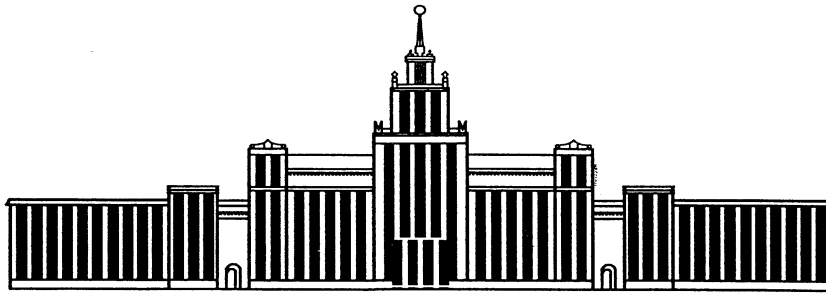
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ



ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

519.7(07)
Б246

Барбасова Т.А., Гудилин А.Е.

ТЕОРИЯ КОНЕЧНЫХ АВТОМАТОВ

Учебное пособие

Челябинск
2014

Министерство образования и науки Российской Федерации
Южно-Уральский государственный университет
Кафедра автоматики и управления

519.7 (07)
Б246

Барбасова Т.А., Гудилин А.Е.

ТЕОРИЯ КОНЕЧНЫХ АВТОМАТОВ

Учебное пособие

Челябинск
Издательский центр ЮУрГУ
2014

УДК 519.713 (075.8)
Б246

Одобрено
учебно-методической комиссией
приборостроительного факультета

Рецензент:
д.т.н. С.И. Лукьянов, к.т.н. П.Л. Самсонов

Барбасова, Т.А.

Б246 Теория конечных автоматов: учебное пособие / Т.А. Барбасова, А.Е. Гудилин. – Челябинск: Издательский центр ЮУрГУ, 2014. – 118 с.

Учебное пособие предназначено для студентов очной и заочной форм обучения специальности 220400 (ФГОС 3), 27.03.04 (ФГОС 3+) “Управление и информатика в технических системах”

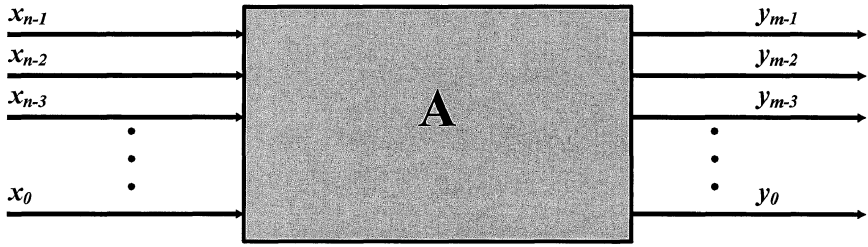
Предлагаемое учебное пособие содержит вопросы формального описания цифровых автоматов применяется аппарат алгебры логики, созданной английским математиком Дж. Булем, рассматриваются теоретические основы построения цифровых автоматов как преобразователей двоичных цифровых сигналов. Изложены свойства элементарных функций алгебры логики. Подробно разобраны методы минимизации функций алгебры логики. Описан синтез абстрактного и структурного цифрового автомата.

УДК 519.713 (075.8)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В курсе «Теория конечных автоматов» рассматриваются теоретические основы построения цифровых автоматов как преобразователей двоичных цифровых сигналов. Все системы обработки цифровой информации (в том числе и цифровые вычислительные машины – ЭВМ) могут в общем виде представлены как кодопреобразователи (рис. 1.1) .

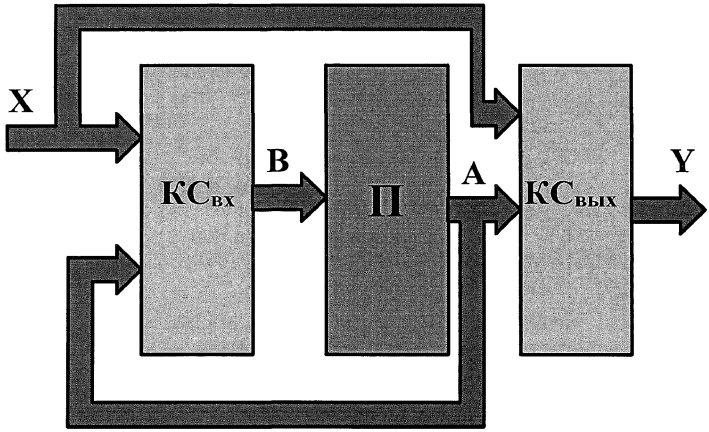


$x_{n-1}\,x_{n-2}\,x_{n-3}..x_0$ – входной цифровой n - разрядный двоичный код;
$y_{m-1}y_{m-2}\,y_{m-3}...y_0$ – выходной цифровой m - разрядный двоичный код;
А – оператор преобразования.

Рис. 1.1. Обобщенная структура системы переработки цифровой информации

Особенностью цифрового автомата является зависимость оператора преобразования А от предыдущих состояний кодопреобразователя, то есть наличие памяти у цифрового автомата. В частном случае отсутствия памяти у цифрового автомата, он является логической схемой. Таким образом, предметами исследования в теории цифровых автоматов являются как собственно цифровые автоматы (системы с памятью), так и автоматы без памяти или логические схемы.

Наиболее разработана теория цифровых автоматов применительно к канонической структуре цифрового автомата, представленной на рис.2. Для дальнейшего рассмотрения используется только эта структура цифрового автомата.



КС$_{вх}$ – входная комбинационная схема;
П – память; КС$_{вых}$ – выходная комбинационная схема;
X – входной цифровой код; B – код возбуждения памяти;

Рис. 1.2. Каноническая структурная схема цифрового автомата

По структурной схеме цифрового автомата видно, что входные коды входной и выходной комбинационных схем получаются в результате конкатенации (объединения) входного кода и кода состояния памяти цифрового автомата.

1. СИСТЕМЫ ИСЧИСЛЕНИЯ

1.1. Славянская глаголическая десятеричная

Эта система была создана для обозначения чисел в священных книгах западных славян. Использовалась она нечасто, но достаточно долго. По организации она в точности повторяет греческую нумерацию. Использовалась она с VIII по XIII в.

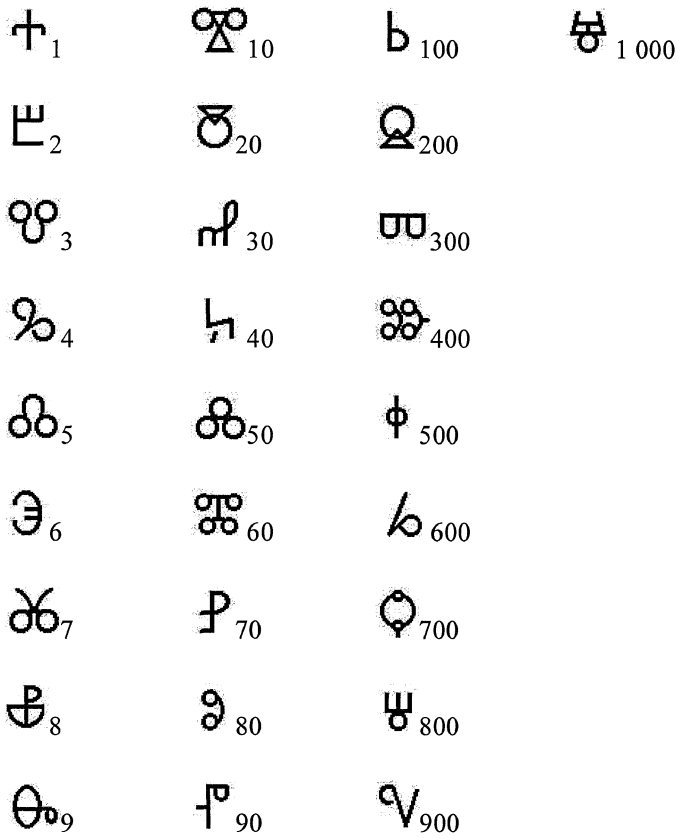
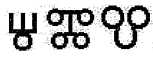


Рис. 1.3

Числа записывали из цифр так же слева, направо, от больших к меньшим цифрам. Если десятков, единиц, или какого-то другого разряда не было, то его пропускали. Такая запись числа аддитивная, то есть в ней используется только сложение:

 = 800+60+3 = 863

Для того чтобы не перепутать буквы и цифры, использовались титла – горизонтальные черточки над числами, или точки.

1.2. Славянская кириллическая десятеричная алфавитная

Эта нумерация была создана вместе со славянской алфавитной системой для перевода священных библейских книг для славян греческими монахами братьями Кириллом и Мефодием в IX веке. Эта форма записи чисел получила большое распространение в связи с тем, что имела полное сходство с греческой записью чисел. До XVII века эта форма записи чисел была официальной на территории современной России, Белоруссии, Украины, Болгарии, Венгрии, Сербии и Хорватии. До сих пор православные церковные книги используют эту нумерацию.

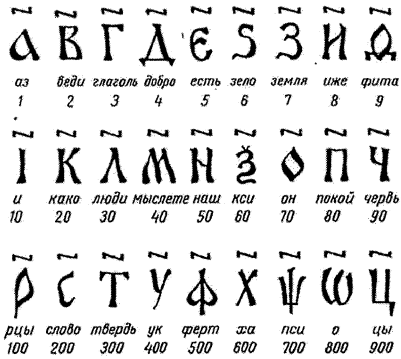


Рис. 1.4

Числа записывали из цифр так же слева, направо, от больших к меньшим. Числа от 11 до 19 записывались двумя цифрами, причем единица шла перед десятком:

АІ - 14

Читаем дословно "четырнадцать" – "четыре и десять". Как слышим, так и пишем: не 10+4, а 4+10, – четыре и десять. Числа от 21 и выше записывались наоборот, сначала писали знак полных десятков.

Запись числа, использованная славянами аддитивная, то есть в ней используется только сложение:

 = 800+60+3

Для того чтобы не перепутать буквы и цифры, использовались титла – горизонтальные черточки над числами, что мы видим на рисунке.

Для обозначения чисел больших, чем 900 использовались специальные значки, которые дорисовывались к букве. Так образовывались числа:

	Тысяча	1000
	Тьма	10 000
	Легион	100 000
	Леодр	1 000 000
	Ворон	10 000 000
	Колода	100 000 000

Рис. 1.5

Славянская нумерация просуществовала до конца XVII столетия, пока с реформами Петра I в Россию из Европы не пришла позиционная десятичная система счисления.

1.3. Основные понятия

Мы увидели, что есть множество способов представления чисел. В любом случае число изображается группой символов. Будем называть такие символы цифрами, символические изображения чисел – кодами, а правила получения кодов – системами счисления (кодирования).

Определение. Система счисления – это совокупность правил для обозначения и наименования чисел.

Все рассмотренные нами нечисловые системы счисления являются непозиционными.

Определение. Непозиционной называется такая система счисления, в которой количественный эквивалент каждой цифры не зависит от ее положения (места, позиции) в записи числа.

Итак, в непозиционных системах счисления позиция, которую цифра занимает в записи числа, роли не играет. Так, например, римская система счисления непозиционная. В числах XI и IX "вес" обоих цифр одинаков, несмотря на их месторасположение.

Система счисления, которой мы пользуемся в настоящее время, носит название десятичной, так как она основана на счете десятками. Исключительная роль десятка восходит к древнейшим временам и, несомненно, связана со счетом по пальцам на двух руках. Для записи любых чисел в ней используется десять всем хорошо известных цифр (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Поэтому ее и называют десятичной.

Определение. Основанием системы счисления называется количество знаков или символов, используемых для изображения числа в данной системе счисления.

Наименование системы счисления соответствует ее основанию (например, десятичной называется система счисления так потому, что ее основание равно 10, т.е. используется десять цифр).

Давайте рассмотрим число 55. Из двух написанных рядом цифр левая выражает число, в десять раз большее, чем правая. Таким образом, для написания цифр в десятичной системе имеет значение не только сама цифра, но и ее место, позиция. Именно поэтому такую систему счисления называют позиционной.

Определение. Система счисления называется позиционной, если значение цифры зависит от ее места (позиции) в записи числа.

Итак, в позиционных системах счислениях имеет значение позиция, которую цифра занимает в записи числа. Так, запись 23 означает, что это число можно составить из 3 единиц и 2 десятков. Если мы поменяем позиции цифр, то получим совсем другое число – 32. Это число содержит 3 десятка и 2 единицы. «Вес» двойки уменьшился в десять раз, а «вес» тройки в десять раз возрос.

Запишем первые несколько целых чисел в различных системах счисления:

В десятичной системе: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,…,19, 20, 21,…

в двоичной системе: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001;

в восьмеричной системе: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, ...

В шестнадцатеричной системе: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12,…, 1A, 1B, …

1.4. Системы счисления, используемые в компьютерах

Двоичная система счисления. Для записи чисел используются только две цифры – 0 и 1. Выбор двоичной системы объясняется тем, что электронные элементы, из которых строятся ЭВМ, могут находиться только в двух хорошо различимых состояниях. По существу эти элементы представляют собой выключатели. Как известно выключатель либо включен, либо выключен. Третьего не дано. Одно из состояний обозначается цифрой 1, другое – 0.

Благодаря таким особенностям двоичная система стала стандартом при построении ЭВМ.

Восьмеричная система счисления. Для записи чисел используется восемь чисел 0, 1, 2, 3, 4, 5, 6, 7.

Шестнадцатеричная система счисления. Для записи чисел в шестнадцатеричной системе необходимо располагать уже шестнадцатью символами, используемыми как цифры. В качестве первых десяти используются те же, что и в десятичной системе. Для обозначения остальных шести цифр (в десятичной они соответствуют числам 10, 11, 12, 13, 14, 15) используются буквы латинского алфавита – A, B, C, D, E, F.

Таблица 1

Таблица соответствия систем счисления

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
0	0	0	0
1	1	1	1
2	10	2	2

Окончание табл. 1

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
…	…	…	…
26	11010	32	1A

Система счисления называется **позиционной**, если одна и та же цифра имеет различное значение, определяющееся позицией цифры в последовательности цифр, изображающей число. Это значение меняется в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону.

Примером позиционной системы счисления является десятичная система, используемая в повседневной жизни.

Количество p различных цифр, употребляемых в позиционной системе определяет название системы счисления и называется **основанием** системы счисления – *"p"*.

Любое число N в позиционной системе счисления с основанием p может быть представлено в виде полинома от основания p:

$$N = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_0 + a_{-1} p^{-1} + a_{-2} p^{-2} + \ldots$$

здесь N – число, a_j – коэффициенты (цифры числа), p – основание системы счисления ($p>1$).

2. ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ АВТОМАТОВ

2.1. Основные понятия алгебры логики

Понятие цифрового автомата было введено как модель для описания функционирования устройств, предназначенных для переработки цифровой или дискретной информации.

Для формального описания цифровых автоматов применяется аппарат алгебры логики, созданной английским математиком Дж. Булем (1815–1864 г.г.). Поэтому алгебру логики называют алгеброй Буля или булевой алгеброй.

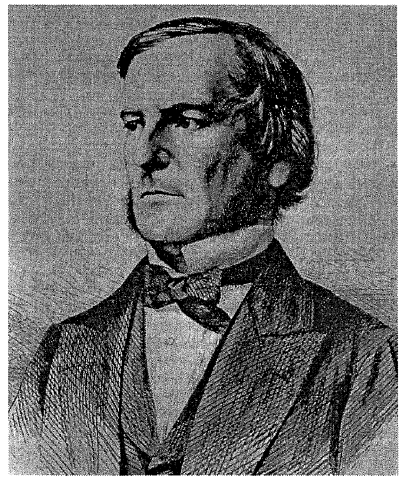


Рис. 2.1. Джордж Буль
[http://www.enezeus.com/blog/wp-content/uploads/2006/10/hacker1.jpg]

Джордж Буль (англ. George Boole; 2 ноября 1815, Линкольн – 8 декабря 1864, Баллинтемпл, графство Корк, Ирландия) – английский математик и логик. Профессор математики Королевского колледжа Корка (ныне Университетский колледж Корк) с 1849 г. Один из предтеч математической логики [ttp://ru.wikipedia.org]

В алгебре логики применительно к описанию цифровых автоматов, работающих в двоичном представлении кодов (или цифровой информации) основными понятиями являются *логическая (булева) переменная* и *логическая функция (функция алгебры логики – ФАЛ)*.

Логическая (булева) переменная – такая величина x, которая может принимать только два значения : $x = \{0,1\}$.

Логическая функция (функция алгебры логики – ФАЛ) – функция многих аргументов $f(x_{n-1}, x_{n-2}, \ldots, x_0)$, принимающая значения равные нулю или единице на наборах логических переменных $x_{n-1}, x_{n-2}, \ldots, x_0$.

В дальнейшем в формальных описаниях наборов переменных и логических функций сами наборы переменных интерпретируются как двоичные коды (числа). В двоичных кодах расположение логических переменных упорядочено в порядке уменьшения индекса слева направо и каждая логическая переменная имеет вес в зависимости от позиции в коде, увеличивающийся справа налево. Вес каждой i-той логической переменной, являющейся значением разряда двоичного числа равен 2^i ($i = 0, \ldots, n-1$).

Для n-разрядного кода общее количество уникальных наборов переменных

$$N = 2^n; \quad (1)$$

Максимальное числовое значение двоичного кода равно

$$A_{макс} = 2^n - 1; \quad (2)$$

Значения всех логических функций от одной переменной представлены в таблице 2.1.

Таблица 2.1

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

Функция $f_0(x)$ называется *константой нуля*, а функция $f_3(x)$ – *константой единицы*. Функция $f_1(x)$, повторяющая значения логической переменной, *тождественная функция* ($f_1(x) \equiv x$), а функция $f_2(x)$, принимающая значения, обратные значениям переменной x, *логическое отрицание* или *инверсия (НЕ)* ($f_2(x) = \overline{x}$).

Значения всех логических функций от двух переменных представлены в таблице 2. Для функции двух переменных, согласно ф.(1), существует четыре уникальных набора переменных. Функции отличаются друг от друга набором значений 0 и 1 в четырех разрядах кода значений функции. Общее количество функций на n-местном или n-разрядном наборе переменных равно

$$N_f = 2^{2^n}; \quad (3)$$

В таблице 2.2 приведены примеры аналитической записи некоторых булевых функций с использованием других булевых функций, то есть существует такой набор булевых функций, из которого можно сконструировать любую булеву функцию, равносильную заданной.

на всех возможных наборах переменных две функции равносильны друг другу, если они принимают одни и те же значения.

Аналитически это свойство описывается следующей формулой:

$$f_1(x_{n-1}, x_{n-2,}, \ldots, x_0) = f_2(x_{n-1}, x_{n-2,}, \ldots, x_0); \quad (4)$$

Обе функции в ф.(4) могут иметь разные формы аналитической записи, но практически наиболее выгодной будет самая простая форма записи.

Таблица 2.2

Функция	$x_2\,x_1$				Примечания
	00	01	10	11	
f_0	0	0	0	0	константа нуля
f_1	0	0	0	1	$x_2 \& x_1$ – конъюнкция
f_2	0	0	1	0	$x_2 \& \bar{x}_1$ – запрет x_1
f_3	0	0	1	1	$x_2\bar{x}_1 \vee x_2x_1 = x_2$
f_4	0	1	0	0	$\bar{x}_2x_1$ – запрет x_2
f_5	0	1	0	1	$\bar{x}_2x_1 \vee x_2x_1 = x_1$
f_6	0	1	1	0	$x_2 \oplus x_1$ – сложение по модулю 2
f_7	0	1	1	1	$x_2 \vee x_1$ – дизъюнкция
f_8	1	0	0	0	$x_2 \downarrow x_1$ – функция Пирса
f_9	*1*	*0*	*0*	*1*	*$x_2 \equiv x_1$ – равнозначность*
f_{10}	1	0	1	0	$\bar{x}_2\bar{x}_1 \vee x_2\bar{x}_1 = \bar{x}_1$
f_{11}	1	0	1	1	$x_1 \rightarrow x_2$ – импликация
f_{12}	1	1	0	0	$\bar{x}_2\bar{x}_1 \vee \bar{x}_2x_1 = \bar{x}_2$
f_{13}	1	1	0	1	$x_2 \rightarrow x_1$ – импликация
f_{14}	1	1	1	0	x_1/x_2 – функция Шеффера
f_{15}	1	1	1	1	константа единицы

Система булевых функций W называется функционально полной, если для любой булевой функции n-переменных $f(x_{n-1}, x_{n-2,}, \ldots, x_0)$ может быть построена равносильная ей функция комбинированием булевых переменных $x_{n-1}, x_{n-2,}, \ldots, x_0$ и функций системы W, взятых в любом конечном количестве экземпляров каждая. Такая система булевых функций (W) называется базисом.

Таким образом, базис – полная система функций алгебры логики (ФАЛ), с помощью которой любая ФАЛ может быть представлена суперпозицией исходных функций W.

Базисом является система функций И (конъюнкция), ИЛИ (дизъюнкция), НЕ (инверсия), свойства которых были впервые изучены Дж. Булем.

Базисами являются системы:

– И, НЕ;

– ИЛИ, НЕ;

– функция Шеффера И-НЕ;

– функция Пирса ИЛИ-НЕ.

Базис является минимальным, если удаление из него хотя бы одной функции превращает систему ФАЛ в неполную. Базис И, ИЛИ, НЕ – избыточный.

2.2. Базис И, ИЛИ, НЕ. Свойства элементарных функций алгебры логики

Пусть x – некоторая логическая переменная. Тогда:

1. $\overline{\overline{x}} = x$, что означает возможность исключения из логического выражения всех членов, имеющих двойное отрицание, заменив их исходной величиной;

2. $x \vee x = x;\ x \,\&\, x = x \cdot x = x$ – правила подобных преобразований, которые позволяют сокращать длину логических выражений;

3. $x \vee 0 = x$;

4. $x \vee 1 = 1$;

5. $x \cdot 0 = 0$;

6. $x \cdot 1 = x$;

7. $\overline{x}x = \mathbf{0}$;

8. $\overline{x} \vee x = \mathbf{1}$.

Дизъюнкция и конъюнкция обладают рядом свойств, аналогичных свойствам обычных арифметических операций сложения и умножения:

1) свойство ассоциативности (сочетательный закон):

$$x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3, \qquad x_1 (x_2\, x_3) = (x_1\, x_2)\, x_3;$$

2) свойство коммутативности (переместительный закон):

$$x_1 \vee x_2 = x_2 \vee x_1, \qquad x_1 x_2 = x_2\, x_1;$$

3) свойство дистрибутивности (распределительный закон):

для конъюнкции относительно дизъюнкции

$$x_1 \& (x_2 \vee x_3) = (x_1 \& x_2) \vee (x_1 \& x_3);$$

для дизъюнкции относительно конъюнкции

$$x_1 \vee x_2 \& x_3 = (x_1 \vee x_2) \& (x_1 \vee x_3).$$

Свойство дистрибутивности фактически определяет правила раскрытия скобок или взятия в скобки логических выражений.

Справедливость указанных свойств легко доказывается с помощью вышеизложенных аксиом.

Докажем, например, что

$$x_1 \vee x_2 \& x_3 = (x_1 \vee x_2) \& (x_1 \vee x_3).$$

В самом деле, $(x_1 \vee x_2)(x_1 \vee x_3) = x_1 x_1 \vee x_1 x_3 \vee x_1 x_2 \vee x_2 x_3 = x_1 \vee x_1 x_2 \vee x_1 x_3 \vee$

$$\vee x_2 x_3 = x_1 (1 \vee x_2 \vee x_3) \vee x_2 x_3.$$

Аналогично можно доказать и другие законы.

Таким же образом доказывается правильность соотношений, известных как *законы де Моргана:*

$$\overline{x_2 x_1} = \overline{x}_2 \vee \overline{x}_1, \tag{5}$$

$$\overline{x_2 \vee x_1} = \overline{x}_2 \,\&\, \overline{x}_1. \tag{6}$$

Из законов де Моргана следует, что:

$$x_2 \,\&\, x_1 = \overline{\overline{x}_2 \vee \overline{x}_1}. \tag{7}$$

$$\overline{\overline{x}_2 \,\&\, \overline{x}_1} = x_2 \vee x_1, \tag{8}$$

помощью которых появляется возможность выражать конъюнкцию через дизъюнкцию и отрицание или дизъюнкцию через конъюнкцию и отрицание.

Законы де Моргана и следствия из них справедливы для любого количества переменных:

$$\overline{x_n \vee x_{n-1} \vee \ldots \vee x_1} = \overline{x}_n \,\&\, \overline{x}_{n-1} \,\&\, \ldots \,\&\, \overline{x}_1, \tag{9}$$

$$\overline{x_n \,\&\, x_{n-1} \,\&\, \ldots \,\&\, x_1} = \overline{x}_n \vee \overline{x}_{n-1} \vee \ldots \vee \overline{x}_1. \tag{10}$$

Для логических функций устанавливаются соотношения, известные как *законы поглощения*:

$$x_1 \vee x_1 x_2 = x_1, \tag{11}$$

$$x_1 \& (x_1 \vee x_2) = x_1. \tag{12}$$

Очень важными в теории ФАЛ являются действия *полного склеивания* и *неполного склеивания*. Примеры выполнения этих действий с двумя конституентами 1 приведены ниже:

$$\mathbf{F}_i \vee \mathbf{F}_j = \mathbf{F}_k x_p \vee \mathbf{F}_k \overline{x}_p = \mathbf{F}_k - \text{полное склеивание;} \tag{13}$$

$$\mathbf{F}_i \vee \mathbf{F}_j = \mathbf{F}_k x_p \vee \mathbf{F}_k \overline{x}_p = \mathbf{F}_i \vee \mathbf{F}_j \vee \mathbf{F}_k - \text{неполное склеивание.} \tag{14}$$

Более важным для практики является неполное склеивание, так как для сложных ФАЛ исходные конституенты 1 F_i и F_j после получения результата склеивания друг с другом F_k сохраняются для сопоставления с другими минтермами в записи заданной ФАЛ и нового склеивания по одной из переменных, входящих в сопоставимые минтермы с отрицанием и без отрицания (отличающихся по одной переменной).

Рассмотренные основные соотношения позволяют описать равносильные булевы функции различными способами, вследствие чего открываются возможности выбора самых простых форм описания ФАЛ. Самые простые по форме ФАЛ реализуются на элементной базе по принципиальным схемам, имеющим самую низкую стоимость.

2.3. Способы описания булевых функций

Известно несколько способов описания булевых функций и выбор конкретного для практического применения определяется в соответствии с условиями решаемой задачи. Теория ФАЛ позволяет получить немедленный практический результат в виде принципиальных схем, ориентированных на заданную элементную базу (серию интегральных схем и типы логических элементов) и именно это определяет условия применения того или иного описания ФАЛ и последующие действия по их упрощению с целью получения
экономичной реализации в виде конструкции для установки в радиоэлектронную аппаратуру.

2.3.1. Табличное описание булевых функций

Вследствие конечности множества наборов заданного количества логических переменных, простейшим и самым естественным способом описания ФАЛ является табличный. Пример описания трёх ФАЛ четырёх переменных представлен в таблице 2.3. Все наборы переменных в таблице упорядочены по возрастанию числового двоичного кода этих наборов. Коды наборов могут быть представлены в восьмеричной, шестнадцатеричной или даже десятеричной (что нежелательно) системе счисления.

Одна ФАЛ описывается одной таблицей, но для функций одинакового количества переменных можно использовать в таблице общее поле наборов переменных и интерпретировать такую таблицу как описание системы булевых функций.

Для примера выбраны функции (система функций):
– *конституента* 1 (F_{12});
– *конституента* 0 (Φ_{14});
– функция общего вида (F).

В таблице 2.3 *конституенты* 1 обозначаются F_j , где j – восьмеричный код набора переменных, на котором конституента равна 1, *конституенты* 0 обозначаются Φ_p , где p – восьмеричный код набора переменных, на котором конституента равна 0. Общее количество конституент 1 и общее количество конституент 0 равны по 2^n , где n – количество булевых переменных в наборе. Табличное описание ФАЛ и систем ФАЛ является простым и наглядным, однако весьма громоздким для практического использования.

Таблица 2.3

x_3	x_2	x_1	x_0	F_{12}	Φ_{14}	F
0	0	0	0	0	1	0
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	1	1
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	0	1	1
1	1	1	0	0	1	0
1	1	1	1	0	1	0

2.3.2. Аналитическое описание булевых функций

На примерах описания ФАЛ, приведенных в таблице 2.3, видно, что конституента 1 может быть описана в виде элементарной конъюнкции переменных:

$$\tilde{x}_n \,\&\, \tilde{x}_{n-1} \,\&\, \ldots \,\&\, \tilde{x}_0 = \tilde{x}_n \tilde{x}_{n-1} \ldots \tilde{x}_0; \qquad (15)$$

где: $\tilde{x}_j = \bar{x}_j$,если соответствующий разряд кода равен 0; $\tilde{x}_j = x_j$,если соответствующий разряд кода равен 1.

