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

519.7(07)
Г935

А.Е. Гудилин

ЦИФРОВАЯ СХЕМОТЕХНИКА

Учебное пособие

Челябинск
Издательство ЮУрГУ
2000

УДК 519.713(075.8)+512.563(075.8)+519.1(075.8)

Гудилин А.Е. Цифровая схемотехника: Учебное пособие – Челябинск: Изд. ЮУрГУ, 2000. – 130 с.

Пособие предназначено для студентов специальности «Управление и информатика в технических системах» дневной и заочной форм обучения. В пособии рассмотрены теоретические основы построения комбинационных схем и цифровых автоматов. Излагаемый материал иллюстрирован примерами построения практических схем.

Ил. 76, табл. 44, список лит. – 5 назв.

Одобрено учебно-методической комиссией приборостроительного факультета.

Рецензенты: Ермакова В.И., Мальцев С.Н.

ISBN 5–696–01724–X

ВВЕДЕНИЕ

В курсе “Цифровая схемотехника” 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).

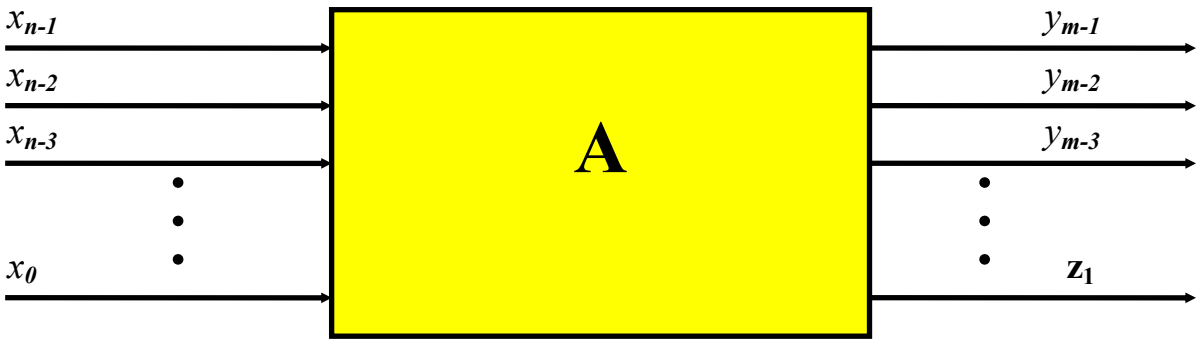


Рис.1. Обобщённая структура системы переработки цифровой информации:
$x_{n-1}x_{n-2}x_{n-3}..x_0$ – входной цифровой ***n*** – разрядный двоичный код;
$y_{m-1}y_{m-2}y_{m-3}...y_0$ – выходной цифровой ***m*** – разрядный двоичный код;
A – оператор преобразования

Особенностью цифрового автомата является зависимость оператора преобразования **A** от предыдущих состояний кодопреобразователя, то есть наличие памяти у цифрового автомата. В частном случае, при отсутствии блока 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 полно разработана теория цифровых автоматов применительно к канонической структуре цифрового автомата, представленной на рис.2. В учебном пособии рассматривается только эта структура цифрового автомата.

По рис.2 видно, что входные коды как входной, так и выходной комбинационных схем получаются в результате конкатенации (объединения) входного кода и кода состояния блока памяти цифрового автомата.

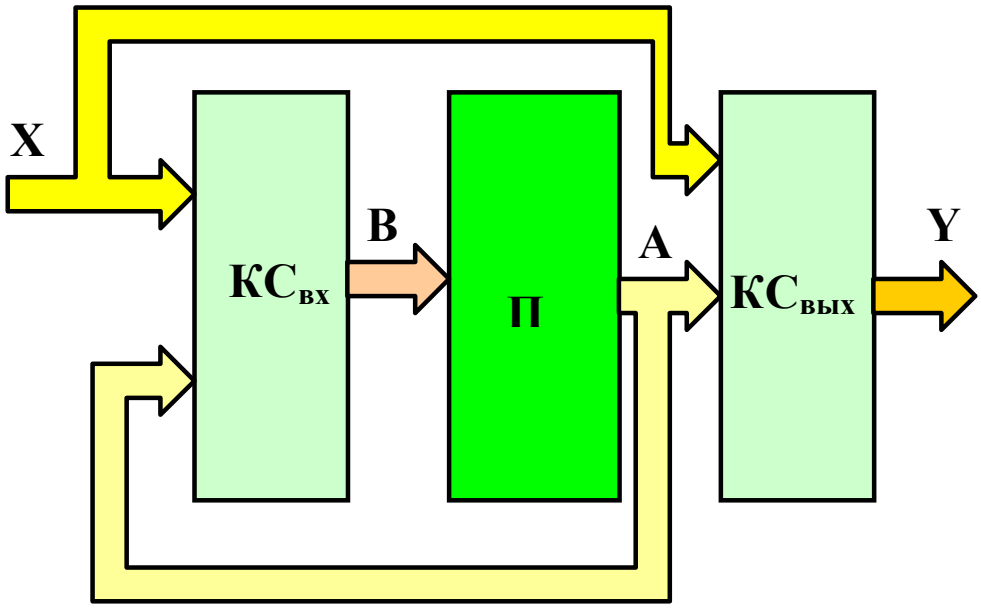


Рис. 2. Каноническая структурная схема цифрового автомата

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

Глава 1

ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ АВТОМАТОВ

1.1. 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). Поэтому алгебру логики называют алгеброй Буля или булевой алгеброй.

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 $f(x_{n-1}, x_{n-2}, ..., x_0)$, принимающая значения равные нулю или единице на наборах логических переменных $x_{n-1}, x_{n-2}, ..., x_0$.

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 ($i = 0, ... , n–1$).

Для n–разрядного кода общее количество уникальных наборов переменных

$$N = 2^n. \qquad (1)$$

Максимальное числовое значение двоичного кода равно

$$A_{макс} = 2^n - 1 \qquad (2)$$

Значения всех логических функций от одной переменной представлены в табл. 1.

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

Таблица 1

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

Значения всех логических функций от двух переменных представлены в табл. 2. Для функции двух переменных согласно (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 табл. 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.

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

Аналитически это свойство описывается следующей формулой:

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

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

Система булевых функций W называется функционально полной, если для любой булевой функции n – переменных $f(x_{n-1}, x_{n-2}, ..., x_0)$ может быть построена равносильная ей функция комбинированием булевых переменных $x_{n-1}, x_{n-2}, ..., x_0$ 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.

1.2. 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;

Таблица 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

3. $x \vee 0 = x$;
4. $x \vee 1 = 1$;
5. $x \cdot 0 = 0$;
6. $x \cdot 1 = x$;
7. $\bar{x}x = 0$;
8. $\bar{x} \vee x = 1$.

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:***

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

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

Из законов де Моргана следует, что

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

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

таким образом, 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:

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

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

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:

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

$$\mathbf{F}_i \vee \mathbf{F}_j = \mathbf{F}_k x_p \vee \mathbf{F}_k \bar{x}_p = \mathbf{F}_i \vee \mathbf{F}_j \vee \mathbf{F}_k$$ – неполное склеивание. (14)

Более важным для практики является неполное склеивание, так как для сложных ФАЛ исходные конституенты 1 $\mathbf{F}_i$ и $\mathbf{F}_j$ после получения результата склеивания друг с другом $\mathbf{F}_k$ сохраняются для сопоставления с другими минтермами заданной ФАЛ и нового склеивания по одной из переменных, входящих в сопоставляемые минтермы с отрицанием и без отрицания (т.е. отличающихся по одной переменной).

Рассмотренные соотношения позволяют описать равносильные булевы функции различными способами, то есть открываются возможности выбора самых простых форм описания ФАЛ. Такие формы ФАЛ реализуются на элементной базе по принципиальным схемам, имеющим самую низкую стоимость. Это придаёт большое практическое значение исследованиям способов минимизации ФАЛ.

1.3. 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.

1.3.1. 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 табл. 3. Все наборы переменных в таблице упорядочены по возрастанию числового двоичного кода, соответствующего этим наборам. Коды наборов могут быть представлены в восьмеричной, шестнадцатеричной или даже десятичной (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*** ($\mathbf{F}_{12}$);

– ***конституента 0*** ($\mathbf{\Phi}_{14}$);

– функция общего вида ($\mathbf{F}$).

В табл. 3 ***конституенты 1*** обозначаются $\mathbf{F}_j$, где j – восьмеричный код набора переменных, единственного, на котором конституента равна 1.

Конституенты 0 обозначаются $\mathbf{\Phi}_p$, где p – восьмеричный код набора переменных, единственного, на котором конституента равна 0. Общее количество конституент 1 и общее количество конституент 0 равны по $\mathbf{2}^n$, где n – количество булевых переменных в наборе.

Табличное описание ФАЛ и систем ФАЛ является простым и наглядным, однако, весьма громоздким для практического использования. Это описание практически можно применять для булевых функций не более, чем от четырёх–пяти входных переменных. Громоздкость табличного описания ФАЛ потребовала разработки и

Таблица 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

других способов описания (или задания). Выбор способа описания ФАЛ выбирается в соответствии с задачами синтеза принципиальных схем, реализующих соответствующие ФАЛ.

1.3.2. Аналитическое описание булевых функций

На примерах описания ФАЛ, приведенных в табл.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:

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

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)***

$$\mathbf{F = F_p \vee F_k \vee \ldots \vee F_z}, \qquad (17)$$

где $\mathbf{F_p}$, $\mathbf{F_k}$,..., $\mathbf{F_z}$ – 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)***

$$\mathbf{\Phi = \Phi_d \,\&\, \Phi_t \,\&\, \ldots \Phi_y}, \qquad (18)$$

где $\mathbf{\Phi_d}$, $\mathbf{\Phi_t}$,..., $\mathbf{\Phi_y}$ – конституенты 0. В контексте аналитической записи ФАЛ в СКНФ все дизъюнктивные термы имеют максимальный ранг и называются ***мак-стермами ранга n*.**

ФАЛ общего вида, приведенная в табл. 3, записывается в СДНФ как

$$\mathbf{F} = \bar{x}_3 \bar{x}_2 x_1 \bar{x}_0 \vee \bar{x}_3 x_2 \bar{x}_1 \bar{x}_0 \vee \bar{x}_3 x_2 \bar{x}_1 x_0 \vee \vee \bar{x}_3 x_2 x_1 x_0 \vee x_3 \bar{x}_2 x_1 \bar{x}_0 \vee x_3 x_2 \bar{x}_1 x_0. \qquad (19)$$

В СКНФ эта же ФАЛ записывается как

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

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

1.3.3. Числовая форма представления булевых функций

Используя представление наборов переменных как числовых двоичных или восьмеричных кодов, запишем и минтермы функции **F** (таблица 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.

1.3.4. Графическая форма представления булевых функций

График функция **F**, приведенной в табл. 3, представлен на рис.3.

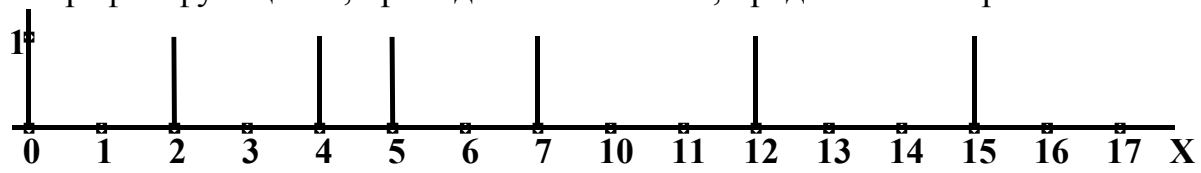


Рис.3. График булевой функции

График конституенты 1 $\mathbf{F_{12}}$ (табл. 3) представлен на рис.4.

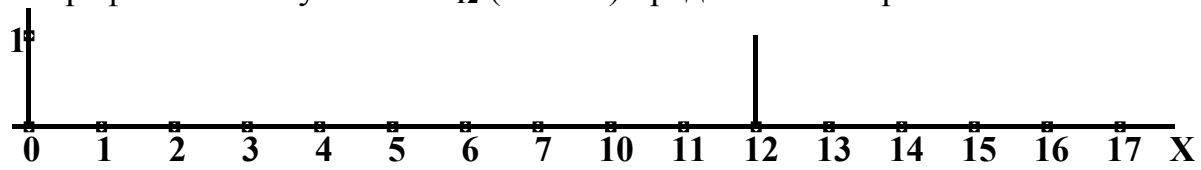


Рис.4. График конституенты 1 F_{12}

График конституенты 0 $\mathbf{\Phi_{14}}$ (таблица 3) представлен на рис.5.

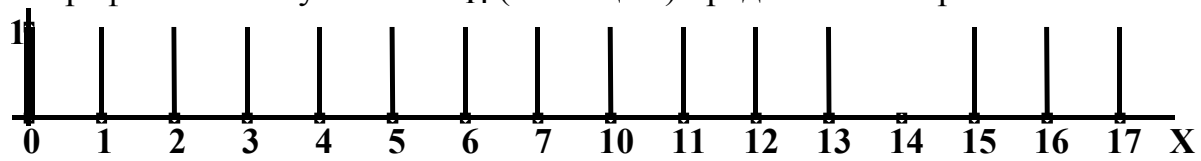


Рис.5. График конституенты 0 Φ_{14}

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

1.3.5. Геометрическое представление булевых функций

В геометрическом представлении ФАЛ значения входных переменных $\boldsymbol{n}$ – местного набора интерпретируются как координаты в $\boldsymbol{n}$ – мерной декартовой системе координат. Координатная система, в которой располагается геометрическая модель ФАЛ, является $\boldsymbol{n}$ – мерным кубом с единичными рёбрами. Кодам наборов входных переменных соответствуют вершины $\boldsymbol{n}$ – мерного куба.

Значения кодов входных наборов переменных, на которых заданная ФАЛ равна единице (при использовании СДНФ для представления ФАЛ) помечаются нечисловыми символами и такой $\boldsymbol{n}$ – мерный куб с помеченными некоторыми вершинами и является геометрической моделью заданной ФАЛ. На рис.6 представлен пример геометрической модели функции двух переменных

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

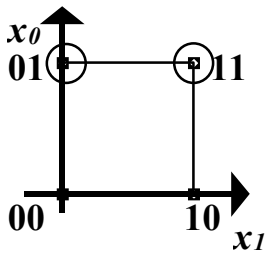


Рис.6. Геометрическая модель функции двух переменных

На рис.7 представлена геометрическая модель ФАЛ трёх переменных

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

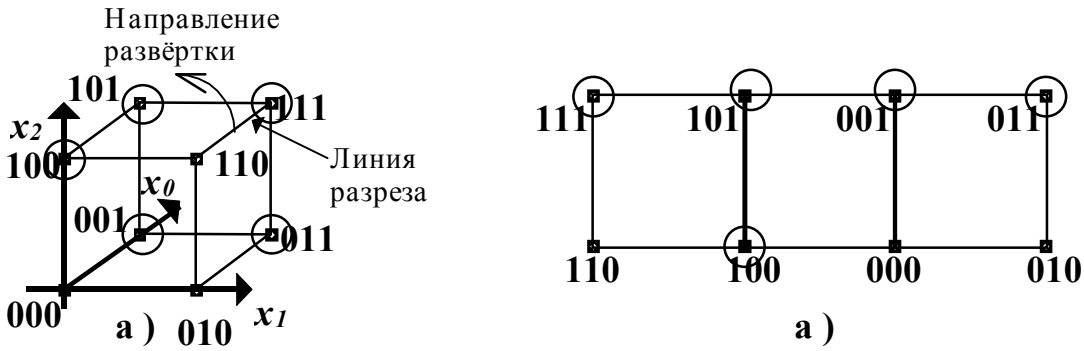


Рис.7. Геометрическая модель функции трёх переменных

На рис.7а изображена модель функции трёх переменных в трёхмерном пространстве, а на рис.7б – 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 и 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 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.

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 незна-

чащие 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.

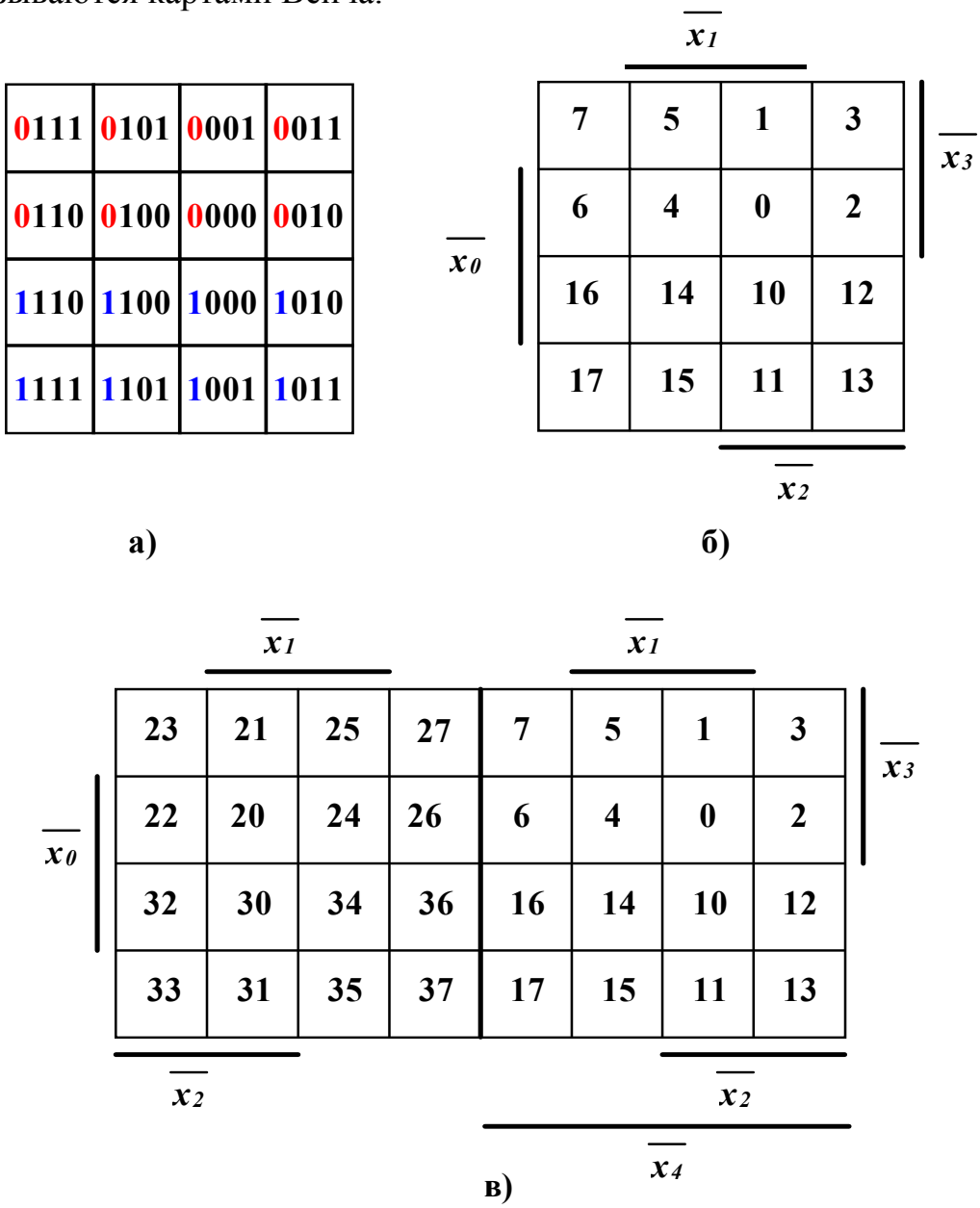


Рис.8. Пример построения карт Карно и Вейча для четырёх– и пятиместных наборов переменных

На рис.8 приведены примеры построения карт Карно и Вейча для четырёх– и пятиместных наборов переменных. На картах Вейча на рис.8б и рис.8в 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.9 приведена геометрическая модель ***частично определённой ФАЛ*** пяти пе-

ременных 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. На практике функции алгебры логики являются частично определёнными, так как не существует технической возможности появления некоторых наборов переменных, называемых ***запрещёнными***.

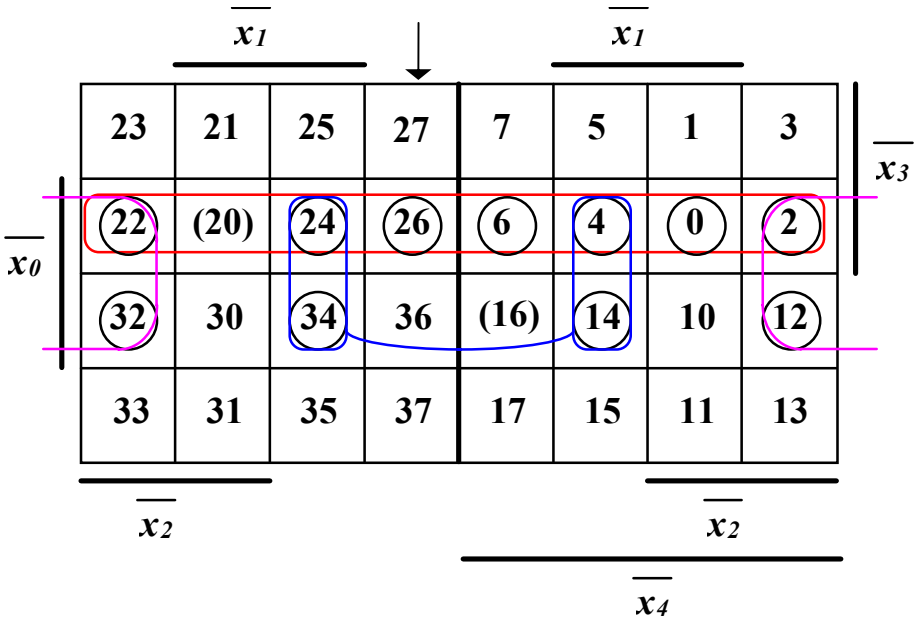


Рис.9. Диаграмма Вейча ФАЛ пяти переменных:

F= 0∨2∨4∨6∨12∨14∨(16)∨(20)∨22∨24∨26∨32∨34 ;

Коды наборов восьмеричные. На **(*)** наборах функция не определена.

На рис.9 отмечены поля из соседних пар наборов переменных, на которых ФАЛ не определена или имеет единичные значения.

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.9 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.

Это обстоятельство ограничивает применения карт Карно и Вейча для практического использования 6–7 – входными переменными.

1.4. Минимизация функций алгебры логики

1.4.1. Минимизация с помощью минимизирующих карт

Как было отмечено выше, одним из способов представления ФАЛ от небольшого числа переменных (обычно не больше 7) 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 значения ФАЛ равны 1 (или не определены). 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.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.9, записывается в дизъюнктивной нормальной форме (ДНФ) как

$$\mathbf{F}=\overline{x_3}\,\overline{x_0} \vee \overline{x_2} x_1 \overline{x_0} \vee x_2 \overline{x_1}\,\overline{x_0}. \qquad (24)$$

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

Из рис.9 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.

