МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

СИСТЕМА
ОТКРЫТОГО
ОБРАЗОВАНИЯ

Л.И. Федосеева, Р.М. Адилов, М.Н. Шмокин

ОСНОВЫ ТЕОРИИ КОНЕЧНЫХ АВТОМАТОВ И ФОРМАЛЬНЫХ ЯЗЫКОВ

Рекомендовано федеральным государственным бюджетным образовательным учреждением высшего профессионального образования “Московский государственный технический университет имени Н.Э. Баумана” в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению “Информатика и вычислительная техника”

Пенза
ПГТА
2013

УДК 519.713

Рецензент –
Уполномоченная Министерством образования
и науки Российской Федерации организация
Федерального Государственного бюджетного образовательного
учреждения высшего профессионального образования “Московский
государственный технический университет им. Н.Э. Баумана”
(регистрационный номер рецензии 2319 от 24 апреля 2013 г.)

Федосеева, Л.И.

Основы теории конечных автоматов и формальных языков : Учебное пособие / Л.И. Федосеева, Р.М. Адилов, М.Н. Шмокин. – Пенза : Изд-во Пенз. гос. технол. ун-та, 2013. – 136 с.: 79 ил., 62 табл., библиогр. 25 назв.

Рассматриваются основы теории конечных автоматов и формальных грамматик, операторных схем алгоритмов, способы задания и общие методы абстрактного и структурного синтеза цифровых автоматов. Уделяется внимание разработке цифровых схем комбинационного действия и схем с памятью.

В начале каждой главы приводится краткое изложение теории, затем подробно рассматриваются примеры и задачи с решениями. Приводятся контрольные вопросы.

Учебное пособие подготовлено на кафедре «Вычислительные машины и системы» Пензенского государственного технологического университета в соответствии с Федеральным государственным образовательным стандартом ВПО и предназначено для студентов, обучающихся по направлению подготовки бакалавров 230101.62 “Информатика и вычислительная техника”.

УДК 519.713

Предисловие

“Теория автоматов” является одной из первых дисциплин, которая закладывает теоретическую базу для освоения специальных дисциплин. При её изучении студенты должны получить знания об основах теории формальных грамматик, о способах задания цифровых автоматах на начальных и на стандартных языках, об общих методах синтеза цифровых схем комбинационного действия и схем с памятью, а также о методах синтеза устройств управления на алгоритмическом и структурном уровнях.

Студенту предлагается переработанное и дополненное издание учебного пособия “Элементы теории цифровых автоматов”, вышедшего с грифом Учебно-методического объединения вузов по образованию в области машиностроения и приборостроения. Опыт использования в учебном процессе этого издания показал необходимость включения дополнительных разделов, в которых излагаются основы теории формальных грамматик и методика абстрактного синтеза автоматов. Большое внимание уделено вопросам синтеза автоматов без памяти – комбинационных логических схем и рассмотрению принципов проектирования цифровых автоматов на структурном уровне.

От студента не требуется предварительного знакомства с теорией автоматов, так как все необходимые определения и понятия изложены в первых главах. При написании пособия предполагалось, что студенту известны основные положения дисциплины “Дискретная математика”.

Глава 1. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ АБСТРАКТНЫХ АВТОМАТОВ

1.1. Определение цифрового автомата

Под *цифровым автоматом* будем понимать устройство, предназначенное для преобразования цифровой (дискретной) информации, способное переходить под воздействием входных сигналов из одного состояния в другое и выдавать выходные сигналы. Отличительные особенности цифровых автоматов заключаются в том, что они имеют дискретное множество внутренних состояний, и переход из одного состояния в другое осуществляется скачкообразно. Дискретность информации, преобразуемой в автомате, проявляется в том, что она представляется посредством набора слов конечной длины в некотором алфавите.

Реальные цифровые автоматы конечны, т. е. множество входных и выходных сигналов, а также число входных и выходных каналов и множество состояний автомата конечны.

Цифровые автоматы функционируют в дискретные моменты времени, временной интервал T между которыми называется *тактом*. В зависимости от определения времени T различают автоматы синхронного и асинхронного действия.

Для *цифрового автомата синхронного действия* входные сигналы действуют в строго определённые моменты времени при $T = \text{const}$, определяемые генератором синхронизирующих импульсов, в которые возможен переход автомата из одного состояния в другое.

Для *цифрового автомата асинхронного действия* $T \neq \text{const}$ и определяется моментами поступления входных сигналов, а переход автомата из одного состояния в другое осуществляется при неизменном состоянии входа.

Для *идеализированных цифровых автоматов* не учитываются переходные процессы в элементах схемы автомата и разница в фактических величинах T для правильного функционирования автомата не имеет значения, поэтому для описания законов функционирования цифровых автоматов вводят абстрактное время, принимающее целые неотрицательные значения $t = 0, 1, 2...$

По степени детализации описания произвольных цифровых автоматов различают автоматы абстрактные и структурные. В соответствии с этими классами различают абстрактную и структурную теорию цифровых автоматов.

В *абстрактной теории* цифровых автоматов изучаются наиболее общие законы поведения без учёта конечной структуры автомата и физической природы информации. На уровне абстрактной теории понятие “работа автомата” понимается как преобразование входных слов в выходные слова. Можно сказать, что цифровой автомат рассматривается как “чёрный ящик” и основное внимание уделяется его поведению относительно внешней среды. В этом случае автомат – это математическая модель, описывающая поведение технического устройства.

Математической моделью технического устройства является *абстрактный автомат,* определяемый как шестикомпонентный кортеж $S = < A, z, w, \delta, \lambda, a_1 >$, у которого:

1) $A = \{a_1, ..., a_m, ..., a_M\}$ – множество состояний (алфавит состояний);

2) $Z = \{z_1, ..., z_f, ..., z_F\}$ – множество входных сигналов (входной алфавит);

3) $W = \{w_1, ..., w_g, ..., w_G\}$ – множество выходных сигналов (выходной алфавит);

4) функция переходов δ ($\delta: A \times Z \to A$) определяет правила перехода автомата из одного состояния в другое в зависимости от значений входных сигналов и состояния автомата;

5) функция выходов λ ($\lambda: A \times Z \to W$) определяет правила формирования выходных сигналов автомата;

6) a_1 – начальное состояние автомата, $a_1 \in A$.

Под *алфавитом* здесь понимается непустое множество попарно различных символов. Элементы алфавита называют буквами, а конечную упорядоченную последовательность букв – словом в данном алфавите. Пустое слово (последовательность нулевой длины) будем обозначать буквой *e*.

Автомат имеет один вход и один выход (рис. 1.1) и работает в некотором идеализированном дискретном времени, принимающем целые неотрицательные значения $t = 0, 1, 2...$

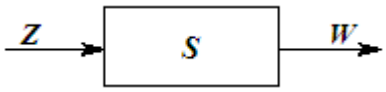


Рис. 1.1. Упрощенная схема абстрактного автомата

В каждый момент t дискретного времени автомат находится в некотором состоянии $a(t) \in A$, причём в начальный момент $t = 0$ он всегда

находится в начальном состоянии $a(0) = a_1$. Будучи в момент времени t в состоянии $a(t)$, автомат способен воспринимать на своём входе сигнал $z(t) \in Z$. В соответствии с функцией выходов в этот же момент времени он выдает выходной сигнал $w(t) \in W$ и в следующий момент времени $(t+1)$ согласно функции переходов перейдёт в состояние $a(t+1)$ A. Смысл понятия абстрактного автомата состоит в том, что он реализует некоторое отображение множества слов входного алфавита Z во множество слов выходного алфавита W. Иначе говоря, если на вход автомата, установленного в начальное состояние a_1, подавать буква за буквой некоторую последовательность букв входного алфавита $z(0)$, $z(1)$, $z(2)$... – входное слово, то на выходе автомата будут последовательно появляться буквы выходного алфавита $w(0)$, $w(1)$, $w(2)$... – выходное слово. Относя к каждому входному слову, соответствующее ему выходное слово, получаем отображение, индуцированное абстрактным автоматом. Термин “абстрактный” используется в связи с идеализированным дискретным временем, а также потому, что в этой модели абстрагируются от реальной физической природы входных и выходных сигналов, рассматривая их как буквы некоторого алфавита.

Исходя из изложенного, можно отметить, что если у двух абстрактных автоматов входной и выходной алфавиты совпадают и они индуцируют одно и то же отображение множества слов во входном алфавите во множество слов в выходном алфавите, то такие автоматы эквивалентны.

Структурный цифровой автомат учитывает структуру входных и выходных сигналов, а также внутреннее устройство цифрового автомата на уровне структурных схем. Это означает, что структурная теория цифровых автоматов изучает общие приёмы построения структурных схем автоматов на основе элементарных автоматов.

1.2. Варианты цифровых автоматов

На практике наибольшее распространение получили два класса автоматов – *автоматы Мили* и *автоматы Мура*, названные по имени впервые исследовавших эти модели американских учёных *G.H. Mealy* и *E.F. Moore*.

Автомат Мили. Закон функционирования автомата Мили задаётся уравнениями:

$$a(t+1) = \delta\ (a(t), z(t));$$
$$w(t) = \lambda\ (a(t), z(t)); \qquad (1.1)$$

$a(0) = a_1$, $t = 0, 1, 2...$

Автомат Мура. Закон функционирования автомата Мура задаётся уравнениями:

$$a(t+1) = \delta\ (a(t), z(t));$$
$$w(t) = \lambda\ (a(t)); \qquad (1.2)$$
$$a(0) = a_1,\ t = 0, 1, 2...$$

Как видно из уравнений (1.1) и (1.2), эти автоматы различаются способом определения выходного сигнала. В автомате Мили функция определяет выходной сигнал в зависимости от состояния автомата и входного сигнала в момент времени *t*, а в автомате Мура накладываются ограничения на функцию λ, заключающиеся в том, что выходной сигнал зависит только от состояния автомата и не зависит от значения входных сигналов. Отсюда следует важное отличие в функционировании этих автоматов: выходные сигналы автомата Мура отстают на один такт от выходных сигналов автомата Мили, эквивалентного ему.

Совмещённая модель автомата (С-автомат). Под абстрактным *С*-автоматом понимают математическую модель цифрового устройства, определяемую восьмикомпонентным вектором

$$S = <A, Z, W, U, \delta, \lambda_1, \lambda_2, a_1>,$$

где $A = \{a_1, ..., a_M\}$ – множество состояний;

$Z = \{z_1, ..., z_F\}$ – входной алфавит;

$W = \{w_1, ..., w_G\}$ – выходной алфавит автомата Мили;

$U = \{u_1, ..., u_H\}$ – выходной алфавит автомата Мура;

δ – функция переходов автомата, δ: $A \times Z > A$;

λ_{11} – функция выходов автомата Мили, λ_1: $A \times Z > W$;

λ_{22} – функция выходов автомата Мура, λ_2: $A \times Z > U$;

$a_1 \in A$ – начальное состояние автомата.

Абстрактный *С*-автомат можно представить в виде устройства (рис. 1.2) с одним входом, на который поступают сигналы из входного алфавита *Z*, и двумя выходами, на которых появляются сигналы из алфавитов *W* и *U*.

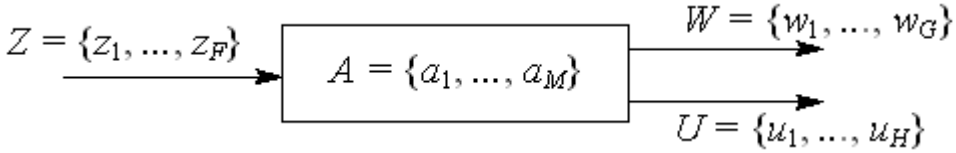


Рис. 1.2. Схема абстрактного С-автомата

Закон функционирования *С*-автомата можно описать уравнениями:

$$a(t + 1) = \delta\ (a(t), z(t));$$
$$w(t) = \lambda_1(a(t), z(t)); \qquad (1.3)$$

$u(t) = \lambda_2(a(t))$;

$a(0) = a_1$; $t = 0, 1, 2...$

Отличие *C*-автомата в том, что он одновременно реализует две функции выходов λ_1 и λ_2, каждая из которых характерна для модели Мили и модели Мура в отдельности. Выходной сигнал $u_h = \lambda(a_m)$ выдаётся всё время, пока автомат находится в состоянии a_m. Выходной сигнал $w_g = \lambda_1(a_m, z_f)$ выдаётся во время действия входного сигнала z_f при нахождении автомата в состоянии a_m.

Очевидно, что от *C*-автомата легко перейти к автоматам Мили и Мура с учетом возможных сдвигов выходных сигналов на один такт, аналогично тому, как возможен переход от автомата Мили к автомату Мура и наоборот. На практике много реальных автоматов работает по модели *C*-автомата.

Автомат без памяти (комбинационная схема). Алфавит состояний такого автомата содержит единственную букву, поэтому понятие функции переходов вырождается и становится ненужным для описания работы автомата. Автомат задается тремя объектами: *Z*, *W*, λ. Функция выходов принимает вид

$$w(t) = \lambda(z(t)), \qquad (1.4)$$

т.е. выходной сигнал в данном такте зависит только от входного сигнала того же такта и никак не зависит от ранее принятых сигналов.

Такое преобразование выполняется комбинационной логической схемой, в которой каждой входной комбинации сигналов соответствует определенная выходная комбинация. Описание работы таких схем осуществляется с использованием таблиц истинности и булевых функций.

Автономный автомат. В таком автомате входной алфавит состоит из одной буквы. Автомат задается четырьмя объектами:

$<A, W, \delta, \lambda>$ с возможным выделением начального состояния a_1. Функции переходов и выходов имеют вид:

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

$$w(t) = \lambda(a(t)). \qquad (1.5)$$

Эта пара выражений однозначно определяет единственную выходную последовательность, выдаваемую автоматом, и последовательность состояний:

$w(1)\ w(2)\ w(3)\ ...$

$a(1)\ a(2)\ a(3)\ ...$ (1.6)

Если автономный автомат конечен и число его состояний равно *k*,

то среди значений $a(1)$, $a(2)$, ..., $a(k)$ найдутся повторяющиеся состояния. Следовательно, каждая из последовательностей (1.6) окажется периодической с длиной периода не больше числа состояний автомата и, возможно, с некоторым предпериодом.

Автономные автоматы используются для построения генераторов периодических последовательностей, генераторов синхросерий и в других задающих устройствах, применяемых в цифровой технике.

Автомат без выхода. В таком автомате выходной алфавит содержит только одну букву. Автомат задается тремя объектами: A, Z, δ. Из функций, задающих поведение автомата, сохраняется лишь функция переходов: $a(t+1) = \delta\ (a(t), z(t))$.

Поведение автомата без выхода не может быть охарактеризовано в терминах операторов (отображений), перерабатывающих входные слова в выходные. Вместо этого можно было бы рассматривать операторы, перерабатывающие входные последовательности в последовательности внутренних состояний. Однако удобнее рассматривать поведение такого автомата, задавая настройку его путем выделения начального состояния и множества финальных состояний. После такой настройки автомат может рассматриваться как устройство, воспринимающее вопросы (входные последовательности) и выдающее ответы «да» или «нет» в зависимости от того, принадлежит ли слово интересующему нас языку или нет. Если после подачи на вход слова автомат окажется в одном из финальных состоянии, то ответ утвердителен, в противном случае – отрицателен. В лингвистической интерпретации вопросы, задаваемые автомату, приобретают следующий смысл: является ли данная цепочка символов грамматически правильным предложением в рассматриваемом языке или нет.

Если поведение автомата без памяти и автономного автомата сравнительно просто, то поведение автомата без выхода оказывается столь же разнообразно, как и поведение автомата с выходом. Это следует из того, что из двух функций, задающих поведение автомата, функция переходов является наиболее информативной.

Модель дискретного преобразователя В.М. Глушкова

Каждый цифровой автомат можно представить моделью В.М. Глушкова, состоящей из двух тесно взаимодействующих между собой блоков, один из которых выполняет функции операционного автомата (ОА), а другой – управляющего автомата (УА). Такое подразде-

ление отражает структуру и функции любого устройства ЭВМ, предназначенного для обработки цифровой информации и называемого операционным устройством. На рис. 1.3 представлена обобщенная структура произвольного операционного устройства, где D_1 и D_2 – шины, по которым поступает исходная информация и результаты ее обработки, соответственно; X – шины, по которым поступают сигналы, характеризующие состояние ОА (например, отрицательный результат, переполнение сумматора и т.д.). Эти сигналы часто называют осведомительными; Y – шины, по которым поступают управляющие сигналы из УА на ОА в соответствии с алгоритмом выполняемой в ОА операции; g – шины, по которым поступают сигналы, определяющие выполняемую операцию; СС – стартовый сигнал пуска операционного устройства; КР – сигнал, характеризующий конец работы алгоритма.

Таким образом, можно отметить, что ОА реализует действия над исходной информацией (словами), т.е. является исполнительной частью операционного устройства, а управляющий автомат управляет работой ОА, т.е. вырабатывает необходимые последовательности управляющих сигналов в соответствии с алгоритмом выполняемой операции.

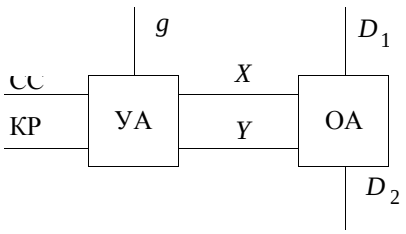


Рис. 1.3. Обобщенная структура операционного устройства

Управляющие автоматы используются не только в операционных устройствах вычислительной техники в системе УА – ОА, но и в разнообразных системах автоматики по управлению технологическими процессами и объектами, которые обобщенно можно назвать объектами управления (ОУ). В этом случае УА, в соответствии с алгоритмом функционирования ОУ и в зависимости от значений осведомительных сигналов о состоянии ОУ, поступающих от датчиков ОУ, и возможных внешних сигналов, вырабатывает и подает на исполнительные механизмы ОУ последовательность управляющих сигналов, обеспечивающих выполнение операций или процедур, предусмотренных целями управ-

ления. Для сложных систем управления, когда требуется выполнять большой объем работ по обработке информации о состоянии ОУ и выработке управляющих воздействий на ОУ в зависимости от целевой функции управления, в качестве УА могут выступать ЭВМ соответствующего класса.

Микропрограммные автоматы. Управляющий автомат, реализующий микропрограмму выполнения операции обработки информации, часто называют микропрограммным автоматом (МПА). Это вытекает из следующей интерпретации взаимодействия ОА и УА 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 УА $Y = \{y_1, y_2, ..., y_N\}$. В случае если несколько микроопераций реализуются в ОА одновременно, то это подмножество микроопераций называют *микрокомандой.*

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

В. Последовательность выполнения микрокоманд, определяемая функциями перехода УА, записывается в виде алгоритма, представленного в терминах микроопераций и ЛУ и называемого *микропрограммой.* В качестве начального языка для описания микропрограмм выполнения операций чаще всего используется язык операторных схем алгоритмов ГСА и ЛСА.

Микропрограммные автоматы с жесткой и программируемой логикой. 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 (ЛЭ) 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 ЛЭ 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 процесс 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.

1.3. Способы задания цифровых автоматов

Можно выделить два класса языков для описания функционирования цифровых автоматов: *начальные языки* и *стандартные,* или *автоматные языки.*

Начальные языки задают функцию переходов и функцию выходов в неявном виде. Поведение автомата описывается в терминах входных и выходных последовательностей, реализуемых оператором, или последовательностей управляющих сигналов, воздействующих на управляющий автомат [8].

Для описания функционирования абстрактных ЦА на начальном языке можно использовать:

- язык регулярных выражений алгебры событий;
- язык исчисления предикатов;
- язык логических схем алгоритмов (ЛСА);
- язык граф-схем алгоритмов (ГСА).

Язык ГСА совместно с языком ЛСА называют одним общим термином – язык операторных схем алгоритмов (ОСА). На практике наиболее часто используется язык ГСА.

Начальные языки и их использование для описания поведения цифровых автоматов будут рассмотрены ниже. В этой главе рассмотрим более известные стандартные, или автоматные языки.

1.4. Задание автоматов на стандартных языках

Стандартные, или автоматные языки задают функции переходов выходов в явном виде. К ним относятся таблицы, графы, матрицы переходов и выходов и их аналитическая интерпретация СКУ и СВФ. Для того чтобы задать автомат, необходимо описать все компоненты вектора

$S = <A, Z, W, \delta, \lambda, a_1>$.

Табличный способ задания автомата Мили

Автомат Мили $S = <A, Z, W, \delta, \lambda, a_1>$, где $A = \{a_1, a_2, a_3, a_4\}$; $Z = \{z_1, z_2\}$; $W = \{w_1, w_2, w_3, w_4, w_5\}$, описывается с помощью двух таблиц: таблицы переходов и таблицы выходов. Таблица переходов задает функцию переходов δ (табл. 1.1), таблица выходов задает функцию выходов λ (табл. 1.2). Каждому столбцу (табл. 1.1 и табл. 1.2) поставлено в соответствие одно состояние из множества A, каждой строке – один входной сигнал из множества Z.

На пересечении столбца a_m и строки z_f в табл. 1.1 записывается состояние a_s, в которое должен перейти автомат из состояния a_m, под действием входного сигнала z_f, т.е. $a_s = \delta\ (a_m, z_f)$. На пересечении столбца a_m и строки z_f в табл. 1.2 записывается выходной сигнал w_g, выдаваемый автоматом в состоянии a_m при поступлении на его вход сигнала z_f, т.е. $w_g = \lambda\ (a_m, z_f)$.

Таблица 1.1. Таблица переходов автомата Мили

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

Таблица 1.2. Таблица выходов автомата Мили

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

Автомат Мили может быть задан также одной совмещенной таблицей переходов и выходов (табл. 1.3), в которой каждый элемент a_s / w_g, записанный на пересечении столбца a_m и строки z_f, определяется следующим образом: $a_s = \delta\ (a_m, z_f)$; $w_g = \lambda\ (a_m, z_f)$.

Таблица 1.3. Совмещенная таблица переходов и выходов автомата Мили

A	a_1	a_2	a_3	a_4
z_1	a_2/w_1	a_2/w_1	a_1/w_2	a_1/w_4
z_2	a_4/w_5	a_3/w_3	a_4/w_4	a_3/w_5

Табличный способ задания автомата Мура

Автомат Мура $S = <A, Z, W, \delta, \lambda, a_1>$, где $A = \{a_1, a_2, a_3, a_4\}$, $Z = \{z_1, z_2\}$, $W = \{w_1, w_2, w_3\}$, задается одной отмеченной таблицей переходов (табл. 1.4), в которой каждому столбцу приписаны не только состояния a_m, но еще и выходной сигнал w_g, соответствующий этому состоянию, где $w_g = \lambda(a_m)$.

Таблица 1.4. Отмеченная таблица переходов и выходов автомата Мура

W	w_3	w_2	w_3	w_1
A	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_1	a_4
z_2	a_2	a_4	a_4	a_1

Одним из преимуществ табличного способа задания является то, что любая таблица переходов и выходов задает конечный автомат.

При задании автоматов с большим числом состояний и переходов наглядность теряется, поэтому оказывается предпочтительным задавать этот граф в виде списка – таблицы переходов.

Прямая таблица переходов – таблица, в которой последовательно перечисляются все переходы сначала из первого состояния, затем из второго и т.д. Табл. 1.5 – прямая таблица переходов автомата Мили. В ряде случаев оказывается удобным пользоваться **обратной таблицей переходов**, в которой столбцы обозначены точно так же, но сначала записываются все переходы в первое состояние, затем во второе и т.д. Табл. 1.6 – обратная таблица переходов автомата Мили. Табл. 1.5 и табл. 1.6 построены по совмещенной таблице переходов (табл. 1.3).

Таблица 1.5. Прямая таблица переходов автомата Мили

$a_m(t)$	$z_f(t)$	$a_s(t+1)$	$w_g(t)$
a_1	z_1	a_2	w_1
	z_2	a_4	w_5
a_2	z_1	a_2	w_1
	z_2	a_3	w_3
a_3	z_1	a_1	w_2
	z_2	a_4	w_4
a_4	z_1	a_1	w_4
	z_2	a_3	w_5

Таблица 1.6. Обратная таблица переходов автомата Мили

$a_m(t)$	$z_f(t)$	$a_s(t+1)$	$w_g(t)$
a_3	z_1	a_1	w_2
a_4	z_1		w_4
a_1	z_1	a_2	w_1
a_2	z_1		w_1
a_2	z_2	a_3	w_3
a_4	z_2		w_5
a_1	z_2	a_4	w_5
a_3	z_2		w_4

Прямая таблица переходов автомата Мура строится так же, как и для автомата Мили. Разница лишь в том, что выходной сигнал $w_g(t)$ приписывается состоянию автомата $a_m(t)$ (табл. 1.7. Вариант 1), либо выходной сигнал $w_g(t+1)$ приписывается состоянию автомата $a_s(t+1)$ (табл. 1.8. Вариант 2).

Таблица 1.7. Прямая таблица переходов автомата Мура. Вариант 1

$a_m(t)$	$w_g(t)$	$z_f(t)$	$a_s(t+1)$
a_1	w_3	z_1	a_1
		z_2	a_2
a_2	w_2	z_1	a_3
		z_2	a_4
a_3	w_3	z_1	a_1
		z_2	a_4
a_4	w_1	z_1	a_4
		z_2	a_1

Таблица 1.8. Прямая таблица переходов автомата Мура. Вариант 2

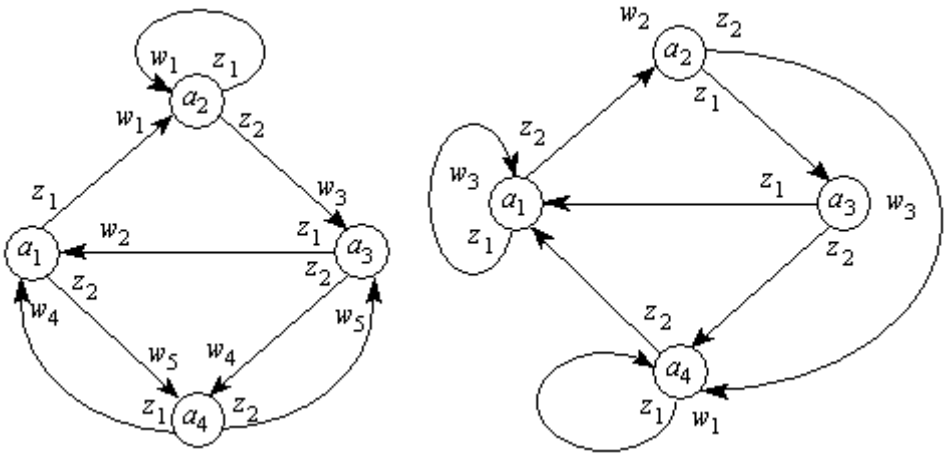
$a_m(t)$	$z_f(t)$	$a_s(t+1)$	$w_g(t+1)$
a_1	z_1	a_1	w_2
	z_1	a_2	w_4
a_2	z_1	a_3	w_1
	z_1	a_4	w_1
a_3	z_2	a_1	w_3
	z_2	a_4	w_5
a_4	z_2	a_4	w_5
	z_2	a_1	w_4

Обратная таблица переходов автомата Мура строится так же, как и для автомата Мили. Разница лишь в том, что выходной сигнал $w_g(t+1)$ приписывается состоянию автомата $a_s(t+1)$ (табл. 1.9). Табл. 1.7–1.9 построены по отмеченной таблице переходов (табл. 1.4).

Таблица 1.9. Обратная таблица
переходов автомата Мура

$a_m(t)$	$z_f(t)$	$a_s(t+1)$	$w_q(t+1)$
a_1	z_1	a_1	w_3
a_3	z_1		
a_4	z_2		
a_1	z_2	a_2	w_2
a_2	z_1	a_3	w_3
a_2	z_2	a_4	w_1
a_3	z_2		
a_4	z_1		

Граф автомата – это ориентированный граф, вершины которого соответствуют состояниям, а дуги – переходам между ними. Дуга, направленная из вершины a_m в вершину a_s, задает переход в автомате из состояния a_m в состояние a_s. В начале этой дуги записывается входной сигнал $z_f \in Z$, вызывающий данный переход, – $a_s = \delta(a_m, z_f)$. Для графа автомата Мили выходной сигнал $w_g \in W$, формируемый на переходе, записывается в конце дуги, а для автомата Мура – рядом с вершиной, отмеченной состоянием a_m, в котором он формируется. Если переход в автомате из состояния a_m в состояние a_s производится под действием нескольких входных сигналов, то дуге графа, направленной из a_m в a_s, приписываются все эти входные и соответствующие выходные сигналы. Графы автоматов Мили и Мура, построенные по табл. 1.3 и табл. 1.4, приведены на рис. 1.4,а, б, соответственно.



а) *б)*

Рис. 1.4. Графы автоматов: а – Мили; б – Мура

При любом способе задания должны соблюдаться условия **однозначности** и **полноты переходов**:

- **условие однозначности** (детерминированности) означает, что для любой пары $a_m z_f$ задано единственное состояние перехода a_s и единственный выходной сигнал w_g , выдаваемый на переходе.

- **условие полной определенности** означает, что для всех возможных пар $a_m z_f$ всегда указаны состояние и выходной сигнал.

Автомат называется **неполностью определенным**, или **частичным**, если либо функция переходов δ определена не на всех парах ($a_m z_f$)$\in A$ x Z, либо функция выходов λ определена не на всех указанных парах в случае автомата Мили и на множестве не всех внутренних состояний для автомата Мура. Для частичных автоматов Мили и Мура в рассмотренных таблицах на месте неопределенных состояний и выходных сигналов ставится прочерк.

Применительно к графу **условия однозначности** и **полной определенности** будут заключены в следующем:

- не существует двух ребер с одинаковыми входными пометками, выходящих из одной и той же вершины;

- для всякой вершины a_m и для всякого входного сигнала z_f имеется такое ребро, помеченное символом z_f , которое выходит из a_m.

В некоторых случаях для задания автомата используются **матрицы переходов** и **выходов**, которые представляют собой таблицу с двумя входами. Строки и столбцы таблицы отмечены состояниями. Если существует переход из a_m под действием z_f в a_s с выдачей w_g , то на пересечении строки a_m и столбца a_s записывается пара z_f, w_g. Ясно, что не всякая матрица задает автомат. Как граф и таблица переходов и выходов она должна удовлетворять условиям однозначности и полноты переходов.

Аналитической интерпретацией таблиц переходов и выходов или графов автоматов являются [8] **системы канонических уравнений (СКУ)** и **системы выходных функций (СВФ)**.

СКУ – определяет функции переходов автоматов, которые для автоматов Мили и Мура одинаковы (см. выражения (1.1)–(1.2). Для записи каждое состояние ЦА интерпретируется как событие, соответствующее множеству переходов в это состояние:

$$a_s(t+1) = \bigvee_{f,m} z_f(t) \,\&\, a_m(t). \qquad (1.7)$$

СВФ – определяет функции выходов ЦА.

Для автомата Мили:

$$w_g(t) = \bigvee_{f,m} z_f(t) \,\&\, a_m(t). \qquad (1.8)$$

Для автомата Мура:

$$w_g(t) = \bigvee_{m} a_m(t). \qquad (1.9)$$

Для сокращения записи СКУ и СВФ будем в дальнейшем везде, где это возможно, опускать знаки конъюнкции и времени t в правой части уравнений типа (1.7)–(1.9).

Для автомата Мили, граф которого приведен на рис. 1.4,а, а таблицы переходов в табл. 1.3, 1.5, 1.6, запишем СКУ и СВФ (выражения (1.10) и (1.11), соответственно):

$$\begin{aligned} a_1(t+1) &= z_1 a_3 \vee z_1 a_4; \\ a_2(t+1) &= z_1 a_1 \vee z_1 a_2; \\ a_3(t+1) &= z_2 a_2 \vee z_2 a_4; \\ a_4(t+1) &= z_2 a_1 \vee z_2 a_3; \end{aligned} \qquad (1.10)$$

$$\begin{aligned} w_1(t) &= z_1 a_1 \vee z_1 a_2; \\ w_2(t) &= z_1 a_3; \\ w_3(t) &= z_2 a_2; \\ w_4(t) &= z_1 a_4 \vee z_2 a_3; \\ w_5(t) &= z_2 a_1 \vee z_2 a_4. \end{aligned} \qquad (1.11)$$

Запишем СКУ и СВФ для автомата Мура, граф которого приведен на рис. 1.4,б, а таблицы переходов в табл. 1.4, 1.6 и 1.7–1.9, (выражения (1.12) и (1.13), соответственно):

$$\begin{aligned} a_1(t+1) &= z_1 a_1 \vee z_1 \vee z_2 a_4; \\ a_2(t+1) &= z_2 a_1; \\ a_3(t+1) &= z_1 a_1; \\ a_4(t+1) &= z_1 a_4 \vee z_2 a_2 \vee z_2 a_3; \end{aligned} \qquad (1.12)$$

$$\begin{aligned} w_1(t) &= a_4; \\ w_2(t) &= a_2; \\ w_3(t) &= a_1 \vee a_3. \end{aligned} \qquad (1.13)$$

1.5. Связь между автоматами Мили и Мура

1.5.1. Преобразование автомата Мили в автомат Мура

Ограничение 1. В автомате Мили не должно быть переходящих состояний, т.е. состояний, в которых имеется хотя бы одна выходящая дуга и не имеется ни одной входящей дуги, так, как показано на рис. 1.5.

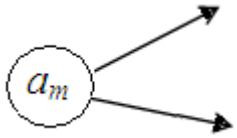
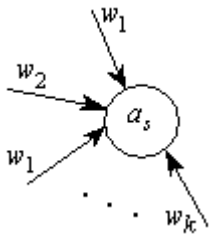


Рис. 1.5 Схема, поясняющая ограничение 1

Пусть задан автомат Мили $S = <A, Z, W, \delta, \lambda, a_1>$. Построим автомат Мура $S' = <A', Z', W', \delta', \lambda', a_1'>$, у которого $Z' = Z$; $W' = W$.

1. Для определения множества A' необходимо выполнить расщепление состояний автомата Мили, исходя из следующего [8]. Если автомат Мили, при переходе в некоторое состояние a_s может выдавать в разные моменты времени один из k выходных сигналов из алфавита W, то такое состояние a_s должно быть расщеплено на k состояний (рис. 1.6). То есть число элементов в множестве A_s равно числу различных выходных сигналов на дугах автомата S, входящих в состояние a_s.



$$A_s=\{(a_s,w_1), (a_s,w_2),..., (a_s,w_k)\}.$$

Рис. 1.6. Схема, поясняющая принцип построения множества A_s

Множество состояний автомата S' получим как объединение множеств A_s:

$$A' = \bigcup_{s=1}^{M} A_s,$$

где M – число состояний в автомате Мили S.

Такое расщепление состояний автомата Мили необходимо потому, что все состояния эквивалентного ему автомата Мура должны быть отмечены только одним выходным сигналом из алфавита W.

2. Для определения функции с каждым состоянием вида (a_s, w_g), представляющим собой пару, отождествим выходной сигнал, являющийся вторым элементом этой пары.

3. Функцию переходов δ' определим следующим образом. Если в автомате Мили S был переход $\delta\ (a_m, z_f) = a_s$ и при этом выдавался выходной сигнал $\lambda\ (a_m, z_f) = w_g$, то в автомате Мура S' (рис. 1.7) будет переход из множества состояний A_m, порождаемых a_m, в состояние (a_s, w_g) под действием того же входного сигнала z_f.

4. В качестве начального состояния автомата Мура a_1' можно взять любое состояние из множества A_1 ($a_1' \in A_1$), порождаемого начальным состоянием a_1

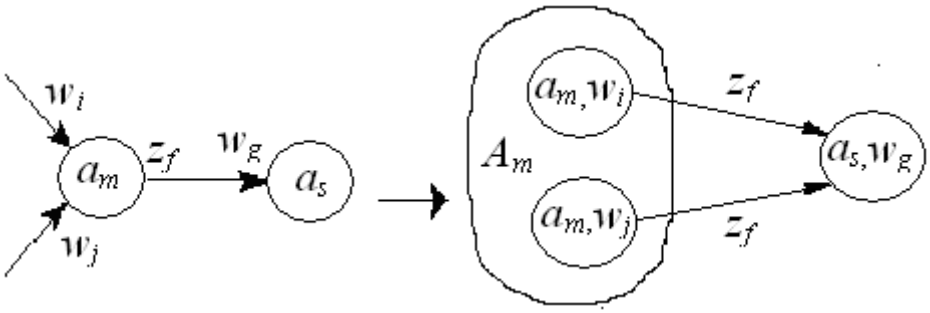


Рис. 1.7. Иллюстрация перехода от автомата Мили к автомату Мура

Пример 1.1. Преобразовать автомат Мили, изображенный на рис. 1.4,а, в автомат Мура.

Решение

Для автомата Мура:

$Z' = Z = \{z_1, z_2\}$;

$W' = W = \{w_1, w_2, w_3, w_4, w_5\}$.

1. Построим множество A'. Для этого найдем множество пар, порождаемых каждым состоянием автомата Мили S. Каждую пару обозначим символами b_1, b_2, ...:

$A_1 = \{(a_1, w_2),(a_1, w_4)\} = \{b_1, b_2\}$; $A_2 = \{(a_2, w_1)\} = \{b_3\}$;

$A_3 = \{(a_3, w_3),(a_3, w_5)\} = \{b_4, b_5\}$; $A_4 = \{(a_4, w_4),(a_4, w_5)\} = \{b_6, b_7\}$.

$A' = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$.

2. Для определения функции $?'$ с каждым состоянием вида (a_s, w_g), представляющим собой пару, отождествим выходной сигнал, являющийся вторым элементом этой пары:

$\lambda'(b_1) = w_2$; $\lambda'(b_2) = \lambda'(b_6) = w_4$; $\lambda'(b_3) = w_1$;

$\lambda'(b_4) = w_3$; $\lambda'(b_5) = \lambda'(b_7) = w_5$.

3. Поясним построение функции δ'.

Так как в автомате Мили S есть переход из состояния a_1, под действием сигнала z_1, в состояние a_2 с выдачей w_1, то из множества состояний $A_1 = \{b_1, b_2\}$, порождаемых a_1, в автомате S' должен быть переход в состояние $(a_2, w_1) = b_3$ под действием сигнала z_1. Аналогично из состояний множества $A_1 = \{b_1, b_2\}$ должен быть переход в состояние $(a_4, w_5) = b_7$ под действием сигнала z_2 и т. д.

4. В качестве начального состояния можно выбрать одно из состояний, порождаемых a_1, например b_1.

На рис. 1.8 приведен граф автомата Мура S', эквивалентного автомату Мили S. Табл. 1.10 – отмеченная таблица переходов автомата Мура S'.

Таблица 1.10. Отмеченная таблица переходов S'

W'	w_2	w_4	w_1	w_3	w_5	w_4	w_5
A'	b_1	b_2	b_3	b_4	b_5	b_6	b_7
z_1	b_3	b_3	b_3	b_1	b_1	b_2	b_2
z_2	b_7	b_7	b_4	b_6	b_6	b_5	b_5

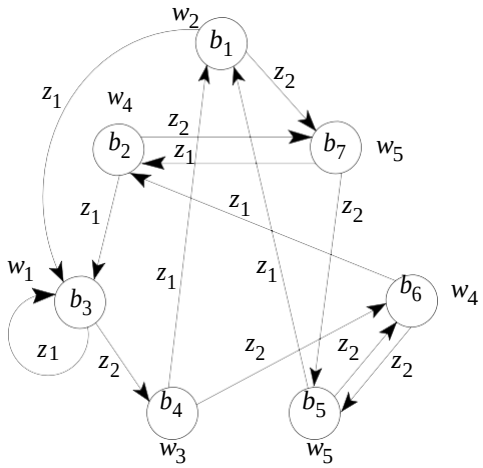


Рис. 1.8. Граф автомата Мура S'

Запишем отмеченную СКУ (1.14) и СВФ (1.15) для автомата Мура S', эквивалентного автомата Мили S.

$$
\begin{aligned}
&b_1(t+1) = z_1(b_4 \vee b_5); \qquad && b_5(t+1) = z_2(b_6 \vee b_7);\\
&b_2(t+1) = z_1(b_6 \vee b_7); \qquad && b_6(t+1) = z_2(b_4 \vee b_5);\\
&b_3(t+1) = z_1(b_1 \vee b_2 \vee b_3); \qquad && b_7(t+1) = z_2(b_1 \vee b_2).\\
&b_4(t+1) = z_2(b_1 \vee b_2).
\end{aligned}
\qquad (1.14)
$$

$$w_1(t) = b_3; \quad w_2(t) = b_1; \quad w_3(t) = b_4; \quad w_4(t) = b_2 \vee b_6; \quad w_5(t) = b_5 \vee b_7. \qquad (1.15)$$

1.5.2. Преобразование автомата Мура в автомат Мили

Пусть задан автомат Мура $S = <A, Z, W, \delta, \lambda, a_1>$, построим автомат Мили $S' = <A', Z', W', \delta', \lambda', a_1'>$, у которого $Z' = Z$; $W' = W$; $A' = A$; $\delta' = \delta$; $a_1' = a_1$.