Конституента 0 может быть описана в виде элементарной дизъюнкции переменных:

$$\tilde{x}_n \vee \tilde{x}_{n-1} \vee ... \vee \tilde{x}_0; \qquad (16)$$

где: $\tilde{x}_j = \bar{x}_j$,если соответствующий разряд кода равен 1; $\tilde{x}_j = x_j$,если соответствующий разряд кода равен 0.

Формулы (15) и (16) представляют аналитическую форму записи конституент, как функций алгебры логики.

ФАЛ общего вида может быть аналитически записана:

– в *совершенной дизъюнктивной нормальной форме (СДНФ)*

$$\mathbf{F = F_p \vee F_k \vee ... \vee F_z}; \qquad (17)$$

где F_p , F_k ,..., F_z – конституенты 1. В контексте аналитической записи ФАЛ в СДНФ все конъюнктивные термы имеют максимальный ранг и называются *минтермами ранга n*.

– в *совершенной конъюнктивной нормальной форме (СКНФ)*

$$\Phi = \Phi_d \,\&\, \Phi_t \,\&\, ... \,\Phi_y; \qquad (18)$$

где Φ_d , Φ_t ,..., Φ_y – конституенты 0. В контексте аналитической записи ФАЛ в СКНФ все дизъюнктивные термы имеют максимальный ранг и называются *макстермами ранга n*.

ФАЛ общего вида, приведенная в таблице 2.3, записывается в СДНФ как:

$$\mathbf{F} = \bar{x}_3\bar{x}_2x_1\bar{x}_0 \vee \bar{x}_3x_2\bar{x}_1\bar{x}_0 \vee \bar{x}_3x_2\bar{x}_1x_0 \vee \bar{x}_3x_2x_1x_0 \vee x_3\bar{x}_2x_1\bar{x}_0 \vee x_3x_2\bar{x}_1x_0; \qquad (19)$$

В СКНФ эта же ФАЛ записывается как:

$$\begin{aligned}\mathbf{F} = {} & (x_3 \vee x_2 \vee x_1 \vee x_0)\,\&\,(x_3 \vee x_2 \vee x_1 \vee \bar{x}_0)\,\&\,(x_3 \vee x_2 \vee \bar{x}_1 \vee \bar{x}_0)\,\& \\ & \&\,(x_3 \vee \bar{x}_2 \vee \bar{x}_1 \vee x_0)\,\&\,(\bar{x}_3 \vee x_2 \vee x_1 \vee x_0)\,\&\,(\bar{x}_3 \vee x_2 \vee x_1 \vee \bar{x}_0)\,\& \\ & \&\,(\bar{x}_3 \vee x_2 \vee \bar{x}_1 \vee \bar{x}_0)\,\&\,(\bar{x}_3 \vee \bar{x}_2 \vee x_1 \vee x_0)\,\&\,(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1 \vee x_0)\,\& \\ & \&\,(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1 \vee \bar{x}_0); \end{aligned} \qquad (20)$$

Для практического применения обычно используется СДНФ и мы в дальнейшем будем пользоваться только этой формой представления ФАЛ.

2.3.3. Числовая форма представления булевых функций

Используя представление наборов переменных как числовых двоичных или восьмеричных кодов, запишем и минтермы функции F (таблица 2.3) как числа:

$$\mathbf{F = 0010 \vee 0100 \vee 0101 \vee 0111 \vee 1010 \vee 1101 = 2 \vee 4 \vee 5 \vee 7 \vee 12 \vee 15}; \qquad (21)$$

В числовой СДНФ форме ФАЛ имеет самую компактную запись и позволяет при необходимости перейти к любому другому описанию этой функции.

2.3.4. Графическая форма представления булевых функций

График функция F, приведенной в таблице 2.3, представлен на рис.2.2.

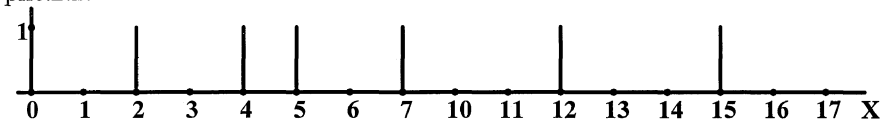


Рис.2.2. График булевой функции

График конституенты 1 F_{12} (таблица 3) представлен на рис.2.3.

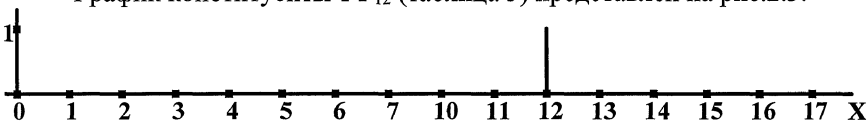


Рис.2.3. График конституенты 1 F_{12}

График конституенты 0 Φ_{14} (таблица 3) представлен на рис.2.4.

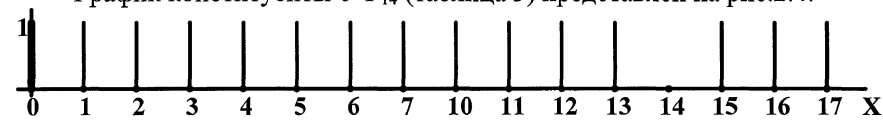


Рис.2.4. График конституенты 0 Φ_{14}

Из графического представления конституент непосредственно следуют описания ФАЛ в СДНФ и СКНФ (фф(19) и (20)).

2.3.5. Геометрическое представление булевых функций

В геометрическом представлении ФАЛ значения входных переменных *n*-местного набора интерпретируются как координаты в *n*-мерной декартовой системе координат. Координатная система, в которой располагается геометрическая модель ФАЛ, является *n*-мерным кубом с единичными рёбрами. Кодам наборов входных переменных соответствуют вершины *n*-мерного куба. Значения кодов входных наборов переменных, на которых заданная ФАЛ равна единице (при использовании СДНФ для представления ФАЛ) помечаются нечисловыми символами и такой *n*-мерный куб с

помеченными некоторыми вершинами и является геометрической моделью заданной ФАЛ. На рис.2.5 представлен пример геометрической модели функции двух переменных

$$\mathbf{F} = \mathbf{1} \vee \mathbf{3} = \bar{x}_1 x_0 \vee x_1 x_0; \qquad (22)$$

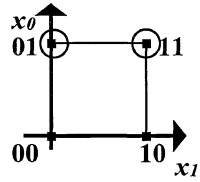
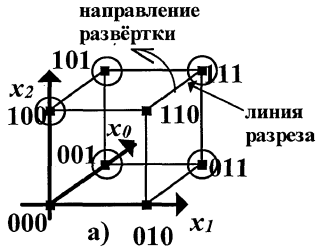


Рис.2.5. Геометрическая модель функции двух переменных

На рис.2.6 представлена геометрическая модель ФАЛ трёх переменных

$$\mathbf{F} = \mathbf{1} \vee \mathbf{3} \vee \mathbf{4} \vee \mathbf{5} \vee \mathbf{7} = \bar{x}_2 \bar{x}_1 x_0 \vee \bar{x}_2 x_1 x_0 \vee x_2 \bar{x}_1 \bar{x}_0 \vee \bar{x}_2 x_1 \bar{x}_0 \vee x_2 x_1 x_0; \qquad (23)$$



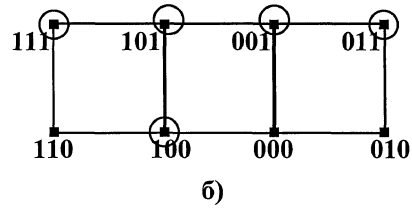


Рис.2.6. Геометрическая модель функции трёх переменных

На рис.2.6а) изображена модель функции трёх переменных в трёхмерном пространстве, а на рис.2.6б) – в двумерном пространстве на развёртке единичного куба на плоскость. Отметим чрезвычайно важное для практики свойство геометрической модели ФАЛ, на которой видно, что *наборы переменных, отличающиеся по одной переменной принадлежат одному ребру единичного куба*. Такие наборы переменных называются соседними. На развёртке соседними являются наборы, расположенные на противоположных краях.

Сопоставляя геометрические модели ФАЛ двух переменных и ФАЛ трёх переменных, сформулируем правило построения развёрток на плоскость *n*-мерных единичных кубов (карт Карно) по имеющимся развёрткам *n–1*-мерных единичных кубов (карт Карно меньшей размерности).

Если строится карта Карно для нечётного количества переменных в наборе, то на расстоянии единицы слева от исходной карты для чётного количества переменных изображается повёрнутая на 180^0 вокруг оси, проходящей между исходной и новой картами, новая карта той же раз-

мерности. После этого в старшем разряде двоичных кодов наборов исходной карты добавляются незначащие нули, а в старшем разряде новой карты добавляются единицы. Эти две карты объединяются в одну большей размерности.

Если строится карта Карно для чётного количества переменных в наборе, то на расстоянии единицы снизу от исходной карты для нечётного количества переменных изображается повёрнутая на 180^0 вокруг оси, проходящей между исходной и новой картами, новая карта той же размерности. После этого в старшем разряде двоичных кодов наборов исходной карты добавляются незначащие нули, а в старшем разряде новой карты добавляются единицы. Эти две карты объединяются в одну большей размерности.

В картах наборы переменных, на которых функция принимает единичные значения, помечаются нечисловыми символами. *Карта с нанесёнными на ней значениями ФАЛ называется диаграммой.*

Карты, на которых коды наборов изображаются в восьмеричной системе счисления, называются картами Вейча.

На рис.2.7 приведены примеры построения карт Карно и Вейча для четырёх– и пятиместных наборов переменных. На картах Вейча на рис.2.7б) и рис.2.7в) дана аналитическая разметка, указывающая области нулевых значений переменных в наборах. На картах в соседних клетках размещены коды наборов переменных, являющихся соседними, т.е. отличающимися по какой-то одной переменной знаком инверсии. На рис.2.8 приведена геометрическая модель *частично определённой ФАЛ* пяти переменных в виде диаграммы Вейча. Те восьмеричные коды наборов переменных, на которых ФАЛ имеет единичное значение, отмечены кружком. Те восьмеричные коды наборов переменных, на которых ФАЛ не определена отмечены круглыми скобками. В большинстве технических применений функции алгебры логики являются частично определёнными, так как не существует технической возможности появления некоторых наборов переменных, называемых в этом случае *запрещёнными*. На рис.2.8 отмечены поля из соседних пар наборов переменных, на которых ФАЛ не определена или имеет единичные значения.

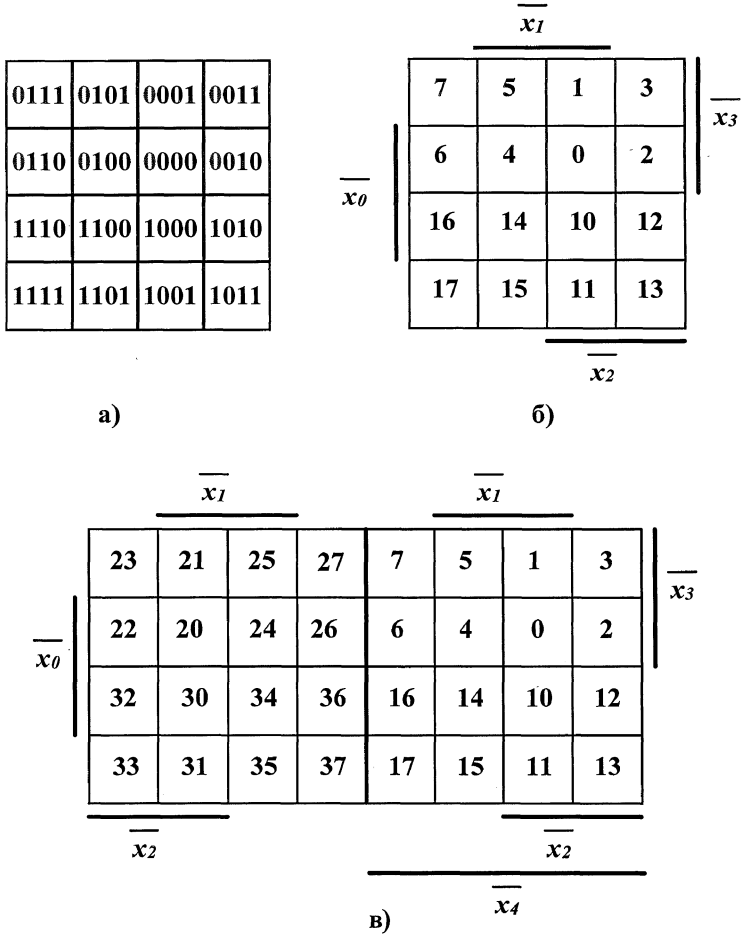
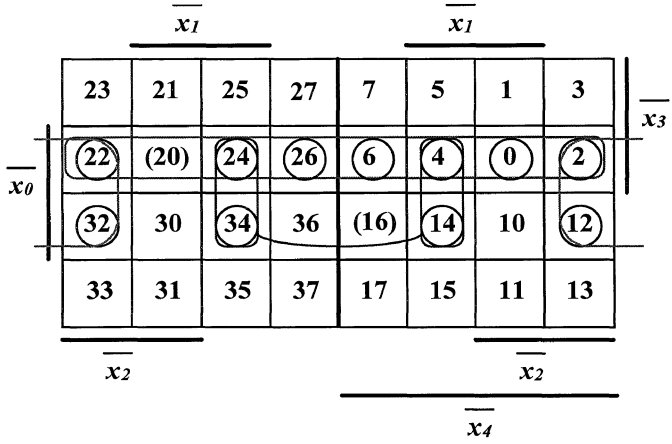


Рис.2.7. Пример построения карт Карно и Вейча для четырёх- и пятиместных наборов переменных



F= 0∨2∨4∨6∨12∨14∨(16)∨(20)∨22∨24∨26∨32∨34 ;
Коды наборов восьмеричные. На (*) наборах функция не определена.

Рис.2.8. Диаграмма Вейча ФАЛ пяти переменных

Соседними являются и наборы, лежащие на границах карты, например: 3–7, 3–13, 3–23, 7–17, 1–11, 5–15, 27–37, 25–35, 21–31, 23–33, 2–22, 12–32.

Для карт с количеством переменных больше четырёх соседними являются наборы, расположенные симметрично относительно соединительной линии (на рис.2.8 указанной стрелкой) карт меньшего на единицу количества переменных. Например: 1–21, 4–24, 0–20, 14–34, 10–30, 11–31, 15–35.

При таких определениях соседних наборов следует заметить, что *развёрткой четырёхмерного куба в трёхмерное пространство является однослойный тор, а развёрткой пятимерного куба в трёхмерное пространство является двухслойный тор.*

С увеличением количества переменных в наборах на одну переменную количество всевозможных наборов переменных удваивается, а, следовательно, размерность карт Карно или Вейча удваивается, так как количество вершин многомерного куба удваивается.

2.4. Минимизация функций алгебры логики

2.4.1. Минимизация с помощью минимизирующих карт

Как было отмечено выше, одним из способов представления ФАЛ от небольшого числа переменных (обычно не больше 5) являются диаграммы Карно или Вейча, которые строятся на развёртках многомерных кубов на

плоскость. При этом вершины куба представляются клетками карты, координаты которых совпадают с координатами соответствующих вершин куба. Карта заполняется путём пометки кодов вершин, соответствующих наборам, на которых ФАЛ равна единице. Другими символами помечаются коды наборов, на которых ФАЛ не определена. Таким образом, диаграмма на карте Карно или Вейча соответствует представлению ФАЛ в СДНФ.

Пример диаграммы Вейча функции пяти переменных представлен на рис.9. По этим диаграммам находятся соседние наборы переменных, на которых ФАЛ равна 1 (или не определена). Соседние на карте наборы переменных отличаются по одной переменной и могут быть по ней склеены, с исключением из наборов этой переменной. Последовательное применение неполного склеивания к наборам, образующим сплошные поля из двух, четырёх, восьми, шестнадцати и. т. д. наборов, на которых ФАЛ равна единице, позволяет исключить одну, две, три, четыре и. т. д. переменных из этих наборов, то есть минимизировать эту ФАЛ. Эти поля должны быть симметричными относительно соединительной линии карт меньшей размерности. На рис.2.9. такие поля описываются как пересечение полей постоянных значений переменных в наборах. Поле из наборов переменных 2–0–4–6–26–24–20–22 описывается пересечением полей постоянных значений переменных $\bar{x}_3\bar{x}_0$, поле из наборов переменных 2–12–22–32 описывается пересечением полей постоянных значений переменных $\bar{x}_2 x_1 \bar{x}_0$, поле из наборов переменных 4–14–24–34 описывается пересечением полей постоянных значений переменных $x_2\bar{x}_1\bar{x}_0$. Таким образом, минимизированная ФАЛ, приведенная на рис.2.9, записывается в дизъюнктивной нормальной форме (ДНФ) как

$$F=\bar{x}_3\bar{x}_0 \vee \bar{x}_2 x_1 \bar{x}_0 \vee x_2\bar{x}_1\bar{x}_0. \qquad (24)$$

Отсюда видно, что минимизированная ФАЛ имеет гораздо более простую форму записи, чем исходная функция.

Из рис.2.8 видно, что на 20_8 (индекс 8 – указание на основание системы счисления) наборе следует придать единичное значение минимизируемой функции, а на 16_8 наборе доопределить ФАЛ как имеющую нулевое значение.

2.4.2. Минимизация функций алгебры логики по методу Квайна

При минимизации по методу Квайна в базисе И, ИЛИ, НЕ исходная ФАЛ задаётся в СДНФ. Целью минимизации является нахождение всех *первичных импликант* и выбор некоторых из них для минимальной записи функции.

Импликанта функции – некоторая логическая функция, обращаемая в нуль при наборе переменных, на котором сама функция также равна нулю.

Поэтому любой конъюнктивный терм, входящий в состав СДНФ, или группа термов, соединённых знаками дизъюнкции являются импликантами исходной ФАЛ. Импликанты имеют единичные значения только на подмножестве наборов из множества наборов, на которых исходная ФАЛ равна единице.

Первичная импликанта функции – импликанта типа элементарной конъюнкции некоторых переменных, никакая часть которой уже не является импликантой.

Задача минимизации по методу Квайна решается путём попарного сравнения всех импликант, входящих в ФАЛ, с целью выявления возможности их неполного склеивания по какой-то переменной на промежуточных этапах

$$\mathbf{F}_i \vee \mathbf{F}_j = \mathbf{F}_k x_p \vee \mathbf{F}_k \bar{x}_p = \mathbf{F}_i \vee \mathbf{F}_j \vee \mathbf{F}_k ; \qquad (25)$$

Из первичных импликант выбираются такие, которые представляют исходную ФАЛ минимальным образом.

При склеивании снижается ранг термов. Склеивание проводится до тех пор, пока не останется ни одного терма, допускающего склеивание с каким-либо другим термом. Термы, подвергшиеся склеиванию, отмечаются. Неотмеченные термы представляют собой первичные импликанты. После получения множества всех первичных импликант исследуется возможность нахождения простейшей записи ФАЛ. Для этого составляется таблица, в первой строке которой записаны минтермы исходной ФАЛ, а в первом столбце записаны все найденные первичные импликанты. Клетки этой таблицы помечаются в том случае, если первичная импликанта входит в состав какого-либо минтерма исходной ФАЛ. После этого задача упрощения сводится к тому, чтобы найти такое минимальное количество первичных импликант, которые покрывают все столбцы минтермов исходной ФАЛ.

Минимизация по методу Квайна алгоритмизирована и выполняется в несколько этапов. Рассмотрим применение этого метода минимизации на конкретном примере функции алгебры логики, приведенной в цифровой форме в восьмеричной системе счисления

$$\mathrm{F}= (2)\vee 3\vee 4\vee 5\vee 7\vee 11\vee 13\vee 14\vee 15\vee (17)=$$
$$= (\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0) \vee \bar{x}_3\bar{x}_2 x_1 x_0 \vee \bar{x}_3 x_2 \bar{x}_1\bar{x}_0 \vee \bar{x}_3 x_2 \bar{x}_1 x_0 \vee \bar{x}_3 x_2 x_1 x_0 \vee x_3\bar{x}_2\bar{x}_1\bar{x}_0 \vee$$
$$\vee x_3\bar{x}_2 x_1 x_0 \vee x_3 x_2\bar{x}_1\bar{x}_0 \vee x_3 x_2\bar{x}_1 x_0 \vee (x_3 x_2 x_1 x_0); \qquad (26)$$

Э т а п 1. <u>*Нахождение первичных импликант*</u>. Прежде всего составляется таблица (таблица 4) и находятся импликанты четвёртого и третьего

рангов, то есть понижается ранг минтермов, входящих в СДНФ. Затем составляется другая таблица (таблица 5), которая включает все импликанты третьего ранга. По этой таблице находятся импликанты третьего и второго рангов. В общем случае составление таблиц и нахождение термов низшего ранга продолжается до тех пор, пока нельзя будет выполнить склеивание по какой-либо переменной для термов низшего ранга.

Таблица 2.4

Термы	$(\bar{x}_3\bar{x}_2x_1\bar{x}_0)$	$\bar{x}_3\bar{x}_2x_1$	$\bar{x}_3x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_1$	$\bar{x}_3x_2x_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2x_1\bar{x}_0$	$x_2\bar{x}_1\bar{x}_0$	$x_2\bar{x}_1x_0$	$(x_3x_2x_1x_0)$
$(\bar{x}_3\bar{x}_2x_1\bar{x}_0)$	X	$\bar{x}_3\bar{x}_2x_1$								
$\bar{x}_3\bar{x}_2x_1x_0$	X	X			$\bar{x}_3x_1x_0$		$\bar{x}_2x_1x_0$			
$\bar{x}_3x_2\bar{x}_1\bar{x}_0$	X	X	X	$\bar{x}_3x_2\bar{x}_1$				$x_2\bar{x}_1\bar{x}_0$		
$\bar{x}_3x_2\bar{x}_1x_0$	X	X	X	X	$\bar{x}_3x_2x_0$				$x_2\bar{x}_1x_0$	
$\bar{x}_3x_2x_1x_0$	X	X	X	X	X					$x_2x_1x_0$
$x_3\bar{x}_2\bar{x}_1x_0$	X	X	X	X	X	X		$x_3\bar{x}_2x_0$	$x_3\bar{x}_1x_0$	
$x_3\bar{x}_2x_1x_0$	X	X	X	X	X	X	X			$x_3x_1x_0$
$x_3x_2\bar{x}_1\bar{x}_0$	X	X	X	X	X	X	X	X	$x_3x_2\bar{x}_1$	
$x_3x_2\bar{x}_1x_0$	X	X	X	X	X	X	X	X	X	$x_3x_2x_0$
$(x_3x_2x_1x_0)$	X	X	X	X	X	X	X	X	X	X

Примечания:

1. X – ячейка таблицы не заполняется;
2. На всех (*) наборах функция доопределена как имеющая значение 1.

Для уменьшения трудоёмкости заполнения таблицы, заполняются только клетки, лежащие по одну сторону от диагонали таблицы, поскольку результат операции склеивания не зависит от порядка следования склеиваемых термов. По этой таблице определяется правильность доопределения ФАЛ. Если термы неопределённости ФАЛ подверглись склеиванию только друг с другом или вообще не склеились, то на этих наборах следует доопределить ФАЛ, как имеющую нулевые значения. В нашем примере по таблице видно, что ФАЛ требуется доопределить как имеющую единичные значения на наборах 2 и 17. Если какие-то термы не подверглись склеиванию, их следует включить в список первичных импликант и исключить из дальнейшего рассмотрения. В нашем примере не склеивающихся минтермов в ФАЛ нет. Для дальнейшего сопоставления все термы, имеющие ранг

на единицу меньший ранга исходных минтермов, из таблицы 4 заносятся в табл. 2.5.

Заголовочные строки и столбец таблицы заполняются термами в одинаковом порядке. Тогда заполнять можно только одну половину таблицы, лежащую по одну сторону от диагонали (пусть будет верхняя часть таблицы).

Таблица 2.5

Термы	$\bar{x}_3\bar{x}_2x_1$	$\bar{x}_3x_1\bar{x}_0$	$\bar{x}_2x_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_1$	$x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2x_0$	$x_2\bar{x}_1x_0$	$x_2x_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_0$	$x_3\bar{x}_1x_0$	$x_3x_1\bar{x}_0$	$x_3\bar{x}_2x_1$	$x_3x_2\bar{x}_0$
$\bar{x}_3\bar{x}_2x_1$	X												
$\bar{x}_3x_1\bar{x}_0$		X									x_1x_0		
$\bar{x}_2x_1\bar{x}_0$			X					x_1x_0					
$\bar{x}_3x_2\bar{x}_1$				X								$x_2\bar{x}_1$	
$x_2\bar{x}_1\bar{x}_0$					X		$x_2\bar{x}_1$						
$\bar{x}_3x_2x_0$						X							x_2x_0
$x_2\bar{x}_1x_0$							X	x_2x_0					
$x_2x_1\bar{x}_0$								X					
$x_3\bar{x}_2x_0$									X				x_3x_0
$x_3\bar{x}_1x_0$										X	x_3x_0		
$x_3x_1\bar{x}_0$											X		
$x_3x_2\bar{x}_1$												X	
$x_3x_2x_0$													X

Из табл. 2.5 видно, что дальнейшее склеивание импликант второго ранга невозможно и, следовательно, в клетках табл. 2.5 находятся первичные импликанты. К первичным относится и терм в первой строке (он же в первом столбце), поскольку он не подвергся склеиванию.

Э т а п 2. *Расстановка меток*. Составляется таблица, число строк которой равно числу полученных первичных импликант, а число столбцов равно количеству минтермов СДНФ. Если в некоторый минтерм СДНФ входит какая-либо из первичных импликант, то на пересечении строки и соответствующего столбца ставится нечисловая метка, например, ♥ (таблица 2.6).

Таблица 2.6

Термы	$(\bar{x}_3 x_2 \bar{x}_1 \bar{x}_0)$	$\bar{x}_3 \bar{x}_2 x_1 \bar{x}_0$	$\bar{x}_3 x_2 \bar{x}_1 \bar{x}_0$	$\bar{x}_3 x_2 x_1 \bar{x}_0$	$\bar{x}_3 x_2 x_1 x_0$	$x_3 \bar{x}_2 \bar{x}_1 \bar{x}_0$	$x_3 \bar{x}_2 x_1 x_0$	$x_3 x_2 \bar{x}_1 \bar{x}_0$	$x_3 x_2 x_1 \bar{x}_0$	$(x_3 x_2 x_1 x_0)$
$x_1 x_0$		♥[3]			♥		♥			♥
$x_2 \bar{x}_1$			♥[c]	♥				♥[c]	♥	
$x_2 x_0$				♥	♥				♥	♥
$x_3 x_0$						♥[c]	♥		♥	♥
$\bar{x}_3 \bar{x}_2 x_1$	♥[c]	♥								

Э т а п 3. *Нахождение существенных импликант*. Если в какомлибо из столбцов таблицы 6 имеется только одна метка, то первичная импликанта в соответствующей строке является существенной, так как без неё не будут получены заданные единичные значения минимизированной ФАЛ на всём множестве заданных минтермов СДНФ ФАЛ. В табл. 2.6 существенные импликанты отмечены знаком ♥[c] синего цвета. По таблице 6 видно, что пятая первичная импликанта $\bar{x}_3 \bar{x}_2 x_1$ является существенной только для неопределённого минтерма $\bar{x}_3 \bar{x}_2 x_1 \bar{x}_0$. Если придать СДНФ исходной ФАЛ нулевое значение на наборе переменных $\bar{x}_3 \bar{x}_2 x_1 \bar{x}_0$, то пятую первичную импликанту нужно исключить, тогда первая импликанта (она отмечена знаком ♥[3] зелёного цвета) станет существенной. На наборе $x_3 x_2 x_1 x_0$ исходной ФАЛ следует придать единичное значение. Столбцы, соответствующие существенным импликантам из таблицы вычёркиваются, а сами существенные импликанты в дальнейшем рассмотрении не используются.

Э т а п 4. *Вычёркивание лишних столбцов*. После третьего этапа в результате вычёркивания столбцов, соответствующих существенным импликантам, строится новая табл.2.7. В эту таблицу включаются оставшиеся столбцы и соответствующие им первичные импликанты. Если в этой таблице есть столбцы, в которых имеются метки в одинаковых строках, то эти столбцы вычёркиваются , кроме одного. Минтерм оставшегося столбца будут покрывать отброшенные минтермы. В нашем примере ФАЛ такого случая нет.

Таблица 2.7

Термы	$\bar{x}_3x_2\bar{x}_1x_0$	$\bar{x}_3x_2x_1\bar{x}_0$	$x_3\bar{x}_2\bar{x}_1x_0$	$x_3\bar{x}_2x_1\bar{x}_0$	$(x_3\bar{x}_2x_1x_0)$
$\blacktriangledown^c\ x_1x_0$		♥	♥		♥
$\blacktriangledown^c x_2\bar{x}_1$	♥			♥	
x_2x_0	♥	♥		♥	♥
$\blacktriangledown^c x_3x_0$			♥	♥	♥

Существенные первичные импликанты в ячейках табл. 2.7 помечены знаком $\blacktriangledown^c$ синего цвета.

