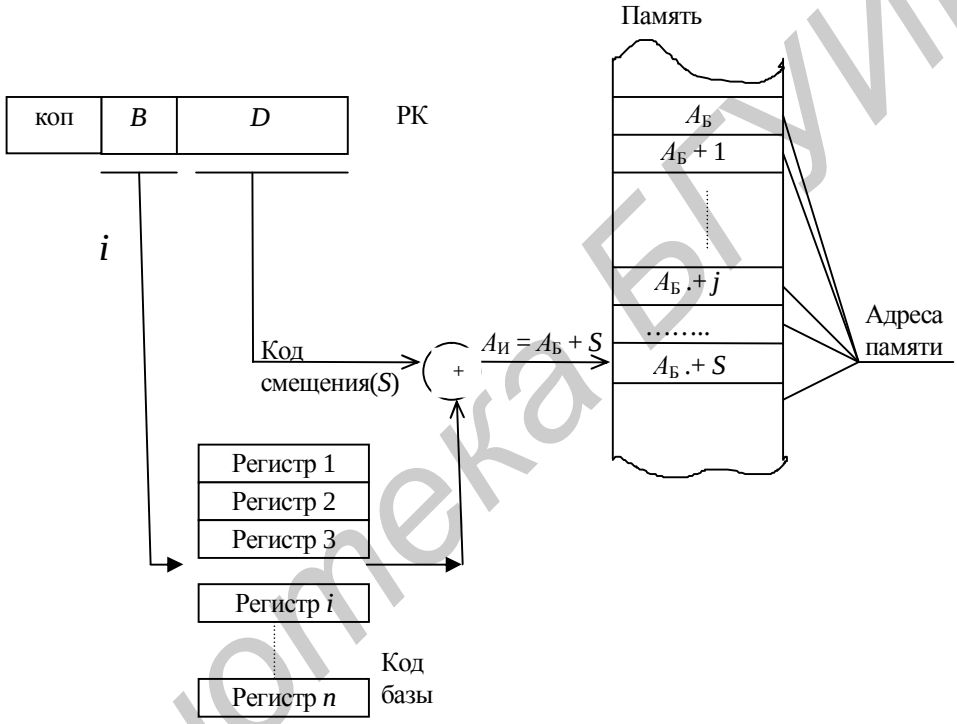


Рис. 2.15

Индексная адресация предполагает формирование адреса на основании адреса предыдущей команды путем его модификации на «1», при этом могут быть использованы два режима индексации:

– инкрементный – модификация выполняется на «+1»;

– декрементный – модификация выполняется на «–1».

2.5. Запоминающие устройства

Запоминающие устройства служат для хранения программ, данных и результатов обработки информации.

По месту в структуре ЭВМ запоминающие устройства бывают оперативными и внешними.

Оперативные запоминающие устройства, или оперативная память (ОП), представляют собой устройства, к которым обращается процессор за командами реализуемой программы, данными, подлежащими обработке; в эту же память помещаются промежуточные и конечные результаты непосредственно по завершении реализации программ. ОП характеризуется высоким быстродействием и сравнительно небольшим объемом памяти, измеряемым сотнями мегабайт.

Внешние запоминающие устройства (ВЗУ) представляют собой устройства, предназначенные для долговременного хранения программ, данных и конечных результатов выполненных программ. По мере необходимости информация, как правило, большими блоками передается в оперативную память и наоборот. ВЗУ характеризуются большой информационной емкостью, измеряемой десятками и сотнями гигабайт, и малой стоимостью хранения единицы информации. Однако по сравнению с ОП внешние запоминающие устройства имеют меньшее быстродействие. Кроме того, время обращения к информации в ВЗУ зависит от местоположения информации в запоминающей среде поэтому на ее поиск может затрачиваться много времени, которое измеряется десятками миллисекунд и более. В отличие от ВЗУ оперативная память – это память с равновероятным доступом, т. е. время обращения к информации у этой памяти не зависит от местоположения этой информации в запоминающей среде.

Кроме этого, по месту в структуре ЭВМ выделяют сверхбыстродействующую память, или *кэш-память*). Данная память имеет меньшую информационную емкость (на порядок и более), чем оперативная память, но обладает быстродействием, в несколько раз превышающим быстродействие оперативной памяти. В структуре ЭВМ эта память располагается между процессором и оперативной памятью.