При графическом способе задания этот переход иллюстрируется рис. 1.9: выходной сигнал (w_g), записанный рядом с вершиной (a_s), переносится на все дуги, входящие в эту вершину.

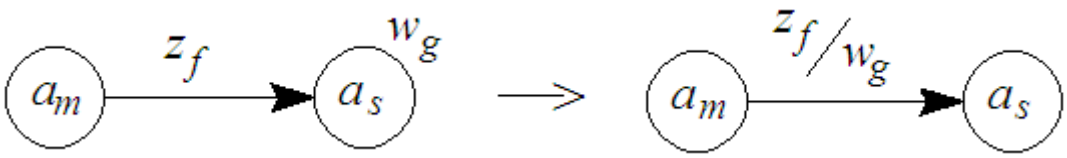


Рис. 1.9. Иллюстрация перехода от автомата Мура к автомату Мили

При табличном методе задания выполняется преобразование отмеченной таблицы переходов автомата Мура. При этом таблица переходов эквивалентного автомата Мили совпадает с таблицей переходов автомата Мура. Для получения таблицы выходов автомата Мили необходимо в полученной таблице переходов заменить состояние a_s, стоящее на пересечении столбца a_m и строки z_f, на символ w_g, отмечающий столбец a_m в отмеченной таблице переходов исходного автомата Мура.

Пример 1.2. Преобразовать автомат Мура, изображенный на рис. 1.4,б, в автомат Мили.

Решение

Для автомата Мили:

$$Z' = Z = \{z_1, z_2\}; \; W' = W = \{w_1, w_2, w_3\}; \; A' = A = \{a_1, a_2, a_3, a_4\};$$
$$\delta' = \delta; \; a_1^7 = a_1.$$

На рис. 1.10 приведен граф автомата Мура S, эквивалентного автомату Мили S'. Табл. 1.11 – совмещенная таблица переходов и выходов автомата Мили S'.

Для получения совмещенной таблицы переходов автомата Мили вместе с состояниями перехода a_s записываем отмечающие их выходные сигналы w_g.

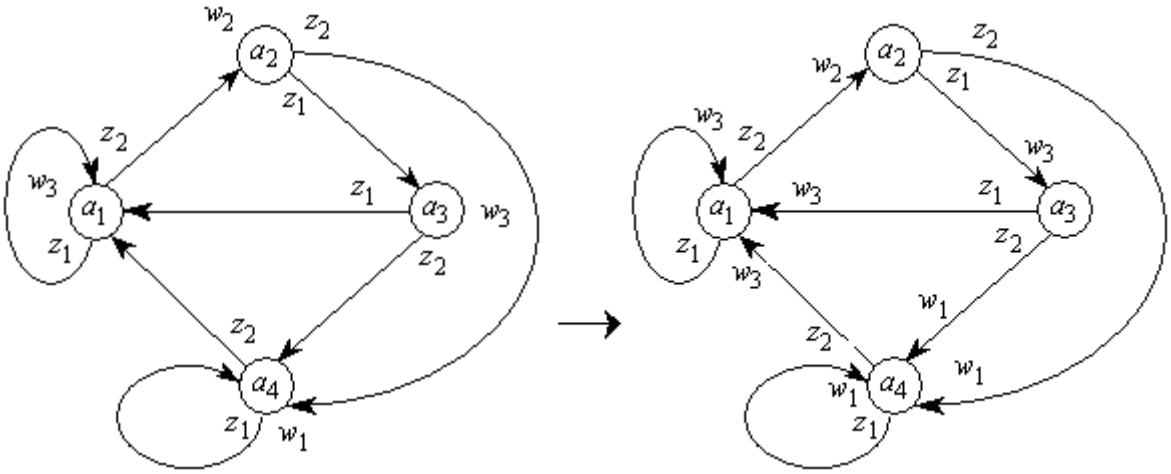


Рис. 1.10. Иллюстрация процесса перехода автомата Мили S' к эквивалентного автомату Мура S

Таблица 1.4.

Отмеченная таблица переходов и выходов автомата Мура

W	w_3	w_2	w_3	w_1
A	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_1	a_4
z_2	a_2	a_4	a_4	a_1

Таблица 1.11.

Совмещенная таблица переходов и выходов автомата Мили

A	a_1	a_2	a_3	a_4
z_1	$a_1/\ w_3$	$a_3/\ w_3$	$a_1/\ w_3$	$a_4/\ w_1$
z_2	$a_2/\ w_2$	$a_4/\ w_1$	$a_4/\ w_1$	$a_1/\ w_3$

1.5.3. Определение реакции автомата

Реакцией автомата называется последовательность выходных сигналов, полученная под воздействием некоторой последовательности входных сигналов. То есть реакция автомата – выходное слово, полученное как отображение конкретного входного слова.

Рассмотрим формирование реакции для автомата Мили из примера 1.2, совмещенная таблица переходов которого приведена в табл. 1.11. Пусть на вход автомата подается последовательность входных сигналов $Z = z_2, z_1, z_2, z_1, z_2$. Исходное состояние автомата – a_1.

В момент времени t_1 на вход автомата поступает сигнал z_2, который переводит автомат из состояния a_1 в состояние a_2. При этом на выходе автомата в тот же момент дискретного времени t_1 формируется выходной сигнал w_2. В следующий момент времени t_2 на вход автомата поступает сигнал z_1, который переводит автомат из состояния a_2 в состояние a_3, и на выходе автомата в тот же момент дискретного времени t_2 формируется выходной сигнал w_3. Аналогично сформированы и ос-

тальные выходные сигналы. В результате как отображение входного слова $Z = z_2, z_1, z_2, z_1, z_2$ получено выходное слово $\boldsymbol{W = w_2, w_3, w_1, w_1, w_3}$, которое и является реакцией автомата (рис. 1.11).

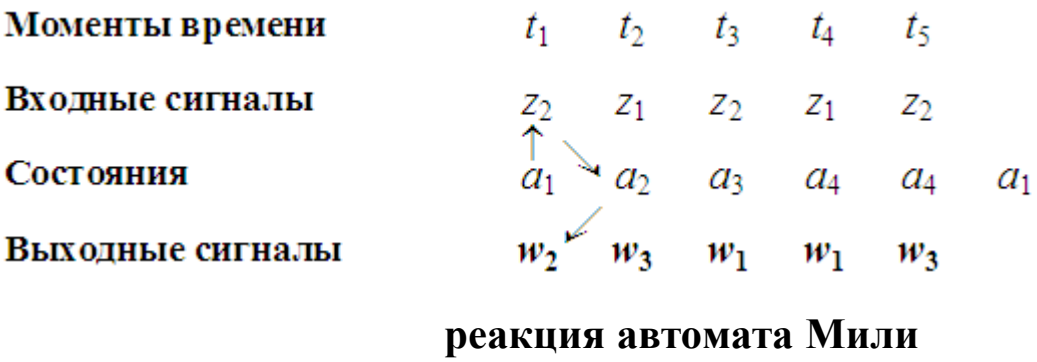


Рис. 1.11. Схема получения реакции автомата Мили

Рассмотрим формирование реакции для автомата Мура из примера 1.2, отмеченная таблица переходов которого приведена в табл. 1.4. Пусть на вход автомата подается та же последовательность входных сигналов, что и для автомата Мили $Z = z_2, z_1, z_2, z_1, z_2$. Исходное состояние автомата – a_1 – отмечено выходным сигналом w_3.

Для автомата Мура формирование реакции происходит аналогично, с той лишь разницей, что выходной сигнал в автомате Мура зависит только от состояния автомата. Поэтому в момент времени t_1 выходной сигнал w_3 на выходе автомата определяется только исходным состоянием a_1 и не зависит от входного сигнала z_2 в текущий момент времени. Выходной сигнал w_2 формируется на выходе автомата в момент времени t_2, когда автомат из состояния a_1 перейдет в состояние a_2 и так далее.

В результате полученное выходное слово $\boldsymbol{W = w_2, w_3, w_1, w_1, w_3}$, которое и является реакцией автомата Мура, отстает на один такт от входного слова $Z = z_2, z_1, z_2, z_1, z_2$ (рис. 1.12).

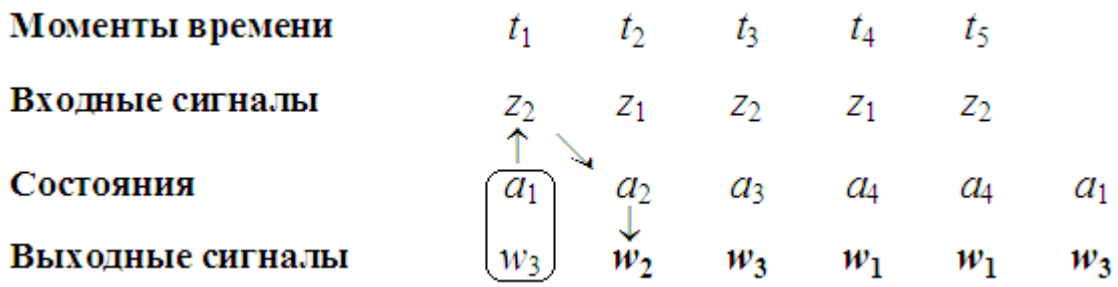


Рис. 1.12. Схема получения реакции автомата Мура

Формирование реакции было рассмотрено для автоматов Мили и Мура из примера 1.2. Эти автоматы являются эквивалентными, что под-

утверждает получение одинаковой реакции $W = w_2, w_3, w_1, w_1, w_3$ на входное слово $Z = z_2, z_1, z_2, z_1, z_2$.

1.6. Минимизация полностью определенных автоматов

Решение задачи минимизации состоит в разбиении всех состояний исходного абстрактного автомата на попарно непересекающиеся классы эквивалентных состояний и замене каждого класса эквивалентности одним состоянием. Получающийся в результате минимизации автомат имеет столько же состояний, на сколько классов эквивалентности разбиваются состояния исходного автомата [6].

Состояния a_m и a_s являются эквивалентными ($a_m \sim a_s$), если совпадают реакции для всевозможных входных слов ξ:

$\lambda(a_m, \xi) = \lambda(a_s, \xi)$.

Состояния a_m и a_s являются k-эквивалентными ($a_m \overset{k}{\sim} a_s$), если $\lambda(a_m, \xi) = \lambda(a_s, \xi)$ для всевозможных слов длины k.

При минимизации полностью определённых автоматов Мили вводится понятие одноэквивалентности состояний. **Одноэквивалентными будут состояния с одинаковыми столбцами в таблице выходов.**

Для автомата Мура вводится понятие 0-эквивалентности состояний и разбиение множества состояний на 0-эквивалентные классы. **0-эквивалентными являются одинаково отмеченные состояния.**

Алгоритм минимизации автомата $S = \langle A, Z, W, \delta, \lambda, a_1 \rangle$ состоит из следующих шагов:

1. Находим разбиение состояний на непересекающиеся классы эквивалентных состояний π.

2. В каждом классе эквивалентности разбиения π выбирается по одному состоянию, в результате чего получаем множество A' состояний минимального автомата $S' = \{A', Z, W, \delta', \lambda', a_1'\}$, эквивалентного автомату S.

3. Для определения функции переходов δ' и функции выходов λ' автомата S' в таблицах переходов и выходов вычёркиваются столбцы, соответствующие не вошедшим в A' состояниям. В оставшихся столбцах не вошедшие в множество A' состояния заменяются на эквивалентные.

4. В качестве начального состояния a_1' выбирается состояние, эквивалентное состоянию a_1. В частности, удобно за a_1' принимать само

состояние a_1.

1.6.1. Метод Ауфенкампа и Хона

Минимизация автомата Мили. Выполним минимизацию автомата Мили $S = <A, Z, W, \delta, \lambda, a_1>$.

1 шаг. Для нахождения эквивалентного разбиения состояний π находим последовательные разбиения π_1, π_2, ..., π_k, π_{k+1} множества A на классы одно-, двух-, K-, K+1- эквивалентных состояний до тех пор, пока в каком-то (K+1) шаге не окажется, что $\pi_k = \pi_{k+1}$.

Выполнение отдельных шагов минимизации автомата Мили рассмотрим на примере.

Пример 1.3 Минимизировать полностью определённый автомат Мили S_1, заданный таблицами переходов и выходов (табл. 1.12 и 1.13).

Таблица 1.12. Таблица переходов неминимального автомата Мили

A	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_4	a_3	a_4	a_5	a_6
z_2	a_5	a_6	a_5	a_6	a_1	a_2

Таблица 1.13. Таблица выходов неминимального автомата Мили

A	a_1	a_2	a_3	a_4	a_5	a_6
z_1	w_1	w_1	w_1	w_1	w_1	w_1
z_2	w_1	w_1	w_2	w_2	w_1	w_1

Решение

1. По таблице выходов (табл. 1.13) находим разбиение π_1 на классы 1-эквивалентных состояний, объединяя в 1-эквивалентные классы одинаковые столбцы в таблице выходов:

$$\pi_1 = \{B_1, B_2\} = \{\{a_1, a_2, a_5, a_6\}, \{a_3, a_4\}\} = \{B_1, B_2\} = \{\overline{1.2.5.6}, \ \overline{3.4}\}.$$

Строим таблицу разбиения π_1 (табл. 1.14), заменяя состояния в таблице переходов исходного автомата (табл. 1.12) соответствующими классами одноэквивалентности.

По табл. 1.14 получим разбиение π_2 на классы 2-эквивалентных состояний (табл. 1.15):

$\pi_2 = \{C_1, C_2, C_3\} = \{\overline{1.2}, \overline{5.6}, \overline{3.4}\}$.

Таблица 1.14. Разбиение π_1 состояний автомата S_1

1	B_1				B_2	
A	a_1	a_2	a_5	a_6	a_3	a_4
z_1	B_2	B_2	B_1	B_1	B_2	B_2
z_2	B_1	B_1	B_1	B_1	B_1	B_1

Таблица 1.15. Разбиение π_2 состояний автомата S_1

2	C_1		C_2		C_3	
A	a_1	a_2	a_5	a_6	a_3	a_4
z_1	C_3	C_3	C_2	C_2	C_3	C_3
z_2	C_2	C_2	C_1	C_1	C_2	C_2

Разбиение π_3 получаем аналогично: $\pi_3 = \{D_1, D_2, D_3\} = \{\overline{1.2}, \overline{5.6}, \overline{3.4}\}$, оно полностью совпадает с π_2. Процедура завершена. Разбиение $\pi_3 = \pi_2 = \pi$ есть разбиение множества состояний автомата Мили S_1 на классы эквивалентных между собой состояний.

2 шаг. Для нахождения множество A' состояний минимального автомата $S' = \{A', Z, W, \delta', \lambda', a_1'\}$ из каждого класса эквивалентности произвольно выбираем по одному состоянию: $A' = \{a_1, a_3, a_6\}$.

3 шаг. Определяем функции переходов δ' и функции выходов λ' автомата S'. Для этого в таблицах переходов и выходов вычёркиваем столбцы, соответствующие не вошедшим в A' состояниям. В оставшихся столбцах не вошедшие в множество A' состояния заменяются на эквивалентные. Получаем таблицы переходов и выходов минимального автомата S' (табл. 1.16, 1.17).

Таблица 1.16. Получение таблицы переходов минимального автомата Мили

A	a_1	~~a_2~~	a_3	~~a_4~~	~~a_5~~	a_6
z_1	a_3	~~a_4~~	a_3	~~a_4~~	~~a_5~~	a_6
z_2	~~a_5~~ a_6	a_6	~~a_5~~ a_6	a_6	a_1	~~a_2~~ a_1

→

A	a_1	a_3	a_6
z_1	a_3	a_3	a_6
z_2	a_6	a_6	a_1

Таблица 1.17. Получение таблицы выходов минимального автомата Мили

A	a_1	~~a_2~~	a_3	~~a_4~~	~~a_5~~	a_6
z_1	w_1	~~w_1~~	w_1	~~w_1~~	~~w_1~~	w_1
z_2	w_1	~~w_1~~	w_2	w_2	~~w_1~~	w_1

→

A	a_1	a_3	a_6
z_1	w_1	w_1	w_1
z_2	w_1	w_2	w_1

4 шаг. $a_1' = a_1$.

Минимизация автомата Мура. При минимизации полностью определённых автоматов Мура на первом шаге находится разбиение π_0 – разбиение множества состояний на 0-эквивалентные классы. 0-эквивалентными являются одинаково отмеченные состояния. Если два состояния автомата Мура 0-эквивалентны и под действием одинаковых входных сигналов попадают в 0-эквивалентные состояния, то они называются 1-эквивалентными.

Все дальнейшие классы эквивалентности для автомата Мура определяются аналогично рассмотренному выше для автомата Мили.

Пример 1.4. Минимизировать полностью определённый автомат Мура S_2, заданный отмеченной таблицей переходов (табл. 1.18).

Решение

1 шаг. По табл. 1.18 находим разбиение π_0 на классы 0-эквивалентных состояний, находя одинаково отмеченные состояния:

$$\pi_0 = \{A_1, A_2, A_3\} = \{\overline{1.2.8}, \quad \overline{6.9.10.11.12}, \quad \overline{3.4.5.7}\}.$$

Таблица 1.18. Отмеченная таблица переходов неминимального автомата Мура S_2

W	w_1	w_1	w_3	w_3	w_3	w_2	w_3	w_1	w_2	w_2	w_2	w_2
A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

Строим таблицу разбиения π_0 (табл. 1.19), заменяя состояния в отмеченной таблице переходов (табл. 1.18) соответствующими классами 0-эквивалентности.

По табл. 1.19 получим разбиение π_1 на классы 1-эквивалентных состояний:

$$\pi_1 = \{B_1, B_2, B_3, B_4\} = \{\overline{1.2.8}, \quad \overline{6.9.11}, \quad \overline{10.12}, \quad \overline{3.4.5.7}\}.$$

Таблица 1.19. Разбиение π_0 состояний автомата S_2

π_0	A_1			A_2					A_3			
A	a_1	a_2	a_8	a_6	a_9	a_{10}	a_{11}	a_{12}	a_3	a_4	a_5	a_7
z_1	A_2	A_2	A_2	A_3	A_3	A_1	A_3	A_1	A_3	A_3	A_3	A_3
z_2	A_3	A_3	A_3	A_2	A_2	A_1	A_2	A_1	A_2	A_2	A_2	A_2

Строим таблицу разбиения π_1 (табл. 1.20), заменяя состояния в отмеченной таблице переходов (табл. 1.18) соответствующими классами 1-эквивалентности.

Таблица 1.20. Разбиение π_1 состояний автомата S_2

π_1	B_1			B_2			B_3		B_4			
A	a_1	a_2	a_8	a_6	a_9	a_{11}	a_{10}	a_{12}	a_3	a_4	a_5	a_7
z_1	B_3	B_3	B_3	B_4	B_4	B_4	B_1	B_1	B_4	B_4	B_4	B_4
z_2	B_4	B_4	B_4	B_2	B_2	B_2	B_1	B_1	B_2	B_2	B_2	B_2

Из табл. 1.20 видно, что $\pi_2 = \pi_1 = \pi$. Это означает, что поиск классов эквивалентности завершён.

2 шаг. Из каждого класса эквивалентности произвольно выбираем по одному элементу: $A' = \{a_1, a_6, a_{10}, a_3\}$.

3 шаг. Строим отмеченную таблицу переходов минимального автомата S_2 (табл. 1.21).

4 шаг. $a'_1 = a_1$.

Таблица 1.21. Получение отмеченной таблицы переходов минимального автомата Мура S_2

W	w_1	w_1	w_3	w_3	w_3	w_2	w_3	w_1	w_2	w_2	w_2	w_2
A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

W'	w_1	w_3	w_2	w_2
A'	a_1	a_3	a_6	a_{10}
z_1	a_{10}	~~a_2~~ a_3	~~a_2~~ a_3	a_1
z_2	~~a_5~~ a_3	a_6	~~a_{11}~~ a_6	~~a_8~~ a_1

→

W'	w_1	w_3	w_2	w_2
A'	a_1	a_3	a_6	a_{10}
z_1	a_{10}	a_3	a_3	a_1
z_2	a_3	a_6	a_6	a_1

1.6.2. Минимизация с помощью треугольной таблицы

Рассмотрим алгоритм минимизации состояний автоматов с помощью треугольной таблицы, предложенный М. Поллом и С. Ангером [6]. Минимизируем автомат Мили S_3 (табл. 1.22).

Таблица 1.22. Совмещенная таблица переходов и выходов неминимального автомата Мили S_3

A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_3 / w_1	a_8 / w_2	a_7 / w_1	a_2 / w_2	a_3 / w_1	a_2 / w_2	a_2 / w_2	a_6 / w_1
z_2	a_2 / w_2	a_4 / w_1	a_5 / w_2	a_4 / w_1	a_2 / w_2	a_7 / w_1	a_6 / w_1	a_5 / w_2

По табл. 1.22 составляется треугольная таблица (табл. 1.23), столбцы и строки которой сопоставляются с состояниями автомата. Для упрощения записи в этой таблице вместо a_i будем писать i.

В треугольной таблице на пересечении строки m и столбца s ставятся:

– + (крест), если в столбцах a_m и a_s таблицы выходов стоят разные выходные сигналы, например a_1 и a_2;

– множество всех пар состояний, следующих за парой (a_m, a_s) и отличных от неё, эквивалентность которых необходима для эквивалентности пары (a_m, a_s). Например, для эквивалентности (a_1, a_8) необходима эквивалентность (a_3, a_6) и (a_2, a_5), так как (a_3, a_8) и (a_2, a_5) следуют за множеством (a_1, a_8) по входам z_1 и z_2, соответственно (см. табл. 1.22).

– V (птичка), если за парой (a_m, a_s) не следуют пары, отличные от (a_m, a_s). В нашем примере это пара (a_6, a_7).

Таблица 1.23. Треугольная таблица для автомата S_3

2	+						
3	3.7 2.5	+					
4	+	2.8	+				
5	V	+	3.7 2.5	+			
6	+	2.8 4.7	+	4.7	+		
7	+	2.8 4.6	+	4.6	+	V	
8	3.6 2.5	+	6.7	+	3.6 2.5	+	+
	1	2	3	4	5	6	7

Для нахождения неэквивалентных пар состояний треугольная таблица (табл. 1.23) просматривается по столбцам. Находится первая клетка, отмеченная крестом. Пусть это будет клетка (i, j). Тогда во всех клетках, где есть пара (i, j), ставится крест, а уже рассмотренная клетка (i, j) отмечается вторым крестом. Эта процедура проводится для всех клеток, отмеченных крестом, включая новые, и заканчивается, когда таких клеток не останется. Очевидно, что в этом случае клетки без крестов соответствуют эквивалентным парам состояний, а клетки с крестами – неэквивалентным.

В результате применения этой процедуры к табл. 1.23 получим таблицу эквивалентных пар для автомата S_3 (табл. 1.24). Из этой таблицы видно, что: 1~5, 3~8, 4~6, 4~7, 6~7.

Таблица 1.24. Таблица эквивалентных пар для автомата S_3

2	++						
3	3.7+ 2.5+	++					
4	++	2.8++	++				
5	V	++	3.7+ 2.5+	++			
6	++	2.8+ 4.7+	++	4.7	++		
7	++	2.8+ 4.6+	++	4.6	++	V	
8	3.6+ 2.5+	++	6.7	++	3.6+ 2.5+	++	++
	1	2	3	4	5	6	7

Для нахождения разбиения π воспользуемся способом, предложенным С. Ангером [6]. Алгоритм сводится к следующему.

1. Начинаем составление списка классов эквивалентных состояний с крайнего правого столбца треугольной таблицы, имеющего, по крайней мере, одну клетку без крестов. В примере $\Phi = \{\overline{6.7}\}$.

2. Продвигаясь влево, выписываем для каждого столбца множества состояний $A_i \sim a_i$, т.е. множества состояний, соответствующих клеткам без крестов в *i*-м столбце таблицы.

В нашем примере $4 \sim \overline{6.7}$ (в 4-м столбце табл. 1.24 нет крестов в строках 6 и 7).

Каждое A_i пересекаем с каждым членом текущего списка Φ. Если такое пересечение содержит более одного состояния, то добавляем в список объединение $\{a_i\}$ с результатом пересечения:

$\overline{4} \cap \overline{6.7} = \overline{4.6.7}$; $\Phi = \{\overline{4.6}, \overline{4.6.7}\}$.

Далее проводится максимизация полученного множества Φ – удаление всех множеств, содержащихся в других множествах списка.

Приведём полностью результат применения второго шага ко всем столбцам:

$$\Phi = \{\overline{6.7}\},$$
$$\overline{\overline{4} \sim 6.7}, \quad \Phi = \{\overline{4.6.7}\},$$
$$\overline{\overline{3} \sim 8}, \quad \Phi = \{\overline{4.6.7},\ \overline{3.8}\},$$
$$\overline{\overline{1} \sim \overline{5}}, \quad \Phi = \{\overline{4.6.7},\ \overline{3.8},\ \overline{1.5}\}$$

3. В список Ф, полученный после второго шага, добавляются все одноэлементные множества, состоящие из состояний, не включенных ни в какие другие максимальные классы эквивалентности. В нашем примере – $\{\overline{2}\}$.

Таким образом, для автомата S_3 множество всех классов эквивалентности – разбиение π имеет вид:

$$\pi = \Phi = \{\overline{4.6.7},\ \overline{3.8},\ \overline{1.5},\ \overline{2}\}.$$

Построение минимального автомата S_3 по разбиению π проводится аналогично пунктам 2–4 алгоритма, рассмотренного выше (см. 1.5.1). Из каждого класса разбиения π выбираем по одному состоянию:

$A' = \{a_1, a_2, a_3, a_4\}$.

Строим совмещенную таблицу переходов и выходов минимального автомата S_3 (табл. 1.25).

Таблица 1.25. Совмещенная таблица переходов и выходов минимального автомата S_3

A	a_1	a_2	a_3	a_4
z_1	a_3 / w_1	a_3 / w_2	a_4 / w_1	a_2 / w_2
z_2	a_2 / w_2	a_4 / w_1	a_5 / w_2	a_4 / w_1

1.6.3. Минимизация ЦА на основе использования таблицы пар

Рассмотрим построение таблицы пар [8] и ее использование для минимизации числа состояний автомата Мили S_3 (табл. 1.22).

Таблица пар (табл. 1.26) строится непосредственно по таблице переходов. Первый основной столбец таблицы пар будет содержать все пары 1-эквивалентных состояний.

Для получения более компактной записи таблицу пар целесообразно разбивать на подтаблицы, число которых равно числу классов 1-эквивалентных состояний. Столбцы таблицы пар обозначаются входными сигналами.

Для нашего примера найдём разбиение π_1 на классы 1-эквивалентных состояний: $\pi_1 = \left\{ \overline{1.3.5.8,} \ \overline{2.4.6.7} \right\}$.

На пересечении строк и столбцов таблицы пар записываются пары состояний, которые являются первоприемниками по отношению к конкретному входному сигналу. На основании вышеизложенного, получена таблица пар для автомата Мили. В каждой клетке для упрощения записаны только индексы, определяющие номера состояний.

Таблица 1.26. Таблица пар состояний

Пары 1-эквивалентных состояний	Z_1	Z_2
1-3	3-7	2-5
1-5	3-3	2-2
1-8	3-6	2-5
3-5	7-3	5-2
3-8	7-6	5-5
5-8	3-6	2-5
2-4	8-2	4-4
2-6	8-2	4-7
2-7	8-2	4-6
4-6	2-2	4-7
4-7	2-2	4-6
6-7	2-2	7-6

Алгоритм нахождения эквивалентного разбиения состояний включает следующие этапы [8]:

1. Последовательно по строкам отыскиваются отличающиеся пары состояний (a_α и a_β, где $\alpha \neq \beta$), которые отсутствуют в первом основном столбце таблицы пар. Если в какой-либо строке имеется хотя бы одна такая пара, то в этой строке зачеркивается пара в первом столбце. В нашем примере это пары (3-7), (2, 5), (3-6), (8-2). Для них вычеркиваем в первом столбце таблицы пары (1-3), (1-8), (3-5), (5-8), (2-4), (2-6), (2-7). Такие строки, в которых зачеркнуты пары в первом столбце, называются *выделенными строками*.

2. Находятся невыделенные строки, в которых имеются пары, вычеркнутые в первом столбце на предыдущем этапе. Если такие строки имеются, то для них зачеркиваются пары в первом столбце. В нашем примере таких пар нет. Такой процесс повторяется до тех пор, пока на очередном этапе не обнаруживаются невыделенные строки, в которых имеются пары, вычеркнутые в первом столбце на предыдущем этапе.

Таблица 1.27. Таблица пар после 1-го этапа алгоритма

Пары 1-эквивалентных состояний	Z_1	Z_2
~~1-3~~	3-7	2-5
1-5	3-3	2-2
~~1-8~~	3-6	2-5
~~3-5~~	7-3	5-2
3-8	7-6	5-5
~~5-8~~	3-6	2-5
~~2-4~~	8-2	4-4
~~2-6~~	8-2	4-7
~~2-7~~	8-2	4-6
4-6	2-2	4-7
4-7	2-2	4-6
6-7	2-2	7-6

3. Оставшиеся незачеркнутые пары в первом столбце таблицы образуют все пары эквивалентных состояний. Для нашего примера это пары: (a_1, a_5), (a_3, a_8), (a_4, a_6), (a_4, a_7), (a_6, a_7), соответствующие невыделенным строкам.

Рассмотренная методика определения пар эквивалентных состояний, основанная на последовательном вычеркивании пар неэквивалентных состояний, базируется на использовании введенных ранее свойств по определению эквивалентности одних состояний при установленной эквивалентности других состояний.

Учитывая свойства транзитивности для эквивалентных состояний, а также состояния, которые не вошли в пары эквивалентных состояний, получим следующее множество классов эквивалентности:

$$\pi_1 = \left\{ \overline{1.5},\ \overline{2}\ ,\ \overline{3.8},\ \overline{4.6.7} \right\}.$$

Построение минимального автомата Мили S_3 рассмотрено выше (см. табл. 1.25).

В том случае если должна быть выполнена минимизация автомата Мура, для построения первого основного столбца таблицы пар используются классы 0-эквивалентных состояний. Дальнейшая методика определения эквивалентного разбиения состояний ничем не отличается от рассмотренной методики для автомата Мили.

1.7. Задачи анализа и синтеза ЦА

При решении задачи синтеза цифровых автоматов выделяются этапы абстрактного и структурного синтеза. На этапе абстрактного синтеза описывается закон функционирования (поведение) автомата на одном из начальных языков, а затем выполняется переход к описанию алгоритма на одном из стандартных языков. Заканчивается этап абстрактного синтеза минимизацией числа состояний автомата.

В качестве стандартного языка используются прямые таблицы переходов, СКУ и СВФ. В ряде случаев язык СКУ и СВФ может быть использован и в качестве начального языка.

На этапе структурного синтеза решается задача построения схемы автомата по заданному описанию на одном из стандартных языков.

Проблема анализа автоматов противоположна проблеме синтеза, ее решение заключается в описании для заданного конечного автомата его поведения средствами исходного языка. Обычно задача анализа проще, чем задача синтеза.

Одна из основных задач теории автоматов заключается в том, чтобы задачу анализа и синтеза цифровых автоматов свести к задаче анализа и синтеза комбинационных схем. При этом в качестве основного математического аппарата используется аппарат алгебры логики.

Контрольные вопросы

1. Дайте определение цифрового автомата.
2. В чем отличие структурного цифрового автомата от абстрактного?
3. Назовите важное различие в функционировании автоматов Мили и Мура.
4. Опишите поведение С-автомата.
5. Перечислите варианты цифровых автоматов.
6. Назовите функции операционного и управляющего автоматов.
7. Основные положения принципа микропрограммного управления.
8. Типы управляющих автоматов. Сравнительная характеристика.
9. Способы задания цифровых автоматов.
10. Назовите основные этапы абстрактного синтеза ЦА.
11. Способы задания цифровых автоматов.
12. Перечислите стандартные языки, которые используются для задания цифровых автоматов.
13. В чем преимущества табличного способа задания автоматов?

14. Какие условия должны соблюдаться при любом способе задания автоматов?

15. В чем заключаются условия однозначности и полной определенности применительно к графу?

16. Какой автомат называется не полностью определенным или частичным?

17. Какой автомат называется недетерминированным автоматом?

18. Что определяют системы канонических уравнений (СКУ) и системы выходных функций (СВФ)?

19. Сформулируйте алгоритм преобразования автомата Мили в эквивалентный ему автомат Мура.

20. Сформулируйте алгоритм преобразования автомата Мура в эквивалентный ему автомат Мили.

21. В чем состоит решение задачи минимизации полностью определенных автоматов?

22. Какие события называются эквивалентными? К-эквивалентными?

23. Назовите основные этапы минимизации полностью определенных автоматов методом Ауфенкампа и Хона.

24. Назовите этапы минимизации СКУ с помощью треугольной таблицы.

25. Назовите основные этапы минимизации с помощью таблицы пар.

26. В чем состоит алгоритм просмотра таблицы пар?

Задания для самостоятельной работы

1. Автомат Мили описывается совмещенной таблицей переходов и выходов (табл. 1.28).

Таблица 1.28. Совмещенная таблица переходов и выходов автомата Мили

A	a_1	a_2	a_3	a_4
z_1	a_2/w_1	a_3/w_2	a_4/w_1	a_1/w_1
z_2	a_3/w_2	a_2/w_3	a_1/w_1	a_3/w_1

А. По заданной совмещенной таблице переходов и выходов автомата Мили:

- построить прямую таблицу переходов;
- построить обратную таблицу переходов;
- построить граф;

- записать СКУ и СВФ;
- определить реакцию автомата Мили на произвольное входное слово.

Б. Построить отмеченную таблицу переходов автомата Мура, эквивалентного автомату Мили (табл. 2.31). Для полученного автомата Мура:
- построить прямую таблицу переходов;
- построить обратную таблицу переходов;
- построить граф;
- записать СКУ и СВФ;
- определить реакцию автомата Мура на произвольное входное слово.

2. Дана отмеченная таблица переходов автомата Мура (табл. 1.29).

Таблица 1.29. Отмеченная таблица переходов и выходов автомата Мура

W	w_3	w_2	w_3	w_1
A	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_1	a_4
z_2	a_2	a_4	a_4	a_1

А. По заданной отмеченной таблице переходов:
- построить прямую таблицу переходов;
- построить обратную таблицу переходов;
- построить граф;
- записать СКУ и СВФ;
- определить реакцию автомата Мура на произвольное входное слово.

Б. Построить совмещенную таблицу переходов и выходов автомата Мили, эквивалентного автомату Мура. Для полученного автомата Мили:
- построить прямую таблицу переходов;
- построить обратную таблицу переходов;
- построить граф;
- записать СКУ и СВФ
- определить реакцию автомата Мили на произвольное входное слово.

3. Минимизировать полностью определённый автомат Мили, заданный таблицами переходов и выходов (табл. 1.30 и 1.31, соответственно) тремя способами:

а) методом Ауфенкампа и Хона,

б) с помощью треугольной таблицы,

в) с помощью таблицы пар.

Правильность преобразований проверить с помощью определения реакции исходного и минимального автоматов.

Таблица 1.30. Таблица переходов автомата Мили

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_3	a_{10}	a_7	a_2	a_3	a_2	a_2	a_1
z_2	a_2	a_4	a_5	a_{11}	a_2	a_9	a_8	a_2

Таблица 1.31. Таблица выходов автомата Мили

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	w_1	w_2	w_1	w_2	w_1	w_2	w_2	w_1
z_2	w_2	w_1	w_2	w_1	w_2	w_1	w_1	w_3

4. Минимизировать полностью определённый автомат Мура, заданный отмеченной таблицей переходов (табл. 1.32) тремя способами:

а) методом Ауфенкампа и Хона,

б) с помощью треугольной таблицы,

в) с помощью таблицы пар.

Правильность преобразований проверить с помощью определения реакции исходного и минимального автоматов.

Таблица 1.32. Отмеченная таблица переходов автомата Мура

	w_1	w_1	w_3	w_3	w_3	w_2	w_3	w_1	w_2	w_2	w_2	w_2
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

Глава 2. АВТОМАТЫ И ФОРМАЛЬНЫЕ ЯЗЫКИ

2.1. Основы теории формальных грамматик

Для описания функционирования цифровых автоматов используются формальные языки. Формальный язык представляет собой множество цепочек в некотором конечном алфавите. К формальным языкам можно отнести искусственные языки для общения человека с машиной – языки программирования.

Для задания описания формального языка необходимо:

1) указать алфавит, т.е. совокупность объектов, называемых символами (или буквами), каждый из которых можно воспроизводить в неограниченном количестве экземпляров (подобно обычным печатным буквам или цифрам);

2) задать формальную грамматику языка, т.е. перечислить правила, по которым из символов строятся их последовательности, принадлежащие определяемому языку – правильные цепочки.

2.1.1. Концепция порождения и распознавания

Отдельные предметы или явления обладают общими признаками (свойствами, отличительными особенностями). Вопрос выбора набора признаков неоднозначен и в общем случае не формализован и относится скорее к курсу "Системы искусственного интеллекта", нежели к "Теории автоматов".

Под ***распознаванием*** будем понимать целевую классификацию объектов по набору у них некоторой совокупности признаков, характеризующих назначение объекта и принадлежность его к тому или иному классу [2].

Одним из основных понятий в теории распознавания является понятие *класса,* или *образа.*

Под ***классом*** будем понимать некоторое множество объектов, характеризуемое определенным набором признаков, общих для всех объектов класса.

Таким образом, при распознавании предполагается наличие набора или алфавита классов объектов:

$A = \{A_1, A_2, \ldots, A_m\}$,

например, A_1 – класс объектов "мониторы", A_2 – класс объектов "столы" и т. п.

Каждый класс в алфавите может быть представлен некоторым количеством объектов. Совокупность объектов для всех классов образует

множество объектов: $B = \{B_1, B_2, \ldots, B_t\}$, например, B_1 – ЖК монитор, B_2 – ЭЛТ монитор, B_t – письменный стол и т.п.

В большинстве случаев количество самих объектов конечно и намного больше количества классов объектов.

Для простоты будем считать, что все классы характеризуются одним и тем же количеством признаков N (данное допущение абстрактно, поскольку мониторы и столы из предыдущих примеров никак нельзя описать одним и тем же набором признаков). Совокупность N признаков для алфавита A можно обозначить символами:

$X = \{x_1, x_2, \ldots, x_k, \ldots, x_n\}$.

Для реализации распознающих автоматов признаки удобно привести к числовой форме, т.е. определить соответствие между какими-то качественными признаками (например, “теплее/холоднее чем”, “больше/меньше, чем”, “тише/громче, чем”, “есть структурный элемент / нет структурного элемента”) или цветовыми характеристиками и числами. У объектов одного класса могут быть разные значения одних и тех же признаков.

Каждый конкретный объект в таком случае будет описываться совокупностью значений признаков:

$$B_j = \begin{Bmatrix} x_1^{j1}, & , & , & x_k^{jk}, & , x_n^{jn} \\ x_2 & & & & \end{Bmatrix},$$

где x_1^{j1} – значение первого признака для объекта B_j.

Таким образом, **распознающая машина** – это некоторая система, на вход которой поступают объекты для распознавания, а на выходе появляются распознанные объекты, отнесенные к определенному классу.

2.1.2. Классификация языков по Хомскому

Пусть алфавит символов (непустое конечное множество), из которых строятся цепочки языка L, представляет собой алфавит терминальных символов V_T, например символы 0 и 1 для двоичной системы счисления, т.е. $V_T = \{0, 1\}$.

