ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА
(РОСАВИАЦИЯ)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

Д.А. Затучный

ТЕОРИЯ АВТОМАТОВ

Учебное пособие

Утверждено редакционно-издательским советом МГТУ ГА в качестве учебного пособия

Москва
ИД Академии Жуковского
2021

УДК 519.713
ББК 6Ф6.5
З-37

Печатается по решению редакционно-издательского совета
Московского государственного технического университета ГА

Рецензенты:
Феоктистова О.Г. (МГТУ ГА) – д-р техн. наук, доцент;
Акиншин Р.Н. (Секция прикладных проблем при Президиуме Российской академии наук) – д-р техн. наук, профессор

Затучный Д.А.

З-37 Теория автоматов [Текст] : учебное пособие / Д.А. Затучный. – М. : ИД Академии Жуковского, 2021. – 76 с.

ISBN 978-5-907275-94-2

Учебное пособие предназначено для обучающихся по направлению 09.03.01 – Информатика и вычислительная техника, изучающих дисциплину «Теория автоматов».

В данном учебном пособии рассматривается комплекс вопросов, связанных с элементами памяти цифровых автоматов, синтезом автоматов с памятью и заданных микропрограммами, а также синтезом блоков управления.

Большое внимание в курсе уделено практическим реализациям теоретических знаний по построению и синтезу различных автоматов. Делается акцент на разнообразие элементов памяти, построенных на различных триггерах.

В учебном пособии рассматриваются такие вопросы как: основные структуры автоматов, элементы памяти цифровых автоматов, методики синтеза автоматов с памятью и заданных с помощью микропрограммы, а также синтез блоков управления с распределителями импульсов.

Рассмотрено и одобрено на заседаниях кафедры 26.01.2021 г. и методического совета 26.01.2021 г.

УДК 519.713
ББК 6Ф6.5
Св. тем. план 2021 г.
поз. 31

ЗАТУЧНЫЙ Дмитрий Александрович
ТЕОРИЯ АВТОМАТОВ

Учебное пособие

В авторской редакции

Подписано в печать 14.05.2021 г.
Формат 60x84/16 Печ. л. 4,75 Усл. печ. л. 4,42
Заказ № 746/0330-УП02 Тираж 30 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68 E-mail: zakaz@itsbook.ru

ISBN 978-5-907275-94-2

Введение

Автомат в теории автоматов – это математическая модель устройства, преобразующего дискретную информацию.

В предыдущем учебном пособии «Основы теории автоматов» по дисциплине «Теория автоматов» были подробно рассмотрены вопросы, связанные с элементами теории алгоритмов, основами алгебры логики, а также методами анализа и синтеза комбинационных схем. При этом следует отметить, что для полного усвоения этой дисциплины, обучающиеся должны также овладеть знаниями о структуре различных автоматов и методике их синтеза по алгоритму, включающему несколько этапов.

Студенты при изучении данного учебного пособия, предназначенного для изучения дисциплины «Теория автоматов» в рамках учебного плана направления подготовки 090301 «Информатика и вычислительная техника», получают теоретические знания, подкреплённые практическими реализациями, по различным способам задания автоматов, их элементах памяти в виде различных триггеров, а также знакомятся с методикой синтеза автомата с памятью.

1. Основные структуры автоматов

Существует большое количество различных структур автоматов. В общем случае любой автомат может иметь различную структуру. Однако структуру любого автомата можно преобразовать к одной из двух типовых структур:

Автомат 1-го рода или автомат Мили.

Автомат 2-го рода или автомат Мура.

1.1. Автомат Мили.

Автомат Мили (англ. *Mealy machine*) — конечный автомат, выходная последовательность которого (в отличие от автомата Мура) зависит от состояния автомата и входных сигналов. Это означает, что в графе состояний каждому ребру соответствует некоторое значение (выходной символ). В вершины графа автомата Мили записываются выходящие сигналы, а дугам графа приписывают условие перехода из одного состояния в другое, а также входящие сигналы. Назван именем Джорджа Мили, учёного в области математики и компьютерных наук, придумавшего этот автомат.

Диаграмма состояний автомата Мили приведена на рис. 1.1.

Автомат Мили — совокупность $A = (S, X, Y, \delta, \lambda, S_0)$,

где

S – конечное непустое множество состояний автомата;

X – конечное непустое множество входных символов;

Y – конечное непустое множество выходных символов;

$\delta : S \times X \to S$ - функция переходов, отображающая пары состояние/входной символ на соответствующее следующее состояние;

$\lambda : S \times X \to Y$ - функция выходов, отображающая пары состояние/входной символ на соответствующий выходной символ;

$S_0 \in S$ - начальное состояние.

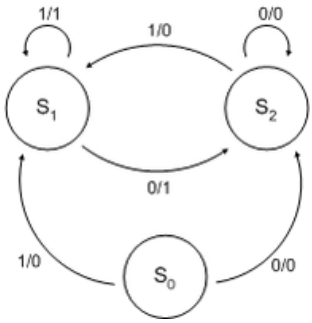

Рис.1.1. Граф состояний автомата Мили

1.2. Обобщённая структура автомата Мили.

Структура автомата Мили представлена на рис. 1.2.

Автомат Мили представляется в виде двух комбинационных схем и памяти, состоящей из отдельных элементов памяти.

Первая комбинационная схема (в дальнейшем будем обозначать её КС1) имеет две группы входов. Одна группа входов является входами автомата в целом ($x_1, x_2, \ldots, x_n$). На другую группу входов поступают сигналы из памяти автомата ($q_t^1, q_t^2, \ldots, q_t^k$)., т.е. состояние автомата. По отношению к автомату в целом эти сигналы являются внутренними. На выходах схемы КС1 формируются сигналы ($q_{t+1}^1, q_{t+1}^2, \ldots, q_{t+1}^k$), которые поступают на входы элементов памяти ($ЭП_r$).

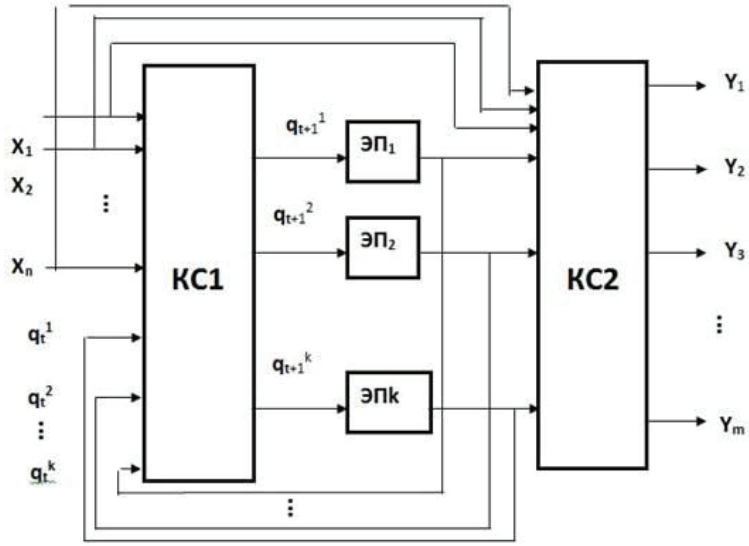

Рис. 1.2. Структура автомата Мили

После записи в память эти сигналы в следующем такте будут представлять собой следующее состояние автомата (Q_{t+1}). Таким образом схема КС1 является схемой формирования следующего состояния автомата.

Вторая комбинационная схема (КС2) называется выходным преобразователем и служит для формирования выходных сигналов автомата ($y_1, y_2, \ldots, y_m$).. На её входы поступают те же сигналы, что и на входы схемы КС1. Поэтому выходные сигналы в автомате Мили зависят как от состояния, так и от входных сигналов, поступающих в данном такте. Обобщенная структура автомата Мили имеет вид, представленный на рис.1.3.

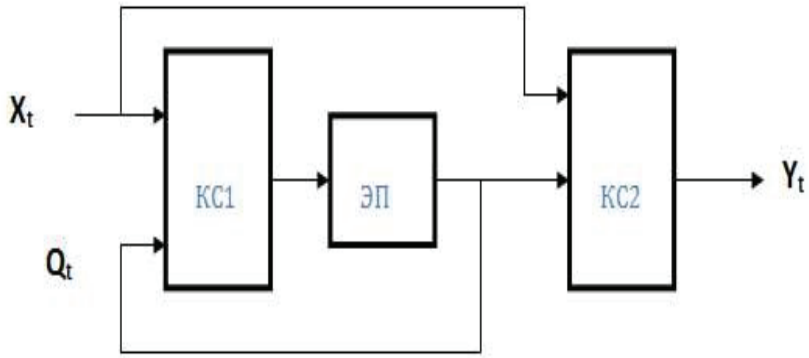

Рис. 1.3. Обобщённая структура автомата Мили

Работа автомата Мили описывается в общем виде уравнениями переходов и выходов:

$\mathbf{Y_t = Y(X_t, Q_t)}$; (1.1)

$\mathbf{Q_{t+1} = Q(X_t, Q_t)}$.

Уравнение (функция) переходов $\mathbf{Q_{t+1}=Q(X_t, Q_t)}$ определяет условия перехода автомата из одного состояния в другое. Это уравнение задает логику работы комбинационной схемы КС1.

Уравнение (функция) выходов $\mathbf{Y_t=Y(X_t,Q_t)}$ определяет условия формирования определенных выходных сигналов. Это уравнение задает логику работы комбинационной схемы КС2.

Анализ уравнений (1.1) показывает, что логика работы автомата Мили совпадает с логикой работы автомата обобщенной структуры, представленной на рис. 1.3.

1.3. Автомат Мура

Автомат Мура в теории вычислений — конечный автомат, выходное значение сигнала в котором зависит лишь от текущего состояния данного автомата, и не зависит напрямую, в отличие от автомата Мили, от входных значений. Автомат Мура назван в честь описавшего его свойства Эдварда Ф. Мура, опубликовавшего исследования в 1956 году.

Пример автомата Мура приведён на рис. 1.4.

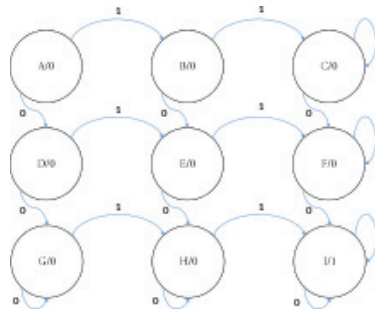

Рис. 1.4. Пример автомата Мура

Автомат Мура может быть определён как кортеж из 6 элементов, включающий:

множество внутренних состояний *S* (внутренний алфавит);

начальное состояние s_0;

множество входных сигналов *X* (входной алфавит);

множество выходных сигналов *Y* (выходной алфавит);

функция переходов $\phi : S \times X \to S$;

функция вывода $G : S \to Y$.

Для любого автомата Мура существует эквивалентный ему автомат Мили: любой автомат Мура путём добавления ряда внутренних состояний может быть преобразован в автомат Мили. Обратное, строго говоря, неверно: дело в том, что сигнал на выходе автомата Мура зависит только от входного сигнала в *предыдущие* моменты времени, а выходной сигнал для автомата Мили может зависеть от входного сигнала и в *текущий* момент времени. Для автомата Мили можно в общем случае построить лишь автомат Мура, который ему *почти* эквивалентен: а именно его выход будет сдвинут во времени на 1.

1.4. Обобщённая структура автомата Мура

Автомат Мура имеет структуру, показанную на рис. 1.5.

Эта структура внешне очень незначительно отличается от структуры автомата Мили. Как видно на рис.1.5, это отличие заключается в том, что в автомате Мура входной сигнал не поступает на комбинационную схему КС2 (схему формирования выходов).

Работа автомата Мура описывается следующими уравнениями переходов и выходов:

$$\mathbf{Y_t = Y(Q_t)}\ ; \qquad (1.2)$$

$$\mathbf{Q_{t+1} = Q\ (\ X_t, Q_t)}.$$

Из уравнений (1.2) видно, что выходной сигнал автомата Мура зависит только от текущего состояния и не зависит от входного сигнала в ***данном***

такте. Следует особо отметить, что выходной сигнал зависит от входных сигналов, поступивших в предыдущих тактах, поскольку от них зависит текущее состояние. Таким образом, выходной сигнал автомата Мура задержан на один такт по отношению к входному сигналу.

Иногда выходные сигналы автомата Мура совпадают с сигналами состояний. В этом случае автомат не имеет комбинационной схемы КС2 и называется автоматом без выходного преобразователя. Обобщенная структура такого автомата показана на рис. 1.6.

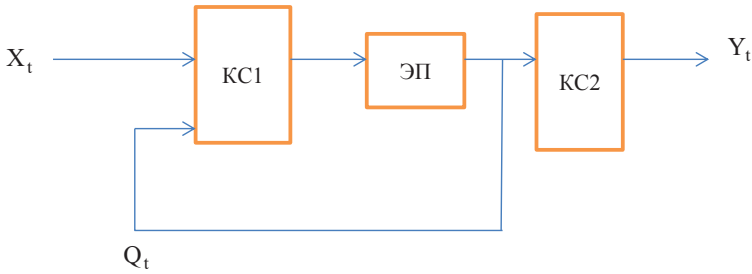

Рис. 1.5. Обобщённая структура автомата Мура

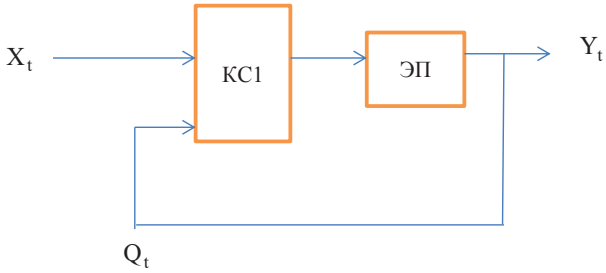

Рис. 1.6. Обобщённая структура автомата Мура без выходного преобразователя

Алгоритм работы автомата Мура без выходного преобразователя описывается следующими уравнениями переходов и выходов:

$$Y_t = Q_t ; \qquad (1.3)$$

$$Q_{t+1} = Q (X_t, Q_t).$$

Контрольные вопросы

1. Укажите основное отличие автоматов с памятью от комбинационных схем по их составу.
2. Укажите основное отличие автоматов с памятью от комбинационных схем по логике их работы.
3. Чем определяется значение выходного сигнала для комбинационной схемы?
4. От чего зависит выходной сигнал автомата Мили?
5. От чего зависит выходной сигнал автомата Мура?
6. От чего зависит следующее состояние автомата?
7. Чем отличаются структуры автоматов Мили и Мура?
8. Чем отличаются функции выходов автоматов Мили и Мура?
9. Можно ли по графу автомата определить тип автомата (автомат Мили или Мура)?
10. Зависит ли выходной сигнал автомата Мура от предыстории входных сигналов?

2. Элементы памяти цифровых автоматов

2.1. Триггеры. Общие сведения о триггерах

Для построения цифровых устройств, кроме логических элементов, требуются элементы памяти, осуществляющие хранение двоичных кодов в течение требуемого времени. В зависимости от способа хранения информации элементы памяти могут быть статическими, позволяющими хранить двоичную информацию сколь угодно долго, и динамическими, хранящими информацию в течение ограниченного отрезка времени.

В качестве статического элемента памяти используются бистабильные ячейки, имеющие два устойчивых состояния. Бистабильные ячейки могут быть построены на двух логических элементах И-НЕ или ИЛИ-НЕ, соединённых перекрёстными связями.

На основе элементов памяти строятся так называемые триггеры. Триггер – это цифровая электронная схема с двумя устойчивыми состояниями, которые устанавливаются при подаче соответствующей комбинации входных сигналов и сохраняются после снятия этих сигналов.

Триггер имеет несколько входов и два выхода – прямой и инверсный (Q и $\bar{Q}$). Сигналы на выходах триггера всегда имеют различные значения. Если на прямом выходе сигнал равен 1, то на инверсном – 0 и наоборот. Состояние триггера определяется значением сигнала на прямом выходе (Q). Если сигнал на прямом выходе равен 1, то триггер находится в состоянии 1.

Триггеры могут быть синхронными или асинхронными. Если изменения сигнала Q происходит только при наличии специального сигнала C, являющегося сигналом синхронизации, то такой триггер называется синхронным триггером. Синхронизация триггера может происходить либо по уровню сигнала, либо по фронту сигнала (переднему или заднему).

Асинхронный триггер не имеет входа синхронизации, поэтому переключение триггера происходит только при изменении на входах информационных сигналов X.

Логика переключения триггера из одного состояния в другое зависит от количества и назначения входов. Наиболее часто используются в цифровой технике следующие типы триггеров: RS-триггеры, D-триггеры, T-триггеры и JK-триггеры. Буквами R,S,J,K,D и T обозначаются информационные входы триггеров.

2.2. RS - триггер

2.2.1. Асинхронный RS-триггер

RS-триггер имеет два информационных входа R и S. Вход S используется для установки триггера в состояние 1, а вход R – для установки в состояние 0. Поэтому RS триггер называют триггером с установочными входами.

Работа триггера описывается таблицей переходов, представленных в таблице 2.1.

Таблица 2.1

Входы		Состояния	
S	**R**	**0**	**1**
0	0	0	1
0	1	0	0
1	0	1	1
1	1	-	-

Входами в таблице 2.1 являются значения входных сигналов R и S, а также значения состояний триггера в текущий момент времени Q_t. В самой таблице приведены значения состояний триггера в следующий момент времени Q_{t+1}. В соответствии с таблицей переходы триггера из одного состояния в другое происходят, если на вход R или S подаётся сигнал 1. Из таблицы видно, что при R=0 и S=0 состояние триггера не меняется. При R=0 и S=1 триггер переходит в состояние 1 независимо от того, в каком состоянии он находился до изменения входных сигналов. При R=1 и S=0 триггер переходит в состояние 0. Комбинация сигналов R=1 и S=1 является запрещённой и состояние триггера при этом не определено.

Таблица переходов триггера может быть интерпретирована как таблица истинности комбинационной схемы, в которой значения сигналов на входах можно рассматривать как логические переменные, а Q_{t+1} - как логическую функцию. Таблица переходов асинхронного RS-триггера в этом случае будет иметь вид таблицы 2.2.

Таблица 2.2

Входы		Текущее состояние	Следующее состояние
R	**S**	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	Запрещённая комбинация
1	1	1	

Уравнение переходов триггера может быть получено из таблицы 2.1 или 2.2. После минимизации уравнение имеет вид:

$$Q_{t+1} = S + \bar{R} \cdot Q_t \qquad (2.1)$$

Из уравнения следует, что при S=1, R=0 всегда $Q_{t+1} = 1$, а при S=0, R=1 всегда $Q_{t+1} = 0$. Комбинация сигналов S=1, R=1 является запрещённой.

Функциональная схема асинхронного RS-триггера на элементах ИЛИ-НЕ показана на рис. 2.1.

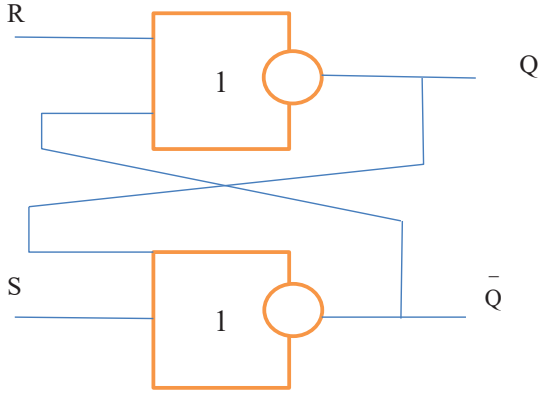

Рис. 2.1 Функциональная схема асинхронного RS-триггера

2.2.2. Синхронный RS-триггер

Синхронный триггер дополнительно имеет вход синхронизации C, на который поступает синхросигнал. Информационные сигналы R и S воздействуют на состояние триггера только при значении синхросигнала C=1. При этом логика работы синхронного триггера совпадает с логикой

асинхронного триггера (таблица 2.1). При C=0 триггер не меняет своего состояния при любой комбинации сигналов на информационных входах.