Существующие виды памяти можно подразделить на энергозависимые и энергонезависимые.

К энергозависимым относятся ЗУ, которые теряют информацию при отключении питания. В энергонезависимых запоминающих устройствах при каждом включении в памяти сохраняется информация, которая была до отключения питания.

По выполняемым функциям различают:

– память для чтения и записи;

– память только для записи, которая часто называется ПЗУ (постоянное запоминающее устройство).

Несмотря на ограниченность функций, ПЗУ широко используется наряду с памятью для чтения и записи. Это объясняется тем, что ПЗУ, как правило, обладает таким же быстродействием и не большей стоимостью, чем память для записи и чтения. Кроме того, ПЗУ – это энергонезависимая память, в то время как память для чтения и записи, как правило, является энергозависимой.

Все типы внешней памяти являются энергонезависимыми.

2.5.1. Оперативная память

Различают два вида оперативной памяти:

– статическая память;

– динамическая память.

Статическая память характеризуется тем, что надежность считывания находящейся в ней информации не зависит от времени хранения.

Динамическая память отличается тем, что спустя некоторый период времени после записи информации в память надежное считывание этой информации не гарантировано, т. к. информация в процессе хранения в динамической памяти как бы «затухает». Поэтому при использовании такой памяти через интервалы времени, не бóльшие, чем время гарантированного хранения, предусматривается процедура регенерации, при реализации которой происходит чтение информации по всему объему памяти и ее запись по старому адресу. Необходимость периодической регенерации снижает эффективность работы памяти, однако динамическая память широко используется благодаря тому, что при прочих равных условиях она обладает существенно меньшей стоимостью, чем память статическая.

Динамические запоминающие устройства

В динамических ЗУ запоминание информации осуществляется на конденсаторах. Пример простейшего устройства данного типа приведен на рис. 2.16.

На рис. 2.16 представлена запоминающая среда с тремя адресами, в каждом из которых хранится 4 бита информации. Бит информации запоминается на элементе памяти (ЭП). ЭП состоит из конденсатора, подключенного к истоку (И) транзистора (Т). Выбор ячейки памяти осуществляется за счет подачи на соответствующую адресную шину $A_и$ сигнала. При этом сигнале, поступающем на затвор (З) транзистора, открываются все четыре транзистора, подключенных к шине A_i. Если конденсатор в запоминающем элементе был заряжен (в элементе ранее была записана «1»), то через стоки (Ст) и исток (И) открытого транзистора течет ток, который, проходя по вертикальной разрядной шине, воспринимается на выходе как сигнал прочитанной «1». Если конденсатор запоминающего элемента не был заряжен (ЭП хранит «0»), то ток на выходной вертикальной линии отсутствует, что означает сигнал прочитанного нуля на выходе соответствующего разряда выбранного адреса.