Определение формальной грамматики требует наличия ещё одного алфавита V_N – непустого конечного множества нетерминальных символов, состоящих из некоторого множества терминальных, таким образом, что алфавиты терминальных и нетерминальных символов являются непересекающимися множествами. Например, двухразрядные двоичные числа 00, 01, 11, т.е. $V_N = \{00, 01, 11\}$.

Объединение терминального V_T и нетерминального V_N алфавитов называется словарем формального языка L.

Продукция языка представляет собой пару <α, β>, каждый элемент этой пары является цепочкой символов словаря формального языка L:

$D = <\alpha, \beta>$.

Продукция обозначается α > β (читается “из α следует β”). Необходимо отметить, что цепочка β является элементом ограниченного использования словаря, поэтому среди продукций не должно быть пар вида α > e, где e – пустая цепочка.

Таким образом, формальная грамматика G – это совокупность четырех объектов:

$G = (V_T, V_N, P, S)$,

где V_T – алфавит терминальных символов,

V_N – алфавит нетерминальных символов,

P – конечное множество продукций формальной грамматики G, являющееся подмножеством множества D.

S – начальный символ грамматики.

Н. Хомский определил четыре типа грамматик, на основе которых оцениваются возможности других способов описания языков. Соответствующий тип грамматики по Хомскому определяется ограничениями, которые налагаются на продукцию P. В табл. 2.1 приводятся типы грамматики, типы языков и автоматы, которые могут быть описаны с помощью грамматики данного типа.

Таблица 2.1. Классификация языков по Хомскому

Тип грамматики	Ограничение	Языки	Автоматы
Тип 0	Нет ограничений	Рекурсивно-перечислимые языки	Машины Тьюринга
Тип 1	Длина цепочки β больше или равна длине цепочки α	Контекстно-зависимые, или грамматики непосредственных составляющих (НС-языки)	Линейно-ограниченные автоматы
Тип 2	Цепочка α состоит из одного символа	Бесконтекстные или контекстно-свободные грамматики (КС-языки)	Автомат с магазинной памятью
Тип 3	Цепочка β состоит либо из одного терминального и одного нетерминального символа, либо из одного терминального символа	Регулярные языки	Конечные автоматы

2.1.3. Порождающие грамматики

В общем случае язык представляет собой бесконечное множество, а бесконечные объекты даже задать трудно: их невозможно задать простым перечислением элементов. Любой конечный механизм задания языка называется грамматикой.

Правила формальной грамматики можно рассматривать как продукции (правила вывода), то есть элементарные операции, которые, будучи применены в определенной последовательности к исходной цепочке (аксиоме), порождают лишь правильные цепочки. Сама последовательность правил, использованных в процессе порождения некоторой цепочки, является ее выводом. Определенный таким образом язык представляет собой формальную систему. По способу задания правильных цепочек формальные грамматики разделяются на порождающие и распознающие.

Порождающая грамматика позволяет построить любую правильную цепочку с указанием ее структуры и не позволяет построить ни одной неправильной цепочки. На порождающей грамматике базируются структурные методы распознавания.

Порождающая грамматика состоит из четырёх частей:

1) основной, или терминальный словарь;

2) вспомогательный словарь;

3) начальный символ;

4) набор правил подстановки исходных элементов, из которых

строят цепочки, порождаемые грамматикой.

Основной, или терминальный словарь – множество базовых, неделимых структурных элементов, из которых может быть построен распознаваемый объект.

Вспомогательный словарь – набор символов (элементов), которыми обозначаются классы (совокупности) исходных элементов или цепочек исходных элементов, а также в отдельных случаях некоторые специальные элементы, вспомогательные или терминальные.

Начальный символ – выделенный нетерминальный символ, обозначающий класс объектов, для описания которых предназначается данная грамматика.

Правила подстановки – выражения вида "$X > Y$", "X вместо Y", где X и Y – цепочки, содержащие терминальные или нетерминальные символы.

Если имеются цепочки X и Y, которые можно представить в виде $X = Z_1 \alpha Z_2$ и $Y = Z_1 \beta Z_2$, где $\alpha > \beta$ – одно из правил грамматики G, то говорят, что цепочка Y *непосредственно выводима* из цепочки X в грамматике G. Другими словами, цепочка X может быть переработана в цепочку Y за один шаг применением одной подстановки: на место некоторого вхождения цепочки α подставляется цепочка β.

Если имеется последовательность цепочек $X_0, X_1, \ldots, X_n$, в которой каждая следующая цепочка непосредственно *выводима* из предыдущей, то цепочка X_n выводима из цепочки X_0, а такая последовательность цепочек называется выводом X_n из X_0 в грамматике G.

Необходимо отметить, что порождающая грамматика не является алгоритмом. Правила подстановки являются совокупностью решений, а не последовательностью действий. Решение может быть применено, а может и не применяться для вывода. Кроме того, порядок применения правил вывода произволен.

Язык, порожденный грамматикой, представляет собой совокупность всех терминальных цепочек, т.е. цепочек, состоящих из терминальных символов, выводимых из начального символа в грамматике G. Язык, порожденный грамматикой G, обозначается $L(G)$.

Две различные грамматики могут порождать один и тот же язык, при этом эти грамматики называются эквивалентными.

2.1.4. Регулярные языки

В теории языков регулярным множеством (или регулярным языком) называется формальный язык, который удовлетворяет приведённым ниже свойствам. Пусть Σ – конечный алфавит.

1. $\varnothing \in R(\Sigma)$ – пустое множество, является регулярным множеством в алфавите Σ.

2. $\{\lambda\} \in R(\Sigma)$ – множество, состоящее из одной лишь пустой строки, является регулярным множеством в алфавите Σ.

3. $\forall a \in \Sigma$: $\{a\} \in R(\Sigma)$ – множество, состоящее из одного любого символа алфавита Σ, является регулярным множеством в алфавите Σ.

4. $P \in R(\Sigma)$ & $Q \in R(\Sigma) > (P \cup Q) \in R(\Sigma)$ – если два каких-либо множества являются регулярными в алфавите Σ, то и их объединение тоже является регулярным множеством в алфавите Σ.

5. $P \in R(\Sigma)$ & $Q \in R(\Sigma) > P\,Q \in R(\Sigma)$ – если два каких-либо множества являются регулярными в алфавите Σ, то и множество, составленное из всевозможных сцеплений пар их элементов, тоже является регулярным множеством в алфавите Σ.

6. $P \in R(\Sigma) > P^* \in R(\Sigma)$ – если какое-либо множество является регулярным в алфавите Σ, то множество всевозможных сцеплений его элементов тоже является регулярным множеством в алфавите Σ.

Ничто другое, кроме перечисленного, не является регулярным множеством в алфавите Σ.

2.2. Распознаватели: машина Тьюринга, магазинный автомат, сеть Петри, конечный автомат

Машина Тьюринга – абстрактное математическое понятие, названное по имени английского математика А. Тьюринга. В каждой машине Тьюринга выделяются следующие три части:

1) неограниченная в обе стороны лента, разделенная на ячейки;

2) управляющее устройство (УУ);

3) головка (Г).

В любой момент времени в каждой ячейке ленты записана одна буква из алфавита внешних символов *Z*, а устройство управления находится в одном из внутренних состояний, входящих в множество *A*.

Совокупность сведений о состоянии УУ и записи на ленте машины с указанием обозреваемой ячейки называется *конфигурацией машины Тьюринга.*

Работа машины Тьюринга состоит из тактов, в каждом из которых

выполняется преобразование из конфигурации, в которой машина Тьюринга находится в данный момент времени t, в конфигурацию, в которой машина будет находиться в момент t+1. Это преобразование зависит только от состояния УУ и содержимого обозреваемой ячейки в момент t и заключается:

а) в изменения состояние $a(t)$ в некоторое состояние $a(t+1)$;

б) в замене внешнего символа z_i, записанного в обозреваемой ячейке, некоторой буквой z_j.

Такое преобразование называется *командой* машины Тьюринга. Совокупность всех команд, которые выполняет машина Тьюринга, называется *программой.*

Магазинный автомат – автомат специального вида (как правило, бесконечный), в основе которого лежит понятие *магазинной памяти* (*стека*). Магазин удобно представлять в виде бесконечной в одну сторону ленты, состоящей из ячеек, пронумерованных целыми натуральными числами. При чтении воспринимается содержимое только первой ячейки памяти. При считывании содержимое первой ячейки памяти стирается, а оставшееся содержимое поднимается на одну ячейку вверх. При записи данных в такую память содержимое верхней ячейки сдвигается вниз, а в верхнюю ячейку записывается новое содержимое. Различают распознающие магазинные автоматы (акцепторы), у которых выходной алфавит пуст, порождающие магазинные автоматы, у которых входной алфавит пуст, и магазинные преобразователи (трансдьюсеры).

Сеть Петри обеспечивает описание как алгоритмов и программ, так и собственно вычислительных систем и их устройств, а также порождаемых вычислительных процессов. Сети Петри используются для решения разнообразных задач анализа, синтеза и оптимизации.

При построении модели дискретной системы сначала происходит абстрагирование от конкретных физических и функциональных особенностей ее компонентов.

В отличие от реальных систем, функционирующих во времени и в которых события привязаны к определенным моментам или интервалам времени, в сетях Петри обычно отказываются от введения в модели дискретных систем времени и тактирования последовательностей изменений состояний. Вместо них используются причинно-следственные связи между событиями (т.е. используются асинхронные модели). Если

требуется осуществить привязку ко времени, то моменты времени или интервалы времени рассматриваются как события. Более подробно сети Петри рассматриваются в курсе “Моделирование”.

Конечный автомат – автомат, у которого множество внутренних состояний и множество входных значений (а следовательно, и множество выходных значений) являются конечными множествами. Абстрактный конечный автомат задается шестеркой $<A, X, Y, \delta, \lambda, a_1>$, где A, X, Y – конечные множества, называемые, соответственно, множеством внутренних состояний, множеством входных сигналов и множеством выходных сигналов, δ и λ – однозначные функции переходов (δ: $A_{\times}X > A$) и выходов (λ: $A_{\times}X > Y$), а также выделяется начальное состояние автомата a_1.

Понятие конечного автомата было предложено в качестве математической модели технических устройств дискретного действия, т.к. любое реальное устройство может иметь конечное число состояний.

2.3. Коллективы автоматов

Коллективы автоматов – сеть, состоящая, как минимум, из двух автоматов. Существуют три основных вида соединения автоматов: параллельное, последовательное и с обратной связью.

1. Параллельное соединение автоматов (рис. 2.1).

Здесь S_1 и S_2 – два параллельно соединенных автомата. Входной алфавит у обоих автоматов один и тот же. Выходы автоматов S_1 и S_2 соединены с функциональным преобразователем φ (автоматом без памяти), реализующим отображение φ: $W_{1\times}\ W_2 > W$.

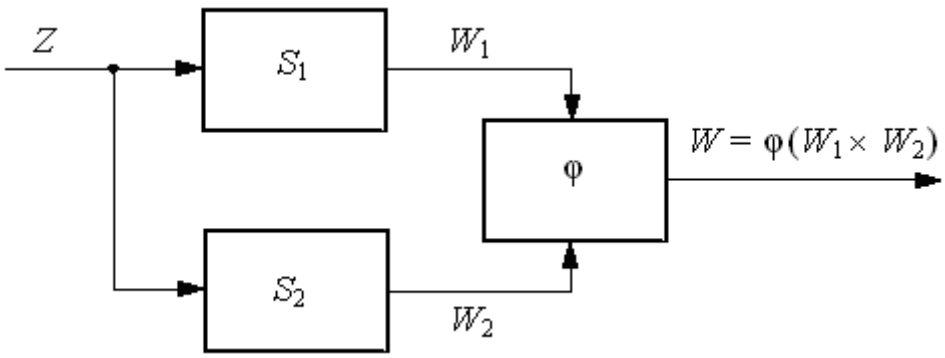


Рис. 2.1. Структурная схема параллельного соединения автоматов

Результирующим автоматом параллельного соединения двух автоматов S_1 и S_2 называется автомат $S = <A, Z, W, \delta, \lambda>$, у которого:

1. Множество состояний $A = A_1 \times A_2$ – множество всевозможных пар, первые и вторые компоненты которых являются состояниями автоматов S_1 и S_2:

$A = \{a_m = (a_{m1}, a_{m2}) \mid a_{m1} \times A_1, a_{m2} \times A_2\}$.

2. Входной алфавит – входной алфавит Z автоматов S_1 и S_2.

3. Выходной алфавит $W = \varphi(W_1 \times W_2)$, где φ – заданное отображение.

4. Функция переходов δ: $A \times Z > A$, определяемая следующим образом:

$\delta(a_m, z_f) = (\delta_1(a_{m1}, z_f), \delta_2(a_{m2}, z_f)), z_f \times Z$

или

$\delta(A \times Z) = (\delta_1(A_1 \times Z), \delta_2(A_2 \times Z))$.

5. Функция выходов λ: $A \times Z > W$, определяемая следующим образом:

$\lambda(a_m, z_f) = \varphi(\lambda_1(a_{m1}, z_f), \lambda_2(a_{m2}, z_f)), z_f \times Z,$

иначе

$\lambda(A \times Z) = \varphi(\lambda_1(A_1 \times Z), \lambda_2(A_2 \times Z))$.

2. Последовательное соединение двух автоматов (рис. 2.2).

Здесь S_1 и S_2 – два последовательно соединенных автомата, выходной алфавит W_1 автомата S_1 является входным алфавитом автомата S_2.

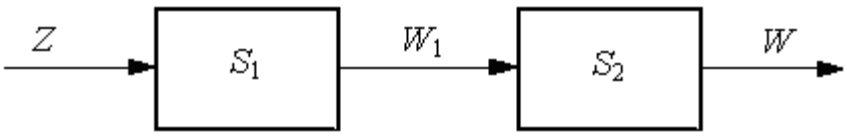


Рис. 2.2. Структурная схема последовательного соединения автоматов

Результирующий автомат S последовательного соединения описывается следующим образом:

$S = <A, Z, W, \delta, \lambda>$.

1. $A = A_1 \times A_2$, иначе $A = \{a_m = (a_{m1}, a_{m2}) \mid a_{m1} \times A_1, a_{m2} \times A_2\}$.

2. $Z = Z$.

3. $W = W$.

4. Функция переходов δ: $A \times Z > A$, определяемая следующим образом:

$\delta(a_m, z_f) = (\delta_1(a_{m1}, z_f), \delta_2(a_{m2}, \lambda_1(a_{m1}, z_f)))$

или

$\delta(A \times Z) = (\delta_1(A_1 \times Z), \delta_2(A_2 \times \lambda_1(A_1 \times Z)))$.

1. Функция выходов λ: $A \times Z > W$, определяемая следующим образом:

$\lambda(a_m, z_f) = \lambda_2(a_{m2}, \lambda_1(a_{m1}, z_f))$

или

$\lambda\,(A\times Z) = \lambda_2\,(A_2\times \lambda_1(A_1\times Z))$.

3. Соединение с обратной связью (рис. 2.3).

Функциональный преобразователь γ (автомат без памяти) реализует отображение γ: $Z \times W_2 > Z_1$.

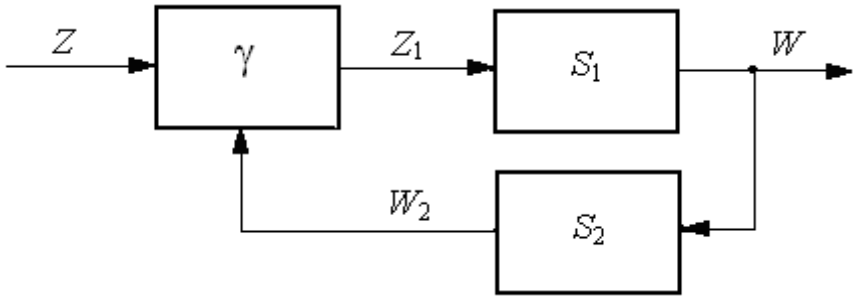


Рис. 2.3. Структурная схема соединения с обратной связью

Необходимо, чтобы, по крайней мере, один из автоматов S_1 или S_2 был автоматом Мура, в противном случае сигнал на выходе автомата будет зависеть от самого же себя, поступающего на вход.

Пусть S_2 – автомат Мура. Тогда результирующим автоматом соединения двух автоматов S_1 и S_2 с обратной связью будет автомат $S = <A, Z, W, \delta, \lambda>$, у которого:

1. $A = A_1\times A_2$, иначе $A = \{a_m = (a_{m1}, a_{m2}) \mid a_{m1}\in A_1, a_{m2}\in A_2\}$.
2. $Z = Z$.
3. $W = W$.
4. δ: $A\times Z > A$, определяемая следующим образом:

$\delta\,(a_m, z_f) = (\delta_1\,(a_{m1}, \gamma\,(z_f, \lambda_2(a_{m2}))),$

$\delta_2\,(a_{m2}, \lambda_1(a_{m1,}\,\gamma\,(z_f, \lambda_2(a_{m2}))))$

или

$\delta\,(A\times Z) = (\delta_1\,(A_1\times \gamma\,(Z\times \lambda_2(A_2))),$

$\delta_2\,(A_2\times \lambda_1(A_1\times \gamma\,(Z\times \lambda_2(A_2))))$.

5. λ: $A\times Z > W$, определяемая следующим образом:

$\lambda\,(a_m, z_f) = \lambda_1(a_{m1},\gamma\,(z_f, \lambda_2(a_{m2}))$

или

$\lambda\,(A\times Z) = \lambda_1(A_1\times \gamma\,(Z\times \lambda_2(A_2)))$.

Примеры решения задач

Пример 2.1. Описать автомат, получаемый при параллельном соединении автоматов S_1 и S_2 (см. рис. 2.1). Автоматы заданы табличным способом:

A_1 / Z_1	a_1^1	a_1^2	a_1^3
z_1^1	a_1^2 / w_1^1	a_1^1 / w_1^2	a_1^2 / w_1^1
z_1^2	a_1^3 / w_1^1	a_1^3 / w_1^1	a_1^1 / w_1^2

W_2	w_2^1	w_2^2
A_2 / Z_2	a_2^1	a_2^2
z_1^1	a_2^2	a_2^1
z_1^2	a_2^1	a_2^2

Для получения полностью определенного автомата S необходимо, чтобы входной алфавит Z автоматов S_1 и S_2 совпадал. Пусть функция ц представляет собой логическую функцию – конъюнкцию.

Решение

А. Алфавит состояний результирующего автомата S определяется как прямое произведение алфавитов состояний автоматов S_1 и S_2, т.е. множество состояний результирующего автомата S состоит из всех комбинаций состояний автоматов S_1 и S_2:

$$A = A_1 \text{ x } A_2 = \{a_1^1, a_1^2, a_1^3\} \text{x} \{a_2^1, a_2^2\} =$$

$$= \{(a_1^1, a_2^1), (a_1^2, a_2^1), (a_1^3, a_2^1), (a_1^1, a_2^2), (a_1^2, a_2^2), (a_1^3, a_2^2)\} =$$

$$= \{a_1, a_2, a_3, a_4, a_5, a_6\}.$$

Б. Входной алфавит результирующего автомата S совпадает с входными алфавитами автоматов S_1 и S_2:

$$Z = \{z_1^1, z_1^2\} = \{z_2^1, z_2^2\} = \{z_1, z_2\}.$$

В. Функция переходов автомата S определяется следующим образом: под воздействие какого-то сигнала z_f автомат S_1 переходит в состояние a_{m1}, а автомат S_2 – в состояние a_{m2}. Следовательно, результирующий автомат под воздействием сигнала z_f перейдет в состояние, которое представляет собой комбинацию состояний a_{m1} и a_{m2} автоматов S_1 и S_2, соответственно.

Таким образом, результирующий автомат S будет описываться следующей таблицей переходов:

A	a_1	a_2	a_3	a_4	a_5	a_6
Z $A_1 x A_2$	$a_1^1 a_2^1$	$a_1^2 a_2^1$	$a_1^3 a_2^1$	$a_1^1 a_2^2$	$a_1^2 a_2^2$	$a_1^3 a_2^2$
$z_1 = z_1^1 = z_1^2$	$a_1^2 a_2^2$	$a_1^1 a_2^2$	$a_1^2 a_2^2$	$a_1^2 a_2^1$	$a_1^1 a_2^1$	$a_1^2 a_2^1$
$z_2 = z_2^1 = z_2^2$	$a_1^3 a_2^1$	$a_1^3 a_2^1$	$a_1^1 a_2^1$	$a_1^3 a_2^2$	$a_1^3 a_2^2$	$a_1^1 a_2^2$

или

Z A	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_5	a_4	a_5	a_2	a_1	a_2
z_2	a_3	a_3	a_1	a_6	a_6	a_4

Г. Выходной алфавит результирующего автомата S определяется как прямое произведение множеств выходных сигналов автоматов S_1 и S_2. т.е. множество выходных сигналов результирующего автомата S состоит из всех комбинаций выходных сигналов автоматов S_1 и S_2, объединенных функцией φ (согласно заданию – конъюнкцией).

$W = W_1 \text{ x } W_2 = \{(w_1^1, w_2^1), (w_1^1, w_2^2), (w_1^2, w_2^1), (w_1^2, w_2^2)\} = \{w_1, w_2, w_3, w_4\}$.

Таблица выходов результирующего автомата S строится с учетом типа автоматов S_1 и S_2:

A	a_1	a_2	a_3	a_4	a_5	a_6
Z $A_1 x A_2$	$a_1^1 a_2^1$	$a_1^2 a_2^1$	$a_1^3 a_2^1$	$a_1^1 a_2^2$	$a_1^2 a_2^2$	$a_1^3 a_2^2$
z_1	$w_1^1 w_2^1$	$w_1^2 w_2^1$	$w_1^1 w_2^1$	$w_1^1 w_2^2$	$w_1^2 w_2^2$	$w_1^1 w_2^2$
z_2	$w_1^1 w_2^1$	$w_1^1 w_2^1$	$w_1^2 w_2^1$	$w_1^1 w_2^2$	$w_1^1 w_2^2$	$w_1^2 w_2^2$

или

Z A	a_1	a_2	a_3	a_4	a_5	a_6
z_1	w_1	w_3	w_1	w_2	w_4	w_2
z_2	w_1	w_1	w_3	w_2	w_4	w_4

Пример 2.2. Описать автомат, получаемый при последовательном соединении автоматов S_1 и S_2 (рис. 2.4), заданных табличным способом в примере 1.

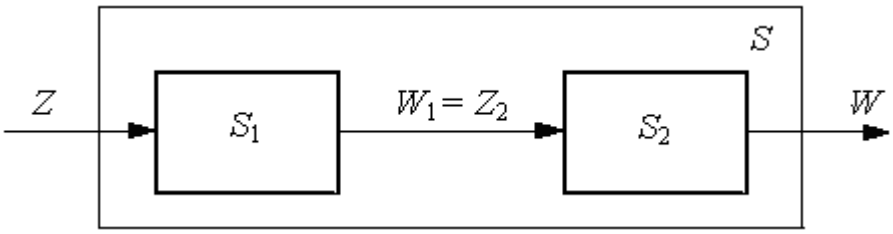


Рис. 2.4. Структурная схема автомата S, представляющего собой последовательное соединение автоматов S_1 и S_2 в группу

Для получения полностью определенного автомата *S*, представляющего собой результат последовательного соединения автоматов S_1, необходимо, чтобы выходной алфавит автомата S_1 совпадал с входным алфавитом автомата S_2: $W_1 = Z_2 = \{ z_2 = w_1, z_2 = w_1 \}$.

Решение

А. Алфавит состояний результирующего автомата *S* определяется как прямое произведение алфавитов состояний автоматов S_1 и S_2, т.е. множество состояний результирующего автомата *S* состоит из всех комбинаций состояний автоматов S_1 и S_2:

$$A = A_1 \times A_2 = \{a_1, a_2, a_3\} \times \{a_1, a_2\} =$$

$$= \{(a_1^1, a_2^1), (a_1^2, a_2^1), (a_1^3, a_2^1), (a_1^1, a_2^2), (a_1^2, a_2^2), (a_1^3, a_2^2)\} =$$

$$= \{a_1, a_2, a_3, a_4, a_5, a_6\}.$$

Б. Входной алфавит результирующего автомата *S* совпадает с входным алфавитом автомата S_1: $Z = \{z_1^1, z_2^1\} = \{z_1^2, z_2^2\} = \{z_1, z_2\}$.

В. Выходной алфавит результирующего автомата S совпадает с выходным алфавитом автомата S_2: $W = W_2 = \{ w_1 = w_2^1, \ w_2 = w_2^2 \}$.

Г. Функция переходов результирующего автомата *S* определяется следующим образом: под воздействием входного сигнала z_f автомат S_1 переходит в состояние a_m, при этом формируется выходной сигнал w_j, а автомат S_2 под воздействием входного сигнала w_j переходит в состояние a_n. Таким образом, результирующий автомат *S* перейдет в состояние, которое является комбинацией состояний a_m автомата S_1 и a_n автомата S_2.

Пусть в нашем примере автомат *S* находится в состоянии a_1. То есть автомат S_1 находится в состоянии a_1^1, а автомат S_2 – в состоянии a_2^1. Тогда автомат S_1 из состояния a_1^1 под воздействием входного сигнала

z_1 перейдет в состояние a_1^2, при этом сформируется выходной сигнал w_1^1, который поступит на вход автомата S_2 и инициирует переход автомата S_2 из состояния a_2^1 в состояние a_2^2. Таким образом, результирующий автомат S перейдет в состояние $a_1^2 a_2^2 = a_5$.

Д. Поскольку автомат S_2 является автоматом Мура, следовательно, выходные сигналы автомата S будут определяться только состоянием автомата S_2: $W_2=\{w_2^1, w_2^2\}$. Таким образом, автомат S будет описываться следующей отмеченной таблицей переходов:

W	w_2^1	w_2^1	w_2^1	w_2^2	w_2^2	w_2^2
A	a_1	a_2	a_3	a_4	a_5	a_6
Z $A_1 x A_2$	$a_1^1 a_2^1$	$a_1^2 a_2^1$	$a_1^3 a_2^1$	$a_1^1 a_2^2$	$a_1^2 a_2^2$	$a_1^3 a_2^2$
z_1	$a_1^2 a_2^2$	$a_1^1 a_2^1$	$a_1^2 a_2^2$	$a_1^2 a_2^2$	$a_1^1 a_2^2$	$a_1^2 a_2^1$
z_2	$a_1^3 a_2^2$	$a_1^3 a_2^2$	$a_1^1 a_2^1$	$a_1^3 a_2^1$	$a_1^3 a_2^1$	$a_1^1 a_2^2$

или

W	w_1	w_1	w_1	w_2	w_2	w_2
Z A	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_5	a_1	a_5	a_4	a_4	a_2
z_2	a_6	a_6	a_1	a_3	a_3	a_4

Пример 2.3. Описать автомат, получаемый при соединении автоматов S_1 и S_2 в группу с обратной связью (рис. 2.5), заданных табличным способом в примере 1.

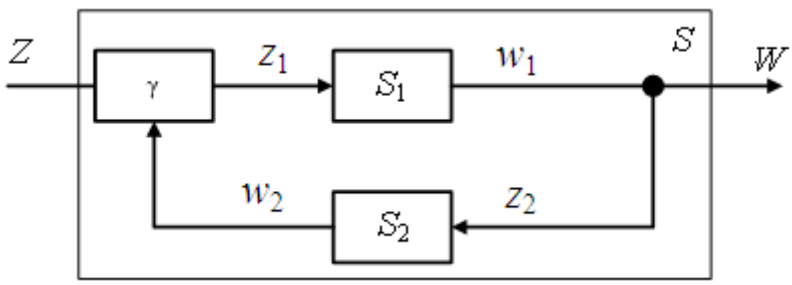


Рис. 2.5. Структурная схема автомата S, представляющего собой соединение автоматов S_1 и S_2 в группу с обратной связью

Для реализации полностью определенного автомата, представля-

ющего собой соединение с обратной связью автоматов S_1 и S_2: входной алфавит автомата S_1 должен совпадать с прямым произведением множеств Z и W_2, а входной алфавит автомата S_2 должен совпадать с выходным алфавитом автомата S_1. Поскольку это условие для автоматов, заданных в примере 1, не выполняется, следовательно, результирующий автомат S не будет полностью определенным.

Решение

Пусть функция γ, объединяющая выходные сигналы автомата S_2 и входные сигналы $Z = \{z\}$ автомата S, представляет собой логическую функцию "И". Тогда результирующий автомат S будет описываться следующими множествами:

А. Алфавит состояний результирующего автомата S определяется как прямое произведение алфавитов состояний автоматов S_1 и S_2, т.е. множество состояний результирующего автомата S состоит из всех комбинаций состояний автоматов S_1 и S_2.

$$A = A_1 \times A_2 = \{a_1^1, a_1^2, a_1^3\} \times \{a_2^1, a_2^2\} =$$
$$= \{(a_1^1, a_2^1), (a_1^2, a_2^1), (a_1^3, a_2^1), (a_1^1, a_2^2), (a_1^2, a_2^2), (a_1^3, a_2^2)\} =$$
$$= \{a_1, a_2, a_3, a_4, a_5, a_6\}.$$

Б. Входной алфавит результирующего автомата S: $Z = \{z_1, z_2\}$.

В. Выходной алфавит результирующего автомата S совпадает с выходным алфавитом автомата S_1: $W = W_1 = \{w_1^1, w_1^2\}$.

Г. Функция переходов результирующего автомата S определяется следующим образом: под воздействием определенной входной комбинации zw^i_2 автомат S_1 переходит в состояние S_m и формирует выходной сигнал w^i_1, который заставит автомат S_2 переключиться в состояние S_m. Комбинация состояний $S_m S_n$ и будет определять состояние результирующего автомата S.

Д. Функция выходов результирующего автомата S совпадает с функцией выходов автомата S_1:

$$W = W_1 = \{w_1 = w_1^1, w_2 = w_1^2\}.$$

Опишем автомат S из нашего примера.

Пусть автомат S находится в состоянии a_1, т.е. автомат S_1 в состоянии a_1^1, а автомат S_2 – в состоянии a_2^1. В этом случае автомат S_2 формирует выходной сигнал w_2^1, который вместе с входным сигналом z_1 ини-

циирует переход автомата S_1 в состояние a_1^2 и формирование им выходного сигнала w_1^1.

Выходной сигнал w_1^1 автомата S_1 инициирует в свою очередь переход автомата S_2 в состояние a_2^2. Таким образом, автомат S из состояния a_1 под воздействием входного сигнала z_1 перейдет в состояние, определяемое комбинацией $a_1^2 a_2^2$.

Поскольку в состоянии a_2^1 автомат S_2 формирует выходной сигнал w_2^1, следовательно, комбинации сигналов $z_1 w_2^2$ и $z_2 w_2^2$ будут неактивными и не будут инициировать переходы из состояния a_1 автомата S. Поэтому в таблице переходов ставятся прочерки.

A / Z	a_1	a_2	a_3	a_4	a_5	a_6
$A_1 x A_2$	$a_1^1 a_2^1$	$a_1^2 a_2^1$	$a_1^3 a_2^1$	$a_1^1 a_2^2$	$a_1^2 a_2^2$	$a_1^3 a_2^2$
$z_1 w_2^1$	$a_1^2 a_2^2$	$a_1^1 a_1^2$	$a_1^2 a_2^2$	–	–	–
$z_2 w_2^1$	$a_1^3 a_2^2$	$a_1^3 a_2^2$	$a_1^1 a_2^1$	–	–	–
$z_1 w_2^2$	–	–	–	$a_1^2 a_2^1$	$a_1^1 a_2^2$	$a_1^2 a_1^2$
$z_2 w_2^2$	–	–	–	$a_1^3 a_2^1$	$a_1^3 a_1^2$	$a_1^1 a_2^2$

или

Z / A	a_1	a_2	a_3	a_4	a_5	a_6
$z_1 w_2^1$	a_5	a_1	a_5	–	–	–
$z_2 w_2^1$	a_6	a_6	a_1	–	–	–
$z_1 w_2^2$	–	–	–	a_2	a_4	a_2
$z_2 w_2^2$	–	–	–	a_3	a_3	a_4

Z \ A	a_1	a_2	a_3	a_4	a_5	a_6
$z_1 w_2^1$	w_1	w_2	w_1	–	–	–
$z_2 w_2^1$	w_1	w_1	w_2	–	–	–
$z_1 w_2^2$	–	–	–	w_1	w_2	w_1
$z_2 w_2^2$	–	–	–	w_1	w_1	w_2

Контрольные вопросы

1. Что собой представляет формальный язык?
2. Что необходимо для задания описания формального языка?
3. Что понимают под распознаванием?
4. Что такое класс?
5. Что такое распознающая машина?
6. Какие типы формальных языков выделил Н. Хомский?
7. Чем формальные языки отличаются друг от друга?
8. Из каких частей состоит порождающая грамматика?
9. Какой формальный язык называется регулярным языком?
10. Что такое “машина Тьюринга”? Магазинный автомат? Сеть Петри?
11. Какой автомат называется конечным?
12. Какие виды соединений цифровых автоматов вам известны?

Задания для самостоятельной работы

1. Даны два автомата. Автомат Мура S_1 описывается отмеченной таблицей переходов, а автомат Мили S_2 описывается совмещенной таблицей переходов и выходов.

S_1

W_1	w_1^1	w_1^2
A_1 / Z_1	a_1^1	a_1^2
z_1^1	a_1^2	a_1^2
z_1^2	a_1^1	a_1^1

S_2

A_2 / Z_2	a_2^1	a_2^2	a_2^3
z_2^1	a_2^2 / w_2^1	a_2^3 / w_2^2	a_2^1 / w_2^2
z_2^2	a_2^1 / w_2^2	a_2^1 / w_2^1	a_2^2 / w_2^2

Опишите автомат, получаемый при параллельном, последовательном и с обратной связью соединении автоматов S_1 и S_2.

2. Разработайте алгоритм функционирования автомата, переводящего цепочку цифр десятичной системы счисления в соответствующую цепочку цифр {2, 8, 16} систем счисления.

3. Разработайте алгоритм функционирования автомата, распознающего тип системы счисления по элементам цепочки цифр, поступающих на его вход. В каких случаях результат распознавания будет неоднозначным?

Глава 3. СИНТЕЗ АВТОМАТОВ БЕЗ ПАМЯТИ

В автоматах без памяти совокупность выходных сигналов (выходное слово *Y*) в любой момент времени определяется входными сигналами (входной сигнал *X*), поступающими на входы в этот же момент времени:

$Y(t) = \lambda(x(t))$.

Реализуемый в этих автоматах способ обработки информации называют комбинационным, а сами автоматы без памяти – комбинационными схемами (КС), т.к. результат обработки информации зависит только от комбинации входных сигналов и вырабатывается сразу при подаче входной информации. В общем случае КС можно представить схемой, приведенной на рис. 3.1, где $x_1, x_2, \ldots, x_l$ – входы КС, $y_1, y_2, \ldots, y_n$ – ее выходы.

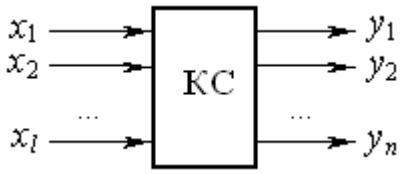


Рис. 3.1. Комбинационная схема

Комбинационная схема состоит из логических элементов и реализует булеву функцию или совокупность булевых функций.

Под логическим элементом понимают техническое устройство, реализующее одну элементарную булеву функцию. Обычно логический элемент рассматривается как «черный ящик» и учитывается только реализующая элементом булева функция.

Конструктивно логические элементы объединяются в единый корпус, называемый *интегральной микросхемой* (ИМС). Под ИМС понимается микроминиатюрное электронное устройство, элементы которого неразделно связаны конструктивно, технологически и электрически. В одном корпусе ИМС могут быть один, два и более логических элементов. Число логических элементов, объединяемых в один корпус ИМС, характеризует *степень интеграции логических элементов*.

Логические элементы, используемые для построения КС, характеризуются определенными техническими параметрами, среди которых нас будут интересовать:

- коэффициент объединения по входу *I* – максимальное число логических элементов, выходы которых могут быть объединены на входе данного элемента;

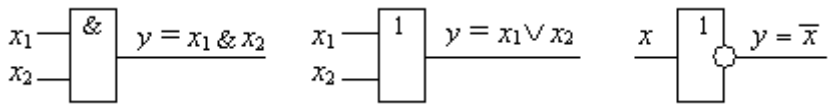
- коэффициент объединения по выходу U (коэффициент разветвления) – максимальное число входов логических элементов, которые могут быть подсоединены к выходу данного элемента без нарушения его работоспособности;

- задержка сигнала τ в логическом элементе – интервал между моментами установления сигналов на входах и выходе элемента.

Базис (совокупность) элементов, выбранных для синтеза КС, всегда должен быть функционально полным, то есть допускать реализацию любой булевой функции на основе принципа суперпозиции.

Если в качестве базиса выбраны элементы И, ИЛИ, НЕ, то считают, что реализован булевый базис. Проектирование схем в булевом базисе наиболее просто, так как все методы минимизации булевых функций в основном ориентированы на него. Поэтому, как правило, на первом этапе КС проектируются в булевом базисе с последующим переходом в заданный базис.

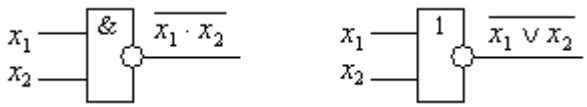
Обозначения логических элементов, реализующих основные булевы функции И, ИЛИ, НЕ: конъюнктор (схема И), дизъюнктор (схема ИЛИ), инвертор (схема НЕ), приведены на рис. 3.2,а,б,в, соответственно.



а) *б)* *в)*

Рис. 3.2. Обозначения логических элементов:
а) конъюнктор; б) дизъюнктор; в) инвертор

Применяемые на практике комплексы ИМС имеют в своём составе такие логические элементы, как И-НЕ, ИЛИ-НЕ (рис. 3.3).



а) *б)*

Рис. 3.3. Обозначения логических элементов:
а) И-НЕ; б) ИЛИ-НЕ

Как правило, на первом этапе логические схемы проектируются в булевом базисе, а затем осуществляется преобразование к тому логическому базису, который отвечает выбранным элементам. Для

удобства проектирования возможна реализация схем с использованием смешанного базиса.

Если КС реализует одну булеву функцию, то она называется *одновыходной* (рис. 3.4). КС, реализующая совокупность булевых функций, называется *многовыходной* КС.

В КС не должно быть обратных связей. Под обратной связью понимается соединение выхода некоторого логического элемента со своим входом, возможно, через цепочку других логических элементов (рис. 3.5).

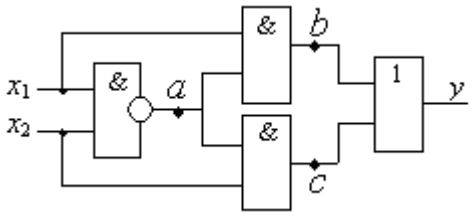


Рис. 3.4. Функциональная схема одновыходной КС

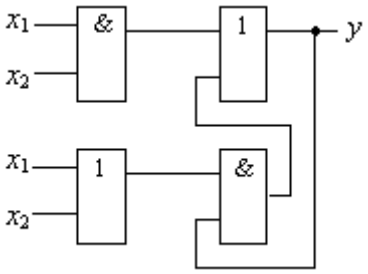


Рис. 3.5. Функциональная схема с обратной связью

Прежде чем перейти к рассмотрению приемов синтеза КС, рассмотрим решение задачи их анализа.

Задача анализа заданной КС сводится к отысканию булевой функции или системы булевых функций, описывающих работу этой КС с помощью аппарата алгебры логики.

В качестве примера построим булеву функцию, реализуемую КС, представленной на рис. 3.4. Для этого каждому логическому элементу схемы поставим в соответствие логический оператор:

$$a = \overline{x_1 \cdot x_2};\ \ {}_b = x_1 \cdot a;\ c = x_2 \cdot a;\ y = b \vee c.$$

Этим установлено однозначное соответствие между элементами схемы и ее математическим описанием.

Затем выполним последовательно операции подстановки и преобразования до тех пор, пока не получится функция, выраженная через входные переменные:

$$y = x_1 \cdot a \vee x_2 \cdot a = x_1 \cdot \overline{x_1 \cdot x_2} \vee x_2 \cdot \overline{x_1 \cdot x_2} = x_1(\overline{x_1} \vee \overline{x_2}) \vee x_2(\overline{x_1} \vee \overline{x_2}) =$$

$$= x_1\overline{x_2} \vee \overline{x_1}x_2.$$

Задача синтеза КС состоит в построении оптимальной схемы проектируемого узла устройства, исходя из физического описания его работы (технического задания на проектирование).

