Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Самарский государственный технический университет»

*Кафедра **“Вычислительная техника”***

Синтез микропрограммных автоматов

Конспект лекций

по курсу «Синтез микропрограммных автоматов»

Самара 2013

Составитель: И.В. ВОРОНЦОВ
УДК 621.382

Синтез микропрограммных автоматов. Конспект лекций по курсу «**Синтез микропрограммных автоматов**»/
СамГТУ; *Сост. И.В. ВОРОНЦОВ*. Самара, 2013, 64 с.

Конспект лекций рекомендуется студентам специальности 230100 , «Информатика и вычислительная техника», профиль подготовки – «Электронные вычислительные машины, комплексы, системы и сети» (бакалавры)

Лекция 1.

Раздел 1. Микропрограммные цифровые автоматы

Тема 1.1. Предмет теории автоматов, его место и задачи в подготовке бакалавра по информатике и вычислительной технике. Краткий исторический обзор

Задачи изучения дисциплины. Требования к знаниям, умениям и навыкам, которыми должны овладеть студенты:

- студент должен знать теоретические и практические вопросы, подробно изложенные ниже, в том числе: принципы построения комбинационных схем и цифровых управляющих автоматов, отличие автоматов различных типов друг от друга, подходы к проектированию различных цифровых устройств, описанных на содержательном уровне.
- студент должен уметь перейти от содержательной формулировки задачи к ее математической постановке в терминах теории автоматов; уметь описывать автоматы с помощью регулярных выражений алгебры событий, ГСА, ЛСА, МСА, таблиц переходов и выходов и т.п.; уметь преобразовывать и объединять автоматы
- студент должен приобрести навыки в проектировании сложных цифровых устройств, как композиции операционного и управляющего автоматов, и в исследовании в исследовании подобных устройств.

Краткая характеристика дисциплины, её место в учебном процессе

Идеи и методы, используемые в теории автоматов, имеют не только теоретическое значение, но и позволяют грамотно проектировать как технические, так и программные системы. Данный курс дает студентам основные понятия о конечных автоматах, их свойствах, преобразованиях над автоматами, о последовательном и параллельном соединении автоматов, о сетях автоматов.

Кроме теоретических представлений об автоматах как абстрактных математических объектах, рассматриваются вопросы технической реализации наиболее распространенных на практике микропрограммных автоматов на жесткой и программируемой логике, а так же на программируемых логических матрицах.

Связь с предшествующими дисциплинами

Курс требует от студентов знаний следующих тем по дисциплинам, изучаемым на младших курсах:

Дискретная математика : переключательные функции (ПФ); способы задания ПФ; специальные разложения ПФ; не полностью определенные (частные) ПФ; минимизация ПФ и не полностью определенных ПФ; теорема о функциональной полноте; примеры функционально-полных базисов; разрешимые и неразрешимые проблемы; схемы алгоритмов;

Математическая логика и теория алгоритмов: логика высказываний

Программирование на языке высокого уровня: весь курс.

Связь с последующими дисциплинами

Так как курс читается на 3-ом семестре, изучаемый материал может быть использован в качестве основы для преподавания следующих дисциплин и конкретных тем:

Организация ЭВМ и систем: функциональная и структурная организация процессора; организация памяти ЭВМ; организация ввода-вывода; периферийные устройства; архитектурные особенности организации ЭВМ различных классов;

Схемотехника ЭВМ: функциональные узлы комбинационного типа; функциональные узлы последовательностного типа: регистры, счетчики, распределители; матричные умножители; БИС/СБИС с программируемой структурой: программируемые логические матрицы, программируемая матричная логика, базовые матричные кристаллы;

Микропроцессорные системы: организация подсистем обработки, управления, памяти и ввода-вывода; основные задачи проектирования МПС; однокристальные микро-ЭВМ и контроллеры, организация и особенности проектирования систем на их основе

Краткий исторический обзор.

Лекция 2.

Тема 1.2. Тема 1.2. Микропрограммные цифровые автоматы. Микропрограммирование. Микропрограммы цифровых автоматов. Граф - схемы алгоритмов цифровых автоматов.

Микропрограммный управляющий автомат – это конечный автомат, обеспечивающий выполнение микропрограмм операционным устройством.

Микропрограмма описывает алгоритм выполнения какой-либо сложной операции (арифметической, логической и т.п.) в терминах микроопераций и логических условий.

К микрооперациям относятся простейшие действия над словами информации, выполняемые элементами операционного автомата (ОА): загрузка и хранение слова, инверсия слова, сложения двух слов, инкремент, декремент, сдвиг на 1 разряд влево или вправо и т.п.

Микропрограмма часто записывается в виде ГСА (подобно ГСА в программировании). Идентификаторами в микрооперациях являются обозначения операционных элементов ОА, например RA – регистр А, СТ – счетчик и т.д., и идентификаторы переменных. Микрооперации записываются в виде операторов присваивания с использованием идентификаторов и символов микроопераций.

Примеры микроопераций

RC: = 0 – в регистр RC записать (загрузить) число 0.

RA: = A – в регистр RA записать (загрузить) число А.

RA: $=$ RA – инвертировать содержимое регистра RA.

RC: = RA + RB – в регистр RC поместить результат сложения чисел, хранящихся в регистрах RA и RB.

RA: = L1 (RA). 0 – сдвинуть влево на 1 разряд содержимое регистра RA, а в освободившийся крайний правый разряд записать 0.

CT: = CT + 1 – изменить состояние счетчика СТ на +1.

Все операционные элементы характеризуются разрядностью. Разряды нумеруются справа налево начиная с нуля. Например, запись CT[3..0] означает, что счетчик СТ – четырехразрядный. Его разряды (начиная со старших) обозначаются так: CT[3], CT[2], CT[1], CT[0].

Микрооперации, допускающие совместное выполнение, называются совместимыми. Они могут записываться в одной и той же операторной вершине ГСА.

В микропрограмме кроме микроопераций (операторные вершины ГСА) используются логические условия (условные вершины ГСА). Логические условия, как правило, вырабатываются операционным автоматом и могут принимать значения «0» (ложно) или «1» (истинно).

Примеры логических условий

CT=0 – содержимое счетчика CT равно нулю.

RA(0) – содержимое младшего разряда регистра RA: если оно равно нулю – переход по выходу «0» условной вершины ГСА, если равно «1» – переход по выходу «1» условной вершины ГСА.

Микропрограмма, записанная в терминах микроопераций и логических условий, называется содержательной или функциональной.

Рассмотрим пример выполнения операции умножения двух целых положительных 8-разрядных двоичных чисел: C = A * B.

Алгоритм выполнения операции умножения двух целых 8-разрядных положительных двоичных чисел можно описать следующим образом. Имеются переменные A и B – операнды операции C = A * B – и переменная C, в которую будем записывать промежуточные суммы и в которой будет, в итоге, сформированы результат и счетчик циклов CT.

Алгоритм выполнения операции

1. C = 0, CT = 0.
2. Если B[0] = 1, то C = C + A.
3. A = L1(A). 0 – сдвиг числа A влево на 1 разряд (умножение на 2).
4. B = R1(B) – сдвиг числа B вправо на 1 разряд.
5. CT = CT + 1.
6. Если CT = 7 – перейти к п. 8.
7. Перейти к п. 2.
8. Операция выполнена.

Совершенно очевидно, что для выполнения этой операции необходим ряд операционных элементов (элементов ОА) соответствующей разрядности, выполняющих простейшие функции: хранения, суммирования, счета циклов сложения и т.п. Это следующие элементы.

RB[7..0] – 8-разрядный регистр множителя – числа B.

Функции регистра:

- загрузка числа: RB := B;

- сдвиг числа вправо на один разряд: RB := R1(B).

RA[15..0] – 16-разрядный регистр множимого – числа A. Разрядность регистра RA в два раза больше разрядности числа A, так как в процессе выполнения операции число A должно сдвигаться влево n-1 раз.

Функции регистра:

- загрузка числа: RA := A;
- сдвиг числа влево на один разряд: RA := L1(RA).

RC[15..0] – 16-разрядный регистр промежуточных сумм и результата. Разрядность регистра RC в два раза больше разрядности чисел A и B, так как в процессе выполнения операции число C может иметь разрядность в два раза большую, чем у чисел A и B.

Функции регистра:

- загрузка числа: RC := C;
- обнуление регистра: RC := 0.

CT[2..0] – 3-разрядный двоичный счетчик для подсчета числа циклов (до семи).

Функции счетчика:

- обнуление счетчика: CT := 0;
- счет: CT := CT + 1.

CM[15..0] – 16-разрядный сумматор – комбинационная схема для выполнения микрооперации: RC := RC + RA.

Пример содержательной микропрограммы выполнения операции умножения двух целых 8-разрядных двоичных чисел A и B приведен на рис. 3.1.

Каждая микрооперация в ОА инициируется микрокомандой. Микрокоманды вырабатываются управляющим автоматом (УА) в зависимости от того, какая микрооперация должна выполняться в данный момент.

По микропрограмме, записанной в форме ГСА, строится ГСА управляющего аппарата. Для этого вместо микроопераций в операторных вершинах записываются микрокоманды (Y_i), а вместо логических условий – их коды – X_i.

В ГСА управляющего автомата используются следующие микрокоманды, инициирующие выполнение соответствующих микроопераций (табл. 3.1).

Таблица 3.1

Микрокоманды	Микрооперации
$У_1$	RA := A , RB := B;
$У_2$	RC := C, CT := 0;
$У_3$	RC := RC + RA;
$У_4$	RA := L1(RA);
$У_5$	RB := R1(B);
$У_6$	CT := CT + 1;

Логические условия закодируем следующим образом (табл. 3.2).

Таблица 3.2

Код логического условия	Логические условия
X_1	RB [0] = 0;
X_2	CT = 7;

Кроме того, в ГСА УА часто предусматривают две дополнительные вершины – одну операторную (перед вершиной «конец») и одну условную (после вершины «начало»). В первой записывается микрокоманда завершения выполнения операции. Это микрокоманда не для операционного автомата, а для информирования о том, что операция выполнена и ОУ готово к выполнению следующей операции (в нашем примере – $У_7$). Вторая соответствует внешней команде «Пуск» (в нашем примере – X_3), по которой ОУ начинает выполнение операции. С учетом этого ГСА управляющего автомата для нашего примера имеет вид, приведенный на рис. 3.2.

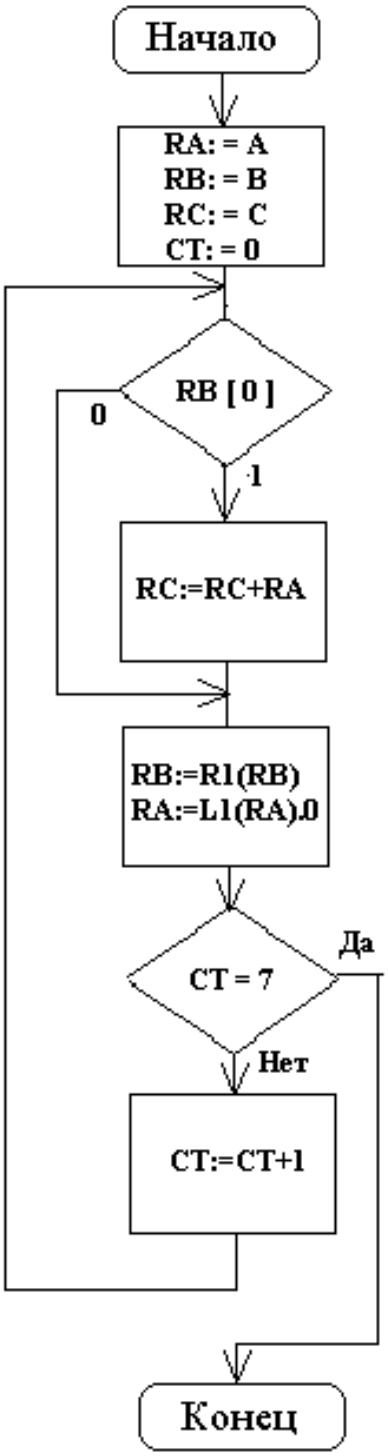
Операционный автомат содержит ряд достаточно сложных элементов, но структура ОА достаточно проста (рис. 3.3).

На структурной схеме ОА показаны описанные выше операционные элементы, связи между ними, указана разрядность данных и элементов. У операционных элементов – слева входы, справа – выходы; снизу показаны управляющие сигналы (микрокоманды $У_i$), инициирующие выполнение определенной микрооперации этим элементом.

На вертикальных линиях (на входах и выходах элементов) указаны разряды данных, поступающих на входы элементов и снимаемых с их выходов.

Входами для ОА являются обрабатываемые данные (числа А и В) и управляющие сигналы (микрокоманды $У_i$, формируемые управляющим автоматом). Выходы ОА – число С (результат выполнения операции умножения) и логические условия X_i (для управляющего автомата). Взаимодействие операционного и управляющего автоматов показано на рис. 3.4.

Когда на вход УА поступает команда «Пуск» (X_3=1), управляющий автомат начинает (в соответствии с приведенной на рис. 3.2 схемой алгоритма – ГСА) вырабатывать микрокоманды $У_i$, которые инициируют выполнение определенных микроопераций операционным


Начало
RA: = A
RB: = B
RC: = C
CT: = 0
RB [0]
0
1
RC:=RC+RA
RB:=R1(RB)
RA:=L1(RA).0
CT = 7
Да
Нет
CT:=CT+1
Конец

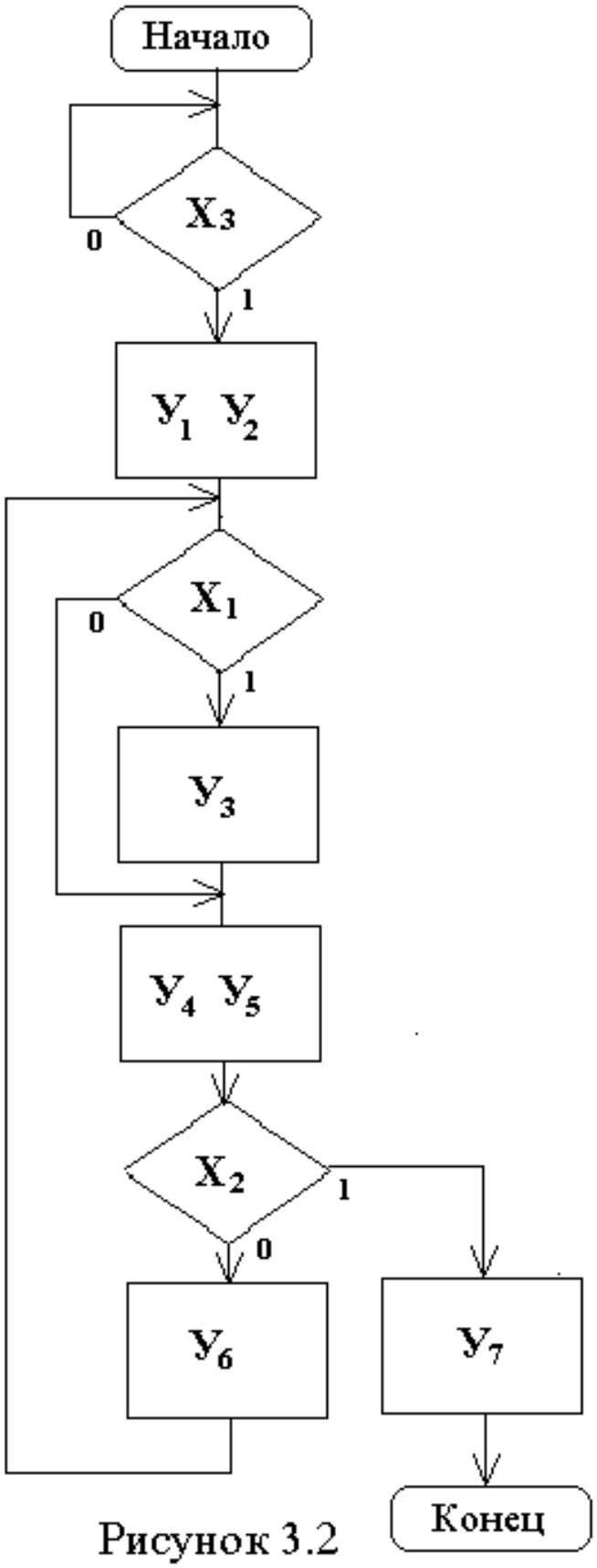


Рисунок 3.2

Рис. 3.1 *Рис. 3.2*

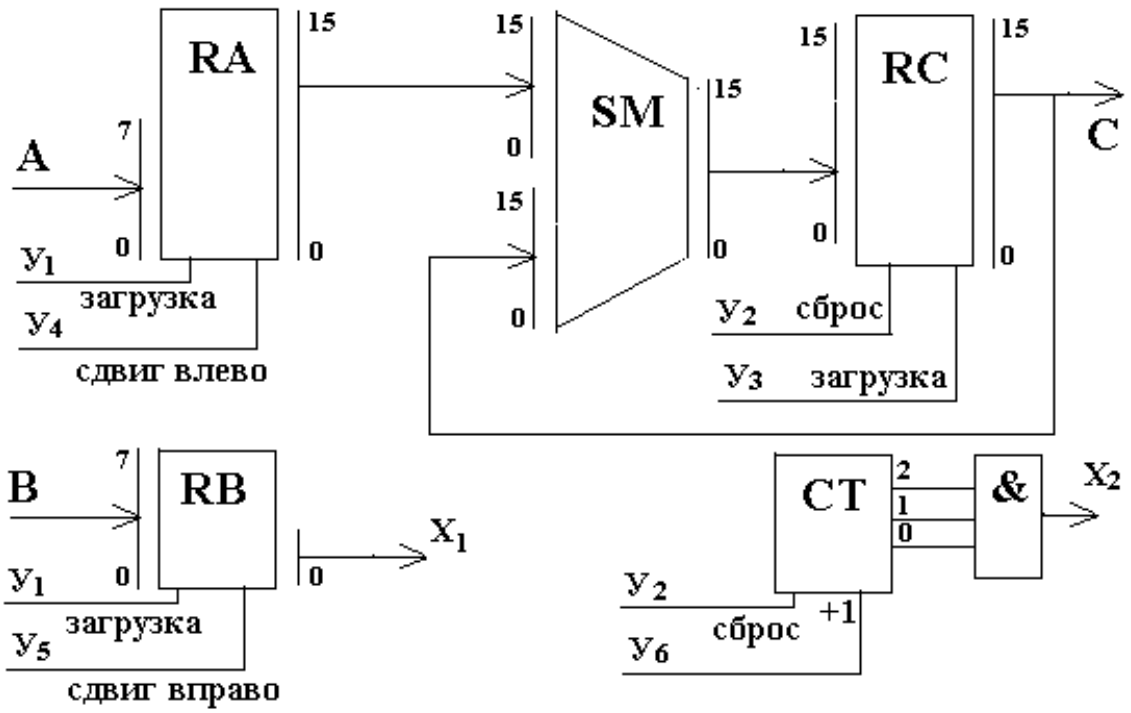


Рис. 3.3. Структурная схема операционного автомата

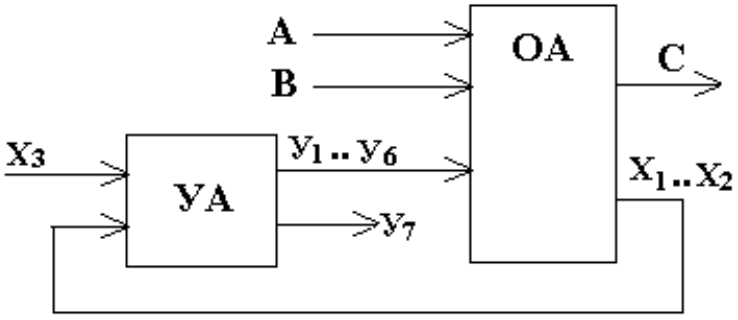


Рис. 3.4. Взаимодействие УА и ОА

автоматом. ОА при этом формирует логические условия X_i, в соответствии с которыми и ГСА управляющий автомат вырабатывает очередные микрокоманды.

Поскольку выполнение микроопераций требует определенного времени, микрокоманды должны выдаваться управляющим автоматом с интервалом времени, достаточным для выполнения самой продолжительной микрооперации. Для этого в УА вводится синхронизация. Синхронный УА анализирует логические условия X_i и вырабатывает микрокоманды $У_i$ в строго определенные моменты времени, задаваемые частотой импульсов синхронизации (машинное или автоматное время).

Управляющий автомат может быть реализован различными способами: на жесткой логике, на программируемых матрицах, на программируемой логике.

Лекция 3.

Раздел 2. Синтез цифровых автоматов на жесткой логике

Тема 2.1. Структурный синтез МПА на жесткой логике. Синтез МПА автоматов Мура на жесткой логике. Вопросы оптимизации МПА

Управляющие автоматы с жесткой логикой являются самыми быстродействующими, но схемные решения автоматов будут зависеть от их функций. Результатом синтеза автомата на жесткой логике является функциональная схема автомата, реализованная на логических элементах.

Элементная база автоматов на жесткой логике состоит из простейших логических элементов: конъюнкторов (элементы «И»), дизъюнкторов (элементы «ИЛИ»), инверторов (элементы «НЕ»), их комбинаций (элементы «И-НЕ», «ИЛИ-НЕ», «И-ИЛИ-НЕ» и т.п.), а также включает в себя и более сложные элементы – дешифраторы (DC), мультиплексоры (MS), элементы памяти (триггера).

Простейшие логические элементы реализуют вычисление простейших булевых функций. Примеры некоторых элементов, их обозначения и реализуемые ими функции приведены на рис. 3.5.

Символ «¬» обозначает инверсию, «&» – конъюнкцию, «∨» – дизъюнкцию. Часто вместо символа «&» используют «*», а в случае однобуквенных переменных – вообще его опускают, например функции, приведенные на рисунке, можно записать так: y = a b; y = ¬ (a b ∨ c d ∨ e).

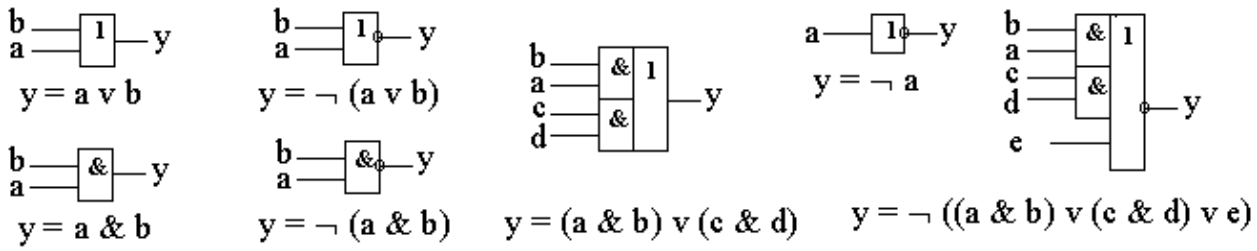


Рис. 3.5

Используя простейшие логические элементы можно строить функциональные схемы, реализующие сложные булевы функции, например построить функциональную схему, реализующую вычисление следующей системы булевых функций

$y_1 = x_2 \neg x_1 x_0 \vee \neg x_2 x_1 x_0 \vee \neg x_2 \neg x_1 \neg x_0$;

$y_2 = x_2 \neg x_1 x_0 \vee \neg x_2 x_0 \vee x_2 \neg x_0$;

$y_3 = x_2 \neg x_0 \vee \neg x_2 x_0 \vee \neg x_1$.