Логика переходов синхронного RS-триггера может быть описана таблицей 2.3. Функциональные схемы синхронных RS-триггеров могут быть реализованы на элементах И-НЕ и И-ИЛИ-НЕ.

Таблица 2.3

Входы			Состояния	
C	**S**	**R**	**0**	**1**
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	-	-

Из таблиц 2.1 и 2.3 может быть получено уравнение синхронного RS-триггера:

$$Q_{t+1} = S \cdot C + \left(\bar{R} + \bar{C} \right) \cdot Q_t . \qquad (2.2)$$

При C=0 $Q_{t+1} = Q_t$, т.е. триггер не меняет своего состояния. При C=1

$$Q_{t+1} = S + \bar{R} \cdot Q_t , \qquad (2.3)$$

т.е. совпадает с уравнением для асинхронного триггера.

2.3. D-триггер

В вычислительной технике широко применяется D-триггер, который реализует функцию временной задержки входного сигнала. D-триггер имеет информационный вход. Логика работы асинхронного D-триггера описывается таблицей переходов, которая имеет вид таблицы 2.4.

Таблица 2.4.

Логика работы асинхронного D-триггера

Вход	Состояния	
D	**0**	**1**
0	0	0
1	1	1

По таблице 3.4 может быть записано уравнение переходов D-триггера:

$Q_{t+1} = D_t$,

где t – текущий момент времени;

t+1 – последующий момент времени.

Как видно из уравнения, в асинхронном D-триггере состояние (выходной сигнал) Q_{t+1} повторяет значение выходного сигнала D_t. Поэтому асинхронный D-триггер по существу не является элементом памяти и рассматривается только как основа для построения синхронного D-триггера.

Функциональная схема асинхронного D-триггера, построенного на основе RS-триггера, показана на рис. 2.2.

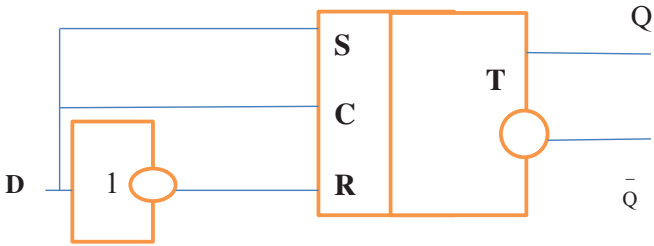

Рис. 2.2. Функциональная схема асинхронного D-триггера

Для построения счётчиков, регистров и других цифровых схем используются синхронные D-триггеры. Логика работы синхронного D-триггера описывается таблицей переходов, которая имеет вид таблицы 2.5.

Таблица 2.5.

Логика работы синхронного D-триггера

Входы		Состояния	
C	**D**	**1**	**0**
0	0	0	1
0	1	0	1
1	0	0	0
1	1	1	1

Уравнение переходов синхронного триггера, записанное по таблице 2.5, имеет следующий вид:

$$\bar{Q}_{t+1} = C_t Q_t + C_t \cdot D_t. \qquad (2.4)$$

В соответствии с уравнением синхронный D-триггер при C=0 сохраняет своё состояние, а при C=1 работает как асинхронный.

2.4. Т-триггер

Т-триггер имеет один информационный вход. Логика работы асинхронного Т-триггера может быть описана таблицей переходов, которая приведена в таблице 2.6.

Таблица 2.6

Логика работы асинхронного Т-триггера

Вход	Состояния	
T	**0**	**1**
0	0	1
1	1	0

По таблице 2.6. может быть получено следующее уравнение асинхронного Т-триггера:

$$Q_{t+1} = TQ_t + \bar{T} \cdot Q_t. \qquad (2.5)$$

Как видно из таблицы 2.6 и уравнения триггера, при Т=1 асинхронный Т-триггер изменяет своё состояние, а при Т=0 состояние триггера не изменяется.

Так как Т-триггер суммирует (или подсчитывает) по модулю два количество единиц, поступающих на его информационный вход, то Т-триггер называют также триггером со счётным входом.

Логика работы синхронного Т-триггера описывается таблицей переходов, которая имеет вид таблицы 2.7.

Таблица 2.7

Логика работы синхронного Т-триггера

Входы		Состояния	
C	**T**	**0**	**1**
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Из таблицы 2.7 видно, что при С=0 триггер не изменяет своего состояния, а при С=1 работает как асинхронный Т-триггер.

Функциональная схема Т-триггера может быть построена на основе синхронного RS-триггера.

Схема асинхронного Т-триггера приведена на рис. 2.3, а синхронного Т-триггера на рис. 2.4.

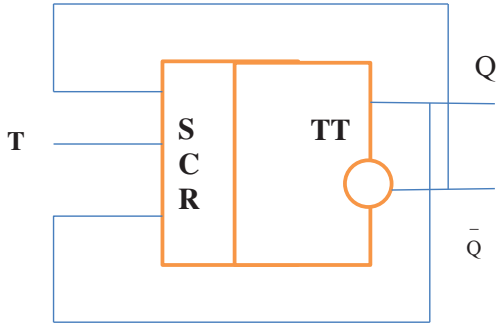

Рис.2.3. Схема асинхронного Т-триггера

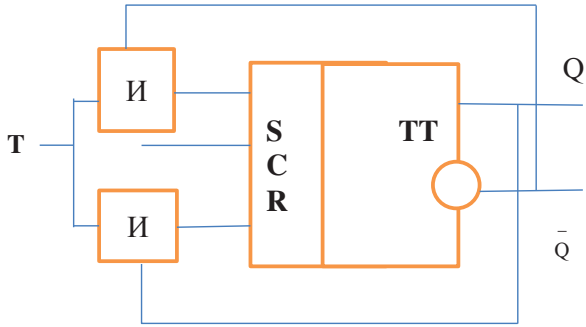

Рис. 2.4. Схема синхронного Т-триггера

2.5. JK – триггер

JK-триггер называется также универсальным триггером. Универсальность схемы JK-триггера состоит в том, что простой коммутацией входов и выходов можно получать схемы других типов триггеров.

JK-триггер имеет два информационных входа. Вход J используется для установки триггера в состояние 1, а вход K – для установки в состояние 0, т.е. входы J и K аналогичны входам R и S RS-триггера. Отличие заключается в том, что на входы J и K могут одновременно поступать сигналы 1. В этом случае JK-триггер изменяет своё состояние.

Таблица переходов JK-триггера представлена в таблице 2.8.

Таблица 2.8

Таблица переходов JK-триггера

Входы		Состояния	
J	**K**	$\bar{Q}(0)$	**Q(1)**
0	0	0	1
0	1	0	0
1	0	1	1
1	1	1	0

Из таблицы 2.8 можно получить следующее уравнение JK-триггера:

$$Q_{t+1} = J_t \bar{Q}_t + \bar{K}_t \cdot Q_t. \qquad (2.6)$$

Следовательно, при $J_t = 1$, $K_t = 0$ всегда $Q_{t+1} = 1$, а при $J_t = 0$, $K_t = 1$ всегда $Q_{t+1} = 0$, т.е. JK-триггер работает как RS-триггер, если рассматривать входы J и K как входы S и R.

В свою очередь, при $J_t = 1$, $K_t = 1$ $Q_{t+1} = \bar{Q}_t$, т.е. триггер переходит в противоположное состояние (работает как Т-триггер).

Функциональная схема JK-триггера показана на рис. 2.5.

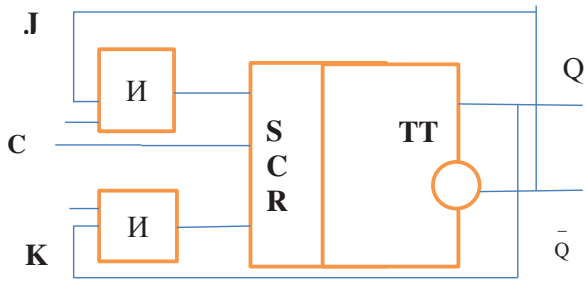

Рис. 2.5. Функциональная схема JK-триггера

JK – триггер, кроме основных информационных входов и входа синхронизации может иметь также дополнительные информационные входы, например, дополнительные инверсные асинхронные входы R и S, которые используются для установки триггера в 0 или 1 независимо от значения сигнала на входе синхронизации. Кроме того, триггер может иметь несколько входов J или K, объединённых по схеме И.

В заключение следует отметить, что из любого триггера можно построить схему триггера другого типа при помощи дополнительных логических элементов.

Контрольные вопросы

1. Какие триггеры применяются в цифровой технике и в чём их различия?
2. Как влияет вход синхронизации на работу триггера?
3. Поясните логику переходов RS-триггера.
4. Поясните логику переходов D-триггера.
5. Поясните логику переходов T-триггера.
6. Поясните логику переходов JK-триггера.
7. Запишите уравнение D-триггера.
8. Запишите уравнение синхронного RS-триггера.
9. Запишите уравнение T-триггера.
10. Запишите уравнение JK-триггера.

3. Методика синтеза автоматов с памятью

3.1. Последовательность синтеза автоматов с памятью

Алгоритм работы автомата с памятью вначале задается в виде словесного описания. В этом случае синтез автоматов с памятью состоит из следующих этапов:

1) формализация задания;
2) выбор типа автомата;
3) составление графа состояний автомата;
4) составление таблицы переходов и выходов;
5) кодирование состояний;
6) составление кодированной таблицы переходов и выходов;
7) выбор типа элементов памяти;
8) преобразование таблицы переходов и выходов в таблицу функций возбуждения триггеров;
9) запись функций возбуждения и функций выходов в СДНФ;
10) минимизация функций возбуждения и функций выходов;
11) выбор типа логических элементов;
12) преобразование функций возбуждения и функций выходов к виду, удобному для реализации на выбранных логических элементах;
13) построение функциональной схемы автомата;
14) проверка правильности работы автомата.

Приведенная последовательность соответствует самому общему случаю синтеза. В некоторых случаях в зависимости от способа задания автомата и других исходных данных некоторые этапы могут отсутствовать или совмещаться с другими этапами.

3.2. Основные этапы синтеза

Как было показано в разделе 1, типовая структура автомата с памятью состоит из двух комбинационных схем (КС1 и КС2), между которыми установлены элементы памяти. Поэтому синтез автомата с памятью сводится к синтезу двух комбинационных схем. Для этого необходимо построить таблицу переходов и выходов автомата, которая представляет собой перестроенную таблицу истинности функций переходов и выходов. Поскольку вид этой таблицы зависит от типа автомата и типа элементов памяти, исходная таблица подвергается некоторым преобразованиям. Рассмотрим содержание отдельных этапов синтеза.

Формализация задания включает определение числа входов и выходов автомата и их обозначение, определение алфавита входных и выходных сигналов и установление соответствия логических значений входных и выходных сигналов и их смысла в соответствии с заданием.

Выбор типа автомата заключается в том, что для разработки принимается одна из двух типовых структур - автомат Мура или Мили. Вообще схемы автоматов Мура и Мили, реализующих один и тот же алгоритм, отличаются друг от друга по сложности. Однако в настоящее время не существует способа определить заранее, какая из двух структур окажется проще. Поэтому для обоснованного выбора типа автомата необходимо построить схемы автоматов Мура и Мили и сравнить их сложность.

Составление графа состояний автомата выполняется на основе тщательного анализа заданного алгоритма работы автомата. Составление графа начинается с начального состояния, для которого рассматривается поведение автомата для каждого из входных сигналов, возможных для данного состояния. При этом устанавливается необходимость изменения состояния автомата и значение сигналов на выходах автомата. Состояние автомата необходимо изменить, если автомат должен запомнить факт поступления данного входного сигнала, т.е. если данный входной сигнал должен изменить поведение автомата.

Например, если автомат с одним входом должен определить факт поступления на вход последовательности входных сигналов «101», то в начальном состоянии при поступлении символа «0» автомат не меняет состояния, так как этот символ не является началом нужной последовательности. При поступлении же символа «1» автомат должен запомнить, что первый символ последовательности уже поступил. Запоминание этого факта автомат выполняет, меняя свое состояние. Образно говоря, автомат в этом случае «настораживается».

Необходимо отметить, что при смене состояния автомат может переходить как в некоторое новое состояние, которого еще нет на составляемом графе, так и в одно из уже имеющихся на графе состояний.

Составление графа состояний автомата является ключевым этапом синтеза, так как граф представляет собой строгую форму задания алгоритма

работы автомата. Все последующие этапы синтеза носят во многом формальный характер и достаточно легко могут быть реализованы с помощью специальных программ.

Составление таблицы переходов и выходов выполняется по графу состояний автомата. По существу таблица переходов и выходов является табличной формой представления графа. Вид таблицы переходов и выходов для автоматов Мили и Мура несколько отличается.

Кодирование состояний заключается в том, что каждому состоянию автомата (Q_i) ставится в соответствие состояние отдельных элементов памяти (q_i). Например, для автомата с тремя элементами памяти за состояние Q_0 может быть принято такое состояние, при котором все три элемента памяти находятся в состоянии «0» (в каждом элементе памяти записан «0»). Обычно кодирование состояний сводится к замене обозначения состояний вида Q_i их двоичными номерами. Такой способ кодирования состояний называют естественным. Количество элементов памяти (n) определяется числом состояний автомата (N):

$$n = \log_2 N \uparrow,$$

где $\uparrow$ означает, что результат округляется в большую сторону.

Составление кодированной таблицы переходов и выходов означает замену обозначения состояний вида Q_i их двоичными номерами. При этом обозначения выходных сигналов остаются без изменения.

Выбор типа элементов памяти выполняется исходя из их наличия или других условий, выходящих за рамки теории автоматов, например стоимости, надежности и т.д. Для получения наиболее простой схемы следует выполнить синтез с использованием различных элементов и оценить сложность полученных схем.

Преобразование кодированной таблицы переходов и выходов в таблицу функций возбуждения триггеров выполняется с использованием характеристической таблицы триггера соответствующего типа.

Запись функций возбуждения и функций выходов в СДНФ выполняется по таблице функций возбуждения триггеров так же, как запись функций по таблице истинности.

Минимизация функций возбуждения и функций выходов производится с использованием графических или аналитических методов минимизации логических функций так же, как и при синтезе комбинационных схем.

Выбор типа логических элементов и преобразование функций возбуждения и функций выходов к виду, удобному для реализации на выбранных логических элементах выполняются так же, как и при синтезе комбинационных схем.

Построение функциональной схемы автомата следует начинать с уточнения общей структуры (состава и расположения комбинационных схем КС1 и КС2). Далее в соответствии с функциями возбуждения и функциями выходов определяются типы и количество логических элементов, а также

связи между элементами. Функциональная схема вычерчивается с использованием условно-графических обозначений (УГО) элементов в соответствии с требованиями ГОСТов.

Проверка правильности работы автомата выполняется для различных сочетаний текущего состояния и входных сигналов автомата. Для автомата Мили по заданному текущему состоянию (Q_t) и входному сигналу (X_t) вначале определяется значение выходного сигнала (Y_t), а затем определяется следующее состояние автомата (Q_{t+1}). Для автомата Мура по значению Q_t и X_t вначале определяется следующее состояние автомата (Q_{t+1}) а затем по значению Q_{t+1} определяется значение выходного сигнала (Y_{t+1}). Полученные значения Q_{t+1} и Y_t сравниваются с данными таблицы переходов и выходов. При совпадении ожидаемых и полученных результатов делается вывод о правильности синтеза автомата.

Для реальных автоматов с большим количеством состояний и входов выполнение проверки вручную становится затруднительным. Для этой цели в настоящее время используются специальные программы.

3.3. Пример синтеза автомата с памятью

Задание. Синтезировать счётчик по модулю 4 с порядком счёта 2-3-1-0. Счётчик синтезировать как автомат Мили с использованием логических элементов И-НЕ и Т-триггеров

1. Формализация задания

На вход счётчика поступает последовательность символов 0 и 1. Счётчик должен подсчитывать количество поступивших на его вход символов 1. На выходе счётчика должно фиксироваться количество символов 1, подсчитанных по модулю 4. При двоичном кодировании выходных сигналов счётчик должен иметь один вход. Тогда обобщенная схема счётчика может иметь вид:

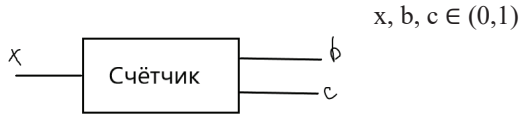

Рис. 3.1. Обобщённая схема счётчика

2. Выбор типа цифрового автомата

Задан автомат Милли

3. Составление графа состояний.

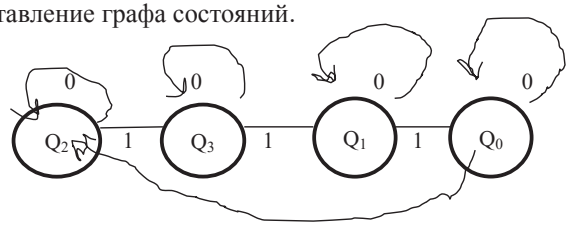

Рис. 3.2. Граф состояний

4. Составление отмеченной таблицы переходов.

Таблица 3.1.

Входы	Состояния и выходы			
x	Q_0	Q_1	Q_2	Q_3
0	Q_0 ,Y_0	Q_1,Y_1	Q_2,Y_2	Q_3,Y_3
1	Q_2 ,Y_2	Q_0, Y_0	Q_3,Y_3	Q_1,Y_1

5. Кодирование состояний.

Кол-во состояний - 4

$n = \log_2 4 = 2$

То есть, у нас будет 2 элемента памяти. Обозначим их q1 и q2

Используем естественное кодирование:

$Q0 \rightarrow \bar{q}1\bar{q}2 \rightarrow 00$

$Q1 \rightarrow \bar{q}1q2 \rightarrow 01$

$Q2 \rightarrow q1\bar{q}2 \rightarrow 10$

$Q3 \rightarrow q1q2 \rightarrow 11$

5. Составление кодированной таблицы.

Таблица 3.2.

Входы	Состояния и выходы			
X	Q_0	Q_1	Q_2	Q_3
	$\bar{q}1\bar{q}2$	$\bar{q}1q2$	$q1\bar{q}2$	$q1q2$
0	00 ,Y_0	01, Y_1	10, Y_2	11, Y_3
1	10, Y_2	00, Y_0	11,Y_3	01,Y_1

7. Выбор типа триггера.

В качестве элементов памяти заданием предусмотрены Т-триггеры, таблица переходов представлена ниже.

Таблица 3.3

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

8. Преобразование кодированной таблицы переходов в таблицу функций возбуждения триггеров.

Таблица 3.4.

Входы	Состояния		Функции возбуждения	
Переход при C=1	Q_n	Q_{n+1}	T1	T2
	q1	q2		
2 -> 3	1 0	1 1	0	1
3->1	1 1	0 1	1	0
1->0	0 1	0 0	0	1
0->2	0 0	1 0	1	0

9. Запись функций возбуждения и функций выхода в СДНФ

T1 = $\overline{q2}q1$ v$\overline{q1}q2$

T2 = $\overline{q1}q2$ v$\overline{q2}q1$

Выходные сигналы счетчика соответствуют значениям выходов q1 и q2 триггеров.

10. Выбор типа логических элементов.

В качестве логических элементов заданы элементы И-НЕ.

11. Преобразование функций возбуждения.

Используя правила двойного отрицания и де Моргана, выполним преобразование функций в базис И-НЕ:

T1 = $\overline{q2}q1$ v$\overline{q1}q2$ -> $\overline{\overline{q1}\overline{q2}} * \overline{q2\overline{q1}}$

T2 = $\overline{q1}q2$ v$\overline{q2}q1$-> $\overline{\overline{q1}\overline{q2}} * \overline{q2\overline{q1}}$

12. Построение функциональной схемы**.**

По полученным выражениям составим схему счетчика на Т-триггерах и логических элементах И-НЕ.