Основные этапы синтеза:

1. Анализ технического задания и составление таблицы истинности.
2. Минимизация логических функций.
3. Преобразование минимальных логических функций для рациональной реализации логической схемы в заданном базисе.
4. Построение функциональной схемы.
5. Проверка работоспособности схемы и её корректировка.

Минимизация логических функций выполняется методами Квайна и Мак-Класки или с помощью карт Карно. Этот этап является очень важным, так как решение одной и той же задачи синтеза может иметь очень большое число вариантов, отличающихся по сложности и быстродействию.

При реализации КС на интегральных схемах малой степени интеграции традиционными критериями минимизации являются минимальное число букв и логических операций в реализуемой функции.

Для комбинационных узлов больших интегральных схем (БИС) критерием сложности является не только число элементов на кристалле, необходимых для реализации функции узла. Большое значение приобретают морфологические свойства реализуемых схем, такие, как регулярность структуры, повторяемость элементов и связей, занимаемая площадь, минимальная длина межсоединений и т.п.

Необходимость третьего этапа обусловлена тем, что применяемые на практике комплексы ИМС имеют в своём составе такие логические элементы, как И-НЕ, ИЛИ-НЕ, И-ИЛИ-НЕ. Следовательно, минимальные функции, выраженные в базисе И, ИЛИ, НЕ, необходимо преобразовать к тому логическому базису, который отвечает выбранным элементам.

На четвёртом этапе составляется функциональная схема в заданном базисе. При этом каждой преобразованной логической функции ставится в соответствии определённый логический элемент заданного базиса. Связи между логическими элементами определяются логической функцией.

На последнем этапе производится проверка правильности работы схемы на основе алгоритма её функционирования. Решаются вопросы временного согласования сигналов при обмене информацией между элементами КС. В случае необходимости функциональная схема корректируется.

При разработке КС за *основные критерии качества технической*

реализации принимают сложность оборудования, минимум применяемых элементов, быстродействие и надёжность.

На практике при реализации КС в заданном базисе количество оборудования оценивается числом корпусов интегральных микросхем, используемых в схеме. На теоретическом уровне используется оценка сложности КС по Квайну.

Сложность (*цена*) *схемы по Квайну* определяется суммарным числом входов логических элементов в составе схемы. Например, сложность схемы, приведенной на рис. 3.4, составляет 8 единиц по Квайну.

Быстродействие оценивается максимальной задержкой сигнала при прохождении его от входа схемы к выходу. Длина самой длинной цепи из последовательно включенных элементов в схеме на рис. 3.3 равна 3, если считать, что каждый элемент задерживает сигнал на время, равное τ, то длительность ее работы будет равна 3τ.

Надёжность КС оценивается интенсивностью отказов

$$\lambda = n / N \cdot t, \qquad (2.1)$$

где n – количество элементов, вышедших из строя за период испытаний t,

N – общее количество логических элементов.

Задача синтеза всегда имеет множество решений. Из этого множества выбираются схемы, критерии качества технической реализации которых удовлетворяют заданию на проектирование.

3.1. Синтез одновыходных комбинационных схем

Схемы с одним выходом относятся к наиболее простым схемам. Задача синтеза схемы состоит в преобразовании описывающих ее логических функций в суперпозицию логических элементов заданного типа. Будем предполагать, что исходная булева функция должна быть представлена в минимальной форме: МДНФ или МКНФ.

Пример 3.1. Реализовать булеву функцию y на логических элементах: а) И, ИЛИ, НЕ; б) И-НЕ; в) ИЛИ-НЕ.

$$y = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1 x_2 x_3 \vee x_1\overline{x}_2 x_3 \vee x_1 x_2 \overline{x}_3 .$$

Решение

А. Для построения схемы на элементах И, ИЛИ, НЕ можно использовать как МДНФ, так и МКНФ функции. В этом примере будем строить схему по МДНФ.

$$y_{\text{МДНФ}} = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee x_1\overline{x_2}x_3 \vee x_1 x_2\overline{x_3} = a_1 \vee b_1 \vee c_1 \vee d_1,$$

где $a_1 = \bar{x}_1\bar{x}_2\bar{x}_3$, $b_1 = \bar{x}_1 x_2 x_3$, $c_1 = x_1\overline{x_2}x_3$, $d_1 = x_1 x_2\overline{x_3}$.

В этом случае функция представляется в виде суперпозиции операторов логических элементов И (конъюнкторов), это a_1, b_1, c_1, и d_1, и оператора логического элемента ИЛИ. Соответствующая схема изображена на рис. 3.6.

Б. Для реализации исходной булевой функции на элементах И-НЕ необходимо от МДНФ функции взять двойное отрицание и одно из них раскрыть по правилу Де Моргана, избавляясь от дизъюнкции между элементарными конъюнкциями.

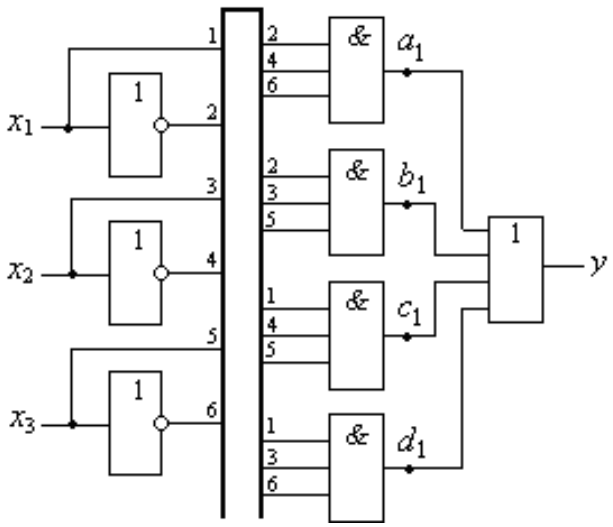


Рис. 3.6. Пример функциональной схемы реализации булевой функции на элементах И, ИЛИ, НЕ

Преобразуем функцию $y_{\text{МДНФ}}$ для реализации в базисе И-НЕ:

$$y = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee x_1\overline{x_2}x_3 \vee x_1 x_2\overline{x_3} =$$

$$= \overline{\bar{x}_1\bar{x}_2\bar{x}_3} \vee \overline{\bar{x}_1 x_2 x_3} \vee \overline{x_1\overline{x_2}x_3} \vee \overline{x_1 x_2\overline{x_3}} = \overline{\bar{x}_1\bar{x}_2\bar{x}_3 \cdot \bar{x}_1 x_2 x_3 \cdot x_1\overline{x_2}x_3 \cdot x_1 x_2\overline{x_3}} =$$

$$= \overline{a_2 b_2 c_2 d_2},$$

где $a_2 = \overline{\bar{x}_1\bar{x}_2\bar{x}_3}$, $b_2 = \overline{\bar{x}_1 x_2 x_3}$, $c_2 = \overline{x_1\overline{x_2}x_3}$, $d_2 = \overline{x_1 x_2\overline{x_3}}$.

В этом случае функция представлена в виде суперпозиции только операторов И-НЕ. Для реализации инверторов можно использовать двухвходовые элементы И-НЕ (2И-НЕ). На рис. 3.7,а,б приведены способы использования элемента 2И-НЕ в качестве инвертора.

В общем случае если элемент типа И-НЕ имеет n входов, а требуется использовать из них только k входов, то оставшиеся $(n-k)$ можно "отключить" одним из способов, показанных на рис. 3.7.

В схеме на элементах И-НЕ, приведенной на рис. 3.8, использован способ, показанный на рис. 3.7,б.

а) *б)*

Рис. 3.7. Способы использования элемента типа 2И-НЕ в качестве инвертора

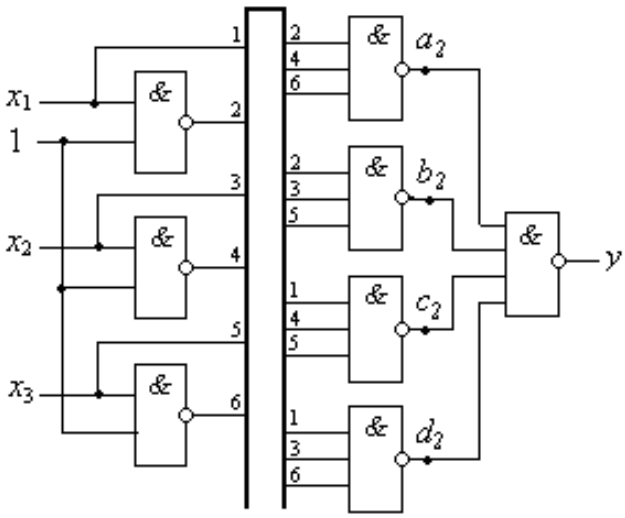


Рис. 3.8. Пример функциональной схемы реализации булевой функции на элементах И-НЕ

В. Для реализации исходной булевой функции на элементах ИЛИ-НЕ необходимо от МКНФ функции взять двойное отрицание и одно из них раскрыть по правилу Де Моргана, избавляясь от конъюнкции между элементарными дизъюнкциями.

Найдем МКНФ функции y:

$$y_{\text{МКНФ}} = (x_1 \vee x_2 \vee \overline{x_3})(x_1 \vee \overline{x_2} \vee x_3)(\overline{x_1} \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}).$$

Преобразуем функцию $y_{\text{МКНФ}}$ для реализации в базисе ИЛИ-НЕ:

$$y = (x_1 \vee x_2 \vee \overline{x_3})(x_1 \vee \overline{x_2} \vee x_3)(\overline{x_1} \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) =$$

$$= \overline{\overline{(x_1 \vee x_2 \vee \overline{x_3})(x_1 \vee \overline{x_2} \vee x_3)(\overline{x_1} \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})}} =$$

$$= \overline{\overline{x_1 \vee x_2 \vee \overline{x_3}} \vee \overline{x_1 \vee \overline{x_2} \vee x_3} \vee \overline{\overline{x_1} \vee x_2 \vee x_3} \vee \overline{\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}}} =$$

$$= \overline{a_3 \vee b_3 \vee c_3 \vee d_3},$$

где $a_3 = \overline{\overline{x_1 \vee x_2 \vee x_3}}$, $b_3 = \overline{x_1 \vee \overline{x_2} \vee x_3}$, $c_3 = \overline{\overline{x_1} \vee x_2 \vee x_3}$, $d_3 = \overline{\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}}$.

В этом случае функция представлена в виде суперпозиции только операторов элементов ИЛИ-НЕ.

Для реализации инверторов можно использовать двухвходовые элементы ИЛИ-НЕ (2ИЛИ-НЕ). На рис. 3.9,а,б приведены способы использования элемента 2ИЛИ-НЕ в качестве инвертора. В схеме на элементах ИЛИ-НЕ, приведенной на рис. 3.10, использован способ, показанный на рис. 3.9,б.

а) *б)*

Рис. 3.9. Способы использования элемента типа 2ИЛИ-НЕ в качестве инвертора

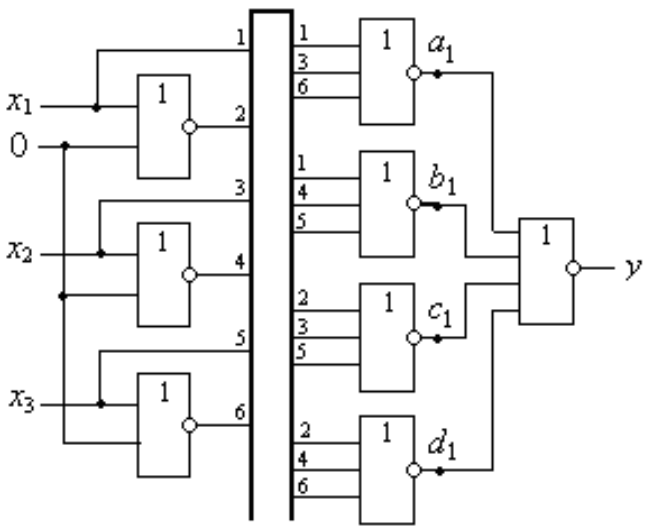


Рис. 3.10. Пример функциональной схемы реализации булевой функции на элементах ИЛИ-НЕ

3.1.1. Оценка основных критериев качества технической реализации

Как было сказано выше, за *основные критерии качества технической реализации* КС принимают сложность оборудования и быстродействие. Определим эти критерии для схем на рис. 3.6, 3.8 и 3.10.

На теоретическом уровне *сложность* (*цена*) *схемы* определяется в единицах по Квайну, т.е. суммарным числом входов логических элементов в составе схемы. Так, сложность схемы, приведенной на рис. 3.6, составляет 19 единиц по Квайну; схем, приведенных на рис. 3.8 и 3.10, – 21 единицу по Квайну.

Быстродействие, или длительность работы схемы оценивается самой длинной цепью из последовательно включенных элементов. Будем называть **глубиной схемы** количество последовательно включенных элементов. Если известна глубина схемы k, то длительность ее работы оценивается как $t = k \cdot \tau$, где τ – длительность задержки сигнала логическим элементом.

Схемы, приведенные на рис. 3.6, 3.8 и 3.10, имеют глубину, равную трем. Если считать, что каждый элемент задерживает сигнал на время, равное τ, то длительность ее работы будет равна 3τ.

3.1.2. Явление риска логических схем

Реальный логический элемент переключается за какое-то конечное время, зависящее от технологии изготовления, условий эксплуатации, емкостей нагрузки и т.д. Прохождение сигнала последовательно через несколько логических элементов будет приводить к накоплению времени задержки и возникновению сдвига во времени выходного сигнала по отношению к входному. Наличие задержки и порождаемого ею временного сдвига сигналов может приводить к появлению на выходе отдельных логических элементов и всей схемы в целом кратковременных сигналов, не предусмотренных булевой функцией, реализуемой схемой. Такое явление называется риском сбоя логических схем.

Различают статический и динамический риски сбоя.

При *статическом риске сбоя* до и после переходного процесса состояние выходного сигнала одно и то же, а во время переходного процесса возможно кратковременное появление противоположного сигнала.

При *динамическом риске сбоя* до и после переходного процесса состояния выходного сигнала противоположные, но в переходном процессе выходной сигнал несколько раз меняет свое значение.

Радикальным способом устранения рисков сбоя является введение стробирования. Стробирующий импульс подается, когда на выходе КС уже установилось необходимое значение выходного сигнала, что исключает влияние возможных сбоев на вырабатываемый схемой сигнал.

При синтезе КС необходимо учитывать *состязания в КС*. В реальных КС из-за ненулевой задержки сигналов в логических элементах могут возникнуть ситуации, когда разница в этих паразитных задержках может привести для асинхронных схем к неправильной их работе. Неправильная работа таких схем называется *переходным состязанием.* Например, неодновременность изменения прямого и инверсного значения переменных на входах логического элемента приводит в какой-то короткий промежуток времени Δt к тому, что значения x_i и $\overline{x_i}$ будут совпадать, что в свою очередь вызовет кратковременную ошибку на выходе проектируемой схемы.

Для устранения переходного состязания при условии, что на входе КС в любой момент времени изменяется только одна переменная, достаточно реализовать не МДНФ, а сокращенную ДНФ функции, т.е. в реализуемую функцию добавляют простые импликанты, покрывающие переходы от одного набора переменных к другому. Например, при реализации МДНФ функции $y = x_1x_2 \vee \overline{x_1}x_3$ необходимо добавить импликанту x_2x_3, т. е. для предотвращения переходных состязаний необходимо реализовать сокращенную ДНФ $y = x_1x_2 \vee \overline{x_1}x_3 \vee x_2x_3$.

3.2. Синтез многовыходных комбинационных схем

Многовыходная КС реализует несколько булевых функций, число которых равно числу выходов схемы. Задача синтеза такой схемы может быть решена [7]:

− путем независимой реализации каждой булевой функции. Такое построение схемы во многих случаях неэкономично, т.к. некоторые функции могут иметь общие члены;

− путем совместной реализации булевых функций.

Рассмотрим некоторые из этих способов.

1. Выполняя совместную минимизацию системы булевых функций можно добиться сокращения общего числа логических элементов для построения схемы.

Пример 3.2. Синтезировать КС преобразователя двоично-десятичного кода 8421 в код 2421 на логических элементах базиса:

а) И-НЕ, б) ИЛИ-НЕ.

Табличное описание преобразователя приведено в табл. 3.1, условное графическое обозначение – на рис. 3.11.

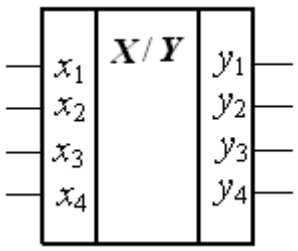


Рис. 3.11. Преобразователь.
Условное графическое обозначение

Решение

Из таблицы истинности преобразователя кодов (табл. 3.1) следует, что функции y_1, y_2, y_3, y_4 не полностью определенные. С целью минимизации доопределяем значения выходных функций на некоторых избыточных входных наборах, которые показаны на картах Карно звездочкой (рис. 3.12, 3.14).

Таблица 3.1. Таблица истинности преобразователя кодов

Десятичное число	Код 8 4 2 1	Код 2 4 2 1
	$x_1\ x_2\ x_3\ x_4$	$y_1\ y_2\ y_3\ y_4$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
8	1 0 0 0	1 1 1 0
9	1 0 0 1	1 1 1 1

А. Для реализации КС преобразователя кодов на элементах базиса И-НЕ определяем МДНФ функций y_1, y_2, y_3, y_4, для чего проведем их

совместную минимизацию (рис. 3.12).

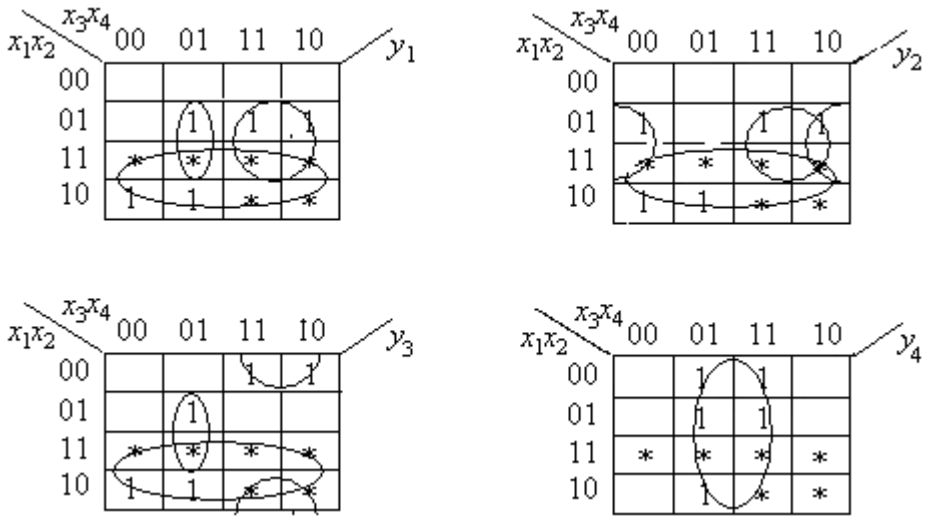


Рис. 3.12. Минимизация функций y_1, y_2, y_3, y_4: получение МДНФ

После минимизации функции могут быть записаны в виде системы уравнений:

$$y_1 = x_1 \vee x_2x_3 \vee x_2\overline{x_3}x_4;$$
$$y_2 = x_1 \vee x_2x_3 \vee x_2\overline{x_4};$$
$$y_3 = x_1 \vee x_2\overline{x_3}x_1 \vee \overline{x_2}x_3;$$
$$y_4 = x_1.$$

Выполним преобразование МДНФ функций для их реализации в базисе И-НЕ:

$$y_1 = \overline{\overline{x_1 \vee x_2x_3 \vee x_2\overline{x_3}x_4}} = \overline{x_1} \cdot \overline{x_2x_3} \cdot \overline{x_2\overline{x_3}x_4};$$
$$y_3 = \overline{\overline{x_1 \vee x_2x_3 \vee x_2\overline{x_4}}} = \overline{x_1} \cdot \overline{x_2x_3} \cdot \overline{x_3x_4};$$
$$y_2 = \overline{\overline{x_1 \vee x_2x_3x_1 \vee x_2x_3}} = \overline{x_1} \cdot \overline{x_2\overline{x_3}x_4} \cdot \overline{x_2x_3};$$
$$y_4 = x_4.$$

Схема преобразователя кодов на элементах И-НЕ приведена на рис. 3.13.

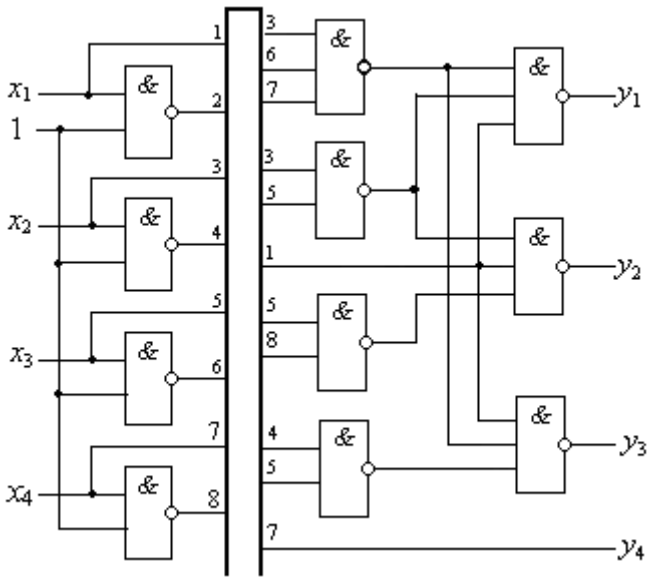


Рис. 3.13. Функциональная схема реализации преобразователя кода 8421 в код 2421 на элементах И-НЕ

Б. Для реализации КС преобразователя кодов на элементах базиса ИЛИ-НЕ определим МКНФ функций y_4, y_3, y_2, y_1, выполнив их совместную минимизацию (рис. 3.14).

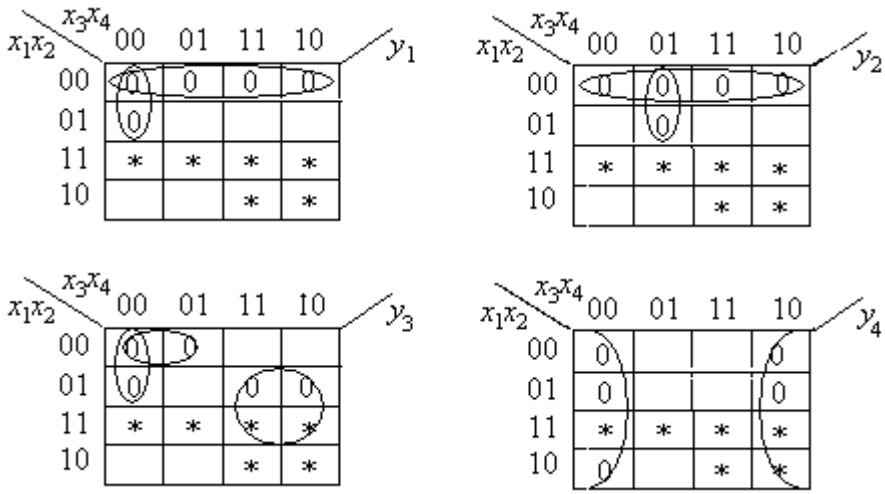


Рис. 3.14. Минимизация функций y_1, y_2, y_3, y_4: получение МКНФ

После минимизации получим следующую систему уравнений:

$$y_1 = (x_1 \vee x_2)(x_1 \vee x_3 \vee x_4);$$

$$y_2 = (x_1 \vee x_2)(x_1 \vee x_3 \vee \overline{x_4});$$

$$y_3 = (x_1 \vee x_3 \vee x_4)(x_1 \vee x_2 \vee x_3)(\overline{x_2} \vee \overline{x_3});$$

$$y_1 = x_4.$$

Выполним преобразования МКНФ функций для реализации схемы преобразователя в базисе ИЛИ-НЕ:

$$y_1 = \overline{\overline{(x_1 \vee x_2)(x_1 \vee x_3 \vee x_4)}} = \overline{\overline{x_1 \vee x_2} \vee \overline{x_1 \vee x_3 \vee x_4}};$$

$$y_2 = \overline{\overline{(x_1 \vee x_2)(x_1 \vee x_3 \vee \overline{x_4})}} = \overline{\overline{x_1 \vee x_2} \vee \overline{x_1 \vee x_3 \vee \overline{x_4}}};$$

$$y_3 = \overline{\overline{(x_1 \vee x_3 \vee x_4)(x_1 \vee x_2 \vee x_3)(\overline{x_2} \vee \overline{x_3})}} =$$

$$= \overline{\overline{x_1 \vee x_3 \vee x_4} \vee \overline{x_1 \vee x_2 \vee x_3} \vee \overline{\overline{x_2} \vee \overline{x_3}}};$$

$$y_4 = x_4.$$

Схема преобразователя кодов на элементах базиса ИЛИ-НЕ приведена на рис. 3.15.

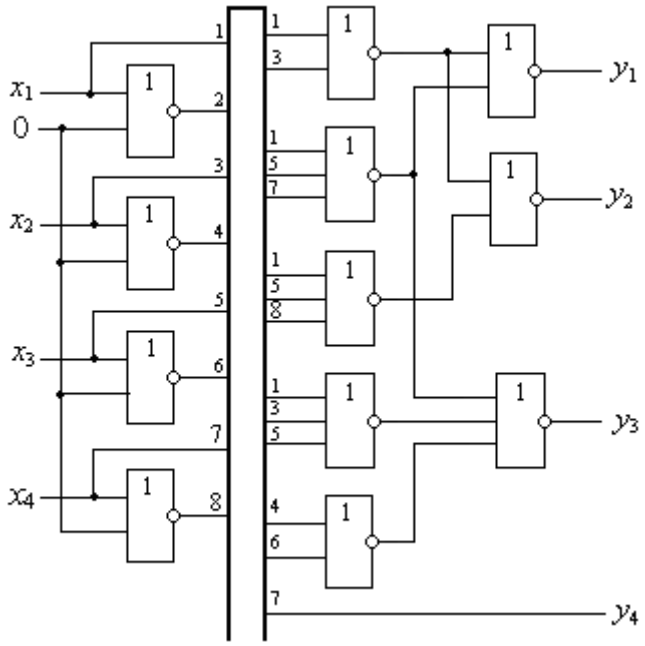


Рис. 3.15. Функциональная схема реализации преобразователя кода 8421 в код 2421 на элементах ИЛИ-НЕ

Количество оборудования (цена схемы) для реализации схемы на элементах И-НЕ (рис. 3.13) составит 26 единиц по Квайну, а цена схемы на элементах ИЛИ-НЕ (рис. 3.15) – 28 единиц по Квайну. Быстродействие в обоих случаях составит 3τ.

Рассмотренный пример наглядно показал, что использование совместной минимизации системы булевых функций позволяет упростить схему преобразователя кодов.

2. Для сокращения общего числа элементов для построения схемы можно выразить булеву функцию через другую.

Пример 3.3. Для заданных булевых функций найти соотношения, выражающие одну функцию через другую:

$$y_1 = \vee(0,1) = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3;$$

$$y_2 = \vee(0,1,7) = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 \vee x_1 x_2 x_3;$$

$$y_3 = \vee(0,1,5,7) = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 x_3.$$

Решение

В результате анализа и минимизации этих функций получим:

$$y_1 = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 = \overline{x_1 x_2};$$

$$y_2 = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 \vee x_1 x_2 x_3 = \overline{x_1 x_2} \vee x_1 x_2 x_3 = y_1 \vee x_1 x_2 x_3;$$

$$y_3 = \overline{x_1 x_2 x_3} \vee \overline{x_1 x_2} x_3 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 x_3 = \overline{x_1 x_2} \vee x_1 x_3 = y_1 \vee x_1 x_3.$$

3. Уменьшить стоимость схемы можно, используя промежуточную функцию, которая определяется как общая составляющая МДНФ нескольких функций.

Пример 3.4. Для заданных булевых функций выделить общую промежуточную функцию и выразить их через нее:

$$y_1 = x_1 \overline{x_2} x_3 \vee x_1 x_2 \overline{x_3} \vee x_2 x_4;$$

$$y_2 = \overline{x_2} x_3 \overline{x_4} \vee x_2 \overline{x_3 x_4} \vee \overline{x_1} x_4.$$

Решение

Промежуточная функция $y_3 = \overline{x_2} x_3 \vee x_2 \overline{x_3}$. С учетом этого реализации подлежит следующая система булевых функций:

$$y_1 = x_1 y_3 \vee x_2 x_4;$$

$$y_2 = \overline{x_4} y_3 \vee x_1 x_4;$$

$$y_3 = \overline{x_2} x_3 \vee x_2 \overline{x_3}.$$

Функция y_3 реализуется один раз и используется при построении функций y_1 и y_2.

Пример 3.5. Используя промежуточную функцию, реализовать

КС одноразрядного сумматора.

Решение

Условное графическое обозначение и таблица истинности одноразрядного сумматора приведены на рис. 3.16,а,б.

Для минимизации функций S_i, P_i отобразим их на карте Карно (рис. 3.16,в). Функция S_i минимизации не подлежит, поэтому запишем ее в совершенной дизъюнктивной нормальной форме (СДНФ):

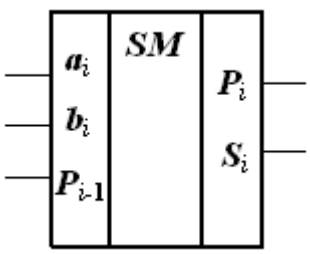
$$S_i = \overline{a_i} \cdot \overline{b_i} \cdot P_{i-1} \vee \overline{a_i} \cdot b_i \cdot \overline{P_{i-1}} \vee a_i \cdot \overline{b_i} \cdot \overline{P_{i-1}} \vee a_i \cdot b_i \cdot P_{i-1}.$$

Функцию P_i после минимизации можно записать в виде

$$P_i = a_i \cdot b_i \vee a_i \cdot P_{i-1} \vee b_i \cdot P_{i-1} = a_i \cdot b_i \vee P_{i-1} \cdot (a_i \vee b_i).$$

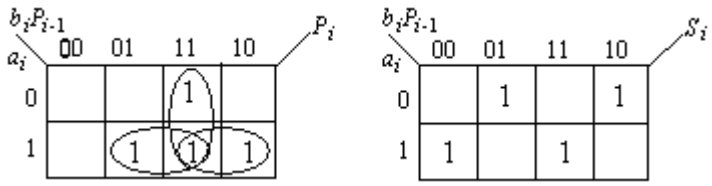
Для реализации функции S_i используем промежуточную функцию $C_i = \overline{a_i b_i} \vee a_i b_i$.

$$S_i = P_{i-1} \cdot \left(\overline{a_i} \cdot \overline{b_i} \vee a_i \cdot b_i \right) \vee \overline{P_{i-1}} \cdot \left(\overline{a_i} \cdot b_i \vee a_i \cdot \overline{b_i} \right) = P_{i-1} \cdot C_i \vee \overline{P_{i-1}} \cdot \overline{C_i}.$$



a_i	b_i	P_{i-1}	P_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

а) *б)*



в)

Рис. 3.16. Одноразрядный сумматор:
а – условное графическое обозначение; б – таблица истинности;
в – карта Карно

Полученные выражения для S_i, C_i, P_i преобразуем к виду, удобному для реализации на элементах базиса И-НЕ:

$$C_i = \overline{a_i b_i} \vee a_i b_i = \overline{\overline{a_i \cdot b_i} \vee a_i \cdot b_i} = \overline{\overline{a_i} \cdot \overline{b_i} \cdot a_i b_i},$$

$$S_i = P_{i-1} C_i \vee \overline{P_{i-1} C_i} = \overline{\overline{P_{i-1} \cdot C_i} \vee P_{i-1} \cdot C_i} = \overline{\overline{P_{i-1} \cdot C_i} \cdot P_{i-1} \cdot C_i},$$

$$P_i = a_i b_i \vee P_{i-1}(a_i \vee b_i) = a_i b_i \vee P_{i-1} \overline{\left(\overline{a_i \vee b_i} \right)} = \overline{a_i b_i} \cdot \overline{P_{i-1} \overline{a_i} \cdot \overline{b_i}}.$$

Схема одноразрядного сумматора на элементах базиса И-НЕ представлена на рис. 3.17.

4. В некоторых случаях можно достичь упрощения исходных функций, применяя способ, основанный на использовании построенной функции в качестве дополнительной входной переменной при построении другой функции. В качестве исходной функции принимается та, которая допускает наиболее простую реализацию.

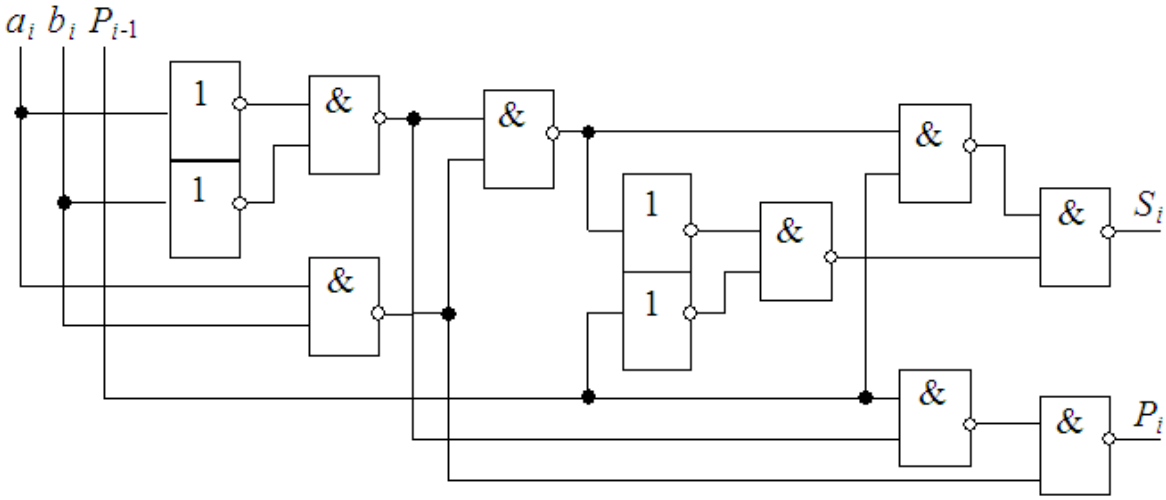


Рис. 3.17. Функциональная схема одноразрядного сумматора

Пример 3.6. Используя предложенную методику, упростить схемную реализацию функции S_i КС одноразрядного сумматора.

Решение

Для упрощения схемной реализации функции S_i выполним следующие преобразования. Представим S_i как функцию четырех переменных: a_i, b_i, P_{i-1} и P_i. Для этого составим таблицу истинности (рис. 3.18,а), где на нереальных входных наборах функция S_i принимает неопределенные значения (*). После минимизации с помощью карты Карно (рис. 3.18,в) запи-

шем:

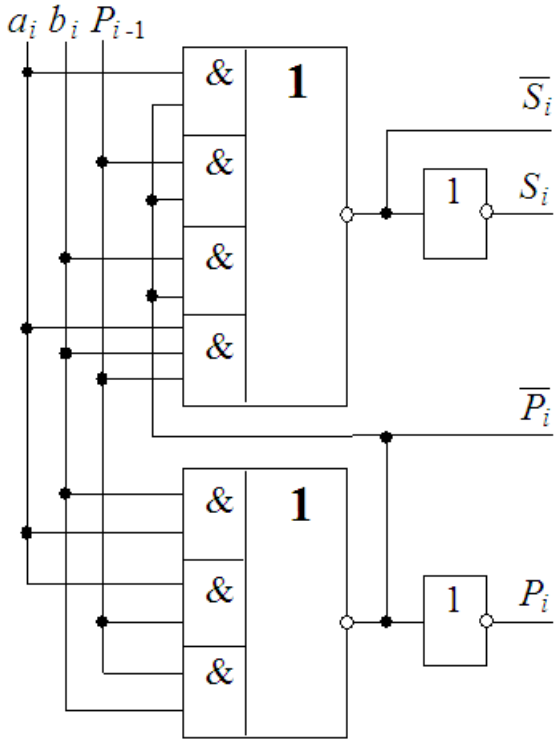
$$S_i = a_i \overline{P_i} \vee P_{i-1} \overline{P_i} \vee b_i \overline{P_i} \vee a_i b_i P_{i-1}.$$

Используя полученные выражения для суммы и полученное в примере 3.5 выражение для переноса, легко построить схему одноразрядного сумматора, которая значительно проще исходной при независимой реализации (рис. 3.18,б).

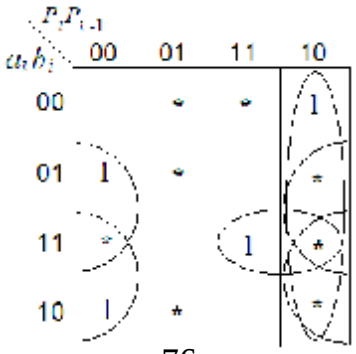
5. Значительного упрощения многовыходных комбинационных схем можно достичь, применяя метод каскадов, когда часть элементов схемы используется одновременно для реализации нескольких функций [4]. Этот метод эффективен, например, при построении дешифраторов пирамидального и прямоугольного типов.

a_i	b_i	P_{i-1}	P_i	S_i
0	0	0	0	0
0	0	0	1	*
0	0	1	0	**1**
0	0	1	1	*
0	1	0	0	**1**
0	1	0	1	*
0	1	1	0	*
0	1	1	1	**0**
1	0	0	0	**1**
1	0	0	1	*
1	0	1	0	*
1	0	1	1	**0**
1	1	0	0	*
1	1	0	1	0
1	1	1	0	*
1	1	1	1	1

а)



б)



в)

Рис. 3.18. Одноразрядный сумматор:

в) минимизация функции Si

3.3. Основы синтеза КС с использованием логических элементов малой степени интеграции

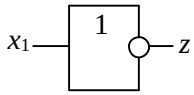
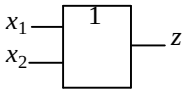
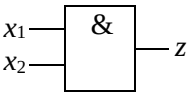
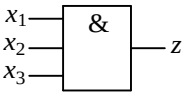
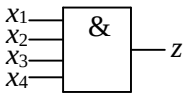
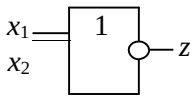
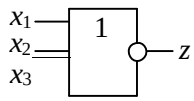
В качестве примера будем использовать логические элементы, входящие в состав интегральных схем серии 555. Их условно можно разделить на четыре группы:

1) реализующие элементарные логические функции И, ИЛИ, НЕ;
2) реализующие логические функции И-НЕ;
3) реализующие логические функции ИЛИ-НЕ;
4) реализующие логические функции И-ИЛИ-НЕ.

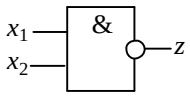
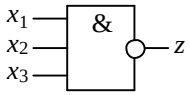
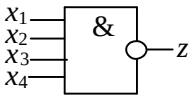
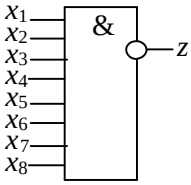
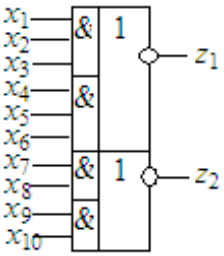
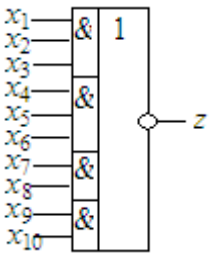
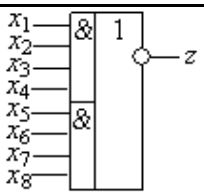
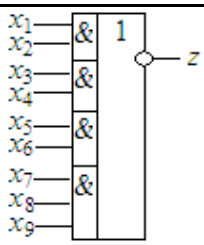
Поскольку все эти схемы расположены в стандартном корпусе, то количество логических элементов в одной конструктивной единице различно.

В таблице 3.2 приведены некоторые логические элементы серии 555.