Э т а п 5. *Вычёркивание лишних первичных импликант*. Если после вычёркивания лишних столбцов на этапе 4 в табл. 2.7 появляются строки, в которых нет ни одной метки, то первичные импликанты, соответствующие этим строкам, исключаются из дальнейшего рассмотрения, так как они не покрывают оставшиеся в рассмотрении минтермы. После выполнения этапов 4 и 5 рассматриваемая табл. 2.6 покрытия минтермов первичными импликантами несколько упрощается.

Э т а п 6. *Выбор минимального покрытия*. Выбирается в упрощенной таблице такая совокупность первичных импликант, которая включает метки во всех столбцах по крайней мере по одной метке в каждом столбце. При нескольких возможных вариантах такого выбора отдаётся предпочтение варианту покрытия с минимальным суммарным числом букв в импликантах, образующих покрытие.

Таким образом, минимальная дизъюнктивная нормальная форма (МДНФ) заданной функции складывается из существенных импликант (этап 3) и первичных импликант, покрывающих оставшиеся минтермы (этап 6). Минимизированная ФАЛ нашего примера запишется как:

$$F = x_1x_0 \vee x_2\bar{x}_1 \vee x_3x_0; \qquad (27)$$

Сравнивая фф.(26) и (27), видим, что аналитическая запись минимизированной ФАЛ значительно проще, чем исходная СДНФ ФАЛ.

2.4.3. Минимизация функций алгебры логики по методу Квайна-Мак-Класки

Недостаток метода Квайна – необходимость исчерпывающего попарного сравнения или сопоставления всех минтермов на этапе нахождения первичных импликант. С ростом числа минтермов увеличивается количество попарных сравнений.

Числовое представление ФАЛ позволяет упростить самый трудоёмкий этап 1. Все минтермы СДНФ ФАЛ записываются в виде их двоичных кодов, а все коды разбиваются по числу единиц на непересекающиеся группы.

Минтермы, подлежащие склеиванию, различаются только по одной переменной, а их коды – только в одном разряде. По этой причине сравнению подлежат только двоичные коды минтермов соседних групп. Группы кодов, различающиеся в двух или большем количестве разрядов просто не имеет смысла сравнивать. Рассмотрим применение метода Квайна-Мак-Класки для минимизации частично определённой функции пяти переменных:

$$F=(2)\vee 3\vee 4\vee 5\vee 7\vee 11\vee 13\vee 14\vee 15\vee(17)\vee(20)\vee(21)\vee(30)=$$
$$=(00010)\vee 00011\vee 00100\vee 00101\vee 00111\vee 01001\vee 01011\vee 01100\vee 01101\vee(01111)\vee$$
$$\vee(10000)\vee(10001)\vee(11000); \qquad (28)$$

Группа 0: пустая
Группа 1: (00010) 00100 (10000)
Группа 2: 00011 00101 01100 01001 (10001) (11000)
Группа 3: 00111 01011 01101
Группа 4: (01111)

Э т а п 1. Нахождение первичных импликант.

Сравнение минтермов соседних групп проведём с использованием таблиц. Двоичный код переменной, по которой склеиваются импликанты, после склеивания заменяется в ячейках таблиц нечисловым символом ♣.

Таблица 2.8

Термы	**00011**	**00101**	**01100**	**01001**	**(10001)**	**(11000)**
(00010)	**0001♣**					
00100		**0010♣**	**0♣100**			
(10000)					**1000♣**	**1♣000**

Таблица 2.9

Термы	00011	00101	01100	01001	(10001)	(11000)
00111	00♣11	001♣1				
01011	0♣011			010♣1		
01101		0♣101	0110♣	01♣01		

Таблица 2.10

Термы	00111	01011	01101
(01111)	0♣111	01♣11	011♣1

В заголовочных строках таблиц 2.8, 2.9 и 2.10 нет двоичных кодов минтермов, которые не подверглись склеиванию. Значит первичных импликант среди минтермов минимизируемой ФАЛ нет. Минтермы с кодами (20), (21) и (30), на которых ФАЛ не определена, склеились только между собой. ФАЛ на этих наборах переменных должна быть доопределена как имеющая нулевые значения и, следовательно, из дальнейшего рассмотрения должны быть исключены и эти минтермы и импликанты, полученные в результате их склеивания в таблицах 2.8, 2.9 и 2.10.

В ячейках таблицы 2.8 находятся импликанты группы 1, в ячейках таблицы 2 находятся импликанты группы 2, в ячейках таблицы 3 находятся импликанты группы 3.

В таблицах 2.11 и 2.12 произведено сопоставление импликант соседних групп.

Таблица 2.11

Импликанты	00♣11	001♣1	0♣011	010♣1	0♣101	0110♣	01♣01
0001♣							
0010♣						0♣10♣	
0♣100					0♣10♣		

Таблица 2.12

Импликанты	00♣11	001♣1	0♣011	010♣1	0♣101	0110♣	01♣01
0♣111			0♣♣11		0♣1♣1		
01♣11	0♣♣11						01♣♣1
011♣1		0♣1♣1		01♣♣1			

Первичные импликанты: 0001♣ 0♣10♣ 0♣♣11 0♣1♣1 01♣♣1

Э т а п 2. Расстановка меток.

Таблица 2.13

Термы	(00010)	00011	00100	00101	00111	01001	01011	01100	01101	(01111)
0001♣	♥^{c}	♥								
0♣10♣			♥^{c}	♥				♥^{c}	♥	
0♣♣11		♥^{3}			♥		♥			♥
0♣1♣1				♥	♥				♥	♥
01♣♣1						♥^{c}	♥		♥	♥

Э т а п 3. Нахождение существенных импликант.

Существенные импликанты в таблице 2.13 отмечены знаком ♥^{c} синего цвета. Первая существенная импликанта 0001♣ представляет набор (00010), на котором СДНФ ФАЛ не определена. Очевидно, что следует доопределить ФАЛ как имеющую нулевое значение на этом наборе переменных. Тогда первая существенная первичная импликанта 0001♣ исключается из дальнейшего рассмотрения, а первичная импликанта 0♣♣11 помеченная знаком ♥^{3} зелёного цвета, становится существенной для набора 00011.

Э т а п ы 4 и 5. Для исходной СДНФ ФАЛ, минимизируемой по методу Квайна-Мак-Класки, эти этапы выполняются точно так же, как и при минимизации СДНФ ФАЛ по методу Квайна и, поскольку получена всего одна несущественная импликанта 0♣1♣1, в описание минимизированной ФАЛ эта первичная импликанта не включается.

Э т а п 6. Выбирается минимальное покрытие минтермов СДНФ ФАЛ существенными первичными импликантами и записывается единственно возможный вариант минимальной ФАЛ:

$$F = 0\clubsuit10\clubsuit \vee 0\clubsuit\clubsuit11 \vee 01\clubsuit\clubsuit1 = \bar{x}_4 x_2 \bar{x}_1 \vee \bar{x}_4 x_1 x_0 \vee \bar{x}_4 x_3 x_0; \qquad (29)$$

Возможна дальнейшая минимизация этой функции с использованием скобочной формы записи. Общая переменная $\bar{x}_4$ всех первичных импликант выносится за скобки, в которые заключается дизъюнкция оставшихся двухместных наборов переменных. Такая форма записи минимальной ФАЛ нецелесообразна при технической реализации этой функции. Скобочная форма записи ФАЛ при технической реализации на логических элементах требует многокаскадного последовательного включения логиче-

ских элементов, что понижает быстродействие всей схемы. Кроме этого, постоянное значение $\bar{x}_4$ во всех первичных импликантах для минимальной ФАЛ при технической реализации означает подключение одного из входов всех конъюнкторов к источнику «логической 1». Такие входы конъюнкторов можно из схемы исключить, то есть использовать логические схемы с количеством входов меньшим на число переменных, имеющих постоянное значение, чем количество переменных. Из аналитической записи МДНФ ФАЛ общие переменные для всех первичных импликант, имеющие постоянное значение, нужно исключить в соответствии с правилом 1&F(X) = F(X), где X – множество булевых переменных. Минимальная дизъюнктивная нормальная форма для рассмотренного примера ФАЛ запишется как:

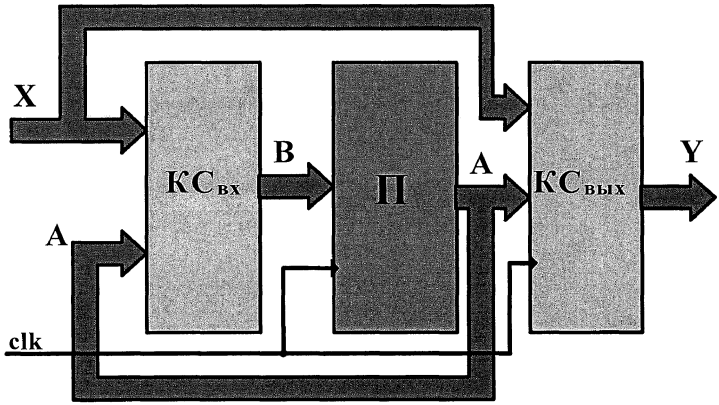
$$F = x_1x_0 \vee x_2\bar{x}_1 \vee x_3x_0; \qquad (30)$$

3. СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

3.1. Определение абстрактного цифрового автомата

Обобщённая структура системы обработки цифровой информации, приведённая на рис.1, соответствует описанию абстрактного цифрового автомата. Для целей технического проектирования в каноническую структурную систему цифрового автомата необходимо включить систему синхронизации или систему задания автоматного времени. Введение системы автоматного времени обеспечивает устойчивость работы технического устройства – цифрового автомата и дискретный характер временных процессов в нём.

С помощью импульсов синхронизации исключается возможность некорректной работы цифрового автомата во время протекания переходных процессов в элементах блока памяти и в комбинационных схемах. Цифровые автоматы, работающие под управлением системы задания дискретного автоматного времени, называются *синхронизированными цифровыми автоматами*. Наличие системы задания дискретного автоматного времени накладывает ограничения и на порядок изменения входных управляющих сигналов множества X. Поскольку сигналы множества X формируются в других блоках сложной информационной системы, то учёт ограничений системы задания дискретного времени приводит к практической необходимости включения в информационную систему любой сложности единой системы синхронизации.



$КС_{вх}$ – входная комбинационная схема; П – память; $КС_{вых}$ – выходная комбинационная схема; X – входной цифровой код; B – код возбуждения памяти; A – код состояния памяти; Y – выходной код; clk – синхроимпульсы системы автоматного времени.

Рис. 3.1

Для абстрактного математического описания цифрового автомата как кодопреобразователя (рис.3.1) используется его представление как шестиэлементного множества:

$$S = \{A, X, Y, \delta, \lambda, a_1\}, \quad (32)$$

где: $A = \{a_1, .., a_m, ..., a_M\}$ – множество состояний автомата (алфавит состояний); $X = \{z_1, ..., z_f, ..., z_F\}$ – множество входных сигналов автомата (входной алфавит); $Y = \{w_1, ..., w_g, ..., w_G\}$ – множество выходных сигналов (выходной алфавит);

δ – функция переходов абстрактного цифрового автомата, реализующая отображение множества D_δ в A (D_δ является подмножеством прямого произведения множеств A×X, то есть $D_\delta \subseteq A \times X$). Таким образом, любое состояние цифрового автомата $a_s = \delta(a_m, z_f)$, поскольку множество A×X является множеством всевозможных пар (a, z) и $a_s \in A$.

λ – функция выходов абстрактного цифрового автомата, реализующая отображение множества D_λ в Y (D_λ является подмножеством прямого произведения множеств A×X, то есть $D_\lambda \subseteq A \times X$). Таким образом, любой выходной сигнал множества Y $w_g = \lambda(a_m, z_f)$;

a_1 – начальное состояние автомата ($a_1 \in A$). Поведение цифрового автомата существенно зависит от начального состояния. Для однозначного управления цифровым автоматом необходимо, чтобы он начинал работу из определённого начального состояния. Цифровой автомат с установленным (выделенным) начальным состоянием a_1 называется инициальным.

Автомат является конечным, если A, X и Y – не являются бесконечными множествами.

Автомат является полностью определённым, если $D_\delta = D_\lambda = A \times X$. Иными словами, у полностью определённого автомата области определения функций δ и λ совпадают с множеством A×X – множеством всевозможных пар (a_m, z_f). У частичного автомата функции δ и λ определены не для всех пар $(a_m, z_f) \subseteq A \times X$.

Теоретически все элементы множеств A, X ,Y могут быть закодированы числами в системах счисления с любым основанием, но практически всегда используется двоичная система счисления (двоичный структурный алфавит).

Для двоичной системы счисления обозначим:

$A = \{a_1, .., a_m, ..., a_M\}$, $X = \{x_1, ..., x_f, ..., x_F\}$, $Y = \{y_1, ..., y_g, ..., y_G\}$ и определим разрядность двоичных кодов состояний, входного сигнала и выходного сигнала. Количество разрядов двоичного кода всегда целое число.

Количество разрядов двоичного кода состояний

$$p =]\log_2 M[. \quad (33)$$

Количество разрядов двоичного кода входных сигналов

$$r =]\log_2 F[. \quad (34)$$

Количество разрядов двоичного кода выходных сигналов

$$d =]\log_2 G[. \qquad (35)$$

В этих формулах]...[– означает ближайшее большее к значению внутреннего выражения целое число.

Согласно структурной схеме рис.3.1 коды наборов переменных комбинационных схем определяются в результате конкатенации кодов входных сигналов и кодов состояний блока памяти. Как наборы входных переменных, так и коды состояний блока памяти содержат запрещённые комбинации и поэтому системы функций алгебры логики, описывающих комбинационные схемы, будут не полностью определёнными.

Максимально возможное количество запрещённых кодов наборов переменных комбинационных схем определится как:

$$\mathbf{N_{ЗМ} = (2^p - M) \cdot 2^r + (2^r - F) \cdot 2^p;} \qquad (36)$$

В зависимости от схемы кодирования входных сигналов и состояний, среди этих запрещённых наборов могут оказаться одинаковые, и поэтому реально количество запрещённых наборов на число совпадающих кодов меньше, чем определённое по ф.(36).

Часто на практике используется две разновидности цифровых автоматов, отличающихся способом формирования выходных сигналов:

– при описании функционирования автомата выражениями:

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

$w(t) = \lambda[a(t), z(t)]$ – он называется автоматом Мили;

– при описании функционирования автомата выражениями:

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

$w(t) = \lambda[a(t)]$ – он называется автоматом Мура.

В этих выражениях t – текущий момент дискретного автоматного времени, t+1 – следующий момент дискретного автоматного времени.

3.2. Методы описания цифровых автоматов

Чтобы задать цифровой автомат S, необходимо описать все элементы множества $S = \{ A, X, Y, \delta, \lambda, a_1\}$, то есть входной и выходной алфавиты и алфавит состояний, а также функции переходов и выходов. Из множества способов описания обычно используются практически равноценные табличный и графический (с помощью ориентированных графов) [4].

Пример общего описания автоматов Мили таблицами переходов и выходов дан в табл. 3.1 и табл. 3.2.

Таблица 3.1

Общий вид таблицы переходов автомата Мили

Вход	Сост	a_1	...	a_M
z_1		$\delta(a_1,z_1)$	...	$\delta(a_M,z_1)$
...		...	...	...
z_F		$\delta(a_1,z_F)$	...	$\delta(a_M,z_F)$

Таблица 3.2

Общий вид таблицы выходов автомата Мили

Вход	Сост	a_1	...	a_M
z_1		$\lambda(a_1,z_1)$	...	$\lambda(a_M,z_1)$
...		...	...	...
z_F		$\lambda(a_1,z_F)$	...	$\lambda(a_M,z_F)$

Строки этих таблиц соответствуют входным сигналам множества X, а столбцы – состояниям A, причём крайний левый столбец обозначен как начальное состояние инициального цифрового автомата a_1.

На пересечении столбца a_m и строки z_f в таблице переходов ставится состояние $a_s=\delta(a_m, z_f)$, в которое автомат переходит из состояния a_m под действием сигнала z_f. В таблице выходов – соответствующий этому переходу выходной сигнал $w_g=\lambda(a_m, z_f)$.

Пример табличного описания полностью определённого цифрового автомата Мили S_1 с тремя состояниями, двумя входными и двумя выходными сигналами приведён в таблице 3.3 и таблице 3.4.

Таблица 3.3

Таблица переходов а. Мили S_1

Вход	Сост	a_1	a_2	a_3
z_1		a_3	a_1	a_1
z_2		a_1	a_3	a_2

Таблица 3.4

Таблица выходов а. Мили S_1

Вход	Сост	a_1	a_2	a_3
z_1		w_1	w_1	w_2
z_2		w_1	w_2	w_1

Заголовочная строка и столбец обозначены одинаково для обеих таблиц, поэтому для экономии времени можно производить описание автомата Мили одной совмещённой таблицей переходов и выходов, например, таблицей 3.5.

Таблица 3.5

Совмещённая таблица переходов и выходов автомата Мили S_1

Вход	Сост	a_1	a_2	a_3
z_1		a_3 / w_1	a_1 / w_1	a_1 / w_2
z_2		a_1 / w_1	a_3 / w_2	a_2 / w_1

Для частичных автоматов, у которых функции δ и λ определены не для всех пар $(a_m, z_f) \subseteq A \times X$, на месте неопределённых состояний и неопределённых выходных сигналов ставится какой либо специальный символ, например, прочерк. Пример табличного описания частичного автомата приведён в совмещённой таблице 3.6.

Таблица 3.6

Совмещённая таблица для частичного а. Мили S_2

Вход	Сост	a_1	a_2	a_3	a_4
z_1		a_2 / w_1	a_3 / w_3	a_4 / w_3	- / -
z_2		a_3 / w_2	- / -	a_2 / w_1	a_2 / w_2

Так как в автомате Мура выходной сигнал зависит только от состояния, то автомат Мура описывается одной отмеченной таблицей переходов (общая форма – таблица 3.7, пример – таблица 3.8), в которой каждому её столбцу приписан, кроме состояния a_m, ещё и соответствующий выходной сигнал $w_g=\lambda(a_m)$.

Таблица 3.7

Общий вид отмеченной таблицы переходов автомата Мура

	Вых	$\lambda(a_1)$	...	$\lambda(a_M)$
Вх	Сост	a_1	...	a_M
z_1		$\delta(a_1, z_1)$	...	$\delta(a_M, z_1)$
...		...	...	...
z_F		$\delta(a_1, z_F)$	...	$\delta(a_M, z_F)$

Таблица 3.8

Отмеченная таблица переходов автомата Мура S_3

	Вых	w_1	w_1	w_3	w_2	w_3
Вх	Сост	a_1	a_2	a_3	a_4	a_5
z_1		a_2	a_5	a_5	a_3	a_3
z_2		a_4	a_2	a_2	a_1	a_1

Граф автомата – ориентированный связный граф, вершины которого соответствуют состояниям, а дуги – переходам между ними.

Две вершины графа автомата a_m и a_s (исходное состояние и состояние перехода) соединяются дугой, направленной от a_m к a_s , если в автомате имеется переход из a_m в a_s , то есть если $a_s = \delta(a_m, z_f)$ при некотором $z_f \in X$. Дуге (a_m, a_s) графа автомата приписывается входной сигнал z_f и выходной сигнал $w_g = \lambda(a_m, z_f)$, если он определён, и ставится прочерк в противном случае. Если переход автомата из состояния a_m в состояние a_s происходит под действием нескольких входных сигналов, то дуге (a_m, a_s) приписываются все эти входные и соответствующие выходные сигналы.

При описании автомата Мура в виде графа выходной сигнал $w_g = \lambda(a_m)$ записывается внутри вершины a_m или рядом с ней.

На рис.3.9, рис.3.10 и рис.3.11 приведены графы цифровых автоматов S_1, S_2 и S_3, описанные ранее в таблице 3.5, таблице 3.6 и таблице 3.8

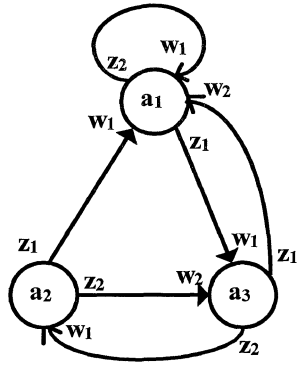


Рис. 3.2. Граф автомата Мили S_1

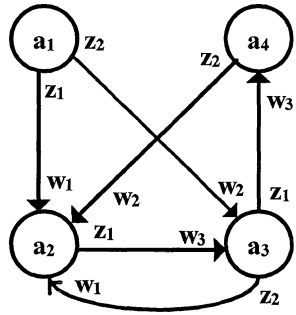


Рис. 3.3. Граф автомата Мили S_2

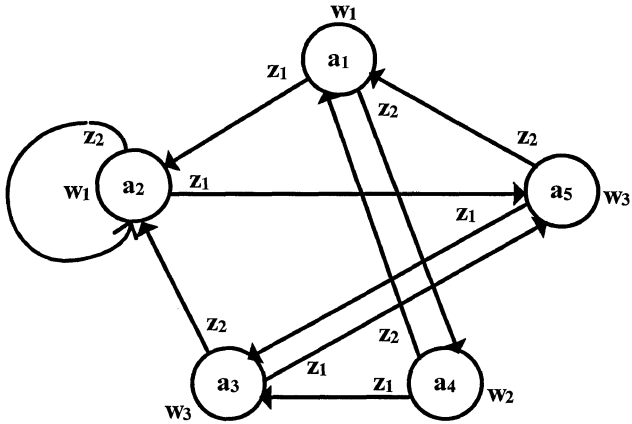


Рис. 3.4. Граф автомата Мили S_4

В технических целях используются только детерминированные цифровые автоматы, в которых выполнено условие однозначности переходов: автомат, находящийся в некотором состоянии, под действием любого входного сигнала не может перейти более, чем в одно состояние. Применительно к табличному способу задания описания автоматов это означает, что в клетках переходов/выходов указывается только по одному состоянию/выходному сигналу. Применительно к графическому способу задания описания автоматов это означает, что в графе автомата из любой вершины не могут выходить две или более дуги, отмеченные одним и тем же входным сигналом.

3.3. Синхронные и асинхронные цифровые автоматы

Состояние a_s автомата S называется устойчивым состоянием, если для любого входа $z_f \in X$, такого, что $\delta(a_m, z_f) = a_s$, имеет место $\delta(a_s, z_f) = a_s$. Это означает, что если автомат перешёл в некоторое состояние под действием z_f, то выйти из этого состояния он может только под действием другого, отличного от z_f сигнала.

Автомат S называется асинхронным, если каждое его состояние $a_s \in A$ – устойчиво. Автомат S называется синхронным, если он не является асинхронным.

Созданные для практических применений цифровые автоматы всегда являются асинхронными, а устойчивость их состояний всегда обеспечивается тем или иным способом, например, введением сигналов синхронизации.

Таким образом, оказывается, что техническая терминология противоречит математической терминологии, так как, согласно приведённым выше определениям, синхронизированные (технический приём) цифровые автоматы всегда являются асинхронными (математическое определение).

На уровне абстрактной теории, когда цифровой автомат – всего лишь математическая модель, не отражающая многих конкретных особенностей его возможной реализации, часто оказывается более удобным оперировать с синхронными цифровыми автоматами.

Пример асинхронного цифрового автомата Мура S_4 приведён в отмеченной таблице 3.9, а его граф – на рис.3.5. Очевидно, что все его состояния устойчивы. Если в таблице переходов асинхронного цифрового автомата некоторое состояние a_s стоит на пересечении строки z_f и столбца a_m ($m \neq s$), то это же состояние a_s обязательно должно встретиться в этой же строке в столбце a_s. В графе асинхронного цифрового автомата, если в некоторое состояние a_s имеются переходы из других состояний под действием каких-то сигналов, то в вершине a_s должна быть петля, отмеченная символами тех же входных сигналов.

Таблица 3.9

Отмеченная таблица переходов асинхронного автомата Мура S_4

	Вых	w_1	w_3	w_2
Вх	Сост	a_1	a_2	a_3
z_1		a_2	a_2	a_2
z_2		a_3	a_2	a_3
z_3		a_1	a_1	a_3

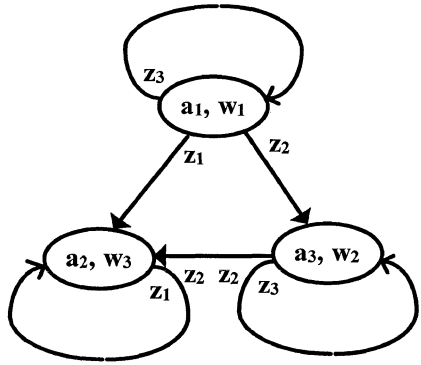


Рис. 3.5. Граф асинхронного автомата Мили S_4

Ранее приведенные примеры описаний цифровых автоматов S_1, S_2 и S_3 таблицами и графами, являются примерами синхронных цифровых автоматов.

3.4. Связь между математическими моделями цифровых автоматов Мили и Мура

Абстрактный цифровой автомат работает как преобразователь слов входного алфавита в слово в выходном алфавите [5]. Рассмотрим это положение, взяв в качестве примера автомат Мили S_1.

На вход этого автомата, установленного в начальное состояние a_1, поступает шестибуквенное (шеститактное) слово $\xi=z_1z_1z_2z_1z_2z_2$.

Проследив по таблицам переходов и выходов или непосредственно по графу поведение цифрового автомата S_1, опишем его тремя сроками, первая из которых соответствует входному слову ξ, вторая – последовательности соответствующих этому слову состояний β, третья – выходному слову ω, которое появляется на выходе автомата.

$$\xi = z_1\ z_1\ z_2\ z_1\ z_2\ z_2 \text{ - k символов}$$

$$\beta = a_1 \to a_3 \to a_1 \to a_1 \to a_3 \to a_2 \to a_3 \text{ - k+1 символ}$$

$$\omega = w_1\ w_2\ w_1\ w_1\ w_1\ w_2 \text{ - k символов}$$

$\omega = \lambda(a_1, \xi)$ – реакция автомата Мили в состоянии a_1 на входное слово ξ.

Как видно из этого примера, в ответ на входное слово длиной k символов автомат Мили S_1 выдаёт последовательность состояний длины k+1 символов и выходное слово длиной k символов.

В общем виде поведение автомата Мили, установленного в состояние a_m, можно описать следующим образом:

входное слово	z_{i1}	z_{i2}	z_{i3}
последовательность состояний	a_m	$a_{i2}=\delta(a_m, z_{i1})$	$a_{i3}=\delta(a_{i2}, z_{i2})$
выходное слово	$w_{i1}=\lambda(a_m, z_{i1})$	$w_{i2}=\lambda(a_{i2}, z_{i2})$	$w_{i3}=\lambda(a_{i3}, z_{i3})$.

Точно так же описывается поведение автомата Мура, находящегося в состоянии a_m, при приходе входного слова $z_{i1}, z_{i2}, \dots, z_{ik}$. Учитывая, что выходной сигнал автомата Мура в момент времени t, то есть w(t), зависит лишь от состояния, в котором находится автомат в момент t, то есть от a(t), получим:

входное слово	z_{i1}	z_{i2}	z_{i3}	
последовательность состояний	a_m	$a_{i2}=\delta(a_m, z_{i1})$	$a_{i3}=\delta(a_{i2}, z_{i2})$	$a_{i4}=\delta(a_{i3}, z_{i3})$
выходное слово	$w_{i1}=\lambda(a_m)$	$w_{i2}=\lambda(a_{i2})$	$w_{i3}=\lambda(a_{i3})$	$w_{i4}=\lambda(a_{i4})$.

Выходной сигнал $w_{i1}=\lambda(a_m)$ в момент времени i1 не зависит от входного сигнала z_{i1}, а определяется только состоянием a_m. Таким образом, выходной сигнал w_{i1} не связан с входным словом, поступающим на вход автомата, начиная с момента i1.

По этой причине реакцией автомата Мура, установленного в состояние a_m, на входное слово $\xi = z_{i1}, z_{i2}, \dots, z_{ik}$ является выходное слово той же длины k, с исключением первого символа выходного слова:

$$\omega = \lambda(a_m, \xi) = w_{i2}, w_{i3}, \dots, w_{i(k+1)}.$$

В качестве примера рассмотрим автомат Мура S_5 , граф которого изображён на рис.26, и определим его реакцию в начальном состоянии a_1 на то же самое слово $\xi=z_1z_1z_2z_1z_2z_2$, которое использовалось для оценки поведения автомата Мили S_1.