В приведенном примере в функциях y_1 , y_2 , y_3 встречаются одинаковые конъюнкции, которые обозначим через вспомогательные переменные Z_i и выразим через x_i:

$Z_1 = x_2 \neg x_1 x_0$; $Z_2 = \neg x_2 x_1 x_0$; $Z_3 = \neg x_2 \neg x_1 \neg x_0$;

$Z_4 = \neg x_2 x_0$; $Z_5 = x_2 \neg x_0$.

Выразим функции y_1 , y_2 , y_3 через дизъюнкцию Z_i :

$y_1 = Z_1 \vee Z_2 \vee Z_3$; $y_2 = Z_1 \vee Z_4 \vee Z_5$; $y_3 = Z_5 \vee Z_4 \vee \neg x_1$.

Функциональные схемы строятся по следующим правилам.

1. Входы схемы (переменные x_i) – с левой стороны схемы, выходы (переменные y_i) – с правой стороны.

2. Входы схемы (переменные x_i), значения которых используются в функциях с инверсиями, соединяются с входами инверторов, на выходах которых формируются значения $\neg x_i$.

3. Схемы, вычисляющие значения Z_i , строятся на конъюнкторах, на входы которых подаются соответствующие переменные x_i или $\neg x_i$.

4. Схемы, вычисляющие значения y_i , строятся на дизъюнкторах, на входы которых подаются соответствующие переменные x_i ,$\neg x_i$ или Z_i .

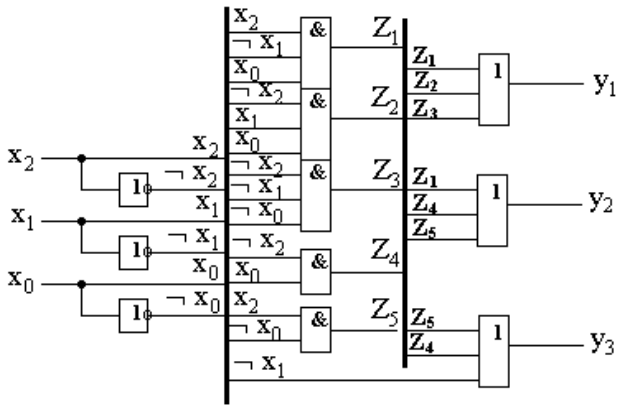


Рис. 3.6

5. Чтобы схема была легкочитаема, при ее изображении используют «шины». Шина изображается более толстой линией на схеме, в которую входят и из которой выходят тонкие линии, отображающие связи между элементами. Для идентификации этих связей непосредственно у шины ставятся либо обозначения соответствующих переменных, либо номера этих связей (первое предпочтительней). На рис. 3.6 приведена функциональная схема, реализующая функции y_1, y_2, y_3 .

Дешифратор (DC) преобразует позиционный двоичный код (двоичное число) в унитарный. Унитарный код содержит «1» только в одном разряде. Входы дешифратора (в отличие от входов простейших логических элементов) характеризуются своим весом (как разряды двоичного числа): младший разряд имеет вес 1, следующий разряд – вес 2, n-разряд имеет вес 2^{n-1}. Веса разрядов указываются на входах дешифратора. Полный дешифратор имеет 2^n выходов, имеющих нумерацию от 0 до $2^n - 1$. Таким образом, дешифратор с двумя входами имеет 4 выхода, с тремя входами – 8 выходов и т.д. На рис. 3.7 приведено изображение дешифратора с двумя входами $x_1 x_0$.

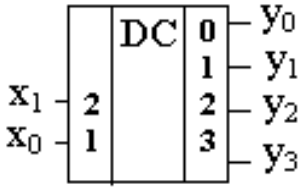


Рис. 3.7

Значения $x_1\, x_0$ – двоичное число, указывающее номер выхода дешифратора, на котором будет сформирован сигнал «1». Функции y_i, на выходе дешифратора имеют вид

$y_0 = \neg\, x_1\ \neg\, x_0$; $y_2 = x_1\ \neg\, x_0$;

$y_1 = \neg\, x_1\ x_0$; $y_3 = x_1\, x_0$.

Функцию дешифратора можно пояснить также следующей таблицей истинности (табл. 3.3).

Таблица 3.3

x_1	x_0	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Многие дешифраторы имеют еще один дополнительный вход – «разрешение работы дешифратора». На рис. 3.8 приведен пример такого дешифратора, который называется дешифратор – демультиплексор.

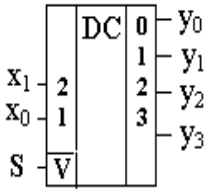


Рис. 3.8

Если на вход V подается «0», то на всех выходах дешифратора – нули; если на вход V подается «1», то дешифратор работает в соответствии с приведенной таблицей истинности, а функции y_i, на выходе дешифратора имеют вид

$y_0 = S\ (\neg\, x_1\ \neg\, x_0)$; $y_2 = S\ (x_1\ \neg\, x_0)$;

$y_1 = S\ (\neg\, x_1\, x_0)$; $y_3 = S\ (x_1\, x_0)$.

Мультиплексор (MS) выполняет функции электронного переключателя, позволяющего выбрать один из нескольких источников сигнала для одного приемника. Пример простейшего мультиплексора приведен на рис. 3.9.

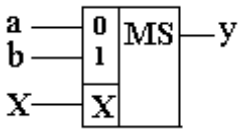


Рис. 3.9

У этого мультиплексора один управляющий вход (X) и два информационных (a и b). Если на управляющий вход X подать «0», то на выходе MS значение y = a, если на X подать «1», то значение y = b. Другими словами, приведенный на рисунке MS – это переключатель на два положения, которым выбирается один из двух источников информации (a и b) для приемника У. Функцию MS можно описать выражением

$$y = \neg x\, a \vee x\, b.$$

Количество управляющих входов у MS может быть 2, 3 и более. В этом случае MS – это переключатель не на два, а на 2^n положений (где n – количество управляющих входов), позволяющий выбирать один из 2^n информационных входов. Номер этого информационного входа задается двоичным числом, подаваемым на управляющие входы: $x_{n-1} \ldots x_1\, x_0$. На рис. 3.10 приведен пример MS с четырьмя информационными входами (a, b, c, d) и двумя управляющими ($x_1 x_0$).

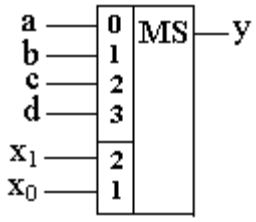


Рис. 3.10

Функцию MS такого можно описать выражением

$y = a\,(\neg x_1 \neg x_0) \vee b\,(x_1 \neg x_0) \vee c\,(\neg x_1\, x_0) \vee d\,(x_1\, x_0)$.

Используя описанные выше функциональные элементы, можно строить так называемые комбинационные логические схемы (или цифровые автоматы комбинационного типа, автоматы без памяти), т.е. схемы, реализующие булевы функции. В микропрограммных автоматах, как правило, используются элементы памяти, на которых строится память состояний автомата.

Простейшим элементом памяти является статический асинхронный RS-триггер. RS-триггер – это элемент, имеющий два устойчивых состояния, которые обычно обозначают как «0» и «1» (рис. 3.11).

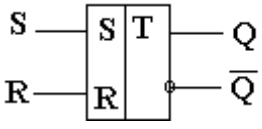


Рис. 3.11

Для установки RS-триггера в одно из устойчивых состояний «0» или «1» у него имеется два входа: R – для установки в состояние «0» и S – для установки в состояние «1». Выходы RS-триггера обычно обозначаются как Q (прямой выход) и ¬Q (инверсный выход). Значение на выходе Q («0» или «1») обычно отождествляют с состоянием триггера «0» или «1». Если через Q_t обозначить значение на выходе Q в момент времени t, а через Q_{t+1} – обозначить значение на выходе Q в момент времени t+1, то поведение RS-триггера можно описать следующей таблицей (табл. 3.4).

Из таблицы видно, что при R=0 и S=1 происходит установка триггера в состояние «1» (Q_{t+1} = 1), при R=1 и S=0 происходит установка триггера в состояние «0» (Q_{t+1} = 0). Эти установки происходят независимо от предыдущего состояния триггера Q_t. При R=0 и S=0 триггер сохраняет свое состояние, а комбинация R=1 и S=1 является недопустимой, так как одновременно установить триггер в состояния «0» и «1» невозможно. На рис. 3.11 приведено изображение статического асинхронного RS-триггера.

Таблица 3.4

R	S	Q_t	Q_{t+1}	Комментарий
0	0	0	0	Хранение «0»
0	0	1	1	Хранение «1»
0	1	0	1	Установка в «1»
0	1	1	1	Установка в «1»
1	0	0	0	Установка в «0»
1	0	1	0	Установка в «0»
1	1	0	X	Запрещенная комбинация R и S
1	1	1	X	Запрещенная комбинация R и S

Недостатком асинхронного RS-триггера является то, что изменение его состояния возможно в произвольный момент времени сразу же после изменения сигналов на его входах R и S, поэтому асинхронный RS-триггер часто используют как элемент памяти для построения синхронных триггеров. В синхронных триггерах изменение состояния возможно только в момент подачи на специальный вход «C» импульса синхронизации. Существуют триггеры, синхронизируемые импульсом C и фронтом импульса C. Фронтом импульса синхронизации называют переход значения C из «0» в «1» (положительный фронт) или из «1» в «0» (отрицательный фронт). Триггеры, синхронизируемые фронтом импульса C, нашли наибольшее применение в цифровых устройствах.

На рис. 3.12 приведено изображение двух RS-триггеров, синхронизируемых фронтом импульса синхронизации C. Наклонная черта на входе C показывает, по какому фронту импульса C – положительному (/) или отрицательному (\) – происходит переход триггера из одного состояния в другое.

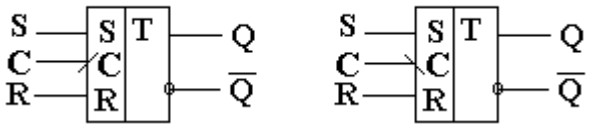


Рис. 3.12

Кроме RS-триггеров широкое применение в цифровых устройствах нашли D-триггеры и JK-триггеры (также синхронизируемые фронтом импульса С).

D-триггер, в отличие от RS-триггера, имеет всего один информационный вход – D. Синхронный D-триггер изменяет свое состояние также в момент поступления на его вход «С» импульса синхронизации С. Новое состояние Q_{t+1} не зависит от предыдущего Q_t , а зависит только от значения на входе D в момент поступления импульса С на вход С триггера. JK-триггер называется универсальным, так как на его основе можно реализовать функции других триггеров. Закон его функционирования можно представить в виде табл. 3.5.

На рис. 3.13 приведены изображения D-триггера, синхронизируемого положительным фронтом импульса С, и JK-триггера, синхронизируемого отрицательным фронтом импульса С.

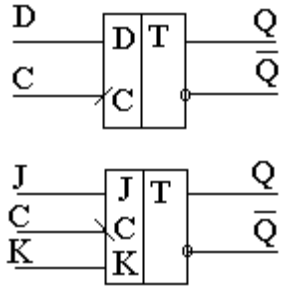


Рис. 3.13

Автоматы с жесткой логикой обычно строятся или как автоматы Мура, или как автоматы Мили. Поскольку функционирование и тех, и других зависит не только от входных сигналов (логических условий X_i), но и от текущего состояния автомата, необходимым элементом УА является память состояний. В качестве элементов памяти обычно используются триггеры различных типов: RS-, D-, JK-, T-триггеры, синхронизируемые фронтом (положительным или отрицательным) импульса синхронизации С. По фронту импульса С автомат переходит из одного состояния в другое. Выходные сигналы автомата – микрокоманды – также вырабатываются синхронно с импульсом С.

Таблица 3.5

K	J	Q_t	Q_{t+1}	Комментарий
0	0	0	0	Хранение «0»

0	0	1	1	Хранение «1»
0	1	0	1	Установка в «1»
0	1	1	1	Установка в «1»
1	0	0	0	Установка в «0»
1	0	1	0	Установка в «0»
1	1	0	1	Изменение состояния на противоположное
1	1	1	0	Изменение состояния на противоположное

Различие автоматов Мура и Мили следующее: в автомате Мура вырабатываемая автоматом микрокоманда Y_i зависит только от текущего состояния автомата, а в автомате Мили – от текущего состояния и значений логических условий на входе автомата.

3.1.1. СИНТЕЗ АВТОМАТА МУРА ПО ГСА. ПРОСТЕЙШАЯ РЕАЛИЗАЦИЯ

3.1.1.1. Разметка состояний автомата Мура по ГСА

Каждой операторной вершине ГСА автомата Мура соответствует определенное состояние – ai . Начальное состояние автомата соответствует началу ГСА, а точнее – входу в первую вершину ГСА. Первая вершина ГСА обычно соответствует логическому условию «Пуск», поэтому начальное состояние можно отметить на входе этой вершины. Поскольку этому состоянию не соответствует никакая операторная вершина, то и автомат в начальном состоянии не выдает никаких микрокоманд.

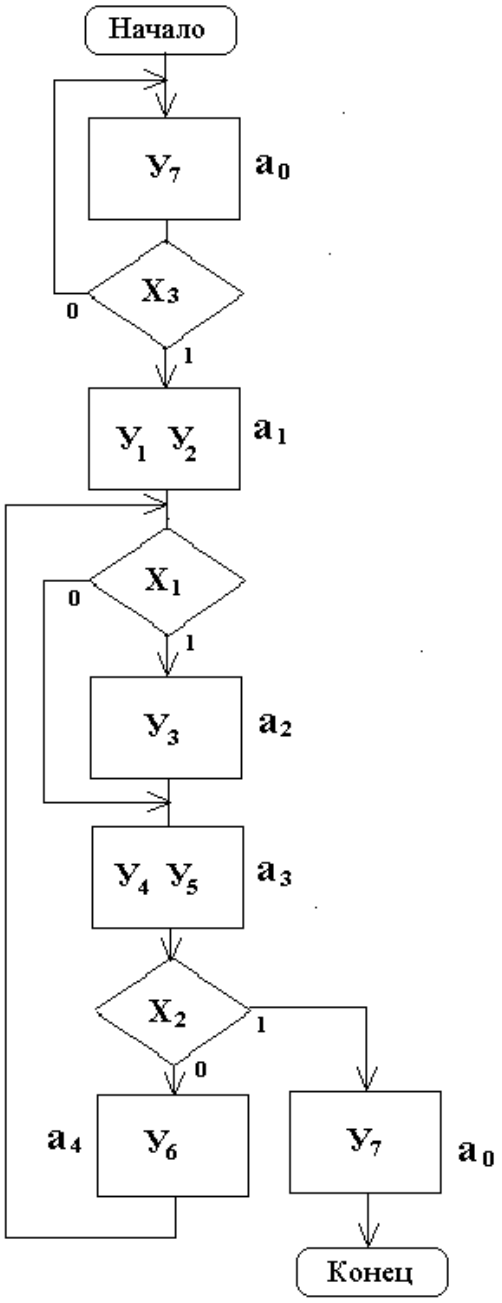


Рис. 3.14

При такой отметке начального состояния конечное состояние автомата соответствует завершению ГСА (перед вершиной «Конец»).

Для корректной работы автомата необходимо, чтобы после завершения выполнения алгоритма автомат вернулся в начальное состояние. Таким образом можно совместить начальное и конечное состояния автомата и обозначить их одинаково a_0.

Часто дополнительно введенную операторную вершину, соответствующую микрокоманде «Операция выполнена», отмечают как конечное состояние автомата (a_0), а так как конечное состояние автомата должно совпадать с начальным, в ГСА вводится еще одна дополнительная операторная вершина, которая отмечается состоянием a_0, так же, как конечная (рис. 3.14). Такое преобразование ГСА имеет определенные преимущества: если

автомат «свободен», он не «молчит», а выдает микрокоманду «Операция выполнена», что говорит о его готовности к работе.

Остальным операторным вершинам соответствуют состояния, которые можно пронумеровать так: a_1, a_2, a_3 и т.д. (см. рис. 3.14).

Таким образом, ГСА автомата Мура (рис. 3.14 в нашем примере) отличается от исходной ГСА (см. рис. 3.2) еще одной дополнительной вершиной с состоянием a_0. Обе вершины, помеченные состоянием a_0, можно мысленно совместить, так как после завершения операции автомат должен вернуться в начальное состояние

3.1.1.2. Построение графа переходов автомата Мура (по ГСА рис. 3.14)

Вершины графа соответствуют состояниям автомата, дуги – переходам из состояния a_m в состояние a_s. У вершин графа записываются микрокоманды, соответствующие состояниям, в начале дуги – логические условия, определяющие переход из состояния a_m в состояние a_s (рис. 3.15).

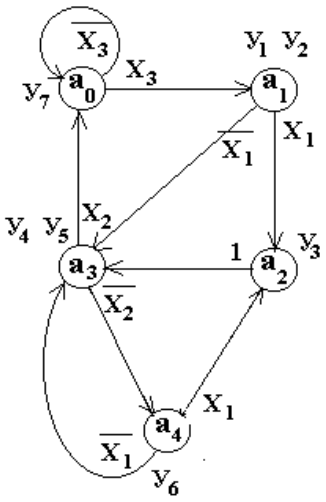


Рис. 3.15

3.1.1.3. Построение прямой таблицы переходов автомата Мура

Прямая таблица переходов (табл. 3.6) строится по графу переходов (см. рис. 3.15). Количество строк в таблице равно количеству переходов в графе. В столбце a_m записываются состояния, из которых начинается переход, в столбце a_s – состояния, в которые перешел автомат из a_m. В столбце Ya_s записываются Y_i – микрокоманды, вырабатываемые автоматом в состоянии a_s. В столбце Xa_ma_s записываются логические условия (их конъюнкция), обеспечивающие переход из состояния a_m в состояние a_s. Прямая таблица позволяет проверить полноту переходов, показанных на графе переходов: дизъюнкция всех Xa_ma_s из состояния a_m должна быть равна «1». В нашем примере дизъюнкция всех Xa_0a_s равна $\neg x_3 \vee x_3 = 1$.

Таблица 3.6

№ п/п	a_m	a_s	Ya_s	Xa_ma_s
1	a_0	a_0	y_7	$\neg x_3$
2		a_1	y_1, y_2	x_3
3	a_1	a_2	y_3	x_1
4		a_3	y_4, y_5	$\neg x_1$
5	a_2	a_3	y_4, y_5	1
6	a_3	a_4	y_6	$\neg x_2$
7		a_0	y_7	x_2
8	a_4	a_2	y_3	x_1
9		a_3	y_4, y_5	$\neg x_1$

3.1.1.4. Кодирование состояний автомата. Выбор элементов памяти

Так как поведение автомата всегда зависит от его текущего состояния a_m, необходимо хранить код состояния a_m в памяти состояний автомата. Объем памяти зависит от способа кодирования состояний. При минимальном кодировании каждому состоянию соответствует число в двоичном представлении, причем количество разрядов этого числа n определяется выражением $n =] \log_2 | A | [$. Другой крайний случай – унитарное кодирование, при котором $n = | A |$. От выбранного способа кодирования и самого кодирования состояний может зависеть сложность схемы автомата.

Используем для нашего примера минимальное кодирование состояний. Так как автомат имеет пять состояний, то минимальное количество элементов памяти

$n =] \log_2 | A | [=] \log_2 5 [= 3$.

Выберем в качестве элементов памяти D-триггеры. Для нашего примера их количество равно трем. Обозначим их как T_2, T_1, T_0, причем T_2 соответствует старшему разряду кода состояний. Выходы триггеров обозначаются соответственно Q_2, Q_1, Q_0. Значение числа $Q_2Q_1Q_0$ на этих выходах есть код состояния автомата.

Закодируем состояния автомата произвольно ($Ka_i = Q_2Q_1Q_0$):

К $a_0 = 100$. К $a_1 = 001$. К $a_2 = 010$.

К $a_3 = 000$. К $a_4 = 011$.

3.1.1.5. Обратная структурная таблица автомата Мура

Обратная структурная таблица автомата Мура (табл. 3.7) строится на основе прямой таблицы переходов путем упорядочивания строк по столбцу a_s и добавления столбцов:

Ka_m – код состояния a_m;

Ka_s – код состояния a_s.

$F a_m a_s$ – функции управления элементами памяти при переходе из состояния a_m в состояние a_s. Поскольку в качестве элементов памяти используем D-триггеры, в этом

столбце записываем только D_i, соответствующие триггерам, которые необходимо установить в состояние «1», чтобы обеспечить переход в состояние с кодом K a_s.

Таблица 3.7

№ п/п	a_m	Ka_m	a_s	Ka_s	Xa_ma_s	Ya_s	Fa_ma_s
1	a_0	100	a_0	100	$\neg x_3$	y_7	D_2
2	a_3	000			x_2		D_2
3	a_0	100	a_1	001	x_3	y_1, y_2	D_0
4	a_1	001	a_2	010	x_1	y_3	D_1
5	a_4	011			x_1		D_1
6	a_1	001	a_3	000	$\neg x_1$	y_4, y_5	-
7	a_2	010			1		-
8	a_4	011			$\neg x_1$		-
9	a_3	000	a_4	011	$\neg x_2$	y_6	$D_1 D_0$

3.1.1.6. Функции управления элементами памяти и функции выходов автомата

Функции управления элементами памяти записываются по обратной структурной таблице автомата:

$D_i = F(a_m, X a_m a_s)$.

Смысл этого выражения следующий (например для D_2): значение функции D_2 должно быть равно 1 (см. обратную структурную таблицу) в двух случаях (1-я и 2-я строки таблицы): если автомат находился в состоянии a_0, а значение $x_3 = 0$; если автомат находился в состоянии a_3, а значение $x_2 = 1$. Таким образом, функция D_2 имеет вид

$$D_2 = a_0 \neg x_3 \vee a_3 x_2.$$

Остальные функции D_1 и D_0 записываются аналогично:

$$D_1 = a_1 x_1 \vee a_4 x_1 \vee a_3 \neg x_2 = x_1 (a_1 \vee a_4) \vee a_3 \neg x_2.$$

$$D_0 = a_0 x_3 \vee a_3 \neg x_2.$$

Функции выходов также записываются по обратной структурной таблице автомата:

$$y_i = F(a_s).$$

Так как y_i в автомате Мура зависят только от текущего состояния автомата, то для нашего примера они имеют вид

$$y_1 = y_2 = a_1. \quad y_3 = a_2. \quad y_4 = y_5 = a_3. \quad y_6 = a_4. \quad y_7 = a_0.$$

Это означает следующее: в состоянии a_1 автомат вырабатывает микрокоманды y_1 и y_2, в состоянии a_2 – микрокоманду y_3 и т.д.