Таблица 3.2. Некоторые логические элементы серии 555

Условное обозначение	**Графическое обозначение**	**Реализуемая функция**	**Количество ЛЭ в корпусе**
1	2	3	4
К555ЛН1		$z = \overline{x}$ НЕ	6
К555ЛЛ1		$z = x_1 \vee x_2$ 2ИЛИ	4
К555ЛИ1		$z = x_1 x_2$ 2И	4
К555ЛИ3		$z = x_1 x_2 x_3$ 3И	3
К555ЛИ6		$z = x_1 x_2 x_3 x_4$ 4И	2
К555ЛЕ1		$z = \overline{x_1 \vee x_2}$ 2ИЛИ-НЕ	4
К555ЛЕ4		$z = \overline{x_1 \vee x_2 \vee x_3}$ 3ИЛИ-НЕ	3

Окончание таблицы 3.2

1	2	3	4
К555ЛА3		$z = \overline{x_1 x_2}$ 2И-НЕ	4
К555ЛА4		$z = \overline{x_1 x_2 x_3}$ 3И-НЕ	3
К555ЛА1		$z = \overline{x_1 x_2 x_3 x_4}$ 4И-НЕ	2
К555ЛА2		$z = \overline{x_1 x_2 x_3 x_4 \,\&\, x_5 x_6 x_7 x_8}$ 8И-НЕ	1
К555ЛР11		$z_1 = \overline{x_1 x_2 x_3 \vee x_4 x_5 x_6}$ 3-3И-ИЛИ-НЕ $z_2 = \overline{x_7 x_8 \vee x_9 x_{10}}$ 2-2И-ИЛИ-НЕ	1
К555ЛР13		$z = \overline{x_1 x_2 \vee x_3 x_4 x_5 \vee x_6 x_7 x_8 \vee x_9 x_{10}}$ 3-2-2-3И-ИЛИ-НЕ	1
К555ЛР4		$z = \overline{x_1 x_2 x_3 x_4 \vee x_5 x_6 x_7 x_8}$ 4-4И-ИЛИ-НЕ	1
К555ЛР3		$z = \overline{x_1 x_2 \vee x_3 x_4 \vee x_5 x_6 \vee x_7 x_8 x_9}$ 2-2-2-3И-ИЛИ-НЕ	1

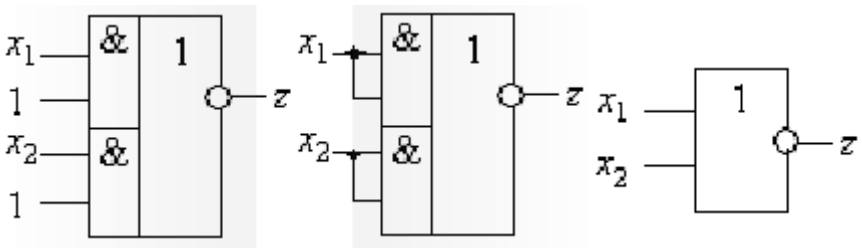
Способы использования логических элементов серии К555

Из логических формул, описывающих функции, реализуемые ло-

гические элементы серии 555 (см. табл. 3.2), следует, что если заменить некоторые входные переменные x_i логическими константами “1” и “0”, то можно получить как бы новые, отсутствующие в комплексе типов логических элементов.

Особенности использования входов логических элементов базисов И-НЕ и ИЛИ-НЕ были рассмотрены в разделе 3.1 (см. рис. 3.6, 3.7). Рассмотрим особенности использования входов элемента типа И-ИЛИ-НЕ.

1. Учитывая рассмотренные выше особенности использования входов одной и той же схемы И, можно указать два варианта включения логического элемента типа И-ИЛИ-НЕ в режиме ИЛИ-НЕ (рис. 3.19).



а) *б)* *в)*

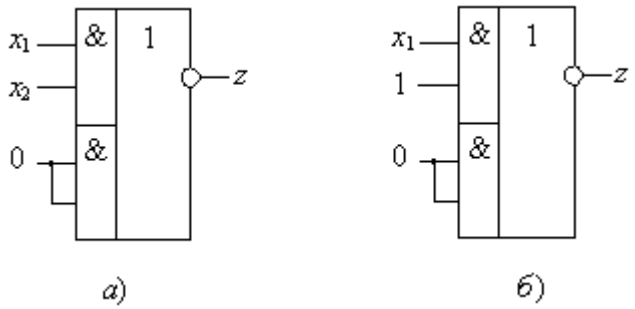
Рис. 3.19. Варианты схем получения ЛЭ типа ИЛИ-НЕ (а, б) и эквивалентное изображение включения (в)

2. Использование логического элемента И-ИЛИ-НЕ в качестве схемы типа И-НЕ. В качестве примера используем КЛЭ типа 2-2И-2ИЛИ-НЕ. На входы одной из схем И достаточно задать логический нуль, а вторую схему применить по прямому назначению (рис. 3.20,а). Очевидно, что всё изложенное можно распространить и на другие логические элементы типа И-ИЛИ-НЕ.

3. Логические элементы И-ИЛИ-НЕ можно использовать в качестве инвертора, что доказывает приведённое ниже уравнение:

$$z = \overline{x_1 x_2 \vee x_3 x_4} = \overline{x_1 \cdot 1 \vee 0} = \overline{x_1}.$$

Вариант такого включения показан на рис. 3.20,б.



а) *б)*

Рис. 3.20. Схемы использования КЛЭ типа И-ИЛИ-НЕ:
а) в качестве элемента типа И-НЕ; б) в качестве инвертора

3.4. Некоторые приемы преобразования функций для реализации на элементах заданного типа

1. При синтезе КС на основе реальных элементов необходимо учитывать их ограниченное число входов и нагрузочную способность. Такое ограничение иногда не позволяет реализовать исходную булеву функцию в виде структуры логических элементов, состоящей только из двух уровней.

Пример 3.7. Реализовать функцию $Y(x_1, x_2, x_3, x_4)$ на элементах 2ИЛИ-НЕ серии 555:

$$Y = (x_1 \vee x_2)(x_1 \vee x_4)(x_1 \vee \overline{x}_3)(x_2 \vee \overline{x}_3).$$

Решение

Функция $Y(x_1, x_2, x_3, x_4)$ задана в МКНФ. Для реализации на элементах ИЛИ-НЕ берём двойное отрицание от МКНФ и одно из них раскрываем по правилу де Моргана, получаем:

$$Y = \overline{\overline{(x_1 \vee x_2)(x_1 \vee x_4)(x_1 \vee \overline{x}_3)(x_2 \vee \overline{x}_3)}} =$$

$$= \overline{\overline{x_1 \vee x_2}} \vee \overline{\overline{x_1 \vee x_4}} \vee \overline{\overline{x_1 \vee \overline{x}_3}} \vee \overline{\overline{x_2 \vee \overline{x}_3}}.$$

С целью уменьшения числа слагаемых воспользуемся правилом двойного отрицания: $\overline{\overline{A}} = A$:

$$Y = \overline{\overline{x_1 \vee x_2}} \vee \overline{\overline{x_1 \vee x_4}} \vee \overline{\overline{x_1 \vee \overline{x}_3}} \vee \overline{\overline{x_2 \vee \overline{x}_3}}.$$

Реализация полученной функции приведена на рис. 3.21.

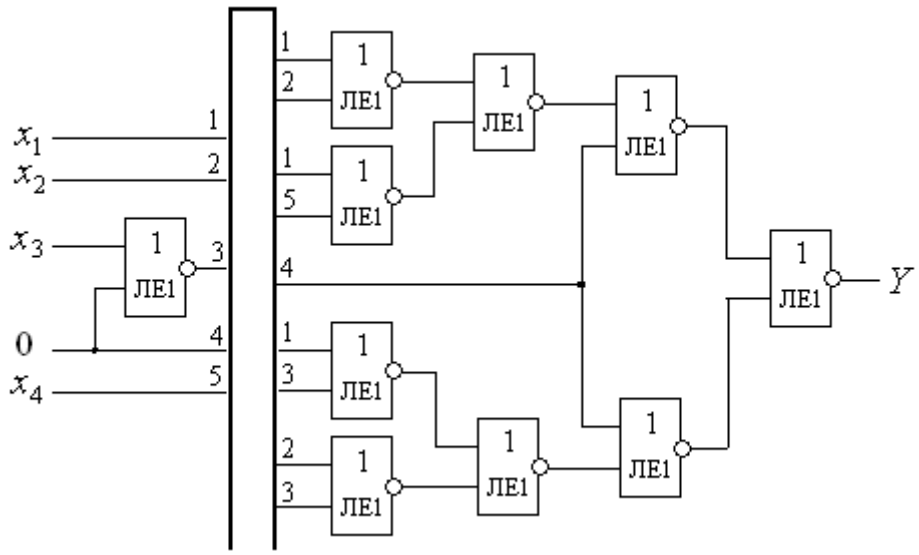


Рис. 3.21. Функциональная схема реализации функции из примера 3.7

2. При реализации довольно сложных булевых функций указанными приемами иногда получаются выражения, которые непосредственно не реализуются на КЛЭ заданного типа. В этом случае *применяют метод тождественных преобразований с предварительной группировкой*.

Пример 3.8. Реализовать функцию $Y(x_1, x_2, x_3, x_4, x_5)$ на элементах И-ИЛИ-НЕ серии 555:

$$Y = x_1\overline{x}_2x_4 \vee x_1\overline{x}_3\overline{x}_5 \vee x_2\overline{x}_4x_5 \vee \overline{x}_3\overline{x}_4x_5.$$

Решение

С помощью карты Карно (рис. 3.22,а) находим МКНФ функции (МДНФ отрицания функции Y), склеивая нули:

$$Y = \overline{\overline{x}_1x_4 \vee \overline{x}_1\overline{x}_5 \vee x_2x_4x_5 \vee x_2x_3\overline{x}_5 \vee \overline{x}_2x_3\overline{x}_4}.$$

Полученное выражение непосредственно не реализуется на логических элементах типа И-ИЛИ-НЕ, имеющихся в серии 555. Поэтому необходимо выполнить преобразование МКНФ с целью уменьшения числа термов под отрицанием. Сгруппируем два последних A и получим:

$$Y = \overline{\overline{x}_1x_4 \vee \overline{x}_1\overline{x}_5 \quad x_2x_4x_5 \vee x_3\left(x_2\overline{x}_5 \vee \overline{x}_2\overline{x}_4\right)} = \overline{\overline{x}_1x_4 \vee \overline{x}_1\overline{x}_5 \vee x_2x_4x_5 \vee x_3A},$$

где $A = x_2\overline{x}_5 \vee \overline{x}_2\overline{x}_4$.

Для $A(x_2, x_4, x_5)$ еще раз применим карту Карно (рис. 3.22,б), с целью приведения этой булевой функции к форме отрицания ДНФ, получим

$$A = \overline{x_2 x_5 \vee \bar{x}_2 x_4}\,.$$

В итоге получили булеву функцию, легко реализуемую на элементах типа И-ИЛИ-НЕ: К555ЛР11 и К555ЛР3 (рис. 3.22,в).

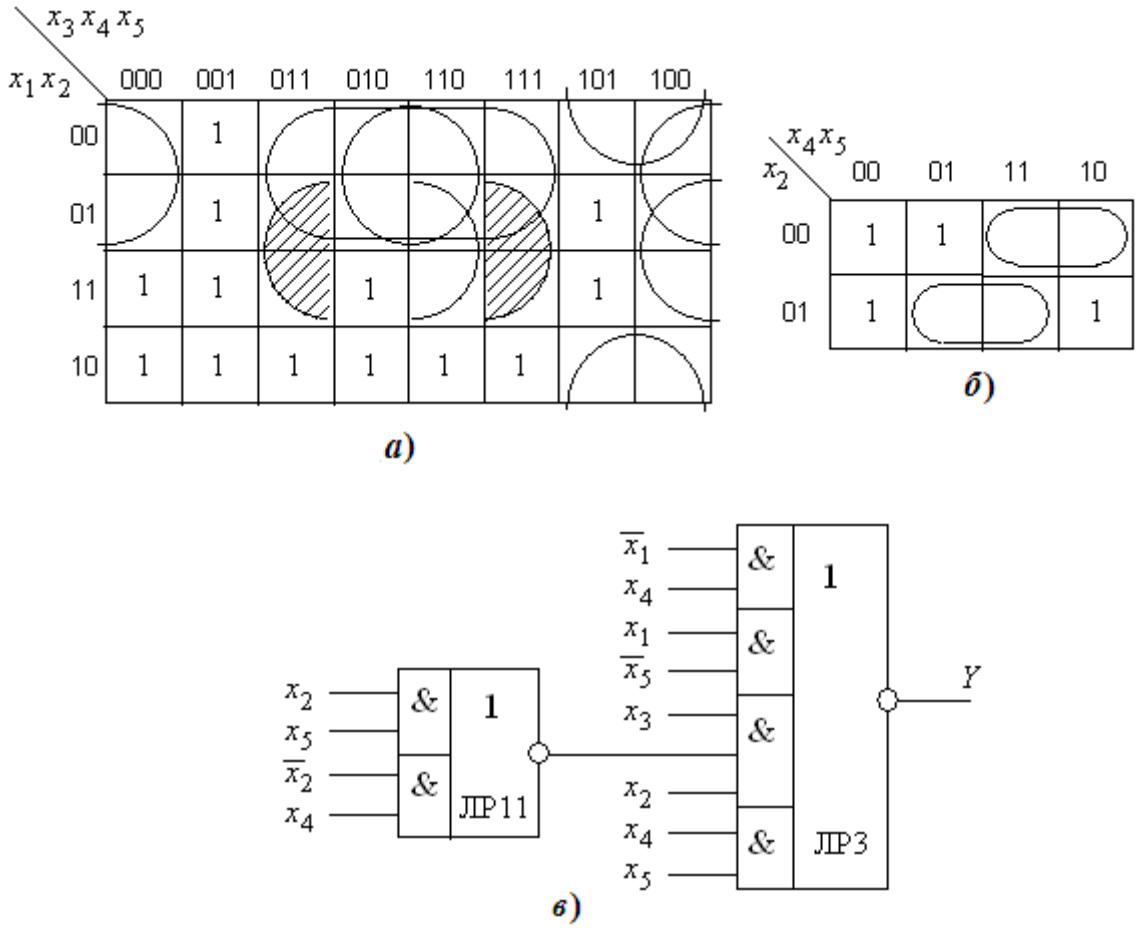


Рис. 3.22. Карты Карно (а, б) и функциональная схема (в) к примеру 3.8

3. *Факторизационный метод синтеза КС* используется для синтеза КС на элементах с меньшим числом входов, чем это требуется по исходной булевой функции. Этот метод базируется на преобразовании исходной булевой функции путем выноса за скобку общих логических переменных. В этом случае схема реализации булевой функции будет состоять из более чем двух уровней, что приведет к уменьшению ее быстродействия.

Пример 3.9. Реализовать функцию $Y(x_1, x_2, x_3)$ на элементах 2И-НЕ (К555ЛА3).

$$Y = \bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 x_3\,.$$

Решение

Преобразуем функцию Y в суперпозицию заданных элементов следующим образом:

$$Y = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2x_3 =$$
$$= \bar{x}_1 \cdot \left(\bar{x}_2x_3 \vee x_2\bar{x}_3\right) \vee x_1 \cdot \left(\bar{x}_2\bar{x}_3 \vee x_2x_3\right) = \bar{x}_1 \cdot A \vee x_1 \cdot \bar{A},$$

где $A = \bar{x}_2x_3 \vee x_2\bar{x}_3$.

От функций Y и A берем двойное отрицание, одно их которых раскрываем по правилу де Моргана:

$$Y = \overline{\overline{\bar{x}_1A \vee x_1\bar{A}}} = \overline{\overline{\bar{x}_1A} \cdot \overline{x_1\bar{A}}}; \qquad A = \overline{\overline{\bar{x}_2x_3 \vee x_2\bar{x}_3}} = \overline{\overline{\bar{x}_2x_3} \cdot \overline{x_2\bar{x}_3}}.$$

Полученные выражения позволяют реализовать схему с использованием элементов К555ЛА3 (рис. 3.23).

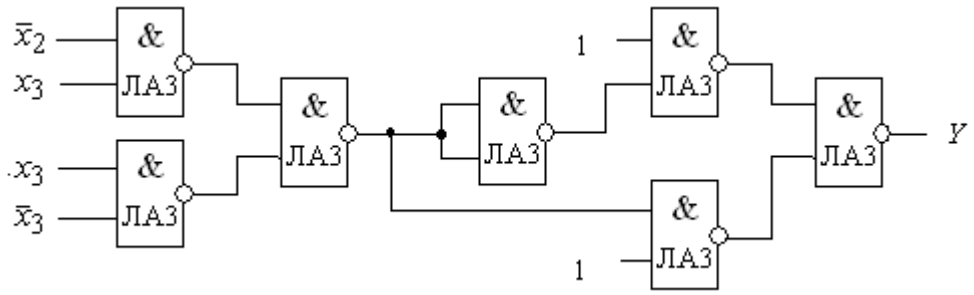


Рис. 3.23. Функциональная схема к примеру 3.9

4. В ряде случаев требуется построить КС, не используя инверсии входных переменных. С этой целью выполняются преобразования исходной функции, рассмотренной в примере 3.10.

Пример 3.10. Реализовать функцию $Y(x_1, x_2)$ на элементах серии 555, не используя инверсии входных переменных.

$Y = x_1\bar{x}_2 \vee \bar{x}_1x_2$.

Решение

Используя правило поглощения для элементарных дизъюнкций, можно записать:

$$Y = x_1\bar{x}_2 \vee \bar{x}_1x_2 = x_1 \cdot \left(\bar{x}_1 \vee \bar{x}_2\right) \vee x_2 \cdot \left(\bar{x}_1 \vee \bar{x}_2\right).$$

Далее берем от функции Y двойное отрицание, одно из которых раскрываем по правилу де Моргана, получим

$$Y = \overline{\overline{x_1 \cdot \left(\bar{x}_1 \vee \bar{x}_2\right) \vee x_2 \cdot \left(\bar{x}_1 \vee \bar{x}_2\right)}} = \overline{\overline{x_1x_1x_2} \cdot \overline{x_2x_1x_2}}.$$

Реализация этой функции на элементах К555ЛА3 показана на

рис. 3.24.

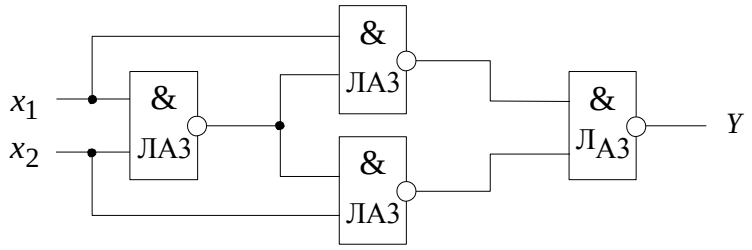


Рис. 3.24. Функциональная схема к примеру 3.10

5. В том случае если ДНФ функции содержит число элементарных конъюнкций ≥8, рекомендуется способ преобразования функции, рассмотренный в примере 3.11.

Пример 3.11. Реализовать функцию

$$Y\left(x_1,x_2,x_3,x_4,x_5,x_6\right)=\overline{x}_1\overline{x}_3\vee\ \overline{x}_2\overline{x}_3\vee\ x_1\overline{x}_2x_6\vee\ \overline{x}_1x_2x_6\vee\ \overline{x}_3\overline{x}_6\vee\ \overline{x}_3\overline{x}_4x_6\vee$$
$$\vee\ x_5x_6\vee x_2x_4\overline{x}_6\vee\ \overline{x}_1\overline{x}_4\overline{x}_5\overline{x}_6\vee\ x_1x_2x_3x_5.$$

Решение

Сгруппируем исходные термы так, чтобы число слагаемых в скобке было равно (или меньше) числу схем И в элементе И-ИЛИ-НЕ, а ранг соответствовал числу входов схем И:

$$Y=\left(\overline{x}_1\overline{x}_3\vee\ \overline{x}_2\overline{x}_3\vee\ x_1\overline{x}_2x_6\vee\overline{x}_1x_2x_6\right)\ \left(\vee\ \overline{x}_3\overline{x}_6\vee\ \overline{x}_3\overline{x}_4x_6\vee\ x_5x_6\vee x_2x_4\overline{x}_6\right)\vee$$
$$\vee\left(\overline{x}_1\overline{x}_4\overline{x}_5\overline{x}_6\vee\ x_1x_2x_3x_5\right).$$

Возьмем от функции *Y* двойное отрицание, одно из которых раскроем, не меняя выражения в скобках, получим:

$$Y=\overline{\overline{\overline{x}_1\overline{x}_3\vee\ \overline{x}_2\overline{x}_3\vee x_1\overline{x}_2x_6\vee\ \overline{x}_1x_2x_6}\cdot\overline{\overline{x}_3\overline{x}_6\vee\ \overline{x}_3\overline{x}_4x_6\vee x_5x_6\vee\ x_2x_4\overline{x}_6}\cdot\overline{\overline{x}_1\overline{x}_4\overline{x}_5\overline{x}_6\vee\ x_1x_2x_3x_5}}.$$

Реализация этой функции показана на рис. 3.25.

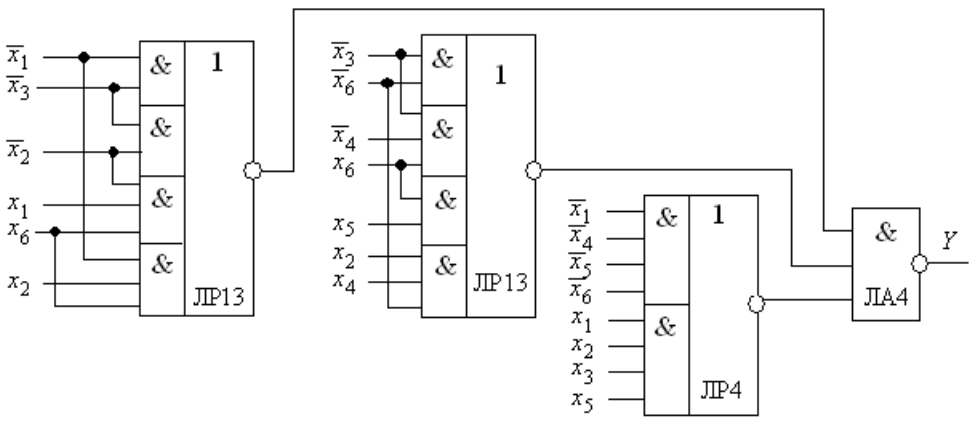


Рис. 3.25. Функциональная схема реализации к примеру 3.11

3.5. Примеры синтеза комбинационных схем на основных элементах серии к555

Пример 3.12. Реализовать функцию $f(x_1, x_2, x_3, x_4) = \vee(0, 2, 4, 6, 7, 10, 11, 14, 15)$ на элементах серии К555:

а) типа И-НЕ; б) типа ИЛИ-НЕ; в) типа И-ИЛИ-НЕ.

Решение

А. Для реализации функции f на элементах И-НЕ с помощью карты Карно (рис. 3.26,а) получим МДНФ функции:

$$f = x_1x_2 \vee x_2x_3 \vee \bar{x}_1\bar{x}_4.$$

Берём двойное отрицание от МДНФ функции и одно из них открываем по правилу де Моргана, получаем

$$f = \overline{\overline{x_1x_2 \vee x_2x_3 \vee \bar{x}_1\bar{x}_4}} = \overline{\overline{x_1x_2}} \cdot \overline{\overline{x_2x_3}} \cdot \overline{\overline{\bar{x}_1\bar{x}_4}}.$$

Реализация функции на элементах типа И-НЕ показана на рис. 3.27,а.

Б. Для реализации функции f на элементах ИЛИ-НЕ получаем МКНФ функции с помощью карты Карно (рис. 3.26,б):

$$f = (\bar{x}_1 \vee x_3) \cdot (x_3 \vee \bar{x}_4) \cdot (x_1 \vee x_2 \vee \bar{x}_4).$$

Берём двойное отрицание от МКНФ и одно из них раскрываем по правилу де Моргана, получаем

$$f = \overline{\overline{(\bar{x}_1 \vee x_3) \cdot (x_3 \vee \bar{x}_4) \cdot (x_1 \vee x_2 \vee \bar{x}_4)}} = \overline{\overline{\bar{x}_1 \vee x_3} \vee \overline{x_3 \vee \bar{x}_4} \vee \overline{x_1 \vee x_2 \vee \bar{x}_4}}.$$

Реализация функции на элементах типа ИЛИ-НЕ показана на рис. 3.27,б.

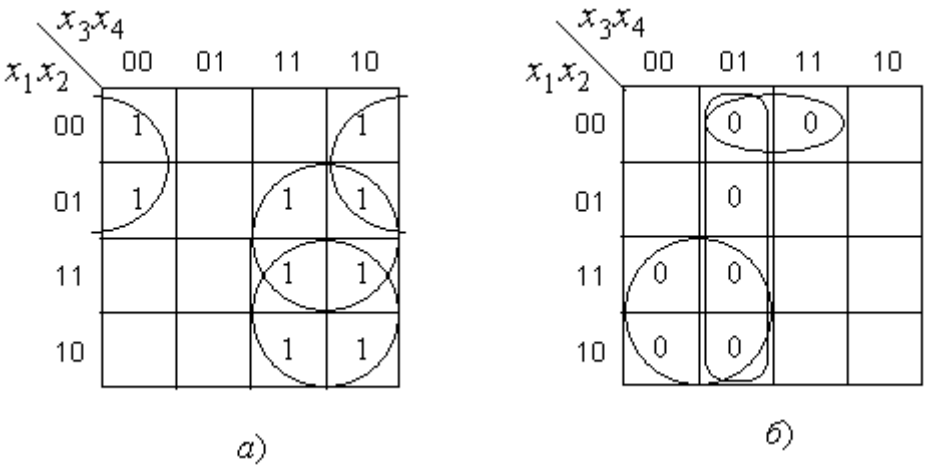
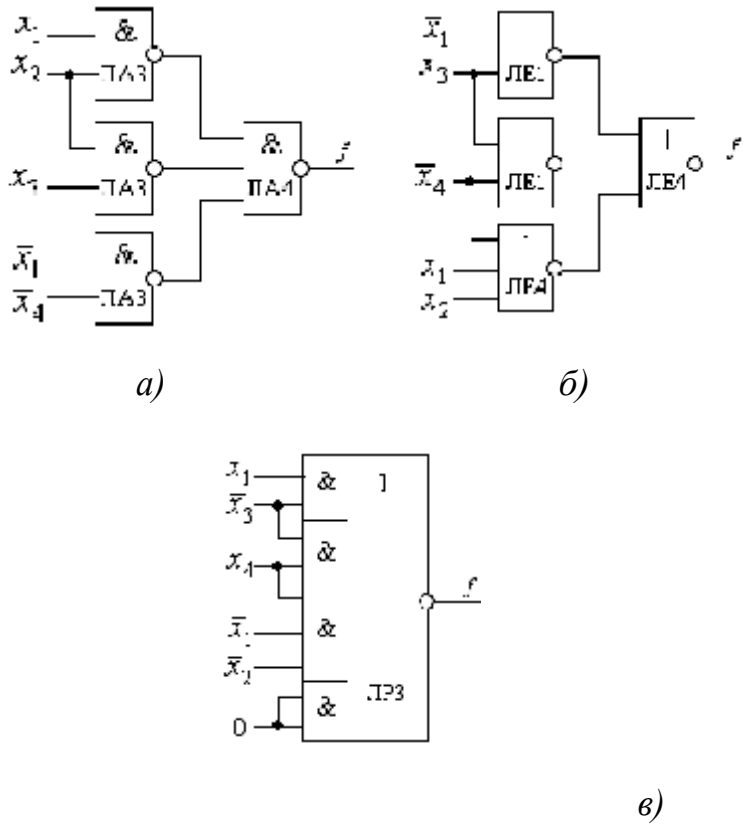


Рис. 3.26. Карта Карно функции $f(x_1, x_2, x_3, x_4)$ к примеру 3.12

В. Для реализации функции *f* на элементах типа И-ИЛИ-НЕ находим с помощью карт Карно (см. рис. 3.26,б) МКНФ, записанную в виде МДНФ отрицания функции:

$$f = \overline{x_1\overline{x}_3 \vee \overline{x}_3 x_4 \vee \overline{x}_1\overline{x}_2 x_4}\,.$$

Реализация полученной функции приведена на рис. 3.27,в.



а) *б)*

в)

Рис. 3.27. Функциональные схемы реализации из примера 3.12: а) на элементах типа И-НЕ; б) на элементах типа ИЛИ-НЕ; в) на элементах типа И-ИЛИ-НЕ

Пример 3.13. Синтезировать КС преобразователя двоично-десятичного кода 3321 в код 5211 на логических элементах серии К555 в базисе: а) И-НЕ, б) ИЛИ-НЕ, выполнив совместную минимизации системы булевых функций.

Решение

Построим таблицу истинности преобразователя кодов (табл. 3.3). Из таблицы следует, что функции y_1, y_2, y_3, y_4 не полностью определенные. С целью минимизации доопределяем значения выходных функций на некоторых избыточных входных наборах, которые показаны на картах Карно звездочкой (рис. 3.28, 3.29).

Таблица 3.3. Таблица истинности преобразователя кодов

Десятичное число	Номер набора	Код 3 3 2 1	Код 5 2 1 1
		x_1 x_2 x_3 x_4	y_1 y_2 y_3 y_4
0	0	0 0 0 0	0 0 0 0
1	1	0 0 0 1	0 0 0 1
2	2	0 0 1 0	0 0 1 1
3	3	0 0 1 1	0 1 0 1
4	5	0 1 0 1	0 1 1 1
5	10	1 0 1 0	1 0 0 0
6	12	1 1 0 0	1 0 1 0
7	13	1 1 0 1	1 1 0 0
8	14	1 1 1 0	1 1 1 0
9	15	1 1 1 1	1 1 1 1

А. Выполним совместную минимизацию системы функций y_1, y_2, y_3, y_4 с помощью карт Карно (рис. 3.28, 3.29).

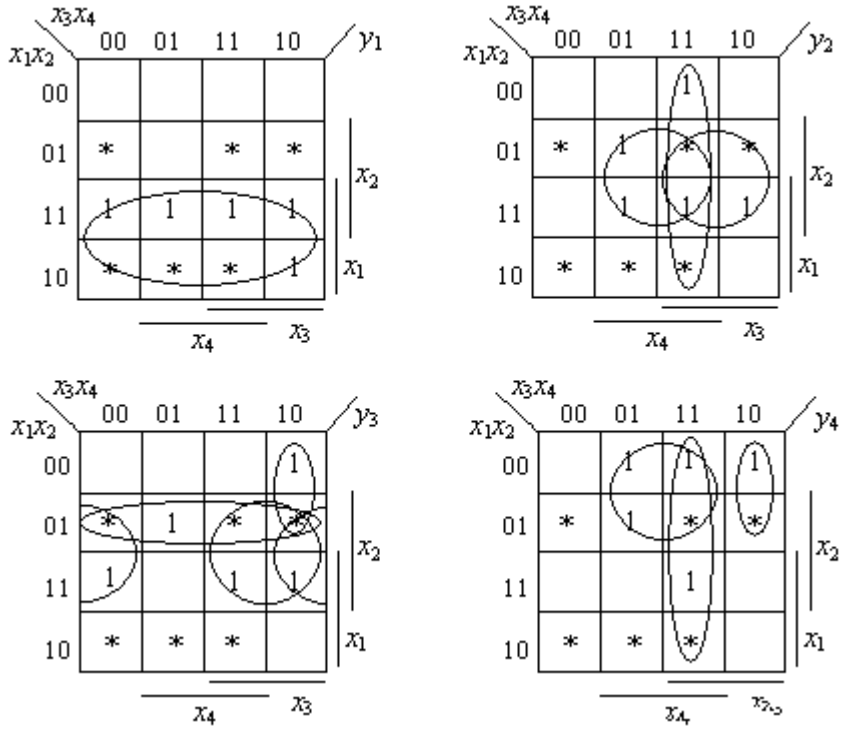


Рис. 3.28. Определение МДНФ функций y_1, y_2, y_3, y_4

МДНФ:

$$y_1 = m_1 = x_1;$$
$$y_2 = m_2 \vee m_3 \vee m_4 = x_3x_4 \vee x_2x_3 \vee x_2x_4;$$
$$y_3 = m_5 \vee m_3 \vee m_6 \vee m_7 = \bar{x}_1x_3\bar{x}_4 \vee x_2x_3 \vee x_2\bar{x}_4 \vee \bar{x}_1x_2;$$
$$y_4 = m_2 \vee m_8 \vee m_5 = x_3x_4 \vee \bar{x}_1x_4 \vee \bar{x}_1x_3\bar{x}_4.$$

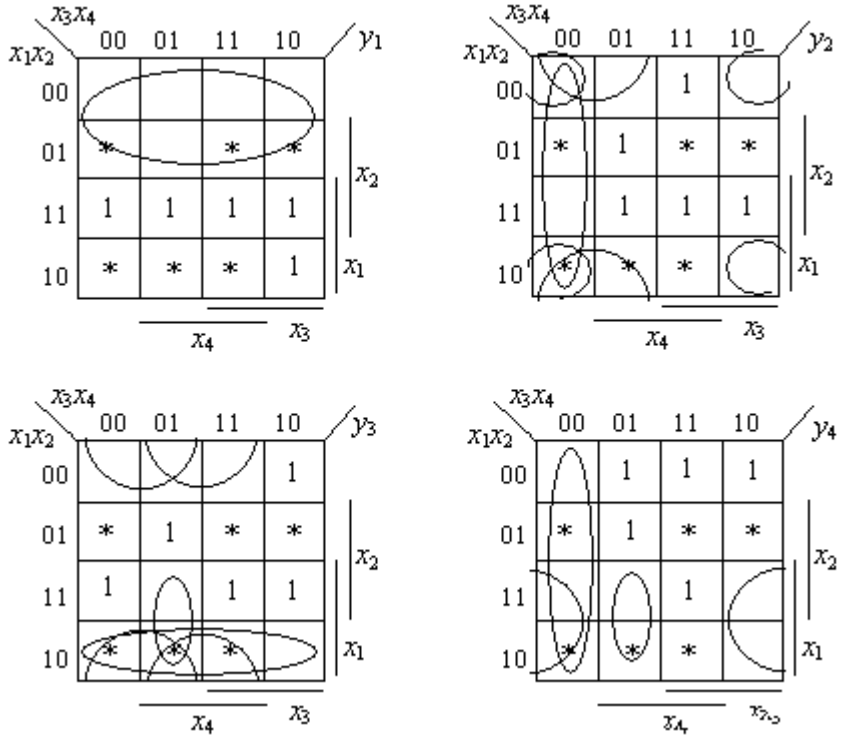


Рис. 3.29. Определение МКНФ функций y_1, y_2, y_3, y_4

МКНФ:

$$y_1 = k_1 = x_1;$$
$$y_2 = k_2 \cdot k_3 \cdot k_4 = (x_3 \vee x_4)(x_2 \vee x_4)(x_2 \vee x_3);$$
$$y_3 = k_5 \cdot k_6 \cdot k_7 \cdot k_8 = (\bar{x}_1 \vee x_3 \vee \bar{x}_4)(\bar{x}_1 \vee x_2)(x_2 \vee x_3)(x_2 \vee \bar{x}_4);$$
$$y_4 = k_2 \cdot k_5 \cdot k_9 = (x_3 \vee x_4)(\bar{x}_1 \vee x_3 \vee \bar{x}_4)(\bar{x}_1 \vee x_4).$$

Выполним преобразование МДНФ функций для их реализации в базисе И-НЕ:

$$y_1 = x_1 = m_1;$$

$$y_2 = x_3x_4 \vee x_2x_3 \vee x_2x_4 = \overline{x_3x_4 \cdot x_2x_3 \cdot x_2x_4} = \overline{m_2 \cdot m_3 \cdot m_4};$$

$$y_3 = x_1x_3x_4 \vee x_2x_3 \vee x_2x_4 \vee x_1x_2 =$$

$$= \overline{x_1x_3x_4 \cdot x_2x_3 \cdot x_2x_4 \cdot x_1x_2} = \overline{m_5 \cdot m_3 \cdot m_6 \cdot m_7};$$

$$y_4 = x_3x_4 \vee x_1x_4 \vee x_1x_3x_4 = \overline{x_3x_4 \cdot x_1x_4 \cdot x_1x_3x_4} = \overline{m_2 \cdot m_8 \cdot m_5}.$$

Схема преобразователя кодов на элементах И-НЕ приведена на рис. 3.30. Количество оборудования (цена схемы) для реализации схемы на элементах базиса И-НЕ составила 5 корпусов ИМС (2 корпуса К555ЛА3, 1 корпус К555ЛА4 и 1 корпус К555ЛА1), а быстродействие составит 3τ.

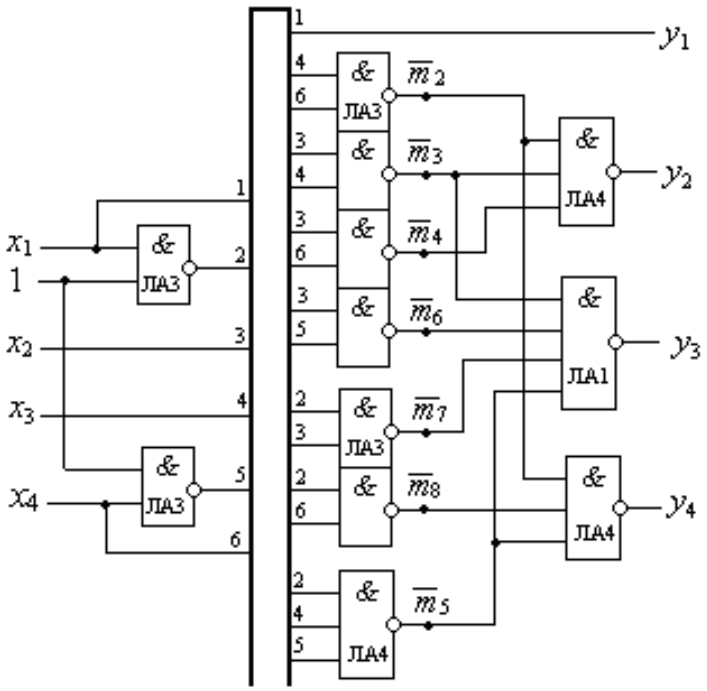


Рис. 3.30. Функциональная схема реализации преобразователя кода 3321 в код 5211 на элементах И-НЕ

Б. Выполним преобразования МКНФ функций для реализации схемы преобразователя в базисе ИЛИ-НЕ:

$$y_1 = x_1 = k_1;$$

$$y_2 = \overline{(x_3 \vee x_4)(x_2 \vee x_4)(x_2 \vee x_3)} =$$

$$= \overline{x_3 \vee x_4 \vee x_2 \vee x_4 \vee x_2 \vee x_3} = \overline{k}_2 \vee \overline{k}_3 \vee \overline{k}_4;$$

$$y_3 = \overline{(\overline{x}_1 \vee x_3 \vee \overline{x}_4)(\overline{x}_1 \vee x_2)(x_2 \vee x_3)(x_2 \vee \overline{x}_4)} =$$

$$= \overline{\overline{x}_1 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_1 \vee x_2 \vee x_2 \vee x_3 \vee x_2 \vee \overline{x}_4} = \overline{k}_5 \vee k_6 \vee \overline{k}_7 \vee k_8;$$

$$y_4 = \overline{(x_3 \vee x_4)(\overline{x}_1 \vee x_3 \vee \overline{x}_4)(\overline{x}_1 \vee x_4)} =$$

$$= \overline{x_3 \vee x_4 \vee \overline{x}_1 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_1 \vee x_4} = \overline{k}_2 \vee \overline{k}_5 \vee \overline{k}_9.$$

Схема преобразователя кодов на элементах базиса ИЛИ-НЕ приведена на рис. 3.31. Количество оборудования (цена схемы) для реализации схемы на элементах базиса ИЛИ-НЕ составила 5 корпусов ИМС (3 корпуса К555ЛЕ1 и 2 корпуса К555ЛЕ4), а быстродействие 5τ.