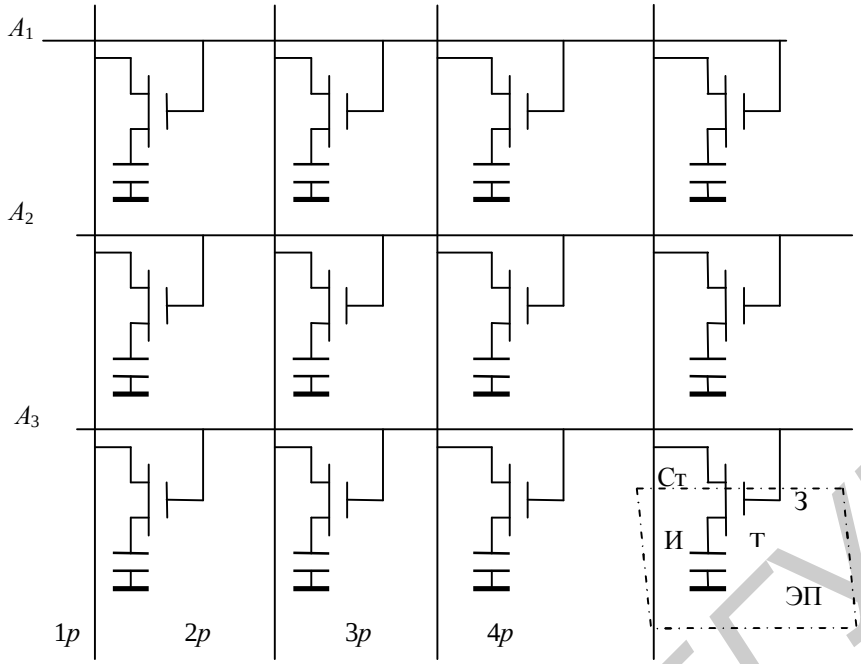


Рис. 2.16

При записи информации также, как и при чтении, выбор нужного адреса осуществляется за счет подачи тока на соответствующую горизонтальную шину A_i. В зависимости от того, что нужно записать, в отдельные разрядные вертикальные шины подаются разные сигналы:

– если в разряд необходимо записать «1», то на соответствующую разрядную шину подается положительное напряжение, которое через открытый транзистор T обеспечивает заряд конденсатора C;

– если в разряд необходимо записать «0», то на соответствующую разрядную шину подается низкое напряжение, что приводит к разряду конденсатора (если он был до этого заряжен).

В рассматриваемом ЭП со временем конденсатор разряжается, чем и объясняется динамический характер такой памяти.

Для увеличения постоянной разряда конденсатора используются более сложные элементы памяти, один из которых представлен на рис. 2.17.

Схема ЭП состоит из трех транзисторов $T1$, $T2$, $T3$ и конденсатора C.

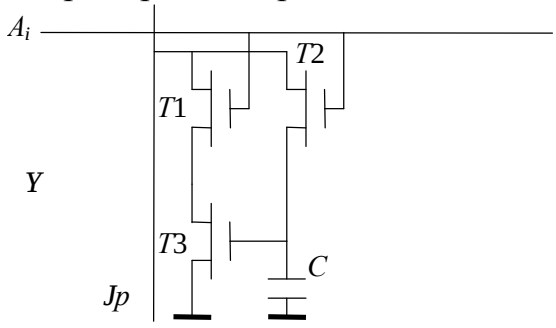


Рис. 2.17

При чтении информации адресным сигналом A_i открывается транзистор $T1$. Если конденсатор C запоминающего элемента был заряжен, то через $T1$, $T3$ по выходной вертикальной разрядной шине протекает ток, что воспринимается как сигнал прочитанной «1»; в противном случае считается, что прочитан «0». При записи информации подаваемым по вертикальной шине сигналом конденсатор C через открытый транзистор $T2$ или заряжается, или разряжается.

Статическая память

Статическая память строится на триггерах, каждый из которых хранит один бит информации. На рис. 2.18, а приведена реализация элемента статической памяти, а на рис. 2.18, б – ее обозначение.