3.1.1.7. Структурная схема автомата Мура на жесткой логике

Структурная схема автомата Мура состоит из следующих цифровых узлов (рис. 3.16).

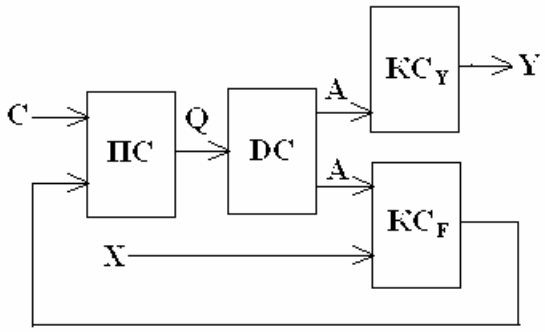


Рис. 3.16

Память состояний (ПС), дешифратор состояний (DC), комбинационная схема формирования сигналов управления элементами памяти состояний ($КС_F$), комбинационная схема формирования выходных сигналов автомата ($КС_У$). Взаимодействие узлов автомата следующее. Автомат находится в некотором состоянии a_m, код которого Ka_m в виде значений Q на выходе триггеров памяти состояний (ПС) подается на вход дешифратора состояний (DC), на выходе которого собственно и формируются значения переменных a_m. На выходах комбинационной схемы $КС_У$ формируются микрокоманды У, а на выходах схемы $КС_F$ формируются значения функций управления элементами памяти, которые обеспечивают переход автомата в новое состояние a_s при поступлении импульса синхронизации С на вход синхронизации ПС.

3.1.1.8. Функциональная схема автомата Мура на жесткой логике

Функциональная схема автомата Мура состоит из следующих цифровых узлов.

- Память состояний. В нашем примере – триггеры T2, T1, T0.
- Дешифратор состояний DC. Дешифратор необходим для преобразования двоичного кода состояний автомата Ka_m в унитарный, соответствующий переменным a_i, используемым в записанных выше функциях. В нашем примере дешифратор DC имеет 3 входа и 8 выходов. На вход DC подается Ka_m – код состояния a_m, а на выходах DC формируется унитарный код состояния автомата a_m: единица на i-том выходе дешифратора DC формируется при $Ka_m = i$.
- Комбинационная схема формирования сигналов управления элементами памяти состояний автомата реализует функции

  $D_i = F(a_m, X a_m a_s)$..
- Комбинационная схема формирования выходных сигналов автомата реализует функции $y_i = F(a_s)$.

В функциональной схеме (рис. 3.17) использованы «шины». Шины представляют собой множество соединений схемы, изображенных в виде одной утолщенной линии. Вход в шину и выход из нее конкретного соединения обозначаются либо одним и тем же числом, либо содержательным обозначением сигнала, передаваемого по этому соединению. Применение шин в схемах позволяет избежать большого числа пересечений на схеме и делает ее более простой для чтения.

С целью упрощения схемы в ней не показаны элементы, обеспечивающие установку автомата в начальное состояние a_0 с кодом $Ka_0 = 100$. Этот вопрос будет рассмотрен ниже.

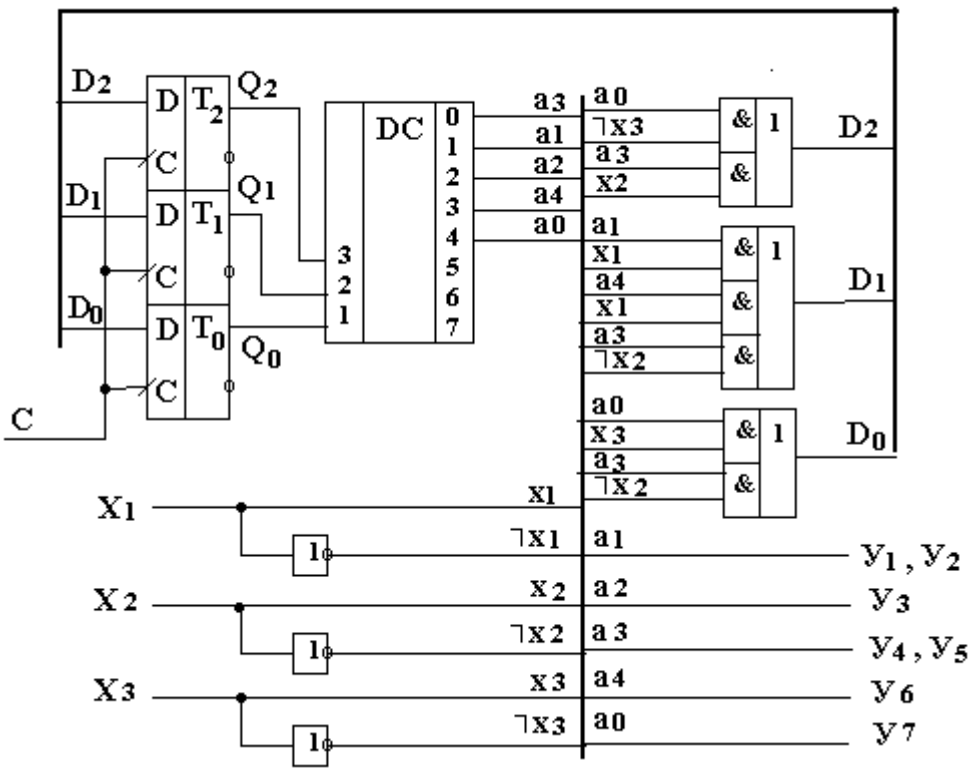


Рис. 3.17. Функциональная схема Мура

На рис. 3.18 приведены временные диаграммы, поясняющие работу автомата Мура. Находясь в некотором состоянии a_i, автомат вырабатывает выходной сигнал (микрокоманду) y_j, соответствующий этому состоянию. В это же время формируются сигналы управления элементами памяти D_i, которые определяют следующее состояние автомата в зависимости от текущего и значений логических условий x_i. При поступлении на вход синхронизации автомата положительного фронта импульса С автомат переходит в новое состояние, определяемое значениями D_i на входах триггеров T_2 T_1 T_0.

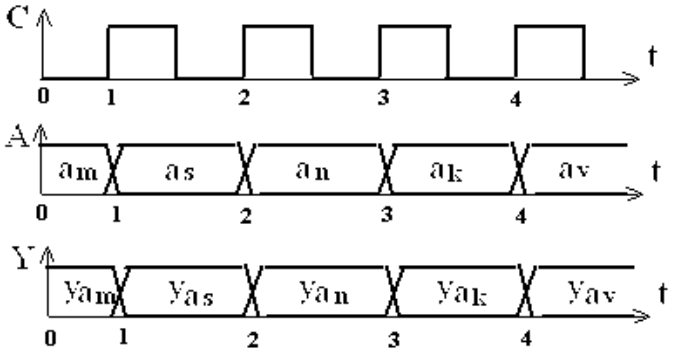


Рис. 3.18

Лекция 4.

Тема 2.2. Структурный синтез МПА на жесткой логике. Синтез МПА автоматов Мили на жесткой логике. Вопросы оптимизации МПА

3.1.2.1. Разметка состояний автомата Мили по ГСА

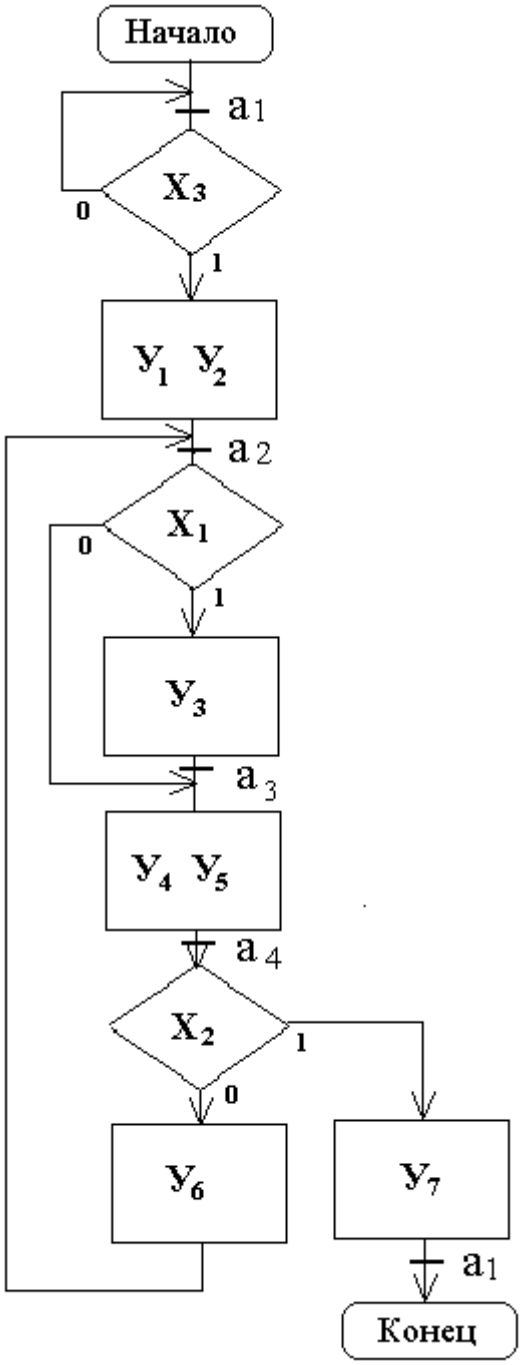


Рис. 3.19

В отличие от автомата Мура состояния автомата Мили не соответствуют операторным вершинам ГСА, а отмечаются на дугах ГСА перед вершинами, следующими за операторными. Исключение составляет начальное (оно же конечное) состояние автомата. Его удобно обозначать символом a_0 или a_1. Символом a_0 или a_1 отмечают вход вершины, следующей за начальной, и вход конечной вершины ГСА. Входы всех остальных вершин, следующих за операторными, также отмечаются символами: a_1, a_2, … .

Используем для примера ГСА УА (см. рис. 3.2) для синтеза автомата Мили. Обозначим начальное состояние как a_1, а остальные – a_2, a_3, a_4.

В случае, когда в вершину, следующую за операторной, входит более чем одна дуга, состояние необходимо отметить на дуге так, чтобы для всех входящих дуг соблюдалось правило разметки состояний. На ГСА (рис. 3.19) это состояния a_2 и a_3. Состояние a_2 необходимо отметить ниже входящей слева стрелки, а состояние a_3 – выше входящей справа стрелки. В первом случае в a_2 сошлись пути из двух операторных вершин, а во втором – путь из a_2 не приводит в состояние a_3 (этот переход был бы «пустым», без прохода через операторную вершину), а приводит в состояние a_4 (после операторной вершины).

3.1.2.2. Построение графа переходов автомата Мили по ГСА

Вершины графа соответствуют состояниям автомата, дуги – переходам из состояния a_m в состояние a_s. У выхода дуги из вершины графа a_m записываются логические условия, определяющие переход из состояния a_m в состояние a_s , а у входа дуги в состояние a_s – микрокоманды, вырабатываемые автоматом при переходе из состояния a_m в состояние a_s (рис. 3.20).

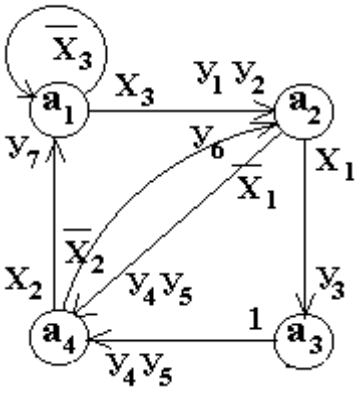


Рис. 3.20

3.1.2.3. Построение прямой таблицы переходов автомата Мили

Прямая таблица переходов (табл. 3.8) строится по графу переходов (см. рис. 3.20).

Количество строк в таблице равно количеству переходов в графе переходов. В столбце a_m записываются состояния, из которых начинается переход, в столбце a_s – состояния, в которые перешел автомат из состояния a_m.

Таблица 3.8

№ п/п	a_m	a_s	Xa_ma_s	Ya_ma_s
1	a_1	a_1	$\neg x_3$	-
2		a_2	x_3	$y_1 y_2$
3	a_2	a_3	x_1	y_3
4		a_4	$\neg x_1$	$y_4 y_5$
5	a_3	a_4	1	$y_4 y_5$

6 7	a_4	a_1 a_2	x_2 $\neg x_2$	y_7 y_6

В столбце $Y\, a_m a_s$ записываются Y_i – микрокоманды, вырабатываемые автоматом при переходе из состояния a_m в состояние a_s. В столбце $Xa_m a_s$ записываются логические условия (их конъюнкция), обеспечивающие переход из состояния a_m в состояние a_s.

Прямая таблица позволяет проверить полноту переходов, показанных на графе переходов: дизъюнкция всех $Xa_m a_s$ из состояния a_m должна быть равна «1» ($\cup Xa_m a_s = 1$). В нашем примере дизъюнкция всех $Xa_1 a_s$ равна $\cup Xa_1 a_s = Xa_1 a_1 \vee Xa_1 a_2 = \neg x_3 \vee x_3 = 1$. Аналогично $\cup Xa_2 a_s = Xa_2 a_3 \vee Xa_2 a_4 = x_1 \vee \neg x_1 = 1$, $\cup Xa_3 a_s = Xa_3 a_4 = 1$,

$\cup Xa_4 a_s = Xa_4 a_1 \vee Xa_4 a_2 = x_2 \vee \neg x_2 = 1$.

3.1.2.4. Кодирование состояний автомата. Выбор элементов памяти

Кодирование состояний автомата Мили производится так же, как и автомата Мура.

Используем для нашего примера минимальное кодирование состояний. Так как автомат имеет четыре состояния, то минимальное количество элементов памяти

$n =] \log_2 | A | [=] \log_2 4 [= 2$.

Выберем в качестве элементов памяти синхронные RS-триггеры. Для нашего примера их количество равно двум. Обозначим их как T_1, T_0, причем T_1 соответствует старшему разряду кода состояний. Выходы триггеров обозначаются соответственно Q_1, Q_0. Значение числа Q_1Q_0 на этих выходах есть код состояния автомата. Закодируем состояния автомата произвольно ($Ka_i = Q_1Q_0$):

$$K\, a_1 = 00, \quad K\, a_2 = 01, \quad K\, a_3 = 10, \quad K\, a_4 = 11.$$

3.1.2.5. Обратная структурная таблица автомата Мили

Обратная структурная таблица (табл. 3.9) автомата Мили строится так же, как и для автомата Мура.

Таблица 3.9

№ п/п	a_m	Ka_m	a_s	Ka_s	$X\, a_m a_s$	$Y\, a_m a_s$	$F\, a_m a_s$
1 2	a_1 a_4	00 11	a_1	00	$\neg x_3$ x_2	- y_7	- $R_1 R_0$

3 4	a_1 a_4	00 11	a_2	01	x_3 $\neg x_2$	$y_1 y_2$ y_6	S_0 R_1
5	a_2	01	a_3	10	x_1	y_3	$S_1 R_0$
6 7	a_2 a_3	01 10	a_4	11	$\neg x_1$ 1	$y_4 y_5$ $y_4 y_5$	S_1 S_0

При заполнении столбца F $a_m a_s$ следует обратить внимание на то, что управление RS-триггерами отличается от управления D-триггерами. Если состояние некоторых разрядов RS-триггеров памяти автомата не изменяется при переходе из a_m в a_s, то нет необходимости вырабатывать соответствующие сигналы управления S=1 или R=1, так как комбинация S=0 и R=0 соответствует режиму хранения в RS-триггерах. Например, в третьей строке структурной таблицы описан переход из состояния a_1 с кодом Ka_m = 00 (Q_1 =0, Q_0 =0) в состояние a_2 с кодом Ka_s = 01 (Q_1 =0, Q_0=1). Чтобы обеспечить переход из a_1 в a_2, нужно сохранить значение Q_1 =0, а в младший разряд памяти состояний Q_0 – установить «1», поэтому в столбце $Fa_m a_s$ третьей строки записано «S_0», что означает S_0=1. При поступлении на вход синхронизации памяти состояний синхроимпульса С триггер T_1 не изменит своего состояния (так как R_1 =0 и S_1 =0), а триггер T_0 перейдет из состояния «0» в состояние «1» (так как R_0=0, а S_0=1). В столбце $Fa_m a_s$ не будем записывать R_i = 0 или S_i = 0, а будем записывать только те R_i и S_i, значения которых должны быть равны «1».

3.1.2.6. Функции управления элементами памяти и функции выходов автомата

Функции управления элементами памяти записываются по обратной структурной таблице автомата:

$$R_i = F(a_m, X a_m a_s);$$
$$S_i = F(a_m, X a_m a_s).$$

Смысл этих выражений следующий (например для R_1). Значение функции R_1 должно быть равно «1» (см. обратную структурную таблицу) в двух случаях (2-я и 4-я строки таблицы): если автомат находился в состоянии a_4, а значение x_2 = 1, или, если автомат находился в состоянии a_4, а значение $\neg x_2$ =1. Таким образом, функция R_1 имеет вид

$$R_1 = a_4 x_2 \vee a_4 \neg x_2 = a_4.$$

Остальные функции R_i и S_i записываются аналогично:

$$S_1 = a_2 x_1 \vee a_2 \neg x_1 = a_2;$$
$$R_0 = a_4 x_2 \vee a_2 x_1;$$
$$S_0 = a_1 x_3 \vee a_3.$$

Функции выходов автомата $Ya_m a_s$ также записываются по обратной структурной таблице автомата:

$$y_i = F(a_m, X a_m a_s).$$

Смысл этого выражения следующий (например для y_4). Значение функции y_4 должно быть равно «1» (см. обратную структурную таблицу) при переходе автомата из состояний a_2 или a_3 в состояние a_4 (6-я и 7-я строки таблицы). Иначе: если автомат находился в состоянии a_2, а значение $\neg x_2 = 1$, или, если автомат находился в состоянии a_3, то при переходе автомата из состояний a_2 или a_3 в состояние a_4 значение y_4 должно быть равно «1». Таким образом, функция y_4 имеет вид

$$y_4 = y_5 = a_2 \neg x_1 \vee a_3 .$$

Остальные функции выходов имеют вид

$y_1 = y_2 = a_1 x_3$; $y_3 = a_2 x_1$;

$y_6 = a_4 \neg x_2$; $y_7 = a_4 x_2$.

3.1.2.7. Структурная схема автомата Мили на жесткой логике

Структурная схема автомата Мили состоит из следующих цифровых узлов (рис. 3.21): память состояний (ПС), дешифратор состояний (DC), комбинационная схема формирования сигналов управления элементами памяти состояний ($КС_F$), комбинационная схема формирования выходных сигналов автомата ($КС_У$). Взаимодействие узлов автомата следующее. Автомат находится в некотором состоянии a_m, код которого Ka_m в виде значений Q на выходе триггеров памяти состояний (ПС) подается на вход дешифратора состояний (DC), на выходе которого собственно и формируются значения переменных a_m. На выходах комбинационной схемы $КС_F$ формируются значения функций управления элементами памяти Fa_ma_s, которые обеспечивают переход автомата в новое состояние a_s при поступлении импульса синхронизации C на вход синхронизации ПС, а на выходах комбинационной схемы $КС_У$ при этом формируются значения функций выходов автомата Ya_ma_s.

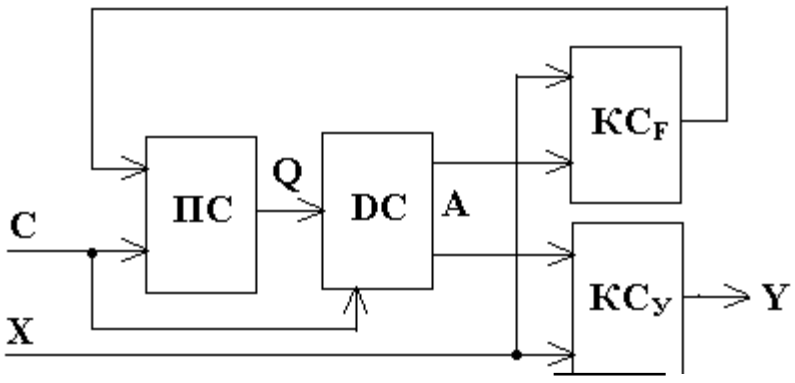


Рис. 3.21

3.1.2.8. Функциональная схема автомата Мили на жесткой логике

Функциональная схема автомата Мили состоит из следующих цифровых узлов.

- Память состояний. В нашем примере – два триггера T_1 T_0.

- Дешифратор состояний DC. В нашем примере – дешифратор DC имеет 2 входа и 4 выхода. На вход DC подается Ka_m – код состояния a_m , а на выходах DC формируется унитарный код состояния автомата а $_m$: единица на i-том выходе дешифратора DC формируется при $Ka_m = i$.

- Комбинационная схема формирования сигналов управления элементами памяти состояний автомата. В нашем примере реализует функции

$$R_i = F(a_m , X a_m a_s);$$

$$S_i = F(a_m , X a_m a_s).$$

- Комбинационная схема формирования выходных сигналов автомата. В нашем примере реализует функции y i = F(a m , X a m a s).

- Память логических условий. В нашем примере это три D-триггера T_{x1} T_{x2} T_{x3} . Значения логических условий на входе автомата Мили могут измениться во время формирования микрокоманды у $_i$, что может привести к формированию «ложных» (лишних) микрокоманд, поэтому необходимо зафиксировать значения x_i , поступившие на входы автомата к моменту прихода импульса синхронизации, на время формирования микрокоманд y_i. Таким образом, по положительному фронту импульса синхронизации С значения x $_i$, запоминаются на триггерах T_{xi}, при С = 1 формируются микрокоманды y_i и функции управления элементами памяти R_i и S_i , а по отрицательному фронту импульса С автомат переходит в следующее состояние, определяемое значениями R_i и S_i на входах памяти состояний автомата.