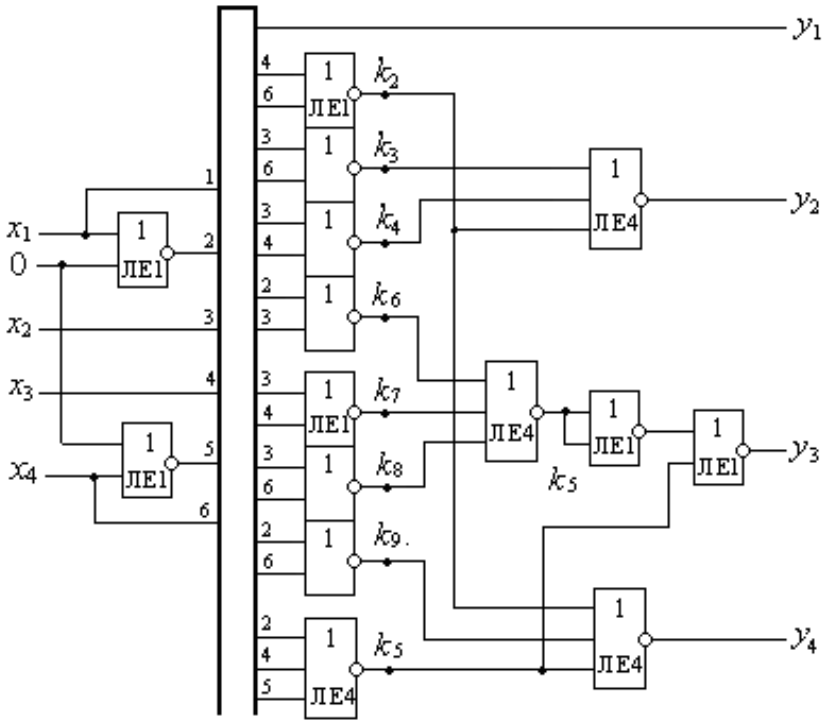


Рис. 3.31. Функциональная схема реализации преобразователя кода 3321 в код 5211 на элементах ИЛИ-НЕ

3.6. Проектирование комбинационных схем на дешифраторах и мультиплексорах

3.6.1. Дешифраторы

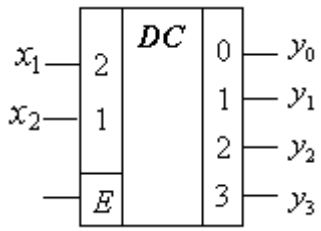
Дешифратором (*DC*) – называется КС, имеющая n входов и 2^n выходов, осуществляющая преобразование входного двоичного n-разрядного кода в сигнал на одном из выходов. По каждому выходу дешифратора реализуется конституента единицы K^1_i (или ее отрицание). Различают полные и неполные дешифраторы. Число выходов полного дешифратора $N_{вых} = 2^n$, неполного – $N_{вых} < 2^n$. Условное графическое обозначение полного дешифратора на два входа и его таблица истинности приведены на рис. 3.32,а,б. E – вход, на который подается сигнал, разрешающий дешифрирование.

Аналитическое описание дешифратора в форме СДНФ:

$Y_i = E \cdot \alpha_i$, $i = 0, 1, ..., n$.

где α_i – i-й минтерм (i-я конституента 1) n входных переменных,

E – сигнал, разрешающий дешифрирование.



E	x_1 x_2	y_0 y_1 y_2 y_3
1	0 0	1 0 0 0
1	0 1	0 1 0 0
1	1 0	0 0 1 0
1	1 1	0 0 0 1
0	* *	0 0 0 0

а) *б)*

Рис. 3.32. Дешифратор: а) условное графическое изображение; б) таблица истинности

Приведем аналитическое описание дешифратора, показанного на рис. 3.32:

$y_0 = E\ \overline{x_1}\,\overline{x_2}$.

$y_1 = E\ x_1\overline{x_2}$.

$y_2 = E\ x_1\overline{x_2}$.

$y_3 = E\ x_1\ x_2$.

В схемах ЭВМ дешифраторы устанавливаются на выходах регистров или счетчиков и служат для преобразования кода слова, находящегося в регистре (в счетчике) в управляющий сигнал на одном из выходов дешифратора.

Рассмотрим примеры использования применения дешифраторов для проектирования КС.

Пример 3.14. Реализовать на основе дешифратора логическую функцию вида: $Y = a\bar{b} \vee \bar{a}\bar{b}c \vee b\bar{c}$.

Решение

1 способ. Исходная функция задана в ДНФ. Преобразуем ее в СДНФ:

$$Y = a\bar{b}\left(c \vee \bar{c}\right) \vee \bar{a}\bar{b}c \vee b\bar{c}\left(a \vee \bar{a}\right) = a\bar{b}c \vee a\bar{b}\bar{c} \vee \bar{a}\bar{b}c \vee ab\bar{c} \vee \bar{a}b\bar{c} =$$

$$= \bar{a}\bar{b}c \vee \bar{a}b\bar{c} \vee a\bar{b}\bar{c} \vee a\bar{b}c \vee ab\bar{c} =$$

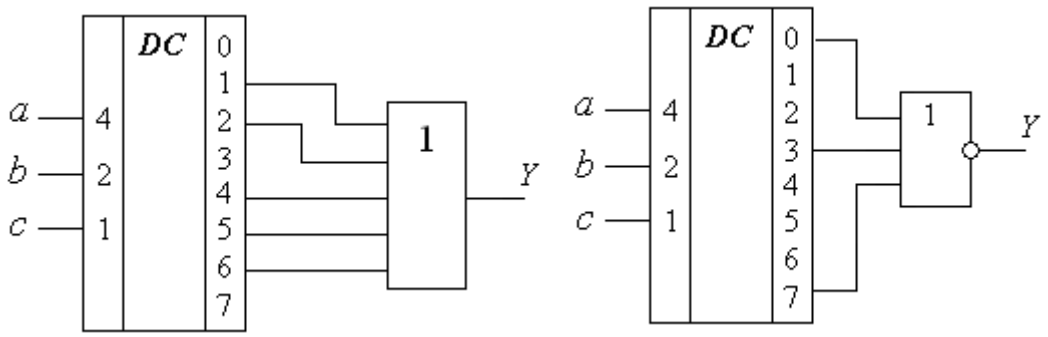
$$= \vee (1, 2, 4, 5, 6) = y_1 \vee y_2 \vee y_4 \vee y_5 \vee y_6.$$

Для получения схемы достаточно определить выходы дешифратора, соответствующие входящим в функцию конституентам единицы и соединить их с входами дизъюнктора. Если на входы дешифратора будут поданы входные переменные, то на выходе дизъюнктора сформируется значение функции (рис. 3.33,а).

2 способ. Найдем СДНФ отрицания функции *Y***:**

$$Y = \overline{\bar{a}\bar{b}\bar{c} \vee \bar{a}bc \vee abc} = \overline{0 \vee 3 \vee 7} = \overline{y_0 \vee y_3 \vee y_7}.$$

Для получения схемы достаточно выходы дешифратора, соответствующие входящим в функцию конституентам единицы, соединить со входами элемента ИЛИ-НЕ (рис. 3.33,б).



а) *б)*

Рис. 3.33. Функциональная схема реализации логической функции $Y = a\bar{b} \vee \bar{a}\bar{b}c \vee b\bar{c}$ *на основе дешифратора*

Пример 3.15. Реализовать на основе дешифратора преобразователь двоично-десятичного кода 3321 в код 5211, рассмотренный в примере 3.13.

Решение

Таблицу истинности преобразователя кода (табл. 3.3) дополним выходными шинами дешифратора y_0, …, y_{15} (табл. 3.4), *A*, *B*, *C*, *D* – выходные сигналы преобразователя.

Из таблицы истинности записываются уравнения выходных функций для реализации преобразователя. Схема преобразователя кодов на основе дешифратора приведена на рис. 3.34.

Таблица 3.4. Таблица истинности преобразователя кодов

Десятичное число	**Код 3 4 2 1**	***DC***	**Код 5 2 1 1**
	$x_1\ x_2\ x_3\ x_4$		***A B C D***
0	0 0 0 0	y_0	0 0 0 0
1	0 0 0 1	y_1	0 0 0 1
2	0 0 1 0	y_2	0 0 1 1
3	0 0 1 1	y_3	0 1 0 1
4	0 1 0 1	y_5	0 1 1 1
5	1 0 0 1	y_9	1 0 0 0
6	1 1 0 0	y_{12}	1 0 1 0
7	1 1 0 1	y_{13}	1 1 0 0
8	1 1 1 0	y_{14}	1 1 1 0
9	1 1 1 1	y_{15}	1 1 1 1

1 способ

$$A = y_9 \vee y_{12} \vee y_{13} \vee y_{14} \vee y_{15},$$
$$B = y_3 \vee y_5 \vee y_{13} \vee y_{14} \vee y_{15},$$
$$C = y_2 \vee y_5 \vee y_{12} \vee y_{14} \vee y_{15},$$
$$D = y_1 \vee y_2 \vee y_3 \vee y_5 \vee y_{15}.$$

2 способ

$$A = \overline{y_0 \vee y_1 \vee y_2 \vee y_3 \vee y_5},$$
$$B = \overline{y_0 \vee y_1 \vee y_2 \vee y_9 \vee y_{12}},$$
$$C = \overline{y_0 \vee y_1 \vee y_3 \vee y_9 \vee y_{13}},$$
$$D = \overline{y_0 \vee y_9 \vee y_{12} \vee y_{13} \vee y_{14}}.$$

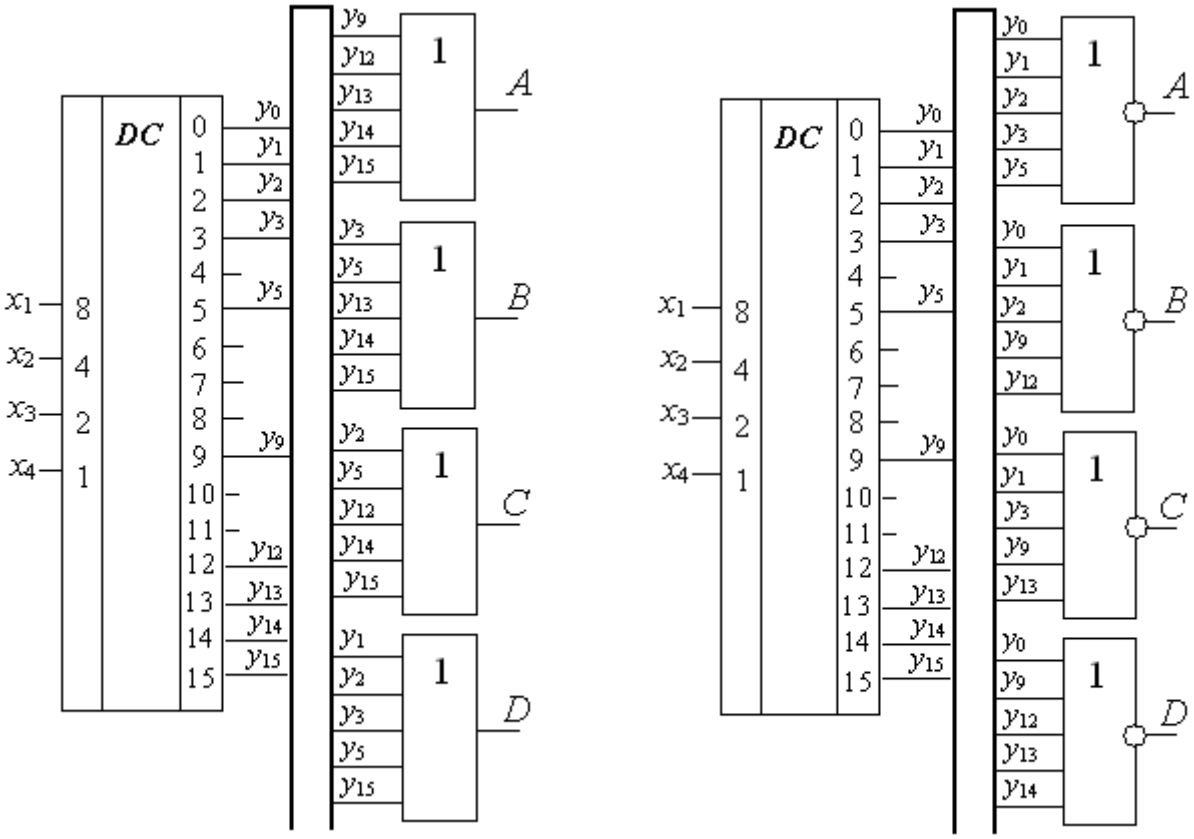


Рис. 3.34. Функциональная схема реализации преобразователя кода 3321 в код 5211 на основе дешифратора

3.6.2. Мультиплексоры

Мультиплексор (*MS*) – это адресный коммутатор, который может выполнить коммутацию на выход сигнала с того информационного входа, адрес которого задан сигналами на адресных входах.

Входы мультиплексора разделяются на информационные (входы данных) и адресные (управляющие). Код (адрес), поступающий на управляющие входы, определяет информационный вход, сигнал с которого передается на выход. Чаще всего используют мультиплексоры на 4, 8, 16 входов. Условное графическое изображение мультиплексора на четыре входа (“4-1”) и его таблица истинности приведены на рис. 3.35,а,б, где E – сигнал, разрешающий мультиплексирование; x_0, x_1, x_2, x_3 – информационные входы; A_2, A_1 – управляющие входы; Y – выход.

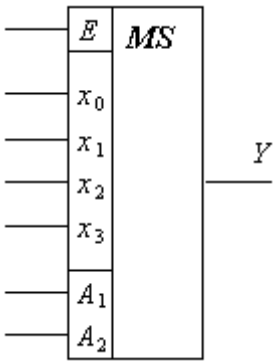
Из таблицы истинности (рис. 3.35,б) получим СДНФ функции Y:

$$Y = x_0 \overline{A}_2 \overline{A}_1 E \vee x_1 \overline{A}_2 A_1 E \vee x_2 A_2 \overline{A}_1 E \vee x_3 A_2 A_1 E .$$

СДНФ выходной функции мультиплексора на 2^n входов имеет вид:

$$Y = \mathrm{V}_i E\, x_i \alpha_i , \qquad i = 0, 1, \ldots , 2^n - 1,$$

где α_i – конституента 1, соответствующая i-му адресному набору.



E	A_2 A_1	Y
1	0 0	x_0
1	0 1	x_1
1	1 0	x_2
1	1 1	x_3
0	* *	0

а) *б)*

Рис. 3.35. Мультиплексор "4-1": а) условное графическое обозначение; б) таблица истинности

Мультиплексор можно рассматривать как преобразователь параллельной информации в последовательную. Мультиплексор на большое число входов, как правило, приходится строить из мультиплексоров меньшей размерности.

Рассмотрим примеры использования мультиплексоров для проектирования КС.

Пример 3.16. Реализовать на мультиплексоре типа "8-1" логическую функцию трех переменных:

$$Y = \vee\,(1,2,6,7) = \overline{a}\overline{b}c \vee \overline{a}b\overline{c} \vee a\overline{b}\overline{c} \vee abc.$$

Решение

Функция задана в СДНФ. Используется мультиплексор типа "8-1". На адресные входы подаются входные переменные, а информационные входы, соответствующие входящим в функцию конституентам единицы, соединяются с шинами питания (1), остальные информационные входы соединяются с шинами земли (0). На выходе мультиплексора формируется значение функции (рис. 3.36).

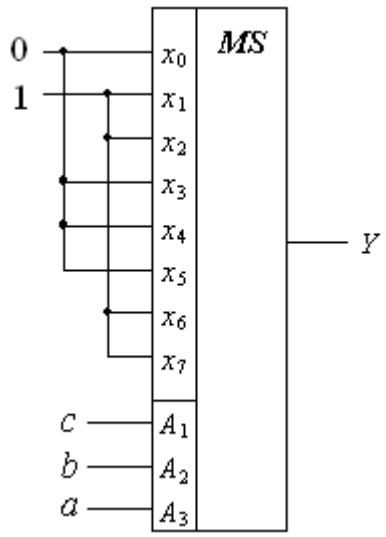


Рис. 3.36. Схема реализации функции $Y = \varpi(1, 2, 6, 7)$ *на мультиплексоре типа "8-1"*

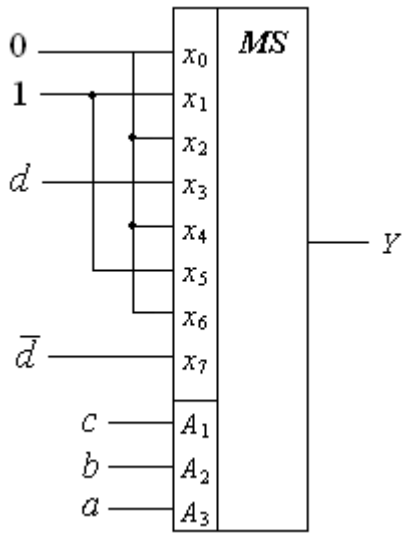


Рис. 3.37. Схема реализации функции $Y(a,b,c,d) = \overline{abc} \vee \bar{a}bcd \vee a\bar{b}c \vee abc\bar{d}$ *на мультиплексоре типа "8-1"*

Пример 3.17. Реализовать на мультиплексоре типа "8-1" логическую функцию четырех переменных:

$$Y(a,b,c,d) = \overline{abc} \vee \bar{a}bcd \vee a\bar{b}c \vee abc\bar{d}\,.$$

Решение

В качестве управляющих сигналов используются переменные *a*, *b* и *c*, которые подаются на адресные входы мультиплексора. На информационные входы поступают переменные *d*, $\bar{d}$, 0, 1 (рис. 3.37).

Пример 3.18. Реализовать с помощью мультиплексора типа "4-1" функцию трех переменных $Y(a, b, c) = \vee(0, 2, 3, 8)$.

Решение

Сначала выбирается любое сочетание двух переменных *ab*, *ac* или *bc*, которые являются управляющими и подаются на адресные входы мультиплексора. На информационные входы в этом случае могут быть поданы четыре функции одной (третьей) переменной.

На рис. 3.38,а,б,в показано соответствие информационных входов мультиплексора x_0, x_1, x_2, x_3 определенным адресным входам мультиплексора "4-1". Например, если в качестве управляющих применить сигналы *a* и *b*, то входу x_0 будут соответствовать две клетки карты Карно, для которых $a = b = 0$; входу x_1 – две клетки с координатами $a = 0$, $b = 1$; входу x_2 – клетки с $a = 1$, $b = 0$; входу x_3 – клетки с $a = b = 1$ (рис. 3.35,а).

Таким образом, карта Карно на три переменные разбивается как

бы на четыре двухклеточные карты на одну переменную. Затем минимизируется набор из четырех функций одной переменной и получаются необходимые значения сигналов на информационных входах мультиплексора для заданной функции.

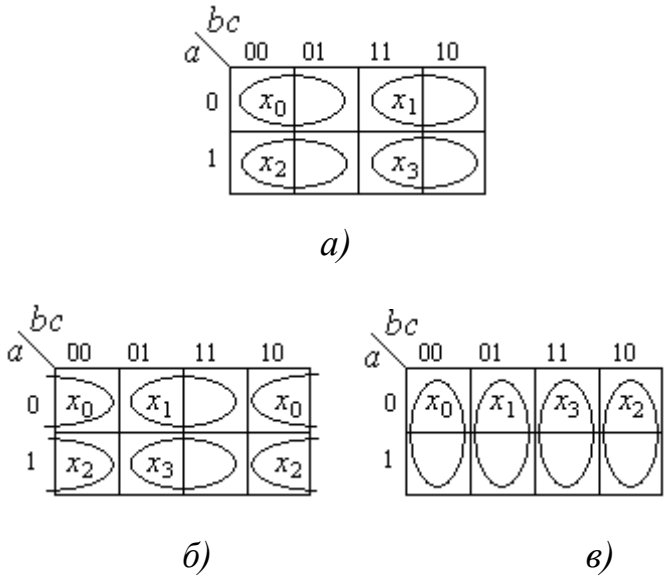


Рис. 3.38. Соответствие информационных входов мультиплексора "4-1" управляющим сигналам: а) – a и b, б) – a и c, в) – b и c

Функция $Y = \vee (0, 2, 3, 6)$ представлена на карте Карно (рис. 3.39,а).

Выберем в качестве управляющих сигналов переменные *a* и *b*. Минимизируем набор из четырех функций от одной переменной *c*, соответствующих каждому информационному входу (x_0, x_1, x_2, x_3). Обе клетки, соответствующие x_1, помечены единицей и, следовательно, на вход x_1 подается сигнал $c \vee \overline{c} = 1$. Оставшиеся входные функции получаем с помощью карты Карно (рис. 3.35,а): $x_0 = \overline{c}$; $x_2 = 0$; $x_3 = \overline{c}$. Подавая на входы мультиплексора найденные значения x_i, реализуем заданную функцию (рис. 3.39,б).

Если в качестве управляющих выбрать сигналы *a* и *c* или *b* и *c*, то получаются новые реализации (рис. 3.39,в,г, соответственно).

Пример 3.19. Реализовать на мультиплексоре типа "4-1" логическую функцию четырех переменных:

$Y(a, b, c, d) = \vee (0, 1, 6, 7, 9, 10, 11, 12, 13)$.

В качестве управляющих сигналов принять переменные:

1) *a* и *b*, 2) *c* и *d*.

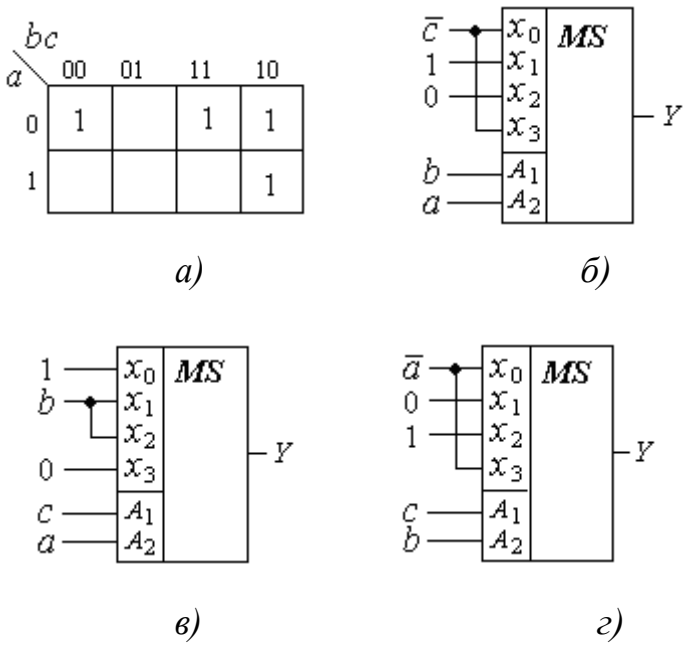


Рис. 3.39. Пример применения мультиплексора для реализации функции $Y = \varpi(0,2,3,6)$

Решение

1. Пусть адрес $a\ b_{(2)}$. Покажем на карте Карно для функции четырех переменных $Y(a, b, c, d)$ расположение информационных сигналов (x_0, x_1, x_2, x_3) мультиплексора "4-1". Для каждого из четырех x_i получим свою четырехклеточную карту Карно для двух переменных c и d. Например, x_0 соответствуют четыре клетки с координатами $a = b = 0$; $x_1 > a = 0, b = 1$; $x_2 > a = 1, b = 0$; $x_3 > a = b = 1$ (рис. 3.40,а).

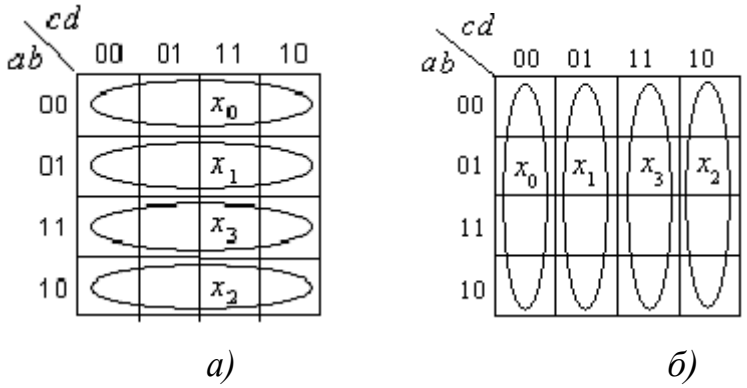
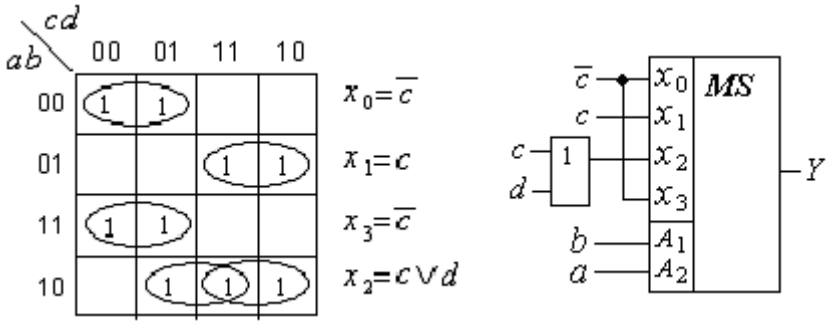


Рис. 3.40. Соответствие информационных входов мультиплексора "4-1" управляющим сигналам: а) – a и b, б) – c и d

Отобразим на карте Карно заданную функцию Y (рис. 3.41,а) и проведем для каждого x_i минимизацию. Получим $x_0 = \overline{c}$, $x_1 = c$, $x_2 = c \vee d$, $x_3 = \overline{c}$. Это значит, что для реализации заданной функции на информационные входы мультиплексора "4-1" необходимо подать соответствующие значения переменных c и d (рис. 3.41,б).

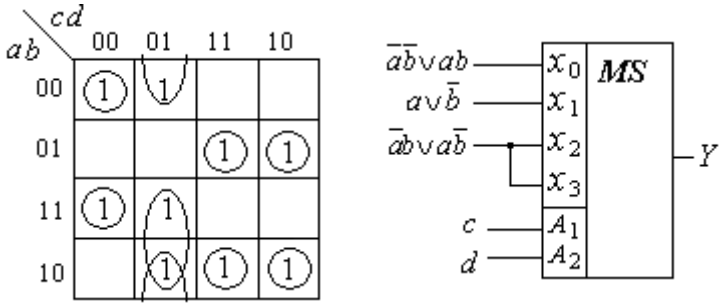


а) *б)*

Рис. 3.41. Реализация функции $Y(a,b,c,d) = \varpi(0,1,6,7,9,10,11,12,13)$ *на мультиплексоре типа "4-1". Адрес* $ab_{(2)}$

2. Пусть адрес с $d_{(2)}$. На карте Карно (рис. 3.40,б) для функции четырех переменных показано расположение информационных сигналов x_0, x_1, x_2, x_3 мультиплексора "4-1".

Для заданной функции Y (рис. 3.42,а) проведем для каждого x_i минимизацию. Получим $x_0 = \overline{ab} \vee ab$, $x_1 = a \vee \bar{b}$, $x_2 = \bar{a}b \vee a\bar{b}$, $x_3 = \bar{a}b \vee a\bar{b}$. Это значит, что для реализации заданной функции на информационные входы мультиплексора "4-1" необходимо подать соответствующие значения переменных a и b (рис. 3.42,б).



б) *в)*

Рис. 3.42. Реализация функции $Y(a, b, c, d) = \varpi(0, 1, 6, 7, 9, 10, 11, 12, 13)$ *на мультиплексоре типа "4-1". Адрес с* $d_{(2)}$

Контрольные вопросы

1. Сформулируйте закон функционирования автомата без памяти.
2. Что понимают под логическим элементом?
3. Какими техническими параметрами характеризуются ЛЭ?
4. Что такое базис?
5. Назовите логические элементы булевого базиса.

6. Какая КС называется одновыходной? Многовыходной?

7. Сформулируйте задачи анализа и синтеза цифровых автоматов.

8. Назовите основные этапы синтеза КС.

9. Назовите основные критерии качества технической реализации КС.

10. Как определить сложность (цену) схемы в единицах по Квайну?

11. Как определить быстродействие схемы?

12. Как преобразовать булеву функцию для реализации на элементах базиса: а) И-НЕ, б) ИЛИ-НЕ, в) И-ИЛИ-НЕ?

13. Какие способы реализации многовыходных КС вы знаете?

14. Какие цели преследует совместная минимизация системы булевых функций?

15. Какие логические элементы входят в состав серии К555?

16. В чем заключается метод тождественных преобразований с предварительной группировкой?

17. В чем заключается факторизационный метод синтеза КС?

18. Что называется дешифратором?

19. Какие способы реализации функций на основе дешифратора вы знаете?

20. Что называется мультиплексором?

21. Назначение и области применения мультиплексоров.

Задания для самостоятельной работы

1. Синтезировать схему контроля двоично-десятичной тетрады кода *D*1 (8421):

1) на элементах серии К555 в базисах:

а) И, ИЛИ, НЕ; б) И- НЕ; в) ИЛИ-НЕ; г) И-ИЛИ-НЕ.

2) на основе дешифратора двумя способами;

3) на основе мультиплексора: а) типа 8-1; б) типа 4-1.

2. Синтезировать КС преобразователя двоично-десятичного кода 2421 в код 8411:

1) на элементах серии К555 в базисе: а) И-НЕ, б) ИЛИ-НЕ, выполнив совместную минимизацию системы булевых функций;

2) на основе дешифратора двумя способами.

3. Выполнить совместную минимизацию системы булевых функций четырех переменных и реализовать на элементах серии К555:

а) типа И-НЕ; б) типа ИЛИ-НЕ; в) типа И-ИЛИ-НЕ.

Для каждого варианта определить основные критерии качества

технической реализации.

y_1 = & (5, 6, 7, 8, 9), * = (10-15);
y_2 = & (4, 6, 7, 8, 9) , * = (10-15);
y_3 = & (2, 3, 5, 8, 9) , * = (10-15) ;
y_4 = & (1, 3, 5, 7, 9), * = (10-15).

4. Реализовать функцию $Y(a, b, c)$ на элементах 2И-НЕ, используя факторизационный метод синтеза:

$$Y = \bar{a}\bar{b}\bar{c} \vee \bar{a}bc \vee a\bar{b}c \vee ab\bar{c}.$$

5. Реализовать функцию $Y = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2x_3$, не используя инверсии входных переменных на двух корпусах микросхемы К555ЛА3.

6. Функцию $Y(x_1, x_2, x_3, x_4) = x_1\bar{x}_2x_4 \vee x_1\bar{x}_3\bar{x}_5 \vee x_2\bar{x}_4x_5 \vee \bar{x}_3\bar{x}_4x_5$ преобразовать в МКНФ и реализовать на элементах типа И-ИЛИ-НЕ серии К555, используя метод тождественных преобразований с предварительной группировкой.

7. Реализовать функцию из примера 6 на основе мультиплексора: а) типа “16-1”; б) типа “8-1”; в) типа “4-1”.

Глава 4. АБСТРАКТНЫЙ СИНТЕЗ ЦА

Как правило, алгоритм работы синтезируемого цифрового автомата описывается на одном из начальных языков. Абстрактный синтез выполняется в два этапа. На первом этапе выполняется переход к описанию алгоритма на одном из стандартных языков. На втором этапе решается задача минимизации числа состояний автомата.

Для описания функционирования цифровых автоматов можно выделить два класса языков: *начальные языки* и *стандартные,* или *автоматные языки.*

Начальные языки задают функцию переходов и функцию выходов в неявном виде. Поведение автомата описывается в терминах входных и выходных последовательностей, реализуемых оператором, или последовательностей управляющих сигналов, воздействующих на управляющий автомат [8].

Для описания функционирования ЦА на начальном языке можно использовать:

- язык регулярных выражений алгебры событий;
- язык исчисления предикатов;
- язык логических схем алгоритмов (ЛСА);
- язык граф-схем алгоритмов (ГСА).

4.1. Задание цифровых автоматов на начальных языках

4.1.1. Граф-схемы алгоритмов

Рассмотрим начальные языки, наиболее часто используемые для описания функционирования ЦА: язык граф-схем алгоритмов (ГСА) и язык логических схем алгоритмов (ЛСА) [6].

Язык ГСА является простейшим и самым распространённым языком, применяемым в практике проектирования устройств ЦВМ и, в частности, микропрограммных устройств. Это объясняется их наглядностью, простотой конструкции, а также простотой преобразований и формального перехода к таблицам перехода и аналитическому представлению алгоритмов в виде систем канонических уравнений (СКУ).

Язык ГСА – графический язык, поэтому символы, применяемые в нём, имеют определённое геометрическое начертание, определяемое ГОСТом.

ГСА – ориентированный связный граф, содержащий одну начальную (A_0), одну конечную (A_k) и произвольное конечное множество условных $X = \{x_1, ..., x_L\}$ и операторных $A = \{A_1, ..., A_M\}$ вершин (рис. 4.1).

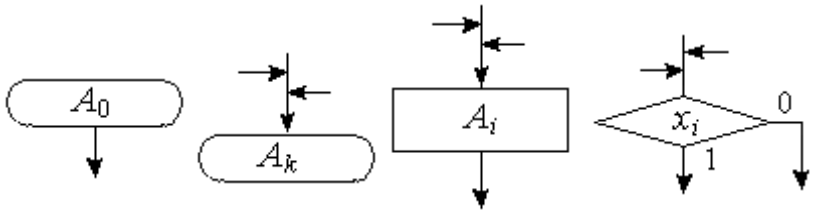


Рис. 4.1. Вершины ГСА

Выполнение любого алгоритма всегда начинается с начальной вершины (оператор A_0) и заканчивается конечной вершиной (оператор A_k). Начальная вершина A_0 рассматривается как оператор, символизирующий начало работы алгоритма. Она не имеет входов, т.е. в нее не входит ни одной стрелки. Конечная вершина A_k рассматривается как оператор, символизирующий конец работы алгоритма. Она не имеет выходов, т.е. из этого оператора не выходит ни одной стрелки.

Произвольные вершины A_i рассматриваются как операторы, обозначающие определенные действия $Y = \{y_1, y_2, ..., y_N\}$, связанные с реализацией алгоритма. Разрешается в различных операторных вершинах запись одинаковых элементов множества Y.

Внутри условных вершин записывается логическое условие. Под логическим условием понимается логическая функция или логическая переменная, принимающая значение 1 или 0. Разрешается в различных условных вершинах запись одинаковых элементов множества $X = \{x_1, x_2, ..., x_L\}$. Условная вершина два имеет два выхода, помеченные значениями анализируемых на данном шаге алгоритма логических функций или переменных, т.е. символами 1 и 0.

Конечная, операторная и условная вершины имеют по одному входу, причём входящая линия может быть образована слиянием нескольких линий.

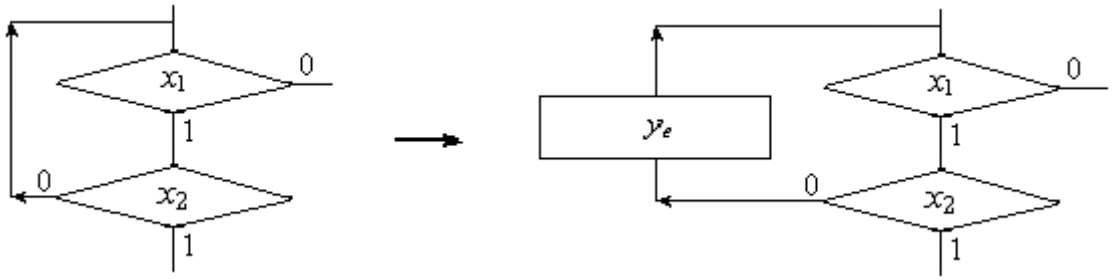
ГСА удовлетворяет следующим условиям:

1. Входы и выходы вершин соединяются друг с другом с помощью дуг, направленных от выхода к входу.
2. Каждый выход соединён только с одним входом.
3. Любой вход соединяется, по крайней мере, с одним выходом.
4. Для любой вершины графа существует, по крайней мере, один путь из этой вершины к конечной вершине.
5. Один из выходов условной вершины может соединяться с её входом, что недопустимо для операторной вершины.

При соединении выхода условной вершины с её входом в цепь

обратной связи вводится пустая операторная вершина, отмеченная пустым выходным сигналом y_e. (рис. 4.2,а).

Пустые операторные вершины ставят на выходе условной вершины в начале цикла. Возможно неверное изображение (рис. 4.2,б).



а) *б)*

Рис. 4.2. Пример добавления пустой операторной вершины

При разработке алгоритма работы ОУ вначале записывается содержательная ГСА, в которой внутри операторных и условных вершин записывается содержательное обозначение микроопераций и логических условий (рис. 4.3).

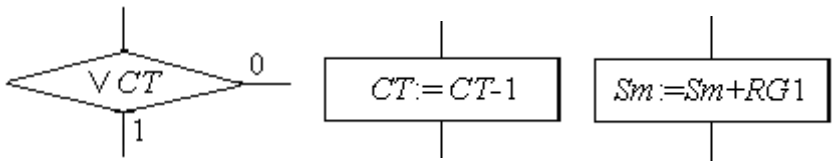


Рис. 4.3. Пример обозначения микроопераций и логических условий

При кодировании содержательной ГСА внутри операторных вершин записываются символы из множества выходных сигналов $Y = \{y_1, y_2, ..., y_N\}$, а внутри условных вершин символы из множества входных сигналов $X = \{x_1, x_2, ..., x_L\}$ (рис. 4.4).

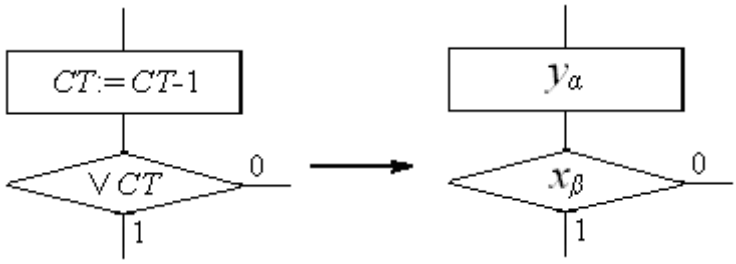


Рис. 4.4. Пример кодирования содержательной ГСА

Пример ГСА приведен на рис. 4.3. Выполнение алгоритма начинается с оператора A_0. На следующих двух шагах выполняются операторы A_1 и A_2. На четвёртом шаге может выполняться один из трёх операторов

A_3, A_4 и A_k в зависимости от значений логических условий x_1, x_2 и x_3. Если $x_1 = 0$, а $x_2 = 1$, то выполняется оператор A_3; если $x_1 = 1$, то выполняется A_4 при $x_3 = 1$ или A_k при $x_3 = 0$. Если же $x_1 = 0$ и $x_2 = 0$, то, независимо от значения сигнала x_3, возникает цикл, выход из которого возможен только при изменении значений логических условий x_1 и x_2. Чтобы этого избежать, необходимо ввести пустую операторную вершину, отмеченную пустым выходным сигналом y_e, как показано на рис. 4.2. В этом случае автомат будет находиться в режиме ожидания, выдавать пустой сигнал y_e до тех пор, пока x_1 или x_2 не станут равными единице (см. рис. 4.6).

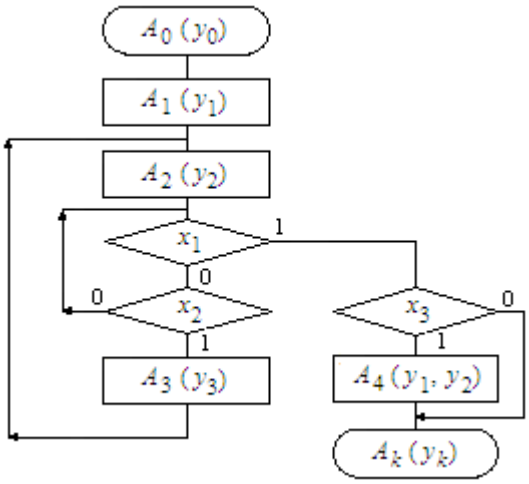


Рис. 4.5. Пример ГСА

Предположим, что после выполнения A_2 имеет место $x_1 = 0$ и $x_2 = 1$. Тогда, независимо от значения x_3, будут последовательно выполняться A_2 и A_3 до тех пор, пока после очередного выполнения оператора A_2 и проверки логических условий логическое условие x_1 не станет равным единице. После этого выполняется оператор A_4 или A_k, как было отмечено выше. Выполнение алгоритма закончится оператором A_k.

Шаг алгоритма – это путь на ГСА от одной операторной вершины к другой, состоящий только из условных вершин.

4.1.2. Логические схемы алгоритмов

Язык ЛСА является аналитической интерпретацией языка ГСА и может быть использован для более компактной формы записи алгоритма функционирования ЦА.

Язык ЛСА был впервые предложен А.А. Ляпуновым в 1953 г. для записи микропрограмм. В дальнейшем он стал широко использоваться в качестве начального языка для задания алгоритмов функционирования управляющих устройств.