1.4.2. 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

$$\mathbf{F}= (2)\vee 3\vee 4\vee 5\vee 7\vee 11\vee 13\vee 14\vee 15\vee (17)=$$

$$= (\bar{x}_3 \bar{x}_2 x_1 \bar{x}_0) \vee \bar{x}_3 \bar{x}_2 x_1 x_0 \vee \bar{x}_3 x_2 \bar{x}_1 \bar{x}_0 \vee \bar{x}_3 x_2 \bar{x}_1 x_0 \vee \bar{x}_3 x_2 x_1 x_0 \vee x_3 \bar{x}_2 \bar{x}_1 x_0 \vee$$

$$\vee x_3 \bar{x}_2 x_1 x_0 \vee x_3 x_2 \bar{x}_1 \bar{x}_0 \vee x_3 x_2 \bar{x}_1 x_0 \vee (x_3 x_2 x_1 x_0). \qquad (26)$$

Э т а п 1. ***Нахождение первичных импликант*****.** 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.

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.

Таблица 4

Термы	$(\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0)$	$\bar{x}_3\bar{x}_2\bar{x}_1x_0$	$\bar{x}_3\bar{x}_2x_1\bar{x}_0$	$\bar{x}_3\bar{x}_2x_1x_0$	$\bar{x}_3x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_1x_0$	$\bar{x}_3x_2x_1\bar{x}_0$	$\bar{x}_3x_2x_1x_0$	$x_3\bar{x}_2\bar{x}_1\bar{x}_0$	$(x_3\bar{x}_2\bar{x}_1x_0)$
$(\bar{x}_3\bar{x}_2x_1\bar{x}_0)$	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.

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 табл. 4 заносятся в табл. 5.

Заголовочные строка 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).

Из табл. 5 видно, что дальнейшее склеивание импликант второго ранга невозможно и, следовательно, в клетках табл. 5 находятся первичные импликанты. К первичным импликантам относится и терм в первой строке (он же – в первом столбце), поскольку он не подвергся склеиванию на первом этапе.

Таблица 5

Термы	$\bar{x}_3\bar{x}_2x_1$	$\bar{x}_3x_1x_0$	$\bar{x}_2x_1x_0$	$\bar{x}_3x_2\bar{x}_1$	$x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_0$	$x_2\bar{x}_1x_0$	$x_2x_1x_0$	$x_3\bar{x}_2x_0$	$x_3\bar{x}_1x_0$	$x_3x_1x_0$	$x_3x_2\bar{x}_1$	$x_3x_2x_0$
$\bar{x}_3\bar{x}_2x_1$	X												
$\bar{x}_3x_1x_0$		X									x_1x_0		
$\bar{x}_2x_1x_0$			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					
$x_2x_1x_0$								X					
$x_3\bar{x}_2x_0$									X				x_3x_0
$x_3\bar{x}_1x_0$										X	x_3x_0		
$x_3x_1x_0$											X		
$x_3x_2\bar{x}_1$												X	
$x_3x_2x_0$													X

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 какая–нибудь нечисловая метка, например, ♥ (табл. 6). Поскольку не подвергшиеся склеиванию на первом этапе минтермы являются одновременно и первичными импликантами и, следовательно, включаются в табл. 6 и в строки и в столбцы, то все столбцы этой таблицы будут помечены различным количеством меток, находящихся в различных строках. Особый интерес представляют столбцы, помеченные единственной меткой.

Таблица 6

Термы	$(\bar{x}_3\bar{x}_2x_1\bar{x}_0)$	$\bar{x}_3\bar{x}_2x_1x_0$	$\bar{x}_3x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2\bar{x}_1x_0$	$\bar{x}_3x_2x_1x_0$	$x_3\bar{x}_2\bar{x}_1x_0$	$x_3\bar{x}_2x_1x_0$	$x_3x_2\bar{x}_1\bar{x}_0$	$x_3x_2\bar{x}_1x_0$	$(x_3x_2x_1x_0)$
x_1x_0		♥[3]			♥		♥			♥
$x_2\bar{x}_1$			♥[c]	♥				♥[c]	♥	
x_2x_0				♥	♥				♥	♥
x_3x_0						♥[c]	♥		♥	♥
$\bar{x}_3\bar{x}_2x_1$	♥[c]	♥								

Э т а п 3. ***Нахождение существенных импликант*****.** Если в каком-либо 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. 6 существенные импликанты отмечены знаком ♥^{c}. По табл. 6 видно, что пятая первичная импликанта $\bar{x}_3\bar{x}_2x_1$ является существенной только для неопределённого минтерма $\bar{x}_3\bar{x}_2x_1\bar{x}_0$. Если придать СДНФ исходной ФАЛ нулевое значение на наборе переменных $\bar{x}_3\bar{x}_2x_1\bar{x}_0$, то пятую первичную импликанту нужно исключить, тогда первая импликанта (она отмечена знаком ♥^{3}) станет существенной. На наборе $x_3x_2x_1x_0$ 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. 7. В эту таблицу включаются оставшиеся столбцы и соответствующие им 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 примере ФАЛ такого случая нет.

Существенные первичные импликанты в ячейках табл. 7 помечены знаком ♥^{c}.

Таблица 7

Термы	$\bar{x}_3x_2\bar{x}_1\bar{x}_0$	$\bar{x}_3x_2x_1x_0$	$x_3\bar{x}_2x_1x_0$	$x_3x_2\bar{x}_1x_0$	$(x_3x_2x_1x_0)$
♥^{c} x_1x_0		♥	♥		♥
♥^{c} $x_2\bar{x}_1$	♥			♥	
x_2x_0	♥	♥		♥	♥
♥^{c} x_3x_0			♥	♥	♥

Э т а п 5. ***Вычёркивание лишних первичных импликант***. Если после вычёркивания лишних столбцов на этапе 4 в табл. 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 рас-

смотрении минтермы. После выполнения этапов 4 и 5 рассматриваемая табл. 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

$$\mathbf{F} = x_1 x_0 \vee x_2 \bar{x}_1 \vee x_3 x_0 . \qquad (27)$$

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

1.4.3. 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:

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

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 ♣.

В заголовочных строках табл. 8, 9 и 10 нет двоичных кодов минтермов, которые не подверглись склеиванию. Значит первичных импликант среди минтермов мини-

мизируемой ФАЛ нет. Минтермы с кодами (**20**), (**21**) (**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 табл. 8, 9 и 10.

Таблица 8

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

Таблица 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 10

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

В ячейках табл. 8 находятся импликанты группы 1, в ячейках табл. 9 находятся импликанты группы 2, в ячейках табл. 10 находятся импликанты группы 3.

В табл. 11 и 12 произведено сопоставление импликант соседних групп.

Таблица 11

Импли-канты	**00♣11**	**001♣1**	**0♣011**	**010♣1**	**0♣101**	**0110♣**	**01♣01**
0001♣							
0010♣						**0♣10♣**	
0♣100					**0♣10♣**		

Таблица 12

Импли-канты	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			

Примечание:

Первичные импликанты – **0001♣ 0♣10♣ 0♣♣11 0♣1♣1 01♣♣1**

Э т а п 2. Расстановка меток.

Э т а п 3. Нахождение существенных импликант.

Существенные импликанты в таблице 13 отмечены знаком ♥^{c}. 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}, 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.

Таблица 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 6. Выбирается минимальное покрытие минтермов СДНФ ФАЛ существенными первичными импликантами и записывается единственно возможный вариант минимальной ФАЛ:

$$\mathbf{F = 0♣10♣ \vee 0♣♣11 \vee 01♣♣1} = \bar{x}_4 x_2 \bar{x}_1 \vee \bar{x}_4 x_1 x_0 \vee \bar{x}_4 x_3 x_0 . \qquad (29)$$

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 логических 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

$$\mathbf{F} = x_1 x_0 \vee x_2 \bar{x}_1 \vee x_3 x_0; \qquad (30)$$

1.5. Элементная база для построения комбинационных схем

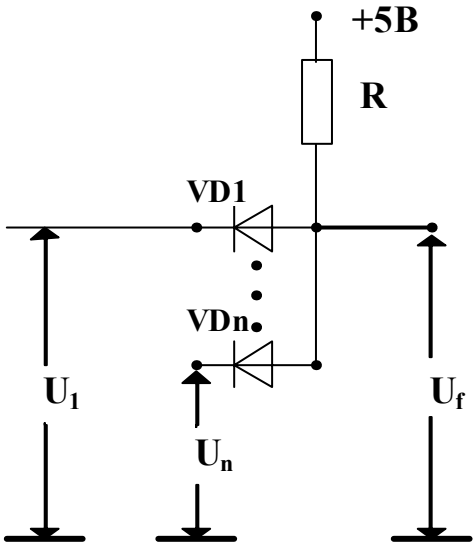
Для технической реализации Функций Алгебры Логики, то есть для построения устройства, входные и выходные сигналы которого связаны между собой такой же функциональной зависимостью, как и переменные в ФАЛ с соответствующими выходными значениями, применяются электронные (иногда и неэлектронные – оптические, пневматические, гидравлические, механические) узлы – логические элементы. Обычно логические элементы соответствуют простейшим логическим операциям – конъюнкции (**И**), дизъюнкции (**ИЛИ**), инверсии (**НЕ**), сложению по модулю 2 (**исключающее ИЛИ**) и другим.

1.5.1. Логические элементы **И**, **ИЛИ**, **НЕ**

1.5.1.1. Логические элементы **И** и **И–НЕ**

(Позитивная логика)

Схема логического элемента **И**, построенного на полупроводниковых диодах и резисторе, приведена на рис.10а.



а)

U_2	U_1	U_f
L	L	L
L	H	L
H	L	L
H	H	H

б)

Рис.10. Схема логического элемента **И**

По схеме логического элемента видно, что выходное напряжение $\mathbf{U_f}$, будет иметь малую величину, равную напряжению на полупроводниковом диоде (примерно 0,5 В), если хотя бы одно напряжение на входах $\mathbf{U_1,...,U_n}$ будет равно нулю. Для двухвходового варианта логического элемента на рис.10б приведена таблица истинности для физических уровней выходного напряжения этой схемы (**L(ow)** – низкий уровень напряжения, **H(igh)** – высокий уровень).

В, так называемой, позитивной логике низкому уровню напряжения соответствует логический 0 (лог.0), высокому уровню – логическая единица (лог.1). В этом случае описание логического элемента таблицей истинности соответствует описанию конъюнктора.

В, так называемой, негативной логике низкому уровню напряжения соответствует логическая 1 (лог.1), высокому уровню – логический нуль (лог.0). В этом случае описание логического элемента таблицей истинности соответствует описанию дизъюнктора.

Техническая простота и низкая стоимость диодно–резисторной схемы логического элемента **И** способствует её широкому применению на практике.

Существенным недостатком этой схемы является невозможность последовательного включения этих элементов. Выходное напряжение лог.0 (для позитивной логики) при нулевых входных напряжениях не равно нулю, а равно падению напряжения на одном открытом полупроводниковом диоде.

При подаче выходного напряжения на вход следующей логической схемы, низкое выходное напряжения второго логического элемента, которое должно соответ-

ствовать логическому нулю, будет равно сумме напряжений на открытых диодах. Таким образом, с увеличением количества последовательно включенных логических элементов, теряется различие между уровнями лог.0 и лог.1 и вся схема становится неработоспособной. Этот недостаток ликвидируется включением на выходе каждого диодно–резисторного логического элемента нормирующего усилителя, обычно с использованием транзисторов, включаемых по схеме с “общим эмиттером” или с “общим истоком” и работающих в ключевом режиме. Такие нормирующие усилители инвертируют свой входной сигнал, то есть являются логическими элементами **НЕ**. Схема логического диодно–резисторного элемента **И–НЕ** (элемента Шеффера) приведена на рис.11.

При последовательном (каскадном) соединении этих логических элементов входные и выходные напряжения логических элементов всегда имеют одинаковые уровни лог.0 и лог.1, что позволяет последовательно включать неограниченное количество логических элементов этого типа.

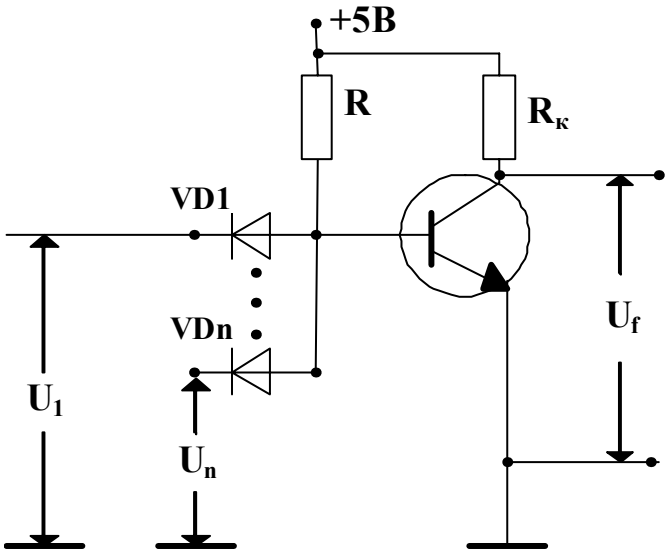


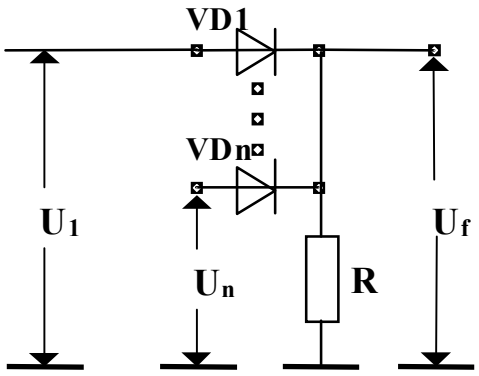
Рис.11. Схема логического элемента **И–НЕ**

1.5.1.2. Логические элементы **ИЛИ, ИЛИ–НЕ**

(Позитивная логика)

Схема логического элемента **ИЛИ**, построенного на полупроводниковых диодах и резисторе, приведена на рис.12а.

По схеме логического элемента видно, что выходное напряжение $\mathbf{U_f}$, будет иметь величину, меньшую на величину напряжения на полупроводниковом диоде (примерно 0,5 В), чем напряжение на входах $\mathbf{U_1}$,...,$\mathbf{U_n}$, равное лог.1. Для двухвходового варианта логического элемента **ИЛИ** на рис.12б приведена таблица истинности для физических уровней выходного напряжения этой схемы (**L(ow)** – низкий уровень, **H(igh)** – высокий уровень).



U_1	U_2	U_f
L	L	L
L	H	H
H	L	H
H	H	H

а) б)

Рис.12. Схема логического элемента **ИЛИ**

Таким образом, очевидно, что логический элемент **ИЛИ** также необходимо дополнить нормализующим усилителем для восстановления величины высокого напряжения на выходе. Схема диодно–резисторного элемента **ИЛИ–НЕ** (логического элемента Пирса) приведена на рис.13.

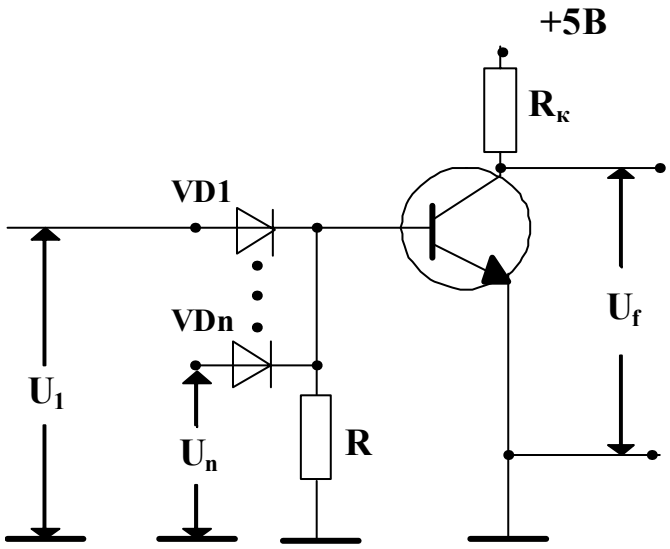


Рис.13. Схема логического элемента **ИЛИ–НЕ**

Условные обозначения логических элементов приведены на рис 14.

На рис.14в показан пример использования элемента **И–НЕ** в качестве инвертора, когда на все входы, кроме одного подаётся высокий уровень напряжения (лог.1), а на оставшийся вход подаётся инвертируемая переменная.

На рис.14г показан пример использования элемента **ИЛИ–НЕ** в качестве инвертора, когда на все входы, кроме одного подаётся низкий уровень напряжения (лог.0), а на оставшийся вход подаётся инвертируемая переменная. Это свойство логических элементов **И–НЕ** или логических схем **ИЛИ–НЕ** широко используется в реальных принципиальных схемах, так как позволяет уменьшить степень избыточности таких принципиальных схем, которые строятся с использованием интегральных микросхем, содержащих в одном корпусе несколько логических элементов.

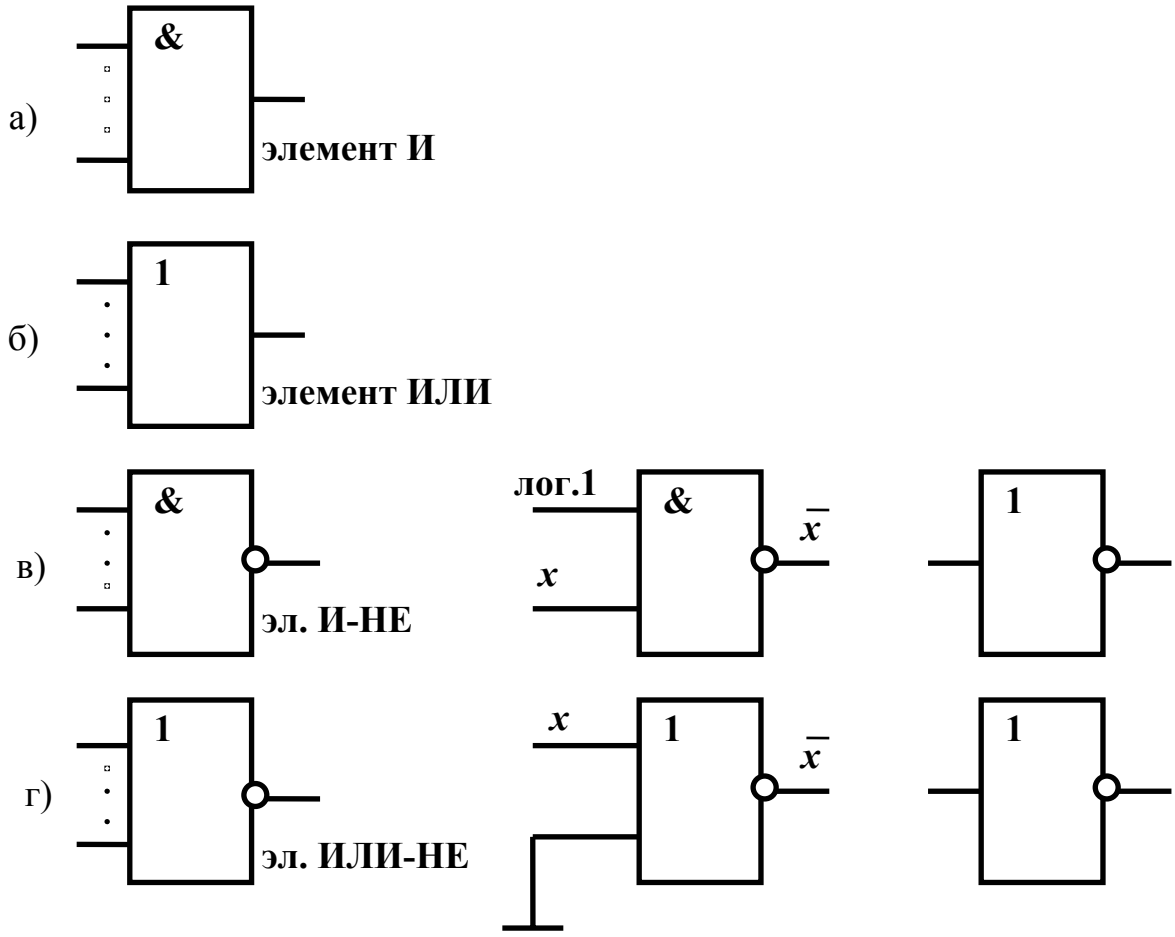
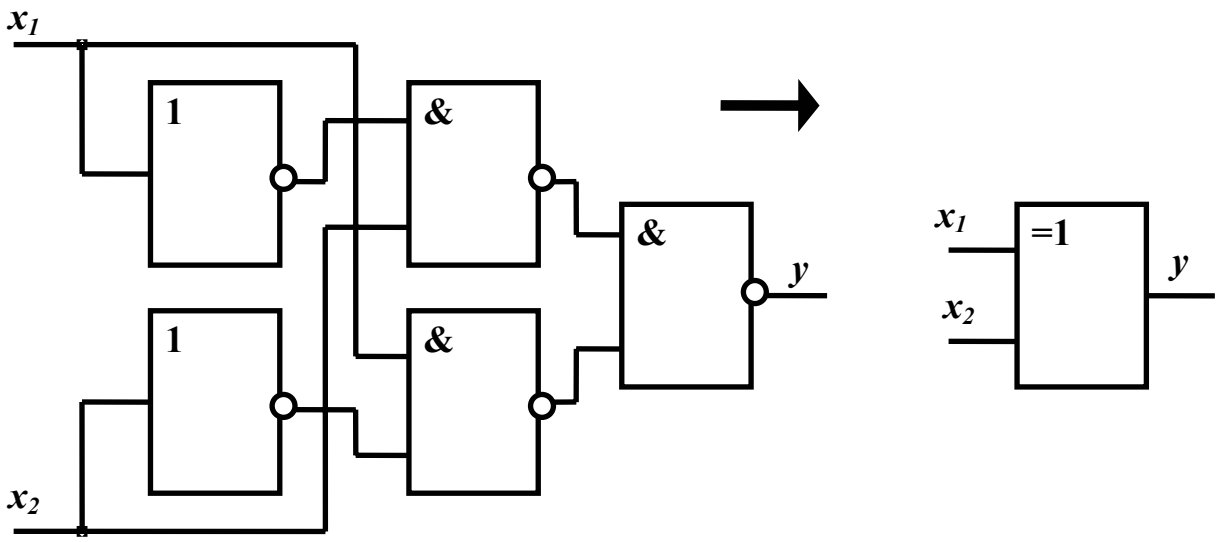


Рис.14. Условные обозначения базовых логических элементов

1.5.2. Примеры технической реализации булевых функций

1.5.2.1. Функция **ИСКЛЮЧАЮЩЕЕ–ИЛИ**

(Сложение по модулю 2)



а)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

б)

Рис.15. Схемная реализация функции **ИСКЛЮЧАЮЩЕЕ–ИЛИ**

ФАЛ **сложение по mod2** описывается уравнением:

$$y = x_1 \oplus x_2 = x_1\bar{x}_2 \vee \bar{x}_1 x_2; \qquad (31)$$

Таблица истинности этой ФАЛ приведена на рис.15б. Эта логическая функция используется для описания работы арифметико-логических устройств (АЛУ) процессоров и имеет отдельное применение как управляемый инвертор. Из таблицы истинности видно, что при фиксированном значении одной переменной, например, при $x_1 = 0$, выходная переменная y повторяет значения входной переменной x_2, а при

$x_1 = 1$ выходная переменная y имеет инверсные значения переменной x_2 . Таким образом, изменяя значения одной переменной, мы изменяем свойства логического элемента как технического устройства.