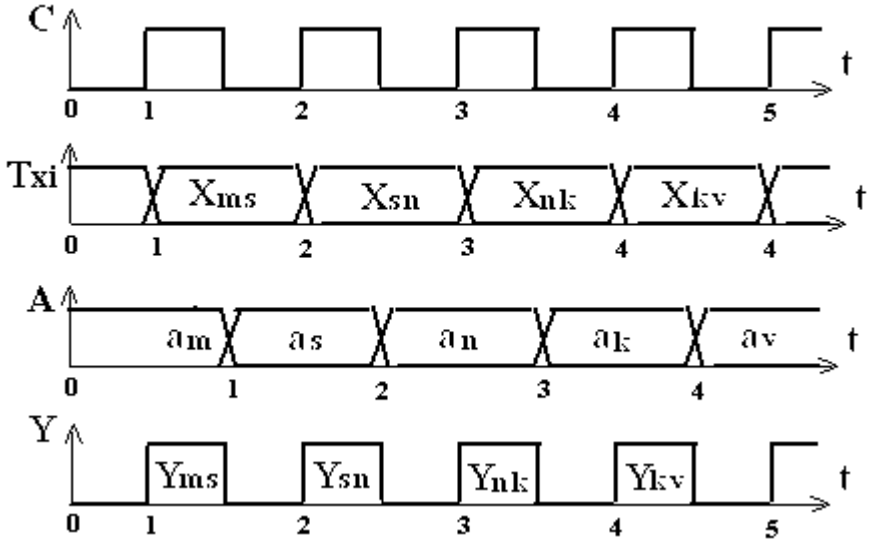


Рис. 3.22

Временная диаграмма, поясняющая работу автомата Мили, приведена на рис. 3. 22, функциональная схема – на рис. 3.23.

Из временной диаграммы видно, что по положительному фронту импульса синхронизации С значения логических условий X на входе автомата запоминаются на триггерах T_{xi}. Значения логических условий с выходов этих триггеров и текущее состояние автомата а $_m$ используются для вычисления $У_{m\,s}$ – микрокоманд, вырабатываемых автоматом на переходе из состояния а $_m$ в состояние а $_s$. Дешифратор состояний DC (см. рис. 3.12)

имеет вход разрешения V: при V=1 дешифратор выдает на одном из своих выходов значение «1», при V=0 – на всех выходах DC логический «0». Это означает, что при C=0 (а значит и V=0) все выходные сигналы автомата $У_{m\ s}$ равны нулю. Автомат вырабатывает микрокоманды только при C=1.

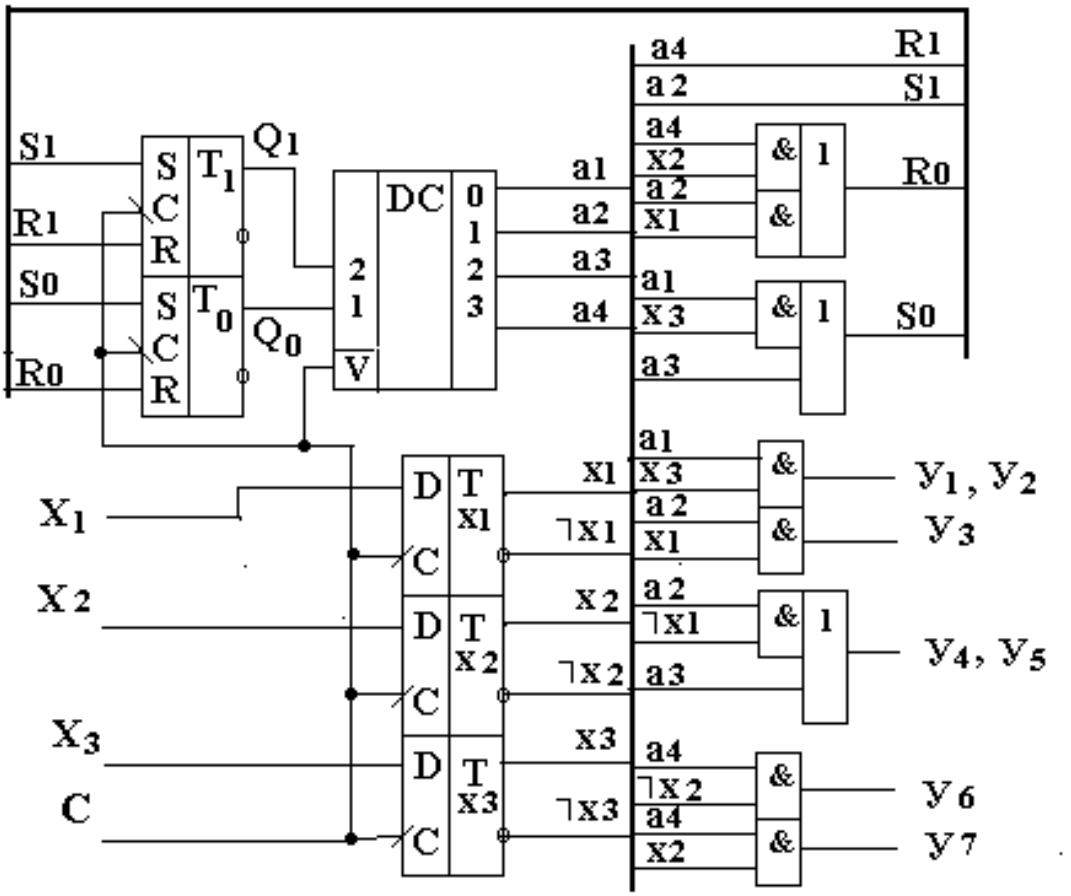


Рис. 3.23. Функциональная схема автомата Мили

Лекция 5.

Раздел 3. **Синтез цифровых автоматов на программируемых матрицах (ПЛМ)**

Тема 3.1. Структурный синтез МПА Мили на ПЛМ. Вопросы оптимизации МПА

МАТРИЧНАЯ РЕАЛИЗАЦИЯ КОМБИНАЦИОННЫХ СХЕМ (КС)

На этапе логического проектирования сложных цифровых устройств управления большие трудности возникают из-за их нерегулярности и малой повторяемости отдельных узлов. В п. 3.1 были рассмотрены методы синтеза управляющих автоматов с жесткой логикой. «Жесткость» заключается в том, что любое изменение в алгоритме работы автомата приводит к изменению в комбинационных схемах, реализующих функции переходов и выходов автомата.

Существуют регулярные, настраиваемые пользователем структуры, называемые программируемыми логическими матрицами (ПЛМ). ПЛМ содержат в себе две матрицы – матрицу «И» ($М_{\&}$) и матрицу «ИЛИ» ($М_1$), соединенные последовательно. Матрица «И»

вычисляет конъюнкцию логических переменных, а матрица «ИЛИ» – дизъюнкцию полученных термов. Таким образом, пару матриц «И» и «ИЛИ» удобно использовать для вычисления булевых функций, заданных в виде ДНФ (дизъюнктивной нормальной формы).

В простейшем случае ПЛМ представляет матрицу – сеть горизонтальных и вертикальных шин. В узлах матрицы могут быть (а могут и не быть) полупроводниковые диоды.

Если в узле есть диод, то горизонтальная шина через него связана с вертикальной, если диода нет – то не связана. Каждая вертикальная шина такой матрицы – это простейший диодный элемент «И» или «ИЛИ» (в зависимости от направления включения диода и значения напряжения на резисторах матрицы).

На рис. 3.29 приведен пример двух матриц – «И» и «ИЛИ». Каждая вертикальная линия в этих матрицах – это один диодный логический элемент «И» или «ИЛИ». На входы матриц подаются напряжения Ux_I, соответствующие логическим переменным x_I. Значению $x_I=0$ в матрице соответствует $Ux_I=0$ [В], значению $x_I = 1$ соответствует $Ux_I=Uип$ [В], где Uип – напряжение источника питания в вольтах. Выходные напряжения матриц Uy_I соответствуют значениям функций y_I, реализуемых данными матрицами. Причем нетрудно заметить, что в матрице «ИЛИ» $Uy_I=Uип$ ($y_I=1$), если в узле матрицы на пресечении линий Uy_I и Ux_J есть диод и $Ux_J=Uип$ ($x_J=1$). В матрице «И» $Uy_I=Uип$ ($y_I=1$), если на всех входах, соединенных диодами с шиной Uy_I, значения $Ux_J=Uип$ ($x_J=1$) и $Uy_I=0$ В ($y_I=0$), если хотя бы на одном из входов $Ux_J= 0$ В ($x_J=0$).

Таким образом, в примере (см. рис. 3.29) матрица «ИЛИ» реализует три функции $У_1$, $У_2$, $У_3$ от трех переменных x_1, x_2, x_3 :

$$У_1 = x_1 \vee x_3; \quad У_2 = x_1 \vee x_2; \quad У_3 = x_2 \vee x_3,$$

а матрица «И» реализует функции

$$У_1 = x_1 \, x_3; \quad У_2 = x_1 \, x_2; \quad У_3 = x_2 \, x_3.$$

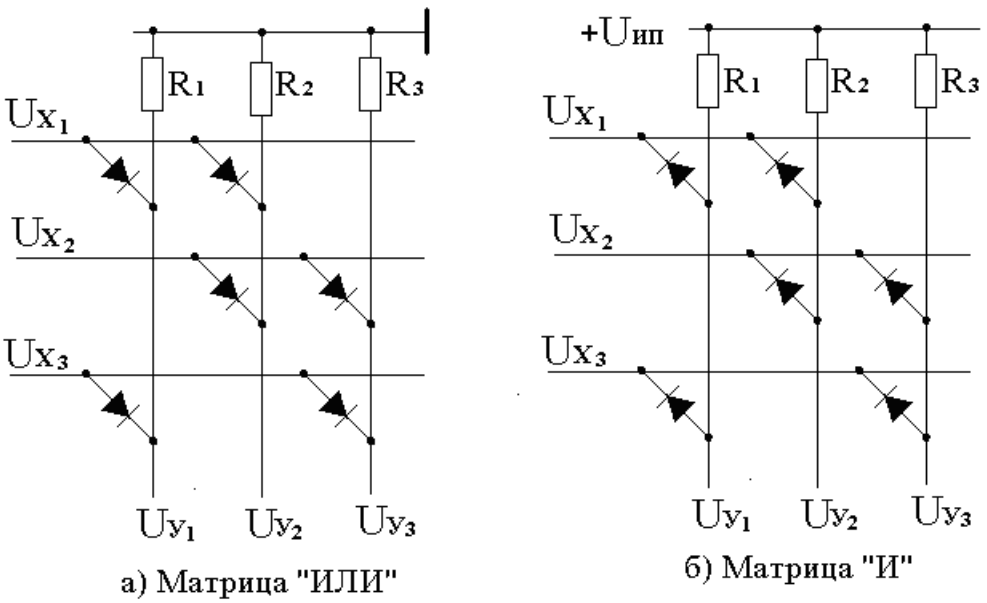


Рис. 3.29

На рис. 3.29 матрицы изображены фактически как принципиальные схемы, в которых логические переменные x_I представлены напряжениями Ux_I, а функции y_I – напряжениями Uy_I. В функциональных схемах будем использовать более простое изображение матриц: в узлах вместо диодов будем ставить точки, а входные и выходные сигналы будем обозначать как x_J и y_I (рис. 3.30).

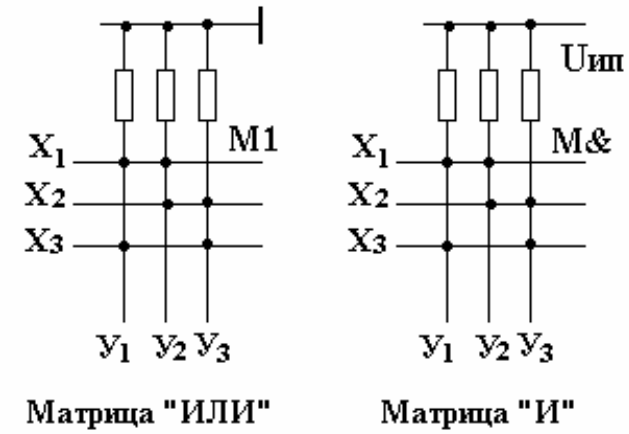


Рис. 3.30

Обычно матрицы используют тогда, когда необходимо реализовать семейство булевых функций от одних и тех же аргументов. Рассмотрим пример такой реализации.

Пусть дана система из трех булевых функций y_1, y_2, y_3 от четырех переменных x_1, x_2, x_3, x_4:

$y_1 = \neg x_1 x_2 \vee x_1 \neg x_2 \neg x_4 \vee x_1 x_3 x_4$;

$y_2 = \neg x_1 x_2 \vee \neg x_1 \neg x_3 x_4 \vee \neg x_1 \neg x_2 \neg x_3 \vee \neg x_2 \neg x_4$;

$y_3 = x_1 \neg x_2 \neg x_4 \vee x_1 \neg x_3 \vee \neg x_1 \neg x_2 \neg x_3$.

Введем вспомогательные переменные T_I, через которые выразим все дизъюнктивные термы, встречающиеся в функциях y_1, y_2, y_3:

$T_1 = \neg x_1 x_2$; $T_2 = x_1 \neg x_2 \neg x_4$; $T_3 = x_1 x_3 x_4$; $T_4 = \neg x_1 \neg x_3 x_4$;

$T_5 = \neg x_1 \neg x_2 \neg x_3$; $T_6 = \neg x_2 \neg x_4$; $T_7 = x_1 \neg x_3$.

Выразим функции y_1, y_2, y_3 через переменные T_I:

$y_1 = T_1 \vee T_2 \vee T_3$;

$y_2 = T_1 \vee T_4 \vee T_5 \vee T_6$;

$y_3 = T_2 \vee T_7 \vee T_5$.

В этих функциях через переменные T_I обозначены термы (логические произведения), которые вычисляются матрицей «И». С выходов матрицы «И» ($M_\&$), сигналы, соответствующие термам T_I, подаются на входы матрицы «ИЛИ» (M_1), где вычисляются значения функций y_1, y_2 и y_3. Схема приведена на рис. 3.31.

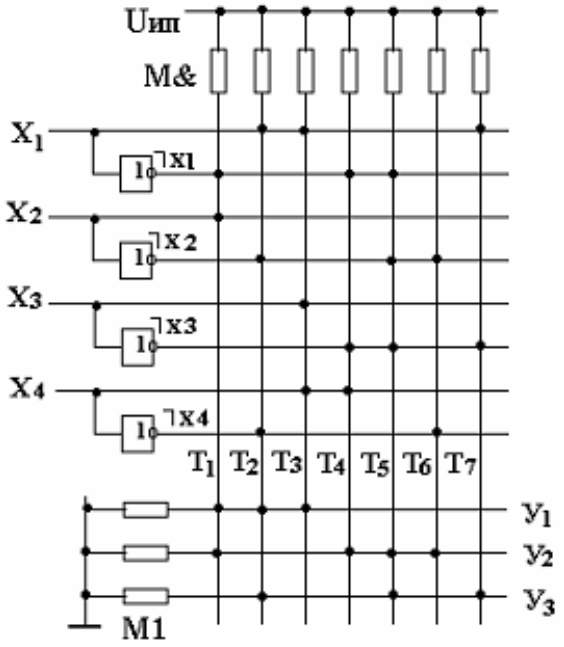


Рис. 3.31

С помощью матриц $M_\&$ («И») и M_1 («ИЛИ») можно реализовывать не только дизъюнктивные нормальные форм, но и скобочные выражения, которые часто являются более компактными.

Например, даны две функции:

$y_1 = (x_1 x_2 \vee \neg x_1 \neg x_2) x_3 \vee \neg(x_1 x_2 \vee \neg x_1 \neg x_2) \neg x_3$;

$y_2 = x_1 x_2 \vee \neg(x_1 x_2 \vee \neg x_1 \neg x_2) x_3$.

Заметим, что выражение в скобках – $(x_1 x_2 \vee \neg x_1 \neg x_2)$ – три раза встречается в функциях y_1 и y_2.

На матрицах $M_\&$ («И») и M_1 («ИЛИ») сформируем функцию $f = x_1 x_2 \vee \neg x_1 \neg x_2$, затем эту функцию используем в матрице $M_\&$ («И») для формирования термов, а затем и значений

функций y_1 и y_2. Введем переменные T_I:

$T_1 = x_1\, x_2$; $T_2 = \neg x_1\, \neg x_2$.

Через T_1 и T_2 выразим функцию f:

$f = T_1 \vee T_2$.

Введем еще три переменные T_I, в которых кроме x_1, x_2, x_3 используем функцию f:

$T_3 = f \wedge x_3$; $T_4 = \neg f \wedge \neg x_3$; $T_5 = \neg f \wedge x_3$.

Преобразуем функции y_1 и y_2 и запишем их в следующем виде:

$y_1 = (T_1 \vee T_2)\, x_3 \vee \neg(T_1 \vee T_2)\, \neg x_3 = f \wedge x_3 \vee \neg f \wedge \neg x_3 = \mathbf{T_3 \vee T_4}$;

$y_2 = T_1 \vee \neg(T_1 \vee T_2)\, x_3 = T_1 \vee \neg f \wedge x_3 = \mathbf{T_1 \vee T_5}$.

Таким образом, порядок вычислений функций y_1 и y_2 следующий. Матрица $M_{\&}$ вычисляет $T_1 = x_1\, x_2$ и $T_2 = \neg x_1\, \neg x_2$, затем матрица M_1 вычисляет $f = T_1 \vee T_2$. Значение f в прямом и инверсном виде подается на вход матрицы $M_{\&}$, где вычисляются T_3, T_4 и T_5, а затем матрица M_1 вычисляет $y_1 = T_3 \vee T_4$ и $y_2 = T_1 \vee T_5$. На рис. 3.32 приведена схема, реализующая данный способ вычисления скобочных булевых функций.

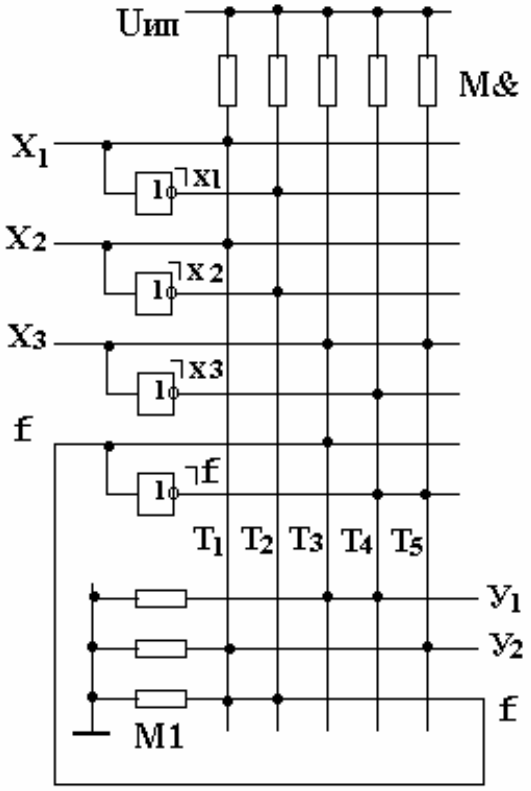


Рис. 3.32

3.3.2. ПРОСТЕЙШАЯ МАТРИЧНАЯ РЕАЛИЗАЦИЯ АВТОМАТА МИЛИ

Рассмотрим реализацию автомата Мили на ПЛМ на примере из п. 3.1.2. Отличие в синтезе автомата Мили на ПЛМ состоит только на этапах построения обратной структурной таблицы и функциональной схемы автомата.

Возьмем за основу структурную таблицу автомата Мили на жесткой логике из п. 3.1.2. Будем использовать в качестве элементов памяти состояний D-триггеры. Добавим в таблицу еще один столбец $Ta_m a_s$, в который будем записывать функции переходов: $Ta_m a_s = a_m \wedge Xa_m a_s$

, где Xa_ma_s – конъюнкция логических условий, обеспечивающих данный переход из a_m в a_s; a_m – состояние автомата, из которого начинается переход, представленное кодом состояния Ka_m , заданного значениями Q_1Q_0 на выходах элементов памяти T_1 T_0. Если, например, код состояния a_m имеет значение $Ka_m = 10$, то значения на выходах элементов памяти состояний Q_1Q_0 также равны 10. Так как здесь $Q_1=1$, то Q_1 войдет в функцию перехода без инверсии, а $Q_0 = 0$ поэтому войдет в функцию с инверсией. Например, для пятой строки структурной таблицы автомата функция перехода из a_2 в a_3 имеет вид $T_2 = a_2x_1$; так как $Ka_2 = 01$, то $T_1 = \neg Q_1Q_0x_1$.

Обратная структурная таблица автомата Мили с памятью состояний на D-триггерах (для матричной реализации) имеет вид (табл. 3.13).

Структурная схема автомата Мили на матрицах (простейшая реализация) имеет вид, показанный на рис. 3.33. В схеме автомата используются две матрицы – $M_{\&}$ («И»), которая используется для вычисления функций переходов Ta_ma_s, и M_1 («ИЛИ»), выполняющая две функции – вычисление функций (Fa_ma_s) управления элементами памяти состояний ПС) D_1 и D_0 и вычисление функций выходов (Ya_ma_s) автомата y_i.

Таблица 3.13

№ п/п	a_m	Ka_m	a_s	Ka_s	Xa_ma_s	Ya_ma_s	Fa_ma_s	Ta_ma_s
1	a_1	00	a_1	00	$\neg x_3$	-	-	$T_1 = \neg Q_1 \neg Q_0 \neg x_3$
2	a_4	11			x_2	y_7	-	$T_2 = Q_1 Q_0 x_2$
3	a_1	00	a_2	01	x_3	y_1, y_2	D_0	$T_3 = \neg Q_1 \neg Q_0 x_3$
4	a_4	11			$\neg x_2$	y_6	D_0	$T_4 = Q_1 Q_0 \neg x_2$
5	a_2	01	a_3	10	x_1	Y_3	D_1	$T_5 = \neg Q_1 Q_0 x_1$
6	a_2	01	a_4	11	$\neg x_1$	y_4, y_5	D_1, D_0	$T_6 = \neg Q_1 Q_0 \neg x_1$
7	a_3	10			1	y_4, y_5	D_1, D_0	$T_7 = Q_1 \neg Q_0$

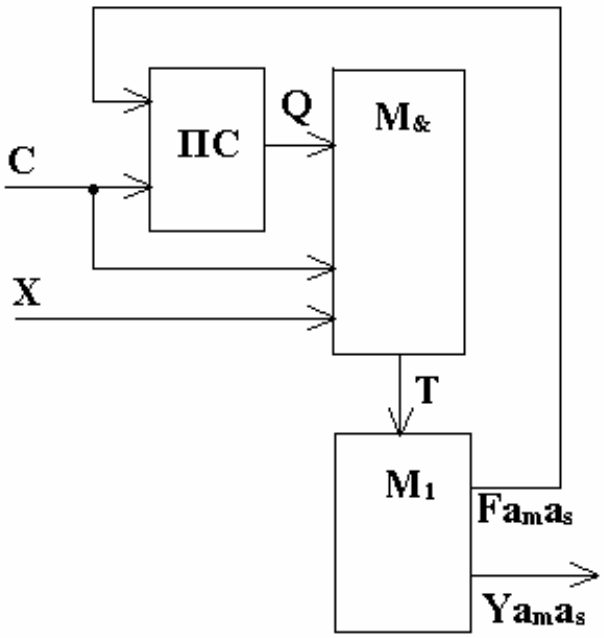


Рис. 3.33

Функции D_1 , D_0 и y_i для нашего примера находятся как дизъюнкции соответствующих функций переходов Ta_ma_s:

$D_1 = T_5 \vee T_6 \vee T_7$; $D_0 = T_3 \vee T_4 \vee T_6 \vee T_7$;

$y_1 = y_2 = T_3$; $Y_3 = T_5$; $y_4 = y_5 = T_6 \vee T_7$; $y_6 = T_4$; $y_7 = T_2$.