Запись алгоритма на языке ЛСА представляет собой конечную строку, состоящую из символов операторов $A = \{A_0, A_1, ..., A_k\}$, логических условий $X = \{x_1, ..., x_L\}$ и верхних и нижних стрелок, которым приписаны натуральные числа ($\overset{i}{\uparrow}\ \overset{i}{\downarrow}$, $i = 1, 2, 3\ldots$). При этом последовательность выполнения операторов будет определяться порядком их записи. Порядок выполнения операторов может быть строго фиксированным или зависеть от некоторых логических условий. В последнем случае применяют логическую переменную $x_m \overset{i}{\uparrow}$ со стоящим справа от нее началом стрелки $\overset{i}{\uparrow}$ с индексом $i = 1, 2\ldots$ Конец стрелки $\overset{i}{\downarrow}$ с этим же индексом стоит слева от того оператора ЛСА, который должен выполняться, если логическая переменная x_m принимает нулевое значение.

Например, $A_1\, x_m \overset{i}{\uparrow}\, A_2 \ldots \overset{i}{\downarrow} A_n$ означает, что после выполнения оператора A_1 в зависимости от значения логического условия x_m может быть выполнен оператор A_2, стоящий непосредственно за $x_m \overset{i}{\uparrow}$, если $x_m = 1$, или оператор A_n справа от стрелки $\overset{i}{\downarrow}$, если $x_m = 0$.

В некоторых случаях используются логические условия, которые всегда принимают нулевое значение, т.е. тождественно ложные логические условия ω (оператор ω). После оператора ω всегда производится переход по стрелке, которая стоит справа от него.

Если в ЛСА имеются циклы из логических условий, то вводится пустой оператор Ae (ye), отмеченный пустым выходным сигналом. Этот оператор помещают в ЛСА путём замены стрелки $\overset{i}{\downarrow}$, стоящей в начале петли из логический условий на следующую группу символов ЛСА:

$$\omega \overset{k}{\uparrow} \overset{i}{\downarrow} A_e(y_e) \overset{k}{\downarrow}. \qquad (4.1)$$

Например, для фрагмента ЛСА: $A_p \overset{2}{\downarrow} x_1 \overset{1}{\uparrow} x_2 \overset{2}{\uparrow} A_n \ldots \overset{1}{\downarrow} A_m$ получим

$$A_p \omega \overset{k}{\uparrow} \overset{2}{\downarrow} A_e(y_e) \overset{k}{\downarrow} x_1 \overset{1}{\uparrow} x_2 \overset{2}{\uparrow} A \ \ldots \overset{1}{\downarrow} A_n.$$

В качестве примера приведем запись ЛСА, эквивалентной ГСА, рассмотренной выше (см. рис. 4.3):

$$A_0(y_0)A_1(y_1)\downarrow^{3} A_2(y_2)\underbrace{\downarrow^{2}\bar{x}_1\uparrow^{1} x_2\uparrow^{2}}_{\text{цикл из ЛУ}} A_3(y_3)\omega\uparrow^{3}\downarrow^{1} x_3\uparrow^{4} A_4(y_1y_2)\downarrow^{4} A_k(y_k).$$

(4.2)

В ЛСА имеется цикл из логических условий:

$$\cdots\downarrow^{2} x_1\uparrow^{1} x_2\uparrow^{2}\cdots$$

С учётом правила (4.1) скорректируем ЛСА, получим:

$$A_0(y_0)A_1(y_1)\downarrow^{3} A_2(y_2)\omega\uparrow^{k}\downarrow^{2} A_e(y_e)\downarrow^{k}\bar{x}_1\uparrow^{1} x_2\uparrow^{2} A_3(y_3)\omega\uparrow^{3}\downarrow^{1} x_3\uparrow^{4} A_4(y_1y_2)\downarrow^{4} A_k(y_k).$$

4.1.3. Преобразование ЛСА в ГСА

При преобразовании ЛСА в ГСА каждому оператору в ЛСА ставится в соответствие операторная вершина и каждому логическому условию (кроме тождественно ложного оператора ω) условная вершина ГСА.

На первом этапе осуществляется построение отдельных фрагментов ГСА, число которых зависит от количества символов ω в ГСА и равно ω + 1. Начальный фрагмент ГСА содержит операторные и условные вершины, соответствующие всем операторам и логическим условиям, которые записаны в ЛСА перед первым символом. При этом все вершины соединены последовательно в том порядке, в каком записаны в ЛСА соответствующие им символы.

Выход любой условной вершины x_i, соединенный со следующей вершиной, отмечается символом “1”, второй выход этой вершины отмечается символом “0” и помечается номером, соответствующим номеру верхней стрелки, которая расположена в ЛСА непосредственно за логическим условием x_i. Если между парой символов ЛСА имеется нижняя стрелка с номером *i*, то на фрагменте ГСА к линии, соединяющей две вершины, которые соответствуют этой паре символов, подводится входящая стрелка с этим же номером. Из последней вершины фрагмента выводится стрелка с номером, соответствующим номеру верхней стрелки, которая стоит непосредственно за символом ω.

Каждый последующий фрагмент ГСА содержит операторные и условные вершины, соответствующие всем операторам и логическим условиям в ЛСА, которые записаны между двумя последовательными символами ω или за последним символом ω для конечного фрагмента. Данные фрагменты строятся в основном по тем же правилам, что и начальный фрагмент.

Дополнительные требования заключаются в следующем: к начальной вершине каждого фрагмента подводится входящая стрелка с номером, соответствующим номеру нижней стрелки, стоящей в ЛСА непосредственно перед символом этой вершины; из последней вершины конечного фрагмента стрелка не выводится.

На втором этапе построения ГСА все одноименные входящие и исходящие стрелки фрагментов соединяются, в результате чего получается искомая ГСА.

Пример 4.1. Преобразовать в ГСА заданную ЛСА:

$$A_0 x_0 \overset{1}{\uparrow} A_1 \overset{1}{\downarrow} x_1 \overset{2}{\uparrow} A_2 \omega \overset{5}{\uparrow} \overset{4}{\downarrow} A_3 \overset{3}{\downarrow} x_2 \overset{3}{\uparrow} A_4 A_5 \overset{2}{\downarrow} A_3 x_3 \overset{4}{\uparrow} \overset{5}{\downarrow} A_k.$$

Решение

Следуя рассмотренному выше алгоритму, была получена ГСА, приведенная на рис. 4.6.

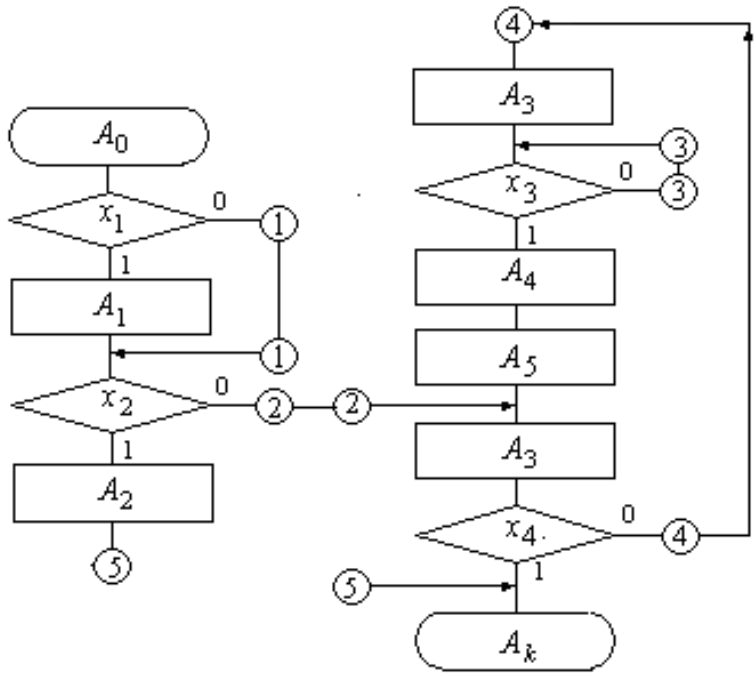


Рис. 4.6. ГСА из примера 4.1

4.1.4. Преобразование ГСА в ЛСА

1. В ГСА отмечаются входы всех вершин, к которым подходит более одной стрелки, натуральными числами от 1 до *k*.

2. Записывается оператор начала A_0. Если после начальной вершины в ГСА следует отмеченная *S* вершина, то после A_0 ставится нижняя стрелка с номером *S*. Далее записывается следующий оператор.

3. После условия x_i ставится верхняя стрелка с номером *S* $-\overset{S}{\uparrow}$ в

том случае, если выход условной вершины по нулю отмечен *S*, в противном случае номер не ставится. Если единичный выход условной вершины отмечен, то после верхней стрелки ставится нижняя стрелка с соответствующим номером, после чего записывается символ оператора, следующего за условной вершиной по выходу “1”.

4. Процедура записи продолжается, пока в строке не появится нижняя стрелка, записанная ранее, или символ A_k до окончания ЛСА. В этом случае вместо нижней стрелки и A_k ставится тождественно ложное условие $\omega\uparrow$.

5. В записанной строке находятся верхние стрелки без номеров, они отмечаются натуральными числами, начиная с *S* +1.

6. Записываются новые строки ЛСА, каждая из которых начинается с нижней стрелки с номером *S* +1, *S* +2…, вслед за которыми записываются символы операторов и условий, соответствующие нулевым выходам условных вершин.

7. Все строки объединяются в одну путем их последовательной записи, причем последним символом в ЛСА будет оператор конца с нижней стрелкой $\downarrow A_k$.

8. Проверяется соответствие верхних и нижних стрелок.

Следует отметить, что важной особенностью ЛСА является возможность неоднозначной записи одного и того же алгоритма.

Пример 4.2. Преобразовать в ЛСА ГСА, заданную на рис. 4.7.

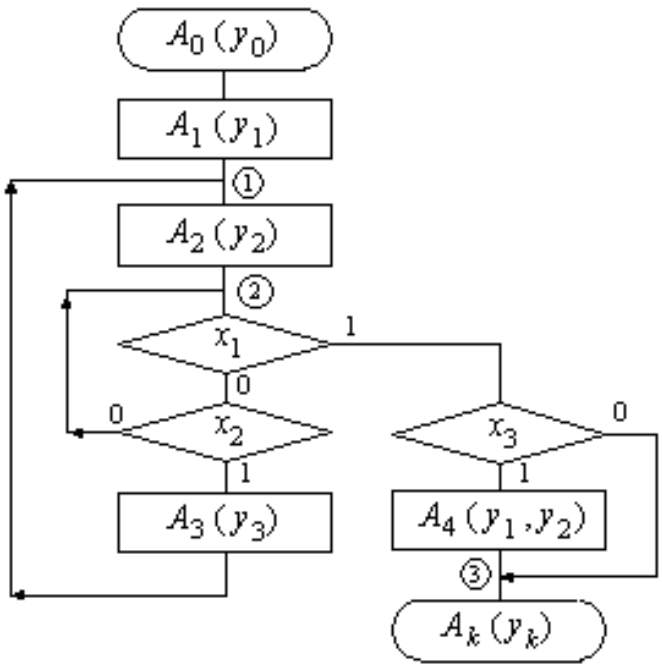


Рис. 4.7. ГСА из примера 4.2

Решение

Заданная ГСА может быть описана несколькими вариантами ЛСА. Например,

1) $A_0(y_0)\,A_1(y_1)\underset{1}{\downarrow}A_2(y_2)\underset{2}{\downarrow}x_1\uparrow\underset{4}{x_2}\uparrow\underset{3}{A_3(y_3)}\;\omega\uparrow\underset{3}{\downarrow}\underset{4}{x_3}\uparrow\underset{2}{A_4(y_1,y_2)}\underset{1}{\downarrow}A_k(y_k),$

2) $A_0(y_0)\,A_1(y_1)\downarrow A_2(y_2)\downarrow x_1\uparrow x_3\uparrow A_4(y_1,y_2)\,\omega\uparrow\downarrow\,x_2\uparrow A_3(y_3)\,\omega\uparrow\downarrow\,A_k(y_k).$

4.2. Переход от начальных языков к стандартным

4.2.1. Построение таблиц переходов по ГСА

Алгоритм функционирования ЦА наиболее часто задается на языке операторных схем алгоритмов (ОСА), к которому относятся язык ГСА и ЛСА. На 1-м этапе абстрактного синтеза ЦА выполняется переход от задания алгоритма его работы на начальном языке к описанию на одном из стандартных языков.

Алгоритм построения таблицы переходов по ГСА для автомата Мура включает следующие этапы:

1. Необходимо пронумеровать все операторные вершины ГСА, предварительно введя пустые операторные вершины, если в ГСА имеются циклы из условных вершин.

2. Каждому оператору ГСА поставить в соответствие вполне определенное состояние автомата, присвоив ему индекс, соответствующий номеру вершины ГСА. Причем одинаковым операторам, стоящим в разных местах ГСА, должны соответствовать разные состояния. Это обеспечивается сквозной нумерацией вершин ГСА. Начальной вершине будет соответствовать исходное состояние автомата. Часто начальную и конечную вершины ГСА совмещают, тогда автомат после выполнения алгоритма переходит в начальное состояние.

3. Необходимо осуществить просмотр всех путей перехода, ведущих от одной операторной вершины к другой, начиная с исходного состояния. Причем каждый такой путь должен соответствовать шагу алгоритма и состоять только из логических условий (условных вершин). Оператор, стоящий вначале пути, должен соответствовать состоянию автомата в момент времени t, а в конце пути – в момент времени $(t+1)$.

Каждому пути из логических условий от одного оператора к другому сопоставить конъюнкцию входных сигналов. Причем в конъюнкцию входной сигнал x_i войдет без отрицания, если выход из условной вершины отмечен единицей, и с отрицанием, если нулем.

4. Построить таблицу переходов автомата Мура. Для этого необходимо анализировать все полученные пути между операторными вершинами, учитывая, что каждая операторная вершина отмечена совокупностью выходных сигналов.

Пример 4.3. Построить прямую таблицу переходов, СКУ и СВФ для автомата Мура по ГСА, приведенной на рис. 4.5.

Решение

Так как в ГСА имеется цикл из условных вершин, введем пустую операторную вершину $A_e(y_e)$, как показано на рис. 4.2, а затем выполним сквозную нумерацию операторных вершин исходной ГСА. Полученная ГСА приведена на рис. 4.8.

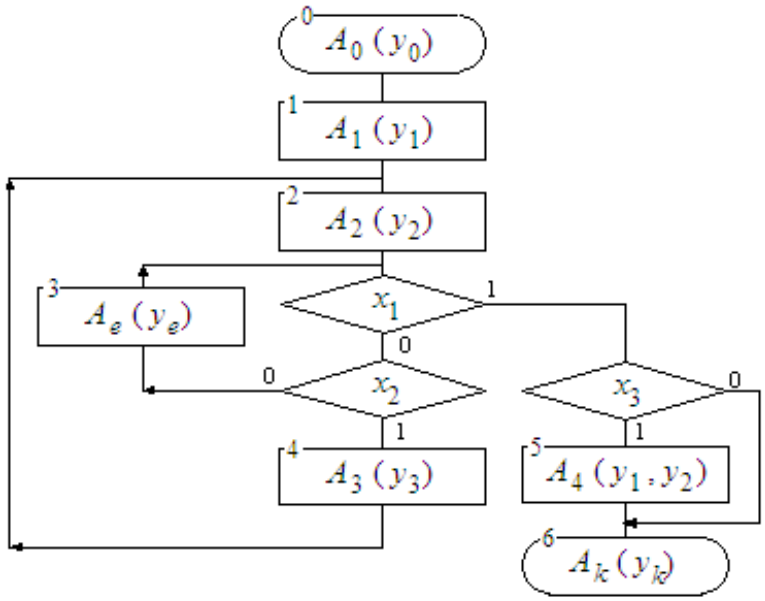


Рис. 4.8. ГСА из примера 4.3

Каждой операторной вершине ГСА поставим в соответствие вполне определенное состояние автомата, присвоив ему индекс, соответствующий номеру вершины ГСА. Следуя рассмотренному выше алгоритму, получим прямую таблицу переходов автомата Мура (табл. 4.1).

На втором этапе абстрактного синтеза решается задача минимизации числа состояний автомата. Для автомата Мура на первом этапе минимизации отыскиваются одинаково отмеченные состояния. Анализ табл. 4.1 показывает, что таких состояний в нашем примере нет. Это значит, что автомат минимален. По табл. 4.1 запишем СКУ и СВФ.

Таблица 4.1. Таблица переходов автомата Мура

$a_i(t)$	$y_i(t)$	$x_{ij}(t)$	$a_j(t+1)$	$y_j(t+1)$
a_0	y_0	1	a_1	y_1
a_1	y_1	1	a_2	y_2
a_2	y_2	$x_1 x_3$ $x_1 \overline{x}_3$ $\overline{x}_1 x_2$ $\overline{x}_1 \overline{x}_2$	a_5 a_6 a_4 a_3	y_1, y_2 y_k y_3 y_e
a_3	y_e	$x_1 x_3$ $x_1 \overline{x}_3$ $\overline{x}_1 x_2$ $\overline{x}_1 \overline{x}_2$	a_5 a_6 a_4 a_3	y_1, y_2 y_k y_3 y_e
a_4	y_3	1	a_2	y_2
a_5	y_1, y_2	1	a_6	y_k

СКУ: **СВФ:**

$a_1(t+1) = a_0$; $y_0 = a_0$;

$a_2(t+1) = a_1 \vee a_4$; $y_1 = a_1 \vee a_5$;

$a_3(t+1) = a_2 \overline{x}_1 \overline{x}_2 \vee a_3 \overline{x}_1 \overline{x}_2$; $y_2 = a_2 \vee a_5$;

$a_4(t+1) = a_2 \overline{x}_1 x_2 \vee a_3 \overline{x}_1 x_2$; $y_3 = a_4$;

$a_5(t+1) = a_2 x_1 x_3 \vee a_3 x_1 x_3$; $y_k = a_6$.

$a_6(t+1) = a_2 x_1 \overline{x}_3 \vee a_3 x_1 \overline{x}_3 \vee a_5$.

Пример 4.4. Построить прямую таблицу переходов СКУ и СВФ по ГСА, показанной на рис. 4.5, для автомата Мили.

Решение

Полученную ранее (см. табл. 4.1) таблицу переходов автомата Мура можно рассматривать и для синтеза эквивалентного ему автомата Мили. При этом в табл. 4.1 необходимо исключить второй столбец ($y_i(t)$) и скорректировать последний ($y_{ij}(t)$), учитывая способ получения выходного сигнала автомата Мили. Получаем таблицу переходов автомата Мили (табл. 4.2).

Этап абстрактного синтеза завершается минимизацией числа состояний автомата. При минимизации автомата Мили в таблице переходов на первом этапе отыскиваются состояния, при переходе из которых под действием одного и того же входного сигнала формируются одинаковые выходные сигналы.

Таблица 4.2. Таблица переходов автомата Мили

$a_i(t)$	$x_i(t)$	$a_i(t+1)$	$y_{ij}(t)$
a_0	1	a_1	y_1
a_1	1	a_2	y_2
a_2	$x_1 \bar{x}_3$ $\bar{x}_1 x_3$ $\bar{x}_1 x_2$ $\bar{x}_1 \bar{x}_2$	a_5 a_6 a_4 a_3	y_1, y_2 y_k y_3 y_e
a_3	$x_1 \bar{x}_3$ $\bar{x}_1 x_3$ $\bar{x}_1 x_2$ $\bar{x}_1 \bar{x}_2$	a_5 a_6 a_4 a_3	y_1, y_2 y_k y_3 y_e
a_4	1	a_2	y_2
a_5	1	a_6	y_k

Если в прямой таблице переходов автомата Мили есть состояния, при переходе из которых под действием одинаковых входных сигналов совпадают не только выходные сигналы, но и состояния автомата, т.е. состояния, переходы из которых полностью совпадают, то такие состояния называются эквивалентными состояниями.

Для нашего примера это состояния (a_1, a_4) и состояния (a_2, a_3). Объединим соответствующие этим состояниям строки и введем следующие обозначения состояний минимального автомата Мили:

$S_0 = a_0$; $S_1 = a_1 \vee a_4$; $S_2 = a_2 \vee a_3$; $S_3 = a_5$; $S_4 = a_6$.

Подставим введенные обозначения в табл. 4.2, получим минимальную таблицу переходов автомата Мили (табл. 4.3).

Таблица 4.3. Минимальная таблица переходов автомата Мили

$S_i(t)$	$x_{ij}(t)$	$S_j(t+1)$	$y_j(t+1)$
S_0	1	S_1	y_1
S_1	1	S_2	y_2
S_2	$x_1 \overline{x}_3$	S_3	y_1, y_2
	$\overline{x}_1 x_3$	S_4	y_k
	$\overline{x}_1 \overline{x}_2$	S_1	y_3
	$x_1 x_2$	S_2	y_e
S_3	1	S_4	y_k

По табл. 4.3 запишем минимальные СКУ и СВФ для автомата Мили:

СКУ: **СВФ:**

$S_1(t+1) = S_0 \vee S_2 \overline{x}_1 x_2$; $y_1 = S_0 \vee S_2 x_1 x_3$;

$S_2(t+1) = S_1 \vee S_2 \overline{x}_1 \overline{x}_2$; $y_2 = S_1 \vee S_2 x_1 x_3$;

$S_3(t+1) = S_2 x_1 x_3$; $y_3 = S_2 \overline{x}_1 x_2$;

$S_4(t+1) = S_2 x_1 \overline{x}_3 \vee S_3$ $y_k = S_2 x_1 \overline{x}_3 \vee S_3$.

4.2.2. Построение таблиц переходов по ЛСА

Методика построения таблицы переходов по ЛСА принципиально мало отличается от аналогичной методики для ГСА. Приведем алгоритм построения таблицы переходов по ЛСА, уточнив некоторые детали отдельных этапов, отличные от алгоритма для ГСА.

1. Пронумеровать все операторные вершины ЛСА, предварительно введя пустые операторные вершины, отмеченные пустым выходным сигналом *Ae* (*ye*), если в ЛСА имеются циклы из условных вершин.

2. Каждому оператору ЛСА поставить в соответствие вполне определенное состояние автомата. Причем одинаковым операторам, стоящим в разных местах ЛСА, должны соответствовать разные состояния. Это обеспечивается сквозной нумерацией операторов ЛСА. Начальному оператору будет соответствовать исходное состояние автомата.

3. Осуществляется просмотр всех путей перехода, ведущих от одного оператора к другому. Причем каждый такой путь должен соответствовать шагу алгоритма и состоять только из логических условий.

4. Каждому пути из логических условий от одного оператора к другому сопоставить конъюнкцию входных сигналов. В конъюнкцию

логических переменных x_i войдет с отрицанием или без отрицания в зависимости от следующих факторов:

а) без отрицания, если $x_i = 1$ (логическое условие истинно), тогда проверяется следующая по порядку записи логическая переменная (логическое условие), если она находится в строке;

б) с отрицанием, если $x_i = 0$ (логическое условие ложно), тогда проверяется по стрелке, стоящей справа от $x_1 \overset{\kappa}{\uparrow}$, следующая логическая переменная (логическое условие), стоящая справа от конца стрелки $\overset{k}{\downarrow}$.

Если в ЛСА имеется безусловный переход, например для фрагмента ЛСА $A_p \omega \overset{k}{\uparrow} \ldots \overset{k}{\downarrow} A_n$, то путь от исходного оператора A_p до конечного в конце пути A_n определится концом стрелки безусловного перехода.

5. Построить таблицу переходов автомата Мура. Для этого необходимо проанализировать все полученные пути между операторами, учитывая, что каждый оператор отмечен совокупностью выходных сигналов.

Пример 4.5. Построить прямую таблицу переходов и СКУ по ЛСА (см. выражение (4.2)):

а) для автомата Мура,

б) для автомата Мили.

Решение

Так как рассматриваемая ЛСА эквивалентна ГСА, рассмотренной выше (рис. 4.9), то, выполнив указанные действия, получим таблицу переходов автомата Мура, приведенную в табл. 4.1.

Дальнейший ход решения поставленной задачи полностью совпадает с решением аналогичной задачи на основе ГСА.

Контрольные вопросы

1. Назовите основные этапы абстрактного синтеза ЦА.
2. Перечислите начальные языки, которые используются для задания цифровых автоматов.
3. Что собой представляет язык ОСА?
4. Дайте характеристику языка ГСА.
5. Каким условиям должна удовлетворять ГСА?
6. При каких условиях в ГСА вводится пустая операторная вершина?

7. Что такое содержательная ГСА?

8. Что собой представляет язык ЛСА? Каковы его достоинства и недостатки?

9. При каких условиях в ЛСА вводится пустой оператор?

10. Сформулируйте алгоритм преобразования ЛСА в ГСА.

11. Сформулируйте алгоритм преобразования ГСА в ЛСА.

12. Назовите основные этапы алгоритма построения прямой таблицы переходов по ЛСА.

13. Назовите этапы построения прямой таблицы переходов по ГСА.

Задания для самостоятельной работы

1. Задана логическая схема алгоритма:

$$A_0(y_0)\overset{1}{\downarrow}x_0\overset{1}{\uparrow}A_1(y_1)x_1\overset{2}{\uparrow}x_2\overset{3}{\uparrow}A_2(y_1,y_2)\omega\overset{42}{\uparrow\downarrow}A_3(y_3)\overset{3}{\downarrow}A_4(y_3,y_4)\overset{4}{\downarrow}A_k(y_k).$$

По ЛСА построить:

− ГСА,

− совмещенную таблицу переходов и выходов автомата Мили,

− отмеченную таблицу переходов автомата Мура,

− прямую таблицу переходов С-автомата,

− обратную таблицу переходов автомата Мили,

− обратную таблицу переходов автомата Мура,

− СКУ и СВФ автоматов Мили и Мура.

2. Задана логическая схема алгоритма:

$$\underset{y_0}{A_0}\overset{0}{\downarrow}x_0\overset{0}{\uparrow}\underset{y_1}{A_1}\overset{1}{\downarrow}\underset{y_2y_3}{A_2}\;x_1\overset{1}{\uparrow}\underset{y_3}{A_3}\,x_2\overset{2}{\uparrow}\underset{y_3y_4}{A_4}\;\omega\overset{k\;3}{\uparrow\downarrow}\underset{y_5}{A_5}\overset{2}{\downarrow}\underset{y_5}{A_6}\,x_3\overset{3\;k}{\uparrow\downarrow}\underset{y_k}{A_k}.$$

А. Преобразовать ЛСА в ГСА.

Б. По ГСА построить:

− совмещенную таблицу переходов и выходов автомата Мили,

− отмеченную таблицу переходов автомата Мура,

− прямую таблицу переходов С-автомата,

− обратную таблицу переходов автомата Мили,

− обратную таблицу переходов автомата Мура,

− СКУ и СВФ автоматов Мили и Мура.

3. Задана граф-схема алгоритма (рис. 4.9).

А. Преобразовать ГСА в ЛСА.

Б. По ЛСА построить:

− совмещенную таблицу переходов и выходов автомата Мили;

− отмеченную таблицу переходов автомата Мура;
− прямую таблицу переходов С-автомата;
− обратную таблицу переходов автомата Мили;
− обратную таблицу переходов автомата Мура;
− СКУ и СВФ автоматов Мили и Мура.

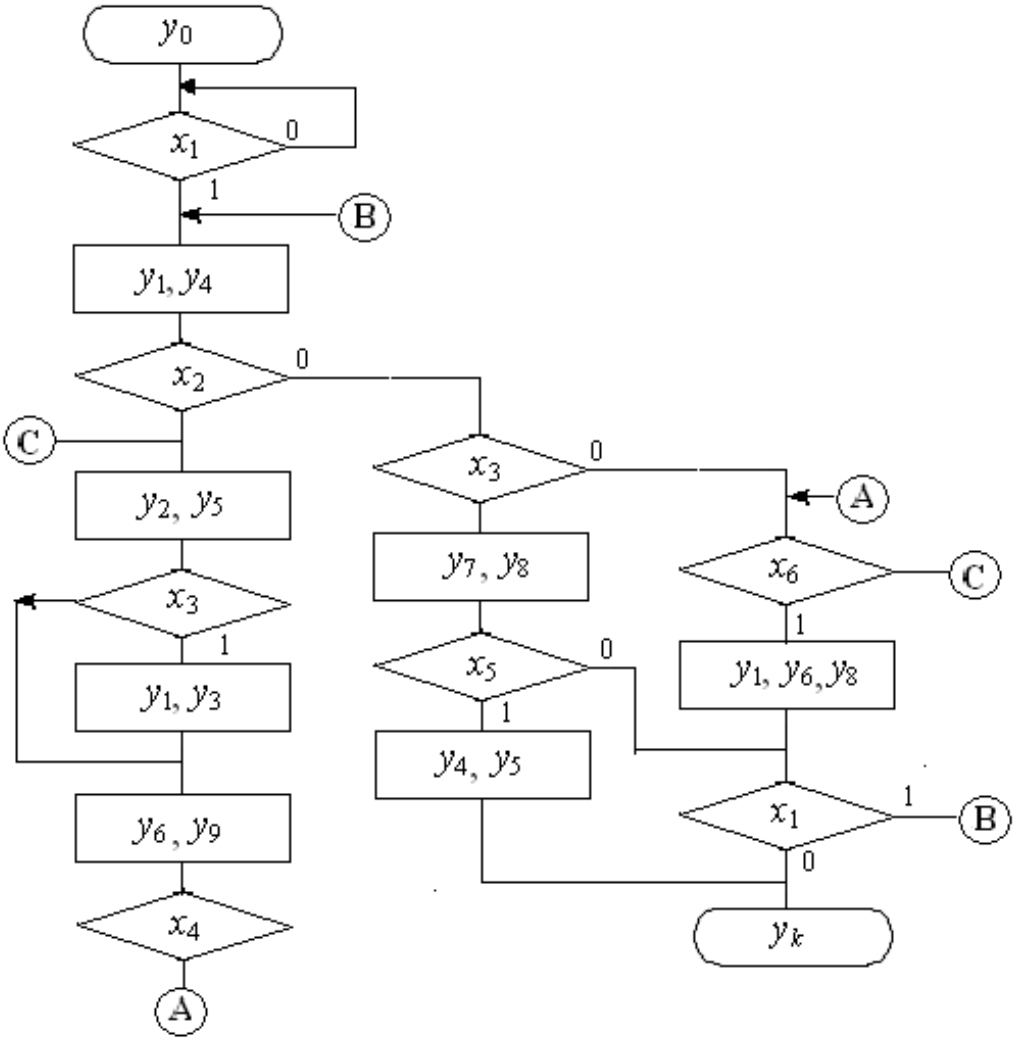


Рис. 4.9. ГСА из примера 3

Глава 5. СИНТЕЗ АВТОМАТОВ С ПАМЯТЬЮ

5.1. Канонический метод структурного синтеза автоматов с памятью

В общем случае задача структурного синтеза автоматов с памятью сводится к нахождению общих приемов построения структурной схемы полученного на этапе абстрактного синтеза автомата Мили, Мура или С-автомата на основе композиции некоторых элементарных автоматов специального вида.

Если абстрактный автомат является лишь математической моделью дискретного устройства, то в структурном автомате учитывается структура входных и выходных сигналов автомата, а также его внутреннее устройство на уровне структурных схем.

Абстрактный С-автомат имеет один входной и два выходных канала, на которые поступают сигналы во входном $Z = \{z_1, ..., z_F\}$ и выходных $W = \{w_1, ..., w_G\}$ и $U = \{u_1, ..., u_H\}$ алфавитах (рис. 5.1,а).

При переходе на структурный уровень каждая буква входного алфавита и выходных W и U алфавитов должны быть представлены в структурном алфавите. Иными словами, каждая буква $z_f \in Z$, $w_g \in W$ и $u_h \in U$ кодируется двоичными векторами или двоичными наборами, число компонентов или длина которых L, N и D, соответственно, равна числу физически реализованных входных и выходных каналов. Очевидно, что в случае двоичного алфавита минимальное число физически реализованных входных каналов L и выходных N и D каналов в автоматах Мили и Мура *a,* соответственно, может быть подсчитано по формулам:

$$L \geq \left\lceil \log_2 F \right\rceil, \quad N \geq \left\lceil \log_2 G \right\rceil, \quad D \geq \left\lceil \log_2 H \right\rceil,$$

где $\left\lceil a \right\rceil$ означает ближайшее целое, большее a или равное ему, если a – целое.

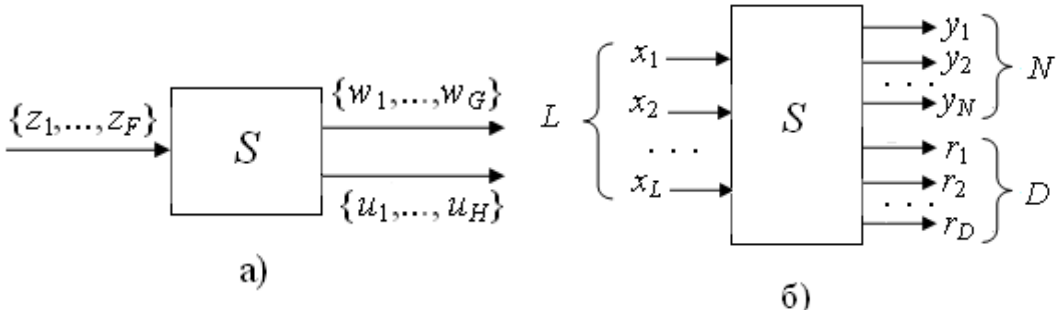


Рис. 5.1. Пример преобразования абстрактного С-автомата

В этом случае символу z_f алфавита $Z = \{z_1, ..., z_F\}$ поставлен во взаимнооднозначное соответствие вектор $X = \{x_1, x_2, ..., x_L\}$, выходному сигналу w_g алфавита $W = \{w_1, ..., w_G\}$ – вектор $Y = \{y_1, y_2, ..., y_N\}$, а выходному сигналу u_h алфавита $U = \{u_1, ..., u_H\}$ – вектор $R = \{r_1, r_2, ..., r_D\}$. Компоненты векторов L, N и D принимают два значения – нуль или единицу (рис. 5.1,б).

На этапе структурного синтеза вначале выбираются элементарные автоматы, из которых затем путем их композиции строится структурная схема полученного на этапе абстрактного синтеза автомата. Если решение задачи структурного синтеза существует, то говорят, что заданная система автоматов структурно полна.

Рассмотрим канонический метод структурного синтеза автоматов с памятью, который оперирует с элементарными автоматами, разделяющимися на два больших класса. Первый класс составляют элементарные автоматы с памятью, имеющие более одного состояния и называемые элементами памяти. Второй класс составляют автоматы без памяти- комбинационные или логические элементы.

Теоретическим обоснованием канонического метода структурного синтеза автоматов служит доказанная В.М. Глушковым ***теорема о структурной полноте*** [6]. *Всякая система элементарных автоматов, которая содержит автомат Мура, обладающий полной системой переходов и полной системой выходов, и какую-либо функциональ нополную систему логических элементов, является структурно полной.*

Канонический метод структурного синтеза предполагает представление схемы С-автомата в виде памяти и комбинационных схем (рис. 5.2). *Память автомата* состоит из предварительно выбранных автоматов памяти $П_1, П_2, \ldots, П_I$, которые служат для отображения текущего состояния автомата. После выбора элементов памяти каждое состояние $a_m \ (m = 1, ..., M)$ кодируется в структурном автомате вектором $Q_1, Q_2, \ldots, Q_I$, компонентами которого являются состояния элементов памяти $П_i$.

При использовании различных методов кодирования состояний автомата количество элементарных автоматов памяти, необходимых для представления M состояний автомата, равно

$$\rceil \log_2 M \lceil \le I \le M. \qquad (5.1)$$

Если $I = \rceil \log_2 M \lceil$, то состояния автомата кодируются кодом минимальной длины. При $I = M$ состояния автомата представляются в унитарном коде, имеющем максимальную длину. То есть каждому символу a_m алфавита $A = \{a_1, ..., a_M\}$ поставлен во взаимно-однозначное соответствие вектор $I = \{q_1, \ldots, q_I\}$, где $q_i \in \{0, 1\}$.

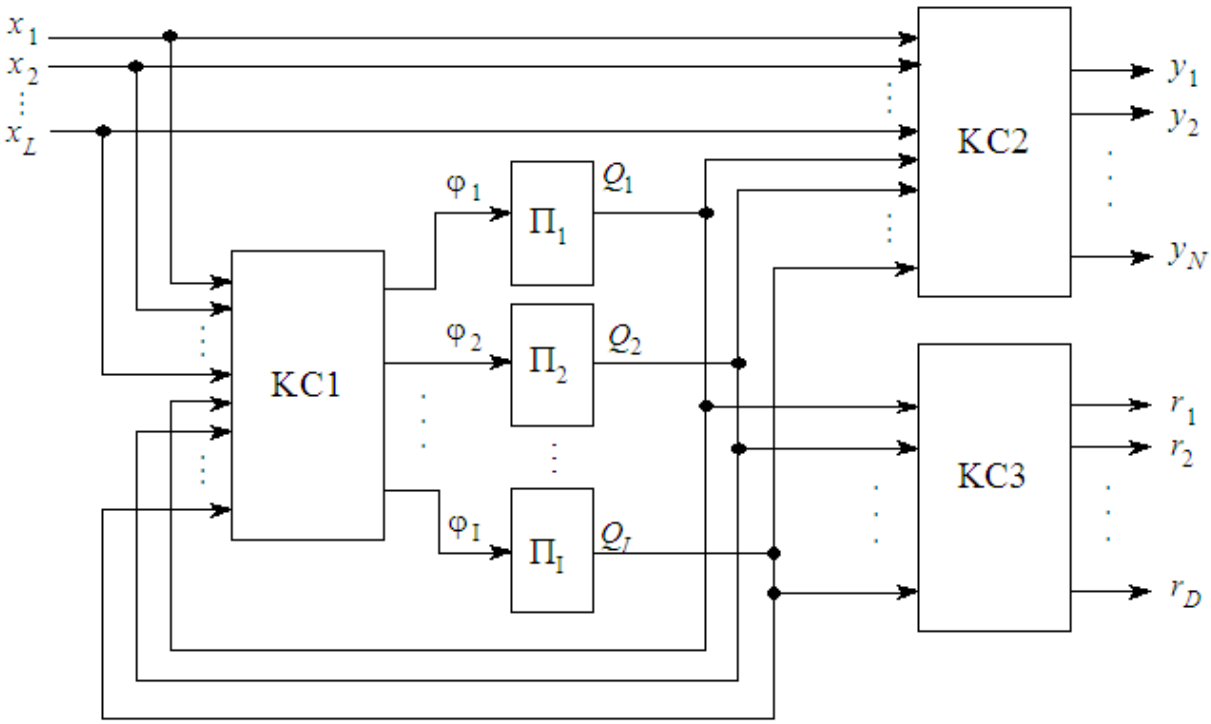


Рис. 5.2. Структурная схема С-автомата

Комбинационная схема КС1 обеспечивает формирование сигналов возбуждения памяти $\varphi_1, \varphi_2, ..., \varphi_I$, под действием которых происходит изменение состояний элементов памяти, $Q_1, Q_2, ..., Q_I$, что соответствует переходу автомата в новое состояние. То есть КС1 реализует функцию переходов автомата.

Комбинационные схемы КС2 и *КС3* обеспечивают формирование выходных сигналов автомата Мили $y_1, y_2, ..., y_N$ и выходных сигналов автомата Мура $r_1, r_2, ..., r_D$, соответственно, то есть реализуют функцию выходов автомата.

Таким образом, после выбора элементов памяти и кодирования состояний синтез структурного автомата сводится к синтезу комбинационных схем, реализующих функции:

$$\chi\pi_i = f_i(x_1, x_2, ..., x_L, Q_1, Q_2, ..., Q_I),\ i = 1, ..., I;$$
$$y_n = f_n(x_1, x_2, ..., x_L, Q_1, Q_2, ..., Q_I),\quad n = 1, ..., N;$$
$$r_d = f_d(Q_1, Q_2, ..., Q_I),\quad d = 1, ..., D.$$

(5.2)

Структурный синтез рассматривается для С-автомата, поскольку эта модель является обобщением моделей Мили и Мура. Если необходимо синтезировать автомат Мили, то можно считать, что в С-автомате не задана функция λ_2 и отсутствует КС3. В случае модели Мура незаданной оказывается функция λ_1 и отсутствует КС2.

Результатом канонического метода структурного синтеза является система логических уравнений, выражающих зависимость выходных сигналов автомата и сигналов, подаваемых на входы элементов памяти, от входных сигналов автомата и сигналов, снимаемых с выхода элементов памяти. Эти уравнения называются каноническими.