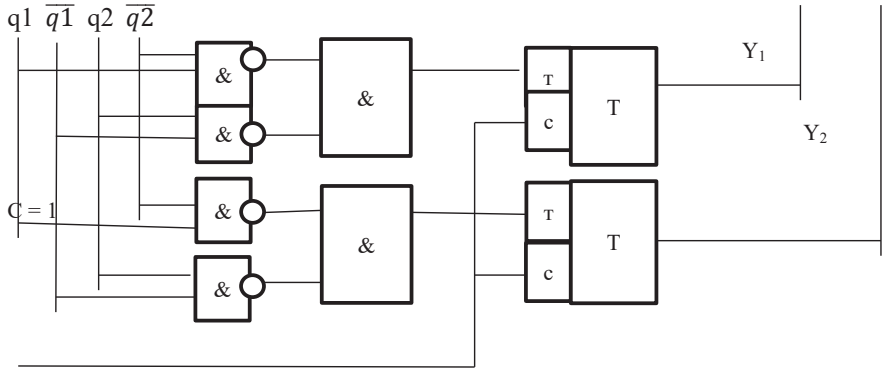

Рис. 3.3. Функциональная схема автомата

13. Проверка правильности работы схемы счетчика.

При контроле работоспособности автомата определяется правильность смены состояний и выходных сигналов (в нашем случае q1 и q2).

При запуске работы схемы автомат находится в состоянии 2 (q1=1, q2=0). После поступления на вход триггеров сигнала синхронизации (C=1) автомат переключится в состояние 1 (q1=1, q2=1).

Второй сигнал синхронизации (C=1) переключается автомат в состояние 3 (q1=0, q2=1).

Следующий сигнал синхронизации (C=1) переключит автомат в состояние 2 (q1 = 0, q2 = 0).

Четвертый сигнал синхронизации переключается автомат в исходное состояние 0 (q1=1 q2=0).

В соответствии с уравнениями переходов и выходов функциональная схема автомата может быть представлена. Проверка работы полученной схемы показала, что счетчик работает с корректно, в заданном режиме счета 2-3-1-0.

Контрольные вопросы

1. Какие способы описания алгоритмов обычно используют при синтезе автоматов с памятью?
2. Перечислите основные этапы синтеза автоматов с памятью.
3. От чего зависит множество входных сигналов автомата с памятью?
4. В каких случаях меняется состояние автомата с памятью?
5. Как определить число возможных состояний автомата с памятью, если известно число элементов памяти?
6. Как по таблице переходов записать функции переходов?
7. Как проверить правильность работы автомата с памятью?
8. Как влияет тип автомата с памятью на последовательность проверки

его работоспособности?

9. Какими уравнениями описывается логика работы комбинационной схемы, формирующей состояния автомата?

4. Частные случаи синтеза автоматов с памятью

4.1. Общие сведения о микрокомандах и микропрограммах.

Управляющий автомат, реализующий микропрограмму выполнения операции обработки информации, часто называют микропрограммным автоматом. Это вытекает из следующей интерпретации взаимодействия операционного блока и блока управления в процессе выполнения каких-либо операций по обработке информации:

а) Любая операция, реализуемая операционным устройством, рассматривается в виде последовательности элементарных неделимых актов обработки информации, выполняемых в течение одного такта автоматного времени (одного шага алгоритма), называемых *микрооперациями.* Множество микроопераций и сигналов, инициирующих их выполнение, обозначают одними и теми же символами, которые составляют выходной структурный алфавит $Y = \{y_1, y_2, ..., y_N\}$. В случае, если несколько микроопераций реализуются в операционном блоке одновременно, то это подмножество микроопераций называют *микрокомандой.*

б) Для управления порядком следования микрокоманд используются *логические условия*, которые в зависимости от результатов преобразования информации в операционном блоке могут принимать значения 1 или 0. Множество логических условий (осведомительных сигналов) обозначают символами, которые составляют входной структурный алфавит $X = \{x_1, x_2, ..., x_L\}$.

в) Последовательность выполнения микрокоманд, определяемая функциями перехода блока управления, записывается в виде алгоритма, представленного в терминах микроопераций и логических условий и называется *микропрограммой.*

Управляющие микропрограммные автоматы аппаратно могут быть реализованы на основе так называемой жесткой логики и программируемой логики.

Автомат с жесткой логикой строится на базе использования логических элементов и элементов памяти (элементарных автоматов с двумя внутренними состояниями). Изменить алгоритм работы такого автомата нельзя, не изменяя соединений между элементами. Для таких автоматов характерны высокое быстродействие, определяемое только задержками используемых логических элементов и элементов памяти, пропорциональный рост объема оборудования в зависимости от сложности реализуемого алгоритма и малые удельные затраты оборудования при реализации простых микропрограмм. Однако автоматы с жесткой логикой не обладают гибкостью при внесении изменений в алгоритм их функционирования, необходимость в

которых особенно часто возникает в процессе проектирования цифровых устройств.

Для *автомата с программируемой логикой* алгоритм работы записывается в управляющую память, в виде микропрограммы, состоящей из микрокоманд. Микрокоманда содержит информацию о микрооперациях, которые должны выполняться в данном такте работы устройства, и об адресе в управляющей памяти следующей микрокоманды, которая будет выполняться в следующем такте. Такие автоматы отличаются большой регулярностью структуры и возможностью оперативного внесения изменений в алгоритм работы проектируемого устройства.

Обычно микропрограмма работы блока управления задаётся в графической форме в виде схемы алгоритма.

Синтез автоматов по микропрограмме выполняется в той же последовательности, что и при словесном задании автомата. При этом микропрограмма содержит ту же информацию, которую содержит граф автомата, т.е. таблица переходов и выходов строится по микропрограмме. Для синтеза блока управления производится разметка микропрограммы. Правила разметки зависят от типа автомата.

Разметка микропрограммы для автомата Мура

1. Начальный и конечный операторы помечаются символом начального состояния.
2. Все безусловные операторы помечаются символами следующих состояний.

Разметка микропрограммы для автомата Мили

1. Выход начального и вход конечного операторов помечаются символом начального состояния.
2. Входы всех операторов, следующих за безусловными операторами, помечаются символами следующих состояний.

После разметки микропрограммы строится таблица переходов и выходов. Далее синтез автомата выполняется в порядке, рассмотренном выше.

4.2. Влияние типа триггера на схему автомата

В каждом такте работы автомата схема формирования следующего состояния (КС1) выдаёт сигнал Q_{t+1}, который поступает на входы элементов памяти. Сигнал Q_{t+1} записывается в память и в следующем такте появляется на выходах элементов памяти как сигнал текущего состояния Q_t. При этом предполагается, что сигнал Q_t совпадает по значению с сигналом Q_{t+1} , но они относятся к разным тактам.

Такое предположение оправдывается лишь в случае, когда в качестве элементов памяти используются D-триггеры. Действительно, состояние D-

триггера (сигнал на прямом выходе) совпадает с сигналом на его входе, поступившим в предыдущем такте.

При использовании триггеров, отличных от D-триггеров, состояние каждого триггера не всегда совпадает с сигналами на входах триггера в предыдущем такте. Это объясняется как различной логикой управления состояниями триггеров различных типов, так и различным количеством входов. Поэтому для сохранения логики переходов автомата, заданной таблицей переходов, на входы триггеров следует подать не сигнал Q_{t+1} , а некоторый другой сигнал, логика формирования которого задаётся в виде ***функции возбуждения триггеров*** $F_в$.

Функции возбуждения триггеров определяют сигналы, которые нужно подавать на входы триггеров в каждом такте, чтобы состояния автомата не зависели от типа триггеров.

Для составления функций возбуждения таблица переходов и выходов преобразуется в таблицу функций возбуждения триггеров. Таблица функций возбуждения триггеров по форме совпадает с таблицей переходов и выходов. Но в таблице функций возбуждения для каждого триггера указываются сигналы, которые следует подать на входы триггера для перевода его из текущего состояния в заданное следующее. Преобразование таблицы переходов и выходов выполняется с помощью характеристической таблицы соответствующего триггера.

4.3. Синтез автоматов с использованием Т-триггеров

Т-триггер имеет один вход (кроме входа синхронизации). Логика работы Т-триггера описывается таблицей переходов, приведённой в таблице 4.1.

Таблица 4.1.

Входы	Состояния	
T	**0**	**1**
0	0	1
1	1	0

Характеристическая таблица Т-триггера составляется по таблице переходов.

Таблица 4.2.

Состояния и выходы		Вход
Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Входами в характеристическую таблицу является текущее (Q_t) и следующее (Q_{t+1}) состояния триггера. В столбце «Вход» указываются значения сигнала T, при поступлении которого триггер переходит из состояния (Q_t) в состояние (Q_{t+1}).

Из таблицы 4.2. видно, что значение сигнала T определяется как значение логической функции

$$T = \overline{Q_t} Q_{t+1} \vee Q_t \overline{Q_{t+1}} = Q_t \oplus Q_{t+1}.$$

Таким образом, на вход T-триггера следует подавать символ «1», если нужно изменить его состояние.

4.4. Синтез автоматов с использованием RS-триггера

В том случае, если элементами памяти автомата являются RS – триггеры, при синтезе автомата необходимо учесть особенности логики работы таких триггеров. Главная особенность RS – триггера по сравнению с D – триггером заключается в том, что RS-триггер имеет два информационных входа. Информационные входы RS – триггера обозначаются буквами S (SET) и R (RESET). Вход S используется для установки триггера в состояние «1», вход R – для установки триггера в состояние «0». Обычно в таких триггерах активным является сигнал «1», т.е. для записи в триггер как символа «1», так и символа «0» на соответствующие входы триггера нужно подавать сигнал «1».

Таблица переходов синхронного RS-триггера приведена ниже.

Таблица 4.3

Входы		Состояния	
S	**R**	**0**	**1**
0	0	0	1
0	1	0	0
1	0	1	1
1	1	-	-

Как видно из таблицы переходов, комбинация сигналов S=0 и R=0 соответствует режиму хранения, при котором триггер сохраняет своё состояние. Если на вход триггера подаётся комбинация сигналов S=0 и R=1, триггер переходит в состояние «0» (или остаётся в нём). При поступлении сигналов S=1 и R=0 триггер переходит в состояние «1» (или остаётся в нём). Комбинация входных сигналов S=1 и R=1 для RS-триггера является запрещённой.

Характеристическая таблица RS-триггера составляется по таблице переходов и имеет вид таблицы 4.4.

Таблица 4.4.

Состояния		Входы	
$\mathbf{Q_t}$	$\mathbf{Q_{t+1}}$	**S**	**R**
0	0	0	-

0	1	1	0
1	0	0	1
1	1	-	0

Из характеристической таблицы можно записать логическую функцию, описывающую значение сигнала S для перевода триггера в состояние $Q_{t+1}=1$:

$S = \overline{Q_t} Q_{t+1}$.

С учётом того, что при $Q_t=Q_{t+1}=1$ значение сигнала S является неопределённым, эту функцию можно упростить следующим образом:

$S = \overline{Q_t} Q_{t+1} \to Q_t Q_{t+1} = Q_{t+1}$.

Таким образом, для установки RS-триггера в состояние 1 на вход S следует подать сигнал 1.

Аналогичным образом может быть получена логическая функция для сигнала R:

$R = \overline{Q_t}\,\overline{Q_{t+1}} \to Q_t \overline{Q_{t+1}} = \overline{Q_{t+1}}$.

4.5. Синтез автоматов с использованием JK-триггера

JK – триггер имеет два информационных входа, которые обозначаются буквами J и K. Логика работы JK – триггера во многом совпадает с логикой работы RS – триггера. При этом назначение входов J и K аналогично назначению входов S и R соответственно. Таблица переходов синхронного JK – триггера (при C=1) показаны в таблице 4.5.

Таблица 4.5

Входы		Состояния	
J	**K**	**0**	**1**
0	0	0	1
0	1	0	0
1	0	1	1
1	1	1	0

Как видно из таблицы переходов, JK – триггер отличается от RS – триггера тем, что для JK – триггера допускаются любые комбинации входных сигналов. При поступлении на входы JK – триггера сигналов J=1 и K=1 триггер изменяет своё состояние. Характеристическая таблица JK – триггера составляется по таблице переходов и имеет вид таблицы 4.6.

Таблица 4.6.

Состояния		Входы	
$\mathbf{Q_t}$	$\mathbf{Q_{t+1}}$	**J**	**K**
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Из характеристической таблицы можно записать логическую функцию, описывающую значение сигнала J для перевода триггера в состояние Q_{t+1}=1:

$$J = \overline{Q_t} Q_{t+1}.$$

С учётом того, что при Q_t=Q_{t+1}=1 значение сигнала J является неопределённым, эту функцию можно упростить следующим образом:

$$J = \overline{Q_t} Q_{t+1} \vee Q_t Q_{t+1} = Q_{t+1}.$$

Таким образом, для установки JK – триггера в состояние 1 на вход J следует подать сигнал 1.

Аналогичным образом может быть получена логическая функция для сигнала K:

$$K = \overline{Q_t}\,\overline{Q_{t+1}} \vee Q_t \overline{Q_{t+1}} = \overline{Q_{t+1}}.$$

Полученные выражения для функций J и K совпадают с аналогичными выражениями для функций S и R соответственно.

Последовательность синтеза автомата на JK-триггерах та же, что и при синтезе автомата на RS-триггерах.

4.6. Причины частичной определённости автоматов.

В некоторых случаях при задании автомата не определены все возможные переходы из отдельных состояний для конкретных входных сигналов. В таких случаях автомат называется **частично определенным**.

Признаком частичной определенности автомата является наличие незаполненных клеток в таблице переходов и выходов. На графе автомата в этом случае из некоторых вершин выходят не все дуги, соответствующие полному набору комбинаций входных сигналов.

Частичная определенность автомата может возникать по нескольким причинам. Прежде всего, она может быть обусловлена тем, что на входы автомата могут поступать не все возможные комбинации входных сигналов, а только некоторые из них. Например, если на вход автомата поступают двоично-десятичные коды чисел, то из 16-ти возможных комбинаций каждой тетрады только 10 комбинаций являются разрешенными, а остальные 6 – запрещенными. При этом для запрещенных комбинаций переходы автомата из одного состояния в другое не определены. Примеры двоично-десятичных кодов приведены в таблице 4.7.

Таблица 4.7.

Десятичные цифры	Веса разрядов в тетрадах	
	8-4-2-1	2-4-2-1
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100

7	0111	1101
8	1000	1110
9	1001	1111
Запрещённые комбинации		
	1010	0101
	1011	0110
	1100	0111
	1101	1000
	1110	1001
	1111	1010

В таблице 4.7 код с весами разрядов 8-4-2-1 является двоично-десятичным кодом с естественным кодированием десятичных цифр. В этом коде веса разрядов те же, что и при записи двоичных чисел. Код с весами разрядов 2-4-2-1 является одним из специальных кодов. Такие коды используются при выполнении арифметических операций над числами, записанными в двоично-десятичном коде.

Второй причиной частичной определенности автомата является использование в качестве элементов памяти RS- или JK-триггеров. Характеристические таблицы этих триггеров имеют вид таблиц 4.8 и 4.9.

Таблица 4.8

Состояния		Входы	
Q_t	Q_{t+1}	S	R
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0

Таблица 4.9

Состояния		Входы	
Q_t	Q_{t+1}	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

В этих таблицах в соответствии с логикой работы триггеров значения входных сигналов определены не для всех возможных переходов. Поэтому при формировании таблицы функций возбуждения эта неопределенность будет внесена и в эту таблицу, а автомат окажется определенным лишь частично.

Следующей причиной частичной определенности автомата является неполное использование возможностей элементов памяти для кодирования состояний автомата. Напомним, что состояние автомата – это информация,

записанная в элементах памяти автомата. Поэтому максимальное количество состояний N_{max} , которое может иметь автомат при фиксированном числе элементов памяти, определяется соотношением

$$N_{max}=2^n,$$

где n – число элементов памяти автомата.

Таким образом, максимальное количество состояний может принимать значения, равные 2, 4, 8, 16, 32 и т.д. Если же количество состояний некоторого автомата не равно целой степени числа 2, то при кодировании состояний автомата используются не все возможные комбинации состояний элементов памяти. Например, при количестве состояний автомата, равном 13, необходимо иметь 4 элемента памяти. В этом случае максимальное количество состояний автомата равно16, из которых используются лишь 13, а 3 состояния являются запрещенными.

4.7. Использование частичной определенности автомата для упрощения его схемы

Частичная определенность автомата может быть использована для построения более простых схем. Так как количество элементов памяти определяется количеством его состояний, то упрощение схемы автомата может быть проведено лишь путем упрощения его комбинационной части. Поэтому сущность использования частичной определенности автомата путем его доопределения не отличается от доопределения при синтезе комбинационных схем.

Рассмотрим пример

Задание. Синтезировать счетчик по модулю 3 с порядком счета 2-0-1. Счетчик синтезировать как автомат Мура с использованием логических элементов И-НЕ и JK-триггеров.

Формализация задания. На вход счетчика поступает последовательность символов 0 и 1. Счетчик должен подсчитывать количество поступивших на его вход символов 1. На выходе счетчика должно фиксироваться количество символов 1, подсчитанных по модулю 3. В исходном состоянии счетчик должен показывать число2. При двоичном кодировании выходных сигналов счетчик должен иметь два выхода. Тогда обобщенная схема счетчика может иметь вид, показанный на рис.4.1.

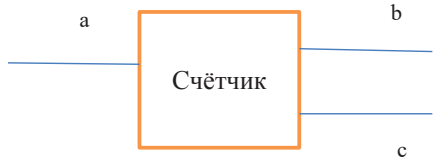

Рис. 4.1. Обобщённая схема счётчика
$a, b, c \in (0,1)$

Выбор типа автомата. Задан автомат Мура.

Составление графа состояний автомата
Граф состояний счётчика представлен на рис. 4.2.

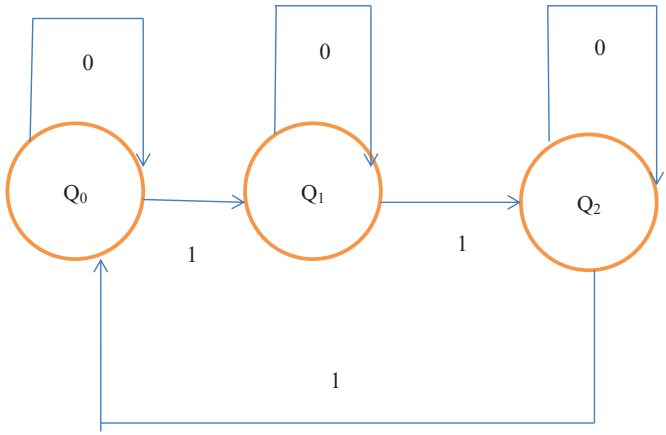

Рис. 4.2. Граф состояний счётчика

Составление таблицы переходов и выходов. Таблица переходов и выходов приведена в таблице 4.10.

Таблица 4.10

Вход	Состояния и выходы		
	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
a	Q_0	Q_1	Q_2
0	Q_0	Q_1	Q_2
1	Q_1	Q_2	Q_0

Кодирование состояний. Для трех состояний необходимо 2 элемента памяти. При естественном кодировании состояний получим:

$$Q_0 \to \bar{q}^1\bar{q}^2 \to 00;$$
$$Q_1 \to \bar{q}^1 q^2 \to 01;$$
$$Q_2 \to q^1\bar{q}^2 \to 10.$$

Составление кодированной таблицы переходов и выходов. Кодированная таблица переходов и выходов приведена в таблице 4.11.