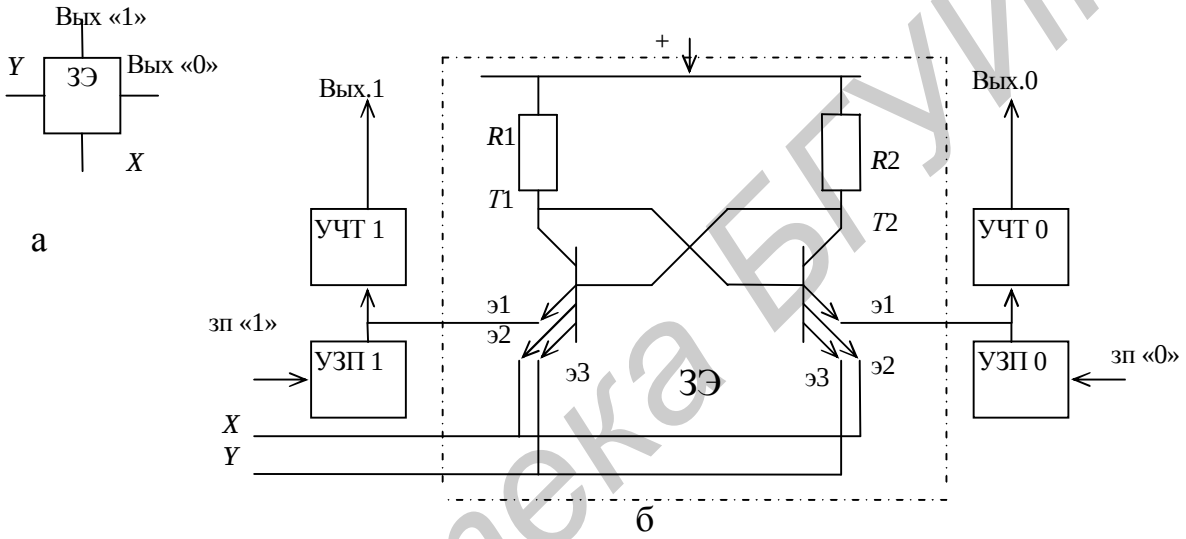


Рис. 2.18

Запоминающий элемент (см. рис. 2.18, а) построен на транзисторах $T1$ и $T2$, которые имеют по три эмиттера э1, э2, э3.

Приведенная схема имеет два устойчивых состояния.

Действительно, если один транзистор, например $T1$, открыт, то на его коллекторе будет низкое напряжение, что способствует закрыванию по базе второго транзистора ($T2$). И наоборот, если один транзистор закрыт, например $T2$, то высокое напряжение на его коллекторе будет способствовать закрыванию другого транзистора ($T1$). Каждому из двух устойчивых состояний данного триггера ставятся в соответствие логические «1» и «0». В приведенной схеме триггер хранит единицу, если транзистор $T1$ закрыт, а транзистор $T2$ открыт; схема хранит «0», если транзистор $T1$ открыт, а транзистор $T2$ закрыт.

Шины X и Y являются координатными шинами матрицы, состоящей из многих подобных запоминающих элементов. При чтении информации нужный адрес (нужный запоминающий элемент) выбирается за счет подачи сигнала (+) по одной из шин X и одной из шин Y. Запоминающий элемент, расположенный в матрице на пересечении этих шин, будет выбранным. В этом случае в выбранном запоми-

нающем элементе оба транзистора будут иметь условие для закрывания по соответствующему эмиттеру положительными сигналами по э2 и э3.

Если схема хранит «1», то в открытом транзисторе *Т*2 весь коллекторный ток пойдет через э1 и попадет на вход усилителя чтения нуля УЧТ 0; усилители имеют отрицательный коэффициент передачи, поэтому на выходе УЧТ 0 будет иметь место «0». В то же самое время *Т*2 сохраняет свое закрытое состояние, ток через его коллектор, а следовательно, и через его эмиттеры отсутствует; а отсутствие тока по э3 приводит к появлению сигнала «1» на выходе усилителя УЧТ 1.

Если рассуждать аналогичным образом, то можно убедиться, что если схема хранит «0», то на выходе УЧТ 0 будет иметь место сигнал «1», а на выходе УЧТ 1 – сигнал «0».

Запись информации осуществляется следующим образом.

После выбора запоминающего элемента подачей сигналов по соответствующим координатным шинам *X* и *Y* вырабатывается сигнал записи (сигнал записи единицы – зп. «1» или записи нуля зп. – «0»).

Предположим, что необходимо записать «1» в элемент, который находится в состоянии «1». В этом случае положительный сигнал с выхода усилителя записи УЗП 1 закрывает транзистор *Т*1 по э1 (по эмиттеру э1 и э2 этот транзистор закрыт сигналами по шинам *X* и *Y*). Однако *Т*1 в начальном состоянии был закрыт, поэтому состояние запоминающего элемента не изменяется и он продолжает хранить единицу и после снятия сигналов *X*, *Y*.

Предположим «1» записывается в элемент, хранящий ноль. В этом случае подаваемый положительный сигнал приводит к закрыванию *Т*1, который, закрываясь, высоким напряжением на своем коллекторе открывает *Т*2 по базе, и через коллектор и эмиттер э1 транзистора *Т*2 начинает течь ток, что в свою очередь приводит к закрыванию Т1 по базе. После снятия сигналов *X*, *Y* картина не изменяется: *Т*1 остается закрытым, а *Т*2 – открытым, т. е. запоминающий элемент оказывается в состоянии «1». Если в выбранном запоминающем элементе записывается «0», то подается сигнал зп «0». Так как схема триггера симметрична относительно входа записи единицы и входа записи нуля легко показать, что независимо от того, что хранил запоминающий элемент, после воздействия сигнала зп «0» схема окажется в нулевом состоянии.