Как видно из этих выражений, функция переходов T_1 не используется ни в одной из функций D_i и y_i, поэтому ее можно не вычислять. Таким образом, функциональная схема автомата Мили на ПЛМ (для нашего примера) будет состоять из следующих элементов.

– Память состояний. В нашем примере – два D-триггера: T_1 T_0.

– Инверторы для формирования инверсных значений логических условий x_I .

– Матрица $M_{\&}$, имеющая шесть вертикальных шин (по количеству формируемых функций переходов Ta_ma_s) и одиннадцать горизонтальных (удвоенное количество элементов памяти и логических условий плюс один – шина для сигнала синхронизации С).

– Матрица M_1 , имеющая шесть вертикальных шин (по количеству используемых функций переходов Ta_ma_s) и семь горизонтальных (количество функций управления элементами памяти и функций переходов).

Функциональная схема автомата приведена на рис. 3.34.

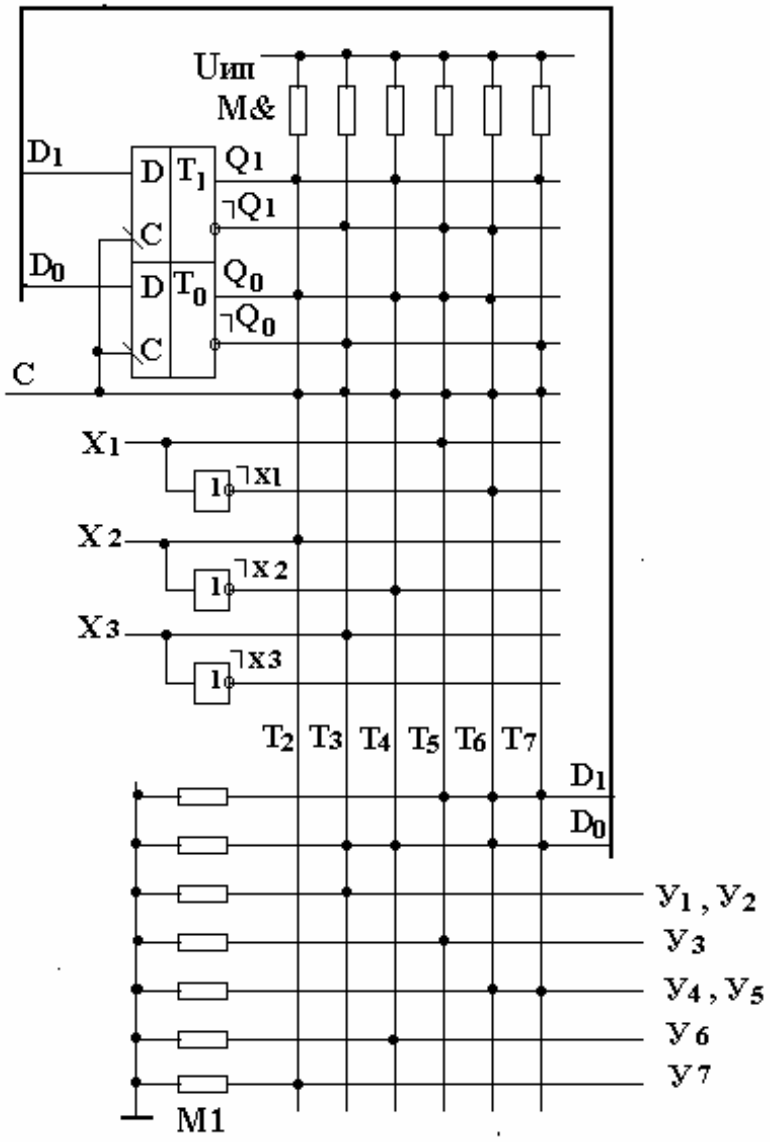


Рис. 3.34

Для упрощения схемы в ней не показаны элементы, используемые для установки автомата в начальное состояние и фиксации значений логических условий при поступлении импульса синхронизации С (смотри автомат Мили на жесткой логике).

Матрицы **$M_{\&}$** и **M_1** характеризуются площадями **$S_{M\&}$** и **S_{M1}**, вычислить которые для каждого конкретного примера можно по структурной таблице автомата (или по функциональной схеме). Например, площадь **$S_{M\&}$** = 66: 11 горизонтальных шин и 6 вертикальных. Как видно из простого примера на рис. 3.34, площади матриц **$M_{\&}$** и **M_1** используются неэффективно по двум причинам:

1) при общем количестве логических условий $|X| = 3$ в каждом конкретном переходе из a_m в a_s участвует всего одно;

2) при общем количестве $|Y| = 7$, на каждом переходе автомат вырабатывает всего одну микрокоманду.

Лекция 6.

Тема 3.2. Структурный синтез МПА Мура на ПЛМ. Вопросы оптимизации МПА

ПРОСТЕЙШАЯ МАТРИЧНАЯ РЕАЛИЗАЦИЯ АВТОМАТА МУРА

Рассмотрим реализацию автомата Мура на ПЛМ на примере из п. 3.1.1. Отличие в синтезе автомата Мура на ПЛМ состоит только на этапе построения обратной структурной таблицы и функциональной схемы автомата.

Возьмем за основу структурную таблицу автомата Мура на жесткой логике из п. 3.1.1. Будем использовать в качестве элементов памяти состояний D-триггеры. Добавим в таблицу еще один столбец Ta_ma_s, в который будем записывать функции переходов: $Ta_ma_s = a_m \wedge Xa_ma_s$ (табл. 3.14).

В схеме автомата Мура используются следующие матрицы:

$M_{\&(1)}$ («$И_1$») – для вычисления функций переходов Ta_ma_s ;

$M_{1(1)}$ («$ИЛИ_{(1)}$») – для вычисления функций управления элементами памяти D_2 , D_1 и D_0;

$M_{\&(2)}$ («$И_2$») – дешифратор состояний; так как в автомате Мура y_i зависит только от состояния a_s, функции переходов Ta_ma_s нельзя использовать для вычисления y_i;

$M_{1(2)}$ («$ИЛИ_{(2)}$») – для вычисления функций y_i как дизъюнкция выходных сигналов (переменных a_s) с матрицы $M_{\&2}$.

Таблица 3.14

№ п/п	a_m	Ka_m	a_s	Ka_s	Xa_ma_s	Ya_s	Fa_ma_s	Ta_ma_s
1	a_0	100	a_0	100	$\neg x_3$	Y_7	D_2	$T_1= Q_2\neg Q_1\neg Q_0\neg x_3$
2	a_3	000			x_2		D_2	$T_2= \neg Q_2\neg Q_1\neg Q_0 x_2$
3	a_0	100	a_1	001	x_3	y_1, y_2	D_0	$T_3= Q_2\neg Q_1\neg Q_0 x_3$
4	a_1	001	a_2	010	x_1	Y_3	D_1	$T_4= \neg Q_2\neg Q_1 Q_0 x_1$
5	a_4	011			x_1		D_1	$T_5= \neg Q_2 Q_1 Q_0 x_1$
6	a_1	001	a_3	000	$\neg x_1$	y_4, y_5	-	$T_6= \neg Q_2\neg Q_1 Q_0 \neg x_1$
7	a_2	010			1		-	$T_7= \neg Q_2 Q_1\neg Q_0$
8	a_4	011			$\neg x_1$		-	$T_8= \neg Q_2 Q_1 Q_0 \neg x_1$
9	a_3	000	a_4	011	$\neg x_2$	y_6	D_1 ,D_0	$T_9 =\neg Q_2\neg Q_1\neg Q_0\neg x_2$

Функции D_2, D_1, D_0 для нашего примера находятся как дизъюнкции соответствующих функций переходов Ta_ma_s, а y_i зависят только от состояний автомата a_s:

$D_2 = T_1 \vee T_2$; $D_1 = T_4 \vee T_5 \vee T_9$; $D_0 = T_3 \vee T_9$;

$y_1 = y_2 = a_1$. $y_3 = a_2$. $y_4 = y_5 = a_3$. $y_6 = a_4$. $y_7 = a_0$.

Структурная схема автомата Мура на матрицах (простейшая реализация) имеет вид, показанный на рис. 3.35.

Матрицы $M_{\&(1)}$ и $M_{1(1)}$ выполняют только функцию вычисления Fa_ma_s. Матрица $M_{\&(2)}$ на входе имеет коды состояний автомата $Q_2Q_1Q_0$, а на выходе формирует значения переменных a_0, a_1 и т.д., соответствующих состояниям автомата. Матрица $M_{1(2)}$ используется для реализации функций выходов y_i в случае, если они выражены через дизъюнкции a_i.

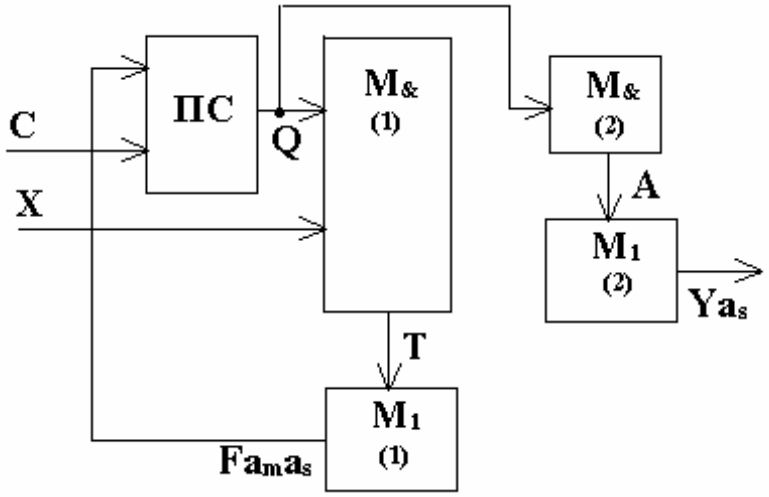


Рис. 3.35

Функциональная схема автомата приведена на рис. 3.36. Для упрощения схемы в ней не показаны элементы, используемые для установки автомата в начальное состояние.

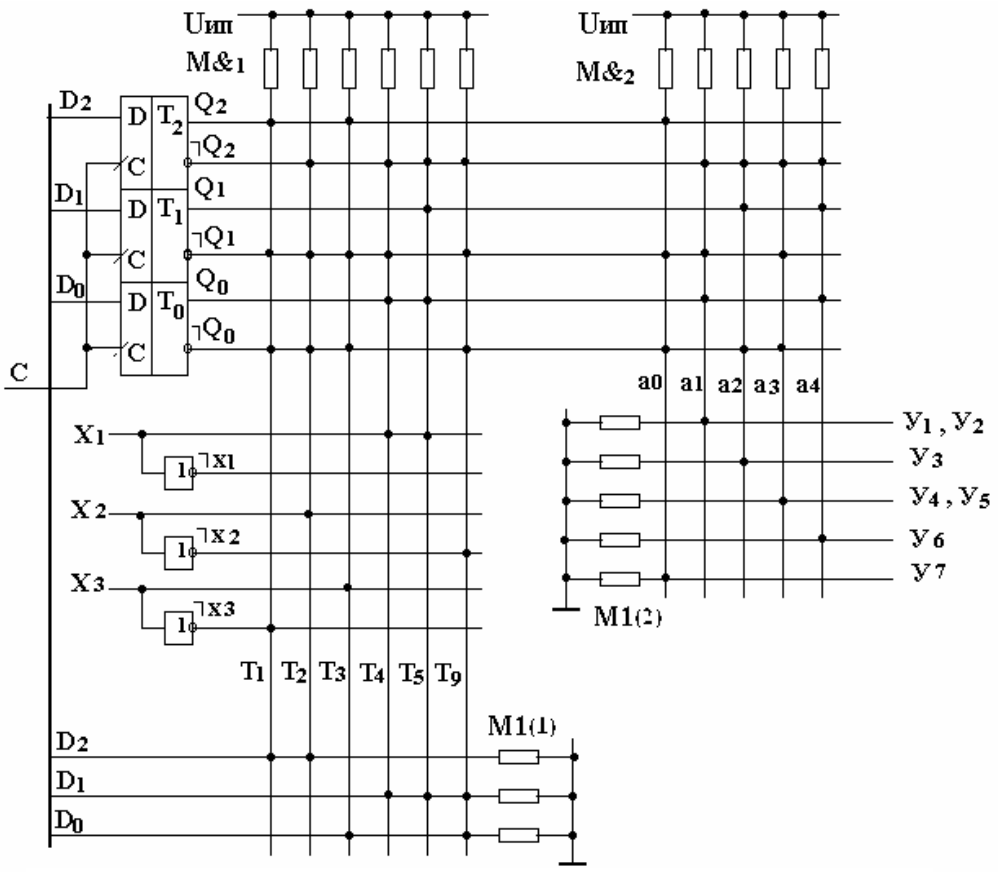


Рис. 3.36. Функциональная схема автомата

Как видно из схемы, матрицу $M_{1(2)}$ можно было не использовать, так как в нашем примере каждая функция y_i зависит только от одной переменной a_i.

3.3.4. ВОПРОСЫ ОПТИМИЗАЦИИ АВТОМАТОВ НА МАТРИЦАХ

Методы оптимизации автоматов на матрицах направлены, в основном, на сокращение суммарной площади матриц. Как было указано в п. 3.3.1, матрицы $\mathbf{M_{\&}}$ и $\mathbf{M_1}$ при тривиальной реализации автомата сильно разрежены: большая часть площади этих матриц не используется, так как в узлах нет диодов. Это касается части матрицы $\mathbf{M_{\&}}$, вычисляющей функции переходов (в области переменных X), и матрицы $\mathbf{M_1}$ (в области переменных У).

Кодирование логических условий X. ***Площадь матрицы $M_{\&}$ в автоматах Мили и Мура зависит:***

1) от количества элементов памяти (количество выходов Q и ¬Q элементов памяти обозначим K_Q; K_Q входит в общее число горизонтальных шин матрицы $\mathbf{M_{\&}}$; $K_Q >= 2 \bullet] \log_2 | A | [ )$;
2) от количества переменных X (количество горизонтальных шин от этих переменных $K_X = 2 \bullet |X|$ также входит в общее число горизонтальных шин матрицы $\mathbf{M_{\&}}$);
3) от числа переходов $T(a_m a_s)$ автомата K_T.

Уменьшить K_Q невозможно, так как в автоматах на матрицах используется всегда минимальное кодирование состояний. Количество переходов можно иногда уменьшить за

счет использования разметки ГСА с узлами. В то же время в каждой конкретной функции переходов Ta_ma_s обычно используются не все переменные из множества X, а только некоторые из них. Например, по структурной таблице из п. 3.3.2 и 3.3.3 видно, что в каждой функции Ta_ma_s используется максимум одна переменная X (при общем их количестве 3), поэтому есть возможность уменьшить площадь $\mathbf{SM_{\&}}$ за счет замены трех переменных X одним логическим условием P, принимающим значение определенной переменной X в зависимости от состояния автомата a_m.

Алгоритм такой замены описан ниже.

1. Разобьем множество входных сигналов автомата (логических условий) X на пересекающиеся подмножества $Xa_м$; элементами $Xa_м$ являются логические условия X_j, определяющие все переходы автомата из состояния $a_м$.
2. Найдем максимум $\alpha=\max|Xa_м|$ и введем новое множество, содержащее α логических условий $P=\{P_1, P_2...P_\alpha\}$.
3. Каждую переменную $x\in Xa_м$ заменим на переменную из множества P; составим таблицу этой подстановки. Желательно, чтобы одноименные логические условия X_j оказались в одном и том же столбце таблицы.
4. Запишем функции $P_i= F(a_м,x_i)$, используя таблицу подстановки.
5. Закодируем состояния автомата таким образом, чтобы можно было реализовать эти функции на матрицах $\mathbf{M_{\&}}$ и $\mathbf{M_1}$ с минимальными затратами. Полученные коды состояний будем использовать далее при составлении структурной таблицы автомата.
6. Используем сформированные функции P_i вместо переменных X в матрице $\mathbf{M_{\&}}$ для вычисления функций переходов автомата Ta_ma_s.

Например, пусть по размеченному графу автомата с шестью состояниями $a_0…a_5$ и семью логическими условиями $x_0…x_6$ получили следующие подмножества $Xa_м$:

$Xa_0=\{x_0,x_1\}$; $Xa_3=\{x_5\}$;

$Xa_1=\{x_2,x_3\}$; $Xa_4=\{x_1,x_3\}$;

$Xa_2=\{x_3,x_4\}$; $Xa_5=\{x_6\}$.

Нетрудно заметить, что $Max\ |Xa_i| = 2$.

Введем два новых логических условия: $P=\{P_1, P_2\}$ и составим таблицу замены переменных X на переменные P.

$a_м$	P_1	P_2
a_0	x_0	x_1
a_1	x_3	x_2
a_2	x_3	x_4
a_3	x_5	-
a_4	x_3	x_1
a_5	x_6	-

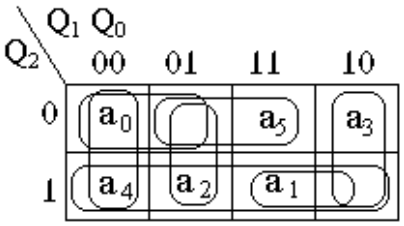
Запишем выражение для P_1 и P_2:

$P_1 = a_0x_0 \vee (a_1 \vee a_2 \vee a_4) x_3 \vee a_3x_5 \vee a_5x_6$

$P_2 = a_1x_2 \vee (a_0 \vee a_4) x_1 \vee a_2x_4.$

В этих выражениях, например, P_1 принимает значение логического условия x_0, если автомат находится в состоянии a_0 (так как при этом a_0=1); значение x_3, если автомат находится в одном из состояний: a_1 или a_2 или a_4 и т.п.

Закодируем состояния автомата таким образом, чтобы выражения в скобках можно было записать в виде одного терма, используя переменные Q_3, Q_2, Q_1 – сигналы с выходов элементов памяти состояний автомата. Используем карту Карно. В ячейках карты расставим состояния a_I таким образом, чтобы была возможность осуществить их минимальное покрытие согласно скобочным частям выражений P_1 и P_2. Получим следующие коды состояний:



$Ka_0 =$ 000; $Ka_3 =$ 000;

$Ka_1 =$ 111; $Ka_4 =$ 100;

$Ka_2 =$ 101; $Ka_5 =$ 011.

Учитывая то, что в карте Карно есть свободные клетки, используем их для минимизации выражений P_1 и P_2 :

$$P_1 = a_0x_0 \vee (a_1 \vee a_2 \vee a_4) x_3 \vee a_3x_5 \vee a_5x_6 =$$

$$= \neg Q_2 \neg Q_1 x_0 \vee Q_2 x_3 \vee Q_1 \neg Q_0 x_5 \vee \neg Q_2 Q_0 x_6 = Z_1 \vee Z_2 \vee Z_3 \vee Z_4 ,$$

где $Z_1 = \neg Q_2 \neg Q_1 x_0$; $Z_2 = Q_2 x_3$; $Z_3 = Q_1 \neg Q_0 x_5$; $Z_4 = \neg Q_2 Q_0 x_6$;

$$P_2 = a_1x_2 \vee (a_0 \vee a_4) x_1 \vee a_2x_4 =$$

$$= Q_2 Q_1 x_2 \vee \neg Q_1 \neg Q_0 x_1 \vee \neg Q_1 Q_0 x_4 = Z_5 \vee Z_6 \vee Z_7 ,$$

где $Z_5 = Q_2 \bullet Q_1 \bullet x_2$; $Z_6 = \neg Q_1 \bullet \neg Q_0 \bullet x_1$; $Z_7 = \neg Q_1 \bullet Q_0 \bullet x_4$.

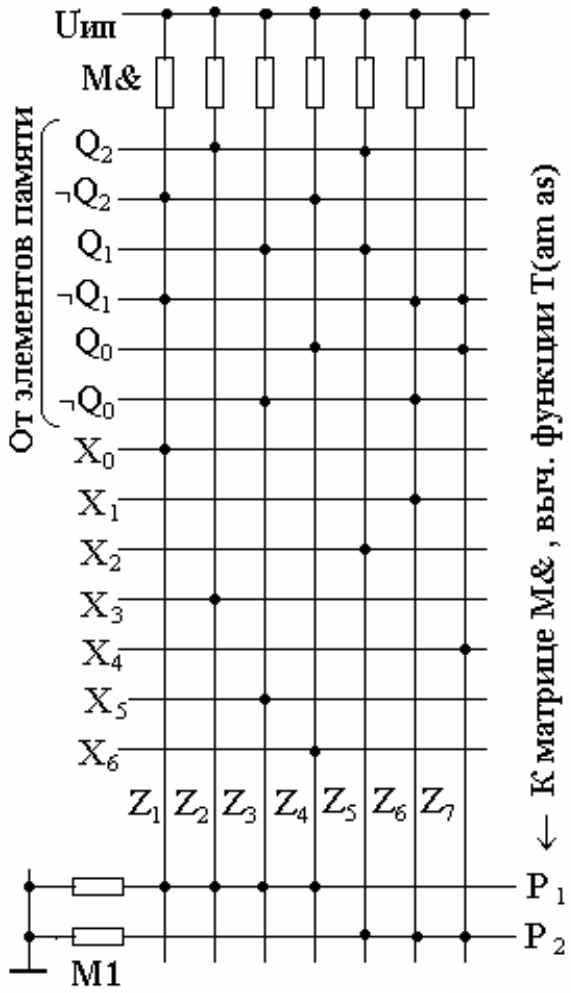


Рис. 3.37

Функциональная схема, реализующая преобразование логических условий X в логические условия P, приведена на рис. 3.37. Матрица $\mathbf{M_{\&}}$ вычисляет термы $Z_1 \ldots Z_7$, матрица $\mathbf{M_1}$ – функции P_1 и P_2.

Площадь матрицы $\mathbf{M_{\&}}$ равна

$$SM_{\&} = (K_X + K_Q) \bullet K_Z ,$$

где K_X – количество логических условий X, в нашем примере – K_X=7; K_Q – количество прямых и инверсных выходов элементов памяти, в нашем примере K_Q = 6; K_Z – количество термов в функциях P_1и P_2 , в нашем примере K_Z=7.

Таким образом, в нашем примере $SM_{\&}$ = 91.

Площадь матрицы $\mathbf{M_1}$ равна $SM_1 = K_Z \bullet K_P$, где K_P – количество логических условий P.