Таблица 4.11.

Вход	Состояния и выходы		
	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
	Q_0	Q_1	Q_2
a	$\overline{q^1}\,\overline{q^2}$	$\overline{q^1}q^2$	$q^1\overline{q^2}$
	00	01	10
0	00	01	10
1	01	10	00

Выбор типа элементов памяти. Заданы JK-триггеры.

Запись функций возбуждения и функций выходов в СДНФ.

Функции возбуждения триггеров:

$J_1 = a\overline{q^1}q^2$;

$K_1 = aq^1\overline{q^2}$;

$J_2 = a\overline{q^1}\,\overline{q^2}$;

$K_2 = a\overline{q^1}q^2$.

Функции выходов:

$b = \overline{q^1}\,\overline{q^2}$;

$c = q^1\overline{q^2}$.

Минимизация функций возбуждения и функций выходов.

Функции возбуждения триггеров после минимизации:

$J_1 = aq^2$;

$K_1 = a$;

$J_2 = a\overline{q^1}$;

$K_2 = a$.

Функции выходов после минимизации:

$b = \overline{q^1}\,\overline{q^2}$;

$c = q^1$.

Преобразование функций возбуждения и выходов.

Применяя правило двойной инверсии, получим:

$J_1 = \overline{\overline{aq^2}}$;

$K_1 = a$;

$J_2 = \overline{\overline{aq^1}}$;

$K_2 = a$.

$b = \overline{\overline{q^1}\,\overline{q^2}}$;

$c = q^1$.

Составление функциональной схемы автомата.

Функциональная схема автомата приведена на рис.4.3.

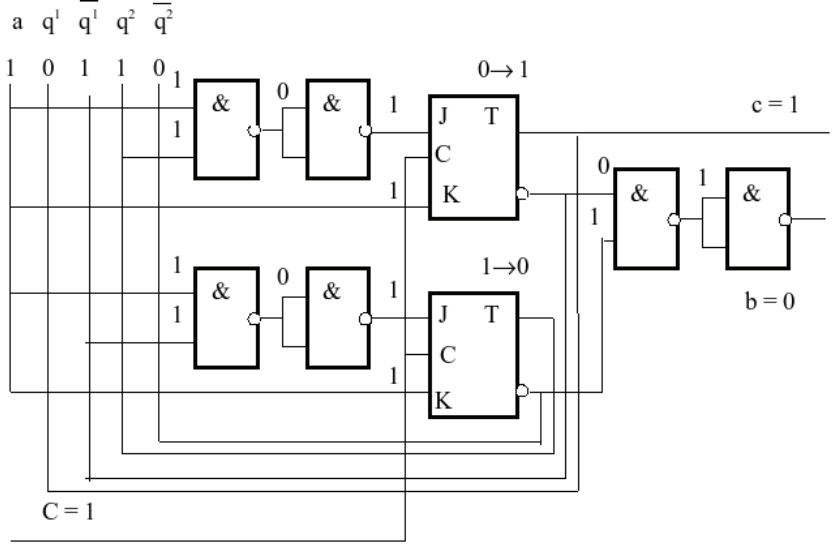

Рис. 4.3. Функциональная схема автомата

На рис.4.3 показаны также значения сигналов на выходах элементов для случая, когда автомат находится в состоянии Q_1 ($q^1 = 0$ и $q^2 = 1$) и на его вход поступает сигнал a = 1. При этом автомат переходит в состояние Q_2 ($q^1 = 1$ и $q^2 = 0$) и выдает сигналы b = 0 и c = 1, что соответствует таблице переходов.

Контрольные вопросы

1. Для какого типа триггера состояние триггера совпадает с входным сигналом в предыдущем такте?
2. Как таблицу переходов и выходов автомата преобразовать в таблицу функций возбуждения триггеров?
3. Какую информацию можно получить из характеристической таблицы триггера?
4. Запишите логическое выражение для функции возбуждения Т-триггера.
5. Что является причиной неполной определённости автоматов?
6. Какие триггеры могут быть причиной неполной определённости

автоматов?

7. Как определить по графу состояний, что автомат определён частично?
8. Как определить по таблице переходов, что автомат определён частично?

5. Синтез блоков управления

5.1. Особенности блоков управления с распределителями импульсов

Блоки управления представляют собой особый тип автоматов с памятью. Ранее было показано, что блоки управления непосредственно не выполняют операций по преобразованию данных. Они только обеспечивают выполнение преобразований в операционных блоках путем формирования управляющих сигналов. Управляющие сигналы, вырабатываемые блоками управления, организуют передачу данных внутри операционного блока. Для этого управляющие сигналы коммутируют в определенные моменты и на определенный промежуток времени цепи передачи данных внутри операционного блока.

В общем случае блок управления может иметь несколько режимов работы. Такими режимами могут быть, например, режимы выполнения различных операций (сложение, вычитание, умножение и т. д.) в блоках управления арифметико-логическими устройствами (АЛУ), режимы записи и чтения данных в блоках управления памятью, режимы приема и передачи данных в блоках управления интерфейсами и т. д. Режим работы блока управления задается в виде кода операции, который принимает определенные значения для каждого режима.

Понятно, что набор и последовательность формирования управляющих сигналов зависит от заданного режима работы блока управления. Кроме того, последовательность управляющих сигналов обычно не является жесткой и заранее заданной, а зависит и от различного вида условий, которые проверяются при работе блока управления. Результат проверки условия зависит от значения исходных данных и промежуточных результатов, обрабатываемых в операционном блоке, поэтому его нельзя предсказать заранее. Результаты проверки условий поступают на блок управления в виде сигналов, называемых признаками. Для блока управления АЛУ, например, такими признаками могут быть знаки чисел, признак равенства результата нулю, признак переполнения разрядной сетки, значение очередного разряда множителя при умножении и т. д.

Физически управляющие сигналы обычно представляют собой импульсы напряжения постоянного тока. Источником таких импульсов является генератор синхронизирующих импульсов (синхроимпульсов), который вырабатывает непрерывную последовательность импульсов, обычно прямоугольной формы.

Блок управления можно рассматривать как преобразователь синхроимпульсов (СИ) в последовательность управляющих сигналов (УС), как это показано на рис.5.1. Кроме синхроимпульсов на блок управления поступают также код операции (КОп) и признаки (П).

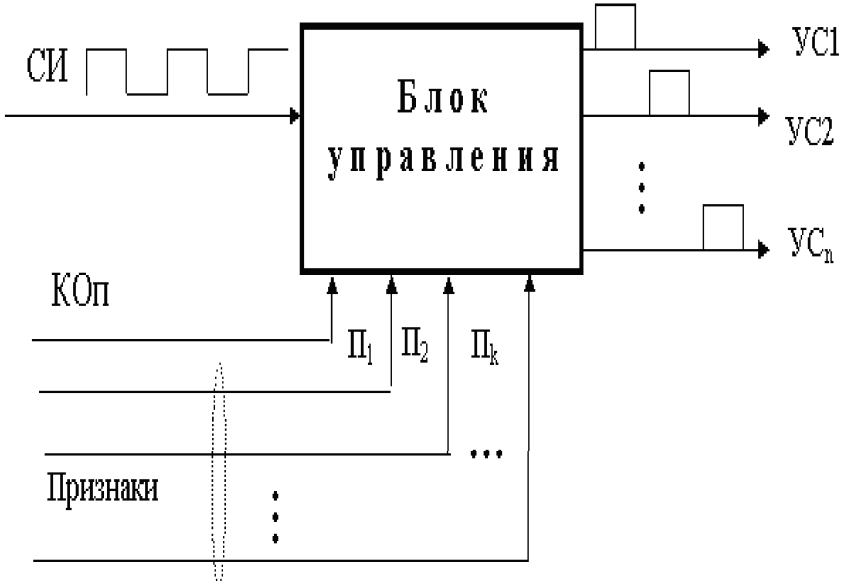

Рис. 5.1. Преобразование блоком управления синхроимпульсов в последовательность управляющих импульсов.

Блок управления можно синтезировать как автомат с памятью по общей методике. В этом случае алгоритм формирования управляющих сигналов жёстко связан со структурой автомата, т.е. с набором элементов и связями между ними. Если необходимо внести даже незначительные изменения в логику работы блока управления, то приходится изменять всю схему, т.е. заново синтезировать автомат. Для обеспечения возможности более простого изменения логики работы блоков управления иногда используют специальные структуры таких блоков. Вариант такой структуры представлен на рис. 5.2.

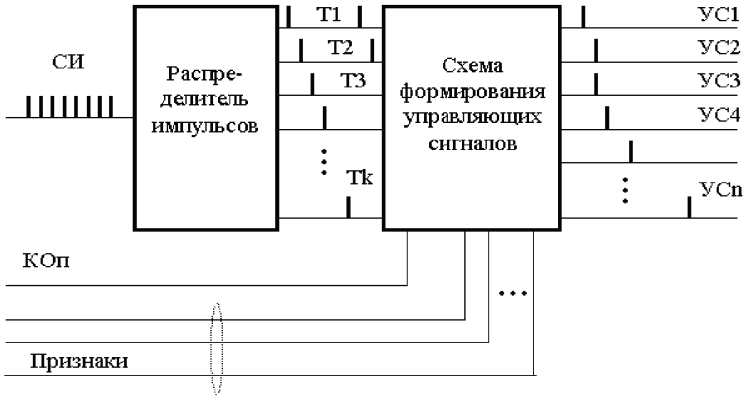

Рис. 5.2. Специальная структура блока управления.

В состав блока управления входят распределитель импульсов и схема формирования управляющих сигналов.

Распределитель импульсов предназначен для распределения синхроимпульсов в пространстве и времени. Из непрерывной последовательности синхроимпульсов распределитель формирует периодически повторяющуюся последовательность тактовых сигналов T_i. Каждый тактовый импульс выдается на отдельный выход распределителя и делит процесс работы блока управления на *такты*. Количество тактовых импульсов и последовательность их формирования являются фиксированными и не зависят ни от режима работы, ни от значения признаков.

Особенностью схемы на рис.5.2 является то, что распределитель импульсов представляет собой типовую схему. Поэтому синтез распределителя импульсов сводится к простому выбору схемы с заданным количеством тактовых сигналов. Кроме того, при изменении логики работы блока управления обычно количество тактовых сигналов не изменяется, поэтому распределитель импульсов остается без изменений.

Существует несколько вариантов схем распределителей импульсов. На рис.5.3 показан вариант распределителя импульсов, построенного на основе двухтактных RS-триггеров.

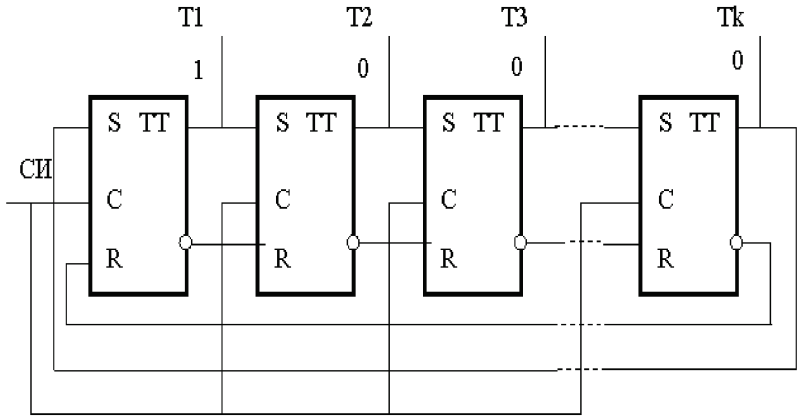

Рис. 5.3. Распределитель импульсов, построенный на основе двухтактных RS - триггеров

Этот распределитель импульсов представляет собой кольцевой сдвигающий регистр, в первый (левый) разряд которого записан символ 1. При поступлении каждого синхроимпульса происходит сдвиг информации в регистре на один разряд вправо. С выхода последнего триггера символ 1 заносится в первый триггер. Таким образом, в каждом такте на выходе одного из триггеров возникает сигнал 1, т. е. тактовый импульс. Количество тактовых импульсов равно разрядности регистра, т. е. количеству триггеров.

В течение одного такта блок управления формирует один или несколько управляющих сигналов. Эти сигналы выдаются схемой формирования управляющих сигналов в зависимости от номера такта, режима работы (кода операции) и значения признаков. Схема формирования управляющих сигналов представляет собой комбинационную схему с несколькими выходами. Её можно рассматривать как совокупность отдельных схем, каждая из которых формирует один управляющий сигнал. Для синтеза этих схем необходимо знать условия, при которых в данном такте формируется определенный управляющий сигнал.

Блоки управления с распределителями импульсов позволяют достаточно просто вносить изменения в логику формирования управляющих сигналов. Для этого обычно достаточно изменить одну или несколько схем формирования отдельных управляющих сигналов, не затрагивая остальных схем.

Следует отметить, что блок управления с распределителем импульсов можно рассматривать как автомат с памятью, у которого отсутствует комбинационная схем формирования нового состояния (КС1). В этой схеме каждому состоянию автомата соответствует один элемент памяти, т. е. принято унарное кодирование состояний. Такой способ кодирования по

сравнению с двоичным кодированием приводит к увеличению числа элементов памяти.

5.2. Последовательность синтеза блоков управления с распределителями импульсов.

Алгоритм работы блоков управления с распределителями импульсов обычно задается при помощи микропрограмм.

Синтез блоков управления с распределителями импульсов выполняется в указанном ниже порядке:

1. На основе анализа микропрограммы определяется максимальное количество тактов, необходимое для реализации самой длинной ветви микропрограммы.
2. Подбирается распределитель импульсов с необходимым количеством тактовых сигналов.
3. Для каждого управляющего сигнала ставят условия его формирования (номер тактового сигнала и значения признаков). Эти условия записываются в виде логической функции.
4. Производится выбор логических элементов.
5. Выполняется преобразование логической функции для реализации её на выбранной системе элементов.
6. Составляется комбинационная схема для формирования каждого управляющего сигнала.
7. Полученные схемы объединяются в общую схему формирования управляющих сигналов.
8. Схемы распределителя импульсов и формирования управляющих сигналов объединяются в общую схему блока управления.
9. Проводится проверка работоспособности схемы.

Контрольные вопросы

1. От чего зависит последовательность сигналов на выходе блока управления?
2. Что такое распределитель импульсов?
3. Почему последовательность управляющих сигналов не всегда можно задать заранее?
4. К какому типу автоматов с памятью можно отнести блок управления с распределителем импульсов?
5. Как построить временную диаграмму работы блока управления?
6. Что такое код операции?
7. Какие сигналы являются входными для схемы формирования управляющих сигналов?

6. Гонки в автоматах

6.1. Сущность эффекта гонок

Ранее при рассмотрении процесса работы автоматов с памятью принималось, что работа автомата происходит в дискретные моменты времени. При этом считалось, что переходы автомата из одного состояния в другое происходят мгновенно. Это означало, что при изменении входных сигналов соответственно мгновенно изменяются и сигналы на выходе схемы.

Кодирование заключается в сопоставлении каждому состоянию автомата набора (кода) состояний элементов памяти. При этом наборы для всех состояний должны иметь одинаковую длину, а разным состояниям автомата должны соответствовать разные наборы.

Переход автомата из одного состояния в другое осуществляется за счет изменения состояний элементов памяти. Если автомат переходит из состояния с кодом 010 в состояние с кодом 100, то это означает, что триггер V_1 переходит из состояния 0 в состояние 1, V_2 – из 1 в 0, V_3 – сохраняет свое состояние.

На практике реальные логические элементы и элементы памяти при изменении входных сигналов изменяют выходные сигналы с некоторой временной задержкой. При функционировании автомата могут появиться так называемые состязания. Это явление возникает вследствие того, что элементы памяти имеют различные, хотя и достаточно близкие, времена срабатывания. Различны также задержки сигналов возбуждения, поступающих на входные каналы элементарных автоматов по логическим цепям неодинаковой длины.

Если при переходе автомата из одного состояния в другое должны изменить свои состояния сразу несколько запоминающих элементов, то между ними начинаются состязания. Тот элемент, который выиграет эти состязания, т.е. изменит свое состояние ранее, чем другие элементы, может через цепь обратной связи изменить сигналы на входах некоторых запоминающих элементов до того, как другие, участвующие в состязаниях элементы, изменят свои состояния. Это может привести к переходу автомата в состояние, не предусмотренное его графом.

Если затем при том же входном сигнале автомат перейдёт в нужное состояние, то такие состязания являются допустимыми или **некритическими.**

Если же в этом автомате есть переход под действием того же сигнала в другое состояние, то правильность его работы будет нарушена. Такие состязания называются **критическими состязаниями** или **гонками** и необходимо принимать меры для их устранения.

Важно заметить, что время задержки сигнала элементами зависит от типа элемента. Кроме того, даже для однотипных элементов время задержки всегда различно, так как различны параметры радиодеталей, из которых составлены элементы. Даже для одного и того же элемента время задержки

может меняться при изменении условий работы, например, при изменении температуры, напряжения питания и т.д.

Помимо этого, при работе автомата входные сигналы элементов памяти проходят через комбинационные схемы, составленные из различного количества элементов, т.е. с различной задержкой.

6.2. Методы борьбы с гонками

Устранить гонки можно аппаратными средствами либо, используя специальные методы кодирования. Один из способов ликвидации гонок состоит ***в тактировании входных сигналов автомата импульсами определенной длительности.*** Предполагается, что кроме входных каналов x_1, ..., x_l имеется еще канал C от генератора синхроимпульсов, по которому поступает сигнал $C = 1$ в момент прихода импульса и $C = 0$ при его отсутствии. В связи с этим входным сигналом на переходе будет не предыдущий сигнал, а умноженный на C. Тогда, если длительность импульса t_c меньше самого короткого пути прохождения тактированного сигнала обратной связи по комбинационной схеме, то к моменту перехода в промежуточное состояние сигнал $C = 0$, $CZ_f=0$, где Z_f – входной сигнал, что исключает гонки. Канал C – это фактически синхровход триггера. Недостаток метода – в трудности подбора требуемой длительности импульса, т.к. она зависит от многих факторов, не поддающихся строгому учету.

6.2.1. Противогоночное кодирование

Рассмотрим пример.

В автомате, граф состояний которого приведён на рис. 6.1. гонки могут возникать при переходе автомата из состояния Q_1 в состояние Q_2 и при переходе из состояния Q_3 в состояние Q_0.

Действительно, при естественном кодировании состояний последовательность переходов, если входной сигнал равен 1, выглядит следующим образом:

$$\begin{array}{cc} & Q_0\ Q_1\ Q_2\ Q_3\ Q_0\ Q_1 \\ q^1 & 0\to0\to1\to1\to0\to0 \\ q^2 & 0\to1\to0\to1\to0\to1 \text{ и т.д.} \end{array}$$

Идея противогоночного кодирования заключается в том, что состояния автомата кодируются так, чтобы при любом переходе автомата из одного состояния в другое переключался только один элемент памяти.

Такое кодирование называется соседним кодированием состояний.

Например, для графа на рис. 6.1. можно отказаться от естественного кодирования и закодировать состояния следующим образом: $Q_0 \to 00$, $Q_1 \to 01$, $Q_2 \to 11$, $Q_3 \to 10$.