$\xi =$ z_1 z_1 z_2 z_1 z_2 z_2 - **k символов**

$\beta =$ $a_1 \to a_4 \to a_2 \to a_1 \to a_4 \to a_3 \to a_5$ - **k+1 символ**

$\omega =$ w_1 w_1 w_2 w_1 w_1 w_1 w_2 - **k+1 символ**

Часть выходного слова, выделенная штриховой рамкой, является реакцией автомата Мура на входное слово.

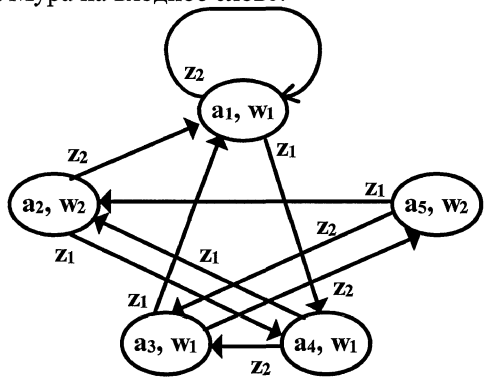


Рис. 3.6. Граф автомата Мили S_5

Таким образом, реакции автоматов S_5 и S_1 в начальном состоянии на входное слово ξ с точностью до сдвига на один такт совпадают. Эти автоматы являются эквивалентными.

Два автомата S_A и S_B с одинаковыми входными выходным алфавитами называются эквивалентными, если после установки начального состояния, их реакции на любое входное слово совпадают. Отсюда следует, что для любого автомата Мили существует эквивалентный ему автомат Мура и, обратно, для любого автомата Мура существует эквивалентный ему автомат Мили, то есть любой автомат Мили можно трансформировать в эквивалентный ему автомат Мура, а любой автомат Мура можно трансформировать в эквивалентный ему автомат Мили.

При рассмотрении алгоритмов взаимной трансформации автоматов Мили и Мура, в соответствии с изложенным выше, всегда будем пренебрегать в автоматах Мура выходным сигналом, связанным с начальным состоянием, то есть функцией $\lambda(a_1)$.

Рассмотрим преобразование автомата Мура в эквивалентный ему автомат Мили.

Исходный автомат Мура

$$S_A = \{A_A, X_A, Y_A, \delta_A, \lambda_A, a_{1A}\}$$

где: $A_A = \{a_1, \dots, a_m, \dots, a_M\}$, $X_A = \{z_1, \dots, z_f, \dots, z_F\}$, $Y_A = \{w_1, \dots, w_g, \dots w_G\}$, δ_A: $A_A \times X_A \to A_A$, $\lambda_A = A_A \to Y_A$, $a_{1A} = a_1$ – начальное состояние.

Эквивалентный ему автомат Мили

$$S_B = \{A_B, X_B, Y_B, \delta_B, \lambda_B, a_{1B}\},$$

имеющий

$$A_B = A_A;\ X_B = X_A;\ Y_B = Y_A;\ \delta_B = \delta_A;\ a_{1B} = a_{1A}.$$

Для S_B функция выходов λ_B определяется иначе, чем для исходного автомата Мура S_A. Если в исходном автомате Мура $\lambda_A(a_s) = w_g$, то в эквивалентном автомате Мили $\lambda_B(a_m, z_f) = w_g$.

Переход от автомата Мура к автомату Мили при графическом описании иллюстрируется рис.27. Выходной сигнал w_g, записанный рядом с вершиной a_s, переносится на все дуги, входящие в эту вершину.

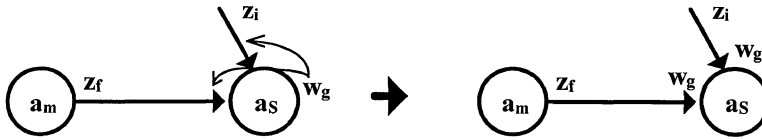


Рис. 3.7. Переход от автомата Мили к автомату Мура

При использовании табличного описания автомата Мура S_A, таблица переходов автомата Мили S_B совпадает с таблицей переходов S_A, а таблица выходов S_B получается из таблицы переходов заменой символа a_s, стоящего на пересечении строки z_f и столбца a_m, символом выходного сигнала w_g, отмечающего столбец a_s в таблице переходов автомата S_A.

Пример трансформации автомата Мура S_3 (таблица 3.8) в автомат Мили S_6 приведён на рис.3.8 и в таблице 3.12 и таблице 3.13.

Таблица 3.10

Отмеченная таблица переходов автомата Мура S_3

	Вых	w_1	w_1	w_3	w_2	w_3
Вх	Сост	a_1	a_2	a_3	a_4	a_5
z_1		a_2	a_5	a_5	a_3	a_3
z_2		a_4	a_2	a_2	a_1	a_1

Таблица 3.11

Выделение совмещённой таблицы переходов и выходов автомата Мили S_6

	Вых	w_1	w_1	w_3	w_2	w_3
Вх	Сост	a_1	a_2	a_3	a_4	a_5
z_1		a_2/w_1	a_5/w_3	a_5/w_3	a_3/w_3	a_3/w_3
z_2		a_4/w_2	a_2/w_1	a_2/w_1	a_1/w_1	a_1/w_1

Таблица 3.12

Таблица переходов эквивалентного автомата Мили S_6

Вх / Сост	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_5	a_5	a_3	a_3
z_2	a_4	a_2	a_2	a_1	a_1

Таблица 3.13

Таблица выходов эквивалентного автомата Мили S_6

Вх / Сост	a_1	a_2	a_3	a_4	a_5
z_1	w_1	w_3	w_3	w_3	w_3
z_2	w_2	w_1	w_1	w_1	w_1

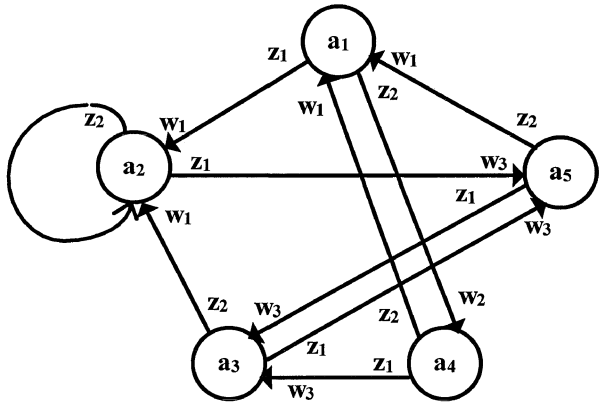


Рис. 3.8. Граф автомата Мили S_6, эквивалентного автомату Мура S_3

По процедуре построения автомата Мили S_B видно, что он эквивалентен автомату Мура S_A.

Действительно, если некоторый входной сигнал $z_f \in X$ поступает на вход автомата S_A, находящегося в состоянии a_m, то он перейдёт в состояние $a_s = \delta_A(a_m, z_f)$ и выдаст выходной сигнал $w_g = \lambda_A(a_s)$. Соответствующий эквивалентный автомат Мили S_B из состояния a_m также перейдёт в состояние a_s, поскольку $\delta_B(a_m, z_f) = \delta_A(a_m, z_f) = a_s$ и выдаст тот же выходной сигнал w_g согласно способу построения функции λ_B . Таким образом, любое входное слово конечной длины, поданное на входы автоматов S_A и S_B, установленных в состояние a_m, вызовет появление одинаковых выходных слов и, следовательно, автоматы S_A и S_B эквивалентны.

При рассмотрении процесса трансформации автомата Мили в автомат Мура сначала на описание исходного автомат Мили накладывается ограничение: автомат Мили не должен иметь преходящих состояний.

Преходящим называется состояние, в которое, при представлении автомата в виде графа, не входит ни одна дуга, но которое имеет, по крайней мере, одну выходящую дугу (например – состояние a_1 автомата S_2 на рис.3.12).

Рассмотрим преобразование автомата Мили в эквивалентный ему автомат Мура.

Исходный автомат Мили

$$S_A = \{A_A, X_A, Y_A, \delta_A, \lambda_A, a_{1A}\}$$

где: $A_A = \{a_1, \dots, a_m, \dots, a_M\}$, $X_A = \{z_1, \dots, z_f, \dots, z_F\}$, $Y_A = \{w_1, \dots, w_g, \dots w_G\}$, δ_A реализует отображение $A_A \times X_A$ в A_A, то есть δ_A: $A_A \times X_A \to A_A$ и $\lambda_A = A_A \to Y_A$, $a_{1A} = a_1$ – начальное состояние.

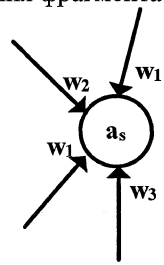
Эквивалентный ему автомат Мура

$$S_B = \{A_B, X_B, Y_B, \delta_B, \lambda_B, a_{1B}\},$$

имеющий

$$X_B = X_A;\ Y_B = Y_A.$$

Для определения A_B каждому состоянию $a_s \in A_A$ поставим в соответствие множество A_S всевозможных пар вида (a_s, w_g), где w_g – выходной сигнал, приписанный входящей в a_s дуге. Это положение иллюстрируется примером преобразования фрагмента автомата Мили на рис.3.9.



$$\mathbf{A_S = \{(a_s, w_1), (a_s, w_2), (a_s, w_3)\}}$$

Рис. 3.9. Построение множества A_s

Множество состояний автомата S_B, получим как объединение множеств A_S (S = 1, ... ,M):

$$\mathbf{A_B = \bigcup_{S=1}^{M} A_S}\ .$$

Для определения функций выходов λ_B, каждому состоянию эквивалентного автомата Мура, представляющего собой пару вида (a_s, w_g), поставим в соответствие выходной сигнал w_g. Если в автомате Мили S_A, был переход $\delta_A(a_m, z_f) = a_s$ и при этом выдавался выходной сигнал $\lambda_A(a_m, z_f) = w_k$, то в эквивалентном автомате Мура S_B (рис.3.10) будет переход из множества состояний A_m, порождаемых a_m, в состояние (a_s, w_k) под действием того же входного сигнала z_f.

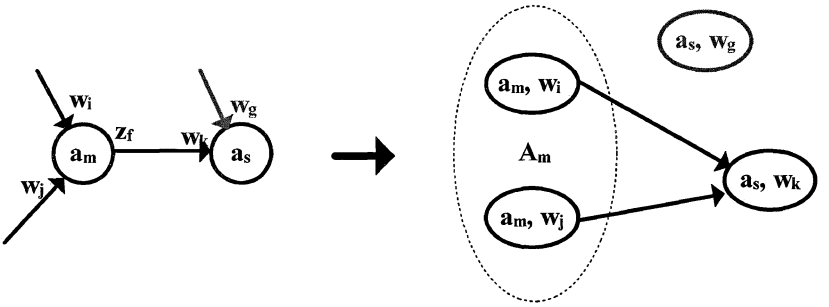


Рис. 3.10. Иллюстрация трансформации автомата Мили в автомат Мура

Как начальное состояние a_{1B} можно взять любое состояние из множества A_1, порождаемого начальным состоянием a_1 автомата S_A.

При последующем сравнении реакций эквивалентных автоматов S_A и S_B на всевозможные входные слова не должен учитываться выходной сигнал в момент $t = 0$, связанный с состоянием a_{1B} автомата Мура S_B.

Рассмотрим пример преобразования автомата Мили S_1, изображённого в виде графа на рис.3.2 и описанного в таблице 3.5.

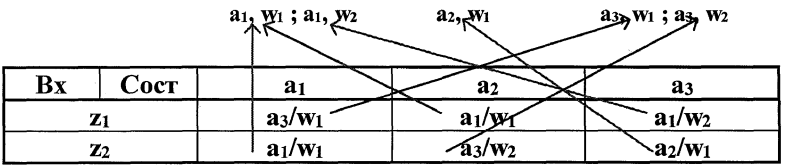


Рис. 3.11

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

$A_3 = \{(a_3,w_1),(a_3,w_2)\} = \{b_4,b_5\}$, где $b_4 = (a_3,w_1)$ $b_5 = (a_3,w_2)$.

Тогда $\mathbf{A_B = A_1 \cup A_2 \cup A_3} = \{b_1, b_2, b_3, b_4, b_5\}$. С каждым состоянием, представляющим теперь пару, свяжем выходной сигнал, являющийся вторым элементом этой пары:

$$\lambda_B(b_1) = \lambda_B(b_3) = \lambda_B(b_4) = w_1;\ \lambda_B(b_2) = \lambda_B(b_5) = w_2.$$

В качестве начального состояния можно выбрать любое из множества A_1. Таким образом строится отмеченная таблица выходов эквивалентного автомата Мура. Оказывается, что это автомат S_5, граф которого ранее изображён на рис. 3.6.

Таблица 3.14

Отмеченная таблица переходов эквивалентного автомата Мили S_5

	Вых	w_1	w_2	w_1	w_1	w_2
Вх	Сост	b_1	b_2	b_3	b_4	b_5
z_1		b_4	b_4	b_1	b_2	b_2
z_2		b_1	b_1	b_5	b_3	b_3

Рассмотрим преобразование автомата Мили в эквивалентный автомат Мура с учётом ранее введённого ограничения на присутствие преходящих состояний. Для примера возьмём автомат Мили S_7, граф которого приведён на рис.3.12.

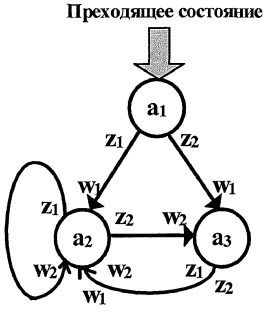


Рис. 3.12. Автомат Мили S_7 с переходящим состоянием a_1

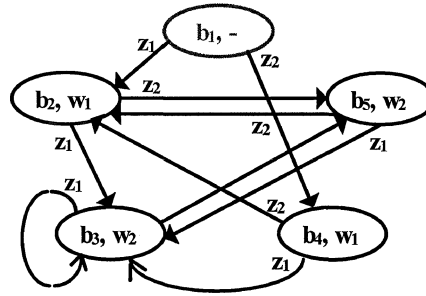


Рис. 3.13. Автомат Мили S_8, эквивалентный автомату S_7

Таблица 3.15

Совмещенная таблица автомата Мили S_7

Вх	Сост	a_1	a_2	a_3
z_1		a_2/w_1	a_2/w_2	a_2/w_2
z_2		a_3/w_1	a_3/w_2	a_2/w_1

Таблица 3.16

Отмеченная таблица эквивалентного автомата S_8

	Вых	-	w_1	w_2	w_1	w_2
Вх	Сост	b_1	b_2	b_3	b_4	b_5
z_1		b_2	b_3	b_3	b_3	b_3
z_2		b_4	b_5	b_5	b_2	b_2

$b_2 = (a_2, w_1)$; $b_3 = (a_2, w_2)$; $b_4 = (a_3, w_1)$; $b_5 = (a_3, w_2)$; $b_1 = (a_1, -)$.

Преходящее состояние в автомате Мили порождает состояние с неопределённым выходным сигналом в эквивалентном автомате Мура.

Рассмотренные взаимные трансформации автоматов Мили и Мура показывают, что при переходе от автомата Мура к автомату Мили число состояний эквивалентного автомата не увеличивается, тогда как при обратном переходе число состояний, с высокой степенью вероятности, увеличивается.

Таким образом, эквивалентные между собой автоматы могут иметь различное количество состояний, в связи с чем формулируется задача

нахождения минимального по количеству состояний автомата в классе эквивалентных между собой автоматов.

3.5. Минимизация абстрактных цифровых автоматов

Абстрактный автомат, построенный по техническому заданию формальным или эвристическим методами, обычно не является минимальным по количеству состояний. Построение эквивалентного ему абстрактного цифрового автомата с наименьшим числом состояний и является задачей оптимизации. При минимизации числа состояний уменьшается стоимость как блока памяти автомата, так и его входной и выходной комбинационных схем.

Два полностью определённых автомата называются эквивалентными, если они индуцируют (производят) одно и то же отображение множества входных слов во множество выходных слов.

Частичный цифровой автомат индуцирует лишь частичное отображение множества входных с лов в выходные слова.

Два частичных автомата с одинаковыми алфавитами входа и выхода называются эквивалентными, если индуцируемые ими частичные отображения входных слов в выходные совпадают.

Полностью определённый автомат является частным случаем частичного автомата.

3.5.1. Минимизация абстрактного автомата Мили

Для табличного описания процедура минимизации цифровых автоматов алгоритмизирована и выполняется в несколько шагов.

Шаг 1 Распространение неопределённости таблицы выходов на таблицу переходов. Если для некоторой пары (a_m, z_f) выходной сигнал автомата не определён, то для этой пары не определяется и функция перехода, так как не определено допустимое слово, осуществляющее переход из этого состояния.

Шаг 2 Исключение недостижимых состояний. Если в автомате имеется состояние (но только не начальное), в которое он не может попасть под воздействием любого допустимого входного слова, то такое состояние называется недостижимым. Недостижимые состояния исключаются из описания абстрактного автомата без изменения индуцируемого автоматом отображения. Автомат, все состояния которого достижимы, является связным автоматом.

Выполнение первых двух шагов минимизации иллюстрируется на примере автомата Мили S_9.

Таблица 3.17

Совмещенная таблица переходов и выходов автомата Мили S_9

Вх	Сост	a_1	a_2	a_3	a_4	a_5	a_6
z_1		-/-	a_4/w_3	a_5/w_5	a_3/w_4	~~a_1~~/-	~~a_1~~/-
z_2		a_3/w_1	a_1/w_4	a_3/w_3	a_6/-	-/w_1	-/w_1
z_3		a_1/w_2	-/-	a_1/w_3	a_2/w_1	a_5/w_2	-/-

Недостижимое Недостижимое Недостижимое

Таблица 3.18

Минимизированный автомат Мили S_9

Вх	Сост	a_1	a_3	a_5
z_1		-/-	a_5/w_5	-/-
z_2		a_3/w_1	a_3/w_3	-/w_1
z_3		a_1/w_2	a_1/w_3	a_5/w_2

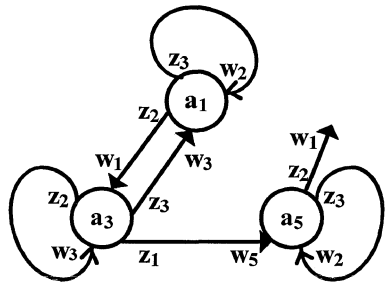


Рис. 3.14. Граф автомата Мили после двух шагов минимизации

Шаг 3 Нахождение совместимых состояний автомата.

Состояния a_i и a_j называются совместимыми, если двигаясь из этих состояний под воздействием любого входного сигнала, автомат индуцирует одинаковое его отображение.

Состояния называются i-совместимыми для i = 1, 2, ..., если результат применения к этим состояниям любого слова длины i будет одинаковым. Классы совместимых состояний могут быть найдены непосредственно по таблице выходов. В один и тот же 1 – класс зачисляются состояния, обозначающие совпадающие (с точностью до неопределённых выходных сигналов) столбцы таблицы выходов. Классы (i+1)-совместимости получаются из классов i-совместимости путём их расщепления на классы (i+1)-совместимости. Для этого у каждого состояния, принадлежащего j - классу i-совместимости $C_j(i)$, номера классов (индексы), в которые автомат пере-

ходит под воздействием каждой входной буквы. Если номер класса не определён, то ставится специальный символ, например, прочерк. Индексы классов, в которые переходит автомат под действием входного сигнала образуют отметку. Множество состояний с одинаковыми отметками в классе $C_j(i)$ образуют классы (i+1)-совместимости. При выполнении операции расщепления классов специальный символ неопределённости может быть заменён номером (индексом) любого класса. Если операцию расщепления i-классов применить последовательно, начиная с 1-класса, то через конечное число шагов процесс расщепления закончится. Нерасщепляемые далее классы образуют классы совместимых состояний. Иногда отметки состояний разных классов совпадают, но объединять такие состояния в один класс (i+1)-совместимости совершенно недопустимо.

Отыскание классов совместимых состояний рассмотрим для примера автомата Мили S_{10}, описываемого совмещённой таблицей 3.19 переходов и выходов.

Таблица 3.19

Совмещенная таблица переходов и выходов автомата Мили S_{10}

Вх / Сост	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_2/w_1	-/-	a_3/w_1	a_5/w_1	-/-	a_7/w_1	$-/w_1$	-/-
z_2	a_3/w_1	$-/w_2$	a_1/w_1	$-/w_2$	a_3/w_2	a_8/w_1	a_8/w_2	a_6/w_1

Класс 1 Класс 2

Процедуру расщепления классов для нахождения классов конечной совместимости удобно проводить с использованием таблиц (рис.3.15).

Классы единичной совместимости

$C_1(1)$	z_1 ; z_2
1	2 ; 1
3	1 ; 1
6	2 ; 1
8	- ; 1

$C_2(1)$	z_1 ; z_2
2	- ; -
4	2 ; -
5	- ; 1
7	- ; 1

Классы двоичной совместимости

$D_1(2)$	z_1 ; z_2
1	3 ; 2
6	3 ; 2(1)
~~8~~	~~- ; 1~~

$D_2(2)$	z_1 ; z_2
3	2 ; 1
8	- ; 1

$D_3(2)$	z_1 ; z_2
2	- ; -
4	3 ; -
5	- ; 2
7	- ; 2(1)

Рис. 3.15. Нахождение классов конечной совместимости

Задачей минимизации методом расщепления классов совместимости является получение как можно меньшего количества, как можно большей ёмкости классов конечной совместимости. Поэтому состояние 8 (a_8) первоначально отнесённое к двум классам двоичной совместимости из-за неопределённой первой отметки окончательно должно быть отнесено ко второму классу. Классы двоичной совместимости далее не расщепляются.

Всё множество совместимых состояний определяет некоторое множество минимизированных автоматов. Все они могут быть представлены нормализованным автоматом, в котором вместо состояний используются классы конечной совместимости. Для получения нормализованного автомата в процессе минимизации нужно строго следить за тем, чтобы начальное состояние всегда находилось в классе с индексом 1.

При построении нормализованного автомата переход $\delta = (C_i, z_j)$ считается неопределённым, если для всех состояний этого класса не определены переходы в другое состояние. Если хотя бы для одного состояния класса переход определён, то в клетку таблицы нормализованного автомата заносится индекс класса, в который переходит цифровой автомат из этого состояния. Таким образом доопределяются неопределённые переходы исходного автомата. Нормализованный автомат является эквивалентным любому из минимизированных автоматов и не имеет, как минимум, ни одной пары совместимых состояний. В соответствии с изложенной методикой минимизации получаются либо полностью определённые, либо частичные нормализованные автоматы. У полностью определённых автоматов класс конечной совместимости не пересекаются, поэтому нормализованный автомат является единственным и процесс минимизации этим заканчивается. В случае получения частичного автомата классы i-совместимости пересекаются. Это приводит к тому, что нормализованный автомат может описываться конечным количеством вариантов таблиц или графов. В случае частичных автоматов часто отказываются от достижения абсолютной минимизации и ограничиваются нахождением нормализованного автомата и его эвристическим доопределением.

Таким образом, основной алгоритм минимизации автомата Мили состоит из следующих шагов:

1 Распространение неопределённости выходов на переходы автомата.

2 Исключение недостижимых состояний.

3 Определение классов совместимости и построение нормализованного автомата.

4 Минимизация нормализованного автомата.

Для полностью определённого цифрового автомата отсутствует последниё шаг алгоритма.

Для сложных автоматов возможно изменение очерёдности выполнения шага 2 и шага 3 этого алгоритма, поскольку для минимизированного нормализованного автомата определение недостижимых состояний требует меньших затрат труда и времени.

Совмещённая таблица переходов и выходов нормализованного минимизированного автомата Мили S_{10} представлена в таблице 3.20, а его граф – на рис. 35.

Таблица 3.20

Совмещенная таблица переходов и выходов нормализованного минимизированного автомата Мили S_{10}

Вх / Сост	D_1	D_2	D_3
z_1	D_3/w_1	D_2/w_1	D_3/w_1
z_2	D_2/w_1	D_1/w_1	D_2/w_2

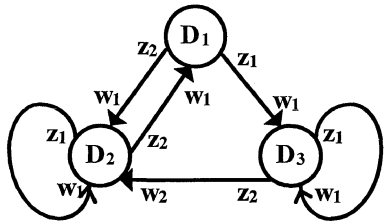


Рис. 3.16. Граф нормализованного минимизированного автомата Мили S_{10}

Более подробно рассмотрим применение методики минимизации автоматов Мили на примере полностью определённого автомата S_{11}.

Таблица 3.21

Совмещенная таблица переходов и выходов автомата Мили S_{11}

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

Классы единичной совместимости

C_1	$z_1 ; z_2$
1	2 ; 1
2	2 ; 1
5	2 ; 2
7	2 ; 2
8	2 ; 2

C_2	$z_1 ; z_2$
3	1 ; 2
4	1 ; 2
6	1 ; 2
9	1 ; 2
10	1 ; 1
11	1 ; 2
12	1 ; 1

Классы двоичной совместимости

D_1	$z_1 ; z_2$
1	4 ; 2
2	4 ; 2

D_2	$z_1 ; z_2$
5	3 ; 3
7	3 ; 3
8	4 ; 3

D_3	$z_1 ; z_2$
3	2 ; 3
4	2 ; 3
6	2 ; 3
9	2 ; 3
11	2 ; 3

D_4	$z_1 ; z_2$
10	1 ; 2
12	1 ; 2

Классы троичной совместимости

E_1	$z_1 ; z_2$
1	5 ; 2
2	5 ; 3

E_2	$z_1 ; z_2$
5	4 ; 4
7	4 ; 4

E_3	$z_1 ; z_2$
8	5 ; 4

E_4	$z_1 ; z_2$
3	2 ; 4
4	2 ; 4
6	2 ; 4
9	2 ; 4
11	2 ; 4

E_5	$z_1 ; z_2$
10	1 ; 4
12	1 ; 4

Рис. 3.17

Классы четверичной совместимости

F_1	$z_1 ; z_2$
1	6 ; 3

F_2	$z_1 ; z_2$
2	6 ; 4

F_3	$z_1 ; z_2$
5	5 ; 5
7	5 ; 5

F_4	$z_1 ; z_2$
8	6 ; 5

F_5	$z_1 ; z_2$
3	3 ; 5
4	3 ; 5
6	3 ; 5
9	3 ; 5
11	3 ; 5

F_6	$z_1 ; z_2$
10	1 ; 4
12	2 ; 4

Классы пятиричной совместимости

G_1	$z_1 ; z_2$
1	6 ; 3

G_2	$z_1 ; z_2$
2	7 ; 4

G_3	$z_1 ; z_2$
5	5 ; 5
7	5 ; 5

G_4	$z_1 ; z_2$
8	6 ; 5

G_5	$z_1 ; z_2$
3	3 ; 5
4	3 ; 5
6	3 ; 5
9	3 ; 5
11	3 ; 5

G_6	$z_1 ; z_2$
10	1 ; 4

G_6	$z_1 ; z_2$
12	2 ; 4

Рис. 3.18

Расщепление на классы пятиричной совместимости является конечным, так как метки в пределах классов для всех состояний класса одинаковы и, следовательно, дальнейшее расщепление классов невозможно. Метки состояний соответствуют переходам автомата. По этим меткам определяется, что классы G_2 и G_7 являются недостижимыми состояниями для нормализованного автомата.

Таблица 3.22

Совмещенная таблица переходов и выходов нормализованного минимизированного автомата Мили S_{12}

Вх / Сост	G_1	G_2	G_3	G_4	G_5	G_6	G_7
z_1	G_6/w_1	G_7/w_1	G_5/w_1	G_6/w_1	G_3/w_2	G_1/w_2	G_2/w_2
z_2	G_3/w_2	G_4/w_2	G_5/w_2	G_5/w_2	G_5/w_1	G_4/w_1	G_4/w_1

Недостижимое Недостижимое

Граф минимизированного нормализованного автомата Мили S_{12} представлен на рис.3.19.

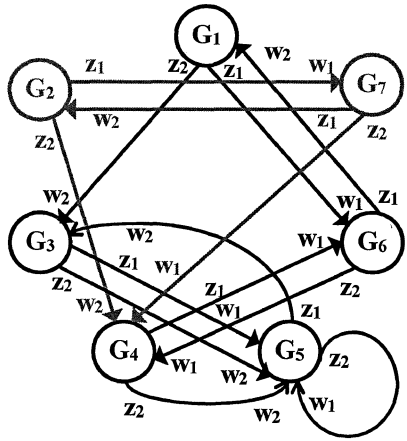


Рис. 3.19. Граф нормализованного минимизированного автомата Мили S_{12}

После исключения недостижимых состояний получен минимальный нормализованный автомат Мили S_{13}, представленный совмещённой таблицей 3.33 переходов и выходов.