В нашем примере SM_1 = 14. Общая площадь матриц $\mathbf{M_{\&}}$ и $\mathbf{M_1}$, используемых для минимизации количества логических условий, равна

$$S_{Mp} = (K_X + K_Q) \bullet K_Z + K_Z \bullet K_P = (K_X + K_Q + K_P)* K_Z .$$

Если не использовать кодирование логических условий X, то площадь части матрицы $\mathbf{M_{\&}}$, вычисляющей функции переходов Ta_ma_s (в нашем примере этой матрицы нет) и использующей в качестве переменных X, равна

$$S_{MTx} = 2 \bullet K_X \bullet K_T,$$

где K_T – количество переходов (строк) в структурной таблице автомата.

При замене логических условий X на P эта площадь составит

$$S_{MTp} = 2 \bullet K_P \bullet K_T.$$

В итоге, при кодировании логических условий X, площадь матриц, зависящая от X, равна

$$S_{Xp} = S_{Mp} + S_{MTp} = (K_X + K_Q + K_P)* K_Z + 2 \bullet K_P \bullet K_T,$$

а без кодирования

$$S_{Xx} = S_{MTx} = 2 \bullet K_X \bullet K_T.$$

Использование описанного преобразования имеет смысл, если S_{Xp} заметно меньше, чем S_{Xx}. Разница в площадях матриц без кодирования и с кодированием логических условий

$$\Delta S_X = S_{Xx} - S_{Xp} = 2 \bullet K_X \bullet K_T - (K_X + K_Q + K_P) \bullet K_Z - 2 \bullet K_P \bullet K_T =$$
$$= 2 \bullet K_T \bullet (K_X - K_P) - (K_X + K_Q + K_P) \bullet K_Z .$$

Чтобы эта разница была положительной ($\Delta S_X > 0$), необходимо выполнение условия

$$2 \bullet K_T \bullet (K_X - K_P) > (K_X + K_Q + K_P) \bullet K_Z ,$$

иначе преобразование логических условий имеет смысл, если выполняется соотношение

$$K_T > ((K_X + K_Q + K_P) \bullet K_Z) / (2 \bullet (K_X - K_P)).$$

Если условие выполняется и при этом ΔS_X имеет значительную величину (относительно S_{Xx}), то можно проделать описанную процедуру и получить при этом более эффективное использование матриц.

В нашем примере

$$\Delta S_X = 2 \bullet K_T \bullet (K_X - K_P) - (K_X + K_Q + K_P) \bullet K_Z .=$$
$$= 2 \bullet K_T \bullet (7 - 2) - (7 + 6 + 2) \bullet 7 = 10 \bullet K_T - 105.$$

Из полученных выражений видно, что $\Delta S_X > 0$, если в автомате $K_T > 11$ (количество переходов больше 11); при выполнении этого условия имеет смысл выполнить преобразование логических условий X в логические условия P.

Кодирование выходных сигналов автомата (микрокоманд y_j). В автоматах Мили при простейшей реализации (п. 3.2) в матрице $\mathbf{M_1}$ есть сильно разреженная часть, где вычисляются y_i. Это происходит потому, что общее число микрокоманд в автомате может быть большим, но на каждом переходе вырабатывается небольшое число микрокоманд. В таком случае можно повысить эффективность использования матрицы $\mathbf{M_1}$, используя кодирование микрокоманд. Для этого необходимо:

1) выписать из ГСА или обратной структурной таблицы одинаковые группы микрокоманд, которые обозначить как B_i;
2) записать выражения для y_i как $\cup B_j$ – дизъюнкция групп B_i, в которых функция $y_i = 1$;
3) составить граф отношений между группами B_i. Вершины графа соответствуют группам микрокоманд B_i, ребрами соединяются вершины (B_i.), содержащие одинаковые микрокоманды;
4) закодировать вершины (B_i) двоичным кодом (b_1 b_2 b_3 …) таким образом, чтобы связанные вершины на графе можно было выразить через конъюнкцию переменных b_i. Количество переменных b_i должно быть минимальным и равно $\lceil \log_2 |B| \rceil$;
5) записать выражения для b_j как $\cup T_j$: где $\cup T_j$ – дизъюнкция функций переходов

автомата Мили, на которых b_j =1;

6) записать выражения $y_i=\cup B_j$ через конъюнкцию переменных b_i: $y_i=\cap b_j$;
7) реализовать функции $b_j = \cup T_j$ с помощью матрицы $\mathbf{M_1}$;
8) реализовать функции $y_i=\cap b_j$ с помощью матрицы $\mathbf{M_{\&}}$.

Рассмотрим пример.

Таблица 3.15

№	$Y(a_m a_s)$	$T(a_m a_s)$	B_i	$b_1b_2b_3b_4$
1	y_1, y_2	T_1	B_1	0 0 1 1
2	y_1, y_3	T_2	B_2	0 1 1 0
3	y_7, y_8	T_3	B_3	1 0 0 1
4	y_1, y_{10}	T_4	B_4	0 1 1 1
5	y_5, y_6	T_5	B_5	1 0 1 1
6	y_3	T_6	B_6	1 1 1 0
7	y_4, y_8	T_7	B_7	1 1 0 0
8	y_4, y_8, y_9	T_8	B_8	1 1 0 1
9	y_1, y_{10}	T_9	B_4	0 1 1 1
10	y_4, y_8	T_{10}	B_7	1 1 0 0
11	y_4, y_8, y_9	T_{11}	B_8	1 1 0 1
12	y_1, y_{10}	T_{12}	B_4	0 1 1 1
13	y_5, y_6	T_{13}	B_5	1 0 1 1
14	y_5, y_6	T_{14}	B_5	1 0 1 1
15	y_4, y_8, y_9	T_{15}	B_8	1 1 0 1
16	y_1, y_{10}	T_{16}	B_4	0 1 1 1
17	y_7, y_8	T_{17}	B_3	1 0 0 1
18	y_9	T_{18}	B_9	0 1 0 1
19	y_5, y_{11}	T_{19}	B_{10}	1 0 1 0

Допустим имеется обратная структурная таблица (табл. 3.15) автомата Мили для реализации на ПЛМ, из которой мы выпишем только столбец Ya_ma_s, а в столбце Ta_ma_s оставим только обозначение функций переходов T_i. Остальные столбцы таблицы нас не интересуют. Из этой таблицы видно, что площадь матрицы $\mathbf{M_1}$ при простейшей реализации автомата, использующаяся для формирования микрокоманд y_i, равна

$$S_{МУ} = К_У \bullet К_Т = 11 \bullet 19 = 209.$$

В этом примере количество микрокоманд, которые может формировать автомат, равно 11 ($y_1 \ldots y_{11}$), но на каждом из 19 возможных переходов $T_1 \ldots T_{19}$ формируется максимум 3 микрокоманды.

Добавим в эту таблицу еще два столбца. Первый – для обозначения групп микрокоманд (B_i); в нашем примере – 10 различных групп микрокоманд (обозначим их $B_1 \ldots B_{10}$). Во втором столбце будем записывать коды групп B_i. Минимальное количество разрядов для кодирования B_i равно четырем. Обозначим код B_i: $b_1\, b_2\, b_3\, b_4$.

Выразим y_i через дизъюнкцию B_j, в которые входит y_i:

$y_1 = B_1 \vee B_2 \vee B_4$;

$y_2 = B_1$;

$y_3 = B_2 \vee B_6$;

$y_4 = B_7 \vee B_8$;

$y_5 = B_5 \vee B_{10}$;

$y_6 = B_5$;

$y_7 = B_3$;

$y_8 = B_3 \vee B_7 \vee B_8$;

$y_9 = B_8 \vee B_9$;

$y_{10} = B_4$;

$y_{11} = B_{10}$.

Используя выражения $y_i = \cup B_j$ построим граф отношений между группами B_i, в котором ребрами соединяются вершины (B_i.), содержащие одинаковые микрокоманды (рис. 3.38). Используя карту Карно, закодируем группы B_i так, чтобы можно было совместно покрыть связанные вершины (смотри карту Карно).

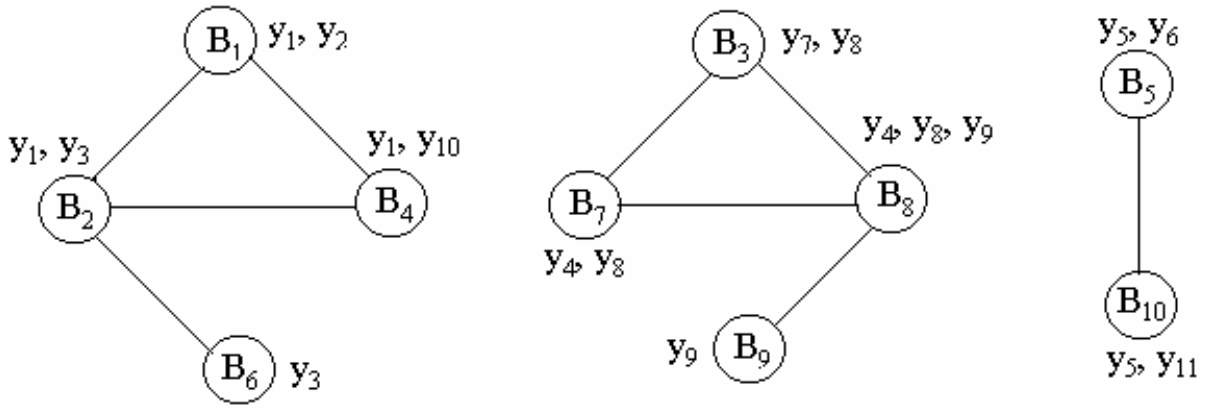
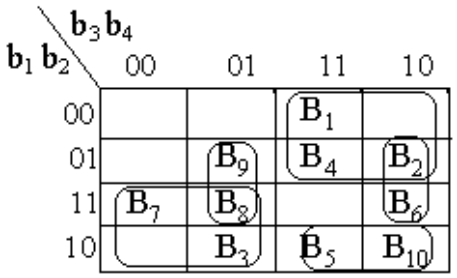


Рис. 3.38

По карте Карно запишем коды B_i ($КВ_i$), выраженные через значения переменных $b_1\, b_2\, b_3\, b_4$:



$КВ_1$=0011; $КВ_2$=0110; $КВ_3$=1001;
$КВ_4$=0111; $КВ_5$=1011; $КВ_6$=1110;
$КВ_7$=1100; $КВ_8$=1101; $КВ_9$=0101;
$КВ_{10}$=1010.

Запишем полученные коды групп микрокоманд в столбец ($b_1b_2b_3b_4$) структурной таблицы автомата и выразим функции b_i как дизъюнкцию соответствующих переменных T_i той же таблицы:

$b_1 = T_3 \vee T_5 \vee T_6 \vee T_7 \vee T_8 \vee T_{10} \vee T_{11} \vee T_{13} \vee T_{14} \vee T_{15} \vee T_{17} \vee T_{19}$;

$b_2 = T_2 \vee T_4 \vee T_6 \vee T_7 \vee T_8 \vee T_9 \vee T_{10} \vee T_{11} \vee T_{12} \vee T_{15} \vee T_{16} \vee T_{18}$;

$b_3 = T_1 \vee T_2 \vee T_4 \vee T_5 \vee T_6 \vee T_9 \vee T_{12} \vee T_{13} \vee T_{14} \vee T_{16} \vee T_{19}$;

$b_4 = T_1 \vee T_3 \vee T_4 \vee T_5 \vee T_8 \vee T_9 \vee T_{11} \vee T_{12} \vee T_{13} \vee T_{14} \vee T_{15} \vee T_{16} \vee T_{17} \vee T_{18}$.

Выразим y_i , записанные через дизъюнкцию B_j, в виде конъюнкции $y_i = \cap b_j$ (согласно покрытиям в карте Карно):

Реализуем функции $b_1 \ldots b_4$ с помощью матрицы **M_1**, а функции $y_1 \ldots y_{11}$ – с помощью

$y_1 = B_1 \vee B_2 \vee B_4 = \neg b_1 b_3$;

$y_2 = B_1 = \neg b_1 \neg b_2$;

$y_3 = B_2 \vee B_6 = b_2 b_3 \neg b_4$;

$y_4 = B_7 \vee B_8 = b_1 b_2 \neg b_3$;

$y_5 = B_5 \vee B_{10} = b_1 \neg b_2 b_3$;

$y_6 = B_5 = b_1 \neg b_2 b_3 b_4$;

$y_7 = B_3 = b_1 \neg b_2 \neg b_3$;

$y_8 = B_3 \vee B_7 \vee B_8 = b_1 \neg b_3$;

$y_9 = B_8 \vee B_9 = b_2 \neg b_3 b_4$;

$y_{10} = B_4 = \neg b_1 b_2 b_3 b_4$.

$y_{11} = B_{10} = b_1 \neg b_2 b_3 \neg b_4$;

матрицы **$M_{\&}$** (рис. 3.39).

Оценим суммарную площадь матриц **M_1** и **$M_{\&}$** , реализующих функции выходов автомата Мили с кодированием микрокоманд:

$S_{Mb} = K_b \cdot K_T = 4 \cdot 19 = 76$, где K_T – количество переменных, используемых для кодирования групп микрокоманд; $S_{MYb} = 2 \cdot K_b \cdot K_Y = 2 \cdot 4 \cdot 11 = 88$; $S_{MYк} = S_{Mb} + S_{MYb} = 76 + 88 = 164$.

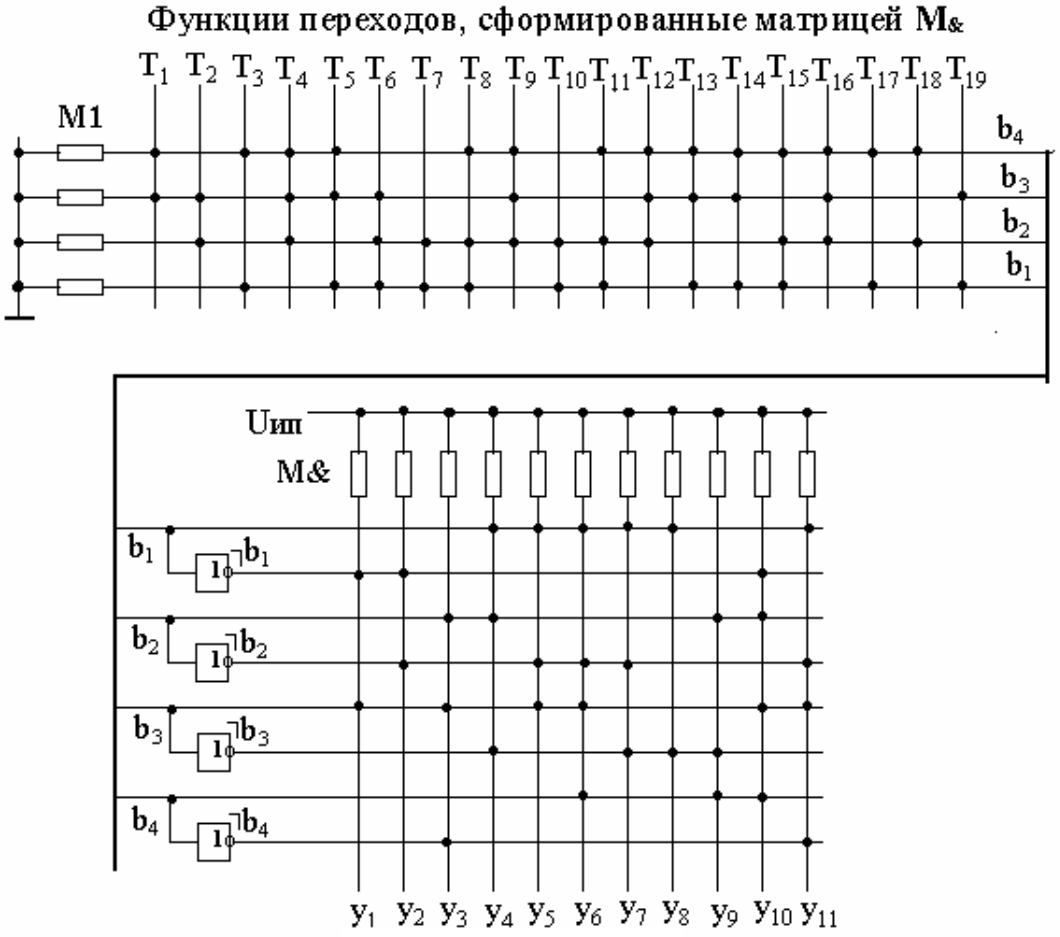


Рис. 3.39

Разница в площадях матриц без кодирования и с кодированием составляет

$\Delta S_У = S_{МУ} – S_{МУк} = 209 – 164 = 45.$

Экономия площади матриц позволяет реализовать более сложный автомат на меньшей площади кристалла матрицы.

Кодирование микрокоманд имеет смысл производить, если выполняется условие

$K_Т \bullet (K_У – K_b) > 2 \bullet K_b \bullet K_У$.

Решение о кодировании можно принять после определения количества групп микрокоманд B_j и возможности их минимального кодирования переменными b_j.

На рисунках приведены обобщенные структурные схемы автомата Мили с кодированием логических условий и микрокоманд (рис. 3.40) и автомата Мура с кодированием логических условий и микрокоманд (рис. 3.41).

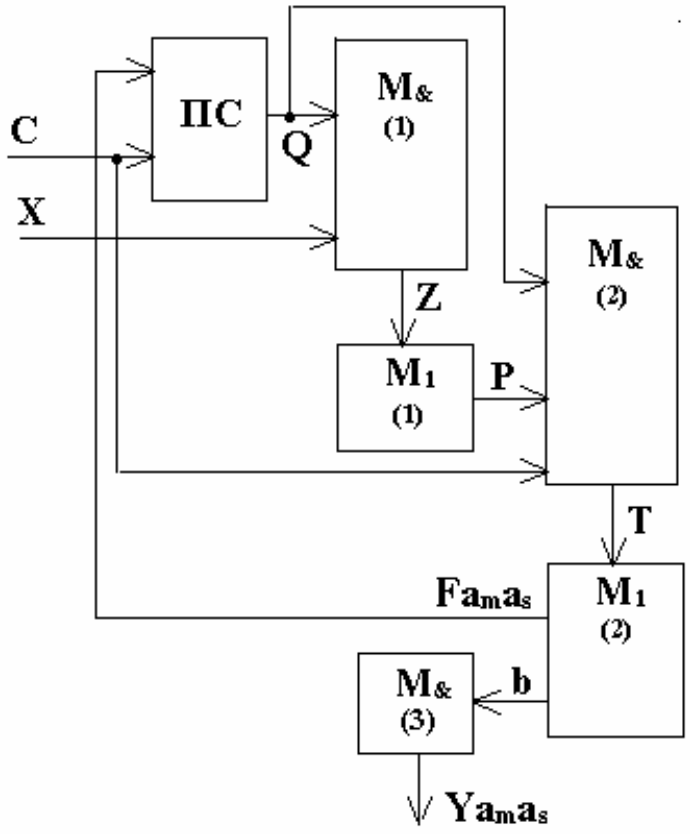


Рис. 3.40

В схеме автомата Мили используются следующие элементы:

– память состояний – ПС;

– матрица $M_{&(1)}$, вычисляющая термы z_i для последующего вычисления логических условий P матрицей $M_{1(1)}$;

– матрица $M_{&(2)}$, вычисляющая функции переходов T_i;

– матрица $M_{1(2)}$, вычисляющая функции управления элементами памяти состояний

Fa_ma_s и коды групп микрокоманд $КВ_i$ (b_1b_2...);

– матрица $М_{&(3)}$, вычисляющая значения функций y_i.

В схеме автомата Мура используются следующие элементы:

– память состояний – ПС;

– матрица $М_{&(1)}$, вычисляющая термы z_i для последующего вычисления логических условий P матрицей $М_{1(1)}$;

– матрица $М_{&(2)}$, вычисляющая функции переходов T_i;

– матрица $М_{1(2)}$, вычисляющая функции управления элементами памяти состояний Fa_ma_s;

– матрица $М_{&(3)}$, выполняющая функции дешифратора состояний автомата и вычисляющая значения функций a_i;

– матрица $М_{1(3)}$, вычисляющая значения функций y_i.

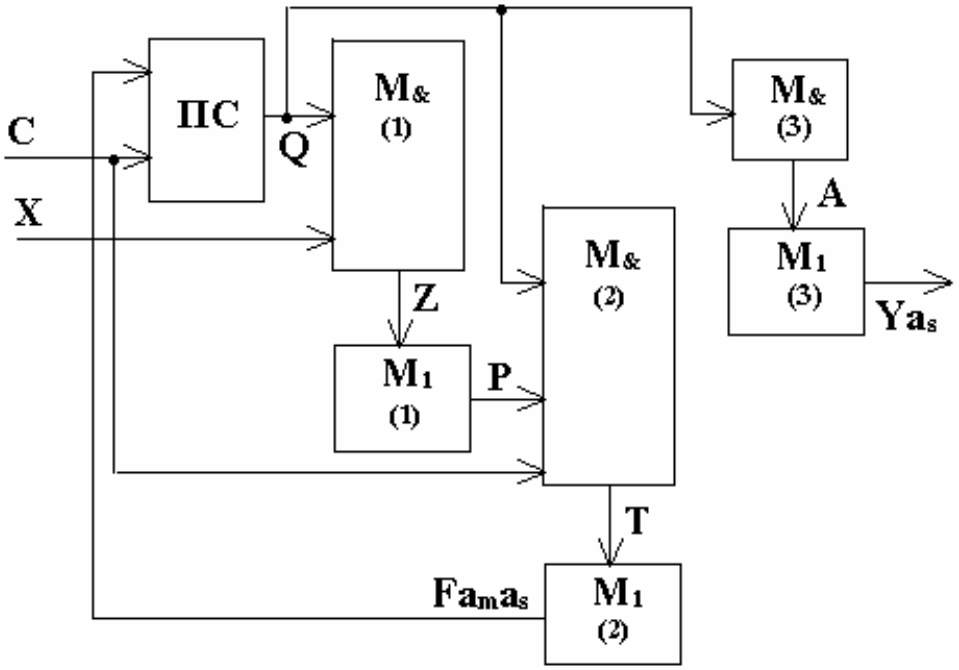


Рис. 3.41

Лекция 7.

Раздел 4.

Тема 4.1. Автоматы с программируемой логикой и естественной адресацией. Вопросы оптимизации

Рассмотренные выше реализации микропрограммного автомата на жесткой логике приводят к различным схемным решениям в зависимости от исходных ГСА. А это значит, что при изменении функции автомата (изменении ГСА) необходимо изменять либо заново проектировать его схему. Реализация автомата на программируемых матрицах (ПЛМ – однократно программируемых), при незначительных изменениях в алгоритме работы автомата, требует программирования новых ПЛМ. Этого недостатка лишены автоматы с программируемой логикой. Алгоритм функционирования автомата с программируемой логикой представляется в виде закодированной микропрограммы, которая может загружаться в перепрограммируемые постоянные запоминающие устройства (ППЗУ) или в оперативные ЗУ (ОЗУ). Изменения в алгоритме отражаются в микропрограмме, но не влияют на схему автомата.