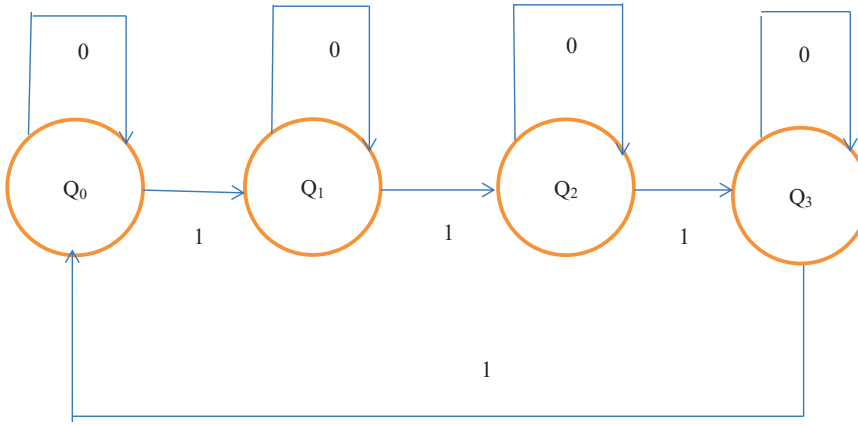

Рис. 6.1. Граф состояний автомата, в котором могут возникать гонки

Тогда последовательность переходов при входном сигнале q, равном 1, будет иметь следующий вид:

$$\begin{array}{lcccccc} & Q_0 & Q_1 & Q_2 & Q_3 & Q_0 & Q_1 \\ q^1 & 0 & \to 0 \to & 1 & \to 1 \to & 0 & \to 0 \\ q^2 & 0 & \to 1 \to & 1 & \to 0 \to & 0 & \to 1 \end{array}$$ и т.д.

Гонки в этом случае не могут возникать.

Граф автомата, допускающий соседнее кодирование, должен удовлетворять ряду требований, а именно:

1) в графе автомата не должно быть циклов с нечетным числом вершин;

2) два соседних состояния второго порядка не должны иметь более двух состояний, лежащих между ними.

На рис. 6.2 приведены графы автоматов, допускающие и не допускающие соседнее кодирование.

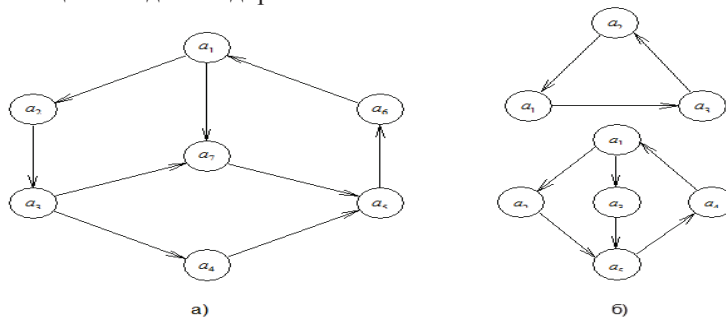

Рис. 6.2. Графы автоматов:

(а) – допускающие соседнее кодирование;

(b) - не допускающие соседнее кодирование

В настоящее время существуют алгоритмы противогоночного кодирования. Эти алгоритмы связаны с перебором всех возможных переходов автомата и являются довольно громоздкими. Кроме того, реализация таких алгоритмов может потребовать введения дополнительных состояний специально для устранения гонок.

6.2.2. Синхронизация работы автомата

Одной из причин гонок в автоматах является наличие обратной связи с выходов элементов памяти на вход схемы формирования состояний автомата. Именно по этой цепи изменение состояния одного элемента памяти передаётся в конечном счёте на входы как других, так и этого же элемента, и может нарушить логику переходов автомата.

Идея синхронизации работы автомата заключается в том, что цепь обратной связи выполняется управляемой, т.е. при помощи специальных синхронизирующих сигналов замыкается лишь в определённый момент и на определённый промежуток времени. Пусть, например, автомат имеет два элемента памяти, на входах которых установлены вентили или ключи так, как это показано на рис. 6.3.

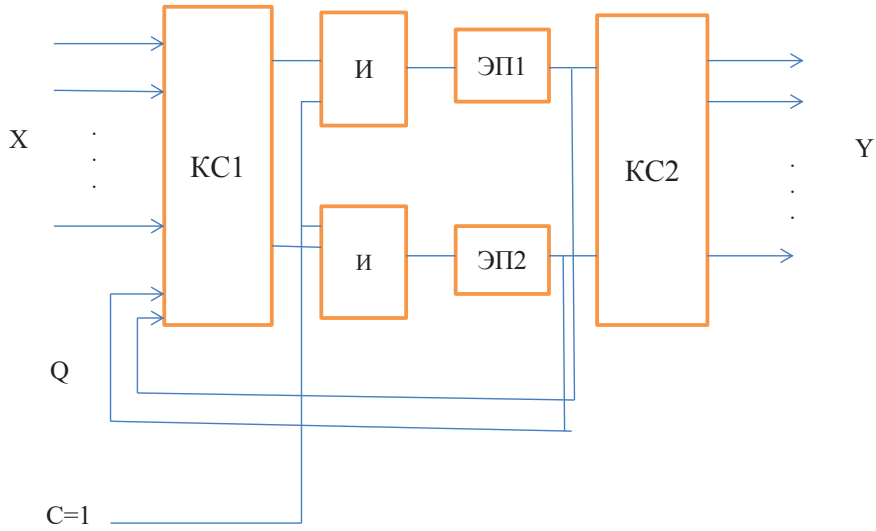

Рис. 6.3. Структурная схема автомата

В качестве вентилей на рис. 6.3. используются элементы И на два входа. На один из входов поступает сигнал нового состояния элемента памяти, на второй – синхронизирующий сигнал С. При С=1 сигнал со входа

вентиля проходит на его выход, т.е. цепь обратной связи замкнута и тем самым разрешается переход элементов памяти в новое состояние.

Если элементами памяти являются синхронные триггеры, то управление цепью обратной связи осуществляется при помощи сигнала, подаваемого на вход синхронизации триггера. В этом случае надобность в использовании вентилей отпадает.

Недостатком данного метода являются жёсткие требования к длительности синхронизирующего импульса. Эта длительность должна быть достаточной для того, чтобы за время действия импульса триггер успел переключиться в новое состояние. Но, с другой стороны, длительность импульса должна быть такой, чтобы за время его действия сигнал с выхода триггера не успел пройти по всей цепи обратной связи и поступить на входы этого и других триггеров.

6.2.3. Использование двухтактных триггеров

Наличие цепи обратной связи приводит к тому, что при формировании нового состояния каждый элемент памяти должен одновременно выполнять две взаимоисключающие функции. Во-первых, на время срабатывания схемы формирования нового состояния элемент памяти должен сохранять старое состояние. Во-вторых, практически в то же самое время этот элемент памяти должен принять новое состояние.

Для решения этой проблемы вводится двойная память. В этом случае каждый элемент памяти дублируется, причем перепись из первого элемента памяти во второй происходит в момент $C = 0$. Пример элемента с двойной памятью приведён на рис.6.4.

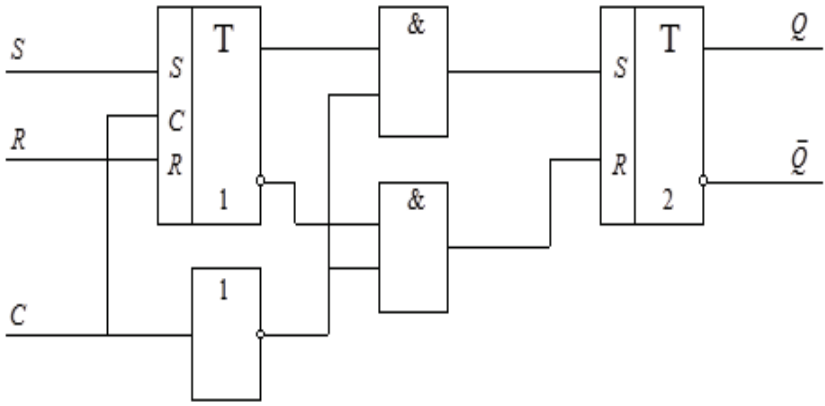

Рис. 6.4. Элемент с двойной памятью

В режиме хранения оба триггера находятся в одинаковом состоянии.

Процесс изменения состояний триггеров первой (1) и второй (2) ступеней триггера показан на временных диаграммах рис. 6.5. На временных диаграммах состояние триггеров первой и второй ступеней обозначены символами Q_1 и Q_2 соответственно.

Если обе ступени триггера находятся в состоянии 0 и на входы поступают сигналы S=1, C=1 и S=0, то переходит в состояние 1 первая ступень (Q_1), а затем, при изменении сигнала C из 1 в 0, вторая ступень (Q_2).Таким образом, в течение действия импульса C первая ступень триггера находится в новом состоянии, а вторая ступень сохраняет старое состояние.

Аналогичным образом происходит переключение триггера из состояния 1 в состояние 0.

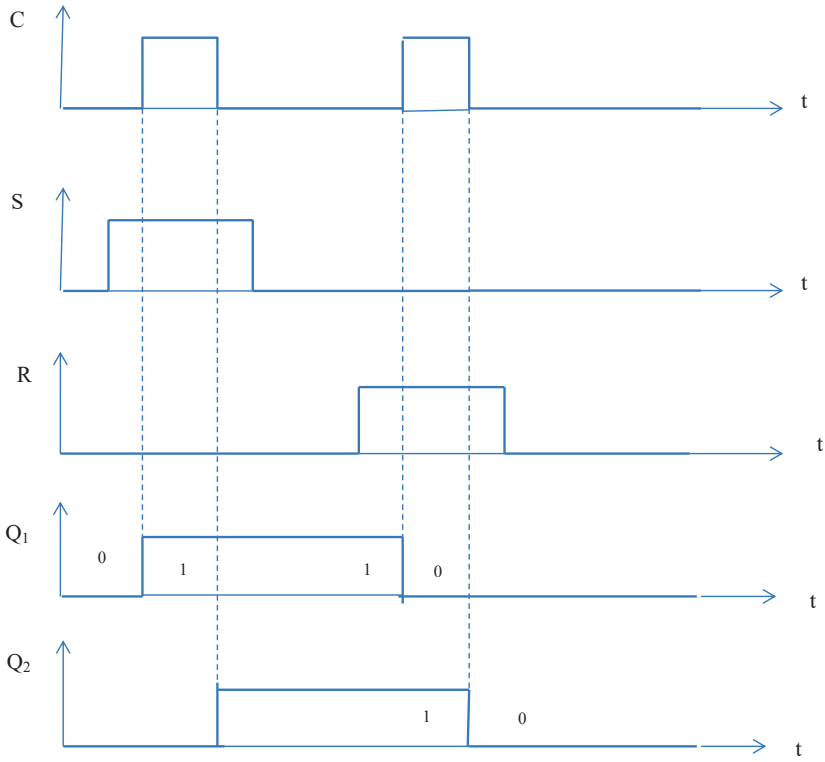

Рис. 6.5. Временные диаграммы процесса изменения состояний триггеров первой и второй ступеней

Использование двухтактных триггеров полностью исключает эффект гонок, так как функция хранения старого состояния и приёма нового

состояния выполняются по существу различными триггерами (триггерами первой и второй ступеней). Недостатком данного способа является увеличение числа триггеров вдвое.

Контрольные вопросы

1. Что является причиной гонок в автоматах?
2. В чём проявляется эффект гонок?
3. При каких условиях могут возникать гонки?
4. В чём заключается идея противогоночного кодирования?
5. Какие недостатки имеет метод противогоночного кодирования?
6. В чём заключается суть синхронизации работы автомата?
7. Какие недостатки имеет метод синхронизации работы автомата?
8. Как переключаются триггеры первой и второй ступеней двухтактного триггера?
9. Какие недостатки имеет использование двухтактных триггеров?
10. Почему использование двухтактных триггеров исключает эффект гонок?

7. Автоматы с программным формированием выходных сигналов

7.1. Способы реализации алгоритмов

Алгоритмы преобразования информации могут быть реализованы аппаратным или программным способами.

Аппаратный способ заключается в том, что для реализации каждого алгоритма создается специальный (специализированный) автомат, Этот автомат может быть построен по структуре автомата Мура или Мили, а также в виде автомата с распределителем импульсов. В этом случае структура автомата жестко связана с реализуемым алгоритмом и при любом изменении алгоритма требуется изменение схемы автомата. Такие автоматы называют автоматами **«с жесткой логикой».** В них набор и последовательность выходных сигналов имеют вид «зашиты» в схеме автомата..

Программный способ заключается в том, что автомат имеет типовую структуру независимо от реализуемого алгоритма, т. е. является универсальным. Такие автоматы называют автоматами **с хранимой логикой** или автоматами с программным формированием выходных сигналов. Настройка автомата на реализацию заданного алгоритма производится с помощью микропрограммы, содержащей информацию о выходных сигналах, необходимых для выполнения этого алгоритма. Микропрограмма записывается в специальной памяти. В каждом такте реализации алгоритма из памяти выбирается микрокоманда и в соответствии с ее содержанием формируются выходные сигналы данного такта. В таких автоматах набор и последовательность выходных сигналов определяется информацией, записанной в памяти.

7.2. Структура автомата с программным формированием выходных сигналов

Типовая структура автомата с программным формированием выходных сигналов представлена на рис.7.1. В состав автомата входят:

1) память микрокоманд (ПМК);
2) регистр адреса (РА);
3) регистр микрокоманд (РМК);
4) мультиплексор (МП)

Память микрокоманд состоит из отдельных ячеек. Каждая ячейка имеет адрес (номер). В ячейке записывается одна микрокоманда. Структура микрокоманды может быть различной. На рис. 7.1. показан один из простых вариантов структуры микрокоманды, которая включает три поля:

1) поле выходных сигналов (С);
2) поле условий (признаков) (У);
3) поле адреса следующей микрокоманды (А).

Поле выходных сигналов содержит информацию о выходных сигналах, которые автомат должен выдать в данном такте. В простейшем случае в этом поле может быть записан перечень этих сигналов. При этом каждому выходному сигналу соответствует один разряд поля С. Если в данном такте определенный сигнал должен быть выдан, то значение разряда устанавливается в 1.

Поле условий используется для организации ветвлений при реализации алгоритма. В этом поле указываются условия, выполнение которых должно быть проверено в данном такте. Например, в разряде поля записывается 1, если соответствующее условие нужно проверить.

Поле адреса следующей микрокоманды используется для указания адреса очередной микрокоманды, как при реализации линейных участков алгоритма, так и при его ветвлении.

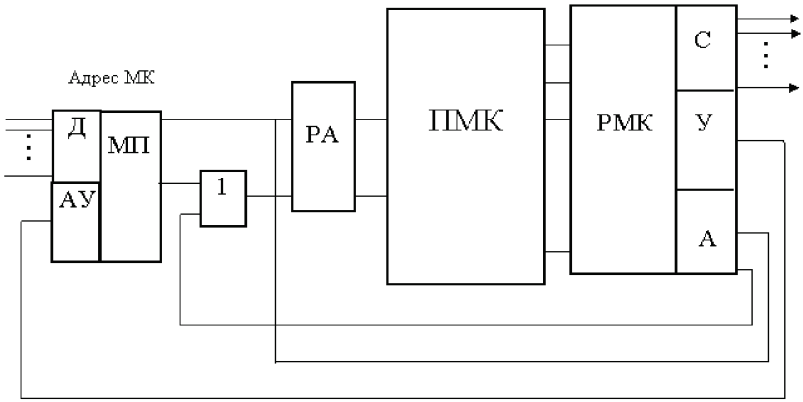

Рис. 7.1. Типовая структура автомата с программным формированием выходных сигналов

Работа автомата происходит следующим образом. Перед началом работы в памяти микрокоманд записаны микропрограммы. Для каждого алгоритма записывается отдельная микропрограмма.

Для выполнения определенного алгоритма адрес первой микрокоманды соответствующей микропрограммы поступает на регистр адреса. Далее адрес поступает в память микрокоманд и дешифрируется. Память работает в режиме чтения и микрокоманда из ячейки с заданным адресом поступает на регистр микрокоманд. В зависимости от содержания поля С формируются выходные сигналы. Одновременно адрес следующей микрокоманды из поля А регистра микрокоманд передается на регистр адреса и начинается цикл выполнения новой микрокоманды. Так продолжается до тех пор, пока в алгоритме не встретится условный оператор. В этом случае адрес очередной микрокоманды не может быть определен и записан заранее.

Организация ветвления. Для реализации условных переходов используются различные способы. Рассмотрим один из них. Так как адрес следующей микрокоманды указывается в поле А текущей микрокоманды, то для его возможного изменения в зависимости от текущего значения некоторого признака достаточно зафиксировать все разряды адреса кроме младшего. Значение же младшего разряда адреса должно зависеть от значения признака, который анализируется в текущей микрокоманде.

При выполнении линейных участков алгоритма во всех разрядах поля условий У записаны нули, и адрес из поля А полностью передается на регистр адреса. Если в одном из разрядов поля У содержится 1, то это означает, что адрес очередной команды зависит от некоторого признака, т. е. необходимо проверить определенное условие. Проверка выполнения условий выполняется с помощью специальных схем, каждая из которых выдаёт сигнал 1, если условие выполняется, и сигнал 0 – в противном случае. Эти сигналы поступают на входы Д мультиплексора МП (рис. 7.1).

Мультиплексор имеет несколько входов и один выход. Входы делятся на две группы. На группу входов Д подаются информационные сигналы. Группу входов АУ образуют адресные входы. При каждой комбинации сигналов на адресных входах сигнал с одного из информационных входов передается на выход мультиплексора. При нулевом адресе на выходе мультиплексора формируется сигнал 0. Поэтому на линейных участках микропрограммы младший разряд адреса следующей микропрограммы из поля А поступает через элемент ИЛИ в регистр адреса без изменения.

Если в текущей микрокоманде поле У содержит символ 1 в одном из разрядов, то в младшем разряде поля А записывается символ 0. Адрес проверяемого условия из поля У поступает на входы АУ мультиплексора, а на его информационные входы – значения признаков. Поэтому результат проверки выполнения условия (сигнал 0 или 1) проходит через мультиплексор и далее через элемент ИЛИ в младший разряд регистра адреса. Таким образом, значение младшего разряда адреса следующей микрокоманды будет равно 0 или 1 в зависимости от выполнения условия.

7.3. Синтез автоматов с программным формированием выходных сигналов

Структура автомата с программируемой логикой не зависит от алгоритма, реализуемого автоматом. Поэтому синтез такого автомата сводится к преобразованию формы заданного алгоритма в форму микропрограммы, которая может быть записана в памяти микрокоманд.

Для случая, когда логика работы автомата задана в виде схемы алгоритма, детализированной до микроопераций, последовательность синтеза может быть следующей:

1) определение общего формата микрокоманды (количество полей и их назначение);
2) определение разрядности микрокоманды (разрядность полей);
3) разметка схемы алгоритма (определение адресов микрокоманд);
4) заполнение полей микрокоманд (составление микропрограммы).
5) уточнение типовой структуры автомата (выбор регистров и мультиплексора с необходимым количеством входов, а также памяти необходимой емкости и разрядности).

В качестве примера рассмотрим составление микропрограммы блока управления арифметико-логического устройства (АЛУ), выполняющего операцию вычитания чисел с фиксированной точкой. В исходном состоянии числа хранятся в памяти в прямом коде. Если АЛУ не занято выполнением очередной операции, то блок управления находится в исходном состоянии и выдает сигнал готовности. Для начала операции на блок управления поступает сигнал старта (**к**), после чего числа из памяти последовательно считываются из памяти в регистры АЛУ.

При вычитании используется дополнительный код. В зависимости от знаков (a и b) числа подаются в сумматор в прямом или дополнительном коде. После определения разности в АЛУ определяются признаки результата. Если не произошло переполнения (признак o=0), то в зависимости от знака разности (s) производится преобразование ее в прямой код и запись в память.

Схема алгоритма вычитания представлена на рис.7.2, на котором приняты следующие обозначения:

k – сигнал начала операции;
a – знак числа A;
b – знак числа B;
o – признак переполнения разрядной сетки;
s – знак результата операции.