Таблица 3.21

Совмещенная таблица переходов и выходов автомата Мили S_{11}

Сост \ Вх	z_1	z_2
a_1	a_{10}/w_1	a_5/w_2
a_2	a_{12}/w_1	a_8/w_2
a_3	a_5/w_2	a_6/w_1
a_4	a_7/w_2	a_{11}/w_1
a_5	a_3/w_1	a_9/w_2
a_6	a_7/w_2	a_{11}/w_1
a_7	a_3/w_1	a_6/w_2
a_8	a_{10}/w_1	a_4/w_2
a_9	a_7/w_2	a_6/w_1
a_{10}	a_1/w_2	a_8/w_1
a_{11}	a_5/w_2	a_9/w_1
a_{12}	a_2/w_2	a_8/w_1

Таблица 3.33

Совмещенная таблица переходов и выходов нормализованного минимизированного автомата Мили S_{13}

Вх \ Сост	G_1	G_3	G_4	G_5	G_6
z_1	G_6/w_1	G_5/w_1	G_6/w_1	G_3/w_2	G_1/w_2
z_2	G_3/w_2	G_5/w_2	G_5/w_2	G_5/w_1	G_4/w_1

Сравнением этих таблиц определяется эффект минимизации количества состояний блока памяти.

3.5.2. Минимизация абстрактного автомата Мура

Минимизация автоматов Мура основана на тех же принципах, что и минимизация автоматов Мили. Для табличного описания эта процедура алгоритмизирована и состоит из трёх шагов.

Шаг 1 Распространение неопределённости выходов на таблицу переходов. Если в автомате Мура для некоторого состояния выходной сигнал не определён, то в это состояние он не может попасть под действием допустимого входного слова. Переход в таблице, соответствующий этому состоянию, исключается, а в остальных клетках таблицы переход в исключённое состояние заменяется специальным символом, например, прочерком.

Шаг 2 Исключение недостижимых состояний. Если нет ни одного слова, приводящего автомат в состояние a_i, отличающееся от начального, то такое состояние исключается вместе с соответствующими переходами таблицы переходов.

Пример выполнения двух начальных этапов минимизации автомата Мура S_{14} приведён в таблице 3.34 и таблице 3.35.

Таблица 3.34

Отмеченная таблица автомата Мура S_{14}

Вых	Сост	Вх	z_1	z_2	z_3
w_1	a_1		a_2	a_3	~~a_4~~
w_2	a_2		~~a_4~~	a_3	a_2
w_3	a_3		~~a_6~~	~~a_4~~	a_3
-	a_4		-	a_6	a_4
w_2	a_5		~~a_6~~	-	a_2
-	a_6		~~a_5~~	a_4	a_3

Недостижимое

Таблица 3.35

Отмеченная таблица минимизированного автомата Мура S_{15}

Вых	Сост	Вх	z_1	z_2	z_3
w_1	a_1		a_2	a_3	-
w_2	a_2		-	a_3	a_2
w_3	a_3		-	-	a_3

Шаг 3 Нахождение совместимых состояний автомата Мура. Состояния автомата Мура являются 0-совместимыми, если, не считая неопределённых отметок, они отмечены одинаковыми выходными сигналами. Состояния являются i-совместимыми для любого i=1;2;... , если они 0-совместимы и автомат, переходя из них, перерабатывает допустимые слова длиной i одинаково. Процедура расщепления классов обязательно закончится за ограниченное количество шагов и, следовательно, более нерасщепляющиеся классы образуют конечные классы совместимых состояний. После нахождения совместимых состояний автомата строится минимизированный нормализованный автомат Мура.

При табличном описании нормализованного автомата Мура, имеющего N_n классов совместимых состояний, переход $\delta(N_i, z_j)$ считается неопределённым, если для всех $a_i \in N_i$ переходы $\delta(a_i, z_j)$ неопределённы. Ес-

ли хотя бы для одного $a_i \in N_i$ переход $\delta(a_i, z_j)$ определён, то в таблице нормализованного автомата указывается именно этот класс N_k, в который происходит переход. У частичного автомата таких переходов возможно несколько. Каждое класс N_i отмечается выходным сигналом который соответствует всем состояниям этого класса.

Построенный таким образом нормализованный автомат является минимальным, если исходный автомат был полностью определённым. Если исходный автомат был частичным, то возможно, либо доопределение нормализованного автомата в процессе нахождения совместимых состояний, либо получение частичного нормализованного автомата. Доопределение и выбор минимального автомата для частичного нормализованного автомата выполняется путем перебора и оценки всех вариантов автоматов, построенных по описанию нормализованного автомата.

Более подробно рассмотрим применение методики минимизации автоматов Мура на примере полностью определённого автомата S_{16}.

Таблица 3.36

Отмеченная таблица переходов автомата Мура S_{14}

Вых	Сост / Вх	z_1	z_2	z_3
w_1	a_1	a_2	a_3	a_4
w_2	a_2	a_9	a_{10}	a_{11}
w_3	a_3	a_5	a_3	a_3
w_4	a_4	a_4	a_{15}	a_4
w_1	a_5	a_5	a_5	a_6
w_2	a_6	a_7	a_8	a_6
w_3	a_7	a_7	a_7	a_1
w_4	a_8	a_1	a_8	a_8
w_2	a_9	a_9	a_9	a_{19}
w_2	a_{10}	a_{10}	a_{10}	a_{19}
w_1	a_{11}	a_{11}	a_{11}	a_{12}
w_2	a_{12}	a_{13}	a_{14}	a_{12}
w_3	a_{13}	a_{13}	a_{13}	a_1
w_4	a_{14}	a_1	a_{14}	a_{14}
w_1	a_{15}	a_{15}	a_{15}	a_{16}
w_2	a_{16}	a_{17}	a_{18}	a_{16}
w_3	a_{17}	a_{17}	a_{17}	a_1
w_4	a_{18}	a_1	a_{18}	a_{18}
w_1	a_{19}	a_{19}	a_{19}	a_{12}

Классы единичной совместимости

$D_1(1)$	$z_1;z_2;z_3$
1	3;5;7

$D_2(1)$	$z_1;z_2;z_3$
5	2;2;4
11	2;2;4
15	2;2;4
19	2;2;4

$D_3(1)$	$z_1;z_2;z_3$
2	3;3;2
9	3;3;2
10	3;3;2

$D_4(1)$	$z_1;z_2;z_3$
6	6;8;4
12	6;8;4
16	6;8;4

$D_5(1)$	$z_1;z_2;z_3$
3	2;5;5

$D_6(1)$	$z_1;z_2;z_3$
7	6;6;1
13	6;6;1
17	6;6;1

$D_7(1)$	$z_1;z_2;z_3$
4	7;2;7

$D_8(1)$	$z_1;z_2;z_3$
8	1;8;8
14	1;8;8
18	1;8;8

Рис. 3.20

Таблица 3.37

Таблица переходов нормализованного минимизированного автомата Мили S_{17}

Вых	Сост / Вх	z_1	z_2	z_3
w_1	D_1	D_3	D_5	D_7
w_1	D_2	D_2	D_2	D_4
w_2	D_3	D_3	D_3	D_2
w_2	D_4	D_6	D_8	D_4
w_3	D_5	D_2	D_5	D_5
w_4	D_6	D_6	D_6	D_1
w_4	D_7	D_7	D_2	D_7
w_4	D_8	D_1	D_8	D_8

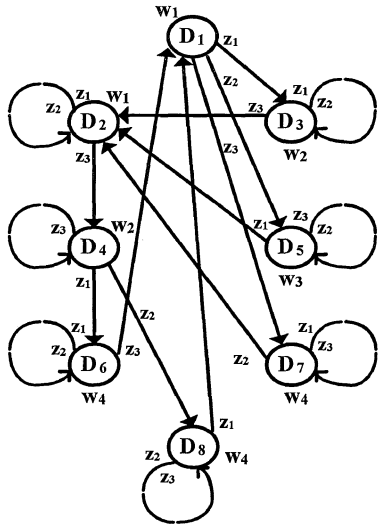


Рис. 3.21. Граф нормализованного автомата Мили S_{17}

3.6. Структурный синтез автоматов

Задачей этапа структурного синтеза является построение принципиальной схемы автомата из элементарных автоматов заданного типа. Элементарные автоматы подразделяются на два больших класса:

– элементарные автоматы памяти (запоминающие элементы);

– элементарные автоматы без памяти (элементарные комбинационные схемы или логические элементы).

Задача синтеза цифрового автомата имеет решение в том случае, если система элементарных автоматов является структурно полной.

Всякая система элементарных автоматов, содержащая элементарный автомат Мура (триггер) и какую-нибудь функционально полную систему логических элементов является структурно полной системой.

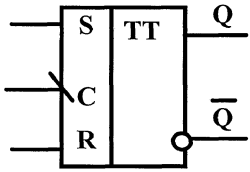
3.6.1. Элементарные автоматы памяти

Комбинационная схема с обратными связями, имеющая два устойчивых состояния и предназначенная для хранения одного бита информации, называется элементарным автоматом или триггером. Современные триггеры представляют собой сложные электронные устройства, содержащие десятки транзисторов и изготавливаемые в виде интегральных схем. Для синтеза цифровых автоматов триггеры рассматриваются, как элементы систем и важным является изучение его поведения в системе, а не внут-

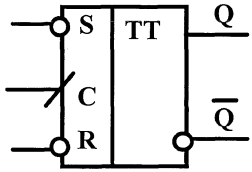
ренняя структура или принципиальная схема. В этом состоит системотехнический подход к изучению триггеров различных типов.

Для корректной работы цифровых автоматов необходимо исключить влияние переходных процессов в триггерах и комбинационных схемах на смену состояний цифрового автомата и на выходной сигнал. Это требование выполняется при использовании сложной многофазной системы синхронизирующих сигналов для блока памяти и выходной комбинационной схемы. Двухступенчатые синхронизированные триггеры, изготовленные по структуре M-S или O-B (M(aster) - S(lave) или O(сновной)-B(едомый)) имеют встроенную двухфазную систему синхронизации, поэтому только они могут использоваться для построения синхронизированных цифровых автоматов. Для примера рассмотрим работу нескольких типов триггеров.

Триггер типа RS. Название триггера происходит от аббревиатур двух английских слов Set (установить) и Reset (сбросить), которые, кроме этого ещё и соседние в латинском алфавите. Этот триггер имеет два входа – R и S и два выхода – Q и $\overline{Q}$. Обозначения различных вариантов исполнения этого триггера приведены на рис.3.22.



Синхронизированный спадом синхроимпульса RS - триггер с прямыми входами



Синхронизированный фронтом синхроимпульса RS - триггер с инверсными входами

Рис. 3.22. Обозначения RS-триггеров

Реакция триггера на входные сигналы зависит от того, в каком состоянии он находился до их подачи. Поведение триггера наглядно описывается таблицей переходов для соседних тактов автоматного времени, два варианта которой для RS триггера приведены на рис.3.23а) и рис.3.23б).

t	t+1
R S Q	Q^+
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	-
1 1 1	-

а)

R S	Q^+
0 0	Q
0 1	1
1 0	0
1 1	-

б)

$Q \to Q^+$	R S
0 0	♣ 0
0 1	0 1
1 0	1 0
1 1	0 ♣

♣ - любое значение

в)

Рис. 3.23. Варианты табличного описания работы RS-триггеров

Факт принадлежности сигналов различным моментам автоматного времени отражается индексами t и t+1 или Q и Q^+. Полную таблицу переходов триггера можно интерпретировать как таблицу истинности булевой функции трёх переменных. Минимизировав эту частично определённую функцию по диаграмме Вейча, получим характеристическое уравнение RS триггера

$$Q^+ = S \vee \overline{R}Q; \qquad (37)$$

$$R\&S \equiv 0.$$

На рис.3.23в) работа триггера описывается матрицей переходов, в которой указывается какие наборы управляющих сигналов в текущий момент автоматного времени следует подать на входы триггера, чтобы в следующий момент автоматного времени он перешёл из текущего указанного состояния в новое указанное состояние. Матрица переходов строится по таблице переходов для триггера.

Триггер типа JK. Триггер типа JK отличается от RS триггера наличием определённости при одновременной подаче единиц на J и K входы. При J=K=1 состояние триггера изменяется на обратное состоянию в предыдущем такте автоматного времени. В остальных ситуациях по управлению вход K эквивалентен входу R, а вход J эквивалентен входу S. Таблицы переходов и матрица переходов JK триггера приведены на рис.3.24.

t	t+1
K J Q	Q^+
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	1
1 1 1	0

а)

K J	Q^+
0 0	Q
0 1	1
1 0	0
1 1	$\overline{Q}$

б)

$Q \to Q^+$	K J
0 0	♣ 0
0 1	♣ 1
1 0	1 ♣
1 1	0 ♣

♣ - любое значение

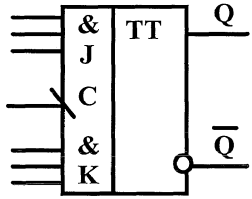
в)

Рис. 3.24. Варианты табличного описания работы JK-триггеров

Характеристическое уравнение JK триггера, заданное таблицей переходов на рис.3.24а), после минимизации с помощью диаграммы Вейча, имеет вид:

$$Q^+ = \overline{K}Q \vee J\overline{Q}; \qquad (38)$$

Условное обозначение JK триггера с встроенными конъюнкторами на входах (микросхема 155ТВ1) приведено на рис.3.25.



Синхронизированный спадом синхроимпульса JK - триггер «с встроенной логикой»

Рис. 3.25. Обозначения JK-триггеров

JK триггер является универсальным, так как может работать в режимах, соответствующих работе RS триггера, T триггера и D триггера.

Триггер типа D. Название происходит от английского термина «Delay» (задержка). Триггер имеет всего один вход (D) и на выходе он повторяет сигнал на входе D, существовавший в предыдущем такте автоматного времени.

Поскольку в пределах периода синхроимпульсов входной сигнал появляется в произвольный момент времени, то на выход входной сигнал проходит с произвольной задержкой, не превышающей длительность периода синхросигнала. Это свойство D триггера объясняет и его название.

Таблицы переходов и матрица переходов D триггера приведены на рис.3.26.

t		t+1
D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

а)

D	Q^+
0	0
1	1

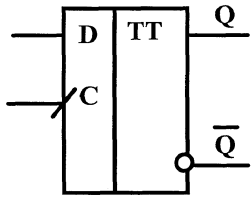
б)

$Q \rightarrow Q^+$		D
0	0	0
0	1	1
1	0	0
1	1	1

в)

Рис. 3.26. Варианты табличного описания работы D-триггеров

Условное обозначение двухступенчатого D триггера приведено на рис.3.27.



Синхронизированный фронтом синхроимпульса D - триггер

Рис. 3.27. Обозначения D-триггеров

Характеристическое уравнение D триггера, заданное таблицей переходов на рис.3.26а), имеет вид:

$$Q^+ = D; \qquad (39)$$

Триггер типа Т. Триггеры этого типа выпускаются и как самостоятельные устройства, но чаще для работы в специальном режиме Т триггера используются универсальные триггеры RS, JK и D типов (рис.3.28).

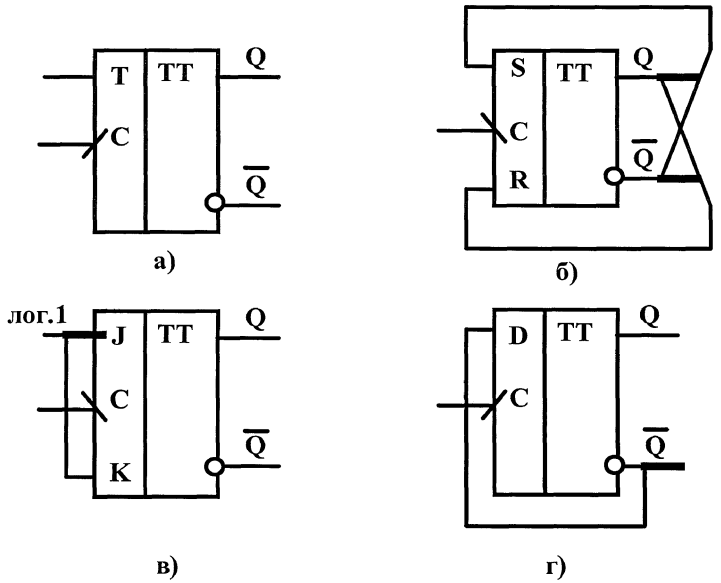


Рис. 3.28. Варианты построения схем Т-триггеров

По рис.3.28 видно, что Т триггеры на основе RS и D триггеров являются несинхронизированными. Варианты табличного описания работы Т триггера приведены на рис.3.29.

t	t+1
T Q	Q^+
0 0	0
0 1	1
1 0	1
1 1	0

а)

T	Q^+
0	Q
1	$\overline{Q}$

б)

$Q \to Q^+$	T
0 0	0
0 1	1
1 0	1
1 1	0

в)

Рис. 3.29. Варианты табличного описания работы Т-триггеров

Характеристическое уравнение Т триггера имеет вид:

$$Q^+ = \overline{T}Q \vee T\overline{Q} = T \oplus Q; \qquad (40)$$

По этому уравнению видно, что Т триггер выполняет сложение по модулю 2 входной логической переменной в текущем такте автоматного времени Т и Q – запомненной в предыдущем такте автоматного времени выходной логической переменной.

Рассмотренные примеры системного описания элементной базы для построения блоков памяти цифровых автоматов позволяют вслед за этапом абстрактного синтеза автомата, заканчивающегося минимизацией числа его состояний, выполнить этап структурного синтеза, целью которого является построение схемы, реализующей автомат из логических элементов и элементов памяти заданного типа.

3.6.2. Синхронизация в цифровых автоматах

Смена состояний в синхронизированных автоматах происходит в определённые моменты времени, задаваемые по цепям синхронизации внешним тактовым генератором. Изменение состояний в реальных цифровых автоматах сопровождаются переходными процессами, имеющими вполне определённые длительности для выбранной элементной базы. Выходные сигналы по цепям обратной связи поступают на вход цифрового автомата и далее на вход его блока памяти. Из-за неопределённости сигналов обратной связи во время протекания переходных процессов неопределёнными будут и сигналы возбуждения блока памяти. По этой причине предусматриваются специальные меры, гарантирующие переход автомата в нужное состояние. В синхронизированных цифровых автоматах устойчивость автомата обеспечивается специальными цепями синхронизации, разрывающими цепи обратной связи во время протекания переходных процессов. Такая синхронизация требует обеспечения определённых запасов времени на протекание переходных процессов и, следовательно, уменьшает быстродействие цифровых автоматов и требует выполнения определённых временных соотношений при смене значений входных сигналов.

Схема цифрового автомата, построенная из элементарных автоматов является корректно построенной, если в каждом её узле неоднозначность сигналов в любой существенный интервал времени отсутствует. Существенным интервалом времени является интервал, в течение которого информация считывается с какого либо узла этой схемы. Корректная схема автомата Мура изображена на рис.46. Комбинационная схема $КС_{вх}$ служит для выработки кода, соответствующего новому состоянию автомата в соответствии с функцией переходов

$$a_s(t+1)=\delta(a_m(t), z_f(t)),$$

схема $КС_{вых}$ – для выработки сигналов, соответствующих функции выходов

$$w_g=\lambda(a_m(t)).$$

Блок памяти состоит из двух одинаковых блоков БП1 и БП2. Передача сигналов от блока к блоку осуществляется через конъюнкторы, управляемые тактовыми сигналами τ_1 и τ_2. Код, соответствующий новому

состоянию, формируется $КС_{вх}$ и при появлении импульса τ_1 устанавливает в нужное состояние БП1. В интервале времени τ_1 БП2 сохраняет код своего прежнего состояния на входах КС1. В течение времени, выделенного цветом, изменение входных сигналов запрещено, поэтому неопределённость на выходах КС1 не возникает. Выполнение ограничений на временные соотношения при формировании входных сигналов обеспечивается в источнике входных сигналов X.

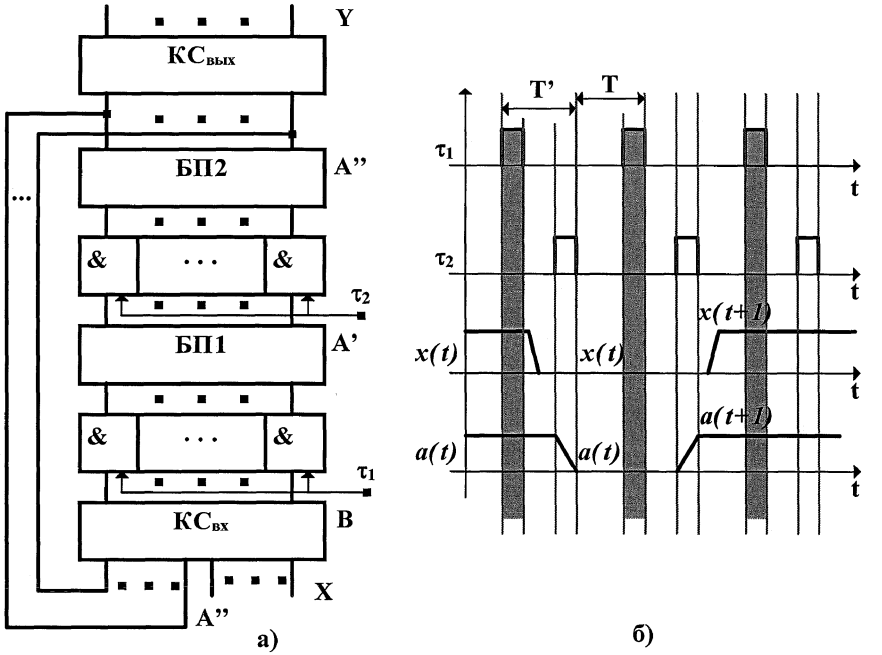


Рис. 3.30. Корректная схема цифрового автомата Мура с двухтактной памятью

Импульсом τ_2 информация переносится в БП2 и этим заканчивается такт работы цифрового автомата. На смену состояния цифрового автомата, таким образом затрачивается время T’. Поскольку у автомата Мура функции выходов не зависят от входных сигналов, то порядок их смены может быть любым, лишь бы на интервале времени τ_1 входные сигналы были неизменны.

Таким образом, корректность работы цифрового автомата Мура обеспечивается введением двухтактного синхронизированного блока памяти. Логика работы двухтактной памяти практически реализуется в двухступенчатых триггерах структуры О-В (или M-S), имеющих встроенную двухтактную систему синхронизации.

Двухтактная синхронизированная память обеспечивает корректность и автомата Мили. Однако только одного этого для него недостаточно, так как одним из аргументов функции выходов автомата Мили является код входного сигнала *x(t)*, то есть

$$w_g=\lambda(a_m(t), z_f(t)).$$

Для синхронизированного цифрового автомата недопустим случайный характер смены кода выходного сигнала в соответствии с изменениями кода входного сигнала, но накладывать жёсткие ограничения на временные соотношения во входном сигнале также недопустимо. Это противоречие разрешается при введении стробирования выходной комбинационной схемы $КС_{вых}$ синхросигналом τ_1. Выходные сигналы цифрового автомата Мили при этом становятся импульсными. Для цифрового автомата Мура выходные сигналы считаются потенциальными, так как нет необходимости в стробировании выходной комбинационной схемы.

Структурная схема корректного цифрового автомата Мили и временная диаграмма его работы приведены на рис.3.31.

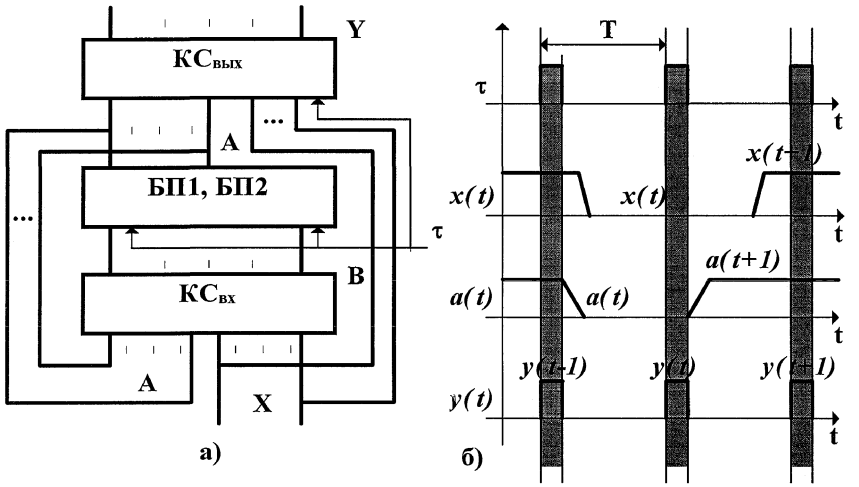


Рис. 3.31. Корректная схема цифрового автомата Мили с двухтактной памятью и стробированием выходной комбинационной схемы

По временной диаграмме рис.47б) видно, что смена состояния блока памяти должна происходить по спаду синхроимпульса. В противном случае
в цифровом автомате Мили будет реализована функция выходов, описываемая уравнением:

$$w_g(t)=\lambda(a_m(t+1), z_f(t)).$$

3.7. Структурный синтез цифровых автоматов по таблицам

Если автомат имеет M состояний, то для двоичного структурного алфавита количество триггеров в блоке памяти этого автомата

$$n=]\log_2 M[\ ,$$

где]...[– ближайшее большее целое число.

Если в каждую клетку таблицы переходов и выходов записать двоичный код, соответствующий размещённым там состояниям или выходным сигналам цифрового автомата, то таким образом получаются кодированные таблицы переходов и выходов.

Кодированная таблица выходов является табличным описанием системы булевых функций, реализуемых схемой $КС_{вых}$. Кодированная таблица переходов только после переработки с использованием матрицы переходов для заданного типа триггеров будет называться кодированной таблицей возбуждений и соответствовать описанию комбинационной схемы $КС_{вх}$.

Таким образом, задача синтеза состоит в определении по таблицам функций выхода и функций возбуждения триггеров заданного типа в блоке памяти, минимизации их для выбранной элементной базы и схемной реализации в функционально полном базисе элементов.

Кодированные таблицы переходов и выходов изображаются в наиболее удобном расположении, либо поперёк страницы, либо вдоль в том случае, если размерность таблицы велика.

Рассмотрим пример синтеза цифрового автомата Мили S_{18}, заданного таблицей 3.37 переходов и таблицей 3.38 выходов. Структурный алфавит для выходных сигналов и состояний – двоичный. Для кодирования трёх входных и трёх выходных сигналов двоичными кодами достаточно двухразрядного кода, а три состояния можно реализовать двумя триггерами. Таким образом, структурный автомат будет иметь две входных и две выходных линии, а память его будет состоять из двух триггеров.

Таблица 3.37

Таблица переходов автомата Мили S_{18}

Вх / Сост	a_1	a_2	a_3
z_1	a_2	a_1	a_1
z_2	a_1	a_3	a_3
z_3	a_1	a_1	a_1

Таблица 3.38

Таблица выходов автомата Мили S_{18}

Вх / Сост	a_1	a_2	a_3
z_1	w_1	w_1	w_2
z_2	w_2	w_1	w_1
z_3	w_3	w_3	w_3

Очевидно, что при кодировании переходов и выходов можно придерживаться двух принципов описания булевых функций. Если желательно получить табличное описание функций выходов с наименьшим количеством единичных значений, то для кодирования часто встречающихся в таблице выходов сигналов следует использовать коды с максимально воз-

можным количеством нулей в коде, а для кодирования следующих по количеству ссылок в таблице выходов сигналов использовать коды с увеличивающимся количеством единиц в кодовых комбинациях. Для кодирования состояний блока памяти на D триггерах также можно использовать этот принцип кодирования, поскольку таблица возбуждений для них совпадает с таблицей переходов. Рекомендовать этот принцип для всеобщего применения при синтезе автоматов нельзя, так как при минимизации булевых функций возможно получение более простых результирующих форм представления функций, имеющих более сложную запись в СДНФ. Этот принцип можно использовать только в том случае, если ФАЛ выходов и ФАЛ возбуждений для D триггеров не подлежат минимизации, поскольку реализуются на мультиплексорах, дешифраторах или постоянных запоминающих устройствах.

Второй принцип кодирования соответствует противоположному подходу и ориентирован на возможность получения значительных упрощений ФАЛ в результате минимизации. Для кодирования выходных сигналов с максимальным количеством ссылок в таблице выходов используется код с максимальным количеством единиц, а для кодирования следующих по количеству ссылок в таблице выходных сигналов использовать коды с уменьшающимся количеством единиц в кодовых комбинациях. Этот принцип также без оговорок применим для кодирования состояний блока памяти на D триггерах для случая применения элементной базы, требующей минимизации для своей реализации. Минимальный по материальным затратам вариант кодирования выбирается из конечных результатов при использовании всевозможных вариантов кодирования.