1.5.2.2. Минимизированная функция алгебры логики
(Дешифратор второго рода)

ФАЛ, имеющая значения, определённые на нескольких наборах входных переменных этим отличается от дешифраторов, имеющих выходные значения, определённые только на одном наборе входных переменных. Поэтому она иногда называется дешифратором второго рода. Рассмотрим принципиальную схему, соответствующую ФАЛ, описываемую ф. (27)

Минимизированная ФАЛ, рассмотренная выше, записана в аналитической форме

$$\mathbf{F} = x_1 x_0 \vee x_2 \overline{x}_1 \vee x_3 x_0 .$$

По этой формуле тотчас же можно изобразить принципиальную схему устройства, реализующего эту ФАЛ (рис.16).

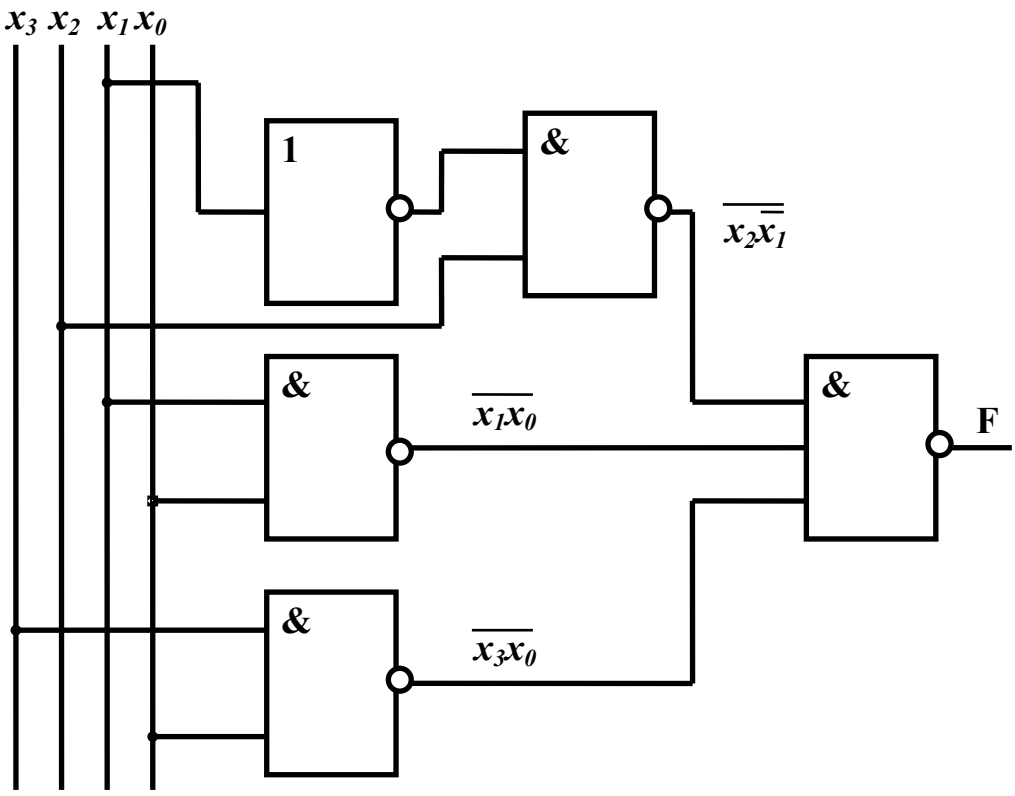


Рис.16. Принципиальная схема физической реализации ФАЛ

1.5.3 Программируемые логические матрицы (ПЛМ)

Программируемая логическая матрица [2] представляет собой функциональный блок, созданный на базе интегральной полупроводниковой технологии и предназначенный для реализации логических схем цифровых систем. В зависимости от внутренней организации ПЛМ можно разделить на ПЛМ комбинационного типа и ПЛМ с памятью. ПЛМ второго типа используются для построения цифровых автоматов по рис.2 на основе всего лишь одной микросхемы. ПЛМ первого типа строятся по двухуровневой структуре рис.17а. Условное обозначение ПЛМ с указанием параметров **s, t** и **q** приведено на рис.17б.

Матрица конъюнкций (МК) имеет s входов и содержит в себе **q** конъюнкций, следовательно, имеет **q** выходов. Эта матрица позволяет реализовать до **q** элементарных конъюнкций $\mathbf{P_1,...,P_q}$ переменных $x_1,...,x_S$, поступающих на её входы. Матрица дизъюнкций (МД) имеет **q** входов и **t** выходов. Она позволяет реализовать до **t** ФАЛ, записанных в ДНФ.

На одной ПЛМ(**s, t, q**) может быть реализована система булевых функций:

$$y_1(x_S, ... ,x_1), ... , y_t(x_S, ... ,x_1),$$

имеющих $\mathbf{B \le q}$, где **B** суммарное количество различных элементарных конъюнкций в системе функций y_i .

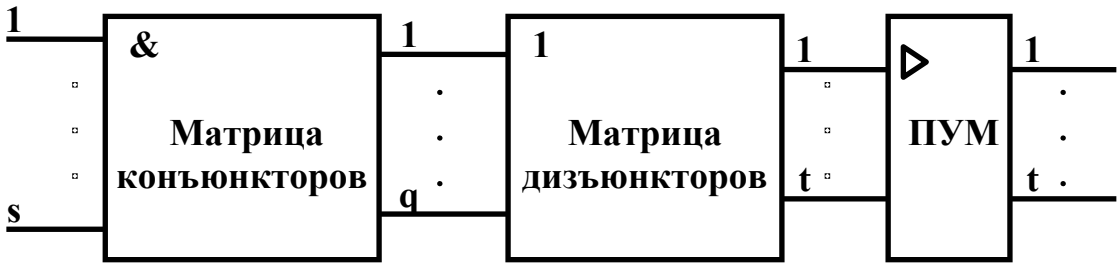
1.5.3.1. Примеры ПЛМ

Промышленностью выпускаются микросхемы ПЛМ 556РТ1 и 556РТ2, отличающиеся параметрами выходных программируемых усилителей. Микросхема 556РТ1 имеет выход с открытым коллектором, 556РТ2 – с тремя состояниями. Условные обозначения микросхем приведены на рис.18.

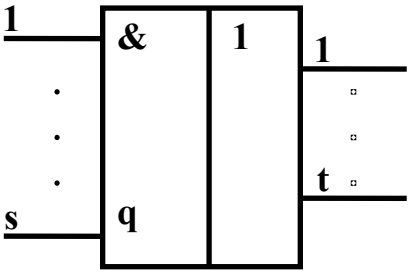
Логическая схема ПЛМ приведена на рис.19 [3].

Микросхемы реализуют восемь функций от 16 входных переменных. При этом логические функции представляются в дизъюнктивной нормальной форме и общее число конъюнкций для всех функций не должно превышать 48. Микросхемы имеют инверсный вход разрешения выборки кристалла **CE** и линию разрешения программирования **FE**.

Логическая схема ПЛМ содержит два уровня программируемой логики.



а)



б)

Рис.17. Структура программируемой логической матрицы

s – количество входов переменных;
q – количество конъюнкций в матрице;
t – количество функций в корпусе микросхемы.
ПУМ – слой программируемых усилителей мощности.

Первый логический уровень включает матрицу из 48 схем **И** (конъюнкторов), организующую логические произведения (конъюнкции) $\mathbf{P_i}$ от входных переменных $\mathbf{A_m}$ и их инверсий $\overline{\mathbf{A}}_\mathbf{m}$. Инвертирование логических переменных A_m осуществляется входными буферными усилителями. В исходном состоянии все плавкие связи сохранены ($\alpha_{im}=\boldsymbol{\alpha}^*_{im}=1$, i={0,47} m={0,15}), что обеспечивает нулевое значений всех конъюнкций. С помощью выплавляемых связей α_{im} и $\boldsymbol{\alpha}^*_{im}$ каждый i–конънктор может быть соединён либо с входной переменной A_m либо с её инверсией $\overline{\mathbf{A}}_\mathbf{m}$.

Удаление перемычек ($\alpha_{im}=\boldsymbol{\alpha}^*_{im}=0$) устанавливает независимость конъюнкции P_i от переменной A_m. *Не допускается в рабочем i–конъюнкторе ПЛМ оставлять одновременно обе парные перемычки* ($\alpha_{im}=\boldsymbol{\alpha}^*_{im}=1$) *для неиспользуемого входа* A_m, *так как это влечёт за собой тождественно нулевое значение* P_i . Все перемычки резервных схем И, как правило, оставляют целыми.

Второй логический уровень образует матрица из восьми 48-входовых схем **ИЛИ** (дизъюнкторов), по одной на каждый выход ПЛМ.

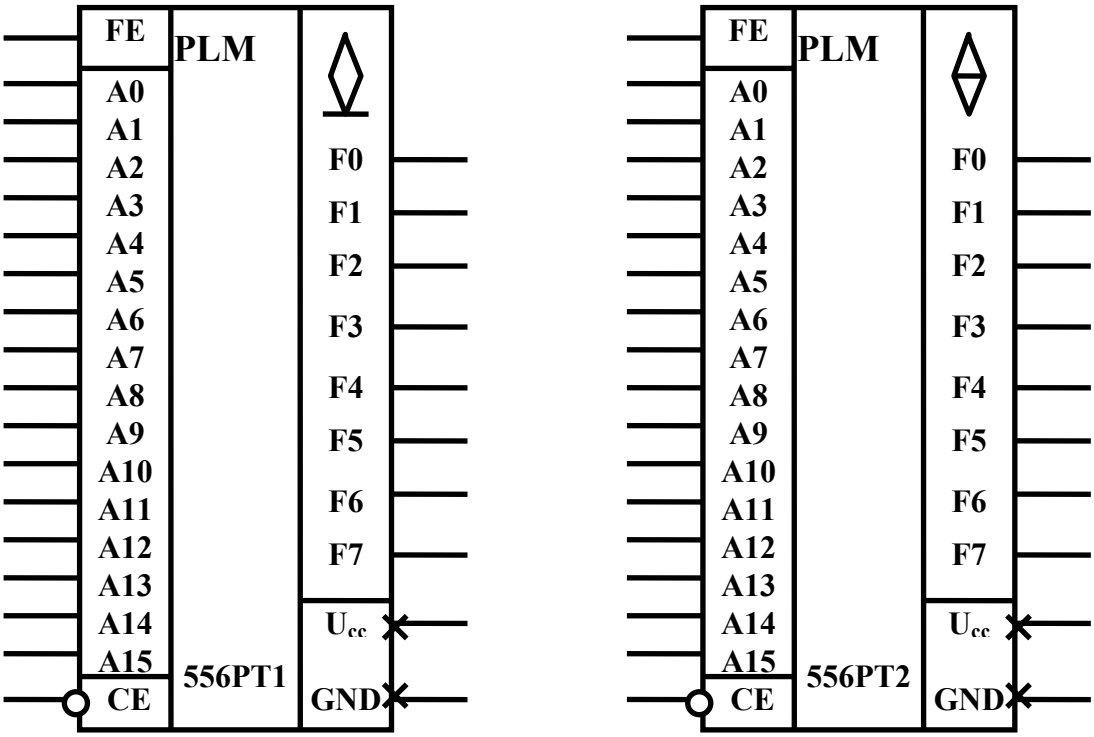


Рис.18. Условные обозначения ПЛМ

Плавкие перемычки β_{ji} позволяют организовать выборочную дизъюнкцию конъюнкций $\mathbf{P_i}$. В исходном состоянии, когда все перемычки целы **(β_{ji}=1, j={0,7}, i={0,47})** и $\mathbf{P_i}$=0, все дизъюнкции $\mathbf{S_j}$ =0. В рабочей ПЛМ допускается оставлять перемычки β_{ji} для каждого неиспользуемого (резервного) **i** – конъюнктора, если его выход равен нулю. Эта связь может в дальнейшем потребоваться при редактировании рабочей ПЛМ, так же как и все другие резервные связи микросхемы.

На выходах схем **ИЛИ** находится слой программируемых инверторов, построенный на схемах двухвходовых логических элементов **ИСКЛЮЧАЮЩЕЕ ИЛИ**, и ряд буферных элементов, открываемых сигналом **CE.** Удалением перемычек γ_j можно выборочно изменить значение уровня выходного сигнала. В исходном состоянии γ_j =1 и при выбранной схеме (**CE**=0), на всех выходах считывается нуль.

1.5.3.2 Процедуры программирования ПЛМ

Физическим принципом программирования ПЛМ является пережигание (испарение) выбранных перемычек.

В процессе эксплуатации ПЛМ могут возникнуть следующие задачи:

– начальное программирование “чистой” ПЛМ;

– повторное программирование (редактирование) ПЛМ.

Выполнение этих задач основано на использовании ряда простейших процедур (примитивов) программирования ПЛМ:

– прожиг связей матрицы **И**;

– контроль связей матрицы **ИЛИ**;

– прожиг связей матрицы **И**;
– контроль связей матрицы **ИЛИ**;
– прожиг связей слоя **НЕ**;
– контроль связей слоя **НЕ.**

Для реализации процедур прожига и контроля в состав ПЛМ введён ряд дополнительных элементов, изображённых на структурной схеме рис.20.

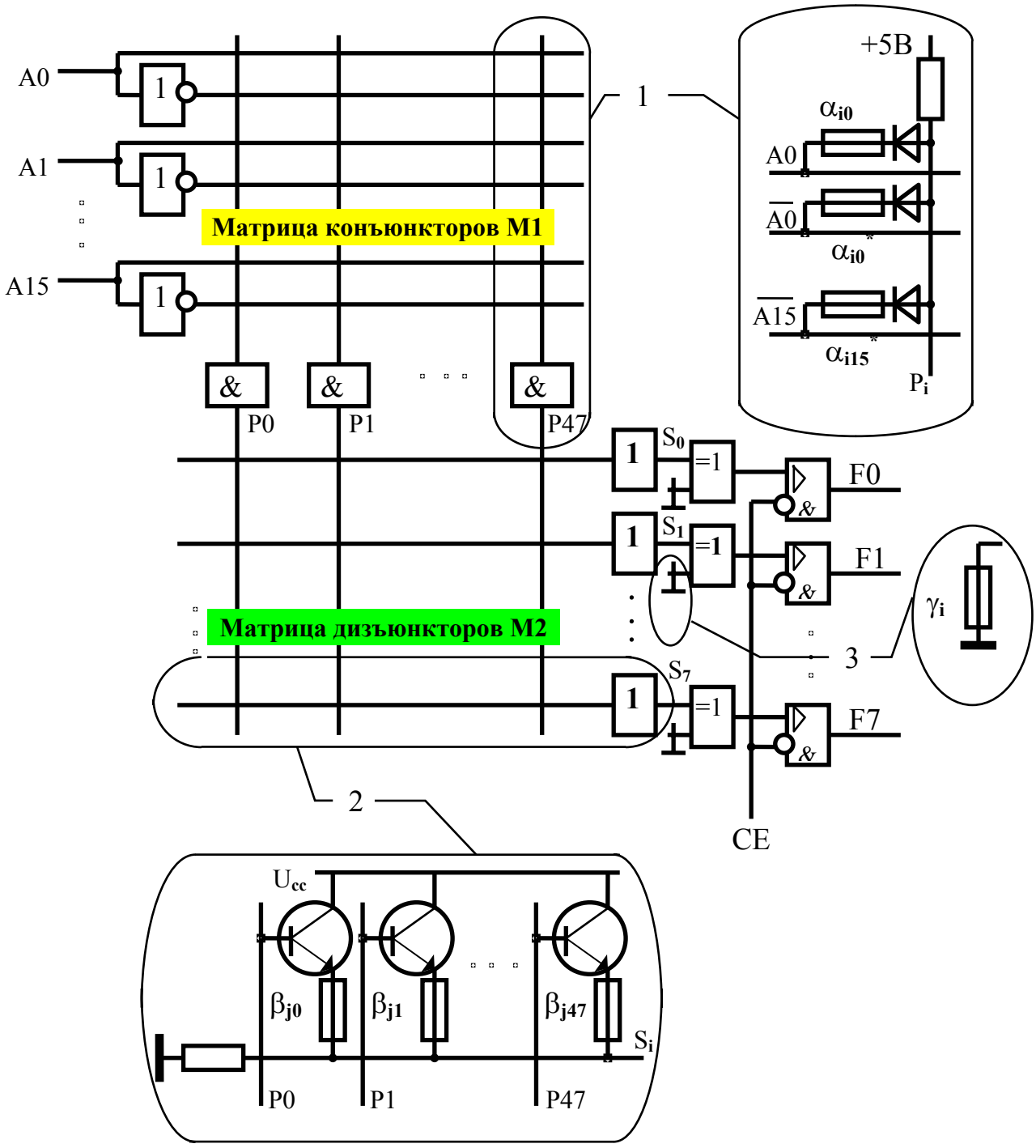


Рис.19. Логическая схема ПЛМ

Программирующий дешифратор и адресные формирователи **АФ** работают только в режимах программирования и контроля. Адресные формирователи управляют

дешифратором, определяя адрес выбранной конъюнкции $\mathbf{P_i}$. Формирователь **АФ1** используется при работе с матрицей **И, АФ2** – при программировании матрицы **ИЛИ**. В состав **АФ** и программирующего дешифратора входят также схемы, обеспечивающие их включение и выключение в зависимости от режима работы. Эти пороговые схемы управляются напряжениями на входах $\mathbf{U_{cc}}$ **, FE и CE** микросхемы.

Начальное состояние напряжений на внешних выводах микросхемы:

– **GND** = 0В;
– $\mathbf{U_{cc} = U_0}$;
– $\mathbf{FE = U_0}$;
– $\mathbf{CE = U_1}$;
– $\mathbf{A15...A0 = U_1}$;
– **F7...F0** = 5 В через 10 кОм.

$\mathbf{U_0}$ и $\mathbf{U_1}$ – стандартные уровни Транзисторно–Транзисторной Логики (ТТЛ) для напряжений “лог.0” и “лог.1” соответственно.

Программирование и контроль матрицы И. Режим реализуется при напряжении питания $\mathbf{U_{cc}}$ = 5В и отключенных выходных каскадах (**CE** = 0В). Если **CE**=10В, то открывается **АФ1** и программирующий дешифратор выбирает сборку **И**, указанную кодом **F5...F0**. Для удаления требуемой плавкой перемычки необходимо закрыть все выходы входных усилителей (прямые и инверсные), кроме программируемого. Для этого на входы всех усилителей, кроме одного, подаётся напряжение 10В.

На вход выбранного усилителя подаётся напряжение $\mathbf{U_1}$, если требуется пережечь перемычку α^*_{im} и $\mathbf{U_0}$, если α_m. Импульс программирующего тока формируется при подаче на вход **FE** напряжения 17В.

В режиме контроля FE = 0В; при этом ток источника втекает через матрицу **И** при наличии проверяемой перемычки или через матрицу **ИЛИ** при её отсутствии. Схема контроля матрицы **И**, связанная с выходом **F7**, фиксирует наличие или отсутствие тока в матрице **ИЛИ**.

Программирование и контроль матрицы ИЛИ. Режим осуществляется при напряжении питания $\mathbf{U_{cc}}$ = 8,75В, которое разрешает работу **АФ2**. На входы **A5...A0** подаётся код, соответствующий номеру выбранного конъюнктора. На выход функции, из которой исключается выбранная конъюнкция, – напряжение 10В. Импульс программирующего тока, протекающий по адресованной таким способом перемычке, формируется при подаче на программирующий вход **FE** напряжения 17В, а на вход **CE** – 10В. Контроль записанной в матрицу **ИЛИ** информации выполняется аналогично, только при напряжении на входах **FE** и **CE**, равном $\mathbf{U_0}$.

О наличии или отсутствии проверяемой перемычки судят по уровню сигнала на выходе микросхемы.

Программирование и контроль слоя НЕ. Пережигание перемычки γ_j происходит при подаче на выход F_j напряжения 17В. При этом срабатывает схема программирования перемычки в выходном каскаде и через прожигаемую перемычку проходит разрушающий её ток. При контроле состояния перемычки на схему подаётся повышенное напряжение питания U_{cc} = 8,75В, а на адресные входы A5...A0 код $111\ 111_2$ в ТТЛ–уровнях. При этом ни одна из 48 конъюнкций не выбирается и, следовательно, ток в матрицу ИЛИ не поступает. По состоянию выхода F_j можно судить о целостности перемычки γ_j .

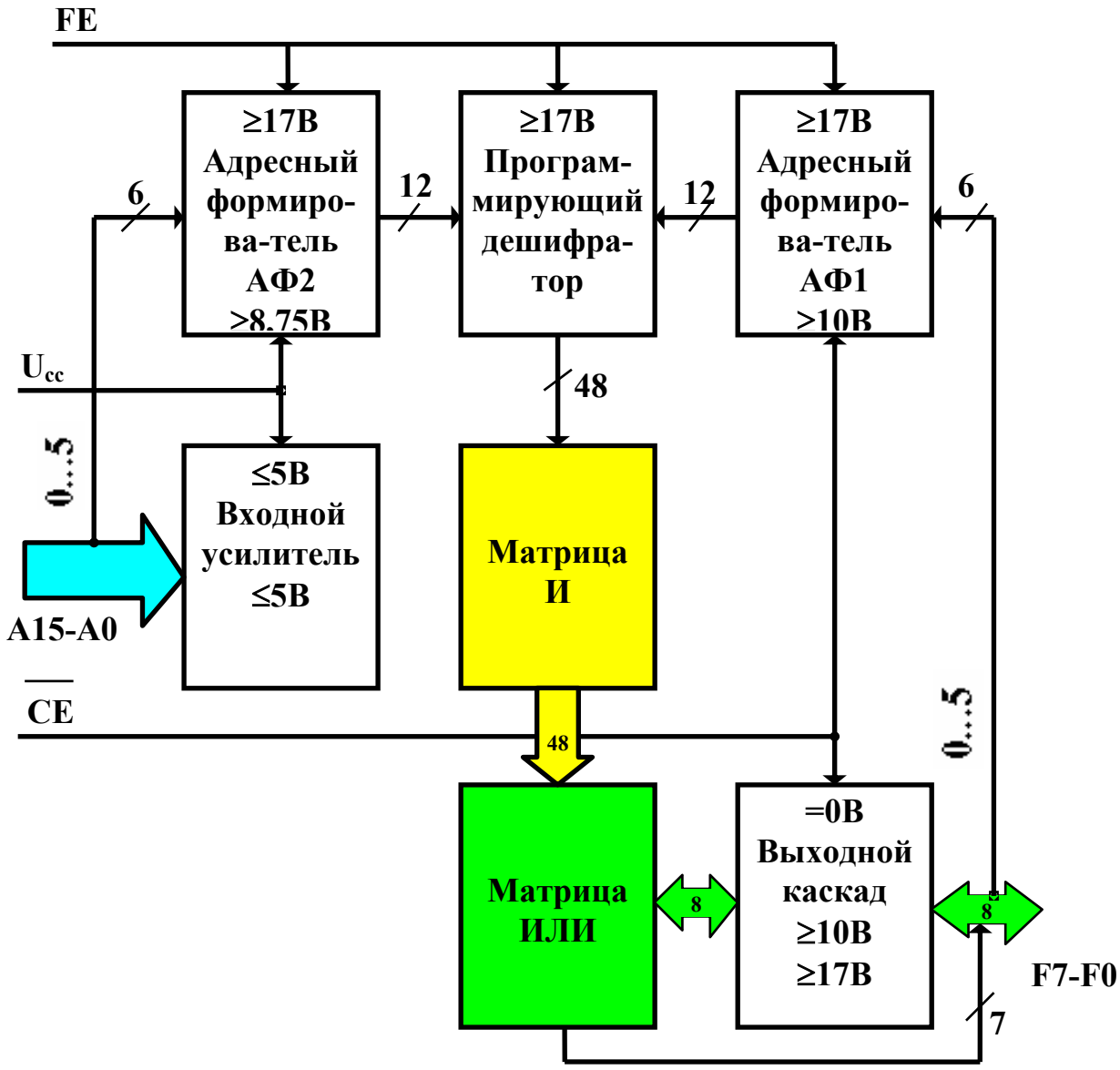


Рис.20. Структурная схема ПЛМ

Глава 2

СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

2.1. 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.