На рис. 2.19 приведена реализация запоминающей среды на шестнадцать адресов с хранением одного бита по каждому адресу.

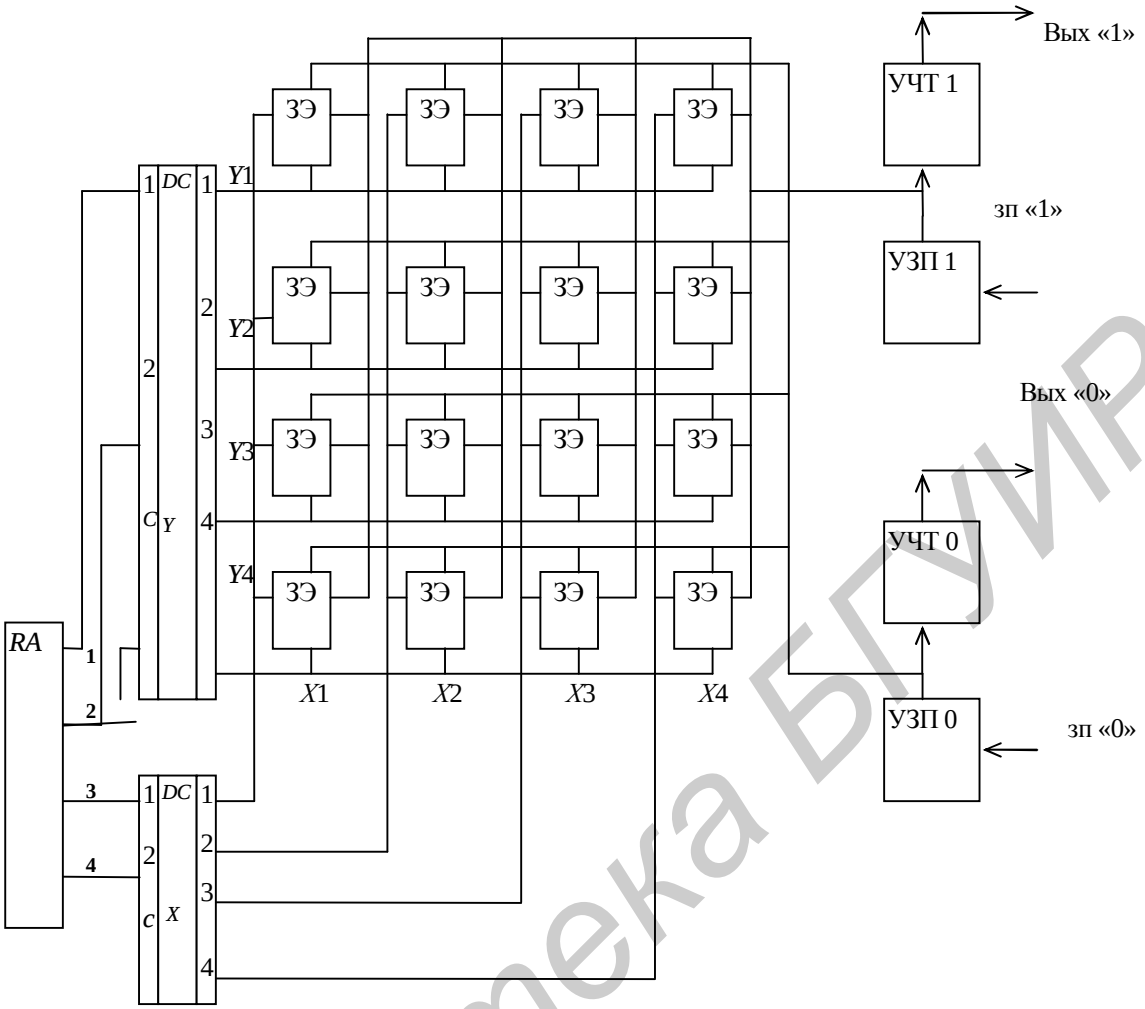


Рис. 2.19

Каждый столбец приведенной матрицы обслуживается одной координатной шиной *X* (*X*1, *X*2, *X*3, *X*4). Каждая строка матрицы обслуживается одной координатной шиной *Y* (*Y*1, *Y*2, *Y*3, *Y*4). Сигналы на вертикальные и горизонтальные координатные шины *X*, *Y* подаются с выходов дешифраторов *DCY* и *DCX*, к информационным входам которых подключены разрядные выходы регистра адреса *RA*. Два первых разряда *RA* подключены к входу *DCY* и обеспечивают выбор одной из четырех вертикальных координатных шин *Y*, а два оставшихся разряда *RA* подключены к входу *DCX* и обеспечивают выбор одной из четырех горизонтальных координатных шин *X*.