Число выходных сигналов блока управления равно числу микроопераций в микропрограмме. При анализе микропрограммы можно установить, что безусловные операторы **2, 3, 11, 14 и 15** содержат по одной микрооперации, а операторы **7, 8, 9 и 10** - по две совместимых микрооперации. Однако операторы **7, 8, 9 и 10** содержат повторяющиеся микрооперации (различных микроопераций в них всего 4). Кроме того, начальному оператору соответствует выходной сигнал, который сообщает о

готовности блока управления к выполнению операции. Таким образом, общее число выходных сигналов равно 10. Обозначения выходных сигналов и соответствующие им микрооперации приведены в таблице 7.1.

Таблица 7.1.

№ п/п	Выходные сигналы	Микрооперации
0	Y_0	Сигнал готовности
1	Y_1	Приём числа А из ОП в регистр РгА
2	Y_2	Приём числа В из ОП в регистр РгВ
3	Y_3	Выдача числа А в сумматор в прямом коде
4	Y_4	Выдача числа А в сумматор в дополнительном коде
5	Y_5	Выдача числа В в сумматор в прямом коде
6	Y_6	Выдача числа В в сумматор в дополнительном коде
7	Y_7	Формирование признака результатов
8	Y_8	Преобразование результата в прямой код
9	Y_9	Выдача результата из регистра сумматора в ОП

С учётом числа входных и выходных сигналов общая схема блока управления может быть представлена в виде рис. 7.3.

На рис. 7.4. показана схема алгоритма с указанием всех входных и выходных сигналов (Y_0… Y_9).

Общий формат микрокоманды принимаем в соответствии с рис. 7.1.

Разрядность микрокоманды определяется разрядностью полей.

Разрядность поля управляющих сигналов зависит от количества сигналов и способа кодирования. Если для каждого сигнала выделить отдельный разряд (горизонтальное кодирование), то поле выходных сигналов должно иметь 10 разрядов.

Разрядность поля условий (входных сигналов) зависит от количества условий и их кодирования. При горизонтальном кодировании поле условий должно включать 5 разрядов. Разрядность поля адреса зависит от ёмкости

памяти. В свою очередь ёмкость памяти определяется количеством микрокоманд.

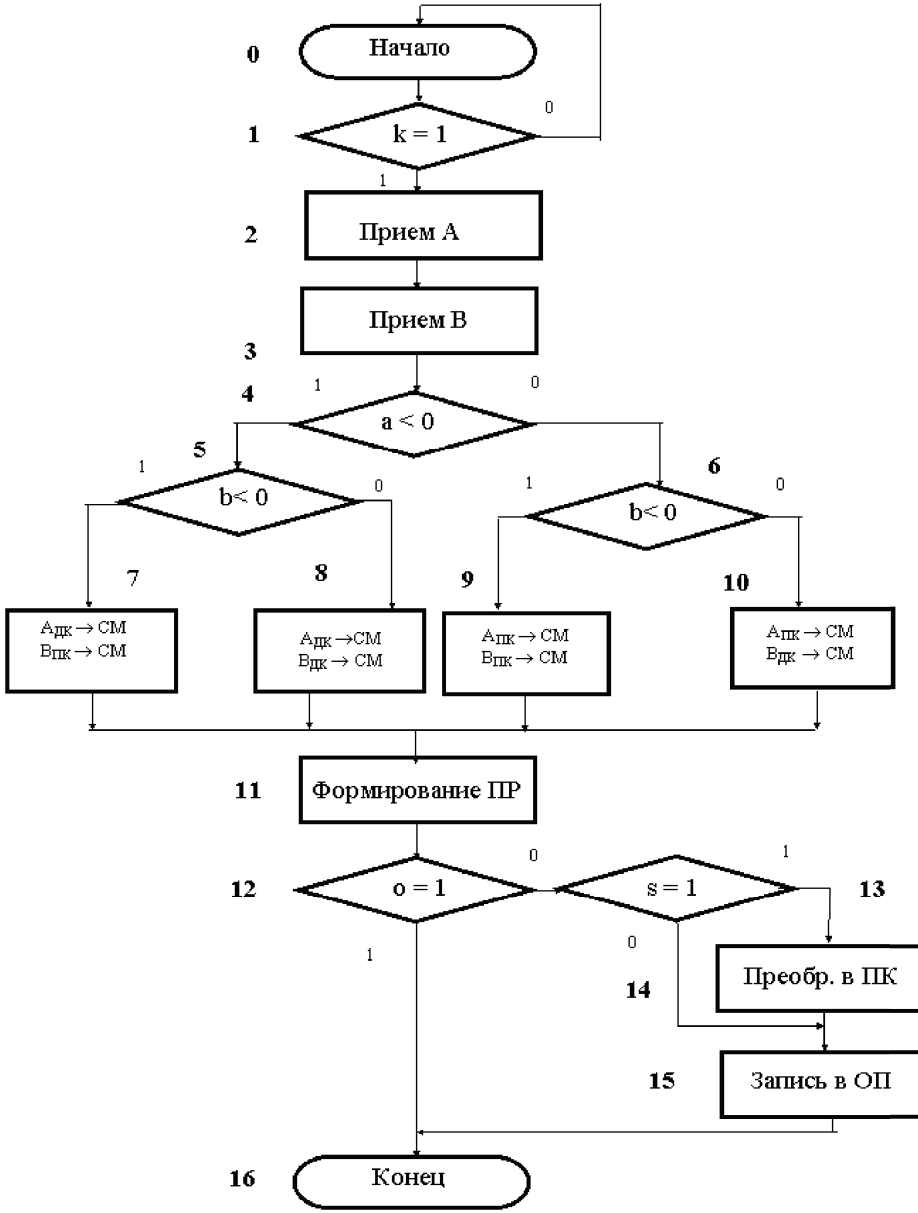

Рис. 7.2. Схема алгоритма вычитания

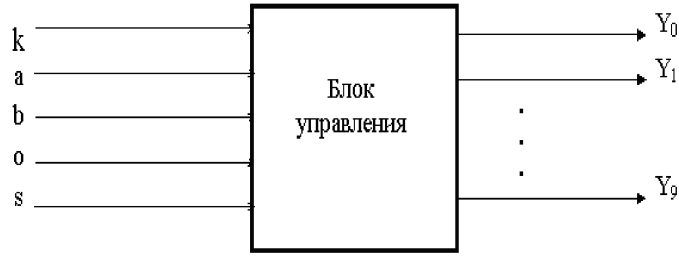

Рис. 7.3. Общая схема блока управления

Если в памяти хранится только одна микропрограмма, то память должна иметь ёмкость не менее 10 ячеек, так как количество микрокоманд (число безусловных операторов) равно 10. При этом разрядность адреса равна

$$N_A = \log_2 E_п \uparrow = \log_2 10 \uparrow = 4,$$

где $E_п$ - ёмкость памяти; $\uparrow$ - знак округления до ближайшего большего целого числа.

Таким образом, общая разрядность микрокоманды составит 19 разрядов.

Разметка схемы алгоритма заключается в распределении памяти, т. е. задании адреса для каждой микрокоманды. Вариант разметки приведен на рис. 7.4. Так как рассматривается вариант с принудительной адресацией микрокоманд, то каждая микрокоманда может быть записана в любой ячейке памяти. Однако для реализации разветвлений по логике рис. 7.1 адреса микрокоманд, следующих за любым оператором условного перехода, должны отличаться значением только младшего разряда. В некоторых случаях это нельзя обеспечить без введения холостых микрокоманд, которые не вызывают никаких действий, а используются исключительно для обеспечения условных переходов. Отметим, что введение холостых микрокоманд увеличивает время выполнения микропрограммы.

В данном примере реализуется одна микропрограмма. Поэтому первая микрокоманда размещается в ячейке с адресом 0000. Для реализации условных переходов введены 4 холостых микрокоманды, которые выделены пунктирными линиями. При этом количество микрокоманд и, соответственно, количество требуемых ячеек памяти увеличилось с 10 до 14, но не превысило 16, поэтому разрядность адреса не изменилась.

Заполнение полей микрокоманд показано в виде таблицы 7.2.

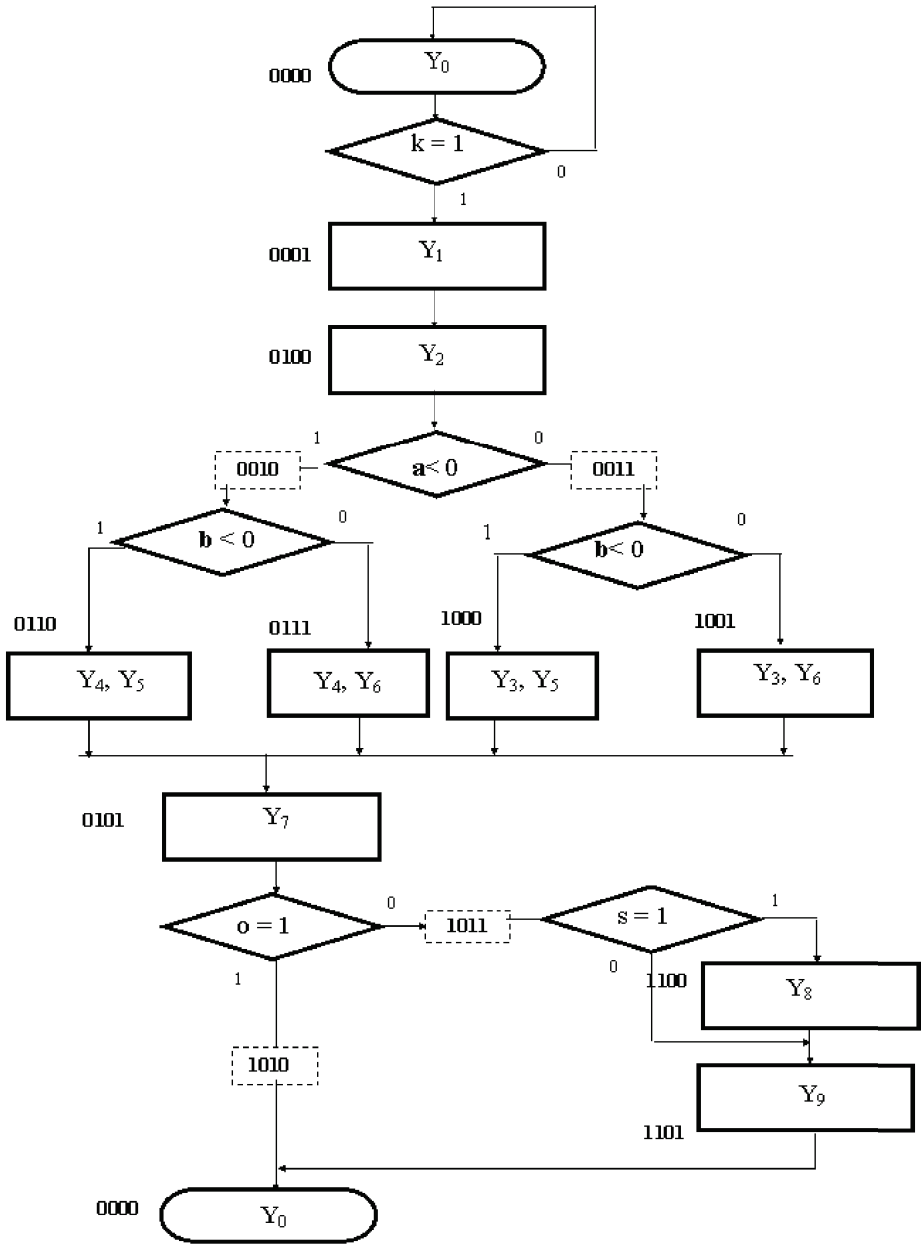

Рис. 7.4. Схема алгоритма с указанием всех входных и выходных сигналов

Таблица 7.2.

Адреса ячеек памяти				Микрокоманды																			
				Поле выходных сигналов										Поле условий					Адрес следующей МК				
A_3	A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	k	a	b	o	s	A_3	A_2	A_1	A_0	
0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	
0	0	0	1	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	1	1	0	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	1	0	0	0	
0	1	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	1	0	1	0	
0	1	1	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	1	0	1	
0	1	1	1	0	0	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0	1	0	1	
1	0	0	0	0	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	1	0	1	
1	0	0	1	0	0	0	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	1	0	1	
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	1	1	0	0	
1	1	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	1	1	0	1	
1	1	0	1	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	

Если в памяти записано несколько микропрограмм, то каждая микропрограмма размещается в отдельной области памяти. Для реализации микропрограммы на вход автомата поступает адрес её первой микрокоманды. После выполнения микропрограммы автомат выдаёт сигнал готовности и переходит в режим ожидания.

Контрольные вопросы

1. Поясните сущность аппаратного способа реализации алгоритмов.
2. Поясните сущность программного способа реализации алгоритмов.
3. Поясните структуру автомата с программируемой логикой.
4. Поясните назначение полей микрокоманды.
5. Как выполняются условные переходы в автомате с программируемой логикой?
6. Поясните алгоритм сложения чисел в арифметико-логическом устройстве.
7. Как проводится разметка микропрограммы?
8. Для чего используются холостые микрокоманды?
9. В каком случае введение холостых микрокоманд приводит к увеличению разрядности адреса микрокоманд?
10. Как выбирается одна из микропрограмм, размещённых в памяти?

8. Вероятностные автоматы

8.1. Особенности вероятностных автоматов

Ранее рассматривались детерминированные автоматы, в которых из любого состояния возможен переход только в одно из следующих состояний.

Опишем теперь структуру автомата, поведение которого в той или иной мере будет отвечать требованиям среды. Этот автомат называется **вероятностным**.

Устроен он подобно автомату с линейной тактикой, но его функции переходов и выходов являются случайными функциями. Т.е. задается вероятность переходов из одного состояния в другое, при поступлении на вход определенного сигнала.

Такой автомат, как правило, задается системой матриц, в которых на пересечении i - того столбца и k -той строки указывается вероятность перехода из i - того состояния в k-тое. В частном случае, когда такие матрицы содержат только «0» и «1», описывается уже знакомый нам детерминированный автомат.

Рассмотрим вероятностный автомат, иногда называемый автоматом Крылова. При поступлении сигнала поощрения этот автомат действует подобно детерминированному автомату, а при поступлении сигнала штрафа он с вероятностью 0,5 останется в прежнем положении, а с вероятностью 0,5 перейдет в другое состояние. Такой автомат, как будто не спешит менять свое действие, а случайным (но не изменяющимся с работой автомата образом) принимает решение о переходе, предварительно «подбросив монетку». Такой автомат еще можно назвать осторожным.

Приведем наглядный пример такого автомата.

Попытаемся формализовать способы спасения ночной бабочки от летучей мыши. Летучая мышь испускает направленный ультразвуковой сигнал и способна улавливать отраженный сигнал, причём с достаточно высокой точностью различать и идентифицировать сигналы, различая подвижные и неподвижные цели, земные цели и воздушные, маленькие и большие. Кроме того, отраженный сигнал позволяет летучей мыши с весьма большой точностью определять направления и расстояние до потенциальных целей.

Ночные бабочки также способны принять сигнал от летучей мыши и определить его интенсивность. Поведение бабочки различно в зависимости от того, как далеко от нее находится летучая мышь. Будем различать три маневра бабочки:

1. Бабочка начинала двигаться в сторону, противоположную прежнему движению.
2. Бабочка меняла направление в вертикальной плоскости, уходя от своего прежнего курса вверх или вниз.
3. Бабочка начинала хаотическое движение, т.е. переходит на такую траекторию полета, которая максимально затрудняет для нападающего предсказание следующей точки на этой траектории.

На рисунке 8.1. изображен граф смены состояний вероятностного автомата. Его особенность состоит в том, что для каждой группы состояний (они обведены на рисунке пунктиром) существует ненулевая вероятность

перейти в особое состояние, описывающее гибель автомата. Состояние 1 можно интерпретировать следующим образом:

1 - с P=0,3 летучая мышь обнаруживает бабочку, а с P=0,7 - пропускает ее;

2 - мышь определяет направление своего движения и с P= 0,8 цель при этом не теряется;

3 - летучая мышь настигает бабочку и уничтожает её с P=0,95.

Что может противопоставить преследователю бабочка? Для простоты изложения будем рассматривать каждую из групп состояний автомата, как определенную среду, задаваемую стратегией бабочки:

E1 - это прямой полет.

E2 - изменение направления движения в горизонтальной или вертикальной плоскости.

E3 - хаотическое движение.

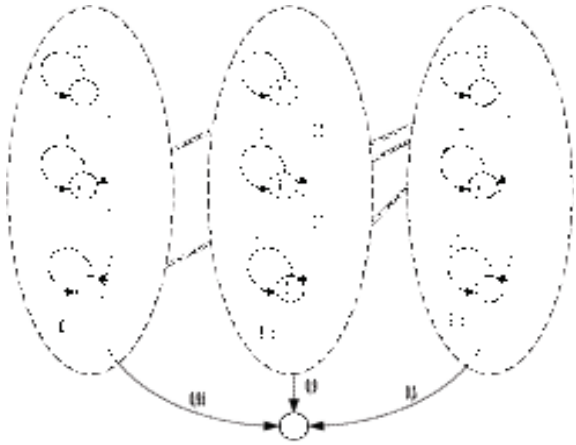

Рис. 8.1. Граф смены состояний вероятностного автомата

Действия бабочки сводятся к смене сред, переключению их. При этом автомат может реализовывать действие только в состоянии 2 или 3.

В приведенном примере действия, позволяющие бабочке максимально увеличить вероятность своего спасения, достаточно просты и прозрачны. Однако в общем случае выбор оптимальной последовательности переключений действий автомата, максимизирующий продолжительность его жизни, далеко не тривиален.

Рассмотрим пример вероятностного автомата на примере автомата управления светофором на перекрёстке улиц с различной интенсивностью движения. Этот автомат преимущественно пропускает транспорт по улице с интенсивным движением (магистрали) и не перекрывает её при появлении на

поперечной улице каждой отдельной машины. Естественно, что численные значения вероятностей переключения светофора и длительность его сигналов выбираются, исходя из реальных условий.

На перекрёстке установлены светофоры (С) и датчики (Д) наличия транспорта на поперечной улице. Сигналы от датчиков являются входными сигналами для автомата. Выходным сигналом автомата является сигнал управления светофором.

Будем считать, что автомат имеет только два состояния: проезд по магистрали открыт (Q_0) или закрыт (Q_1). Очевидно, что при открытом движении по магистрали, движение по поперечной улице запрещено и наоборот. Входным сигналом является сигнал от датчика наличия транспорта на поперечной улице. Граф автомата в этом случае может быть представлен в виде рис. 8.2.

На рис. 8.2. приняты следующие обозначения:

x – входной сигнал (x=1, если на поперечной улице есть транспорт);

p_{ij} – вероятность (переходная) перехода из состояния i в состояние j;

Q_0 – начальное состояние (проезд по магистрали открыт).

На рис. 8.2. видно, что из вершин графа вероятностного автомата могут выходить несколько дуг, отмеченных одинаковым входным сигналом.

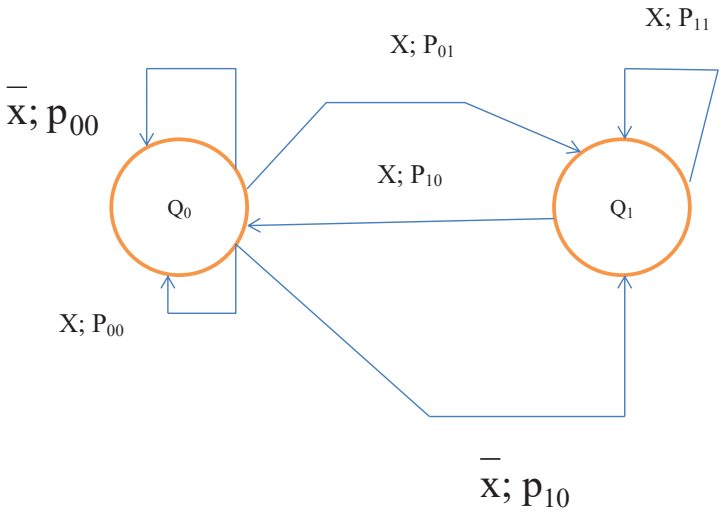

Рис. 8.2. Граф вероятностного автомата управления светофором

Вариант графа для конкретных значений переходных вероятностей приведён на рис. 8.3.