Принцип управления по хранимой программе

Напомним, что функция микропрограммного управляющего автомата определяется:

- множеством входных сигналов (логических условий): $X = \{x_1, x_2 \ldots x_n\}$;
- множеством выходных сигналов (микрокоманд): $У = \{y_1, y_2 \ldots y_k\}$;
- микропрограммой (ГСА), задающей порядок следования выходных сигналов У в зависимости от значений входных сигналов Х.

Если автомат Мура описан прямой таблицей переходов, то для каждого состояния a_m известно:

- множество выходных сигналов $У(a_m)$, вырабатываемых автоматом в состоянии a_m;
- множество входных сигналов $X(a_m)$, определяющих все возможные переходы автомата из состояния a_m в состояние a_s;
- множество состояний a_s, в которые может перейти автомат из состояния a_m (обозначим $A(a_m)$.

Для каждого состояния автомата Мура (а значит для каждой операторной вершины ГСА) эту информацию, необходимую и достаточную для описания закона функционирования автомата, можно закодировать и представить двоичным словом (микрокомандой – не путать с выходным сигналом автомата y_i, который часто так же называют микрокомандой !), которое имеет следующую структуру:

№	y_{m1}	y_{m2}	…	y_{mk}	x_{m1}	x_{m2}	…	x_{mn}	a_{m1}	a_{m2}	…	a_{ms}

где:

№ – номер двоичного слова, соответствующего состоянию автомата a_m (или его адрес ЗУ, где будет храниться микропрограмма);

y_{mi} – выходной сигнал (микрокоманда), который должен вырабатывать автомат в данном состоянии a_m; в общем случае таких сигналов k; эта часть двоичного слова называется **операционной**;

x_{mi} – входной сигнал (логическое условие), участвующее хотя бы в одном переходе из a_m в a_s ; в общем случае таких сигналов n; эта часть двоичного слова называется **управляющей**;

a_{mi} – одно из состояний a_s , в которое может перейти автомат из состояния a_m при определенных значениях логических условий x_{mi}. Поскольку описание одного состояния – это одно двоичное слово, имеющее свой адрес в ЗУ, a_{mi} можно считать адресом состояния a_{mi} в ЗУ; в общем случае таких состояний s. Если количество логических условий в слове равно n, то количество a_{mi} равно 2 в степени n. Эта часть двоичного слова называется **адресной**.

Воспользуемся прямой таблицей переходов автомата Мура из примера (п.3.1).

№	a_m	a_s	$Y\, a_s$	$X a_m a_s$
1	a_0	a_0	y_7	$\neg x_3$

2		a_1	y_1, y_2	x_3
3 4	a_1	a_2 a_3	y_3 y_4, y_5	x_1 $\neg x_1$
5	a_2	a_3	y_4, y_5	1
6 7	a_3	a_4 a_0	y_6 y_7	$\neg x_2$ x_2
8 9	a_4	a_2 a_3	y_3 y_4, y_5	x_1 $\neg x_1$

Если использовать приведенную выше структуру двоичного слова, то для описания данного автомата достаточно пяти двоичных слов (по числу состояний автомата), имеющих следующий формат:

№	y_{m1}	y_{m2}	x_m	$a_{x=0}$	$a_{x=1}$

Сам номер (№) в слово помещать не нужно, так как это адрес слова в ЗУ. В слове под y_{mi} отведено два поля, так как максимальное количество y_{mi} в состоянии a_m равно двум; под x_{mi} – достаточно отвести одно поле, так как в любом переходе из a_m в a_{mi} (или a_s) участвует только одно логическое условие. Под a_{mi} достаточно отвести 2 поля: для адреса перехода при $x_m = 0$ ($a_{x=0}$) и при $x_m = 1$ ($a_{x=1}$). Используя выбранный формат двоичного слова можно составить в виде таблицы содержательную микропрограмму, описывающую работу автомата.

В этой таблице – каждая строка соответствует одному из состояний автомата (столбец a_m); № - соответствует адресу слова, описывающего это состояние в ЗУ. В колонках y_{m1} и y_{m2} записаны микрокоманды, вырабатываемые автоматом в состоянии a_m, в колонке x_m записаны логические условия, определяющие переходы из состояния a_m в состояния, адреса которых в ЗУ записаны в колонках $a_{x=0}$ и $a_{x=1}$, в колонке a_m записаны пояснения: какому состоянию автомата соответствует данная строка.

№	y_{m1}	y_{m2}	x_m	$a_{x=0}$	$a_{x=1}$	a_m
0	y_7	-	x_3	0	1	a_0
1	y_1	y_2	x_1	3	2	a_1
2	y_3	-	x_3	3	3	a_2
3	y_4	y_5	x_2	4	0	a_3
4	y_6	-	x_1	3	2	a_4

Каждая из колонок этой таблицы (кроме a_m) должна быть представлена двоичными кодами. Количество двоичных разрядов для каждой колонки можно выбрать таким:

- y_{m1} – 3 разряда: здесь будем записывать номер y_{m1} (от 1 до 7);
- y_{m2} – 3 разряда: здесь будем записывать номер y_{m2} (от 1 до 7);
- x_m – 2 разряда: здесь будем записывать номер x_m (от 1 до 3);
- $a_{x=0}$ – 3 разряда: здесь будем записывать адрес $a_{x=0}$, по которому в автомате должен произойти переход к следующему слову при x_m =0 (от 0 до 4);
- $a_{x=1}$ – 3 разряда: здесь будем записывать адрес $a_{x=1}$, по которому в автомате должен произойти переход к следующему слову при x_m =1(от 0 до 4).

Таким образом разрядность двоичного слова в нашем примере равна 14. Закодированная микропрограмма имеет вид, приведенный в таблице. В каждой колонке записаны двоичные числа в соответствии с принятыми выше соглашениями. Требуемый объем памяти – 70 бит. Рассмотрим подробнее 2-ю строку в этих таблицах.

№	y_{m1}	y_{m2}	x_m	$a_{x=0}$	$a_{x=1}$
0	111	000	11	000	001
1	001	010	01	011	010
2	011	000	11	011	011
3	100	101	10	100	000
4	110	000	01	011	010

В этой и предыдущей таблицах в строке №2 описывается состояние автомата a_2. Из этого состояния – безусловный переход автомата в состояние a_3. Поэтому в столбце x_m может быть записан номер любой переменной x_m , а в обоих столбцах $a_{x=0}$ и $a_{x=1}$ одинаковый адрес перехода. Так как записанный в столбце x_m обязательно имеет одно из значений: 0 или 1, то переход будет безусловным по адресу $a_{x=0} = a_{x=1}$.

Схема, обеспечивающая работу с закодированной микропрограммой, должна содержать следующие функциональные узлы:

- память для хранения микропрограммы (ЗУ) объемом не менее пяти слов; чтение слова из памяти осуществляется по адресу, подаваемому на адресный вход ЗУ; разрядность слов – 14 бит;
- схему, преобразующую двоичный код y_{m1} и y_{m2} в унитарный код, соответствующий закодированному y_i; эта схема – дешифратор (DCy);

- мультиплексор (MSx), выбирающий из множества входных сигналов X всего один, номер которого записан в столбце x_m;
- мультиплексор (MSa), выбирающий из слова адрес $a_{x=0}$ (если выбранное мультиплексором MSx значение x_m=0) или адрес $a_{x=1}$ (если выбранное мультиплексором MSx значение x_m=1);
- регистр адреса (RG), в который необходимо по импульсу синхронизации записать выбранный мультиплексором MSa адрес и подать его на адресный вход ЗУ, из которого будет считано следующее слово (иначе – переход в следующее состояние).

y_{m1}		y_{m2}		x_m	$a_{x=0}$				$a_{x=1}$				
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Пронумеруем разряды двоичного слова с 1 по 14 (как показано в таблице); номера разрядов будут соответствовать номерам выходов ЗУ.

Функциональная схема автомата приведена на рис.27.

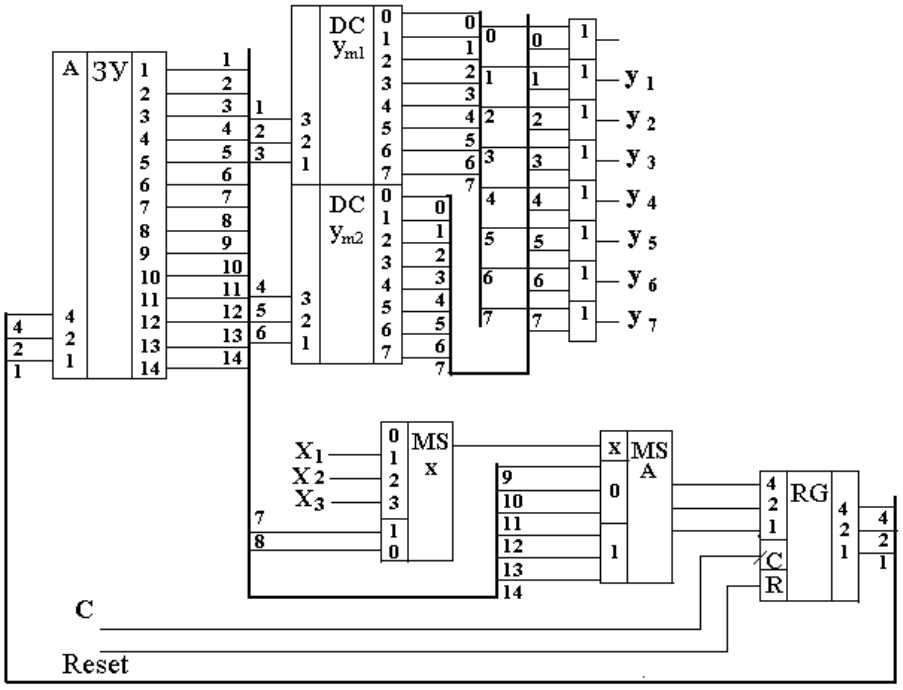


Рисунок 27.

На этой схеме кроме описанных выше элементов имеется 8 дизъюнкторов, объединяющих одноименные выходы дешифраторов DCy_{m1} и DCy_{m2} . Эти элементы необходимы в случае, если одна и та же микрокоманда y_i будет записана в разных строках микропрограммы в разных столбцах y_{m1} и y_{m2}.

Автомат (рис.27) работает следующим образом. Сигнал Reset=1 устанавливает значение «0» во всех разрядах регистра RG. Код (или двоичное число) A=000 с выхода RG подается на адресный вход ЗУ. На выходах ЗУ (разряды 1..14) появляется слово, записанное по адресу 0 (см. таблицу). В этом слове:

разряды 1,2,3 – подаются на вход DCy_{m1} и, поскольку в этих разрядах записано число 111 (7), на выходе «7» DCy_{m1} формируется 1, а значит и выходной сигнал y_7 = 1;

разряды 4,5,6 – подаются на вход DCy_{m2} и, поскольку в этих разрядах записано число 000 (0), (которое мы не использовали для кодирования y_i) на выходе «0» DCy_{m1} формируется «1», но ни на каких либо других выходах y_i эта «1» не появится;

разряды 7,8 – подаются на управляющие входы мультиплексора MSx; в этих разрядах записано число 11 (3). Значение логического условия x_3 появляется на выход MSx;

мультиплексор MSa, в зависимости от значения x_3 («0» или «1») подает на входы регистра адреса RG либо адрес перехода, записанный в разрядах 9,10,11 (если $x_3 = 0$), либо адрес перехода, записанный в разрядах 12,13,14 (если $x_3 = 1$);

при поступлении на вход С импульса синхронизации, этот адрес запишется в регистр RG и с его выходов поступит на адресный вход ЗУ, на выходе которого появится слово, записанное по этому адресу.

Таким образом за один цикл (время между синхроимпульсами С) автомат считает из ЗУ некоторое двоичное слово, выработает микрокоманды y_m, проанализирует входные сигналы x_m, и в зависимости от их значения выберет адрес следующего двоичного слова в ЗУ.

Спроектированный нами интуитивно автомат содержит много избыточности и был приведен в качестве примера, как естественное воплощение наших рассуждений. Недостатками такого подхода являются, прежде всего, сложность схемы (при достаточно простых функциях автомата) и не оптимальная длина микрокоманды (двоичного слова), что приводит к неэффективному использованию памяти ЗУ.

Рассмотрим некоторые приемы, позволяющие сделать автомат с программируемой логикой более оптимальным, чем приведенный на рис.27.

Кодирование наборов микроопераций

Выходные сигналы y_i (микрокоманды), вырабатываемые управляющим автоматом, сгруппированы в операторных вершинах ГСА в виде наборов, инициирующих выполнение операционным автоматом (ОА) совместимых микроопераций. Совместимые микрооперации – это такие, которые могут быть выполнены ОА одновременно.

Пусть автомат должен генерировать n различных наборов микрокоманд из множества $У = \{y_1, y_2 \ldots y_k\}$; обозначим i-ый набор как:

$У_i = \{y_{i1}, y_{i2} \ldots y_{ij}\}$.

Составим матрицу M_{N*K}, где N – количество различных наборов микрокоманд, а К – количество различных микрокоманд y_i, вырабатываемых автоматом. Элемент матрицы m_{ij} = 1, если микрокоманда y_j входит в набор $У_i$, а если не входит, то $m_{ij} = 0$. Если строки матрицы М (а значит и наборы $У_i$) не пересекаются, то в операционной части двоичного слова можно обойтись одним полем, в которое записывается номер набора $У_i$. Этот номер будет подаваться на один дешифратор DCу, на выходе которого будут сформированы микрокоманды y_i.

В нашем примере все микрокоманды y_i можно представить в виде следующих наборов:

$У_0 = \{y_7\}$; $У_1 = \{y_1, y_2\}$; $У_2 = \{y_3\}$; $У_3 = \{y_4, y_5\}$; $У_4 = \{y_6\}$;

Матрицу М представим в виде таблицы:

	y_1	y_2	y_3	y_4	y_5	y_6	y_7
$У_0$							1
$У_1$	1	1					
$У_2$			1				
$У_3$				1	1		
$У_4$						1	

Из матрицы видно, что пересечение строк для всех пар $У_i$ и $У_j$ (при $i \neq j$) пусто. Закодируем наборы $У_i$ следующим произвольным образом:

$К(У_0) = 000$; $К(У_1) = 001$; $К(У_2) = 010$; $К(У_3) = 011$; $К(У_4) = 100$,

и изменим формат микрокоманды:

№	$К(У_i)$	x_m	$a_{x=0}$	$a_{x=1}$

Количество разрядов двоичного слова при этом уменьшится до 11, а значит и уменьшится емкость необходимой памяти ЗУ до 55 бит. Пронумеруем разряды двоичного слова с 1 по 11 (как показано в таблице); номера разрядов будут соответствовать номерам выходов ЗУ на функциональной схеме автомата (рис.28).

$К(У_i)$			x_m		$a_{x=0}$			$a_{x=1}$		
1	2	3	4	5	6	7	8	9	10	11

Схема на рис.28 незначительно отличается от схемы на рис.27 по своему функционированию. Отличие – один дешифратор микрокоманд $У_i$, назначение которого описано выше.

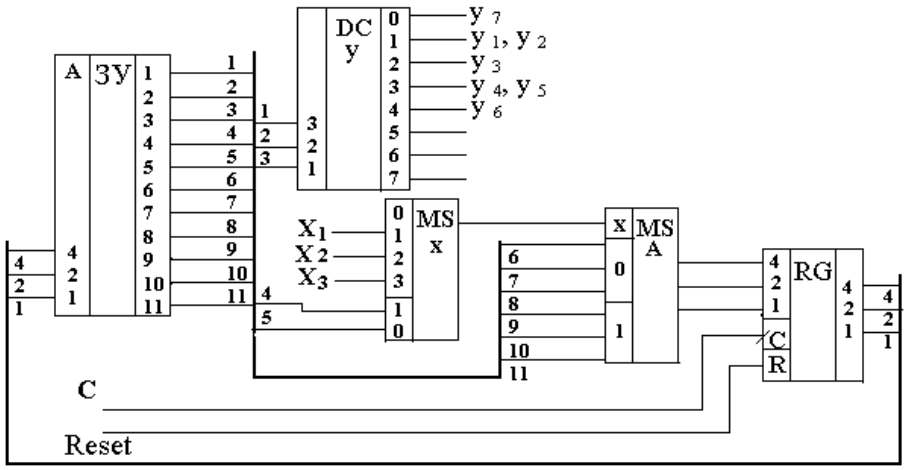


Рисунок 28.

Если наборы микрокоманд в матрице М пересекаются, то описанный в последнем примере подход к их формированию невозможен. Поэтому можно поступить следующим образом.

1. Если общее количество микрокоманд $У_i$ невелико, то в операционной части двоичного слова можно выделить по одному разряду на каждую микрокоманду. В этом случае отпадает необходимость в дешифраторе DCy, так как в соответствующие разряды операционной части двоичного слова записываются значения (0 или 1) для всех $У_i$. Таким образом сигналы $У_i$ берутся непосредственно с соответствующих выходов ЗУ. Если использовать такой подход для нашего примера, то разрядность двоичного слова увеличится до 15 бит, емкость ЗУ – до 75 бит, но из схемы автомата исчезнет дешифратор Dcy. Формат двоичного слова приведен ниже.

2. Если

y_1	y_2	y_3	y_4	y_5	y_6	y_7	x_m		$a_{x=0}$			$a_{x=1}$		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

общее количество микрокоманд $У_i$ достаточно велико и использование приема, описанного в п1, приведет к значительному увеличению длины операционной части двоичного слова, то можно поступить следующим образом: закодировать пересекающиеся наборы

микрокоманд минимальным образом (например, как в предыдущих примерах), в операционной части двоичного слова записать коды этих наборов, а в схеме автомата вместо DCy использовать дополнительное ЗУ. Разрядность адреса этого ЗУ равна разрядности кодов наборов микрокоманд, разрядность слов, записанных в ЗУ, равна количеству $У_i$ в автомате. Каждый разряд слова в этом ЗУ соответствует определенному $У_i$ и в него записывается «0» или «1» в зависимости от значения соответствующего $У_i$ в данном наборе микрокоманд. Например, если использовать данный прием для предыдущего примера, то дополнительное ЗУ должно иметь разрядность адреса равное 3, разрядность слов – 5. Вид такого ЗУ приведен на рис.29.

Если использовать кодирование наборов микрокоманд такое же, как в предыдущих примерах, то таблица программирования этого ЗУ будет иметь вид:

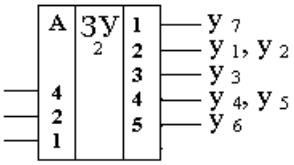


Рисунок 29.

	$у_7$	**$у_1$,$у_2$**	**$у_3$**	**$у_4$,$у_5$**	**$у_6$**

А\бит	1	2	3	4	5
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1

Адресация микрокоманд в автомате

В автоматах с программируемой логикой используются, в основном, два метода адресации микрокоманд: принудительная и естественная.

При принудительной адресации адрес перехода к следующему двоичному слову в явном (или неявном) виде присутствует в самой микрокоманде. Так, в рассмотренных выше примерах, адресация была принудительная. В двоичном слове для хранения адресов переходов отводилось два поля (при одном логическом условии в слове). Такая микрокоманда называется двухадресной.

Естественная адресация предполагает наличие двоичных слов разного вида, например, операционные и управляющие слова. Операционные слова соответствуют операторным вершинам ГСА управляющего автомата, а управляющие – условным. Если в ГСА автомата есть линейные части (не содержащие условных вершин), то адреса слов, описывающих следующие друг за другом операторные вершины ГСА, должны формироваться в естественном нарастающем порядке: $a_{t+1} = a_t + 1$,

где: a_t – адрес текущего слова,

a_{t+1} – адрес следующего слова.

Управляющее слово может быть эквивалентно управляющей конструкции некоторого языка программирования:

IF $X_i = 1$ THEN GOTO $a_{xi=1}$;

если значение логического условия $X_i = 1$, то адрес следующего слова $a_{t+1} = a_{xi=1}$; если $X_i = 0$, то адрес следующего слова: $a_{t+1} = a_t + 1$.

Естественная адресация микрокоманд в автомате

Формат микрокоманд (двоичных слов) при естественной адресации может быть следующим. Операционная микрокоманда содержит два поля: поле признака вида микрокоманды Р (операционная или управляющая; пусть для операционной – Р=0) и поле, в котором любым из возможных способов записаны значения y_i, которые должен вырабатывать автомат. Управляющая микрокоманда содержит три поля: поле признака Р (пусть для управляющей – Р=1), поле логического условия (может содержать № логического условия X_i) и поле адреса перехода при $X_i = 1$.

Формат операционного слова:

0	$У_i$

Формат управляющего слова:

1	X_i	Адр $a_{xi=1}$.

Поскольку для хранения и операционных и управляющих слов используется одно и то же ЗУ, разрядность слов должна быть одинаковой.

Возьмем, для примера, ГСА управляющего автомата, приведенную на рис.5 и запишем алгоритм работы автомата, используя следующие конструкции:

Оператор присваивания вида: $y_i = 1$, для описания операторных вершин ГСА;

Оператор условного перехода вида: IF $X_i = 1$ THEN GOTO $a_{xi=1}$, для описания условных вершин ГСА.

Алгоритм, представленный в виде таблицы, напоминает программу на языке BASIC. Для более эффективного представления, в строках с адресами 1 и 3 перед логическими условиями X_3 и X_1 поставлен знак инверсии (¬), что позволяет сократить количество строк в записи алгоритма. В строке 8 использована конструкция: GOTO 3 – безусловный переход к строке 3. Для реализации безусловного перехода введем дополнительное логическое условие, например X_0, а в схеме автомата предусмотрим, что бы всегда $X_0 = 1$. В этом случае, конструкцию «GOTO 3 » можно заменить на конструкцию: IF $X_0 = 1$ THEN GOTO 3.

Адрес	Оператор
0	$У_7 := 1$;
1	IF $\neg X_3 = 1$ THEN GOTO 0;
2	$У_1 := 1$; $У_2 := 1$;
3	IF $\neg X_1 = 1$ THEN GOTO 5;
4	$У_3 := 1$;
5	$У_4 := 1$; $У_5 := 1$;
6	IF $X_2 = 1$ THEN GOTO 0;
7	$У_6 := 1$;
8	GOTO 3; (IF $X_0 = 1$ THEN GOTO 3;)

Из таблицы видно, что закодированная микропрограмма должна содержать 9 строк с адресами в ЗУ от 0 до 8. Поэтому адресное поле управляющей микрокоманды (в которое

будем записывать адрес перехода - А) должно содержать не менее четырех разрядов (возьмем ровно 4), а поле логического условия (в которое будем записывать номер логического условия - Нх) должно содержать не менее двух разрядов (так как в нашем примере – 4 различных логических условия). В таблице показано соответствие разрядов двоичного слова и информации, которую содержит слово:

Разряды слова	1	2	3	4	5	6	7
Значения	1	Hx_1	Hx_0	A_3	A_2	A_1	A_0

В первом разряде всегда записывается 1 – признак управляющего слова. В разрядах 2 и 3 записывается номер логического условия Нх в виде двоичного числа: Hx_1 Hx_0, где индексы 0 и 1 – указывают младший и старший разряды числа Нх. В разрядах 4, 5, 6 и 7 записывается адрес перехода А в виде двоичного числа A_3 A_2 A_1 A_0, где индексы при А так же указывают старшинство разрядов в числе А.