Для правильной работы схем нельзя, очевидно, разрешать, чтобы сигналы на входе элементов памяти непосредственно участвовали в образовании выходных сигналов, которые по цепям обратной связи подавались бы в тот же самый момент времени на эти входы. В связи с этим элементами памяти должны быть не автоматы Мили, а полные автоматы Мура, т.е. автоматы, имеющие полную систему переходов и полную систему выходов.

Полнота системы переходов автомата Мура означает, что для любой пары состояний (g_m, g_s) найдется входной сигнал, переводящий первый элемент этой пары g_m во второй g_s. В таком автомате в каждом столбце таблицы переходов должны встречаться все состояния автомата (табл. 5.1).

Таблица 5.1

	w_3	w_2	w_1
	g_1	g_2	g_3
$\chi\pi_1$	g_1	g_3	g_2
$\chi\pi_2$	g_2	g_1	g_3
$\chi\pi_3$	g_3	g_2	g_1

Полнота системы выходов автомата Мура означает, что каждому состоянию автомата поставлен в соответствие свой выходной сигнал, отличный от выходных сигналов других состояний. То есть в таком автомате число выходных сигналов должно быть равно числу состояний автомата, а это означает, что состояния автомата можно отождествить с выходными сигналами.

5.2. Основные этапы канонического метода структурного синтеза

5.2.1. Кодирование

Канонический метод структурного синтеза условно можно разделить на следующие этапы:

1) кодирование,
2) выбор элементов памяти,
3) выбор функционально полной системы логических элементов,
4) построение булевых функций возбуждения памяти и функции выходов,
5) построение функциональной схемы автомата.

На этапе кодирования каждая буква входного, выходного алфавита и алфавита состояний автомата должна быть закодирована двоичными векторами (см. 5.1.1). Процесс замены букв алфавитов *Z, W, U, A* абстрактного автомата двоичными векторами носит название ***кодирования*** и может быть описан таблицами кодирования.

В общем случае каждой кодируемой букве может быть приписан произвольный вектор, но обязательно две различные буквы (одного и того же алфавита) должны кодироваться различными двоичными векторами. Если заменить в исходной таблице переходов буквы на двоичные векторы, то получим структурную таблицу переходов. На этом этап кодирования заканчивается.

5.2.2. Выбор элементов памяти автомата

В качестве элементарных автоматов памяти применяются триггеры. Триггер – это элемент электронных схем с двумя устойчивыми состояниями, который может многократно переходить из одного состояния в другое. Обычно нулевому состоянию соответствует нулевой сигнал, единичному – единичный.

Для удобства выходные сигналы триггера обозначаются теми же буквами, что и его состояние. Триггеры обычно имеют два выхода: прямой Q (называется также “выход 1”) и инверсный $\overline{Q}$ (“выход 0”). В единичном состоянии на выходе Q – высокий уровень сигнала, а в нулевом – низкий. На выходе $\overline{Q}$ – наоборот.

Наиболее широкое распространение получили триггеры типа *RS*,

D, *T*, *JK*. Условные обозначения триггеров приведены на рис. 5.3.

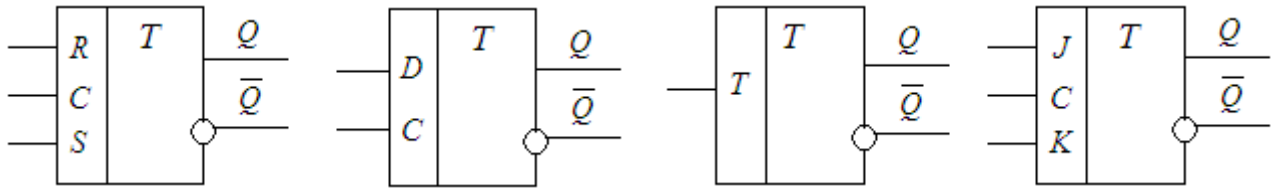


Рис. 5.3. Условное обозначение триггеров

В основном поле помещают букву *T* (для двухступенчатых триггеров – *TT*), через *Q* и $\overline{Q}$ обозначают прямой и инверсный выходы, в левом поле указывают тип входа. При этом для информационных входов триггеров приняты следующие обозначения:

S, *R* – входы для раздельной установки триггера в состояние *Q* = 1 (*Set* – установка) и *Q* = 0 (*Reset* – сброс);

T – счетный вход триггера (*Toggle* – релаксатор);

JK – входы для раздельной установки *JK*-триггера в состояние *Q* = 1 и *Q* = 0;

D – вход для установки триггера в состояние 1 или 0 с временной задержкой (*Delay* – задержка) относительно момента появления информационного сигнала;

C – вход синхронизации для подачи тактовых импульсов (*Clock* – источник синхросигнала).

Триггеры могут различаться по способу записи информации. Они могут быть асинхронные (не тактируемые), когда запись информации осуществляется непосредственно с поступлением информационного сигнала, и синхронные (тактируемые), когда запись информации осуществляется только при подаче разрешающего тактирующего импульса, поступающего на вход синхронизации.

***RS*-триггер**. Асинхронный *RS*-триггер имеет два информационных входа: *S* – установка в состояние *Q* = 1, *R* – установка в состояние *Q* = 0. Последующее состояние триггера *Q*(*t*+1) зависит не только от входных сигналов, но и от предшествующего состояния. Алгоритм работы задается таблицей переходов (табл. 5.2). Функция входов *RS*-триггера приведена в табл. 5.3.

Аналитически функционирование триггера описывается выражени-

ем

$$Q(t+1) = S(t) \vee Q(t) \cdot \bar{R}(t), \quad S(t) \cdot R(t) = 0.$$

(5.3)

Таблица 5.2. Таблица переходов *RS*-триггера

$S(t)$	$R(t)$	$Q(t)$		Режим работы
		0	1	
0	0	0	1	Хранение
0	1	0	0	Сброс в 0
1	0	1	1	Установка 1
1	1	*	*	Запрещен

Таблица 5.3. Функция входов *RS*-триггера

$Q(t)$	$S(t)$ $R(t)$	$Q(t+1)$
0	$00 \vee 01$	0
0	10	1
1	01	0
1	$00 \vee 10$	1

Функция входов *RS*-триггера приведена в табл. 5.3, из которой видно, что триггер сохраняет состояние 0, независимо от сигнала на входе *R* и, аналогично, сохраняет состояние 1, независимо от сигнала на входе *S*. Преобразованная с учетом этого функция входов дана в табл. 5.4.

Таблица 5.4. Преобразованная функция входов *RS*-триггера

$Q(t)$	$S(t)$	$R(t)$	$Q(t+1)$
0	0	–	0
0	1	0	1
1	0	1	0
1	–	0	1

***T*-триггер.** *T*-триггер, или триггер со счетным входом как бы считает поступающие на вход импульсы. То есть изменение состояния триггера происходит при поступлении на его вход одиночного сигнала. Алгоритм работы задается таблицей переходов (табл. 5.5). Функция входов *T*-триггера приведена в табл. 5.6.

Таблица 5.5. Таблица переходов Т-триггера

Таблица 5.6. Функция входов Т-триггера

$T(t)$	$Q(t)$	
	0	1
0	0	1
1	1	0

$Q(t)$	$T(t)$	$Q(t+1)$
0	0	0
0	1	1
1	1	0
1	0	1

Аналитически функционирование *T*-триггера описывается выражением

$$Q(t+1) = \overline{T}(t) \cdot Q(t) \vee T(t) \cdot \overline{Q}(t). \quad (5.4)$$

***D*-триггер.** *D*-триггер реализует функцию временной задержки, т.е. осуществляет задержку поступившего на его вход сигнала на один такт. Функция переходов и функция входов *D*-триггера приведены в табл. 5.7 и 5.8, соответственно.

Таблица 5.7. Таблица переходов D -триггера

$D(t)$	$Q(t)$	
	0	1
0	0	0
1	1	1

Таблица 5.8. Функция входов D -триггера

$Q(t)$	$D(t)$	$Q(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1

D-триггер реализует функцию вида

$$Q(t+1) = D(t). \quad (5.5)$$

***JK*-триггер**. Универсальный по своим возможностям *JK*-триггер нашел наибольшее применение. Алгоритм работы задается таблицей переходов (табл. 5.9).

Таблица 5.9. Таблица переходов JK-триггера

$J(t)$	$K(t)$	$Q(t)$	
		0	1
0	0	0	1
0	1	0	0
1	0	1	1
1	1	1	0

Функция входов и преобразованная функция входов даны в

табл. 5.10 и 5.11, соответственно. Функцию переходов *JK*-триггера можно представить в виде булевой функции

$$Q(t+1) = \overline{K}(t) \cdot Q(t) \vee J(t) \cdot \overline{Q}(t). \quad (5.6)$$

Входы *J* и *K* соответствуют входам установки в состояния $Q = 1$ и $Q = 0$, соответственно. В отличие от *RS*-триггера, в *JK*-триггере сигналы 1 могут одновременно прийти на входы *J* и *K*. При этом состояние триггера изменяется на противоположное, т.е. при $J = K$ схема ведет себя, как триггер со счетным входом.

Таблица 5.10. Функция входов *JK*-триггера

$Q(t)$	$J(t)$ $K(t)$	$Q(t+1)$
0	$00 \vee 01$	0
0	$10 \vee 11$	1
1	$01 \vee 11$	0
1	$00 \vee 10$	1

Таблица 5.11. Преобразованная функция входов *JK*-триггера

$Q(t)$	$J(t)$	$K(t)$	$Q(t+1)$
0	0	-	0
0	1	-	1
1	-	1	0
1	-	0	1

JK-триггер удобен тем, что при различных вариантах подключения его входов можно получить схемы, функционирующие как *D*-, *T*-, и *RS*-триггеры (рис. 5.4).

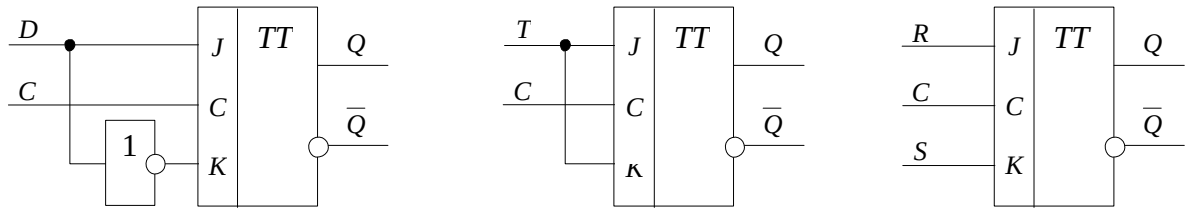


Рис. 5.4. Способы использования JK-триггера

5.2.3. Выбор функционально полной системы логических элементов

После выбора элементов памяти и кодирования состояний автомата синтез структурного автомата сводится к синтезу КС, реализующих функции (5.2). Для реализации КС необходимо использовать систему логических элементов, которая должна быть функционально полной, т.е. допускать реализацию любой булевой функции на основе принципа суперпозиции.

5.2.4. Построение булевых функций возбуждения памяти и функции выходов

Кодирование и выбор системы элементов однозначно определяют комбинационную часть автомата.

Вначале строится таблица функции возбуждения памяти автомата, получившая название таблицы функции возбуждения. Из этой таблицы записываются канонические уравнения функций возбуждения, которые затем могут быть минимизированы любым известным способом. Исходными данными для построения таблицы функций возбуждения являются структурная таблица переходов автомата и таблица переходов элемента памяти. Подробное заполнение и использование таблицы функций возбуждения рассматривается ниже в примерах 5.1 и 5.2.

Получение канонических уравнений булевых функций выходов структурного автомата проще и может быть сделано непосредственно по структурной таблице выходов автомата.

Уравнения функций выходов не зависят от типа используемых элементов памяти, но зависят от их количества. Следует отметить, что уравнения функций выходов не изменяются при переходе к синтезу автомата на триггерах другого типа.

5.2.5. Построение функциональной схемы автомата

На основании полученных выражений для булевых функций возбуждения памяти автомата и булевых функций выходов автомата строится комбинационная схема функций возбуждения памяти КС1 и комбинационные схемы формирования выходных сигналов автомата (КС2 или КС3). Элементы памяти подключаются к построенным комбинационным схемам, как показано на рис. 5.2.

5.3. Примеры канонического метода структурного синтеза

Пример 5.1. Синтезировать автомат Мили *S*, заданный совмещенной таблицей переходов и выходов (табл. 5.12). В качестве элементарных автоматов памяти использовать *T*-триггеры, комбинационные схемы реализовать в булевом базисе.

Таблица 5.12. Совмещенная таблица переходов и выходов автомата Мили *S*

	a_1	a_2	a_3	a_4
z_1	a_2 / ω_1	a_2 / ω_1	a_1 / ω_2	a_1 / ω_4
z_2	a_4 / ω_5	a_3 / ω_3	a_4 / ω_4	a_3 / ω_5

Решение

1 этап. Перейдем к структурному представлению, для чего закодируем входные и выходные сигналы и состояния автомата *S*. В общем случае кодирование произвольное, но две различные буквы одного алфавита должны кодироваться различными векторами.

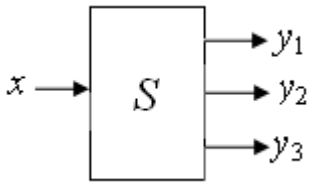
В абстрактном автомате *S* четыре состояния (a_1, a_2, a_3, a_4) – M = 4. Два абстрактных входных сигнала (z_1, z_2) – F = 2 и пять выходных сигналов (ω_1, ω_2, ω_3, ω_4, ω_5) – G = 5.

Минимальное число входных каналов:

$L =]\log_2 F[=]\log_2 2[= 1;\ X = \{x\}.$

Минимальное число выходных каналов:

$N =]\log_2 G[=]\log_2 5[= 3;\ Y = \{y_1, y_2, y_3\}.$



Минимальное число элементов памяти (триггеров), необходимых для представления состояний автомата (M = 4), равно:

$I =]\ \log_2 M\ [=]\ \log_2 4\ [= 3.$

$Q_1\ Q_{2(2)}$ – код состояния автомата.

Результаты кодирования состояний, входных и выходных сигналов автомата *S* представлены в табл. 5.13 – 5.15.

Таблица 5.13. Кодирование состояний автомата S

Таблица 5.14. Кодирование входных сигналов автомата S

Таблица 5.15. Кодирование выходных сигналов автомата S

A	Q_1	Q_2
a_1	0	0
a_2	0	1
a_3	1	1
a_4	1	0

z	x
z_1	0
z_2	1

ω	y_1	y_2	y_3
ω_1	0	0	0
ω_2	0	0	1
ω_3	0	1	0
ω_4	0	1	1
ω_5	1	0	0

Заменив в совмещенной таблице переходов и выходов автомата Мили *S* (табл. 5.12) состояния, входные и выходные сигналы их кодами, получим совмещенную таблицу переходов структурного автомата Мили (табл. 5.16).

Таблица 5.16. Совмещенная таблица переходов и выходов структурного автомата *S*

x	$Q_1\,Q_2$			
	00	01	11	10
0	01 / 000	01 / 000	00 / 010	00 / 011
1	10 / 100	11 / 010	10 / 011	11 / 100

2 этап. Выбор элементов памяти автомата. В этом примере для построения памяти автомата будем использовать *T*-триггер.

3 этап. Выбор функционально полной системы логических элементов. В этом примере комбинационные схемы КС1 и КС2 будем строить на элементах булевого базиса.

4 этап. На этом этапе производится построение булевых функций возбуждения памяти и функций выходов:

$$y_n = f_n\left(x, Q_1, Q_2\right), \quad n = 1, 2, 3;$$
$$T_i = f_i\left(x, Q_1, Q_2\right), \quad i = 1, 2, \tag{5.7}$$

где Q_1, Q_2 – функции обратной связи от памяти автомата к его КС,

T_1, T_2 – функции возбуждения элементов памяти автомата,

y_1, y_2, y_3 – функции выходов.

Найдем функции возбуждения памяти T_1 и T_2. Для этого воспользуемся таблицей переходов *T*-триггера (см. табл. 5.5). Учитывая, что $T_1 = 1$ и $T_2 = 1$ только в случае перехода соответствующего триггера из нулевого состояния в единичное и наоборот, используя структурную таблицу переходов автомата (см. табл. 3.16), получим таблицу функций возбуждения памяти (табл. 5.17). Например, при переходе автомата *S* из состояния 00 в состояние 10 под действием входного сигнала 1

(первый столбец, вторая строка табл. 5.16), на входы его памяти должен поступить векторный сигнал функции возбуждения 10. Занесем этот результат в соответствующее место таблицы функций возбуждения (табл. 5.17) – на пересечении первого столбца и второй строки. Поскольку функции возбуждения памяти T_1 и T_2 зависят от тех же переменных x, Q_1, Q_2, столбцы и строки табл. 5.16 и табл. 5.17 отмечены одинаково. Аналогичным образом для остальных переходов в табл. 5.17 получим всю таблицу функций возбуждения памяти автомата S.

Таблица 5.17. Функции возбуждения памяти автомата S

x	$Q_1\ Q_2$			
	00	01	11	10
0	01	00	11	10
1	10	10	01	01

Перейдем от табл. 5.16 и 5.17 к табл. 5.18. Последняя представляет собой более привычную для нас таблицу истинности системы булевых функций.

Таблица 5.18. Таблица истинности булевых функций возбуждения памяти и функций выходов

x	Q_1	Q_2	T_1	T_2	y_1	y_2	y_3
0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	1	1	0	1	0
1	0	0	1	0	1	0	0
1	0	1	1	0	0	1	0
1	1	0	0	1	1	0	0
1	1	1	0	1	0	1	1

Из табл. 5.17 имеем:

$$
\begin{aligned}
T_1 &= \bar{x}\cdot Q_1\cdot\bar{Q}_2 \vee \bar{x}\cdot Q_1\cdot Q_2 \vee x\cdot\bar{Q}_1\cdot\bar{Q}_2 \vee x\cdot\bar{Q}_1\cdot Q_2 = \vee\,(2,3,4,5);\\
T_2 &= \bar{x}\cdot\bar{Q}_1\cdot\bar{Q}_2 \vee \bar{x}\cdot Q_1\cdot Q_2 \vee x\cdot Q_1\cdot\bar{Q}_2 \vee x\cdot Q_1\cdot Q_2 = \vee\,(0,3,6,7);\\
y_1 &= x\cdot\bar{Q}_1\cdot\bar{Q}_2 \vee x\cdot Q_1\cdot\bar{Q}_2 = \vee\,(4,6);\\
y_2 &= \bar{x}\cdot Q_1\cdot\bar{Q}_2 \vee \bar{x}\cdot Q_1\cdot Q_2 \vee x\cdot\bar{Q}_1\cdot Q_2 \vee x\cdot Q_1\cdot Q_2 = \vee\,(2,3,5,7);\\
y_3 &= \bar{x}\cdot Q_1\cdot\bar{Q}_2 \vee x\cdot Q_1\cdot Q_2 = \vee\,(2,7).
\end{aligned}
\qquad (5.8)
$$

Ясно, что табл. 5.18 можно не строить, а выражения (5.8) полу-

чить непосредственно по табл. 5.16 и 5.17.

5 *этап.* Построение функциональной схемы автомата. Не будем рассматривать вопросы минимизации комбинационных схем автомата и непосредственно по выражениям (5.8) построим логическую схему автомата *S* на элементах И, ИЛИ, НЕ (рис. 5.5).

Следует отметить, что при синтезе автоматов на *D*-триггерах процесс построения функций возбуждения памяти может быть упрощен. Как видно из табл. 3.7, состояние, в которое переходит *D*-триггер, совпадает с поступившим на его вход сигналом. В связи с этим таблица функций возбуждения памяти синтезируемого автомата *S* будет полностью совпадать со структурной таблицей переходов этого автомата (см. - табл. 5.16).

Пример 5.2. Синтезировать частичный автомат Мура S_1, заданный отмеченной таблицей переходов (табл. 5.19). В качестве элементарных автоматов памяти использовать *RS*-триггер, комбинационные схемы реализовать в булевом базисе.

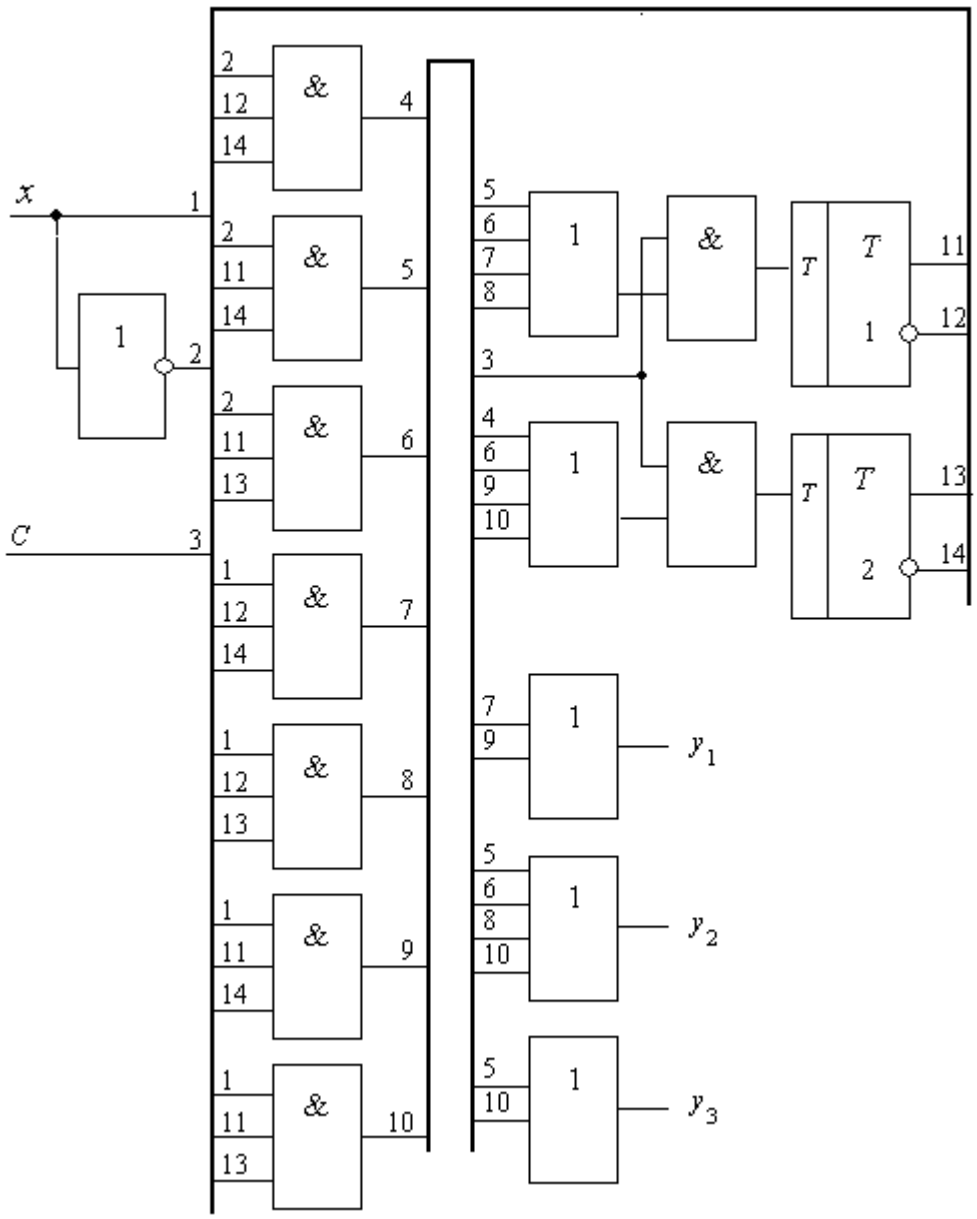


Рис. 5.5. Логическая схема автомата Мили S
из примера 5.1

Таблица 5.19. Отмеченная таблица переходов автомата Мура S_1

	μ_3	μ_2	μ_3	μ_1
	a_1	a_2	a_3	a_4
z_1	a_1	-	a_1	a_4
z_2	a_2	a_3	-	a_1
z_3	-	a_4	a_4	a_2

Решение

Закодируем входные и выходные сигналы и состояния автомата S_1. Три абстрактных входных сигнала (z_1, z_2, z_3) и три выходных сигнала (ω_1, ω_2, ω_3) требуют два входных (x_1, x_2) и два выходных (y_1, y_2) канала. Так как в абстрактном автомате четыре состояния, то структурный ав-

томат будет иметь два *RS*-триггера – T_1 и T_2 . Q_1, Q_2 – код состояния автомата S_1. Результаты кодирования представлены в табл. 5.20 – 5.22.

Таблица 5.20. Кодирование входных сигналов

z	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0

Таблица 5.21. Кодирование выходных сигналов

ω	y_1	y_2
ω_1	0	1
ω_2	1	0
ω_3	0	0

Таблица 5.22. Кодирование состояний автомата

A	Q_1	Q_2
a_1	0	0
a_2	0	1
a_3	1	1
a_4	1	0

Заменив в табл. 5.19 состояния, входные и выходные сигналы их кодами, получим структурную таблицу переходов автомата Мура S_1 (табл. 5.23).

Комбинационные схемы структурного автомата Мура S_1 реализуют функции:

$$y_n = f_n(Q_1, Q_2), \quad n = 1, 2;$$
$$R_i = f_i(x_1, x_2, Q_1, Q_2), \quad i = 1, 2;$$
$$S_i = f_i(x_1, x_2, Q_1, Q_2), \quad i = 1, 2, \tag{5.9}$$

где R_i, S_i – функции возбуждения элементов памяти автомата.

Таблица 5.23. Отмеченная таблица переходов структурного автомата

$y_1\ y_2$	0 0	1 0	0 0	0 1
$Q_1\ Q_2$ / $x_1\ x_2$	0 0	0 1	1 1	1 0
0 0	0 0	–	0 0	1 0
0 1	0 1	1 1	–	0 0
1 0	–	1 0	1 0	0 1

Таблица 5.24. Функции возбуждения памяти структурного автомата

$x_1\ x_2$	$Q_1\ Q_2$			
	0 0	0 1	1 1	1 0
0 0	0-0-	–	0101	-00-
0 1	0-10	10-0	–	010-
1 0	–	1001	-001	0110

Выражения для функций выходов y_1, y_2 получим непосредственно из табл. 5.23:

$$y_1 = \bar{Q}_1 Q_2;\ y_2 = Q_1 \bar{Q}_2. \tag{5.10}$$

Таблицу функций возбуждения памяти (табл. 5.24) строим на основании структурной таблицы переходов (табл. 5.23). Поскольку в *RS*-триггере имеются два входных канала, на пересечении строк и столбцов в таблице функций возбуждения памяти будем указывать значения

четырех функций: S_1, R_1, S_2, R_2. Функция входов *RS*-триггера приведена в табл. 5.3. Эта таблица дает систему подстановок при переходе от таблицы переходов структурного автомата к таблице его функций возбуждения. Прочерк (–) означает, что переход не зависит от сигнала на данном входе. Функции возбуждения триггеров не полностью определенные, так как набор входных сигналов $x_1 = 1$ и $x_2 = 1$ никогда на вход автомата не поступает.

Из табл. 5.24 получаем выражения для функций S_i, R_i:

$$
\begin{aligned}
S_1 &= \overline{x}_1 \cdot x_2 \cdot \overline{Q}_1 \cdot Q_2 \vee x_1 \cdot \overline{x}_2 \cdot \overline{Q}_1 \cdot Q_2;\\
R_1 &= \overline{x}_1 \cdot x_2 \cdot Q_1 \cdot \overline{Q}_2 \vee x_1 \cdot \overline{x}_2 \cdot Q_1 \cdot \overline{Q}_2 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot Q_1 \cdot Q_2;\\
S_2 &= \overline{x}_1 \cdot x_2 \cdot \overline{Q}_1 \cdot \overline{Q}_2 \vee x_1 \cdot \overline{x}_2 \cdot Q_1 \cdot \overline{Q}_2;\\
R_2 &= \overline{x}_1 \cdot \overline{x}_2 \cdot Q_1 \cdot Q_2 \vee x_1 \cdot \overline{x}_2 \cdot \overline{Q}_1 \cdot Q_2 \vee x_1 \cdot \overline{x}_2 \cdot Q_1 \cdot Q_2.
\end{aligned}
\qquad (5.11)
$$

Для минимизации функций возбуждения триггеров (5.11) используем карты Карно (рис. 5.6).

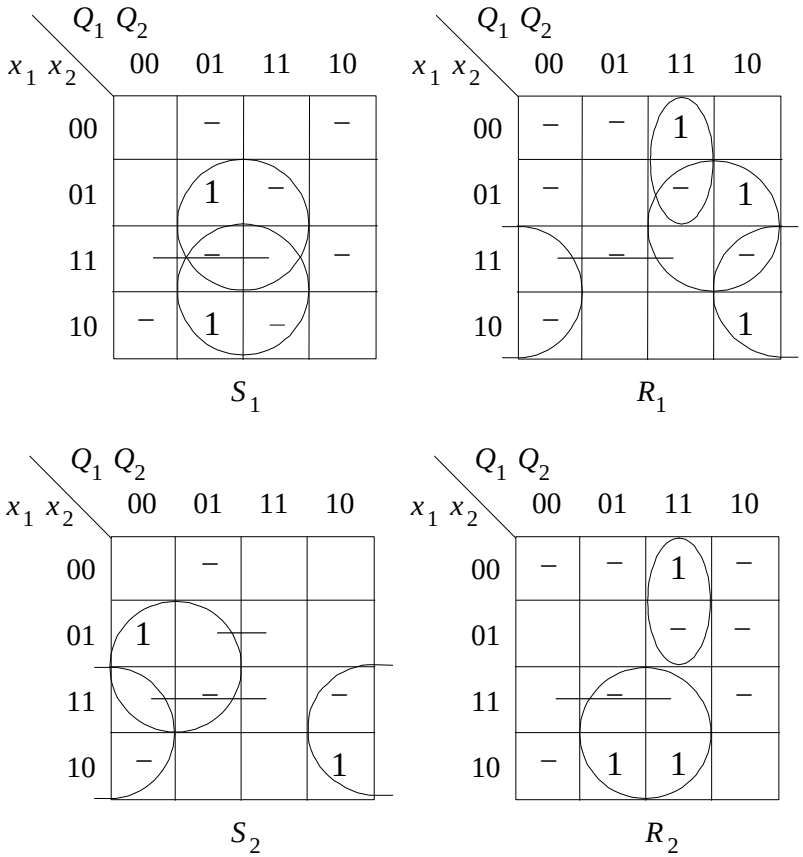


Рис. 5.6. Карты Карно для определения функций возбуждения RS-триггеров

Выполнив совместную минимизацию системы булевых функций, получим:

$$S_1 = x_1 \cdot Q_2 \vee x_2 \cdot Q_2;$$
$$R_1 = x_2 \cdot \overline{Q}_2 \vee x_2 \cdot Q_1 \vee \overline{x}_1 \cdot Q_1 \cdot Q_2;$$
$$S_2 = x_1 \cdot \overline{Q}_2 \vee x_2 \cdot \overline{Q}_1;$$
$$R_2 = x_1 \cdot Q_2 \vee \overline{x}_1 \cdot Q_1 \cdot Q_2. \qquad (5.12)$$

По выражениям (5.10) и (5.12) построим логическую схему автомата Мура S_1 на элементах И, ИЛИ, НЕ (рис. 5.7).

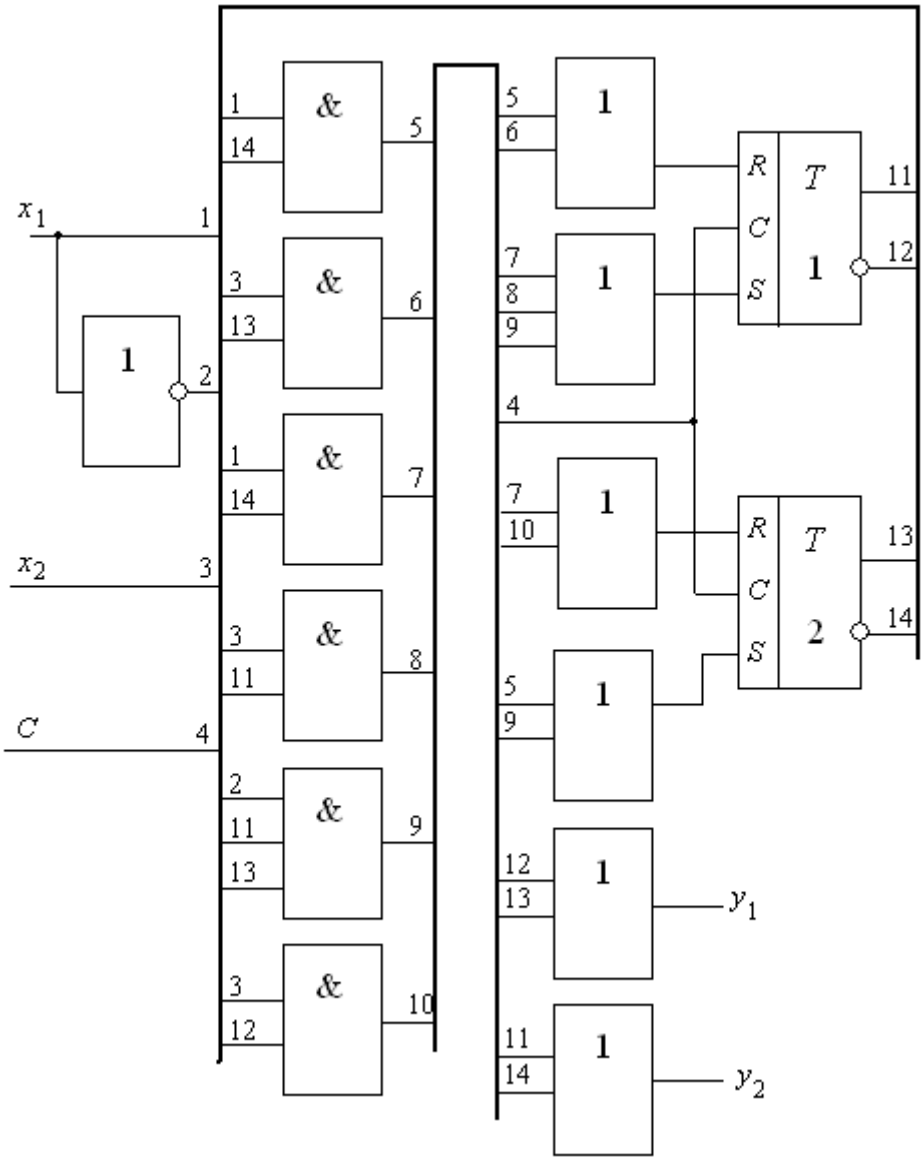


Рис. 5.7. Логическая схема автомата Мура S_1 из примера 5.2

Рассмотренная методика структурного синтеза цифровых автоматов применима лишь для несложных цифровых устройств с небольшим

числом входов и состояний.

Синтез структурного автомата на *JK*-триггерах производится аналогично.

5.4. Гонки в автомате

Переход автомата из одного состояния в другое осуществляется за счет изменения состояния элементов памяти Q_1, Q_2, ..., Q_I, которое производится сигналами возбуждения $\varphi_1, \varphi_2, ..., \varphi_I$ (см. рис. 5.2), вырабатываемыми в комбинационной схеме КС1 и поступающими на входы триггеров. Сигналы возбуждения поступают на входы триггеров не одновременно, так как логические цепи, их реализующие, возможно, не идентичные по количеству используемых в них логических элементов, характеризуются различным временем задержки сигнала. К тому же триггеры имеют различное время переключения. Из-за различия во временных характеристиках триггеры изменяют свои состояния не одновременно.

Если при переходе автомата из одного состояния в другое должны изменить свои состояния сразу несколько запоминающих элементов, то между ними начинаются состязания или гонки. Тот элемент, который выиграет эти состязания, то есть 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 законом функционирования.

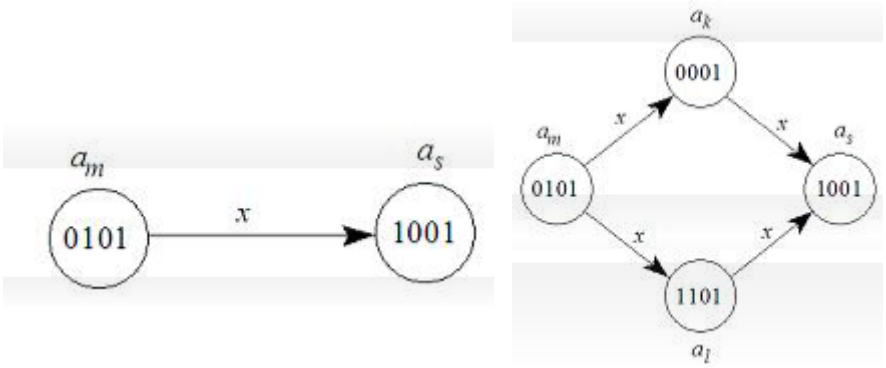
Например, при переходе автомата a_m в состояние a_s под действием входного сигнала x (рис. 3.7,а) автомат может оказаться в некотором промежуточном состоянии a_k или a_l. Если затем при том же входном сигнале автомат из a_k и a_l перейдет в состояние a_s, то такие состязания являются допустимыми, или *некритическими*.

Если же в этом автомате есть переход, например из a_k в a_j № a_s под действием того же сигнала x (рис. 3.7,б), то автомат может перейти в a_j, а не в a_s и правильность его работы тем самым будет нарушена. Гонки, приводящие автомат в устойчивое состояние, не соответствующее закону функционирования, называются *критическими*.

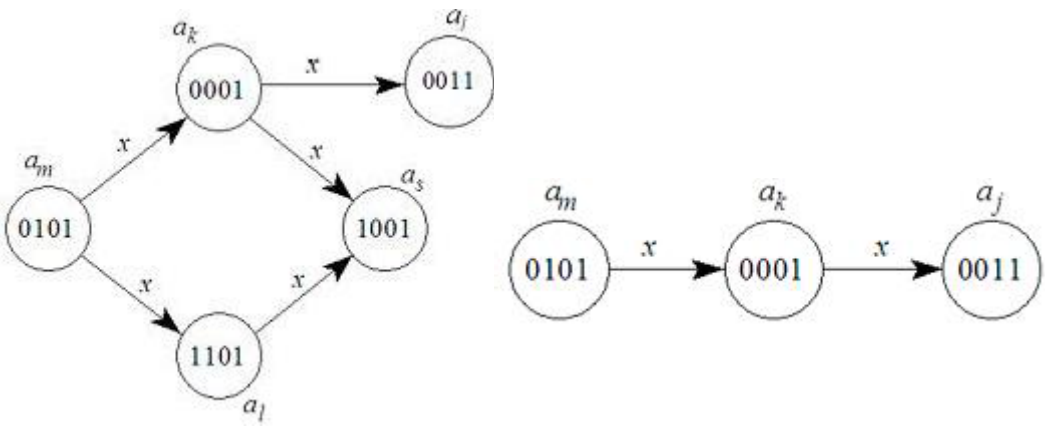
Гонки в автомате связаны с разбросом во временных параметрах сигналов, проходящих через логические и запоминающие элементы, и

имеют место в любой реальной логической схеме.

Для обеспечения заданного закона функционирования автомата необходимо исключить возможность появления критических гонок. Критические гонки в схеме могут исключаться различными способами.



а)



б)

Рис. 5.8. Примеры состязаний между элементами памяти: а – некритические, б – критические

Методы устранения гонок

В схемах на рис. 5.6 и 5.7 гонки устраняются путем ограничения длительности сигнала *C*, поступающего в цепь синхронизации. Этот способ устранения гонок называется способом *импульсной синхронизации*. Если длительность импульса t_c меньше самого короткого пути прохождения тактированного сигнала обратной связи по комбинационной схеме КС1, то к моменту перехода в промежуточное состояние a_k (рис. 5.8,б) сигнал *C* равен нулю, что и исключает гонки. Естественно, что такой способ устранения гонок приемлем только в том случае, если элементы памяти могут переключаться под действием импульсов с длительностью t_c.

Другой способ ликвидации гонок заключается в разделении во времени процессов выработки сигналов возбуждения и процесса переключения состояний. Такого рода разделение достигается использованием *двойной* (*двухступенчатой*) памяти (рис. 5.9). При использова-

Оглавление