Выходы «1» всех запоминающих элементов матрицы объединены и подключены к усилителям записи и чтения единицы. Выходы «0» всех запоминающих элементов матрицы объединены и подключены к усилителям записи и чтения нуля.

2.5.2. Постоянные запоминающие устройства

Постоянные запоминающие устройства предназначены для многократного чтения однажды записанной информации. С точки зрения записи в них информации ПЗУ подразделяются на следующие виды:

– *память только для чтения* – запись информации в таких устройствах осуществляется в процессе их производства; при их эксплуатации возможно только чтение записанных данных;

– *память с однократной записью* – потребитель перед началом использования ПЗУ записывает сам свои данные, и в дальнейшем информация только читается;

– *ПЗУ с возможностью перезаписи* – в таких устройствах при эксплуатации возможно выполнение как операции чтение, так и операции записи информации, однако основной операцией в таких ЗУ является операция чтения, так как операция записи занимает сравнительно большое время и, как правило, для ее реализации требуется специальное оборудование.

На рис. 2.20 приведена диодная реализация ПЗУ. На начальном этапе изготовления такого ПЗУ создается матрица, в которой каждая вертикальная шина связана с каждой горизонтальной шиной цепочкой, состоящей из последовательно включенных диода (*D*) и легкоплавкой перемычки (ЛПП). При записи информации ненужные связи вертикальных и горизонтальных шин удаляются за счет посылки в них тока повышенной мощности, в результате чего ЛПП разогреваются и испаряются. Перезапись информации в таких ПЗУ невозможна.

В приведенном ПЗУ запоминающая среда состоит из трех ячеек памяти, в каждой из которых хранится четырехразрядное слово:

– по первому адресу – 1011;
– по второму адресу – 1100;
– по третьему адресу – 0011.

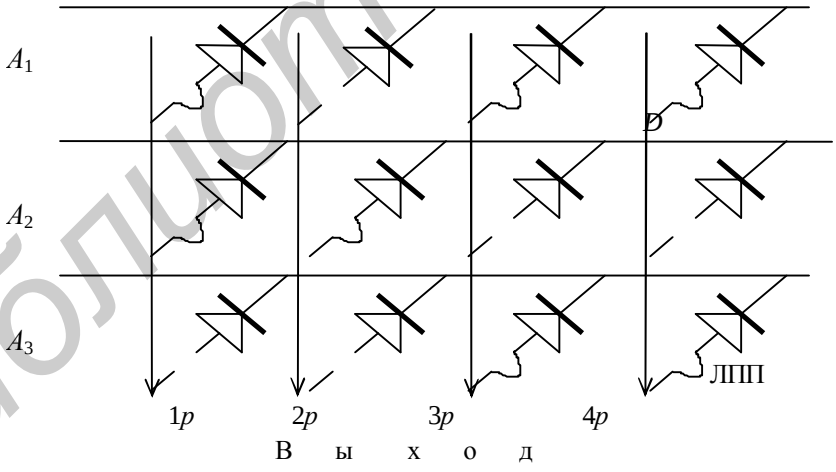


Рис. 2.20

Выбор ячейки (адресация) осуществляется за счет подачи отрицательного сигнала на одну из адресных шин A_1, A_2, A_3. Это приводит к тому, что на четырехразрядном выходе появляется код, соответствующий слову, записанному в выбранном адресе.

На рис. 2.21, а приведена транзисторная реализация ПЗУ. В схеме используются трехэмиттерные транзисторы *Т*1, *Т*2, *Т*3. Запись информации, как и

в предыдущем случае, осуществляется посредством удаления соответствующих легкоплавких перемычек.

В приведенной схеме реализована запоминающая среда, состоящая из трех слов по три бита в каждом. Выбор нужного слова осуществляется за счет подачи положительного сигнала по соответствующей адресной шине A_i, при этом на выходе появляется код, соответствующий комбинации перемычек, и их отсутствия в цепях эмиттеров выбранного транзистора. Возможна только однократная запись.