Формат операционного слова выберем следующий:

Разряды слова	1	2	3	4	5	6	7
Значения	0	y_7	y_1,y_2	y_3	y_4,y_5	y_6	-

В первом разряде всегда записывается 0 – признак операционного слова. В остальных разрядах будем записывать 1 или 0, в зависимости от того, какое значение имеют микрокоманды y_i в данном слове. В следующей таблице, в колонках, обозначенных номерами разрядов двоичного слова от 1 до 7, приведена закодированная микропрограмма управляющего автомата с естественной адресацией.

Адрес	Оператор	Разряды слова						
		1	2	3	4	5	6	7
0	У $_7$:= 1;	0	1	0	0	0	0	0
1	IF $\neg X_3 = 1$ THEN GOTO 0;	1	1	1	0	0	0	0
2	У $_1$:= 1; У $_2$:= 1;	0	0	1	0	0	0	0
3	IF $\neg X_1 = 1$ THEN GOTO 5;	1	0	1	0	1	0	1
4	У $_3$:= 1;	0	0	0	1	0	0	0
5	У $_4$:= 1; У $_5$:= 1;	0	0	0	0	1	0	0
6	IF $X_2 = 1$ THEN GOTO 0;	1	1	0	0	0	0	0
7	У $_6$:= 1;	0	0	0	0	0	1	0
8	IF $X_0 = 1$ THEN GOTO 3;	1	0	0	0	0	1	1

Для большей наглядности серым фоном в таблице выделены 1-ый разряд – признак вида микрокоманды, и разряды 4..7 – адрес перехода в управляющей микрокоманде. Объем памяти ЗУ – 63 бита.

Схема, обеспечивающая работу с закодированной микропрограммой автомата с естественной адресацией , должна содержать следующие функциональные узлы:

- память для хранения микропрограммы (ЗУ) объемом не менее девяти слов; чтение слова из памяти осуществляется по адресу (четырехразрядному), подаваемому на адресный вход ЗУ; разрядность слов – 7 бит;
- схему, обрабатывающую операционную микрокоманду - преобразующую двоичный код в разрядах слова с 2-го по 6-ой в сигналы y_i на выходе автомата. Эта схема может быть реализована пятью элементами «И»;
- мультиплексор (MSx), выбирающий из множества входных сигналов X всего один, номер которого записан в разрядах 2 и 3 управляющего слова; мультиплексор должен иметь дополнительный вход разрешения V; при V=1 – режим нормальной работы MSx; при V=0 – режим запрета: на выходе MSx значение «0» независимо от сигналов на его входах;
- четырехразрядный двоичный счетчик (СТ), имеющий два режима работы: параллельная загрузка адреса перехода (из разрядов 4..7 управляющего слова) в счетчик (когда выполняется условие X_i=1) и режим счета: СТ:=СТ+1 - во всех остальных случаях. Для этого счетчик СТ имеет вход управления V; при V=0 – режим счета, при V=1 – режим параллельной загрузки. У счетчика СТ должен быть еще один вход R для установки его в начальное состояние СТ:=0 (Reset).

Функциональная схема автомата с программируемой логикой и естественной адресацией приведена на рис.30.

Автомат (рис.30) работает следующим образом. Сигнал Reset=1 устанавливает значение «0» во всех разрядах двоичного счетчика СТ. Код (или двоичное число) A=0000 с выхода СТ подается на адресный вход ЗУ. На выходах ЗУ (разряды 1..7) появляется слово, записанное по адресу A=0 (см. таблицу). В нашем примере по адресу A=0 записана операционная микрокоманда, в которой:

первый разряд имеет значение 0 (признак операционной микрокоманды), поэтому на выходе мультиплексора MSx так же значение 0, а счетчик СТ находится в режиме счета; проинвертированное значение первого разряда (1) подается на входы элементов «И», выполняющих функцию электронных ключей, и на выходах этих элементов – значения 2..7 разрядов микрокоманды, т.е. – значения y_i ; в нашем примере - $У_7$;

так как счетчик в режиме счета (V=0), то при поступлении на вход С счетчика СТ импульса синхронизации, его состояние изменится: СТ:=СТ+1, а значит изменится на +1 адрес: A=0001; этот адрес поступит на адресный вход ЗУ, на выходе которого появится слово, записанное по данному адресу;

в нашем примере слово по адресу A=0001 – управляющая микрокоманда; первый разряд в этом слове имеет значение 1 (признак управляющей микрокоманды); проинвертированное значение первого разряда (0) подается на входы элементов «И», и на выходах всех этих элементов – значения «0» (все y_i=0); на разрешающий вход мультиплексора MSx подается значение «1»: MSx находится в режиме нормальной работы;

значения разрядов 2 и 3 – подаются на управляющие входы мультиплексора MSx; в нашем примере в этих разрядах записано двоичное число 11 (3); значение логического условия $\neg X_3$ появляется на выходе MSx;

если $\neg X_3 = 1$ ($X_3 = 0$), то на вход V счетчика СТ подается «1» (режим параллельной загрузки счетчика) и, при подаче на его вход С импульса синхронизации, в СТ запишутся значения разрядов 4..7 микрокоманды (в нашем примере – это адрес 0000); из ЗУ будет считано слово по этому адресу;

если $\neg X_3 = 0$ ($X_3 = 1$), то на вход V счетчика СТ подается «0» (режим счета) и, при подаче на его вход С импульса синхронизации, состояние СТ изменится: СТ:=СТ+1; в нашем

примере СТ:= 0001+1=0010; запишутся значения разрядов 4..7 микрокоманды (в нашем примере – это адрес 0000); из ЗУ будет считано слово по адресу 0010 (2); и т.д.

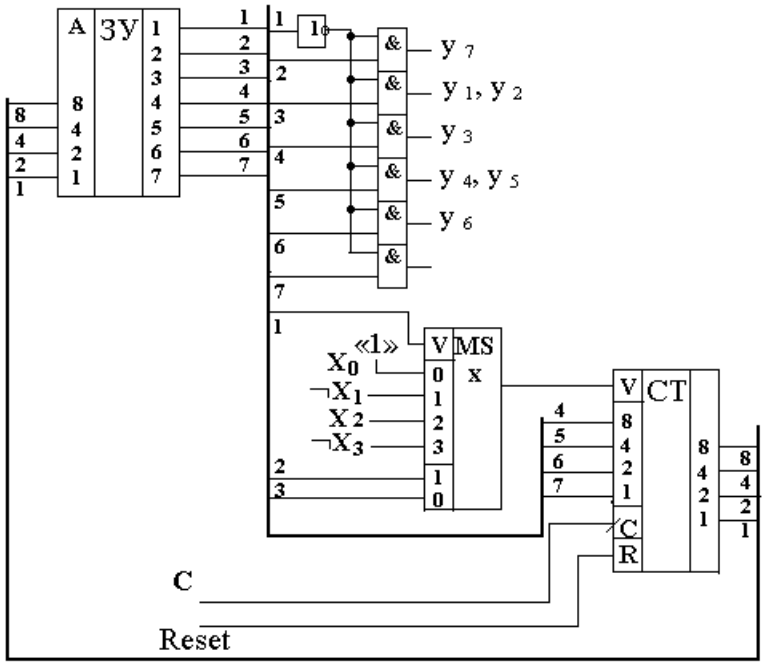


Рисунок 30.

Лекция 8.

Тема 4.2. Автоматы с программируемой логикой и принудительной адресацией. Вопросы оптимизации

Принудительная адресация микрокоманд в автомате.

В начале раздела на рис.27 и 28 приведены схемы автомата с принудительной адресацией. В отличие от естественной, принудительная адресация сводится к указанию в каждом слове адреса (или адресов) следующего слова. Однако даже при двухадресной микрокоманде, когда в каждом двоичном слове всего одно логическое условие, длина адресной части может быть значительной. Есть различные способы уменьшения длины адресной части в двоичном слове. Один из них заключается в следующем. В адресной части слова записывают адрес перехода при $X_i = 0$ (Адр $_{xi=0}$), а если $X_i = 1$, то адрес перехода Адр вычисляется так:

$Адр = Адр_{xi=0} + 1$.

Если использовать логическое значение переменной X_i как арифметическое (0 или 1), то адрес перехода можно вычислить так:

$Адр = Адр_{xi=0} + X_i$.

Таким образом, сократив вдвое длину адресной части двоичного слова, мы не потеряли возможность осуществлять условный переход в автомате с естественной адресацией. Однако при этом получили некоторое неудобство: двоичные слова, к которым происходит переход, всегда находятся в ЗУ в соседних ячейках, так как адреса у них различаются всегда на 1. Поэтому в микропрограмме автомата с описанным способом адресации может появиться избыточность в виде дублирования некоторых слов и безусловных переходов.

Воспользуемся ГСА (рис.5), прямой таблицей переходов автомата Мура из примера в п.3.1. и форматом микрокоманды из первого примера текущего раздела, в котором в адресной части оставим только одно поле $a_{x=0}$.

№	a_m	a_s	Y a_s	X $a_m a_s$
1 2	a_0	a_0 a_1	y_7 y_1, y_2	$\neg x_3$ x_3
3 4	a_1	a_2 a_3	y_3 y_4, y_5	x_1 $\neg x_1$
5	a_2	a_3	y_4, y_5	1
6 7	a_3	a_4 a_0	y_6 y_7	$\neg x_2$ x_2
8 9	a_4	a_2 a_3	y_3 y_4, y_5	x_1 $\neg x_1$

К($У_i$)			x_m		$a_{x=0}$		
1	2	3	4	5	6	7	8

Используем кодирование наборов микрокоманд и дополнительное ЗУ (смотри рис. 29). Микропрограмма с комментариями приведена ниже.

$У_0 = \{y_7\}$; $У_1 = \{y_1, y_2\}$; $У_2 = \{y_3\}$; $У_3 = \{y_4, y_5\}$; $У_4 = \{y_6\}$;

$К(У_0) = 000$; $К(У_1) = 001$; $К(У_2) = 010$; $К(У_3) = 011$; $К(У_4) = 100$.

Для организации безусловного перехода в этом примере введено дополнительное логическое условие X_0 . В схеме автомата на вход X_0 подадим значение «0», а значит переход в этом случае будет всегда по адресу $a_{x=0}$, записанному в микрокоманде.

Адрес	Комментарий \ разряды								
0	a_0; IF $X_3 = 0$ THEN GOTO 0;								
1	a_1; IF $\neg X_1 = 0$ THEN GOTO 2;								
2	a_2; IF $X_0 = 0$ THEN GOTO 3;								
3	a_3; IF $X_2 = 0$ THEN GOTO 4;								
4	a_4; IF $\neg X_1 = 0$ THEN GOTO 2;								
5	a_0; IF $X_3 = 0$ THEN GOTO 0;								

Из таблицы видно, что на состояние a_0 пришлось использовать два слова в ЗУ, тем не менее, общий объем ЗУ в нашем примере 48 бит (меньше, чем у двухадресной микрокоманды).

Функциональная схема автомата приведена на рис.31.

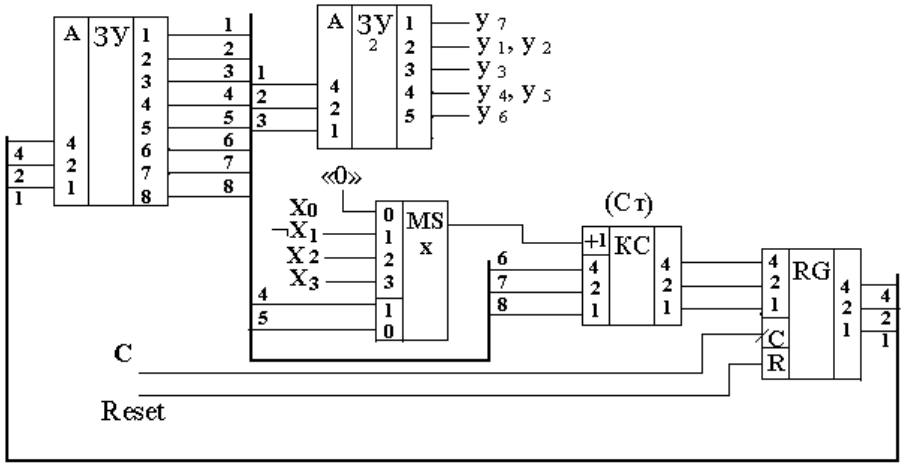


Рисунок 31.

Отличие этой схемы от рис.28 состоит в использовании вместо мультиплексора адреса MSa комбинационного счетчика (Ст). Комбинационная схема (КС), обозначенная как Ст, выполняет функцию: Ст:=A + X_i, где:

Ст – двоичное число на выходе КС;

А – адрес перехода при X_i= 0;

X_i – значение логического условия X_i, выбранного мультиплексором MSx в соответствии с кодом, записанным в разрядах 4 и 5 микрокоманды и интерпретируемым как число 0 или 1.

Функционирование схемы (рис.31) практически не отличается от описанной выше схемы на рис.28.

Используя описанный выше прием формирования адресов переходов, можно еще на один разряд сократить адресную часть микрокоманды. Так как адреса переходов всегда отличаются на 1, то в качестве младшего разряда адреса очередной микрокоманды можно использовать значение логического условия X_i. Формат микрокоманды в этом случае имеет вид, приведенный в таблице. Старшие разряды адресной части (a_C) записаны в микрокоманде, а младший разряд формируется из значения логического условия X_i.

К($У_i$)			x_m		a_C		X_i
1	2	3	4	5	6	7	8

Адрес	Комментарий \ разряды								8
0	a_0; IF X_3 = 0 THEN GOTO 0;								X_3
1	a_1; IF $\neg X_1$ = 0 THEN GOTO 2;								$\neg X_1$
2	a_2; IF X_0 =0 THEN GOTO 6;								X_0
3	a_3; IF X_2 = 0 THEN GOTO 4;								X_2
4	a_4; IF $\neg X_1$ = 0 THEN GOTO 2;								$\neg X_1$
5	a_0; IF X_3 = 0 THEN GOTO 0;								X_3
6	a_3; IF X_2 = 0 THEN GOTO 4;								X_2
7									

В следующей таблице приведена микропрограмма (на основе предыдущего примера), в которой функцию младшего разряда адреса перехода выполняет соответствующее логическое условие X_i, а на рис. 32 – функциональная схема автомата.

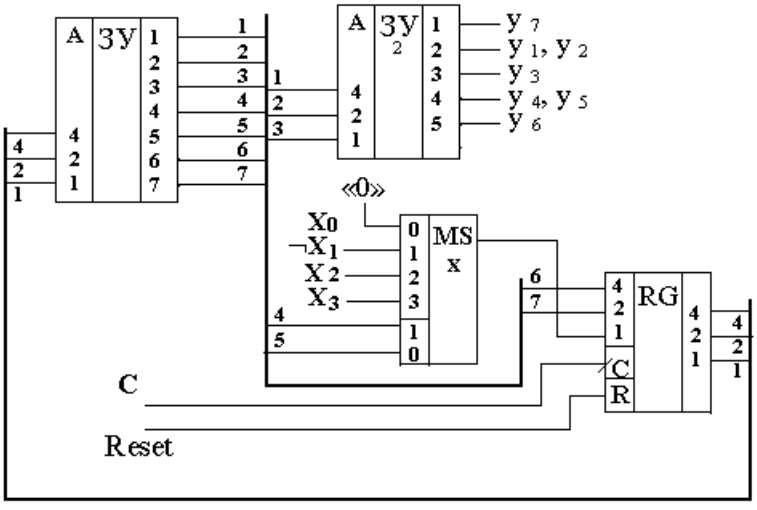


Рисунок 32.

Схема не содержит комбинационного счетчика КС, однако такой формат микрокоманды накладывает на микропрограмму ограничение: при X_i=0 – адрес перехода всегда четный, а при X_i=1 –всегда нечетный.

Таблица для ЗУ 2

Лекция 9.

Тема 4.3. Сегментация памяти в автоматах с программируемой логикой. Функциональные схемы автоматов на программируемой логике и сегментацией памяти. Вопросы оптимизации

Запоминающие устройства, используемые в автоматах с программируемой логикой, имеют достаточно большую емкость, а значит и большую длину адресной части микрокоманды. Большая емкость ЗУ позволяет хранить в нем не одну, а множество микропрограмм, реализующих разные функции, причем каждой из них требуется небольшой объем памяти.

Для того, что бы максимально использовать ЗУ и при этом не увеличивать длину адресной части микрокоманды используют прием, который называется «сегментация ЗУ». Весь объем ЗУ делится на сегменты (страницы), а адрес А, поступающий на адресный вход ЗУ рассматривается как адрес сегмента Ас (обычно старшие разряды адреса) и адрес ячейки в сегменте Ая (обычно младшие разряды адреса). Число сегментов обычно выбирают равным целой степени двойки (2, 4, 8 и т.д. сегментов).

Микропрограммы в ЗУ необходимо размещать так, чтобы было как можно меньше переходов из сегмента в сегмент, а значит лучше всего, что бы первая микрокоманда микропрограммы размещалась в первой ячейке некоторого сегмента. В этом случае в адресной части микрокоманды в качестве адреса перехода будет записываться адрес ячейки Ая и только при необходимости перехода в другой сегмент микрокоманда должна содержать полный адрес: Ас и Ая. Полный адрес необходимо так же записать в регистр адреса при выборе той или иной микропрограммы в ЗУ.

Использование сегментации ЗУ в автоматах с принудительной адресацией и одноадресной микрокомандой рассмотрим на примере автомата, описанного ГСА рис.5 с кодированием групп микрокоманд (как в примере на рис.28).

Пусть ЗУ имеет 8-ми разрядный адрес (емкость ЗУ - 256 слов). Представим ЗУ состоящим из 32-х сегментов, в каждом из которых – 8 ячеек. Таким образом, 5 старших разрядов адреса – это адрес сегмента (Ас), а 3 младших – адрес ячейки (Ая).

Формат микрокоманды практически не отличается от используемого в примере рис.31. Для записи признака перехода из одного сегмента в другой в микрокоманду введем дополнительный разряд (0): значение этого разряда «0» - обычная микрокоманда (как в примере рис.31), если «1» - то все остальные 8 разрядов микрокоманды интерпретируются как полный адрес перехода А = Ас Ая.

Пусть адрес первой микрокоманды микропрограммы А=00000011, т.е. Ас=00000, Ая=011. Микропрограмма с комментариями приведена ниже.(90 бит)

Первые 3 строки (по смыслу) совпадают с примером рис.31. В нулевом разряде записан «0», в адресной части – Ая сегмента с адресом Ас=00000.

Конструкция «GOTO-S Ас, Ая » в комментариях означает переход по полному адресу А = Ас Ая. Следует обратить внимание на то, что в нулевом разряде этой микрокоманды - «0», а в операционной части – Ас (а не $У_i$).

По аналогии с примером (рис.31), в 4-ой строке следовало бы записать конструкцию: «а $_3$; IF $X_2 = 0$ THEN GOTO 7», однако Ая = 7 – это адрес последней ячейки сегмента с адресом Ас=0, и при $X_2 = 1$ адрес Ая = 7 + $X_2 = 8$. Поскольку под Ая отведено 3 разряда, то

после этой операции получим Ая=0. Поэтому в 4-ой строке «GOTO-S 1,0» - переход в сегмент с Ас=00001 в ячейку с Ая=000. 5-ая строка при этом не используется.

В 6-ой строке конструкция «а $_3$; IF $X_2 = 0$ THEN GOTO 1;» по смыслу совпадает с конструкцией «а $_{3;}$ IF $X_2 = 0$ THEN GOTO 4;» (рис.31), причем «GOTO-S 0,3;» в 8-ой строке – переход в начало микропрограммы (где «а $_0$; IF $X_3 = 0$ THEN GOTO 3;» эквивалентно «а $_0$; IF $X_3 = 0$ THEN GOTO 0;»(рис.31).

В 7-ой строке «а $_4$; IF $\neg X_1 = 0$ THEN GOTO 3;» и далее «GOTO-S 0,5;» и «GOTO-S 1,0;» эквивалентно одной строке из примера рис.31 «а $_4$; IF $\neg X_1 = 0$ THEN GOTO 2;»

На рис.33 приведена функциональная схема автомата с принудительной адресацией и сегментацией ЗУ. Для упрощения в ней не показаны элементы, обеспечивающие запись в регистр адреса начального адреса микропрограммы.

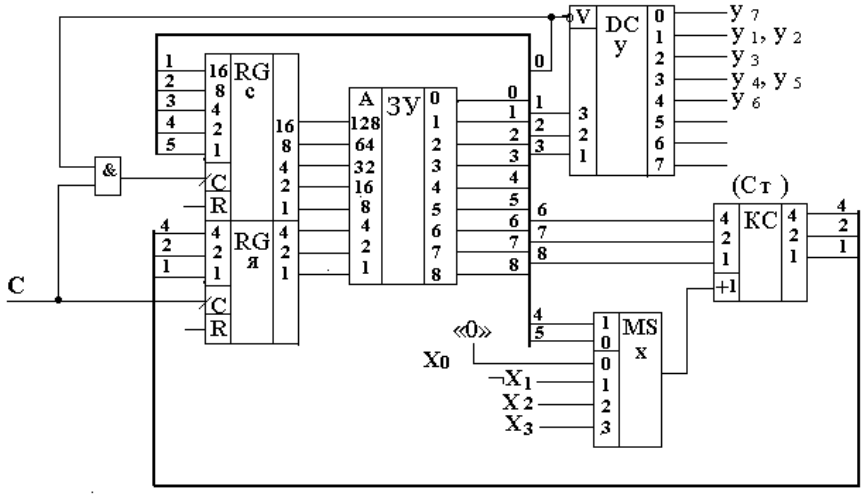


Рисунок 33.

Регистр адреса состоит из двух независимых регистров: RGc – регистр адреса сегмента и RGя – регистр адреса ячейки. Пока значение нулевого разряда слова равно «0» - разрешается работа дешифратора DCy и разряды слова 1..3 интерпретируются как код группы микрокоманд $У_i$. В RGя при каждом синхроимпульсе записывается Ая, формируемый КС (как в примере рис.31). Если значение нулевого разряда слова «1», то работа DCy запрещается (на всех выходах DCy – «0»), а импульс синхронизации через элемент «И» подается не только на вход C RGя , но и RGc. Происходит запись полного адреса перехода в оба регистра: А=Ас Ая.