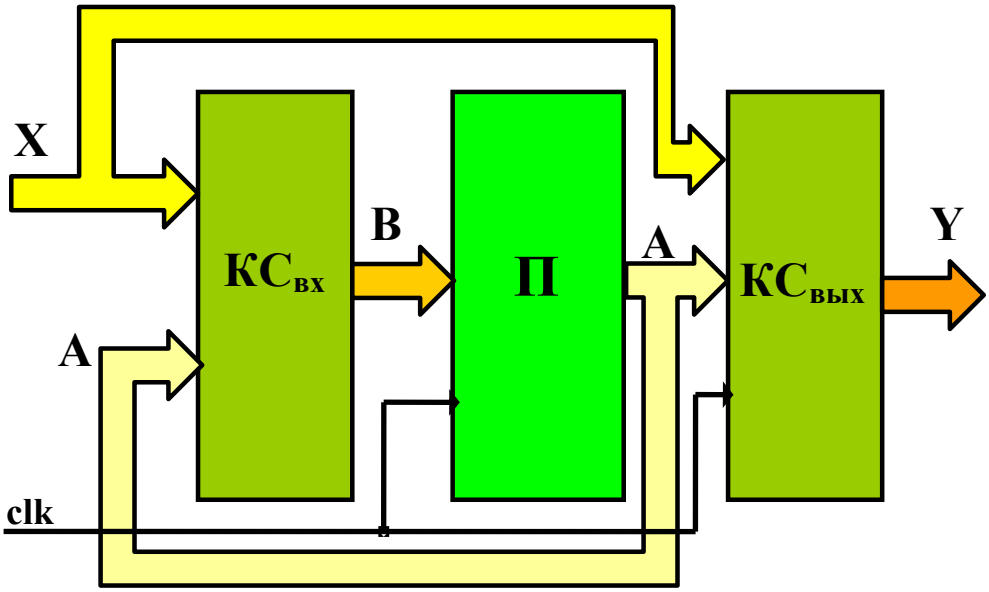


Рис. 21.Каноническая структурная схема цифрового автомата
КС$_{вх}$ – входная комбинационная схема;
П – память; **КС$_{вых}$** – выходная комбинационная схема;
X – входной цифровой код; **B** – код возбуждения памяти;
A – код состояния памяти; **Y** – выходной код;
clk – синхроимпульсы системы автоматного времени.

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 при-

водит к практической необходимости включения в информационную систему любой сложности единой системы синхронизации.

Для абстрактного математического описания цифрового автомата как кодопреобразователя (см. рис.1) используется его представление как шестиэлементного множества:

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

где

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

$\mathbf{X = \{z_1, ..., z_f, ..., z_F\}}$ – множество входных сигналов автомата (входной алфавит);

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

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

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

$\mathbf{a_1}$ – начальное состояние автомата $\mathbf{(a_1 \in A)}$**.** 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 $\mathbf{D_\delta = D_\lambda = A \times X}$. Иными словами, у полностью определённого автомата области определения функций δ и λ совпадают с множеством **A×X** – множеством всевозможных пар $\mathbf{(a_m, z_f)}$. У частичного автомата функции δ и λ определены не для всех пар $\mathbf{(a_m, z_f) \subseteq A \times X}$.

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:

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

Количество разрядов двоичного кода состояний

$$\mathbf{p =]\log_2 M[}. \qquad (33)$$

Количество разрядов двоичного кода входных сигналов

$$\mathbf{r =]\log_2 F[}. \qquad (34)$$

Количество разрядов двоичного кода выходных сигналов

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

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

Согласно структурной схеме рис.21 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) = δ[a(t), z(t)],

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

– при описании функционирования автомата выражениями:

a(t+1) = δ[a(t), z(t)],

w(t) = λ[a(t)] – он называется автоматом Мура.

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

2.2. Методы описания цифровых автоматов

Чтобы задать цифровой автомат **S**, необходимо описать все элементы множества **S = { A, X ,Y, δ, λ, 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. 14 и табл. 15. Строки этих таблиц соответствуют входным сигналам множества **X**, а столбцы – состояниям **A**, причём крайний левый столбец обозначен как начальное состояние инициального цифрового автомата $\mathbf{a_1}$. На пересечении столбца $\mathbf{a_m}$ и строки $\mathbf{z_f}$ в таблице переходов ставится состояние $\mathbf{a_s=\delta(a_m, z_f)}$, в которое автомат переходит из состояния $\mathbf{a_m}$ под действием сигнала $\mathbf{z_f}$. В таблице выходов – соответствующий этому переходу выходной сигнал $\mathbf{w_g=\lambda(a_m, z_f)}$.

Таблица 14

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	$\delta(a_1,z_F)$	...	$\delta(a_M,z_F)$

Таблица15

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	$\lambda(a_1,z_F)$	...	$\lambda(a_M,z_F)$

Пример табличного описания полностью определённого цифрового автомата Мили $\mathbf{S_1}$ с тремя состояниями, двумя входными и двумя выходными сигналами приведён в табл. 16 и табл. 17.

Таблица 16

Таблица переходов а. Мили $\mathbf{S_1}$

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 17

Таблица выходов а. Мили $\mathbf{S_1}$

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, табл. 18.

Таблица 18

Совмещённая таблица переходов и выходов а. Мили $\mathbf{S_1}$

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 табл. 19.

Таблица 19

Совмещённая таблица для частичного а. Мили $\mathbf{S_2}$

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 –

табл. 20, пример – табл. 21), в которой каждому её столбцу приписан, кроме состояния $\mathbf{a_m}$, ещё и соответствующий выходной сигнал $\mathbf{w_g=\lambda(a_m)}$.

Таблица 20

Общий вид отмеченной таблицы переходов а. 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 21

Отмеченная таблица переходов а. Мура $\mathbf{S_3}$

	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 $\mathbf{a_m}$ и $\mathbf{a_s}$ (исходное состояние и состояние перехода) соединяются дугой, направленной от $\mathbf{a_m}$ к $\mathbf{a_s}$, если в автомате имеется переход из $\mathbf{a_m}$ в $\mathbf{a_s}$, то есть если $\mathbf{a_s = \delta(a_m, z_f)}$ при некотором $\mathbf{z_f \in X}$.

Дуге $\mathbf{(a_m, a_s)}$ графа автомата приписывается входной сигнал $\mathbf{z_f}$ и выходной сигнал $\mathbf{w_g = \lambda(a_m, z_f)}$, если он определён, и ставится прочерк в противном случае. Если переход автомата из состояния $\mathbf{a_m}$ в состояние $\mathbf{a_s}$ происходит под действием нескольких входных сигналов, то дуге $\mathbf{(a_m, a_s)}$ 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.22, рис.23 и рис.24 приведены графы цифровых автоматов S_1, S_2 и S_3, описанные ранее в табл. 18, табл. 19 и табл. 21.

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.

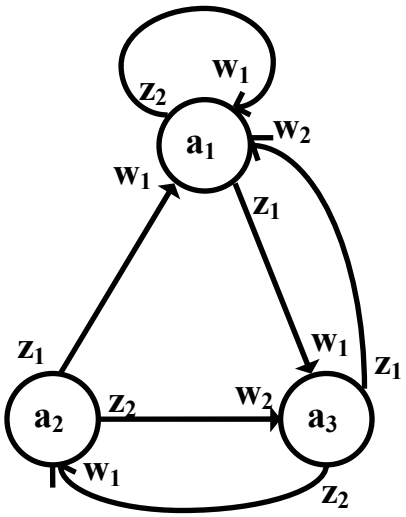


Рис.22. Граф автомата Мили $\mathbf{S_1}$

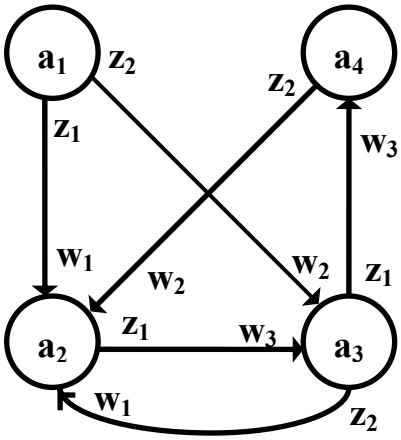


Рис. 23. Граф автомата Мили $\mathbf{S_2}$

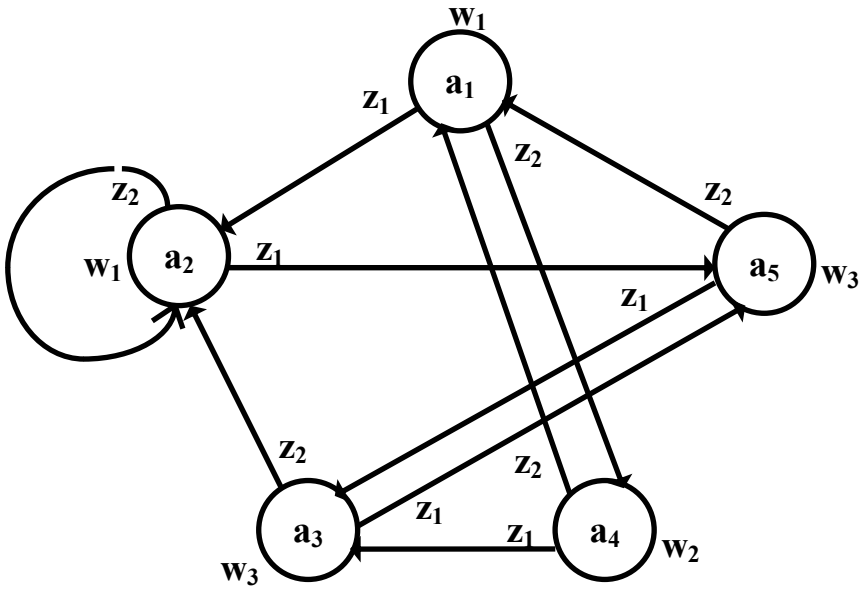


Рис.24. Граф автомата Мура $\mathbf{S_3}$

2.3. 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.

Таблица 22

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

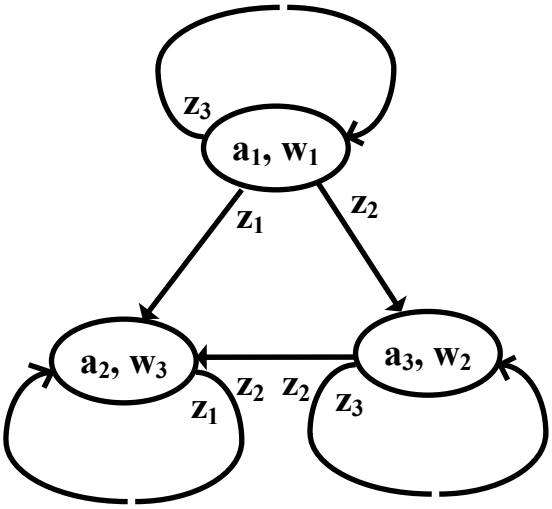


Рис. 25. Граф асинхронного автомата Мура S_4

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 табл. 22, а его граф – на рис.25. 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 ≠ 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.

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

2.4. 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 строками, 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 - **k** символов

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

$\omega =$ w_1 w_2 w_1 w_1 w_1 w_2 - **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 $z_{i1}, z_{i2}, \ldots, z_{ik}$. 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:

Выходной сигнал $w_{i1}=\lambda(a_m)$ в момент времени i_1 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 i_1.

По этой причине реакцией автомата Мура, установленного в состояние a_m, на

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	$w_{i1}=\lambda(a_m)$	$w_{i2}=\lambda(a_{i2})$	$w_{i3}=\lambda(a_{i3})$		$w_{i4}=\lambda(a_{i4})$.

входное слово $\xi = z_{i1}, z_{i2}, \ldots, z_{ik}$ является выходное слово той же длины **k**, с исключением первого символа выходного слова:

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

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

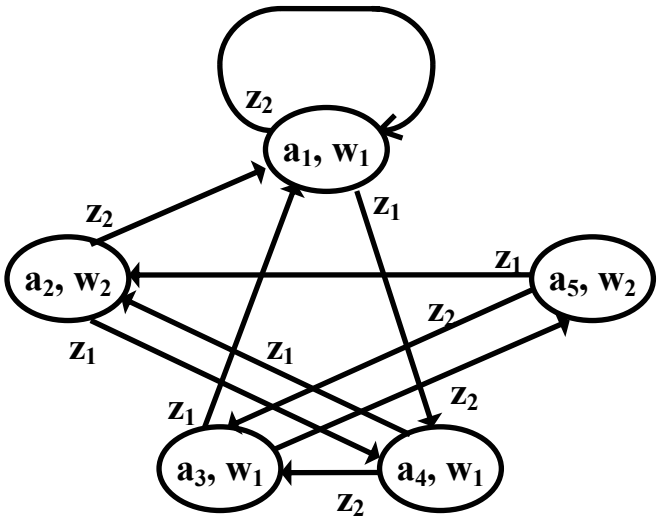


Рис.26. Граф автомата Мура $\mathbf{S_5}$

$\xi =$ 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.

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

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.

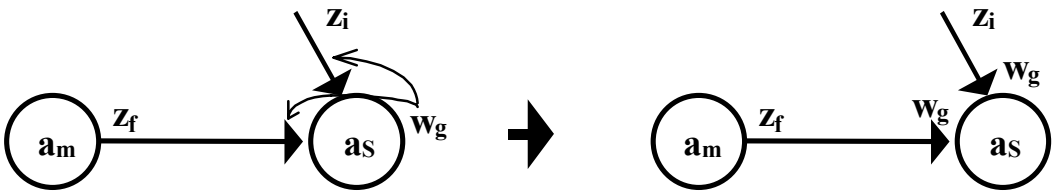


Рис.27. Переход от автомата Мура к автомату Мили

Пример трансформации автомата Мура S_3 (табл. 21) в автомат Мили S_6 приведён на рис.28 и в табл. 23 – 25.

(Второй экземпляр) Таблица 21

Отмеченная таблица переходов автомата Мура $\mathbf{S_3}$

	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 23

Выделение совмещённой таблицы переходов и выходов автомата Мили $\mathbf{S_6}$

	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 24

Таблица переходов эквивалентного автомата Мили $\mathbf{S_6}$

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

Таблица25

Таблица выходов эквивалентного автомата Мили $\mathbf{S_6}$

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

По процедуре построения автомата Мили $\mathbf{S_B}$ видно, что он эквивалентен автомату Мура $\mathbf{S_A}$.

Действительно, если некоторый входной сигнал $z_f \in X$ поступает на вход автомата $\mathbf{S_A}$, 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 $\mathbf{S_B}$ 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 автома-

тов 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.23).

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

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

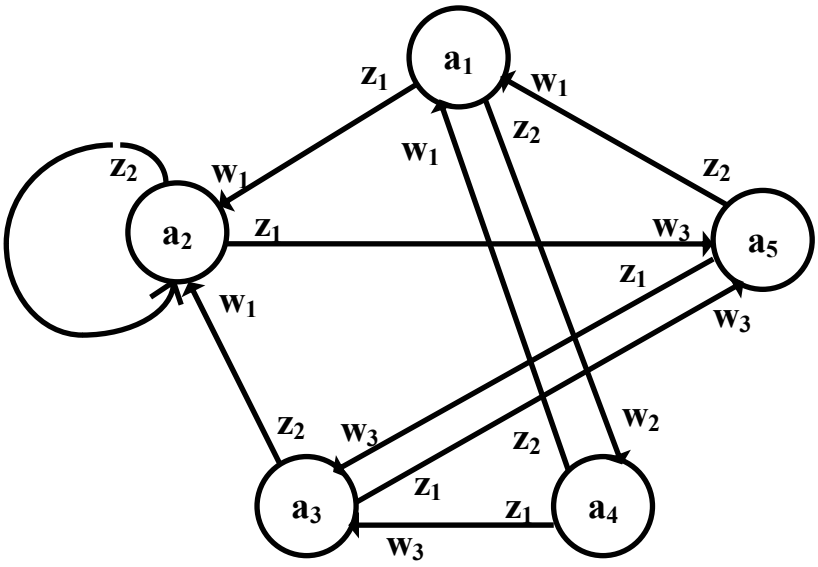


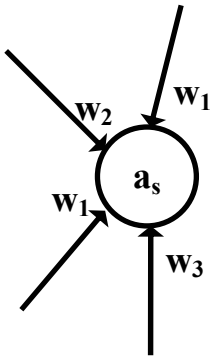
Рис.28. Граф автомата Мили S_6,
эквивалентного автомату Мура S_3

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.29.

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

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

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.30) будет переход из множества состояний A_m, порождаемых a_m, в состояние (a_s, w_k) под действием того же входного сигнала z_f.



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

Рис.29. Построение множества A_S

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

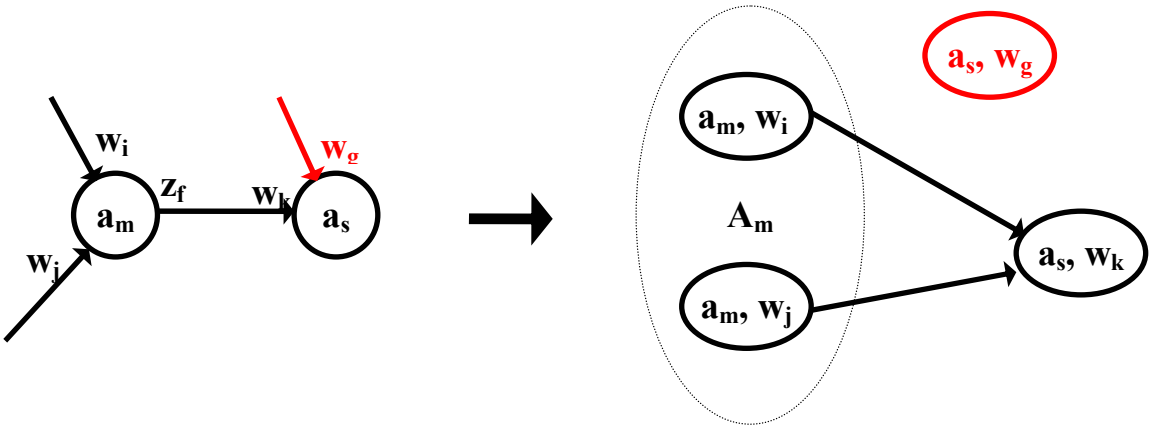
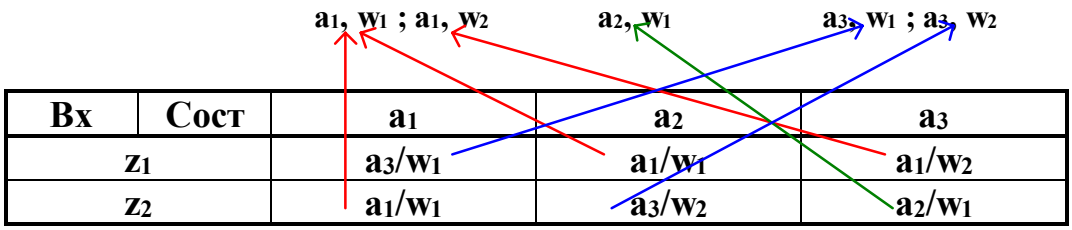


Рис.30. Иллюстрация трансформации автомата Мили в автомат Мура

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.22 и описанного в табл. 18.



Вх	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

$A_1 = \{(a_1,w_1),(a_1,w_2)\} = \{b_1,b_2\}$, 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 $A_B = A_1 \cup A_2 \cup A_3 = \{b_1, b_2, b_3, b_4, b_5\}$. С каждым состоянием, представляющим теперь пару, свяжем выходной сигнал, являющийся вторым элементом этой пары:

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

Таблица 26

Отмеченная таблица переходов эквивалентного автомата Мура 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

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. 26.

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.31.

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

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.

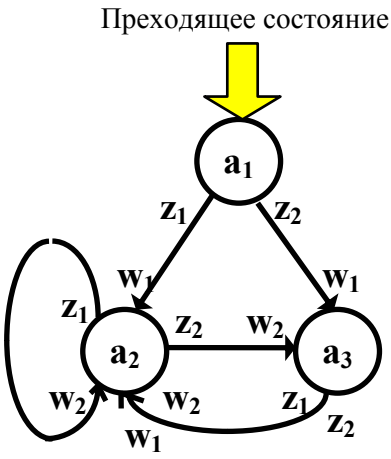


Рис. 31. Автомат Мили S_7
с преходящим состоянием a_1

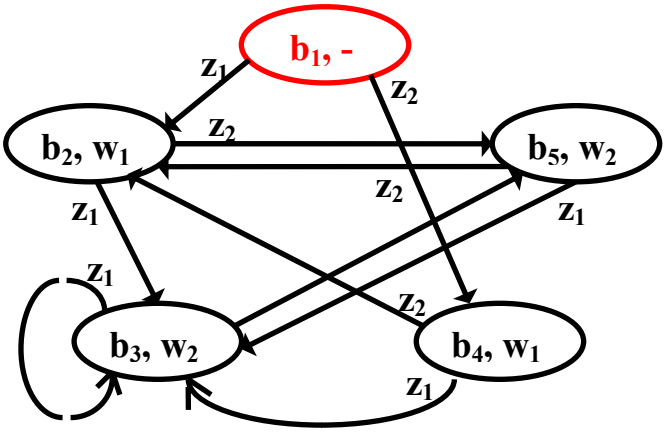


Рис.32. Автомат Мура S_8 , эквивалентный автомату Мили S_7

Таблица 27

Совмещённая таблица а. 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 28

Отмеченная таблица эквивалентного автомата Мура 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

2.5. 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 слов в выходные слова.

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

Таблица 29

Совмещённая 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 30

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

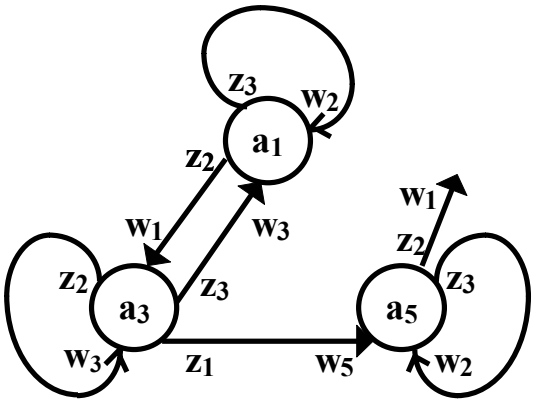


Рис.33. Граф автомата Мили после двух шагов минимизации

Полностью определённый автомат является частным случаем частичного автомата.