На рис. 2.21, б приведен запоминающий элемент ПЗУ с возможностью многократной перезаписи. Запоминающий элемент представлен в виде цепочки из последовательно включенных транзисторов $T1$ и $T2$. Транзистор $T1$ открыт, если на его затворе (З) имеет место соответствующий заряд, открывающий переход «сток (СТ) – исток (И)». Такое условие можно создать, подав соответствующий сигнал на адресную шину A_i. Транзистор $T2$ имеет плавающий, т. е. изолированный, затвор (ПЗ). Чтобы транзистор был открыт, на его затворе необходимо создать соответствующий заряд, что обеспечивается за счет подачи повышенного положительного напряжения на его переход «исток и сток». Заряд на плавающих затворах создают в $T2$ только в тех запоминающих элементах, в которых записывается единица.

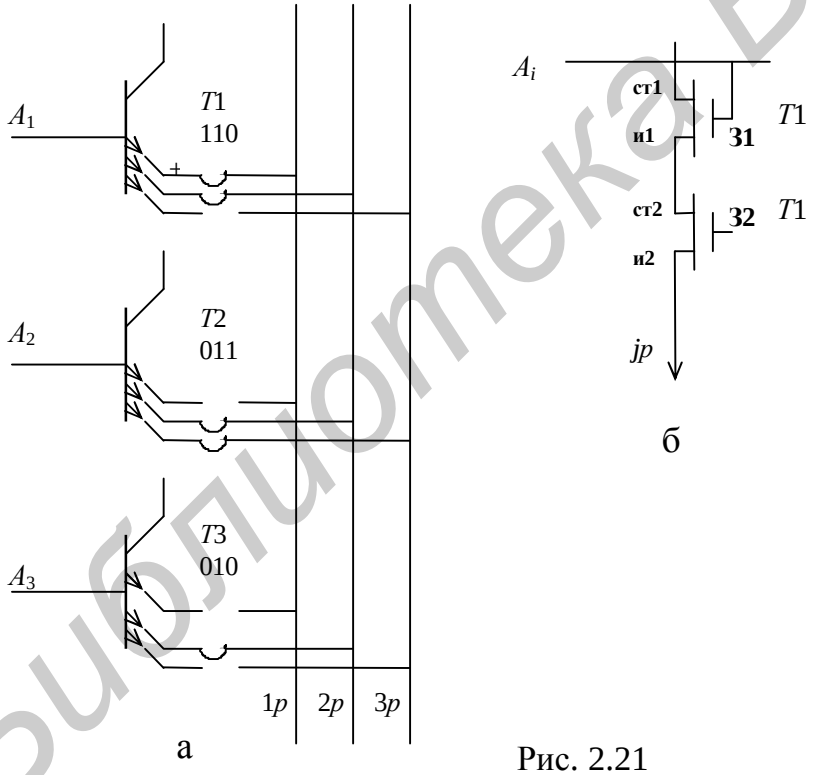


Рис. 2.21

Перезапись информации осуществляется за счет удаления ранее записанной информации и записи новых данных.

Удаление информации, т. е. ликвидация заряда в плавающем затворе, осуществляется за счет подачи напряжения обратной полярности.

Операция перезаписи в таких ПЗУ занимает большое время, поэтому основной операцией в них является операция чтения.

Св. план 2009, поз. 55

Учебное издание

ОРГАНИЗАЦИЯ И ФУНКЦИОНИРОВАНИЕ ЭВМ

Методическое пособие для студентов специальности 1-40 01 01
«Программное обеспечение информационных технологий»
дневной формы обучения

В 3-х частях

Часть 3

Пешков Анатолий Тимофеевич
Кобайло Александр Серафимович

СХЕМОТЕХНИЧЕСКИЕ ОСНОВЫ ЭВМ

Редактор Е. Н. Батурчик
Корректор Л. А. Шичко
Компьютерная верстка Е. С. Чайковская

Подписано в печать 03.06.2009.	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л.
Уч.-изд. л. 3,5.	Тираж 150 экз.	Заказ 638.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494171 от 03.04.2009.
220013, Минск, П. Бровки, 6