Для второго принципа кодирования по изложенной выше методике построим кодированные таблицы переходов, выходов и возбуждений для JK триггера с использованием матрицы переходов на рис.40в). Кодирование входных, выходных сигналов и состояний автомата выполним следующим образом:

Вх	Код	x_1x_0	
z_1		00	$\bar{x}_1\bar{x}_0$
z_2		01	$\bar{x}_1x_0$
z_3		10	$x_1\bar{x}_0$
-		11	x_1x_0

Вых	Код	y_1y_0	
w_1		11	y_1y_0
w_2		01	$\bar{y}_1y_0$
w_3		10	$y_1\bar{y}_0$
-		00	$\bar{y}_1\bar{y}_0$

Сост	Код	Q_1Q_0	
a_1		11	Q_1Q_0
a_2		01	$\bar{Q}_1Q_0$
a_3		10	$Q_1\bar{Q}_0$
-		00	$\bar{Q}_1\bar{Q}_0$

Рис.3.32. Таблицы кодирования входных сигналов, выходных сигналов и состояний (по второму варианту)

В таблице выходов выходной сигнал w_1 встречается 4 раза, выходной сигнал w_3 – 3 раза, выходной сигнал w_2 – 2 раза. В таблице переходов состояние a_1 встречается 6 раз, состояние a_3 – 2 раза, состояние a_2 – 1 раз. В соответствии с количеством этих ссылок заполнены таблицы кодирования на рис.3.32.

Кодированная таблица переходов

Сост \ Вх	00	01	10
Q_1Q_0	Q_1Q_0	Q_1Q_0	Q_1Q_0
11	01	11	11
01	11	10	11
10	11	10	11

Кодированная таблица выходов

Сост \ Вх	00	01	10
Q_1Q_0	y_1y_0	y_1y_0	y_1y_0
11	11	01	10
01	11	11	10
10	01	11	10

Матрица переходов JK триггера

$Q \to Q^+$	K J
0 0	♣ 0
0 1	♣ 1
1 0	1 ♣
1 1	0 ♣

♣ - любое значение

Кодированные таблицы возбуждений блока памяти цифрового автомата

Сост \ Вх	00	01	10
Q_1Q_0	J_1J_0	J_1J_0	J_1J_0
11	♣♣	♣♣	♣♣
01	1♣	1♣	1♣
10	♣1	♣0	♣1

Сост \ Вх	00	01	10
Q_1Q_0	K_1K_0	K_1K_0	K_1K_0
11	10	00	00
01	♣0	♣1	♣0
10	0♣	0♣	0♣

Запрещённые наборы переменных $x_1x_0Q_1Q_0$
1100; 1101; 1110; 1111; 0000; 0100; 1000.
Где 11♣♣ – запрещённый код входных сигналов,
♣♣00 – запрещённый код состояний.

Рис.3.33. Кодированные таблицы переходов, выходов и возбуждений блока памяти на JK триггерах

Выполним конкатенацию кодов входных сигналов и кодов состояний по порядку следования переменных $x_1x_0Q_1Q_0$ и заполним таблицу истинности для функций выхода и возбуждений. В таблице на запрещённых наборах входных сигналов и состояний значения функций не определены и обозначены символом красного цвета $♣^к$.

Таблица истинности функций выходов и возбуждений

x_1	x_0	Q_1	Q_0	y_1	y_0	J_1	K_1	J_0	K_0
0	0	0	0	♣к	♣к	♣к	♣к	♣к	♣к
0	0	0	1	1	1	1	♣	♣	0
0	0	1	0	0	1	♣	0	1	♣
0	0	1	1	1	1	♣	1	♣	0
0	1	0	0	♣к	♣к	♣к	♣к	♣к	♣к
0	1	0	1	1	1	1	♣	♣	1
0	1	1	0	1	1	♣	0	0	♣
0	1	1	1	0	1	♣	0	♣	0
1	0	0	0	♣к	♣к	♣к	♣к	♣к	♣к
1	0	0	1	1	0	1	♣	♣	0
1	0	1	0	1	0	♣	0	1	♣
1	0	1	1	1	0	♣	0	♣	0
1	1	0	0	♣к	♣к	♣к	♣к	♣к	♣к
1	1	0	1	♣к	♣к	♣к	♣к	♣к	♣к
1	1	1	0	♣к	♣к	♣к	♣к	♣к	♣к
1	1	1	1	♣к	♣к	♣к	♣к	♣к	♣к

Рис.3.34. Таблица истинности булевых функций входной и выходной схем цифрового автомата Мили S_{18}

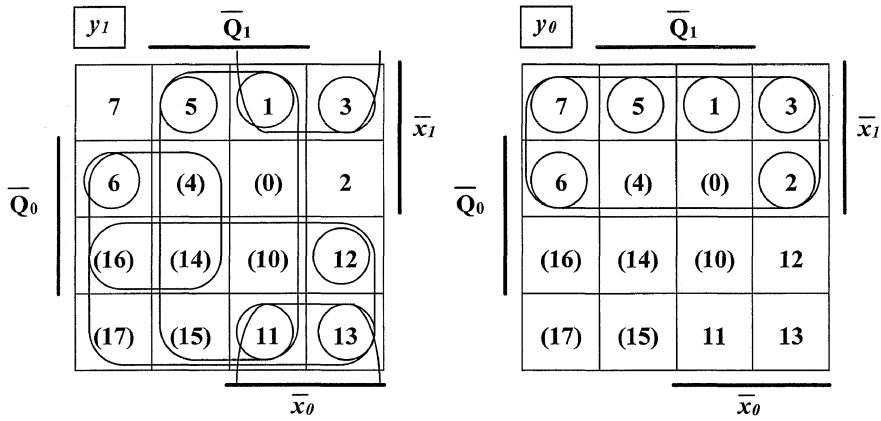


Рис.3.35. Минимизация функций выходов

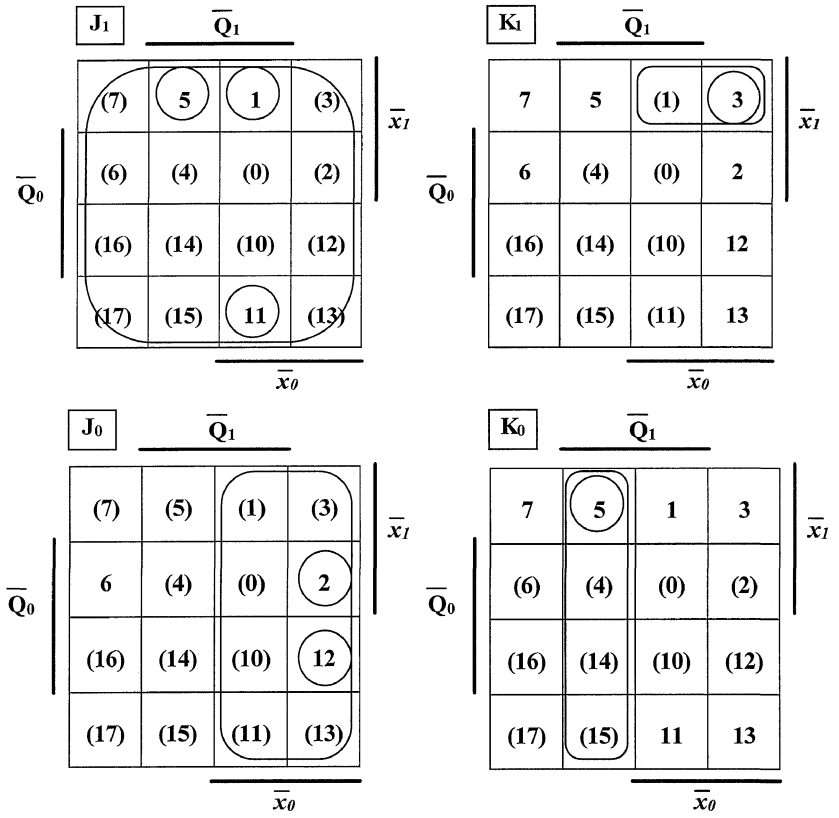


Рис.3.36. Минимизация функций возбуждения

Минимизированные функции выходов в ДНФ:

$$\left.\begin{aligned} y_1 &= x_1 \vee \overline{Q}_1 \vee x_0\overline{Q}_0 \vee \overline{x}_0 Q_0; \\ y_0 &= \overline{x}_1; \end{aligned}\right\} \text{ А}$$

Минимизированные функции возбуждений в ДНФ:

$$\left.\begin{aligned} J_1 &= 1; \quad K_1 = \overline{x}_1\overline{x}_0 Q_0; \\ J_0 &= \overline{x}_0; \quad K_0 = x_0\overline{Q}_1; \end{aligned}\right\} \text{ А}$$

Для первого принципа кодирования по изложенной выше методике построим кодированные таблицы переходов, выходов и возбуждений для JK триггера с использованием матрицы переходов на рис.3.24в). Кодиро-

вание входных, выходных сигналов и состояний автомата выполним следующим образом:

Вх	Код	x_1x_0	
z_1		00	$\bar{x}_1\bar{x}_0$
z_2		01	$\bar{x}_1x_0$
z_3		10	$x_1\bar{x}_0$
-		11	x_1x_0

Вых	Код	y_1y_0	
w_1		00	$\bar{y}_1\bar{y}_0$
w_2		01	$\bar{y}_1y_0$
w_3		10	$y_1\bar{y}_0$
-		11	y_1y_0

Сост	Код	Q_1Q_0	
a_1		00	$\bar{Q}_1\bar{Q}_0$
a_2		01	$\bar{Q}_1Q_0$
a_3		10	$Q_1\bar{Q}_0$
-		11	Q_1Q_0

Запрещённые коды $x_1x_0Q_1Q_0$
1100; 1101; 1110; 1111; 0011; 0111; 1011.
Где 11♣♣ – запрещённый код входных сигналов,
♣♣11 – запрещённый код состояний

Рис.3.37. Таблицы кодирования входных сигналов, выходных сигналов и состояний (по первому варианту)

В кодированных таблицах переходов, выходов и возбуждений при таком кодировании значительно меньше единичных значений функций выходов и возбуждений для D триггеров. По кодированным таблицам возбуждений для двух вариантов кодирования видно, что по общему количеству единиц и неопределённых значений они совершенно эквивалентны.

Кодированная таблица переходов

Сост \ Вх	00	01	10
Q_1Q_0	Q_1Q_0	Q_1Q_0	Q_1Q_0
00	01	00	00
01	00	10	00
10	00	10	00

Кодированная таблица выходов

Сост \ Вх	00	01	10
Q_1Q_0	y_1y_0	y_1y_0	y_1y_0
00	00	01	10
01	00	00	10
10	01	00	10

Матрица переходов JK триггера

$Q \to Q^+$	K J
0 0	♣ 0
0 1	♣ 1
1 0	1 ♣
1 1	0 ♣

♣ - любое значение

Кодированные таблицы возбуждений блока памяти цифрового автомата

Сост \ Вх	00	01	10
Q_1Q_0	J_1J_0	J_1J_0	J_1J_0
00	01	00	00
01	0♣	1♣	0♣
10	♣0	♣0	♣0

Сост \ Вх	00	01	10
Q_1Q_0	K_1K_0	K_1K_0	K_1K_0
00	♣♣	♣♣	♣♣
01	♣1	♣1	♣1
10	1♣	0♣	1♣

Рис.3.38. Кодированные таблицы переходов, выходов и возбуждений

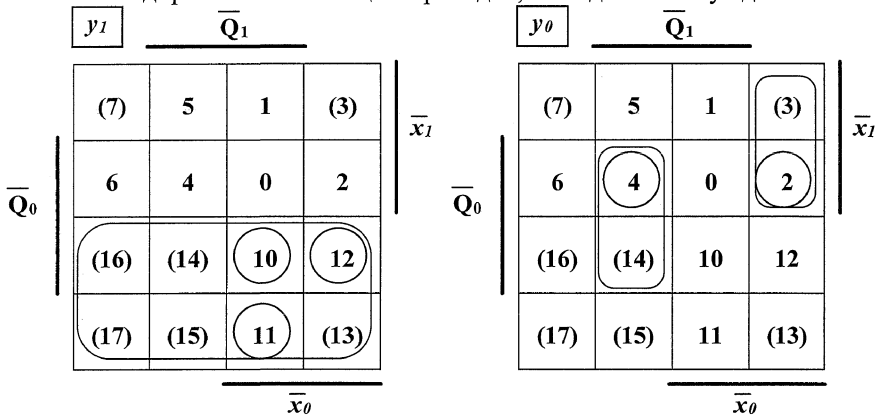


Рис. 3.39 Минимизация функций выхода

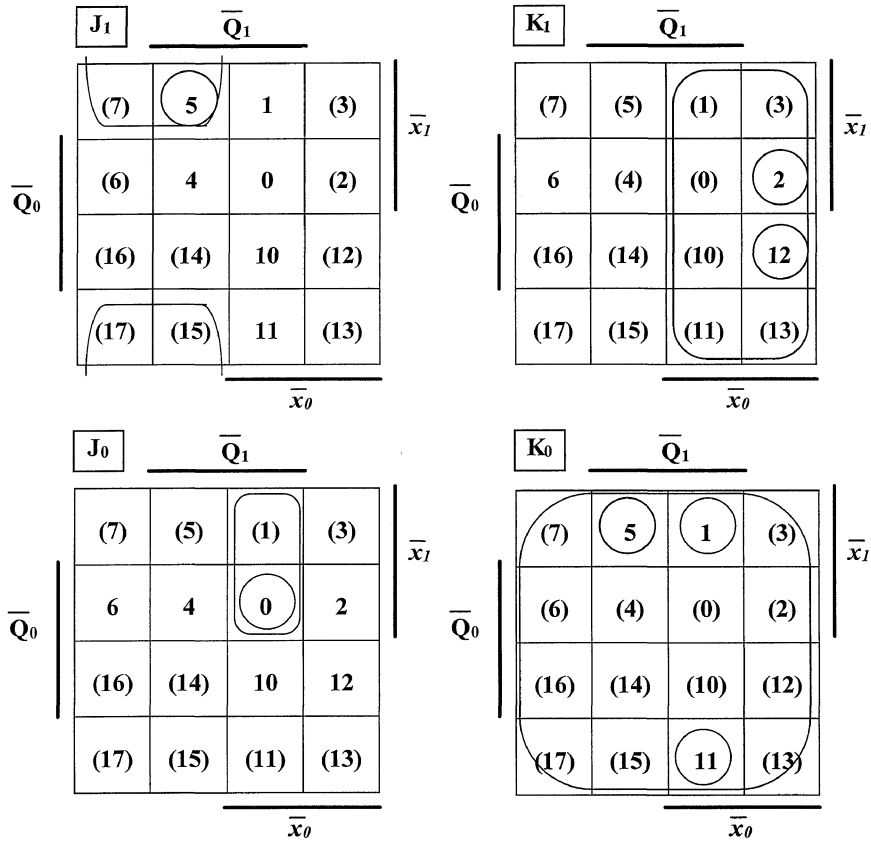


Рис.3.40. Минимизаций функций возбуждения JK триггеров блока памяти

Сопоставим результаты синтеза комбинационных схем цифрового автомата по двум вариантам кодирования выходов и состояний (формулы А и Б). Рассмотрение минимизированных функций выходов и возбуждений не позволяет сделать вывод о преимуществе какого-то из способов кодирования. Очевидно, что для получения оптимального варианта кодирования нужно сопоставить результаты минимизации комбинационных схем при использовании всевозможных вариантов кодирования.

Минимизированные функции выходов в ДНФ:

$$\left.\begin{aligned} y_1 &= x_1 \vee \overline{Q}_1 \vee x_0\overline{Q}_0 \vee \overline{x}_0 Q_0; \\ y_0 &= \overline{x}_1; \end{aligned}\right\} \text{А}$$

$$\left.\begin{aligned} y_1 &= x_1; \\ y_0 &= \overline{x}_1\overline{x}_0 Q_1 \vee x_0\overline{Q}_1\overline{Q}_0; \end{aligned}\right\} \text{Б}$$

Минимизированные функции возбуждений в ДНФ:

$$\left.\begin{aligned} J_1 &= 1; \quad K_1 = \overline{x}_1\overline{x}_0 Q_0; \\ J_0 &= \overline{x}_0; \quad K_0 = x_0\overline{Q}_1; \end{aligned}\right\} \text{А}$$

$$\left.\begin{aligned} J_1 &= x_0 Q_0; \quad K_1 = \overline{x}_0; \\ J_0 &= \overline{x}_1\overline{x}_0\overline{Q}_1; \quad K_0 = 1; \end{aligned}\right\} \text{Б}$$

На рис.57 приведена принципиальная схема цифрового автомата S_{18} с использованием кодирования по первому варианту, который имеет явные преимущества при использовании элементной базы, не требующей минимизации булевых функций выходов и возбуждений.

В формулах А и Б функции возбуждения JK триггеров блока памяти не содержат дизъюнкций, а количество переменных в конъюнкции не больше трёх. Для блока памяти следует использовать JK триггеры с встроенной логикой (трёхвходовые конъюнкторы в триггерах типа 155ТВ1). Такие схемы дают большую экономию и повышенную надёжность. Функция K_0 «константа 1» задаётся подключением соответствующего входа через резистор сопротивлением 1 кОм к источнику питания схемы (для ТТЛ схем это напряжение равно +5В). Для соединения элементов схемы между собой использован «жгут». Соединены между собой проводники, имеющие одинаковую нумерацию входов в «жгут» и выходов из «жгута».

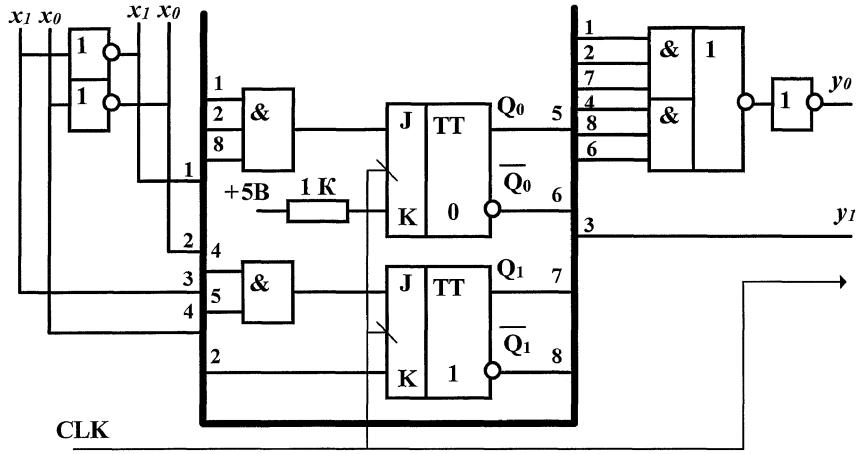


Рис.3.41. Принципиальная схема автомата Мили S_{18}

При построении принципиальной схемы цифрового автомата принимаются меры против подачи на вход запрещённых кодов входного сигнала и случайного попадания автомата в состояния, соответствующие запрещённым кодам. Задача исключения нахождения в состояниях, соответствующих запрещённым кодам, и аварийной реакции на поступление запрещённых кодов входных сигналов решается двумя способами.

Первый способ. Во входную строку (столбец) таблицы переходов включаются все состояния, кодируемые запрещёнными кодами, а все сигналы, кодируемые запрещёнными кодами, включаются в соответствующий входной столбец (строку) таблицы переходов. Функции переходов, соответствующие запрещённым кодам состояний или запрещённым кодам входных сигналов, определяются как переходы в установленное (обычно начальное) состояние. Для полностью определённого таким образом кодированного цифрового автомата обычным образом синтезируются комбинационные схемы входа и выхода.

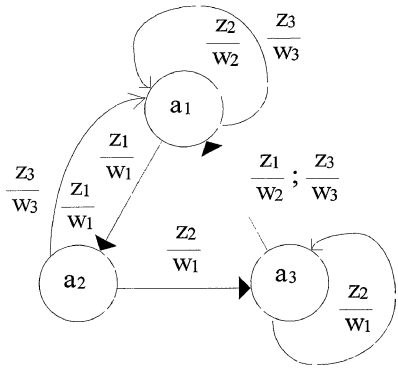
Второй способ. Все триггеры, предназначенные для применения в составе блоков памяти цифровых автоматов, имеют отдельные несинхронизированные входы сброса или установки. Для всех запрещённых кодов входных сигналов и состояний выполняется дешифрирование и объединение выходных сигналов дешифраторов на входы сброса или установки триггеров в какое-то состояние с разрешённой кодировкой (обычно начальное). Таким образом синтезируется отдельная специальная система сброса и начальной установки цифрового автомата.

В обоих случаях усложняется принципиальная схема цифрового автомата. Иногда проектировщики ограничиваются упрощенной схемой начального сброса цифрового автомата при включении питающего напря-

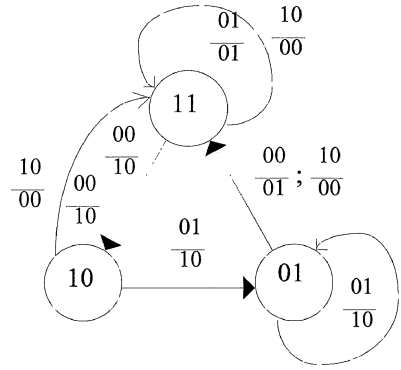
жения. Для надёжной работы цифрового автомата такой схемы недостаточно.

3.8. Структурный синтез цифрового автомата по графу

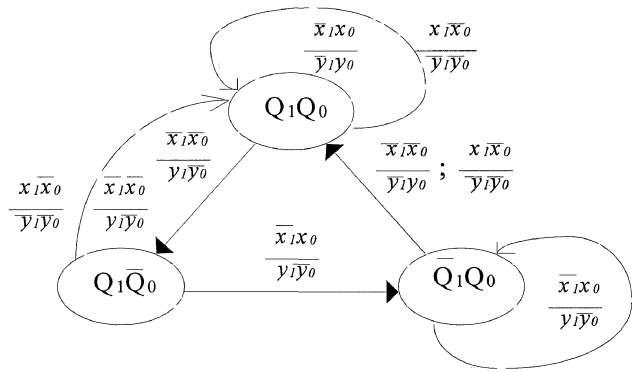
Табличный и графический способы задания автоматов эквивалентны, поэтому граф автомата содержит всю необходимую информацию о функциях выходов и функциях переходов. На граф кодированного цифрового автомата следует нанести все необходимые данные по функциям возбуждения триггеров заданного типа. Однако дальнейшее использование информации, заданной в виде разметки графа цифрового автомата, выполняется с использованием табличного или аналитического представления ФАЛ. Таким образом, синтез цифрового автомата только по графу обычно не делается и этот метод синтеза цифрового автомата является гораздо менее распространённым, чем синтез цифрового автомата с использованием таблиц. На рис.3.42 представлен пример различных видов разметки графа цифрового автомата Мили S_{18}.



а) граф абстрактного автомата Мили S_{18}



б) граф структурного кодированного автомата Мили S_{18}



в) граф с аналитической разметкой для структурного кодированного автомата Мили S_{18}

Рис.3.42. Различные виды разметки графа автомата Мили S_{18}

Функцию выходов по графу рис.3.42в) получим следующим образом: для каждой дуги графа, выходящей из вершины и помеченной выходным сигналом y_i, образуем конъюнкцию из членов, обозначающих эту вершину, и входного сигнала, обеспечивающего движение по этой дуге (то есть выполним конкатенацию этих элементов). Дизъюнкция построенных конъюнкций (конкатенаций) и даёт булеву функцию выхода y_i. Вершины обходятся последовательно, начиная с начальной.

Несколько сложнее обстоит дело с получением функций возбуждения триггеров блока памяти. Функции возбуждения получаются по размет-

ке дуг графа синтезируемого цифрового автомата, которая производится с использованием матриц (таблиц) переходов для заданного типа триггеров в блоке памяти.

Триггеры типа RS. Пример разметки дуг фрагмента графа автомата приведён на рис.3.43.

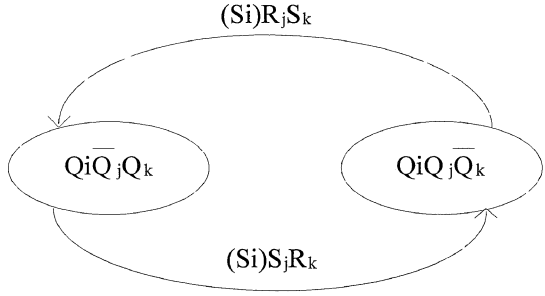


Рис. 3.43. Пример разметки дуг графа для RS триггеров в блоке памяти

При разметке переходы триггера $\overline{Q} \to \overline{Q}, \overline{Q} \to Q, Q \to \overline{Q}, Q \to Q$ интерпретируются как переходы 0→0, 0→1, 1→0, 1→1 соответственно. Если на каком-то из переходов функция возбуждения имеет единичное значение, то дуга графа помечается символом этой функции с соответствующим индексом. Если же на каком-то из переходов функция возбуждения не определена, то дуга помечается символом функции возбуждения заключённым в скобки и имеющим соответствующий индекс.

Триггеры типа T. На рис.3.44 приведён пример разметки дуг фрагмента графа цифрового автомата для T триггеров в блоке памяти.

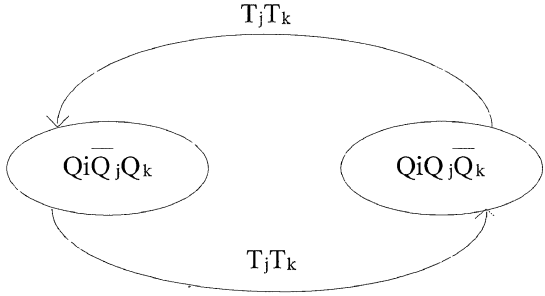


Рис. 3.44. Пример разметки дуг графа для T триггеров в блоке памяти

Триггеры типа D. На рис.3.45 приведён пример разметки дуг фрагмента графа цифрового автомата для D триггеров в блоке памяти.

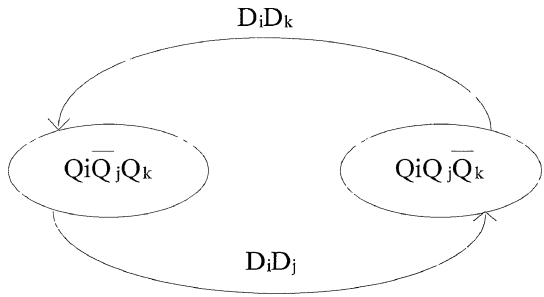


Рис. 3.45. Пример разметки дуг графа для D триггеров в блоке памяти

Триггеры типа JK. На рис.3.46 приведён пример разметки дуг фрагмента графа цифрового автомата для JK триггеров в блоке памяти.

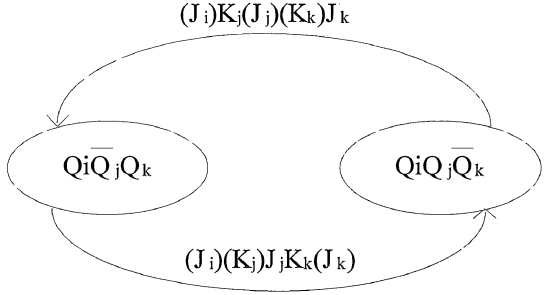


Рис. 3.46. Пример разметки дуг графа для JK триггеров в блоке памяти

Для каждой помеченной дуги образуется конкатенация кода состояния, из которого выходит дуга, и кода входного сигнала, вызвавшего переход по этой дуге. Соответствующие конъюнкции объединяются в дизъюнкцию и, таким образом, получаются функции возбуждения триггеров блока памяти цифрового автомата. В случае необходимости эти функции подвергаются минимизации любым из ранее рассмотренных способов.

Для структурного синтеза цифровых автоматов предпочтительнее использовать табличные методы, поскольку они выполняются в более жёсткой форме, чем структурный синтез по графу, который требует глубокого сосредоточения внимания на процедуре синтеза и проверке её результатов. Количество ошибок при использовании метода структурного синтеза по графу превосходит количество ошибок при использовании таблично-

го метода структурного синтеза при равных прочих условиях выполнения процедур синтеза.

Несмотря на обоснованное утверждение о равноценности двух рассмотренных методов структурного синтеза, в учебной литературе обычно используется структурный синтез цифровых автоматов по таблицам. В учебных заданиях студенты приобретают навыки выполнения структурного синтеза цифровых автоматов по таблицам и в дальнейшем, в случае необходимости, используют его и в практической инженерной работе.

4. МЕТОДИКА МОДЕЛИРОВАНИЯ В VISSIM ПРЕОБРАЗОВАТЕЛЯ СИГНАЛА

4.1. Моделирование задающей входной последовательности

Входная последовательность записывается в файл с расширением txt или doc в виде вектор-столбца. На рисунке 4.1 представлен пример файла с входной последовательностью двоичного сигнала.

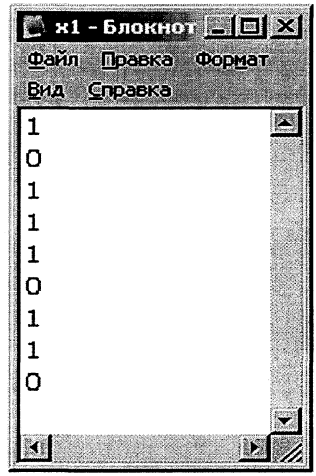


Рис. 4.1

Данный файл под именем x1.txt подсоединяется к модели автомата в Vissime с помощью блока import (рис. 4.2).