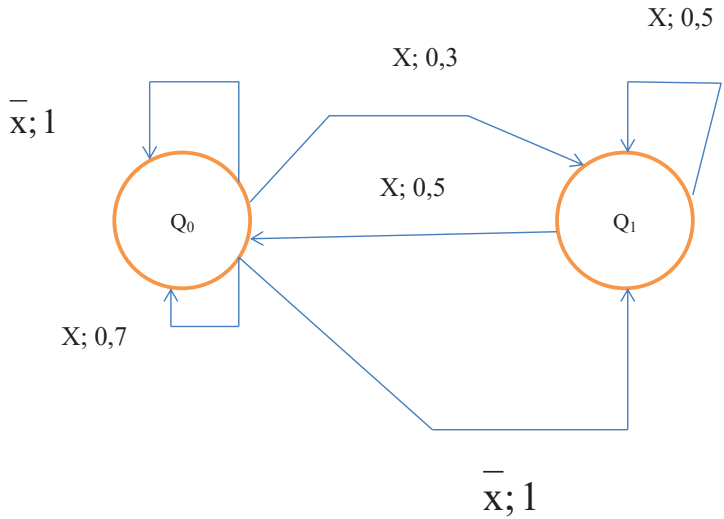

Рис. 8.3. Граф вероятностного автомата для конкретных значений переходных вероятностей

Таблица переходов вероятностного автомата приведена в таблицах 8.1 и 8.2 и имеет вид матрицы переходных вероятностей, которая составляется для каждого входного сигнала.

Таблица 8.1

X=0		
	P_0	P_1
P_0	1	1
P_1	0	0

Таблица 8.2

X=1		
	P_0	P_1
P_0	0,7	0,5
P_1	0,3	0,5

Вероятностный автомат может быть реализован в виде системы, состоящей из детерминированного автомата и датчика случайных чисел, который выдаёт сигналы с заданным распределением вероятностей. Для автомата, реализующего граф (рис. 8.3), достаточно использовать датчик, формирующий числа, равномерно распределённые в интервале от 0 до 1. В зависимости от состояния и входного сигнала случайное число сравнивается с пороговым значением (в данном случае это величины 0,3 или 0,5). Сигнал с

выхода схемы сравнения поступает на дополнительный вход автомата вместе с основным сигналом от датчика наличия транспорта. Тем самым реализуется вероятностная логика работы автомата.

Для большего приближения к реальности граф (рис. 8.3) можно дополнить состояниями Q_2 и Q_3, соответствующими жёлтому сигналу светофора. В этом случае граф принимает вид, приведённый на рис. 8.4.

Матрицы переходных вероятностей приведены в таблицах 8.3 и 8.4.

Таблица 8.3

X=0				
	P_0	P_1	P_2	P_3
P_0	1	0	0	0
P_1	0	0	0	1
P_2	0	0	0	0
P_3	0	0	0	0

Таблица 8.4

X=0				
	P_0	P_1	P_2	P_3
P_0	0,7	0	0,3	0
P_1	0	0,5	0	0,5
P_2	0	1	0	0
P_3	1	0	0	0

Из таблиц 8.3 и 8.4 видно, что сумма вероятностей переходов в любом состоянии равна 1 или 0 (если состояние недостижимо при данном входном сигнале).

Приведём определение недетерминированного автомата. Конечным недетерминированным распознающим автоматом называют совокупность (X, Q, Ф), где:

1) X – конечное множество символов входного алфавита;

2) Q – конечное множество состояний автомата, удовлетворяющее условиям:

а) во множестве Q выделено непустое подмножество начальных состояний $Q_0 \in Q$;

б) множество Q разбито на два непересекающихся класса – допускающие и не допускающие состояния.

3) Ф – функция переходов.

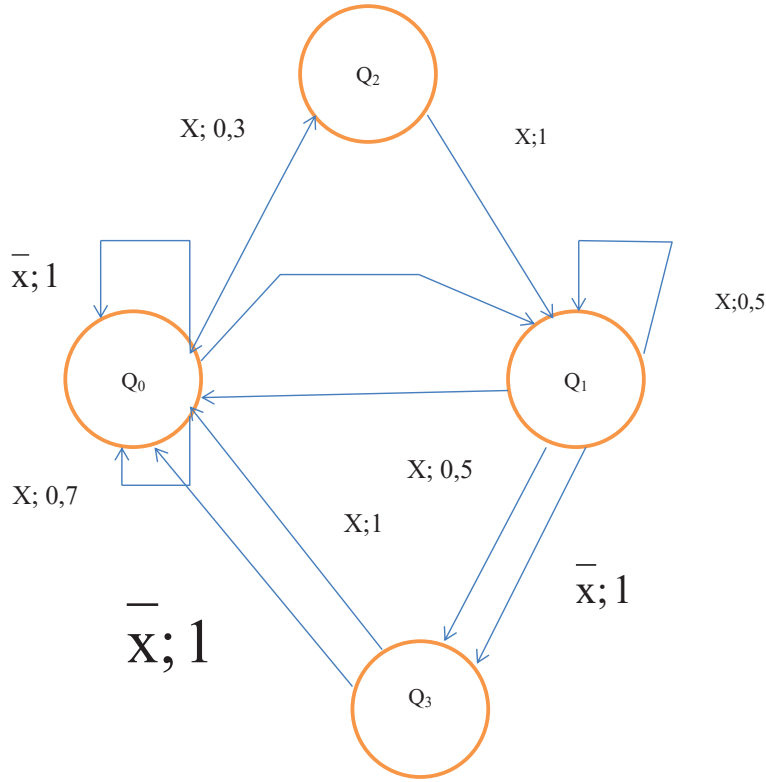

Рис. 8.4. Граф вероятностного автомата управления светофором с учётом жёлтого сигнала

8.2. Частные случаи вероятностных автоматов

По аналогии с детерминированными автоматами, можно определить вероятностные автоматы Мили и Мура. Вероятностные автоматы, у которых вероятности появления выходных сигналов (закон распределения) зависят лишь от состояний автомата, но не зависят от входных сигналов, называются вероятностными автоматами Мура. Если же вероятности появления выходных сигналов (закон распределения) зависят как от состояний автомата, так и от входных сигналов, имеем автомат Мили.

Рассмотрим некоторые частные случаи вероятностных автоматов. Может быть, что выходные сигналы автомата определяются детерминировано, а переходы автомата – случайно. Такие автоматы называются Y – детерминированными вероятностными автоматами. Если

состояния определяются детерминировано, то имеем А- детерминированный вероятностный автомат.

Если в процессе функционирования автомата законы распределения вероятностей появления выходных сигналов и вероятности перехода автомата в новые состояния не меняются во времени, то такие вероятностные автоматы называются **вероятностными автоматами с постоянной структурой**.

Очевидно, можно рассмотреть общий случай, когда эти законы распределения зависят от времени. Такие автоматы называются **вероятностными автоматами с переменной структурой.**

Вероятностные автоматы с переменной структурой в каждый фиксированный такт работы является некоторым обычным вероятностным автоматом, но в период между тактами вероятностный автомат может изменять свои матрицы переходных вероятностей или таблицы выходных вероятностей, или и то и другое вместе.

Часто при построении вероятностного автомата изменение вероятностей производят по некоторому закону, причем закон зависит от истории функционирования автомата (т.е. зависит от входных сигналов, поданных на него и от выходных сигналов, т.е. реакции автомата). Такие вероятностные автоматы с переменной структурой называются автоматами компенсирующего типа. Их разработке и уделяется основное внимание.

В этом случае можно сказать, что вероятностный автомат работает в некоторой среде (рис. 8.5), в которую он выдает выходные сигналы и из которой он получает входные.

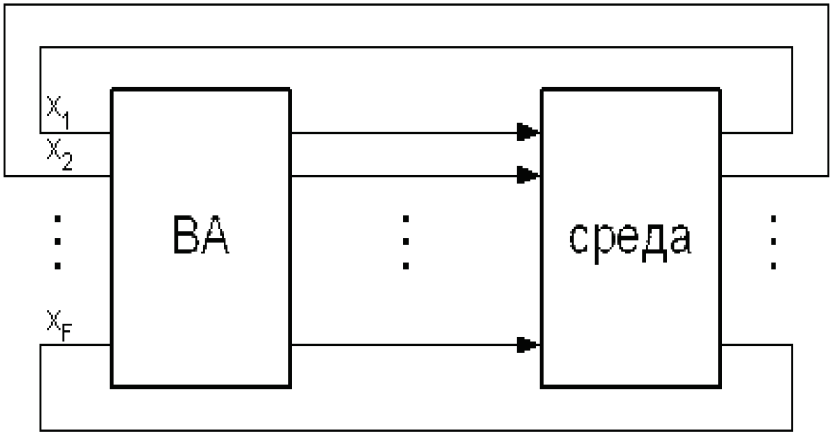

Рис.8.5. Работа вероятностного автомата в некоторой среде

В теории часто рассматриваются автоматы, называемые автоматами с ε- переходами (с эпсилон-переходами). Эпсилон-переходом называется

переход между состояниями, который может быть выполнен без входного сигнала. Такие переходы обозначаются символом ε. Использование эпсилон-переходов позволяет просто объединять несколько автоматов в один.

В теории автоматов доказано утверждение, что любой вероятностный автомат может быть заменён эквивалентным детерминированным (обычным) автоматом.

Вероятностные автоматы используются в тех случаях, когда схема вероятностного автомата проще, чем схема эквивалентного детерминированного автомата. Вероятностные автоматы могут применяться при моделировании умственной деятельности человека, например, при машинном переводе с одного языка на другой, в частности, при разработке трансляторов.

Контрольные вопросы

1. Поясните особенности вероятностных автоматов.
2. Как отмечаются дуги вероятностного автомата?
3. Как составляются матрицы переходных вероятностей?
4. Дайте определение конечного недетерминированного распознающего автомата.
5. Что такое эпсилон-переход?
6. Приведите пример вероятностного автомата.
7. Приведите способы задания вероятностного автомата.

9. Формальные грамматики и автоматы

9.1. Типы языков

В теории языков рассматриваются принципы и особенности построения различных языков. До начала XX века существовали только естественные (разговорные языки). При этом под языком понималось средство общения между людьми. С развитием лингвистики было установлено, что средства общения присущи не только человеку. В настоящее время под языком понимается любое средство общения.

Язык включает следующие составные части:
1) знаковую систему (множество допустимых последовательностей знаков);
2) множество смыслов этой системы;
3) соответствие между последовательностями знаков и смыслами.

В качестве знаков языка могут выступать: символы (буквы) некоторого алфавита (письменная форма языка); звуки (устная форма языка); жесты, цвет, запахи и т.д.

Наиболее развитыми являются знаковые системы на основе символов. Символ является простейшим элементом знаковой системы. Из символов строятся более сложные конструкции. При анализе разговорных языков иерархия конструкций языка выглядит следующим образом:

$$\underset{\text{(знак, символ)}}{\text{Буква}} \Rightarrow \text{слово} \Rightarrow \underset{\text{(фраза)}}{\text{предложение}} \Rightarrow \text{текст}.$$

В теории формальных языков используется формальный подход, при котором набор конструкций языка выглядит следующим образом:

Символ $\Rightarrow$ строка $\Rightarrow$ текст.

(знак, буква) (цепочка, предложение)

Отметим, что в ЭВМ вся информация представляется в виде строк. Таким образом, любое преобразование данных в ЭВМ заключается в преобразовании одних строк в другие.

В любом языке можно выделить правильные (допустимые) и неправильные конструкции. Правила построения правильных текстов составляют ***синтаксис*** языка. Описание соответствия между смыслами и текстами составляют ***семантику*** языка.

Семантика языка зависит от происхождения и характера языка, т.е. от характера объектов, описываемых языком. Синтаксис языка меньше зависит от характера языка. Поэтому при изучении синтаксиса можно использовать формальный подход.

Суть формального подхода заключается в том, что язык рассматривается как множество формальных объектов, построенных по определенным правилам. В качестве формальных объектов выступают последовательности символов. При построении таких последовательностей их смысл не учитывается. Появление и развитие формального подхода связано с необходимостью решения задач следующего типа:
1)машинный перевод с одного естественного языка на другой;
2) разработка трансляторов;
3)распознавание образов.

В зависимости от происхождения и степени универсальности языки можно разделить на типы, представленные на рис.9.1.

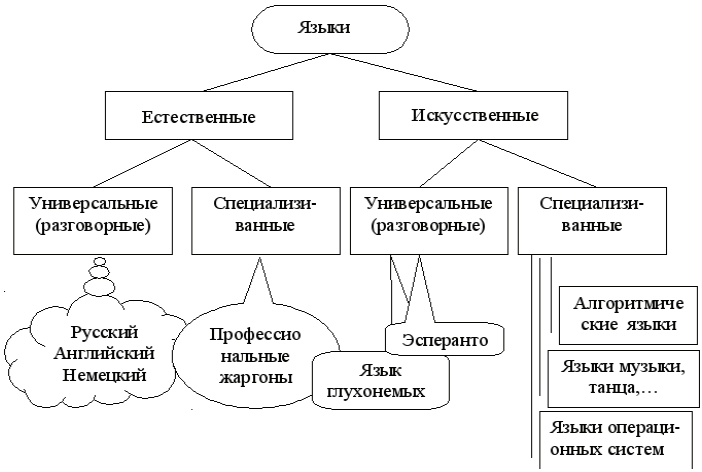

Рис. 9.1. Классификация языков

Естественные языки возникают и развиваются постепенно с развитием общества в течение длительного времени.

Искусственные языки разрабатываются специально для определенной области применения за относительно короткий период времени.

Универсальные языки используются для общения людей в повседневной жизни.

Специализированные языки являются средством общения достаточно узкого круга людей при обмене информацией в некоторой специальной области знаний. Примерами специализированных языков могут быть различные профессиональные жаргоны (язык пользователей ЭВМ), язык алгебры, язык алгебры логики и т.д.

9.2. Естественные языки и их особенности.

Особенности естественных языков вытекают из их происхождения. Основные особенности, затрудняющие формальный подход к изучению естественных языков, перечислены ниже.

Зависимость синтаксиса от семантики (смысла). Например, окончания слов могут зависеть от того, к каким объектам относятся эти слова, одушевленным или неодушевленным. Для примера рассмотрим две похожих фразы:

«Я увидел пень» (Что?).
«Я увидел оленя» (Кого?).

Семантическая и синтаксическая неоднозначность. Семантическая неоднозначность возникает из-за того, что некоторые слова могут иметь различный смысл. Например, фраза «Косой шел с косой» может иметь различный смысл в зависимости от контекста. Синтаксическая неоднозначность возникает из-за недостаточной строгости синтаксических правил. Например, фраза «Бытие определяет сознание» может быть истолкована различным образом. Если считать, что базовым элементом является бытие, то исходную фразу можно заменить на фразу «Бытие является главным и оно определяет сознание». Но исходная фраза может быть истолкована и так: «Бытие определяется сознанием».

Возможность появления парадоксальных предложений. Парадоксальные предложения построены так, что их нельзя отнести ни к истинным, ни к ложным. Примером парадоксального предложения является фраза: «Данное предложение является ложным».

Зависимость смысла от ситуации.

Изменчивость языка во времени.

Большое число синтаксических правил с многочисленными исключениями.

9.3. Формальные языки и их особенности

Большинство искусственных языков использует при построении предложений формальные, т.е. не зависящие от смысла, правила. Такие языки называются формальными. Синтаксис формальных языков должен обеспечивать возможность формального подхода к построению предложений. Поэтому формальные языки имеют следующие особенности:

1) независимость синтаксических правил от смысла;

2) ***невозможность появления парадоксов;***

3) ***строгость синтаксических правил и отсутствие исключений;***

4) ***конечное число синтаксических правил.***

Формальные языки, как и естественные, могут со временем изменяться. Но в отличие от естественных языков эти изменения проявляются не постепенно, а путем появления новых версий языков. При этом различные версии одного и того языка можно рассматривать как различные языки.

Наименьшей синтаксической единицей формального языка является символ.

Символ (буква) - это простой неделимый знак. Множество символов языка составляют ***алфавит***.

Примеры алфавитов:

Русский алфавит - {А, а, Б, б, В, в, …, Я, я }.

Латинский алфавит - {A, a, b, b, C, c, …, Z, z }.

Арабские цифры - { 0, 1, 2, . . . , 9 }.

Символы языка объединяются в строки. ***Строка*** – это упорядоченная последовательность символов.

Примеры строк:

Строки в русском алфавите – «упорядоченная» , «последовательность», «а».

Строки в латинском алфавите – «BEGIN» , «THEN» , «END» .

Строки в двоичном алфавите – «0», «1», «00», «10», «01010101» .

Количество символов в строке называется длиной строки. Строка символов $a_1a_2a_3 \ . \ . \ . \ a_n$ имеет длину n. Если длина строки равна нулю, то строка называется пустой. Пустая строка обозначается символом l . Заметим, что символ l не является символом алфавита.

Множество всех строк алфавита A называется замыканием над алфавитом A и обозначается символом A^*.

$$A^* = A^0 \cup A^1 \cup A^2 \cup ... \cup A^n = \sum_{n=0}^{\infty} A^n ,$$

где: $A^0 \in \{\lambda\}$ - пустая строка;

A^n - строка длины n.

Множество непустых строк A^+ определяется следующим образом:

$$A^+ = A^* / \{\lambda\} = \sum_{n=1}^{\infty} A^n .$$

Язык в общем случае представляет собой произвольное подмножество множества A^*. Так как множество A^* по определению бесконечно, то и язык в общем случае тоже является бесконечным. Поэтому язык невозможно задать явно, т.е. перечислением всех допустимых строк. Например, невозможно перечислить все правильные предложения русского языка. Для описания языка нужна специальная система, которая в сжатой форме описывает правила построения допустимых предложений языка. Такая система называется порождающей или формальной грамматикой. Язык, построенный при помощи формальной грамматики, называется формальным языком. Заметим, что строки языка, в том числе и формального, наделены определенным смыслом. Смысл строк языка определяется благодаря некоторой дополнительной информации.

9.4. Формальные грамматики

В теории формальных языков основными являются две задачи: Каким образом строить правильные предложения заданного языка?; Как установить, является ли данное предложение синтаксически правильным, допустимым для данного языка?

Первая задача решается при помощи формальных грамматик. Формальная грамматика содержит строгие правила порождения правильных предложений языка. Определение формальной грамматики имеет следующий вид:

Формальная (порождающая) грамматика **G** - это формальная система, определяемая четверкой символов:

G = { N, T, P, S },

где: **N** – конечное непустое множество нетерминальных (или вспомогательных) символов;

T – конечное непустое множество терминальных (или конечных) символов;

S – начальный или корневой символ;

P – конечное множество правил вывода (правил продукции).

Множества **N** и **T** являются непересекающимися множествами: $N \bigcap T = \varnothing$.

Множества **N** и **T** представляют собой алфавиты нетерминальных и терминальных символов.

Начальный символ **S** является исходным символом, из которого при помощи правил вывода строятся все предложения (цепочки) данного языка.

Правила вывода **P** определяют синтаксис формального языка, т.е. правила построения правильных предложений языка.