2.5.1. 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.

Шаг 3. Нахождение совместимых состояний автомата.

Состояния $\mathbf{a_i}$ и $\mathbf{a_j}$ 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 $\mathbf{C_j(\ i\ )}$, номера классов (индексы), в которые автомат переходит 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 $\mathbf{C_j(\ i\ )}$ образуют классы **(i+1)** – совместимости.

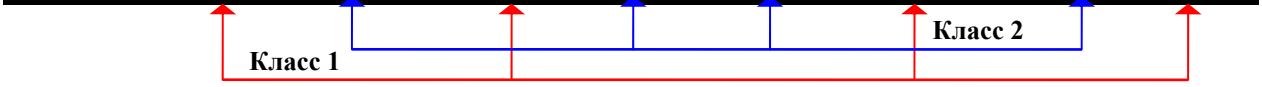
При выполнении операции расщепления классов специальный символ неопределённости может быть заменён номером (индексом) любого класса. Если операцию расщепления **i** – классов применить последовательно, начиная с 1 – класса, то через конечное число шагов процесс расщепления закончится. Нерасщепляемые 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 $\mathbf{S_{10}}$, описываемого совмещённой табл. 31 переходов и выходов.

Таблица 31

Совмещённая таблица переходов и выходов автомата Мили $\mathbf{S_{10}}$

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



Процедуру расщепления классов для нахождения классов конечной совместимости удобно проводить с использованием таблиц (рис.34).

Задачей минимизации методом расщепления классов совместимости является получение как можно меньшего количества, как можно большей ёмкости классов конечной совместимости. Поэтому состояние 8 ($\mathbf{a_8}$) первоначально отнесённое к двум классам двоичной совместимости из–за неопределённой первой отметки окон-

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. 34. Нахождение классов конечной совместимости

чательно 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 $\delta = (\mathbf{C_i, z_j})$ 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.

Таблица 32

Совмещённая таблица переходов и выходов
нормализованного минимизированного автомата Мили $\mathbf{S_{10}}$

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

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 классы 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 последний 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 $\mathbf{S_{10}}$ представлена в табл. 32, а его граф – на рис. 35.

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

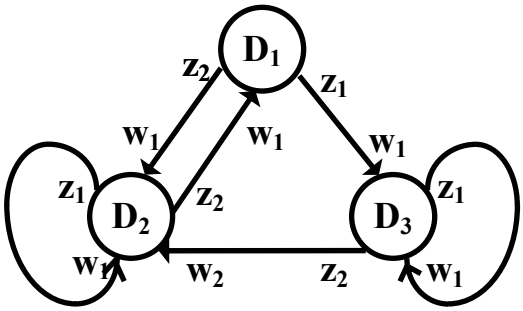


Рис.35. Граф нормализованного минимизированного автомата Мили $\mathbf{S_{10}}$

Таблица 33

Совмещённая таблица переходов и выходов автомата Мили $\mathbf{S_{11}}$

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

Определение классов конечной совместимости производится с использованием табличного описания цифрового автомата.

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**

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

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 34

Совмещённая таблица переходов и выходов минимизированного нормализованного 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

Таблица 35

Совмещённая 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 36

Совмещённая таблица переходов и выходов минимального нормализованного 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

Сопоставлением исходной табл. 35 автомата Мили S_{11} и результирующей табл. 36 эквивалентного минимального нормализованного автомата Мили S_{13} определяется эффект минимизации количества состояний блока памяти.

Граф минимизированного нормализованного автомата Мили S_{12} представлен на рис.36. После исключения недостижимых состояний получен минимальный нормализованный автомат Мили S_{13}, представленный совмещённой табл. 36 переходов и выходов.

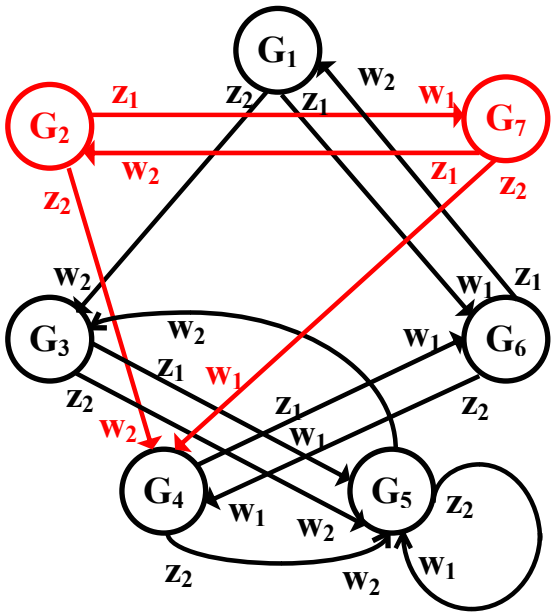


Рис.36. Граф минимизированного нормализованного автомата Мили S_{12}

2.5.2 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 $\mathbf{a_i}$, отличающееся от начального, то такое состояние исключается вместе с соответствующими переходами таблицы переходов.

Пример выполнения двух начальных этапов минимизации автомата Мура $\mathbf{S_{14}}$ приведён в табл. 37 и табл. 38.

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.

Таблица 37

Отмеченная таблица автомата Мура $\mathbf{S_{14}}$

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 38

Отмеченная таблица минимизированного автомата Мура $\mathbf{S_{15}}$

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

После нахождения совместимых состояний автомата строится минимизированный нормализованный автомат Мура.

При табличном описании нормализованного автомата Мура, имеющего $\mathbf{N_n}$ классов совместимых состояний, переход $\mathbf{\delta(N_i, z_j)}$ считается неопределённым, если для всех $\mathbf{a_i \in N_i}$ переходы $\mathbf{\delta(a_i, z_j)}$ неопределённы. Если хотя бы для одного $\mathbf{a_i \in N_i}$ переход $\mathbf{\delta(a_i, z_j)}$ определён, то в таблице нормализованного автомата указывается именно этот класс $\mathbf{N_k}$, в который происходит переход. У частичного автомата таких переходов возможно несколько. Каждое класс $\mathbf{N_i}$ 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 абстрактных автоматов Мура на примере полностью определённого автомата $\mathbf{S_{16}}$ (табл. 39).

Автомат имеет четыре выходных сигнала, поэтому его состояния распределяются на четыре класса нулевой совместимости. Абстрактный автомат Мура $\mathbf{S_{16}}$ – полностью определённый, как по выходам, так и по переходам, поэтому сразу находятся конечные классы совместимых состояний.

2.6. 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.

Функционально полной системой логических элементов являются, например, (**И, НЕ**), (**ИЛИ, НЕ**), (**И–НЕ**), (**ИЛИ–НЕ**).

2.6.1. 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.

Таблица 39

Отмеченная таблица переходов автомата Мура $\mathbf{S_{16}}$

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**

Таблица 40

Отмеченная таблица переходов
нормализованного автомата Мура 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

Для синтеза цифровых автоматов триггеры рассматриваются как элементы систем и важным является изучение его поведения в системе, а не внутренняя 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.

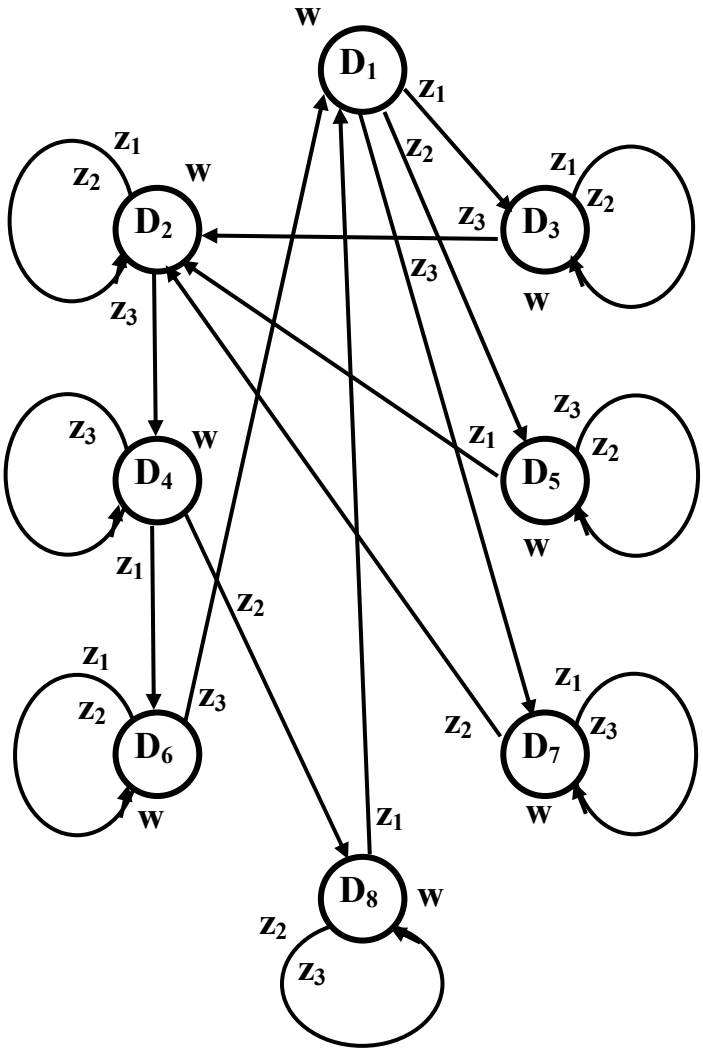
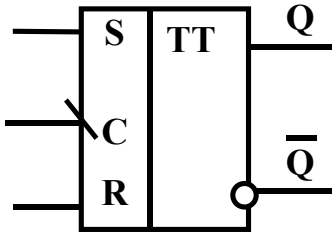
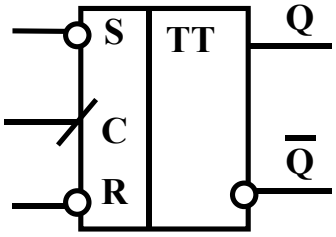


Рис. 37. Граф нормализованного автомата Мура $\mathbf{S_{17}}$

Триггер типа RS. Название триггера происходит от аббревиатур двух английских слов **S**et (установить) и **R**eset (сбросить), которые, кроме этого ещё и соседние в латинском алфавите. Этот триггер имеет два входа – **R** и **S** и два выхода – **Q** и $\overline{\mathbf{Q}}$. Обозначения различных вариантов исполнения этого триггера приведены на рис.38.



Синхронизированный спадом синхроимпульса **RS** – триггер с прямыми входами



Синхронизированный фронтом синхроимпульса **RS** – триггер с инверсными входами

Рис. 38. 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.39а и рис.39б.

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. 39. Варианты табличного описания работы **RS** – триггеров

Факт принадлежности сигналов различным моментам автоматного времени отражается индексами **t** и **t+1** или **Q** и $\mathbf{Q}^+$. 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

$$\mathbf{Q^+ = S \vee \overline{R}Q\ ;} \qquad (37)$$
$$\mathbf{R\&S \equiv 0.}$$

На рис.39в 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.40.

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	Q

б)

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

♣ – любое значение

в)

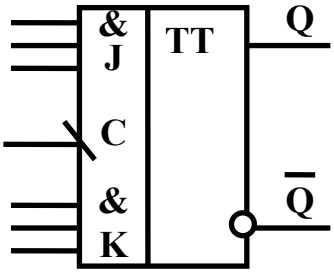
Рис. 40. Варианты табличного описания работы **JK** – триггеров

Характеристическое уравнение JK триггера, заданное таблицей переходов на рис.40а, после минимизации с помощью диаграммы Вейча, имеет вид:

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

Условное обозначение **JK** триггера с встроенными конъюнкторами на входах (микросхема 155ТВ1) приведено на рис.41.

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



Синхронизированный спадом синхроимпульса **JK** – триггер “с встроенной логикой”

Рис. 41 Обозначение **JK** – триггера

Триггер типа D. Название происходит от английского термина “**D**elay” (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.

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)

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

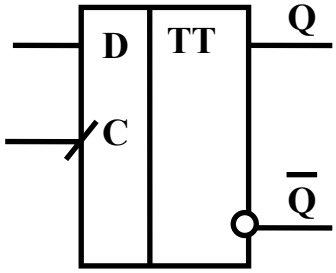
в)

Рис. 42. Варианты табличного описания работы **D** – триггеров

Таблицы переходов и матрица переходов **D** триггера приведены на рис.42.

Условное обозначение двухступенчатого **D** триггера приведено на рис.43. Характеристическое уравнение **D** триггера, заданное таблицей переходов на рис.42а, имеет вид:

$$\mathbf{Q^+ = D}. \qquad (39)$$



Синхронизированный фронтом синхроимпульса **D** – триггер

Рис. 43. Обозначение **D**-триггера

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.44).

По рис.44 видно, что **Т**– триггеры на основе **RS** и **D**–триггеров являются несинхронизированными. Варианты табличного описания работы **Т**-триггера приведены на рис.45.

Характеристическое уравнение **Т**-триггера имеет вид:

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

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.

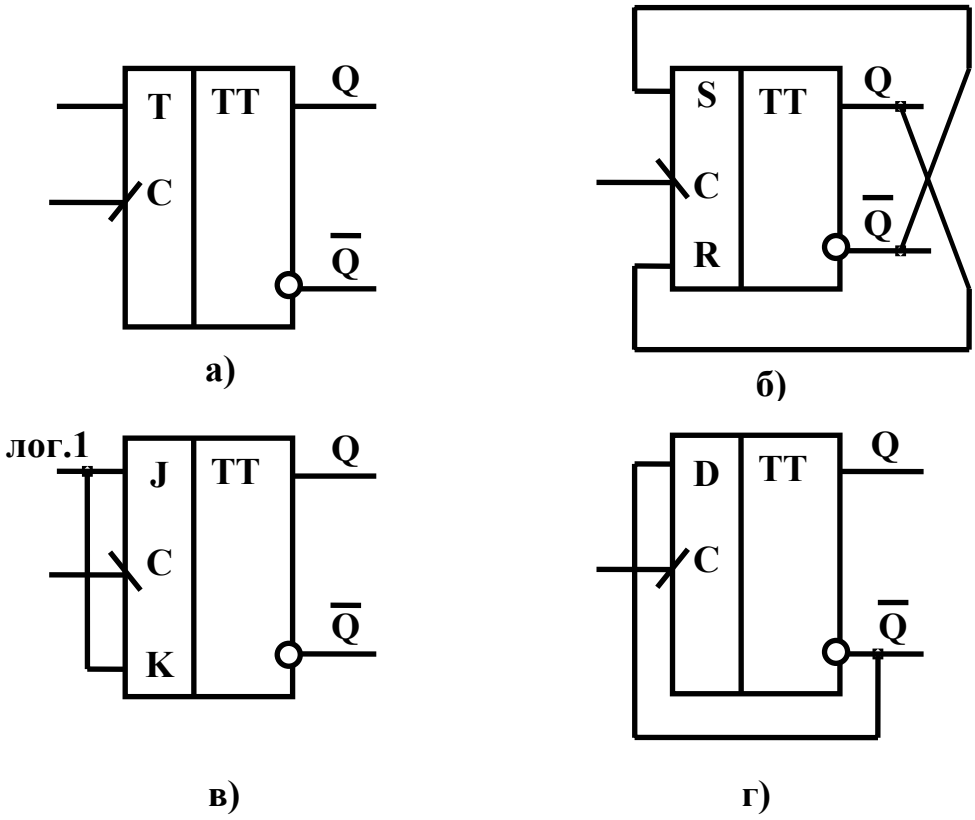


Рис.44. Варианты схем **Т**-триггеров

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.

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)

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

в)

Рис. 45. Варианты табличного описания работы **T**-триггеров

2.6.2. 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 (см. рис.2, рис.21) 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 $\mathbf{КС_{вх}}$ 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 $\mathbf{КС_{вых}}$ – 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 $\mathbf{КС_{вх}}$ 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 вре-

мени, 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.**

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.

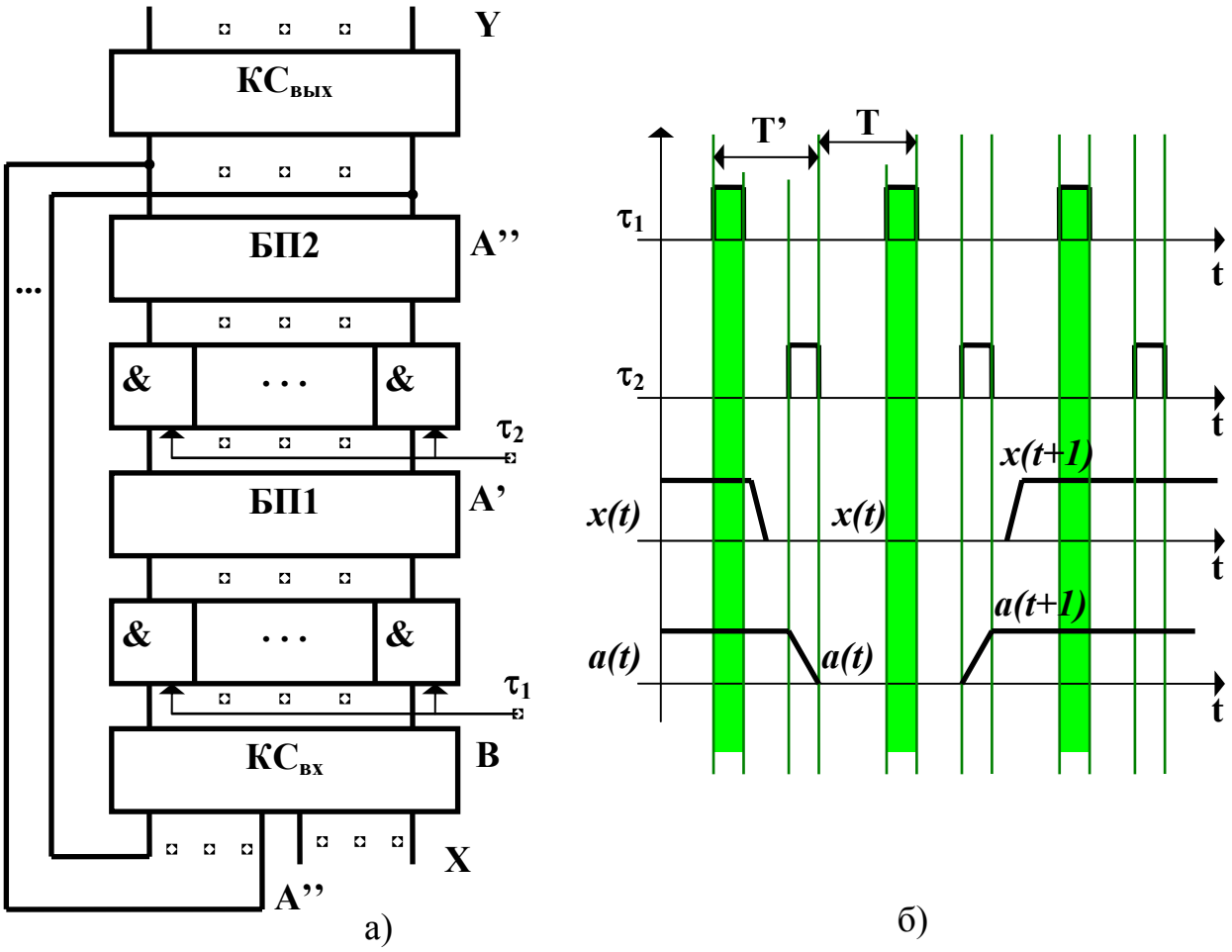


Рис.46. Корректная схема автомата Мура с двухтактной памятью

Таким образом, корректность работы цифрового автомата Мура обеспечивается введением двухтактного синхронизированного блока памяти. Логика работы двухтактной памяти практически реализуется в двухступенчатых триггерах структуры **O–B** (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 $\mathbf{КС_{вых}}$ 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.47.

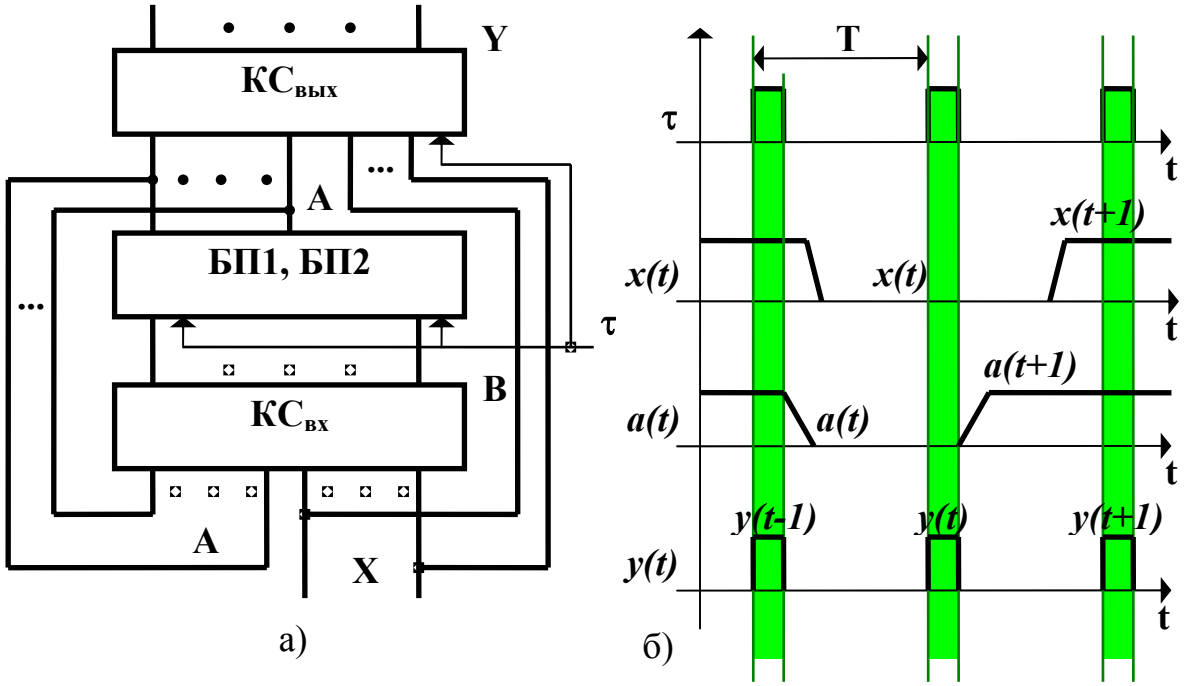


Рис.47. Корректная схема цифрового автомата Мили с двухтактной памятью и стробированием выходной комбинационной схемы

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

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

2.7. Структурный синтез цифровых автоматов по таблицам

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

$$\mathbf{n=]\log_2 M[\ ,}$$

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 $\mathbf{КС_{вых}}$. Кодированная таблица переходов только после переработки с использованием матрицы переходов для заданного типа триггеров будет называться кодированной таблицей возбуждений и соответствовать описанию комбинационной схемы $\mathbf{КС_{вх}}$.