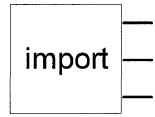


Рис. 4.2

Свойства блока import представлены на рис. 4.3.

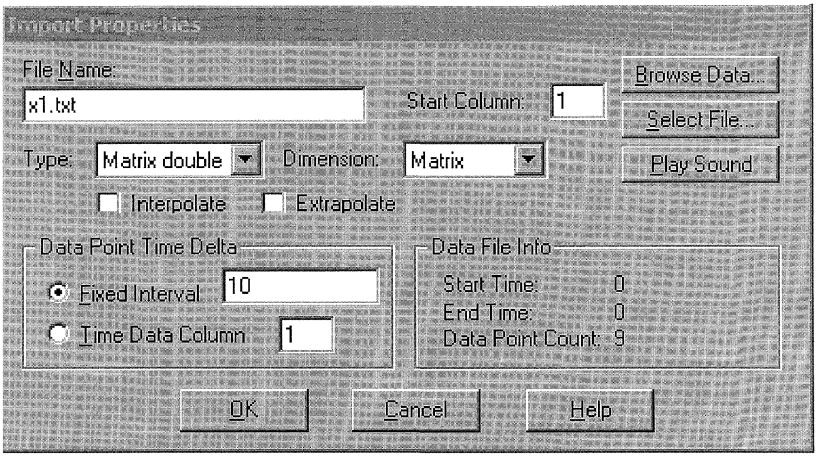


Рис. 4.3

После импортирования файла с входной последовательностью получаем вектор-столбец, представленный на рис. 4.4.

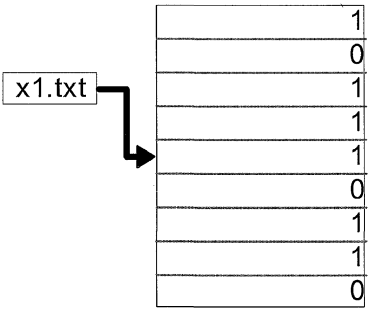


Рис. 4.4

4.2. Преобразование вектора входного сигнала во временную последовательность

Преобразование вектора во временную последовательность осуществляется путем использования блока отсчета времени в реальном масштабе (рис. 4.7) и блока работы с матрицами – блок индексации для определения одного из элементов матрицы (рис. 4.8).

Общая схема преобразования сигнала представлена на рис. 4.5.

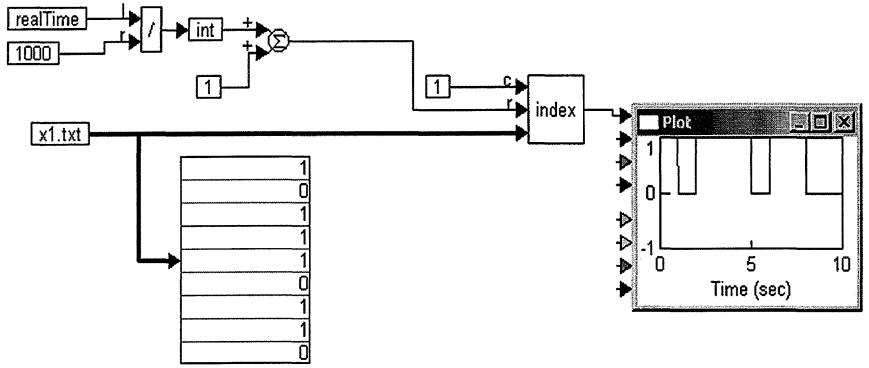


Рис. 4.5

На рис. 4.6 представлен блок пересчета времени из миллисекунд в секунды. Отсчет начинаем с первой секунды.

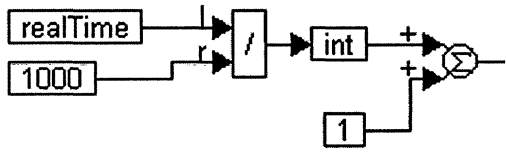


Рис. 4.6

Рис. 4.7. Блок отсчета времени в реальном масштабе (в миллисекундах)

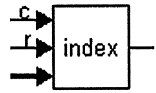


Рис. 4.8. Блок индексации для определения одного из элементов матрицы

Определение очередного элемента матрицы осуществляется путем подачи на нижний вход исходного вектора- столбца, подачи на вход с (column) номера столбца, подачи на вход r (row) – номера строки матрицы.

4.3. Моделирование триггера для реализации преобразователя

Пример моделирования кодопреобразователя приведен на основе D-триггера. D-триггер рекомендуется моделировать на основе DV-триггера.

DV-триггер (в некоторых литературных источниках DF-триггер) также является универсальным элементарным автоматом с двумя входами. Когда $V=X_1=1$, то такой триггер работает как D-триггер. Если же $V=X_1=0$, то этот триггер сохраняет свое предыдущее состояние.

Рис. 4.9

Схемы моделирования DV-триггера на основе элементов И-НЕ и ИЛИ-НЕ представлены на рис. 4.10 и 4.11.

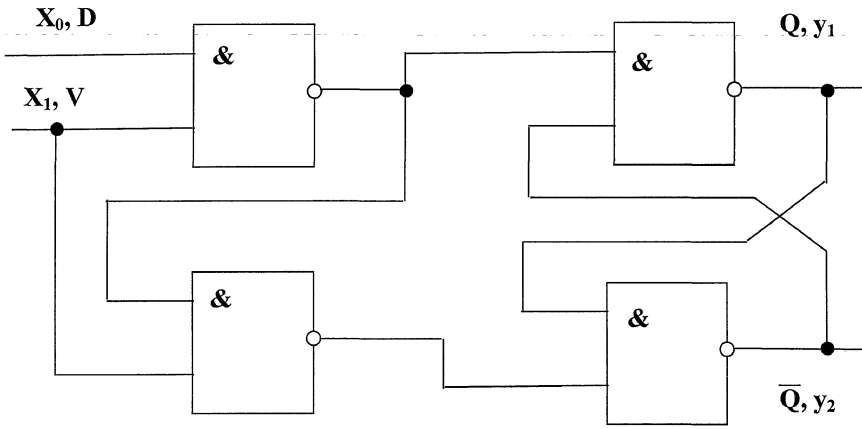


Рис. 4.10. DV-триггер на элементах И-НЕ

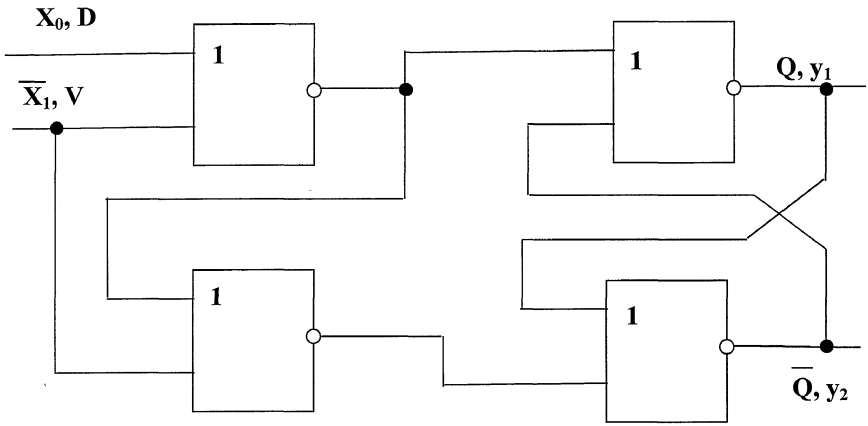


Рис. 4.11. DV-триггер на элементах ИЛИ-НЕ

Пример моделирования триггера на основе элементов И-НЕ представлен на рис. 4.13.

Схемы моделирования остальных триггеров на логических элементах представлены в [3].

Для реализации задержки на один такт работы кодопреобразователя в схему триггера необходимо включать в точках А блок задержки (рис. 4.13).

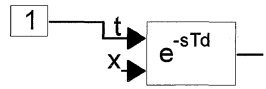


Рис. 4.12. Блок задержки на один такт работы преобразователя

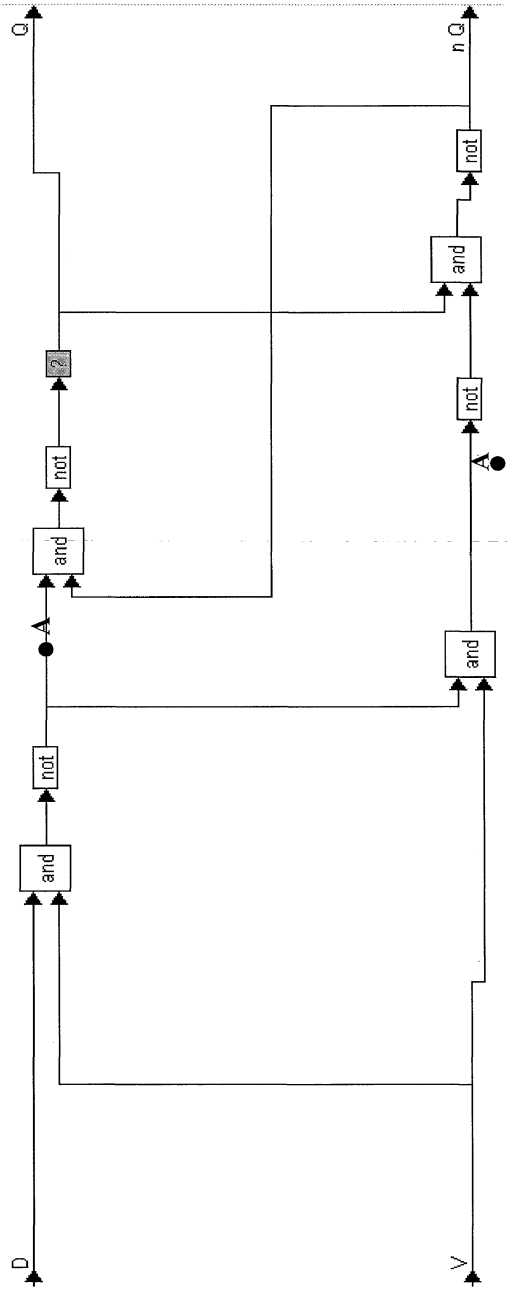


Рис. 4.13

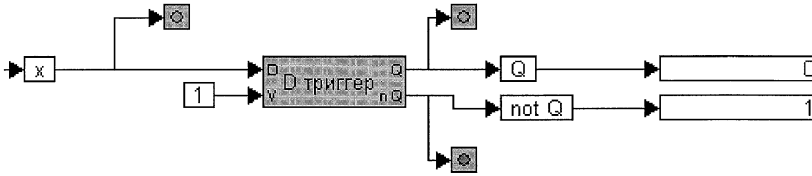


Рис. 4.14

4.4. Результаты моделирования кодопреобразоателя входной последовательности

Результаты моделирования кодопреобразователя, состоящего из D-триггера представлены на рисунках 4.15–4.17.

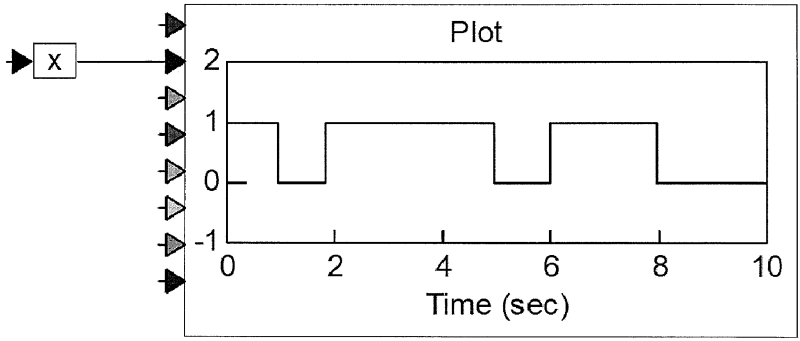


Рис. 4.15. Входная последовательность кодопреобразователя

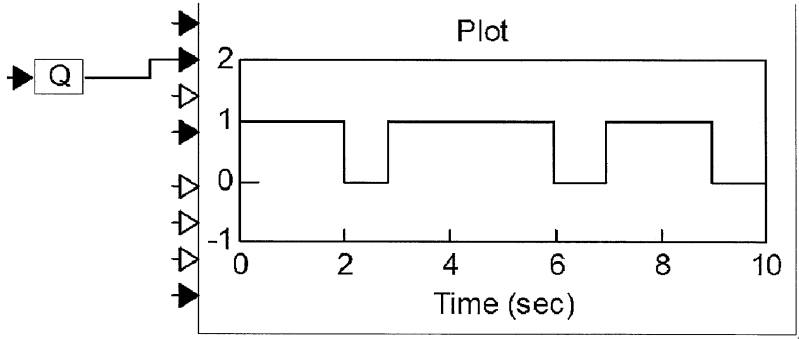


Рис. 4.16. Выходная последовательность Q кодопреобразователя

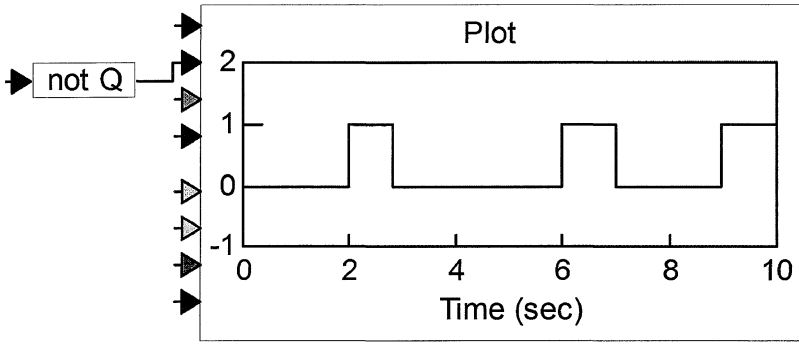


Рис. 4.17. Выходная последовательность $\overline{Q}$ кодопреобразователя

Результатом моделирования созданного в ходе курсовой работы кодопреобразователя должно являться соответствие полученных входных и выходных последовательностей работы автомата заданным условием работы.

ЗАКЛЮЧЕНИЕ

В рассмотренном выше учебно-методическом пособии теория конечных автоматов рассмотрена с позиции оценки её практической ценности. Как раздел теории конечных автоматов без памяти (теории булевых функций), так и раздел теории цифровых автоматов с памятью содержат множество примеров, показывающих, что цикл проектирования цифровых автоматов, от составления описания до получения принципиальной схемы, имеет минимальную длительность и малое количество этапов проектирования. По этому показателю теория конечных автоматов вообще превосходит любую другую теорию. Основная перспектива развития и практического применения теории цифровых автоматов лежит в области автоматизации проектирования сложных схем цифровых автоматов с помощью программ для цифровых вычислительных машин.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Савельев А.Я. Прикладная теория цифровых автоматов. – М.: Высшая школа, 1987.
2. Баранов С.И., Скляров В.А. Цифровые устройства на программируемых БИС с матричной структурой. – М.: Радио и Связь, 1986.
3. Щелкунов Н.Н., Дианов А.П. Процедуры программирования логических матриц, – Микропроцессорные средства и системы, 1986, №2.
4. Иванов В.И. Синтез цифровых автоматов для систем связи и управления.Челябинск, ЧПИ, 1980.
5. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979.
6. Андерсон Д.А. Дискретная математика и комбинаторика – Пер. с англ. – М.: ИД «Вильямс», 2004.
7. Дискретная математика // В. А. Горбатов, А. В. Горбатов, М. В. Горбатова Издательство: АСТ, Астрель , Высшая школа – 2006. – 448 с.
8. Ерусалимский Я. М. Дискретная математика Издательство: Вузовская книга – 2009. – 288с.
9. Баврин И. И. Дискретная математика Издательство: Высшая школа Серия: Для высших учебных заведений, 2007. – 200 с.
10. Редькин Н. П. Дискретная математика Издательство: ФИЗМАТЛИТ – 2009. – 264 с.
11. Иванов Б. Н. Дискретная математика. Издательство: ФИЗМАТЛИТ – 2007. – 408 с.
12. Дискретная математика //М. С. Спирина, П. А. Спирин Издательство: Академия Серия: Среднее профессиональное образование – 2010. – 368 с.
13. Набебин А. А. Дискретная математика Издательство: Научный мир. 2010. – 512 с.
14. Дискретная математика //С. Н. Поздняков, С. В. Рыбин Издательство: Образовательно-издательский центр "Академия", Серия: Высшее профессиональное образование. – 2008. – 448 с.
15. Дискретная математика. Практическая дискретная математика и математическая логика //С. Ф. Тюрин, Ю. А. Аляев. Издательство: Финансы и статистика, Инфра-М 2010. – 445 с.
16. Просветов Г. И. Дискретная математика. Задачи и решения Издательство: Альфа-Пресс – 2009. – 240 с.

ПРИЛОЖЕНИЯ

Приложение А

А1 Введение в теорию множеств

Теоретико-множественные представления – описание исследуемой системы, процессов средствами теории множеств, т.е. как *множества* взаимосвязанных и/или взаимодействующих частей – *элементов.* Связи между элементами задаются через *отношения* и/или *соответствия.* Множества, элементы, отношения, соответствия характеризуются определенными *свойствами* и набором допустимых *операций* над ними.

А1.1 Множества

Состав объекта исследования может быть представлен в виде дискретного множества. Множество — основное понятие в теории множеств, которое вводится без определения.

Основные понятия

Множество—состоит из *элементов.* Принадлежность элемента a множеству M обозначается $a \in M$ ("а принадлежит M"), непринадлежность – $a \notin M$ или $a \ni M$.

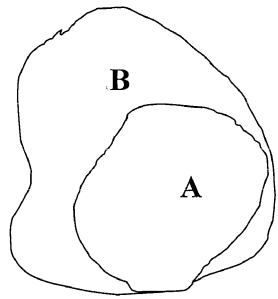


Рисунок А.1.1

Множество A называется *подмножеством* множества B (обозначается $A \subseteq B$), если всякий элемент из A является элементом B (рис А.1). Если $A \subseteq B$ и $A \neq B$, то A называется *строгим (собственным) подмножеством* (обозначается $A \subset B$).

Примеры обозначения числовых множеств:

N – {1, 2, 3, …} – множество натуральных чисел,

N_0 – {0, 1, 2, 3, …} – множество неотрицательных целых чисел,
Z – {0, ±1, ±2, ±3, …} – множество целых чисел,
R – множество действительных чисел.

Два определения *равенства множеств:*
I. Множества A и B равны $(A=B\}$, если их элементы совпадают.
П. Множества A и B равны, если $A \subseteq B$ и $B \subseteq A$.

Множество, состоящее из конечного числа элементов, называется *конечным,* в противном случае – *бесконечным* (например, множества N, R — бесконечные множества). Число элементов в конечном множестве M называется его *мощностью* и обозначается $\backslash M \backslash$.

Множество мощности 0, т.е. не содержащее элементов, называется *пустым* (обозначается Ø): $|Ø| = 0$. Принято считать, что пустое множество является подмножеством любого множества.

Способы задания множеств:

• *Перечислением,* т.е. списком своих элементов. Списком можно задать лишь конечные множества. Обозначение списка – в фигурных скобках. Например, множество A устройств домашнего компьютера, состоящего из системного блока a, а также периферийных устройств B (монитора *b,* клавиатуры *c* и принтера *d),* может быть представлено списком:
$A = \{a, B\}$ или $A = \{a, b, c, d\}$.

• *Порождающей процедурой,* которая описывает способ получения элементов множества из уже полученных элементов либо других объектов. В таком случае элементами множества являются все объекты, которые могут быть построены с помощью такой процедуры.

Например, множество всех целых чисел, являющихся степенями двойки M_2^n, $n \in N$, где N-множество натуральных чисел, (допустимое обозначение M_2^n = 1, 2, 4, 8, 16,...) может быть представлено порождающей процедурой, заданной двумя правилами:
а) $1 \in M_2^n$;
б) если $m \in M_2^n$, то $2m \in M_2^n$

• *Описанием характеристических свойств,* которыми должны обладать его элементы; обозначается:
$M = \{x \mid P(x)\}$ или $M = \{x : P(x)\}$.
"Множество M состоит из элементов x таких, что x обладает свойством Р").

Например, множество A периферийных устройств персонального компьютера может быть определено:
$A = \{x: x$ – периферийное устройство персонального компьютера$\}$.

Если свойство элементов множества M может быть описано коротким выражением, это упрощает его символьное представление. Например, множество всех натуральных четных чисел M_{2n} может быть представлено:

$M_{2n} = \{x: x = 2n, n \in N\}$.

Пример 1. Задать различными способами множество N всех натуральных чисел: *1,2,3, ...*

Списком множество N задать нельзя ввиду его бесконечности.

Порождающая процедура содержит два правила:

а) $1 \in N$;

б) если $n \in N$, то $n+1 \in N$.

Описание характеристического свойства элементов множества N:

$N = \{x: x$– целое положительное число$\}$.

Пример 2. Задать различными способами множество M всех четных чисел 2, 4, 6, ..., не превышающих 100.

$M_{2n}=\{2,4,6,...,100\}$.

а)$2 \in M_{2n}$;

б) если $n \in N$, то $(n+2) \in M_{2n}$;

в) $n \leq 98$.

$M_{2n} = \{n: n$– целое положительное число, не превышающее 100$\}$ или $M_{2n} = \{n: n \in N$ и $n/2 \in N, n \leq 100\}$.

Пример 3. Какие из приведенных определений множеств *A, B, C, D* являются корректными:

$A=\{1,2,3\}$;

$B=\{5,6,6,7\}$;

$C=\{x: x \in A\}$;

$D=\{A,C\}$.

1. Определение множества $A=\{1,2,3\}$ списком своих элементов формально корректно.

2. При перечислении элементов множества не следует указывать один и тот же элемент несколько раз. Корректное определение $B=\{5,6,7\}$.

3. Определение множества $C=\{x: x \in A\}$ заданием характеристического свойства его элементов «принадлежать множеству А» корректно.

4. Определние списком множества $D=\{A,C\}$ корректно.

Пример 4. Пары (1,2) и (2,1) не совпадают, хотя множества {1,2} и {2,1} равны.

Пример 5. Покажем, что множества $M_1 = \{x: \sin x=1\}$ и $M_2 = \{x: x=\pi/2+2k\pi, k\in Z\}$ совпадают.

Если $x\in M_1$, то x является решением уравнения $\sin x=1$. Но это значит, что x можно представить в виде $x=\pi/2+2k\pi$ и поэтому $x\in M_2$. Таким образом, $M_1 \subseteq M_2$. Если же $x\in M_2$, т. е. $x=\pi/2+2k\pi$, то $\sin x=1$, т.е. $M_2 \subseteq M_1$. Следовательно $M_1=M_2$.

Упражнения.

Задание 1. Задать различными способами множество M_3^n, всех чисел, являющихся степенями тройки, не превышающих 300.

Задание 2. Задать различными способами множество натуральных чисел, кратных пяти: 5, 10, 15, 20 …

Задание 3. Задать различными способами множество *M* нечетных чисел.

Задание 4. Задать различными способами множество *M* арабских чисел.

Операции над множествами

Объединением множеств *A* и *B* (обозначается $A \cup B$) называется множество, состоящее из всех тех элементов, которые принадлежат хотя бы одному из множеств *A, B* (рис А.2):

$$A \cup B= \{x: x\in A \text{ или } x\in B\}.$$

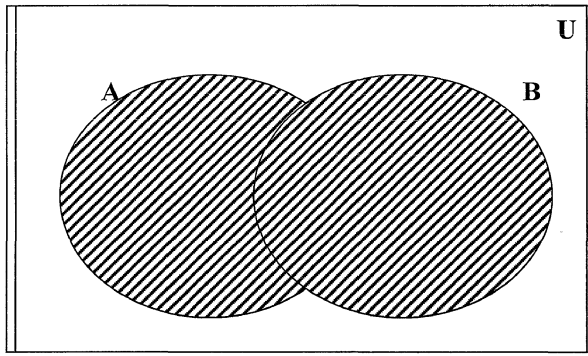


Рисунок А.1.2

Пример1.

Пусть заданы множества $A=\{a, b, d\}$,$B=\{b, d, e, h\}$.

Найти объединение данных множеств.

Решение.

$A \cup B = \{a, b, d, e, h\}$.

Пересечением множеств A и B (обозначается $A \cap B$) называется множество, состоящее из всех тех и только тех элементов, которые принадлежат и A и B (рис А.3):

$$A \cap B = \{x : x \in A \text{ и } x \in B\}.$$

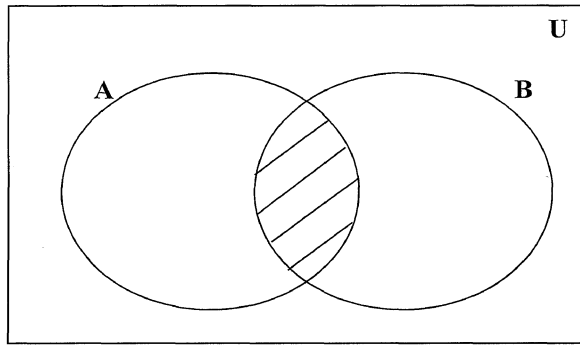


Рисунок А.1.3

Пример 2.

Пусть заданы множества $A=\{a, b, d\}$,$B=\{b, d, e, h\}$.

Найти пересечение данных множеств.

Решение.

$A \cap B = \{b, d\}$.

Разностью множеств A и B (*обозначается* $A \setminus B$) называется множество всех тех и только тех элементов A, которые не содержатся в B (рис А.4):

$$A \setminus B = \{x : x \in A \text{ и } x \notin B\}.$$

Разность – операция строго двухместная и некоммутативная: в общем случае $A \setminus B \neq B \setminus A$.

Пусть U – *универсальное множество* такое, что все рассматриваемые множества являются его подмножествами.

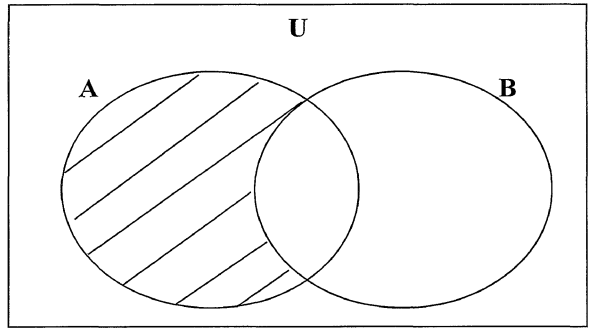


Рисунок А.1.4

Пример 3. Пусть заданы множества $A=\{a, b, d\}$,$B=\{b, d, e, h\}$. Найти разность данных множеств.

Решение.

$A\backslash B = \{a\}$.

$B\backslash A=\{e, h\}$.

Дополнением (до *U)* множества *A* (обозначается $\overline{A}$) называется множество всех элементов, не принадлежащих *A* (но принадлежащих *U)* (рис. А.5):

$$A = U\backslash A.$$

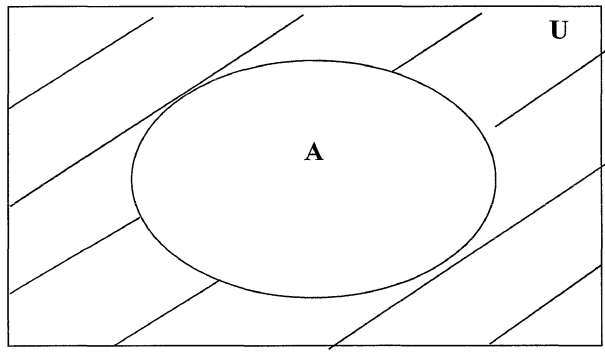


Рисунок А.1.5

Операции объединения, пересечения, дополнения часто называют *булевыми операциями над множествами.*

Упражнения

1. Пусть A={1, 2, 3, 4, 10, 20}, B={6, 10, 11, 15}. Найти A ∪ B.
2. Пусть C={100, 105, 106 120}, D={95, 100, 105, 130, 140}. Найти A ∩ D.
3. Пусть A={a, b, d, e, f, i, j}, B={e, f, i, k, l, m}. Найти A\B, B\A.

4. Пусть U={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, F={2, 3, 5, 6, 7}. Найти $\overline{F}$.

Свойства операций объединения и пересечения

Для произвольных множеств A, B, C выполняются следующие соотношения.

1. Идемпотентность ($A \cap A=A$; $A \cup A=A$).
2. Коммуникативность ($A \cap B=B \cap A$; $A \cup B=B \cup A$).
3. Ассоциативность ($(A \cap B) \cap C=A \cap (B \cap C)$; $(A \cup B) \cup C=A \cup (B \cup C)$).
4. Дистрибутивность ($A \cup (B \cap C)=(A \cup B) \cap (A \cup C)$; $A \cap (B \cup C)=(A \cap B) \cup (A \cap C)$).
5. Поглощение ($(A \cap B) \cup A=A$; $(A \cup B) \cap A=A$).
6. Свойство нуля ($A \cap \varnothing=\varnothing$; $A \cup \varnothing=A$).
7. Свойство единицы ($A \cap U=A$; $A \cup U=U$).
8. Инволютивность ($\overline{\overline{A}} = A$).
9. Законы де Моргана ($\overline{A \cap B} = \overline{A} \cup \overline{B}$, $\overline{A \cap B} = \overline{A} \cup \overline{B}$).
10. Свойства дополнения ($A \cup \overline{A} = U$, $A \cap \overline{A} = \varnothing$).