Обычно при описании формальных грамматик принимаются следующие обозначения:

1) символы нетерминального алфавита обозначаются прописными латинскими буквами (**A, B, C, D, . . . ,Z**);

2) символы терминального алфавита обозначаются строчными

буквами (**a, b, c, d, . . .**) первой половины латинского алфавита;

3) цепочки терминальных символов обозначаются строчными буквами (**. . . , x, y, z**) второй половины латинского алфавита;

4) цепочки терминальных и нетерминальных символов (смешанные цепочки) обозначаются строчными греческими буквами ($\alpha, \beta, \gamma, \varepsilon, \eta, \omega, \ldots$).

Правила вывода представляют собой правила подстановки вида:

$$\varepsilon \to \eta,$$

где ε и η - смешанные цепочки (цепочки в алфавите $V = N \bigcup T$).

Правило вывода $\varepsilon \to \eta$ означает, что цепочка ε заменяется цепочкой η.

Цепочка β ***непосредственно выводима*** из цепочки α в грамматике **G**, если $\alpha = \gamma\varepsilon\delta$, $\beta = \gamma\eta\delta$ и $\varepsilon \to \eta$ является правилом вывода из грамматики **G.** Например, если цепочка α представляет собой строку символов «ПРИМЕР» и существует правило вывода $МЕ \to БО$, то из цепочки α непосредственно выводится цепочка $\beta =$ «ПРИБОР».

Цепочка β называется ***выводимой*** из цепочки α, если существует последовательность правил вывода, переводящих цепочку α в цепочку β. Так, если $\alpha =$ «ПРИМЕР» и существуют правила вывода $МЕ \to БО$ и $И \to О$, то из цепочки α ***выводится*** цепочка $\beta =$ «ПРОБОР».

Формальным языком L (G), порожденным грамматикой G, называется множество всех цепочек в терминальном алфавите, выводимых из корневого символа S. Другими словами, формальный язык, порождаемый грамматикой G - это множество цепочек, удовлетворяющих двум условиям:

1) Цепочки состоят только из терминальных символов.

2) Каждая цепочка выводима из S при помощи последовательности правил вывода.

9.5. Типы формальных грамматик

Формальные грамматики отличаются друг от друга, прежде всего, типом правил вывода. Классификацию формальных грамматик по типу правил вывода ввел американский лингвист Ноам Хомский. По Хомскому формальные грамматики делятся на 4 типа.

***Формальная грамматика типа* 0** (неограниченная грамматика или грамматика произвольного типа) использует подстановки вида:

$$\alpha \to \beta,$$

где α и β - цепочки произвольного вида.

***Формальная грамматика типа* 1** или контекстная грамматика (контекстно-зависимая грамматика) использует подстановки вида:

$$\alpha A \beta \to \alpha \omega \beta,$$

где A – нетерминальный символ;

α, ω, β - цепочки произвольного вида, при этом цепочка ω не является пустой или $\omega \in (N \bigcup T)^+$;

α,β - контекст правила.

***Формальная грамматика типа* 2** или бесконтекстная грамматика (контекстно-свободная грамматика) использует подстановки вида:

$$A \to \alpha,$$

где α - произвольная непустая строка, т.е. $\alpha \in (N \cup T)^+$.

***Формальная грамматика типа* 3** или регулярная грамматика использует подстановки вида:

$A \to aB$ или $A \to a$,

где A и B – нетерминальные символы, a – терминальный символ.

В теории формальных языков доказывается, что все регулярные грамматики бесконтекстны, все бесконтекстные – контекстны, все контекстные – неограничены.

Пример регулярной грамматики:

$G = \{N, T, P, S\}$.

$N = \{S\}$;

$T = \{a, b\}$;

P: $S \to a$; $S \to b$; $S \to aS$; $S \to bS$.

При помощи этой грамматики порождаются строки символов a и b. Последовательность порождения строк можно пояснить следующей схемой, приведённой в таблице 9.1.

Таблица 9.1.

Применяемое правило вывода	Содержание строки
	S
$S \to aS$	aS
$S \to aS$	aaS
$S \to b$	aab

Результатом вывода является строка aab.

Пример бесконтекстной грамматики:

$G = \{N, T, P, S\}$.

$N = \{S\}$;

$T = \{a, b\}$;

P: $S \to aSb$; $S \to ab$.

При помощи этой грамматики порождаются строки вида $a^n b^n$.

Пример вывода строки a^3b^3 приведён в таблице 9.2.

Таблица 9.2.

Применяемое правило вывода	Содержание строки
	S
$S \to aSb$	aSb
$S \to aSb$	aaSbb
$S \to ab$	aaabbb

Пример бесконтекстной грамматики более сложного вида:

$G = \{N, T, P, S\}$.
$N = \{S\}$;
$T = \{IF, THEN, ELSE, U, B\}$
P: $S \to B$;
$S \to IF\ U\ THEN\ S$;
$S \to IF\ U\ THEN\ S\ ELSE\ S$.

Эта грамматика позволяет формировать различные условные операторы языка типа Паскаль. Например, оператор IF U THEN IF U THEN B ELSE B может быть сформирован следующим образом, приведённым в таблице 9.3.

Таблица 9.3.

Применяемое правило вывода	Содержание строки
	S
$S \to IF\ U\ THEN\ S$	IF U THEN S
$S \to IF\ U\ THEN\ S\ ELSE\ S$	IF U THEN IF U THEN S ELSE S
$S \to B$	IF U THEN IF U THEN B ELSE S
$S \to B$	IF U THEN IF U THEN B ELSE B

Возможен второй вариант. Он приведён в таблице 9.4.

Таблица 9.4.

Применяемое правило вывода	Содержание строки
	S
$S \to IF\ U\ THEN\ S\ ELSE\ S$	IF U THEN S ELSE S
$S \to IF\ U\ THEN\ S$	IF U THEN IF U THEN S ELSE S
$S \to B$	IF U THEN IF U THEN B ELSE S
$S \to B$	IF U THEN IF U THEN B ELSE B

9.6. Грамматический разбор

Грамматический разбор – это процедура установления правильности данного предложения в данной грамматике. Грамматический разбор может выполняться двумя способами – разбор «сверху вниз» и разбор «снизу вверх».

Разбор «сверху вниз» начинается с корневого символа. Сущность разбора заключается в попытках путем повторяющегося применения правил вывода получить заданное предложение. Если предложение представляет собой арифметическое или логическое выражение, то вначале подбирается подстановка, которая интерпретирует последнюю по порядку выполнения операции.

Разбор «снизу вверх» начинается с заданного предложения и заключается в попытках получить корневой символ при помощи инверсных правил подстановки. Если предложение представляет собой арифметическое или логическое выражение, то первым следует применить правило для

интерпретации той операции, которая выполняется первой при вычислении значения данного предложения.

Процедуру грамматического разбора удобно проводить, используя специальную разновидность графов – деревья.

Дерево - это конечное множество, состоящее из одного или более узлов. Узлы соединяются между собой ветвями. Из каждого узла может выходить несколько ветвей. При этом:

1) Существует один выделенный узел, называемый корнем дерева.

2) Остальные узлы разделены на непересекающиеся множества, каждое из которых является деревом.

Эти непересекающиеся множества называются поддеревьями. Число поддеревьев узла называется степенью узла. Узел с нулевой степенью называется листом. Листья представляют собой узлы, из которых не выходит ни одной ветви.

Отметим, что на рисунках корень дерева располагается вверху, а само дерево вычерчивается сверху вниз.

Рассмотрим пример использования деревьев при грамматическом разборе. Пусть задана следующая грамматика:

$G = \{N, T, P, S\}$.

$N = \{S\}$;

$T = \{a, b, c, \vee, \wedge, \neg, (,)\}$;

P: $S \to (S \vee S)$; $S \to (S \wedge S)$; $S \to \neg S$;

$S \to a$; $S \to b$; $S \to c$.

Данная грамматика позволяет записывать логические функции трех переменных. Например, последовательность разбора функции

$$\overline{a \vee b} \wedge \overline{c}$$

может быть представлена следующим образом, приведённым в таблице 9.5.

Таблица 9.5.

Применяемое правило вывода	Содержание строки
	S
$S \to (S \wedge S)$	$(S \wedge S)$
$S \to \neg S$	$(\neg S \wedge S)$
$S \to (S \vee S)$	$(\neg(S \vee S) \wedge S)$
$S \to \neg S$	$(\neg(S \vee S) \wedge \neg S)$
$S \to a$	$(\neg(a \vee S) \wedge \neg S)$
$S \to b$	$(\neg(a \vee b) \wedge \neg S)$
$S \to c$	$(\neg(a \vee b) \wedge \neg c)$

При помощи дерева последовательность вывода представляется в виде рис. 9.2.

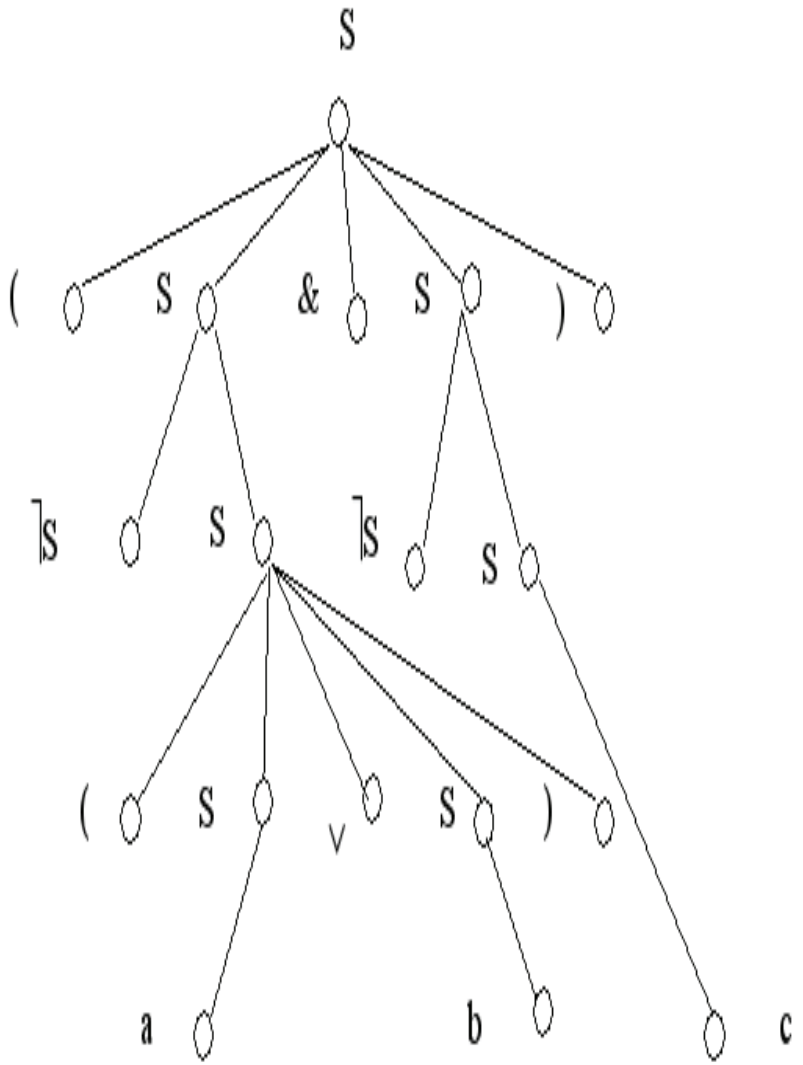

Рис. 9.2. Последовательность вывода при помощи дерева

При обходе листьев дерева слева направо можно записать полученное предложение:

$(\neg(a \vee b) \wedge \neg c)$.

При грамматическом разборе «снизу вверх» в качестве исходных данных используется проверяемое предложение. Последовательность разбора предложения $(\neg(S \vee S) \wedge \neg S)$ показана на рис. 9.3.

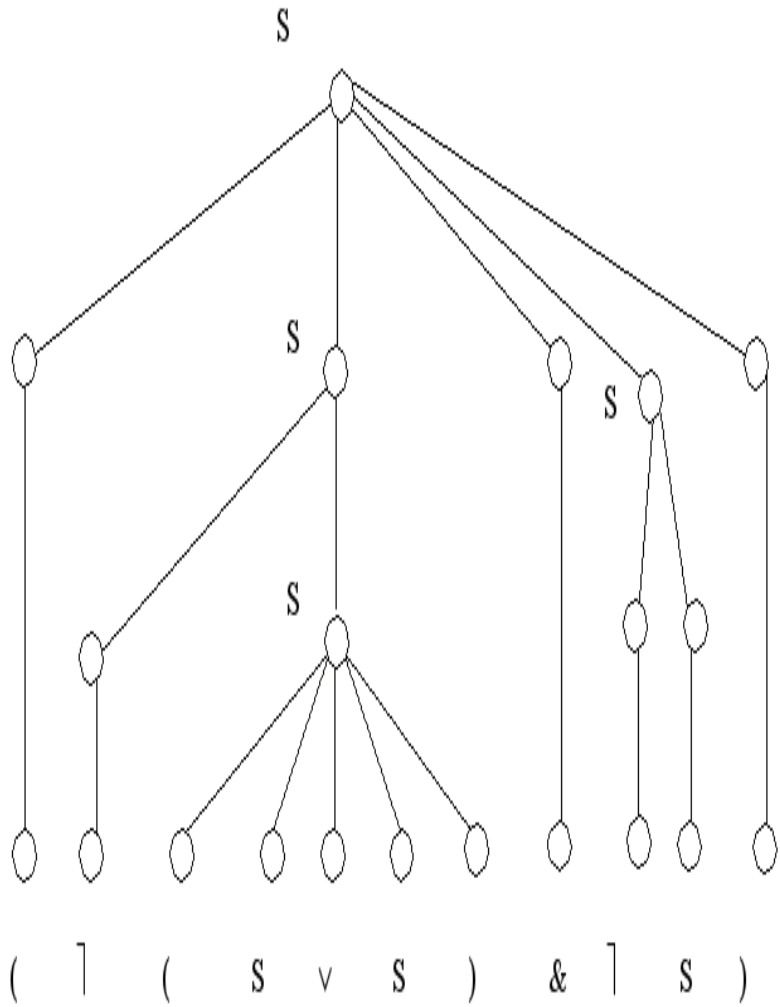

Рис. 9.3. Последовательность разбора предложения $(\neg(S \vee S) \wedge \neg S)$.

При грамматическом разборе основными операциями являются операции поиска символов в строке и операции сравнения. Эти операции могут быть выполнены при помощи автоматов с памятью.

9.7. Автоматы и формальные грамматики

В формальных грамматиках основной операцией преобразования данных является реализация одного из прямых или инверсных правил вывода (правил продукции или подстановки). В общем случае подстановка заключается в замене строки одного алфавита на другую строку (обычно другого алфавита). Задача замены строк может быть решена в несколько этапов:

1) Разбиение исходного текста на конструкции языка.
2) Выделение простых конструкций.
3) Подбор правила вывода.
4) Замена простой конструкции в соответствии с выбранным правилом вывода.

Реализация каждого этапа является строго формальной процедурой. Поэтому для каждого этапа может быть построен автомат с памятью. Для первых трех этапов необходимо построить автоматы, которые могут обнаружить в тексте символы начала и окончания определенных конструкций языка. Автомат такого типа называют распознавателем. При обнаружении заданной строки в последовательности входных символов он выдает определенный сигнал. Для четвёртого этапа можно использовать автомат-преобразователь, выдающий по сигналу распознавателя определенную последовательность выходных сигналов.

Таким образом, для решения задачи грамматического разбора вполне достаточно рассмотренных выше методов синтеза автоматов. В первую очередь для этого подходят автоматы с программным формированием выходных сигналов, которые легко перестраиваются с учетом заданной грамматики. Таким автоматом может быть компьютер. Для решения задачи построения правильных предложений в заданной грамматике в настоящее время также используют компьютер.

Контрольные вопросы

1. Какие составные части включает язык, рассматриваемый как средство общения?
2. Из каких элементов может состоять знаковая система языка?
3. Что такое семантика языка?
4. Перечислите возможные типы языков.
5. Приведите особенности естественных языков.
6. Приведите особенности формальных языков.
7. Что такое алфавит?
8. Как задаётся формальная грамматика?
9. Какие типы символов используются при задании формальной

грамматики?

10. Что такое формальный язык?
11. Дайте определение неограниченной формальной грамматики.
12. Дайте характеристику контекстной формальной грамматики.
13. Дайте характеристику бесконтекстной формальной грамматики.
14. Дайте характеристику регулярной формальной грамматики.
15. Что такое грамматический разбор?

Литература

1. А.Г.Рощин , Р.М.Половов. Теория автоматов. Учебное пособие. Часть 2.– М.:МГТУГА, 2008.
2. Н.Н.Горнец, А.Г.Рощин, В.В.Соломенцев. Организация ЭВМ и систем. Учебное пособие для ВУЗов. –М.: Издательский центр «Академия», 2008.
3. Микушин А.В., Сажнев А.М., Синин В.И. Цифровые устройства и микропроцессоры. СПб, БХВ-Петербург, 2010.
4. Угрюмов Е. П. Цифровая схемотехника. СПб, БХВ-Петербург, 2010.

Содержание

Введение		3
1.	Основные структуры автоматов	4
1.1.	Автомат Мили	4
1.2.	Обобщённая структура автомата Мили	5
1.3.	Автомат Мура	6
1.4.	Обобщённая структура автомата Мура	7
	Контрольные вопросы	9
2.	Элементы памяти цифровых автоматов	9
2.1.	Триггеры. Общие сведения о триггерах	9
2.2.	RS – триггер	10
2.2.1.	Асинхронный RS-триггер	10
2.2.2.	Синхронный RS-триггер	11
2.3.	D-триггер	12
2.4.	T-триггер	14
2.5.	JK – триггер	15
	Контрольные вопросы	17
3.	Методика синтеза автоматов с памятью	17
3.1.	Последовательность синтеза автоматов с памятью	17
3.2.	Основные этапы синтеза	18
3.3.	Пример синтеза автомата с памятью	20
	Контрольные вопросы	23
4.	Частные случаи синтеза автоматов с памятью	24
4.1.	Общие сведения о микрокомандах и микропрограммах	24
4.2.	Влияние типа триггера на схему автомата	25
4.3.	Синтез автоматов с использованием T-триггеров	26
4.4.	Синтез автоматов с использованием RS-триггера	27
4.5.	Синтез автоматов с использованием JK-триггера	28
4.6.	Причины частичной определённости автоматов	29
4.7.	Использование частичной определенности автомата для упрощения его схемы.	31
	Контрольные вопросы	34
5.	Синтез блоков управления	35
5.1.	Особенности блоков управления с распределителями импульсов	35
5.2.	Последовательность синтеза блоков управления с распределителями импульсов	39
	Контрольные вопросы	39
6.	Гонки в автоматах	40
6.1.	Сущность эффекта гонок	40
6.2.	Методы борьбы с гонками	41
6.2.1.	Противогоночное кодирование	41
6.2.2.	Синхронизация работы автомата	43
6.2.3.	Использование двухтактных триггеров	44
	Контрольные вопросы	46
7.	Автоматы с программным формированием выходных сигналов	46
7.1.	Способы реализации алгоритмов	46
7.2.	Структура автомата с программным формированием выходных сигналов	47
7.3.	Синтез автоматов с программным формированием выходных сигналов	49
	Контрольные вопросы	54
8.	Вероятностные автоматы	54
8.1.	Особенности вероятностных автоматов	54
8.2.	Частные случаи вероятностных автоматов	60
	Контрольные вопросы	62

9. Формальные грамматики и автоматы…… 62
9.1. Типы языков…… 62
9.2. Естественные языки и их особенности…… 64
9.3. Формальные языки и их особенности…… 65
9.4. Формальные грамматики…… 66
9.5. Типы формальных грамматик…… 67
9.6. Грамматический разбор…… 69
9.7. Автоматы и формальные грамматики…… 73
Контрольные вопросы…… 73
Литература…… 74