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 $\mathbf{S_{18}}$, заданного табл. 41 переходов и табл. 42 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.

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

Таблица 41

Таблица переходов автомата Мили $\mathbf{S_{18}}$

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 42

Таблица выходов автомата Мили $\mathbf{S_{18}}$

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

кодирования часто встречающихся в таблице выходов сигналов следует использовать коды с максимально возможным 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.

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

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

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

Рис.48. Таблицы кодирования входных сигналов, выходных сигналов и состояний (по второму варианту)

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. Минимальный по материальным затратам вариант кодирования выбирается из конечных результатов при использовании всевозможных вариантов кодирования и минимизации.

Для второго принципа кодирования 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:

– в таблице выходов выходной сигнал $\mathbf{w_1}$ встречается 4 раза, выходной сигнал $\mathbf{w_3}$ – 3 раза, выходной сигнал $\mathbf{w_2}$ – 2 раза. В таблице переходов состояние $\mathbf{a_1}$ встречается 6 раз, состояние $\mathbf{a_3}$ – 2 раза, состояние $\mathbf{a_2}$ – 1 раз. В соответствии с количеством этих ссылок заполнены таблицы кодирования на рис.48.

В этих таблицах представлены как двоичные коды входных, выходных сигналов и состояний блока памяти цифрового автомата, так и их аналитические представления.

Запрещённые или неиспользуемые коды в таблицах на рис.48 получаются вследствие того, что количество всевозможных кодовых комбинаций превосходит количество кодируемых входных, выходных сигналов и количество состояний блока памяти цифрового автомата.

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♣

Рис.49. Кодированные таблицы переходов, выходов и возбуждений блока памяти на **JK** – триггерах

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.

По этой причине функции алгебры логики, описывающие как входную, так и выходную комбинационные схемы цифрового автомата, являются частично определёнными. Частично определённым является также и цифровой автомат, так как имеются запрещённые коды состояний его блока памяти.

Частичная определённость всех компонентов цифрового автомата является положительным фактором в том случае, если они подлежат минимизации при использовании в качестве элементной базы наборов логических элементов или программируемых логических матриц.

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

При использовании в качестве элементной базы дешифраторов, мультиплексоров или ПЗУ все ФАЛ доопределяютя как имеющие нулевые значения. Для заполнения таблицы на рис.50, выполнена конкатенация кодов входных сигналов и кодов состояний по порядку следования переменных $x_1x_0Q_1Q_0$ и заполнена таблица истинности для функций выхода и возбуждений. В таблице, на запрещённых наборах входных сигналов и состояний, значения функций не определены и обозначены символом ♣к.

Для используемых в качестве элементной базы наборов логических элементов или программируемых логических матриц, выполним минимизацию ФАЛ, описывающих входную и выходную комбинационные схемы цифрового автомата.

Для ФАЛ четырёх переменных задача минимизации проще всего решается при использовании карт Вейча (или карт Карно).

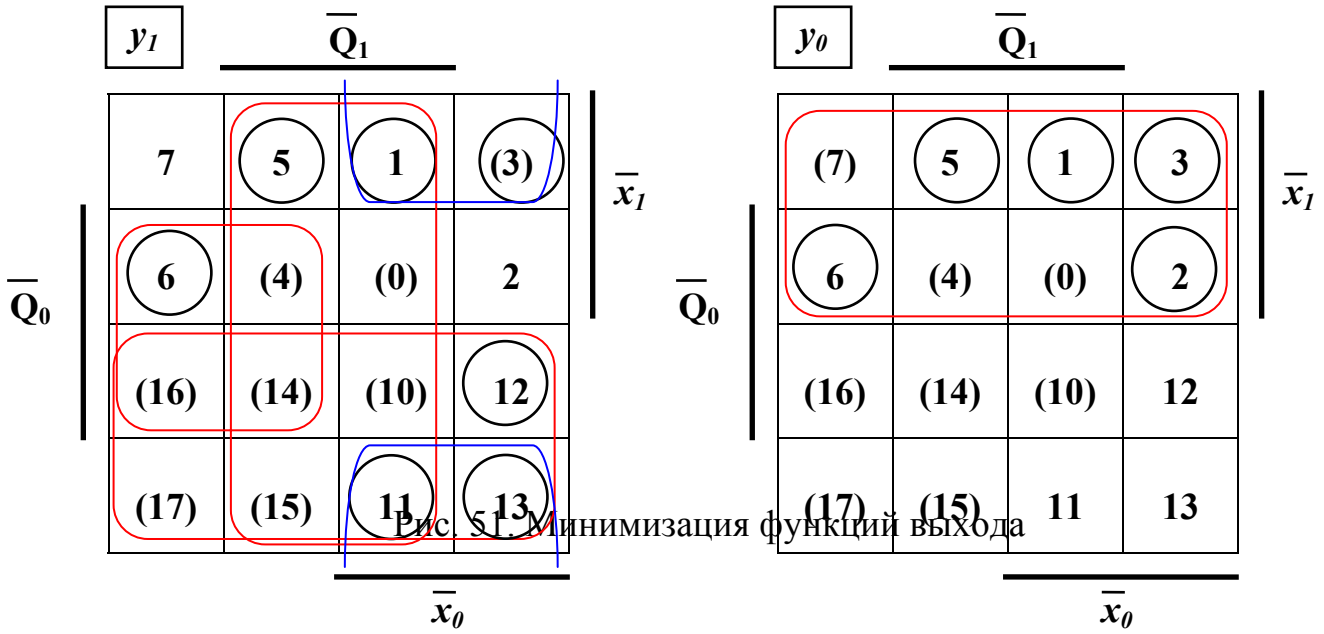


Рис. 51. Минимизация функций выхода

Рис. 51. Минимизация функций выходов

По диаграммам Вейча функций выходов и возбуждения получим:

– минимизированные функции выходов в ДНФ:

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

– минимизированные функции возбуждения в ДНФ:

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

Для первого принципа кодирования построим кодированные таблицы переходов, выходов и возбуждений для **JK**-триггера с использованием матрицы переходов на рис.40в.

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.

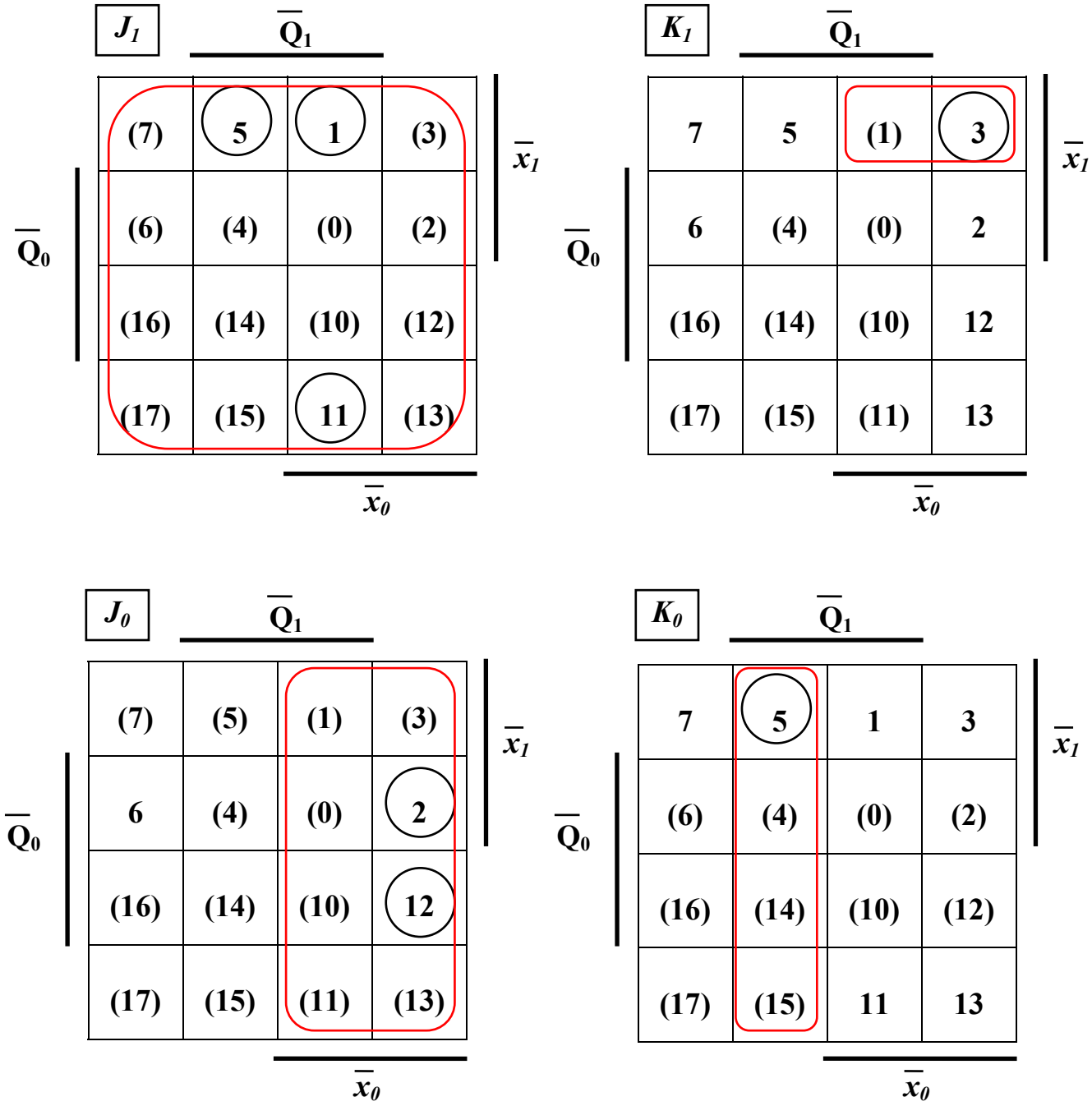


Рис.52. Минимизация функций возбуждения

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.

Таким образом формулируется общий принцип минимизации дискретных цифровых устройств, то есть, минимальный вариант построения принципиальной схемы может быть получен только после перебора и сравнения всех возможных вариантов построения цифрового устройства.

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

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

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

Рис.53. Таблицы кодирования входных сигналов, выходных сигналов и состояний (по первому варианту)

Запрещённые коды $x_1x_0Q_1Q_0$:

1100; 1101; 1110; 1111; 0011; 0111; 1011,

где **11**♣♣ – запрещённый код входных сигналов,

♣♣**11** – запрещённый код состояний

Очевидно, что на практике реализовать полную оптимизацию дискретного цифрового устройства невозможно, вследствие большой трудоёмкости выполнения процедуры минимизации.

Для практического использования методов минимизации цифровых автоматов исключительное значение имеет инженерная интуиция при выборе вариантов кодирования и минимизации, которая подкрепляется практическим опытом по проектированию цифровых автоматов различного назначения. По этой причине теоретическую и практическую ценность имеют программы для ЭВМ, решающие задачу автоматизированного проектирования и минимизации таких цифровых автоматов.

Функции выхода цифрового автомата требуется задать сравнительно редко, поскольку обычно применяются цифровые автоматы (счётчики, регистры и т.п.) не имеющие выходной комбинационной схемы.

В случаеналичия выходной комбинационной схемы она решает простейшие задачи, например, задачу формирования предупреждающего сигнала о переполнении счётчика или о его нулевом содержимом, тогда выходная комбинационная схемапросто не нуждается в минимизации.

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.54. Кодированные таблицы переходов, выходов и возбуждений

Для сложных цифровых автоматов выходная комбинационная схема обычно представляет собой преобразователь кода (шифратор) состояния блока памяти цифрового автомата в выходной код цифрового автомата.

Примером довольно сложной выходной комбинационной схемы цифрового автомата является преобразователь кода состояния счётчика в код для управления семисегментными цифровыми индикаторами.

Для многих стандартных применений выходные комбинационные схемы цифровых автоматов минимизированы, разработыны и выпускаются в виде интегральных схем.

Таким образом, задача минимизации выходной комбинационной схемы цифрового автомата часто сводится к задаче выбора интегральных микросхем для конкретного применения.

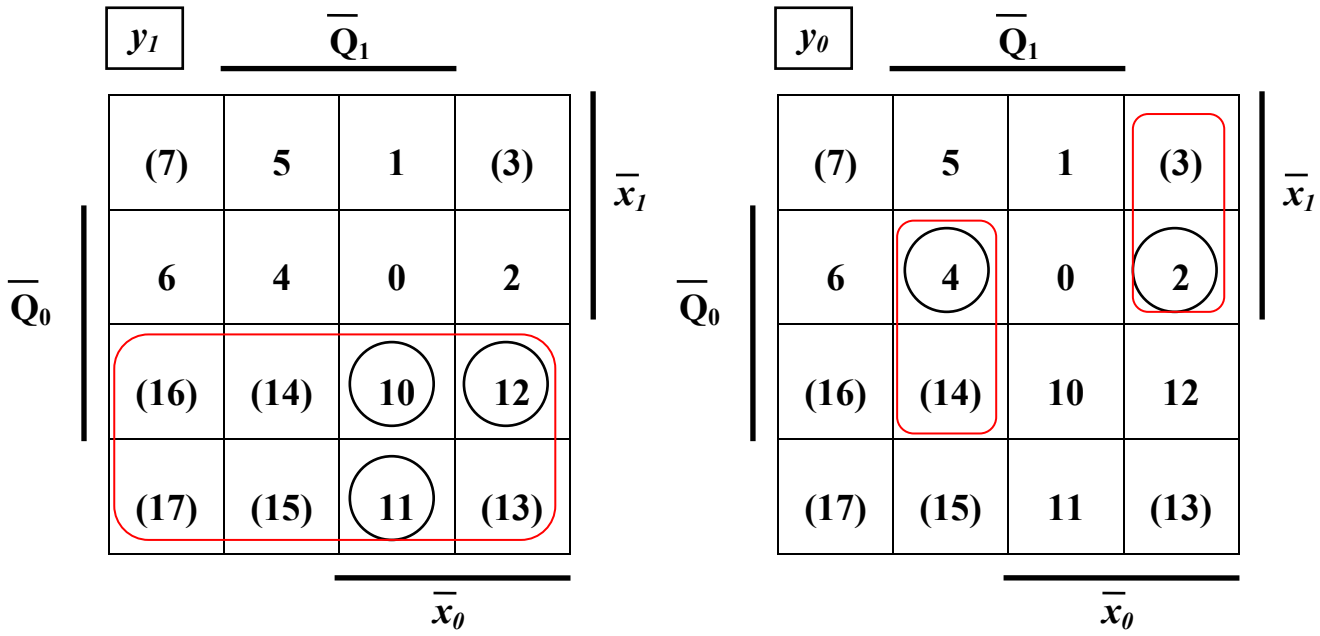


Рис. 55. Минимизация функций выхода

На рис.57 приведена принципиальная схема цифрового автомата $\mathbf{S_{18}}$ 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 $\mathbf{K_0}$ “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”

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.

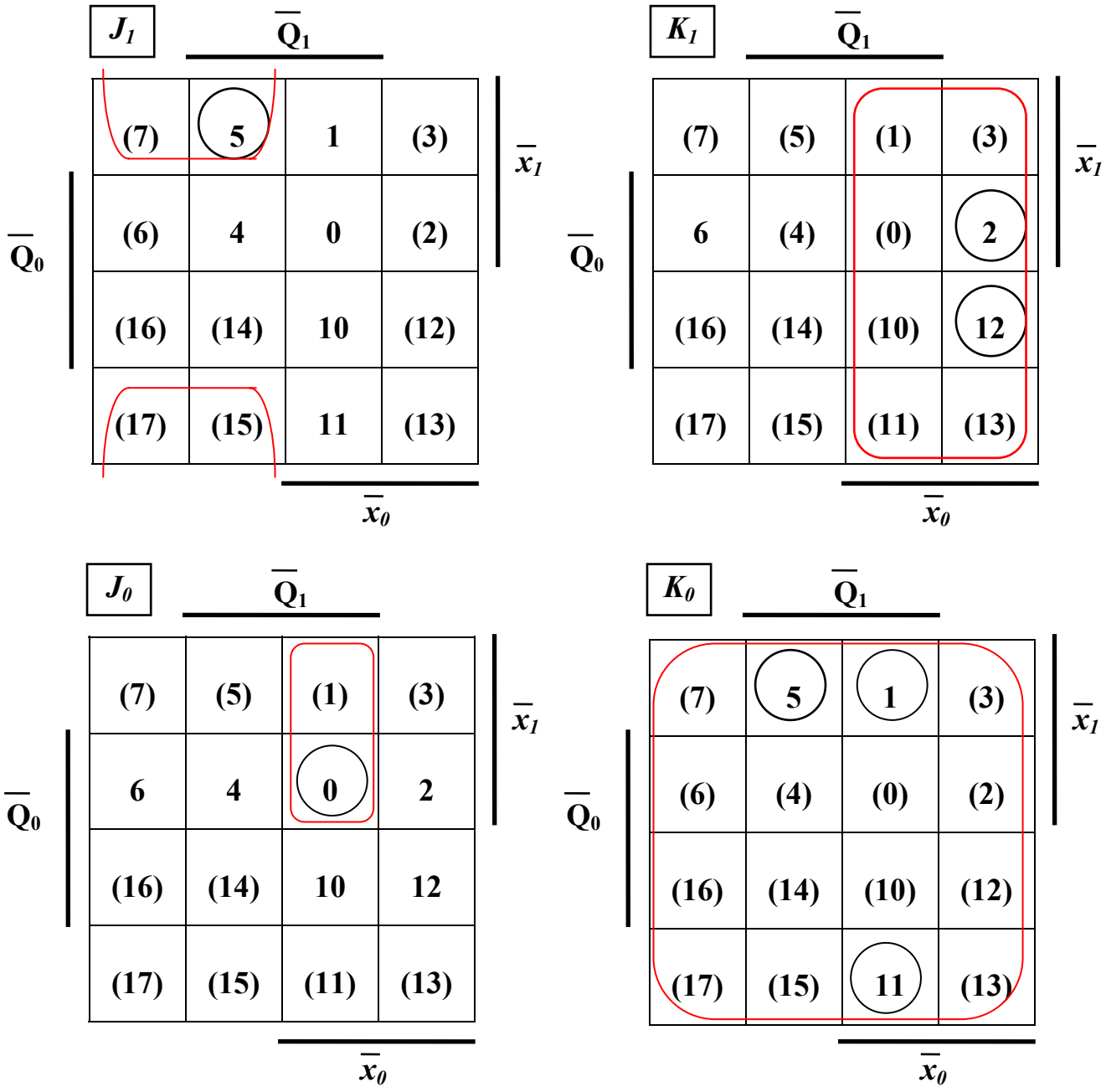


Рис.56. Минимизаций функций возбуждения **JK**– триггеров памяти

Минимизированные функции выходов в ДНФ:

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

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

Минимизированные функции возбуждений в ДНФ:

$$\left.\begin{array}{l} J_1 = 1;\ \ K_1 = \bar{x}_1 \bar{x}_0 Q_0; \\ J_0 = \bar{x}_0;\ \ K_0 = x_0 \bar{Q}_1; \end{array}\right\} \text{А}$$

$$\left.\begin{array}{l} J_1 = x_0 Q_0;\ \ K_1 = \bar{x}_0; \\ J_0 = \bar{x}_1 \bar{x}_0 \bar{Q}_1;\ K_0 = 1; \end{array}\right\} \text{Б}$$

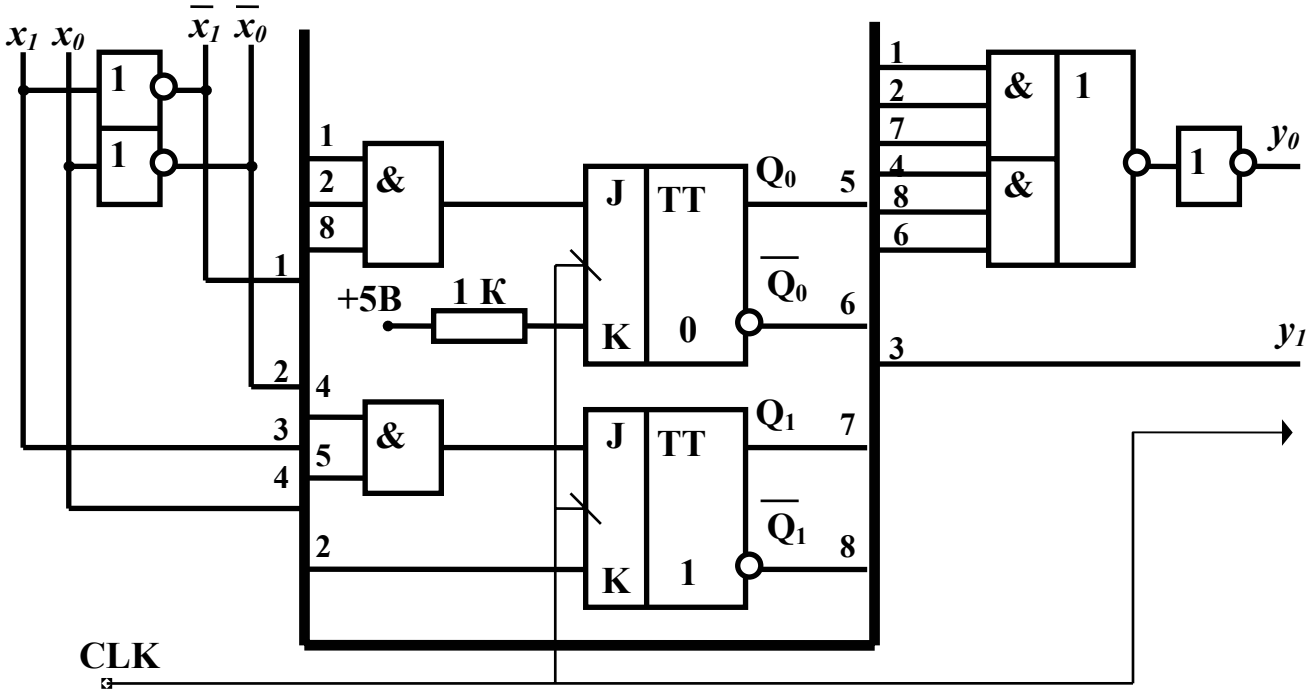


Рис.57. Принципиальная схема автомата Мили $\mathbf{S_{18}}$

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 напряжения. Для надёжной работы цифрового автомата такой схемы недостаточно.