Диаграммы Венна

Диаграммы Венна – геометрические представления множеств. Построение диаграммы заключается в изображении большого прямоугольника, представляющего универсальное множество *U,* а внутри его замкнутых фигур), представляющих множества. Фигуры должны пересекаться в наиболее общем случае, требуемом в задаче, и должны быть соответствующим образом обозначены. Точки, лежащие внутри различных областей диаграммы, могут рассматриваться как элементы соответствующих множеств. Имея построенную диаграмму, можно заштриховать определенные области для обозначения вновь образованных множеств.

Приведенные на рис. А.1.2 – А.1.5 иллюстрации операций объединения, пересечения, разности и дополнения двух множеств являются диаграммами Венна.

Пример 1. Представить множество $A \cup (B \cap \overline{C})$ диаграммой Венна.

Начнем с общей диаграммы, показанной на рис. А.1.6,а.

Заштрихуем B диагональными линиями в одном направлении, а $\overline{C}$ – в другом (рис. А.1 .6, б). Площадь с двойной штриховкой представляет собой их пересечение, т.е. множество $(B \cap \overline{C})$. Выделим это множество жирной линией. На новой копии диаграммы заштрихуем эту область $(B \cap \overline{C})$ линиями одного направления, а A – другого. Вся заштрихованная на рис. 1..6,в область представляет объединение множеств $A \cup (B \cap \overline{C})$, т.е. множество $A \cup (B \cap \overline{C})$. Обведем искомую область также жирной линией.

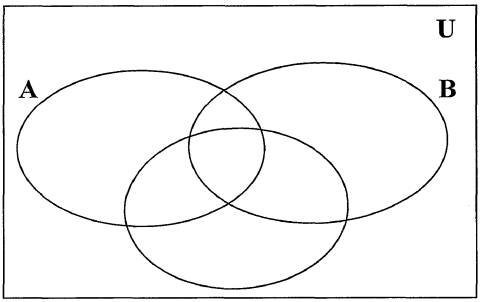


Рисунок А.1.6, а

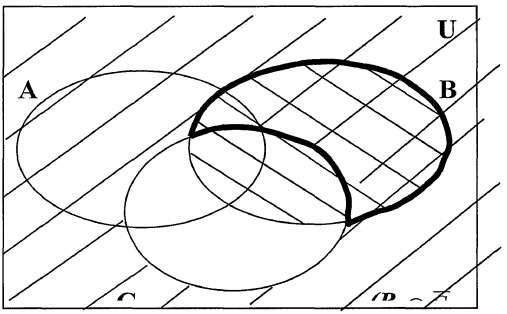


Рисунок А.1.6, б

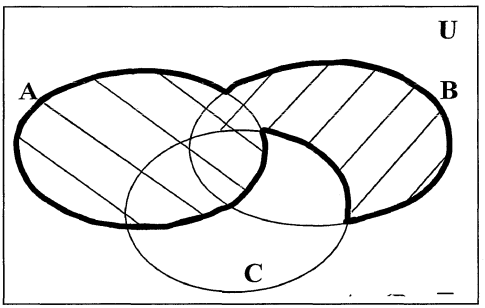


Рисунок А.1.6, в

Упражнения.

Пусть A, B, C $\subseteq$ U. Проиллюстрировать на примере конкретных множеств и с помощью диаграмм Венна справедливость следующих соотношений:

1. $A \cap (B \cap C) = (A \cap B) \cap C$,

2. $A \cup (B \cup C) = (A \cup B) \cup C$,
3. $\overline{A \cap B} = \overline{A} \cup \overline{B}$,
4. $\overline{A \cup B} = \overline{A} \cap \overline{B}$,
5. $A \cup (A \cap B) = A$,
6. $A \cap (A \cup B) = A$,
7. $(A \cap B) \cup (A \cap \overline{B}) = A$,
8. $A \cup (\overline{A} \cap B) = A \cup B$,

А1.2 Отношения

Отношения – один из способов задания взаимосвязей между элементами множества. Наиболее изученными и чаще всего используемыми являются так называемые унарные и бинарные отношения.

Унарные (одноместные) отношения отражают наличие какого-то определенного признака R (свойства и т.п.) у элементов множества M (например, "быть белым" на множестве шаров в урне). Тогда все такие элементы a из множества M, которые отличаются данным признаком R, образуют некоторое подмножество в M, называемое унарным отношением R, т.е. $a \in R$ и $R \subseteq M$.

Бинарные (двухместные) отношения используются для определения каких-то взаимосвязей, которыми характеризуются пары элементов в множестве M (так, на множестве людей могут быть заданы, например, следующие бинарные отношения: "жить в одном городе", "быть моложе", "быть сыном", "работать в одной организации" и т.п.). Тогда все пары (a, b) элементов из M, между которыми имеет место данное отношение R, образуют подмножество пар из множества всех возможных пар элементов $M \times M = M^2$, называемое бинарным отношением R, т.е. $(a, b) \in R$, при этом $R \subseteq M \times M$.

В общем случае могут рассматриваться *n-местные отношения*, например отношения между тройками элементов трехместные (тернарные) отношения и т.д.

А1.2.1 Бинарные отношения. Основные определения

Двухместным, или *бинарным*, отношением R называется подмножество пар $(a, b) \in R$ прямого произведения $M_1 \times M_2$, т.е. $R \in M_1 \times M_2$. При этом множество M_1 называют областью определения отношения R, множество M_2 – областью значений. Часто рассматривают отношения R между парами элементов одного и того же множества M, тогда $R \subseteq M \times M$. Если a, b находятся в отношении R, это часто записывается как aRb.

Пусть $R \in A \times B$ определено в соответствии с изображением на рисунке 2.1. *Область определения D(R)* и *область значений Q(R)* определяются соответственно:

$$D(R) = \{a: (a, b) \in R\},$$
$$Q(R) = \{b: (a, b) \in R\}.$$

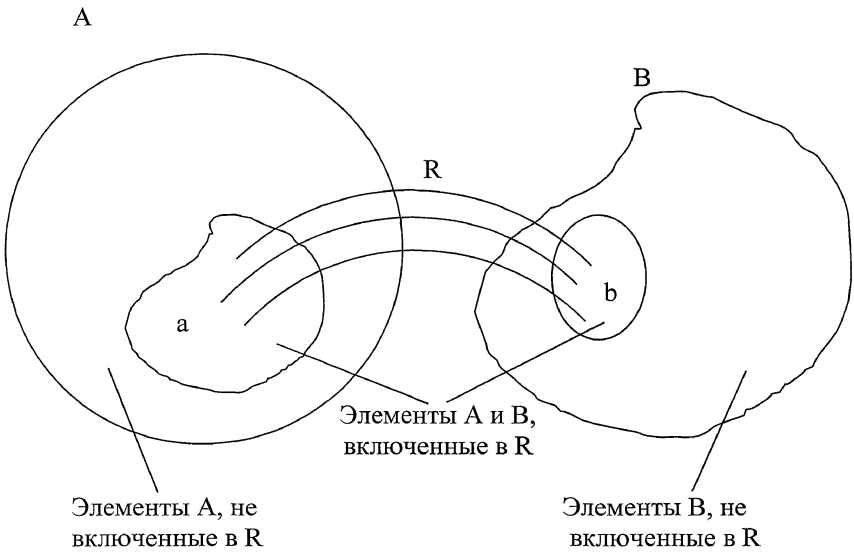


Рисунок А.2.1

Способы задания бинарных отношений – любые способы задания множеств (так как отношения определены выше как подмножества некоторых множеств – прямых произведений). Отношения, определенные на конечных множествах, обычно задаются:

1. *Списком (перечислением) пар,* для которых это отношение выполняется. Например, $R = \{(a, b), (a, c), (b,d)\}$.

2. *Матрицей* – бинарному отношению $R \subseteq M$ x M, где $M = \{a_1, a_2, ..., a_n\}$, соответствует квадратная матрица порядка n, в которой элемент c_{ij}.., стоящий на пересечении i-и строки j-го столбца, равен 1, если между a_i и a_j имеет место отношение R или 0, если оно отсутствует:

$$c_{ij} = \begin{cases} 1, \text{если } a_i R a_j, \\ 0 \text{ в противном случае.} \end{cases}$$

Пример 1. Пусть $M = \{1,2,3,4,5,6\}$. Задать в явном виде (списком) и матрицей отношение $R \in M \times M$, если R означает – "быть строго меньше".

Отношение R как множество содержит все пары элементов a, b из M такие, что $a < b$:

$$R = \{(a, b): a,b \in M; a < b\}.$$

Тогда R = *{(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (5, 6)}*.

Матрица отношения приведена на рисунке А.2.2.

R	1	2	3	4	5	6
1	0	1	1	1	1	1
2	0	0	1	1	1	1
3	0	0	0	1	1	1
4	0	0	0	0	1	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

Рисунок А.2.2

Пример 2. Пусть M= *{*1,2,3,4,5,6}. Составить матрицу отношения R_1, R_2, $R_3 \subseteq M \times M$, если:

1) R_1 – “быть делителем”;

2) R_2 – “иметь общий делитель”;

3) R_3 –“иметь один и тот же остаток от деления на 3”.

1. R_1={(a, b): a,b $\in$M; a – делитель b} и выполняется для пар {(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (2,2), (2,4), (2,6), (3,3), (3,6), (4,4), (5,5), (6,6)}. Эти пары (a,b)$\in R_1$ определяют наличие единицы в матрице отношения $R_1 \in M^2$ на пересечении строки элемента a и столбца b, a,b $\in$M (рисунок А.2.3).

R	1	2	3	4	5	6
1	1	1	1	1	1	1
2	0	1	0	1	0	1
3	0	0	1	0	0	1
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	0	0	0	0	0	1

Рисунок А.2.3

2. R_2={(a, b): a,b $\in$M; a и b имеют общий делитель, c$\neq$1}. Матрица отношения приведена на рисунке А.2.4.

R	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	1	0	1	0	1
3	0	0	1	0	0	1
4	0	1	0	1	0	1
5	0	0	0	0	1	0
6	0	1	1	1	0	1

Рисунок А.2.4

3. R_3={(a, b): a,b ∈M; a и b имеют один и тот же остаток от деления на 3}. Матрица отношения приведена на рисунке А.2.5.

R	1	2	3	4	5	6
1	1	0	0	1	0	0
2	0	1	0	0	1	0
3	0	0	1	0	0	1
4	1	0	0	1	0	0
5	0	1	0	0	1	0
6	0	0	1	0	0	1

Рисунок А.2.5

А1.2.2. Свойства бинарных отношений

Пусть R – отношение на множестве M, $R \subseteq M \times M$. Тогда:

1) R – *рефлексивно,* если имеет место $a\ R\ a$ для любого $a \in M$ (например, отношение "жить в одном городе" – рефлексивно);

2) R – *антирефлексивно,* если ни для какого a ∈ M не выполняется $a\ R\ a$ (например, отношение "быть сыном" – антирефлексивно);

3) R – *симметрично,* если $a\ R\ b$ влечет $b\ R\ a$ (например, отношение "работать на одной фирме" – симметрично);

4) R – *антисимметрично,* если $a\ R\ b$ и $b\ R\ a$ влекут $a = b$, т.е. ни для каких различающихся элементов a и b $(a \neq b)$ не выполняется одновременно aRb и bRa (например, отношения "быть сыном", "быть начальником" – антисимметричны);

5) R – *транзитивно,* если aRb и bRc влекут aRc (например, отношения "быть моложе", "быть братом" – транзитивны).

А2 Введение в теорию графов

Графические представления в широком смысле – любые наглядные отображения исследуемой системы, процесса, явления на плоскости. К ним могут быть отнесены рисунки, чертежи, графики зависимостей характеристик, планы-карты местностей, блок-схемы процессов, диаграммы и т.п. Такие изображения наглядно представляют различные взаимосвязи, взаимообусловленности: топологическое (пространственное) расположение объектов, хронологические (временные) зависимости процессов и явлений, логические, структурные, причинно-следственные (каузальные) и другие взаимосвязи.

А2.1 Основные понятия

Графические представления – удобный способ иллюстрации содержания различных понятий, относящихся к другим способам формализованных представлений (см., например, диаграммы Венна и другие графические иллюстрации основных теоретико-множественных и логических представлений).

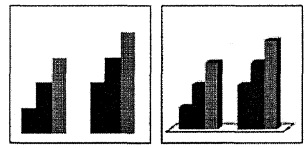


Рисунок А.2.6

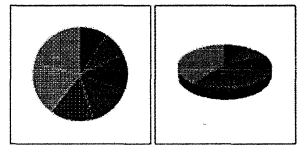


Рисунок А.2.7

Все более распространенными становятся представления количественных характеристик, взаимосвязей между объектами в виде разного рода одно-, двух- и более мерных гистограмм (рисунок А.2.6), круговых диаграмм (рисунок А.2.7), других аналогичных способов представления в виде тех или иных геометрических фигур, по наглядным характеристикам которых (высоте, ширине, площади, радиусу и пр.) можно судить о количественных соотношениях сравниваемых объектов, значительно упрощая их анализ.

Мощным и наиболее исследованным классом объектов, относящихся к графическим представлениям, являются так называемые *графы,* изучаемые в *теории графов.* Теория графов имеет огромные приложения, так как ее язык, с одной стороны, нагляден и понятен, а с другой – удобен в формальном исследовании. На языке теориивграфов формулируются и решаются многие задачи управления, в том числе задачи сетевого планирования и управления, анализа и проектирования организационных структур управления, анализа процессов функционирования и целеполагания, многие задачи принятия решений в условиях неопределенности и др.

Следует иметь в виду, что при изображении графа не все детали рисунка одинаково важны; в частности, несущественны геометрические свойства ребер (длина, кривизна и т.д.) и взаимное расположение вершин на плоскости.

Графические представления в узком смысле – это описание исследуемой системы, процесса, явления средствами теории графов в виде совокупности двух классов объектов: *вершин* и соединяющих их линий – *ребер* или *дуг.* Графы и их составляющие характеризуются определенными свойствами и набором допустимых преобразований (операций) над ними.

Графом G называется совокупность двух множеств: *вершин V* и *ребер E,* между элементами которых определено *отношение инцидентности* — каждое ребро *ее E* инцидентно ровно двум вершинам v', v"∈ *V,* которые оно соединяет. При этом вершина v' (v") и ребро *е* называются *инцидентными* друг другу, а вершины v' и v", являющиеся для ребра *е* концевыми точками, называются *смежными.* Часто вместо v ∈ *V* и *е* ∈ *E* пишут соответственно v ∈ *G, е* ∈ *G.*

Ребро, соединяющее две вершины, может иметь направление от одной вершины к другой; в этом случае оно называется *направленным,* или *ориентированным,* или *дугой* и изображается стрелкой, направленной от вершины, называемой *началом,* к вершине, именуемой *концом.*

Граф, содержащий направленные ребра (дуги) с началом v' и концом v", называется *ориентированным* (орграфом), а ненаправленные – *неориентированным* (назовем *н-графом).*

Ребра, инцидентные одной и той же паре вершин, называются *параллельными,* или *кратными.* Граф, содержащий кратные ребра, именуется *мультиграфом.* Ребро, концевые вершины которого совпадают, называется *петлей.*

Граф называется *конечным,* если множество его элементов (вершин и ребер) конечно, и *пустым,* если его множество вершин F(а значит и ребер *Е)* пусто. Граф без петель и кратных ребер именуется *полным,* если каждая пара вершин соединена ребром.

Дополнением графа G называется граф $\overline{G}$, имеющий те же вершины, что и граф *G,* и содержащий только те ребра, которые нужно добавить к графу *G,* чтобы получить полный граф.

Каждому неориентированному графу *канонически соответствует* ориентированный граф с тем же множеством вершин, в котором каждое ребро заменено двумя ориентированными ребрами, инцидентными тем же вершинам и имеющими противоположные направления.

Локальной степенью (или просто *степенью) вершины* $v \in V$ н-графа *G* называется количество ребер $\rho(v)$, инцидентных вершине v. В н-графе сумма степеней всех вершин равна удвоенному числу ребер *m* графа, т.е. четна (предполагается, что в графе с петлями петля дает вклад 2 в степень вершины):

$$\sum_{v \in G} \rho(v) = 2m,$$

отсюда следует, что в н-графе число вершин нечетной степени четно.

Для вершин орграфа определяются две локальные степени:

- ρ_1 (v) –число ребер с началом в вершине v, или количество выходящих из v ребер;
- ρ_2 (v) – количество входящих в v ребер, для которых эта вершина является концом.

Петля дает вклад 1 в обе эти степени.

В орграфе суммы степеней всех вершин $\rho_1(v)$ и $\rho_2(v)$ равны количеству ребер *m* этого графа, а значит, и равны между собой:

$$\sum_{v \in G} \rho_1(v) = \sum_{v \in G} \rho_2(v) = m,$$

Графы G_1 и G_2 *равны,* т.е. $G_1 = G_2$, если их множества вершин и ребер (выраженных через пары инцидентных им вершин) совпадают: $V_1 = V_2$ и $E_1 = E_2$ Графы G_1 и G_2 на рисунке А.2.8 равны. Граф *G* считается *полностью заданным* в строгом смысле, если нумерация его вершин и ребер зафиксирована. Графы, отличающиеся только нумерацией вершин и ребер, называются *изоморфными.*

Пример 1. Задать граф G_1 представленный на рисунке А.2.8, через множества вершин V_1 и ребер E_1.

Граф G_1 может быть полностью определен:

1) двумя множествами поименованных вершин $V_1 = \{v_1, v_2, v_3, v_4, v_5\}$ и поименованных ребер $E_1 = \{e_1, e_2, e_3, e_4\}$ (в строгом смысле требуется установление отношения инцидентности ребер соответствующим вершинам).

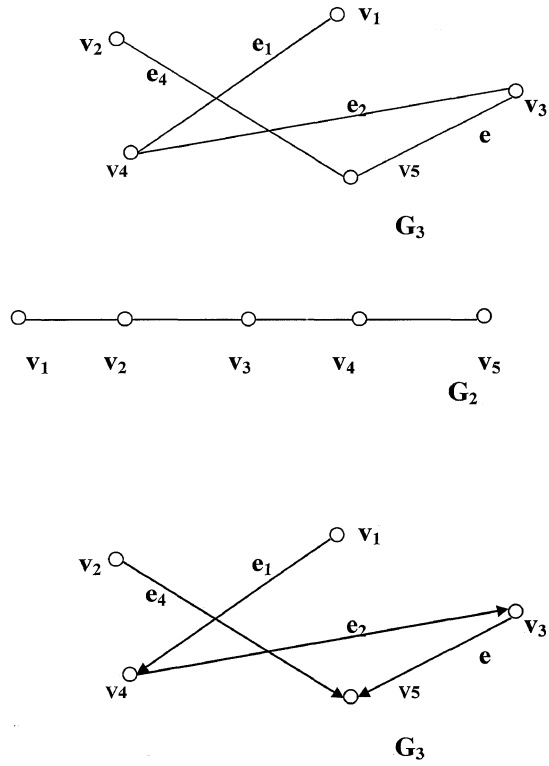


Рисунок А.2.8

2) множеством ребер, каждое из которых представлено парой своих концевых вершин: $E_1 = \{(v_1\ v_4), (v_4, v_3), (v_3, v_5), (v_5, v_2)\}$.

Порядок указания вершин при описании ребра здесь безразличен, так как все ребра в графе G неориентированные.

Пример 2. На рисунке А.2.9 изображены графы G_1–G_{12} с четырьмя вершинами в каждом. Сравнить графы.

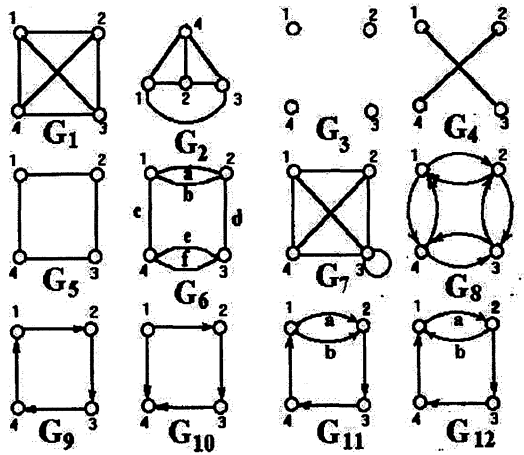


Рисунок А.2.9

Результаты сравнения графов таковы:

G_1–G_7 – неориентированные; G_8 – G_{12} - ориентированные;

G_1, G_2 – полные, причем $G_1 = G_2$;

G_7 – не является полным, так как хотя каждая пара вершин и соединена ребром, но имеется одна петля. (Иногда полным называют граф с петлями во всех вершинах, каждая пара которых соединена ребром. Граф G_7 не отвечает и этому определению.

G_3 – все вершины этого графа являются изолированными (граф с пустым множеством ребер);

G_4 и G_5 являются дополнением друг другу: $G_4 = \overline{G}_5$ и $G_5 = \overline{G}_4$;

G_6 – мультиграф, так как содержит кратные ребра *a* и b, а также e и f;

G_8 – ориентированный, канонически соответствующий неориентированному графу G_5;

G_9 и G_{10} не являются равными, так как имеют отличающиеся ребра: (4, 1) – в G_9 и (1,4) –в G_{10};

G_{11} – ориентированный мультиграф: ребра *a* и b – кратные, тогда как G_{12} мультиграфом не является, поскольку в нем ребра a и b различно ориентированы.

Пример 3. Чему равны степени вершин графов G_1 и G_3 на рисунке 2.6?

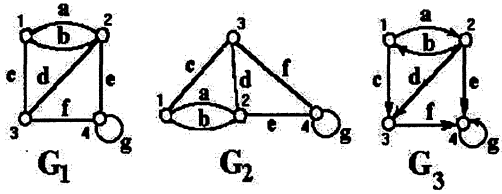


Рисунок А.2.10

Оба графа имеют по четыре вершины: V= {1,2,3,4}.

Степени вершин неориентированного графа G_1 : $\rho(1) = 3$, $\rho(2) = 4$, $\rho(3) = 3$, $\rho(4) = 4$, если условиться считать вклад петли в степень вершины, равный 2, и $\rho(4) = 3$, если петля дает вклад 1 в степень вершины. Сумма степеней всех вершин графа G_1 равна 14, т.е. вдвое больше числа ребер графа:

$$\sum_{v \in G} \rho(v) = 14 = 2m,$$

где $m = 7$ – число ребер графа.

Степени вершин ориентированного графа G_3:

$\rho_1(1) = 2$, $\rho_1(2) = 3$, $\rho_1(3) = 1$, $\rho_1(4) = 1$;

$\rho_2(1) = 1$, $\rho_2(3) = 1$, $\rho_2(3) = 2$, $\rho_2(4) = 3$.

Суммы степеней вершин первого и второго типа графа G_3 совпадают и равны числу m ребер графа:

$$\sum_{v \in G} \rho_1(v) = \sum_{v \in G} \rho_2(v) = 7 = m.$$

Упражнения

1. Задать графы $G_2 - G_3$ (см. рисунок 2.3) множествами их вершин и ребер. Сравнить графы $G_1 - G_3$ (см. пример 1).

2. Равны ли графы $G_1 - G_2$ на рисунке 2.5? Задать графы $G_1 - G_3$ множествами их вершин и ребер. Сравнить графы.

3. Определить дополнение графа G, если:

а) G – пятиугольник;

б) G – треугольник.

А2.2 Способы задания графов

Выше мы познакомились с двумя способами описания графов: графическим, и в виде двух множеств вершин V и ребер E, когда каждое ребро $e \in E$ определено парой инцидентных ему концевых вершин (v', v''). Рассмотрим другие способы, используемые в теории графов.

В общем, виде задать граф – значит, описать множества его вершин и ребер, а также отношение инцидентности. Для описания вершин и ребер

достаточно их занумеровать. Пусть $v_1, v_2, \ldots, v_j \ldots, v_n$ – вершины графа G: $e_1, e_2, \ldots, e_j, \ldots, e_m$ – ребра. Отношение инцидентности задается:

1) *матрицей инцидентности* $\|\varepsilon_{ij}\|$ размера *m*xn: по вертикали и горизонтали указываются вершины и ребра соответственно, а на пересечении i-й вершины и j-го ребра в случае неориентированного графа проставляется 1, если они инцидентны, и 0 — в противном случае, т.е.

$$\varepsilon_{ij} = \begin{cases} 1, \text{если ребро } e_i \text{ инцидентно вершине } v_j, \\ 0 \text{ - в противном случае,} \end{cases}$$

а в случае орграфа: –1, если вершина является началом ребра, 1 – если вершина является концом ребра, и 0 – если вершина и ребро не инцидентны; если некоторая вершина является для ребра и началом, и концом (т.е. ребро – петля), проставляется любое другое число, например 2, т.е.

2) *списком ребер* графа, представленным двумя столбцами: в левом перечисляются все ребра $e_i \in E$, а в правом – инцидентные ему вершины для н-графа порядок вершин в строке произволен, для орграфа первым стоит номер начала ребра;

3) *матрицей смежности* $\|\delta_{kl}\|$ – квадратной матрицей размера *n*x*n*: по вертикали и горизонтали перечисляются все вершины $v_j \in V$, а на пересечении *k*-й и *l*-й вершин в случае н-графа проставляется число, равное числу ребер, соединяющих эти вершины; для орграфа δ_{kl} равно числу ребер с началом в *k*-й вершине и концом в *l*-й.

Если два графа равны, то их матрицы совпадают. Если в графе поменять нумерацию вершин, матрицы (и список ребер) в общем случае изменяются, т.е. вид матриц и списка ребер зависит от нумерации вершин и ребер графа. Строго говоря, граф считается *полностью заданным,* если нумерация его вершин зафиксирована. Графы, отличающиеся только нумерацией вершин, являются *изоморфными.* Проверка изоморфности графов – в общем случае трудоемкая задача.

Пример 1. Задать матрицами инцидентности и смежности, а также списком ребер графы G_1, G_3 (см. рисунок А.2.11).

Матрицы инцидентности графов G_1 и G_3 приведены на рисунке А.2.12. В матрице инцидентности в каждом столбце только два элемента, отличных от 0 (или один, если ребро – петля).

G_1	a	b	c	d	e	f	g
1	1	1	1	0	0	0	0
2	1	1	0	1	1	0	0
3	0	0	1	1	0	1	0
4	0	0	0	0	1	1	1

G_3	a	b	c	d	e	f	g
1	–1	1	–1	0	0	0	0
2	1	–1	0	–1	–1	0	0
3	0	0	1	1	0	–1	0
4	0	0	0	0	1	1	2

Рисунок А.2.11

Список ребер является более компактным описанием графа. Список ребер орграфа G_3 приведен на рисунке А.2.12, для н-графа G_1 он аналогичен, однако последовательность указания вершин здесь безразлична.

Ребро	Вершины	
a	1	2
b	2	1
c	1	3
d	2	3
e	2	4
f	3	4
g	4	4

Рисунок А.2.12

Матрицы смежности графов даны на рисунке А.2.13.

G_1	**1**	**2**	**3**	**4**
1	0	2	1	0
2	2	0	1	1
3	1	1	0	1
4	0	1	1	1

G_3	**1**	**2**	**3**	**4**
1	0	1	1	0
2	1	0	1	1
3	0	0	0	1
4	0	0	0	1

Рисунок А.2.13

Приложение Б

Варианты задания к курсовой работе

Варианты	$\tilde{x}$ $p_3p_2p_1p_1$	$\tilde{z}$ $p_3p_2p_1p_1$
1	7421	5321
2	7421	5421
3	7421	631–1
4	7421	5421
5	7321	4311
6	7421	5211
7	7421	5321
8	5421	4311
9	5421	5211
10	6421	7421
11	6421	2421
12	6421	5321
13	6421	5211
14	6421	4311
15	6421	3321
16	6421	6412
17	532–1	5221
18	532–1	4311
19	532–1	4231
20	631–1	5421
21	631–1	4311
22	6311	7421
23	6311	5421
24	4311	5421
25	7421	5421
26	5421	7421
27	4221	4421
28	4221	7421
29	4221	5421
30	3321	4221

Учебное издание

Барбасова Татьяна Александровна,
Гудилин Алексей Евгеньевич

ТЕОРИЯ КОНЕЧНЫХ АВТОМАТОВ

Учебное пособие

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 16.05.2014. Формат 60×84 1/16. Печать цифровая.
Усл. печ. л. 6,97. Тираж 30 экз. Заказ 177/79.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, пр. им. В.И. Ленина, 76.