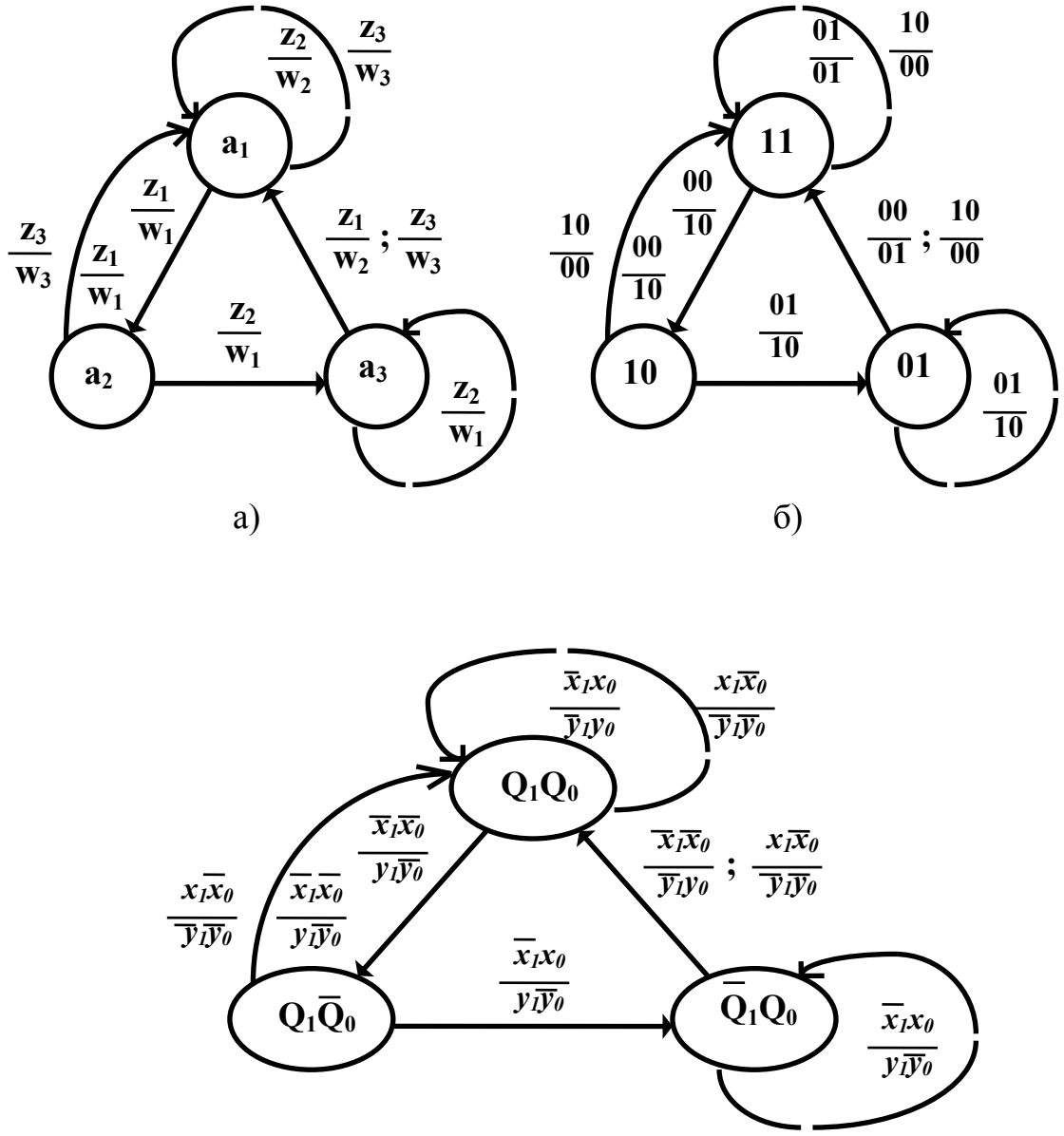


Рис. 58. Различные виды разметки графа автомата Мили $\mathbf{S_{18}}$

а). граф абстрактного автомата Мили $\mathbf{S_{18}}$;

б). граф структурного кодированного автомата Мили $\mathbf{S_{18}}$;

в). граф с аналитической разметкой для структурного кодированного автомата Мили $\mathbf{S_{18}}$

2.8. 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.58 представлен пример различных видов разметки графа цифрового автомата Мили $\mathbf{S_{18}}$.

Функцию выходов по графу рис.58в 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 разметке 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.59.

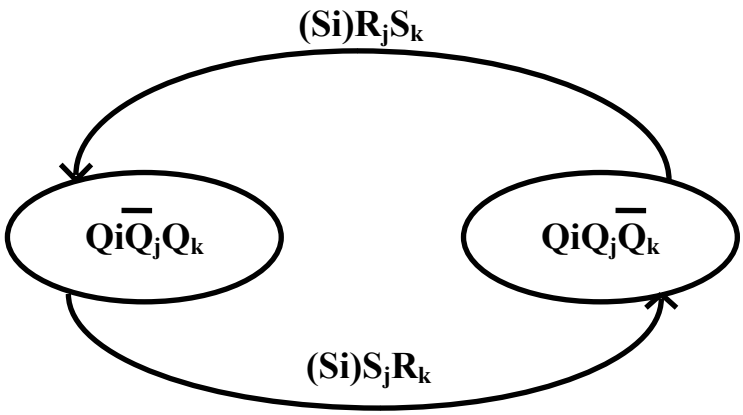


Рис.59. Пример разметки дуг графа для **RS**–триггеров в блоке памяти

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.60 приведён пример разметки дуг фрагмента графа цифрового автомата для **T**–триггеров в блоке памяти.

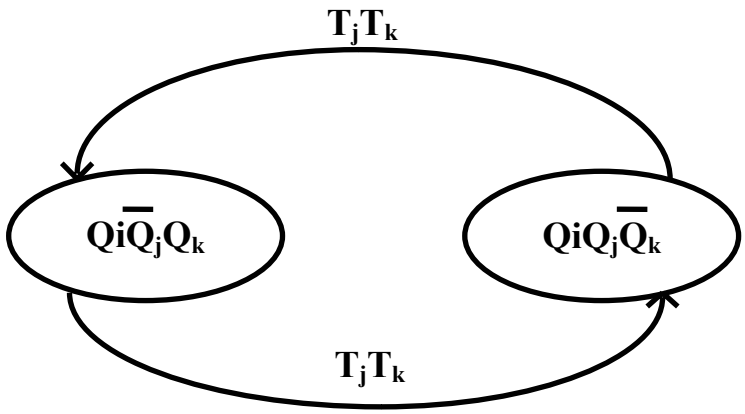


Рис.60. Пример разметки дуг графа для **T**–триггеров в блоке памяти

Триггеры типа D. На рис.61 приведён пример разметки дуг фрагмента графа цифрового автомата для **D**–триггеров в блоке памяти.

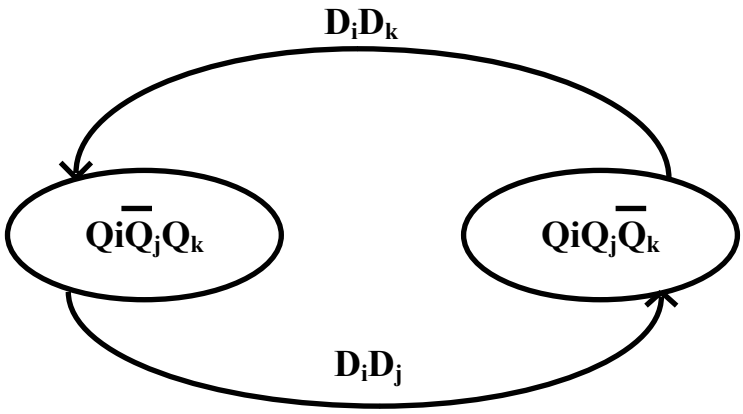


Рис.61. Пример разметки дуг графа для **D**–триггеров в блоке памяти

Триггеры типа JK. На рис.62 приведён пример разметки дуг фрагмента графа цифрового автомата 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.

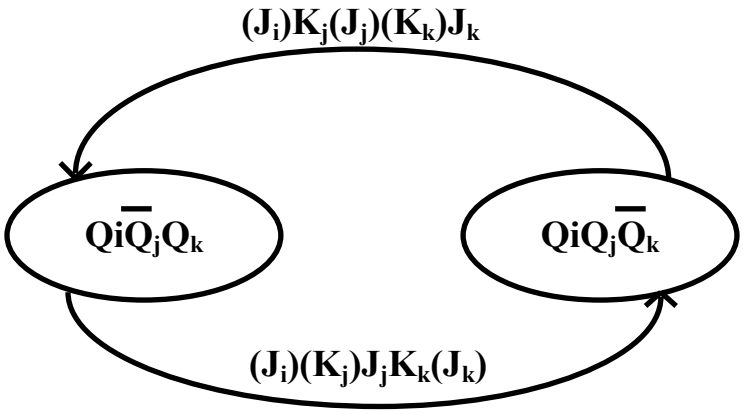


Рис.62. Пример разметки дуг графа для **JK**–триггеров в блоке памяти

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

Глава 3
МИКРОПРОГРАММНЫЕ АВТОМАТЫ

3.1. Декомпозиция устройств обработки цифровой информации

В любом устройстве или системе обработки цифровой информации можно выделить два существенно различающихся блока (рис.63):

– операционный блок (или операционный автомат);

– управляющий блок (или управляющий автомат).

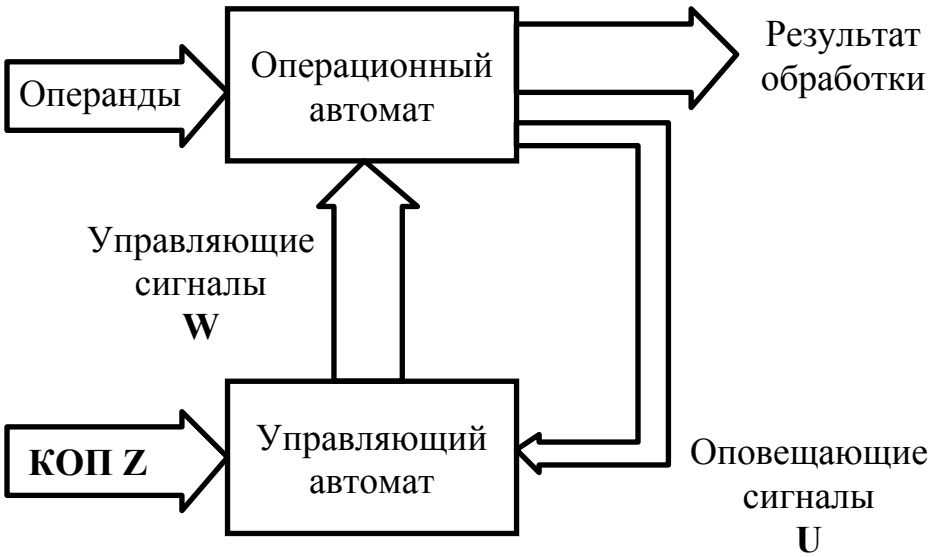
Операционный автомат состоит из регистров, сумматоров и других узлов, производящих приём из внешней среды операндов и их хранение, преобразование (обработку) и выдачу во внешнюю среду результатов преобразования (обработки), а также выдачу в управляющий автомат и внешнюю среду оповещающих сигналов, принадлежащих множеству

$$U=\{u_1, ...,u_i, ..., u_l\},$$

о знаках и особых значениях операндов, их отдельных разрядов, особых значениях промежуточных и конечных результатов и конечных результатов операции (например, равенство нулю результата операции). Процесс функционирования во времени устройства обработки цифровой информации состоит из последовательности тактовых интервалов, в течение которых операционный автомат (блок) производит заданные элементарные преобразования операндов.

Выполнение этих элементарных преобразований (операций) инициируется поступающими в операционный автомат соответствующими управляющими сигналами из множества

$$W=\{w_1, ..., w_g, ..., w_G\}.$$



$$Z=\{z_1, ..., z_f, ..., z_F\};\ W=\{w_1, ..., w_g, ..., w_G\};$$
$$U=\{u_1, ..., u_i, ..., u_I\}.$$

Рис.63. Декомпозиция устройства обработки цифровой информации

Элементарная функциональная операция (или их некоторая комбинация), выполняемая за один тактовый интервал и приводимая в действие одним управляющим сигналом w_g, называется микрооперацией. В некоторые такты могут поступать несколько управляющих сигналов, вызывая параллельное во времени выполнение соответствующих им микроопераций. Такая совокупность нескольких микроопераций называется микрокомандой. В частности микрокоманда может состоять из одной микрооперации.

Управляющий автомат (блок) вырабатывает распределённую во времени последовательность управляющих сигналов $w_{t1}, ..., w_{tj}, ..., w_{tJ}$ $(w_{tj} \in W)$, порождающих в операционном автомате нужную последовательность микроопераций.

Последовательность управляющих сигналов определяется сигналами, соответствующими поступившему на вход из внешней среды коду операции (**КОП**), принадлежащему множеству

$$Z=\{z_1, ..., z_f, ..., z_F\},$$

и сигналами

$$U=\{u_1, ...,u_i, ..., u_I\},$$

зависящими от операндов и промежуточных результатов преобразований.

Операционный автомат задаётся его структурой, то есть составом узлов и связями между ними, и выполняемым операционным блоком набором микроопераций.

Последовательность микрокоманд, обеспечивающая выполнение данной операции (например, операции нормализации числа в форме с плавающей точкой (запятой)), называется микропрограммой данной операции.

Таким образом, функционирование устройства или системы обработки цифровой информации может быть описано совокупностью реализуемых в нём микропрограмм.

Все самые сложные операции, выполняемые цифровыми автоматами (в том числе и ЭЦВМ) могут быть реализованы последовательным выполнением некоторых элементарных операций, например, функциональной полнотой обладает набор из четырёх микроопераций:

– пересылка информации из любой ячейки памяти в любую другую ячейку;
– сложение кода с +1 (инкремент) или с –1 (декремент);
– условный переход по совпадению кодов;
– безусловный останов.

По этой причине операционные автоматы (блоки) имеют фактически унифицированную структуру для различных цифровых устройств и систем обработки цифровой информации и фиксированный набор выполняемых элементарных операций (например, арифметико–логические устройства (АЛУ) микропроцессоров **580ВМ80**, **80×86**, **1804ВС1** и т. д.). Таким образом, операционные автоматы при проектировании цифровых устройств и систем выбираются из серийно выпускаемых промышленностью.

Управляющие автоматы (блоки) отличаются большим разнообразием для применяемых на практике устройств и систем обработки цифровой информации. Обычно требуется спроектировать такой автомат оптимальным для решаемой практической задачи и поэтому общих принципов их проектирования нет. Согласно рис.63, управляющий автомат соответствует общей структуре цифрового автомата, как кодопреобразователя с внутренней памятью (см. рис.1).

3.2. Управляющие автоматы

Любая команда, операция или процедура, выполняемая в операционном блоке, описывается некоторой микропрограммой и реализуется за несколько тактов, в каждом из которых выполняется шаг микропрограммы в одну или несколько микроопераций.

Интервал времени, отводимый на выполнение микрооперации, называется рабочим тактом или просто тактом устройства или системы обработки цифровой информации.

Для реализации команды, операции или процедуры (микропрограммы) необходимо на соответствующие управляющие входы операционного блока подать определённым образом распределённую во времени последовательность управляющих сигналов.

Часть устройства или системы обработки цифровой информации, предназначенная для выработки последовательностей управляющих сигналов, называется управляющим блоком (или управляющим автоматом).

Генерируемая управляющим автоматом последовательность управляющих сигналов задаётся поступающими на входы этого автомата кодом операции (**КОП**) **Z**, сигналами из операционного блока **U**, несущими информацию об особенностях операндов, промежуточных и конечного результата, а также синхросигналами, задающими границы тактов.

Таким образом, управляющий автомат (блок) формально можно рассматривать как цифровой автомат, определяемый:

– множеством двоичных выходных сигналов $\mathbf{W=\{w_1, ..., w_g, ..., w_G\}}$, соответствующих множеству микроопераций операционного блока. При $\mathbf{w_g=1}$ инициализируется **g**–я микрооперация;

– множествами входных сигналов

$$\mathbf{Z=\{z_1, ..., z_f, ..., z_F\}},\ \mathbf{U=\{u_1, ...,u_i, ..., u_I\}},$$

соответствующих задаваемому блоку извне двоичному коду операции (**КОП**) **Z** и двоичным оповещающим сигналам **U**;

– множеством подлежащих реализации микропрограмм, устанавливающих в зависимости от значений входных сигналов управляющие сигналы, выдаваемые блоком в определённые такты.

По множествам входных и выходных сигналов и всем микропрограммам определяется множество внутренних состояний блока

$$\mathbf{A=\{a_1, ..., a_m, ..., a_M\}},$$

мощность которого (объём памяти управляющего автомата) в процессе проектирования необходимо минимизировать.

Это замечание поясняет, почему управляющие блоки называются управляющими автоматами. Все свойства этих автоматов определяются микропрограммами. По этой причине управляющие автоматы иначе называются микропрограммными автоматами.

Управляющий микропрограммный автомат может быть задан как автомат Мура или как автомат Мили, в любом из которых функции переходов δ и функции выходов λ определяются заданной микропрограммой.

Управляющие автоматы строятся по двум основным принципам:

– как цифровые автоматы с жёсткой (или схемной логикой). Для каждой операции, задаваемой, например, кодом операции, строится набор комбинационных схем, которые в нужных тактах возбуждают соответствующие управляющие сигналы. Таким образом, строится цифровой автомат, в котором необходимое множество состояний определяется состоянием **K** запоминающих элементов

$$\mathbf{q=\{q_1, ..., q_k, ..., q_K\}},$$

где $\mathbf{K=]\log_2 M[}$ **при** $\mathbf{A=\{a_1, ..., a_m, ..., a_M\}}$.

Функции переходов δ и выходов λ реализуются с помощью комбинационных схем;

– как управляющие автоматы с хранимой в памяти логикой (с “запоминаемой или программируемой логикой”). Каждой выполняемой в операционном устройстве операции ставится в соответствие совокупность хранимых в памяти слов – микрокоманд, содержащих каждая информацию о микрооперациях, подлежащих выполнению в течение одного машинного такта, и указание (в общем случае зависящее от

значений входных сигналов), какое должно быть выбрано из памяти следующее слово (т.е. следующая микрокоманда).

Таким образом, в этом случае функции переходов δ и выходов λ управляющего автомата реализуются хранимой в памяти совокупностью микрокоманд.

Последовательность микрокоманд, выполняющих одну машинную команду (заданную своим **КОП**) или отдельную процедуру, образуют микропрограмму. Микропрограммы хранятся в специальной памяти микропрограмм (или управляющей памяти). Таким образом, система команд процессора реализуется в виде фрагментов–микропрограмм, которые независимо адресуются и выполняются по соответствующему **КОП** в программе на языке машинных кодов. В управляющих автоматах с хранимой в памяти программой микропрограммы используются в явной форме, они программируются в кодах микрокоманд и в таком виде запоминаются в памяти.

Такой метод управления цифровым устройством обработки информации называется микропрограммированием, а использующие этот метод управляющие блоки – микропрограммными управляющими устройствами (или автоматами).

3.3. Принцип действия управляющего автомата с хранимой в памяти логикой и микропрограммное управление

Хранимая в памяти микропрограмма должна содержать информацию о функциях переходов и выходов управляющего микропрограммного автомата. Рассматривая управляющий автомат (**УА**) в терминах цифровых автоматов, отметим, что он функционирует подобно автомату Мили с задержанными на один такт выходными сигналами.

Аргументами функций переходов и выходов автомата являются входные переменные **Z(t)** и **U(t)** и переменные **Q(t),** задающие своими значениями состояния автомата **A(t).** Набор значений аргументов обычно интерпретируется как адрес микрокоманды в блоке памяти микропрограммного автомата. Этот адрес заносится в регистр адреса микрокоманды (**РгАМК**) на время одного такта. В управляющей (или микропрограммной) памяти (**УП**) по заданному адресу хранится набор значений выходных сигналов **W(t+1)** управления операционным блоком (или блоком обработки данных (**БОД**)) и набора значений переменных **Q(t+1)**, представляющих состояние **A(t+1)**. Значения **W(t+1)** и **A(t+1)** и определяются функциями переходов и выходов. Структурная схема простейшего варианта управляющего автомата с хранимой в памяти логикой приведена на рис.64. В управляющем автомате синхросигнал **CLK** определяет такты работы, при этом значение **CLK=1** выделяет такт, а значение **CLK=0** – паузу между тактами. Значения входных и выходных сигналов и состояние автомата должны быть неизменными во время импульса такта, но могут изменяться только в паузах. Состояние автомата **A(t)** представляется набором значений переменных **Q(t).** Пусть в такте **(t–1)** в **РгАМК** занесены **U(t–1)** , **Z(t–1)** и набор (возможно – код) **Q(t–1).** Тогда в паузе перед тактом **t** при **CLK=0** на **РгАМК** эти значения сохранятся и из **УП** выбираются наборы **W(t),** которые как и **Q(t)** зависят от предыдущих значений **Q(t–1), Z(t–1)** и **U(t–1)**. Эти наборы (возможно – коды) при **CLK=0** заносятся в регистр микрокоманды (**РгМК**) и одновременно изменяются

значения входных сигналов. После, при **CLK=1**, задающего такт **t** в **РгМК** хранятся наборы **W(t)** и **Q(t)**, при этом сигналы **W(t)** используются при выполнении микроопераций в **БОД** а набор **Q(t)** переносится в **РгАМК**, после чего цикл работы управляющего автомата повторяется.

Воздействие управляющих сигналов **W(t)** на операционный блок синхронизируется сигналом **CLK=1**, обеспечивающим выдачу **W(t)** строго в такте **t** из **РгМК**, находящегося в режиме хранения.

Управляющая память бывает как постоянной (**ПЗУ**), так и оперативной, то есть допускающей запись и считывание информации. Загрузка **УП ОЗУ** производится с внешнего запоминающего устройства по шине загрузки управляющей памяти (**Ш Зг УП**) при каждом включении системы обработки цифровой информации в работу.

Анализируя структурную схему рис.64, можно заметить, что число требуемых слов, хранимых в **УП**, велико из-за большой разрядности **РгАМК** (это можно назвать большой шириной адресного пространства **УП**), обусловленной значительным числом используемых формирующих адрес сигналов **U(t)**, **Z(t)**, **Q(t)**. Сократить объём **УП** можно, если учесть, что для каждого **Q(t)** существенными являются только некоторые переменные **Z(t)** и **U(t),** задающие различные переходы из **Q(t)** в **Q(t+1)**. Количество типов таких переходов обычно невелико и, поэтому для каждого **Q(t)** можно выделить лишь небольшую группу адресов **УП**, хранящих наборы (или коды) только отличающихся переходов **Q(t+1)**.

Объём **УП** при этом во много раз сокращается, так как в **УП** не будут храниться многократно повторяющиеся наборы (или коды), описывающие одинаковые переходы автомата. Адрес микрокоманды при таком подходе формируется специальной комбинационной схемой формирования адреса микрокоманды (**СхФАМК** на рис.64) по значениям **Q(t), U(t), Z(t)**. Схема **СхФАМК** подключается ко входам **РгАМК**, как показано штриховыми линиями на рис.64.

Адрес очередной микрокоманды можно назначить без учёта значений **U(t)** и **Z(t)**, если эта микрокоманда задаёт функцию перехода автомата в состоянии, имеющем единственный переход, не зависящий от значений входных сигналов. В этом случае адрес очередной микрокоманды можно указать значениями отдельной группы разрядов исполняемой микрокоманды. Если очередная микрокоманда должна задавать функцию перехода, зависящую от значений входных сигналов, то её адрес должен также зависеть от входных сигналов.

Для упрощения **СхФАМК** обычно используется своеобразный способ формирования очередного адреса. В микрокоманде выделяется помимо операционной части (поля) ещё и адресная часть (поле). Адресная часть содержит несколько полей (групп разрядов):

– поле типа формирования адреса (**ТФА**);

– поля формирования отдельных групп разрядов адреса (**ПФА**).

При определённых значениях типа формирования адреса (**ТФА**) очередной адрес формируется только из значений бит полей формирования адреса (**ПФА**), а в простейшем случае в **РгАМК** переносятся значения бит **ПФА**, заданные в микрокоманде.

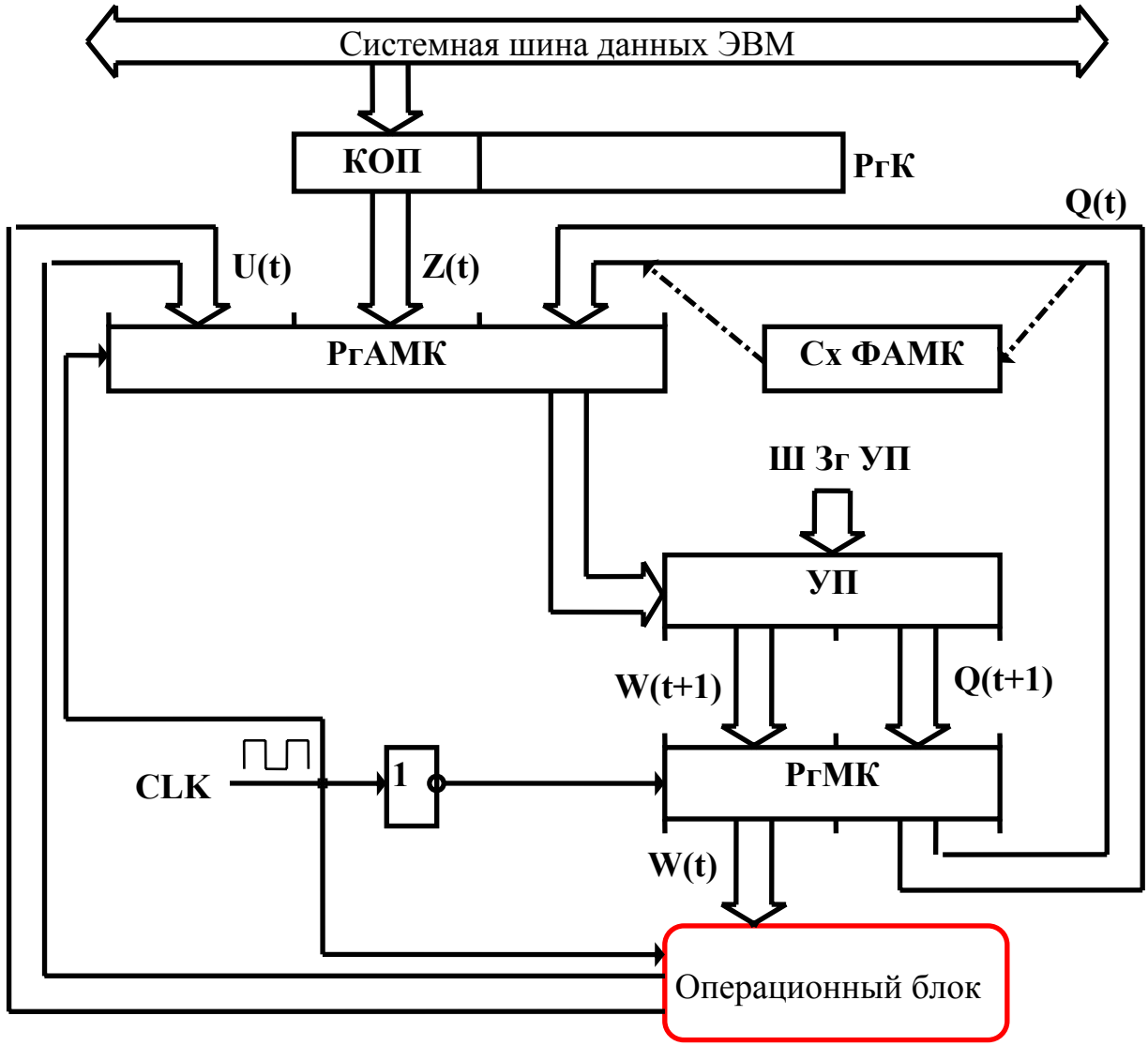


Рис.64. Структурная схема системы обработки цифровой информации с управляющим автоматом с “хранимой в памяти логикой”

Если адрес очередной микрокоманды должен формироваться с учётом значений входных сигналов, то в поле типа формирования адреса (**ТФА**) заносится специальный набор бит (или код), настраивающий **СхФАМК** на обработку полей формирования адреса (**ПФА**) по какому–либо алгоритму. При этом содержимое некоторых **ПФА** по–прежнему переносится в **РгАМК**, а биты из других полей **ПФА** обеспечивают занесение в **РгАМК** значений, указываемых этими **ПФА** входных переменных.

Таким образом, очередной адрес оказывается зависящим не только от ранее исполнявшейся микрокоманды (состояния автомата), но и от значений входных сигналов, влияющих на переходы автомата в новое состояние.

Рассмотренный способ формирования адреса следующей микрокоманды называется принудительным формированием адреса.

Иногда формирование адреса следующей микрокоманды выполняется способом естественной адресации с помощью программного счётчика (**РС**). Безусловный переход от микрокоманды с адресом **i** осуществляется к микрокоманде с адресом **i+1**, что позволяет иметь в микрокоманде только операционную часть. Однако, для принятого единственным такого формата микрокоманды, позволяющего уменьшить объём микропрограммной памяти, приходится в микропрограммы, помимо операционных, включать и адресные микрокоманды, отличающиеся специальными признаками (атрибутами) от операционных команд. В адресных микрокомандах отсутствует операционная часть, а все биты используются в качестве полей типа формирования адреса (**ТФА**) и полей формирования адреса (**ПФА**), таких же, как и в адресной части рассмотренных ранее комплексных микрокоманд.

Такой способ формирования адреса микрокоманды приводит к усложнению схем дешифрирования микрокоманд и увеличению длины микропрограмм, но сокращает длину микрокоманд, делает их структуру регулярной и тем самым повышает степень использования объёма микропрограммной памяти или уменьшает её требуемый объём.

Управляющие автоматы с хранимой в памяти логикой различаются по способу формирования управляющих функциональных сигналов. Для этого возможно использование

– горизонтального,
– вертикального,
– смешанного микропрограммирования.

3.3.1. Горизонтальное микропрограммирование

При горизонтальном микропрограммировании каждому биту операционной части микрокоманды ставится в соответствие определённый управляющий функциональный сигнал, то есть определённая микрооперация. Если бит имеет значение, равное 1, то соответствующая микрооперация выполняется независимо от значений других бит микрокоманды. При таком способе операционная часть микрокоманды содержит **m** – разрядов, где m – общее число используемых микроопераций. Достоинством горизонтального микропрограммирования является возможность одновременного выполнения в одном такте любого набора из **m** – микроопераций и простота формирования функциональных сигналов, так как последние могут инициализироваться непосредственно от битовых сигналов из регистра микрокоманд **РгМК**.

Существенным недостатком горизонтального микропрограммирования является то, что требуется большая длина микрокоманды, поскольку число функциональных сигналов в современных устройствах и системах обработки цифровой информации (процессорах ЦЭВМ) достигает нескольких сотен.

3.3.2. Вертикальное микропрограммирование

При вертикальном микропрограммировании микрооперация определяется не состоянием одного из бит микрокоманды, а ***двоичным кодом***, содержащимся в операционной части микрокоманды (отдельным кодом задаётся и отсутствие микрооперации).

Количество разрядов операционной части микрокоманды:

$$\mathbf{n_{оч}=]\log_2(m+1)[.}$$

Достоинством вертикального микропрограммирования является небольшая длина микрокоманды. Однако в этом случае требуются сложные дешифраторы для большого количества используемых микроопераций, а главное – в каждой микрокоманде указывается лишь одна микрооперация, что приводит к увеличению длины микропрограмм по сравнению с их длиной при горизонтальном микропрограммировании.

Оптимальные параметры управляющих автоматов с хранимой в памяти логикой достигаются при использовании смешанного микропрограммирования, то есть при сочетании принципов горизонтального и вертикального микропрограммирования.

3.3.3. Смешанное микропрограммирование

При смешанном микропрограммировании множество микроопераций **V** разбивается на **k** непересекающихся подмножеств (или полей)

$$\mathbf{V}=\bigcup_{l=1}^{k}\mathbf{v}_l\,.$$

Микрооперации внутри каждого из подмножеств кодируются либо горизонтальным, либо вертикальным способом. Таким образом, возможно использование двух видов смешанного микропрограммирования.

3.3.3.1. Вертикально – горизонтальное микропрограммирование

Всё множество микроопераций **V** разделяется на множество **k** подмножеств, в каждом из которых объединяют микрооперации, наиболее часто встречающиеся вместе в одном такте. Подмножества стремятся по возможности сделать равномощными, но если это не удаётся, то формат микрокоманды рассчитывается на множество максимальной мощности. Операционная часть микрокоманды (рис.65) состоит из двух частей (полей).

В первом поле, длина которого равна max $|\mathbf{V_L}|$, применён горизонтальный способ микропрограммирования, то есть каждый бит соответствует определённой микрооперации из подмножества $\mathbf{V_L}$, а другое поле, имеющее длину $\mathbf{m=]\log_2 k[}$, указывает, к какому из **k** подмножеств принадлежат микрооперации в первом поле микрокоманды.

3.3.3.2. Горизонтально – вертикальное микропрограммирование

В этом случае подмножества $\mathbf{V_L}$ представляются горизонтальным способом, а микрооперации внутри каждого из подмножеств – вертикальным способом (рис.66). Для каждого подмножества $\mathbf{V_L}$ выделяется отдельное поле в операционной части микрокоманды.

Длина операционной части микрокоманды

$$\mathbf{n_{оч} = \sum_{l=1}^{k}]\log_2 (m_l + 1)[},$$

где $\mathbf{m_l}$ – количество микроопераций, представляемых полем **l**. Микрокоманды такого типа называются микрокомандами с полевой структурой. При таком способе кодирования желательно, чтобы $\mathbf{V_i \cap V_j = \varnothing}$ (пересечение подмножеств микроопераций должно быть пусто) при $\mathbf{i \neq j}$, то есть, чтобы каждая микрооперация встречалась только в одном поле.

Рис.65. Структура микрокоманды при вертикально-горизонтальном микропрограммировании

При прямом кодировании микрокоманд каждому полю микрокоманды присваиваются фиксированные функции. Косвенное кодирование микрокоманд характеризуется наличием дополнительных полей, содержимое которых модифицирует значение функций основных полей микрокоманды.

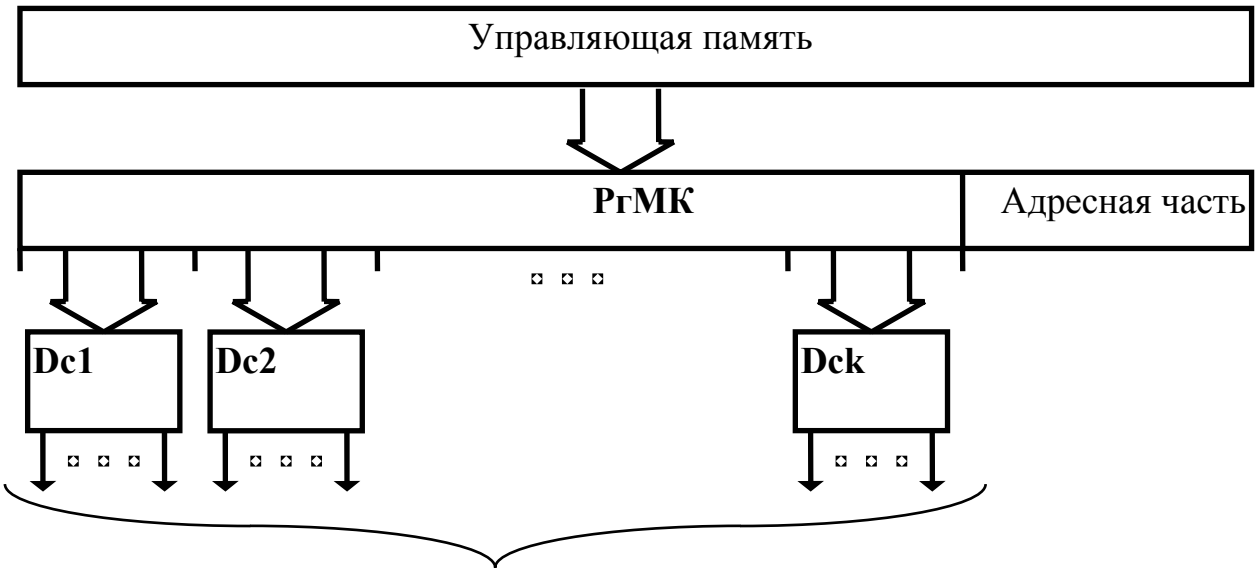


Рис.66. Фрагмент управляющего автомата при горизонтально-вертикальном микропрограммировании

Таким образом, интерпретация кодов полей, из которых дешифраторами формируются управляющие сигналы, зависит от значений дополнительных полей.

Косвенное кодирование часто используется, так как позволяет уменьшить длину микрокоманды, однако оно нарушает стройность микропрограммного управления, усложняет дешифраторы и приводит к снижению скорости работы из–за потерь времени на дешифрирование дополнительных полей микрокоманды.

По временным характеристикам микрокоманды подразделяются на однофазные и многофазные

В первом случае все микрооперации, указанные в микрокоманде, выполняются одновременно в течение одного такта.

Во втором – такт разбивается на части по фазам (или по микротактам), а указанные в микрокоманде микрооперации выполняются в различные микротакты (фазы такта).

3.4. Управляющие автоматы с “жёсткой логикой”

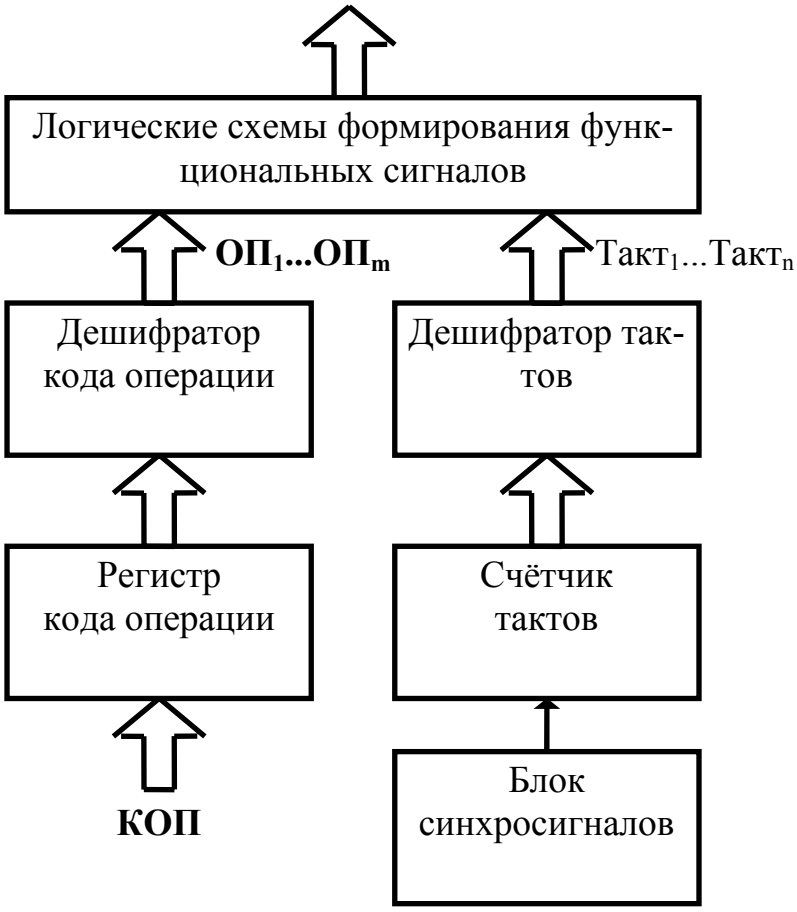


Рис.67. Структура управляющего автомата с “жёсткой логикой”

Управляющие автоматы с “жёсткой логикой” представляют собой логические схемы, вырабатывающие распределённые во времени управляющие функциональные сигналы. В отличие от цифровых управляющих автоматов с “хранимой в памяти логикой” у этих автоматов можно изменить логику работы только путём переделки схемы автомата, созданием нового управляющего автомата соответствующего изменённому алгоритму функционирования. Затраты на такую переделку управляющего автомата велики по сравнению с управляющим автоматом с “хранимой в памяти логикой”. Типичная структурная схема управляющего автомата с “жёсткой логикой” приведена на рис.67.

На счётчик тактов поступают сигналы от блока синхросигналов, а счётчик с каждым сигналом меняет своё состояние. Состояние счётчика представляет собой номера тактов, изменяющихся от 1 до **n**. Дешифратор тактов формирует на **i** – м выходе единичный сигнал при **i** – м состоянии счётчика тактов (то есть во время **i**– го такта).

Дешифратор кода операции вырабатывает единичный сигнал на **j** – м выходе, если выполняется **j**– й тип микрокоманды.

Логические схемы формирования управляющих функциональных сигналов для каждой микрокоманды образуют функциональные сигналы для выполнения требуемых в данном такте микроопераций.

3.5. Граф – схемы микропрограммных автоматов

Для описания микропрограмм необходимо знать и задавать последовательности микрокоманд и функции перехода, определяющие порядок выполнения микрокоманд. Для описание микропрограмм обычно используется язык граф–схем алгоритмов (**ГСА**).

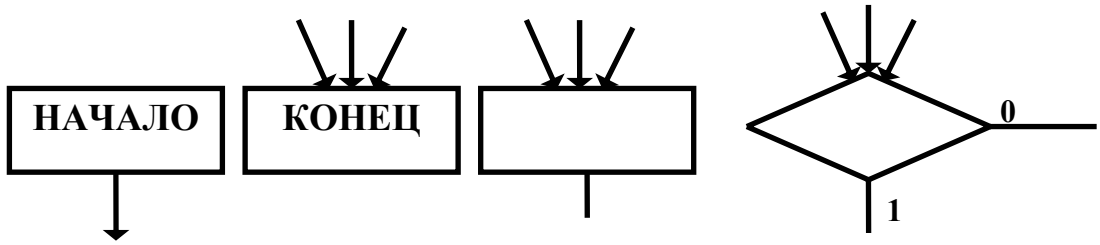


Рис. 68. Типы вершин ГСА

***Граф–схема алгоритма* (ГСА) – *ориентированный связный граф, содержащий одну начальную вершину* (A_0), *одну конечную* ($A_к$) *и произвольное количество условных* ($P=\{p_1, ..., p_F\}$) *и операторных* ($A=\{A_1, ..., A_G\}$) *вершин*.**

Конечная, операторная и условная вершины имеют по одному входу, начальная вершина входов не имеет. Начальная и операторная вершины имеют по одному выходу, а условная – два выхода, помеченных символами 0 и 1. Конечная вершина выходов не имеет.

ГСА удовлетворяет следующим условиям:

1. Входы и выходы вершин соединяются друг с другом с помощью дуг направленных всегда от выхода ко входу.

2. Каждый выход соединён точно с одним входом.

3. Любой вход соединяется по крайней мере с одним выходом.

4. Любая вершина графа лежит по крайней мере на одном пути из начальной вершины к конечной вершине.

5. Один из выходов условной вершины может соединяться с её входом, что недопустимо для операторной вершины. Такие условные вершины называются возвратными вершинами.

6. В каждой условной вершине записывается один из элементов множества $\mathbf{X=\{x_1, ..., x_l, ..., x_L\}}$, называемого множеством логических условий. Разрешается в различных условных вершинах запись одинаковых элементов множества **X**.

7. В каждой операторной вершине записывается оператор (микрокоманда) $\mathbf{Y_t}$ – подмножество множества $\mathbf{Y=\{y_1, ..., y_n, ..., y_N\}}$, называемого множеством микроопераций $\mathbf{Y_t=(y_{t1}, ..., y_{tu}, ..., y_{tU})}$; $\mathbf{y_{tn} \in Y}$, $\mathbf{u=1, ..., U}$.

При $\mathbf{U=0}$, $\mathbf{Y_t=\varnothing}$ (пусто), что допустимо. Разрешается также запись в различных операторных вершинах одинаковых подмножеств множества микроопераций.

При большом количестве дуг, приходящих на один вход вершины, **ГСА** выглядит неаккуратно, поэтому разрешается изображать дуги, входящие в вершины, так, как показано на рис.69.

Пример **ГСА**, содержащего 6 условных и 7 операторных вершин приведён на рис.70.

Предположим, что в операторных вершинах **ГСА** записаны операторы $\mathbf{Y_t, ... , Y_T}$ – все разные, так что операторную вершину можно отождествить с записанным в ней оператором (в вершине $\mathbf{A_i}$ записан оператор $\mathbf{Y_i}$. В таком случае вместо обозначения $\mathbf{A_i}$ для операторной вершины можно использовать символы $\mathbf{Y_i}$).

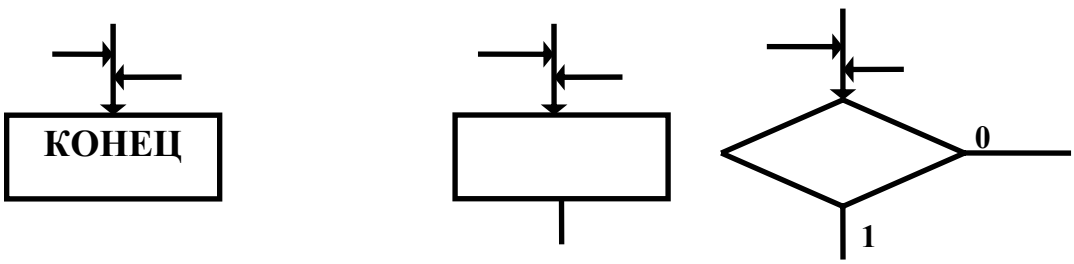


Рис. 69. Допустимые изображения вершин **ГСА**

Начальной вершине поставим в соответствие оператор $\mathbf{Y_0}$, а конечной – оператор $\mathbf{Y_k=Y_{T+1}}$. Пусть в **ГСА** имеется путь из вершины $\mathbf{Y_i}$ **(i=0, 1, ... , T)** в вершину $\mathbf{Y_j}$ **(j=1, ... , T, T+1)** вида

$$\mathbf{Y_i p_{i1}^{li1} ... p_{ir}^{lir} ... p_{iR}^{liR} Y_j} \ , \qquad \mathbf{(A)}$$

проходящий только через условные вершины $p_{i1}, \ldots, p_{iR}$; где $l_{ir} \in \{0,1\}$ – символ, приписанный выходу условной вершины p_{ir}, через который проходит путь (**A**).

Каждому такому пути соответствует элементарная конъюнкция $б_{ij} = x_{i1}^{li1} \ldots x_{iR}^{liR} = \&_{r=1}^{R} x_{ir}^{lir}$, где x_{ir} – логическое условие, записанное в условной вершине p_{ir}; $x_{ir}^{0} = \bar{x}_{ir}$, $x_{ir}^{1} = x_{ir}$.

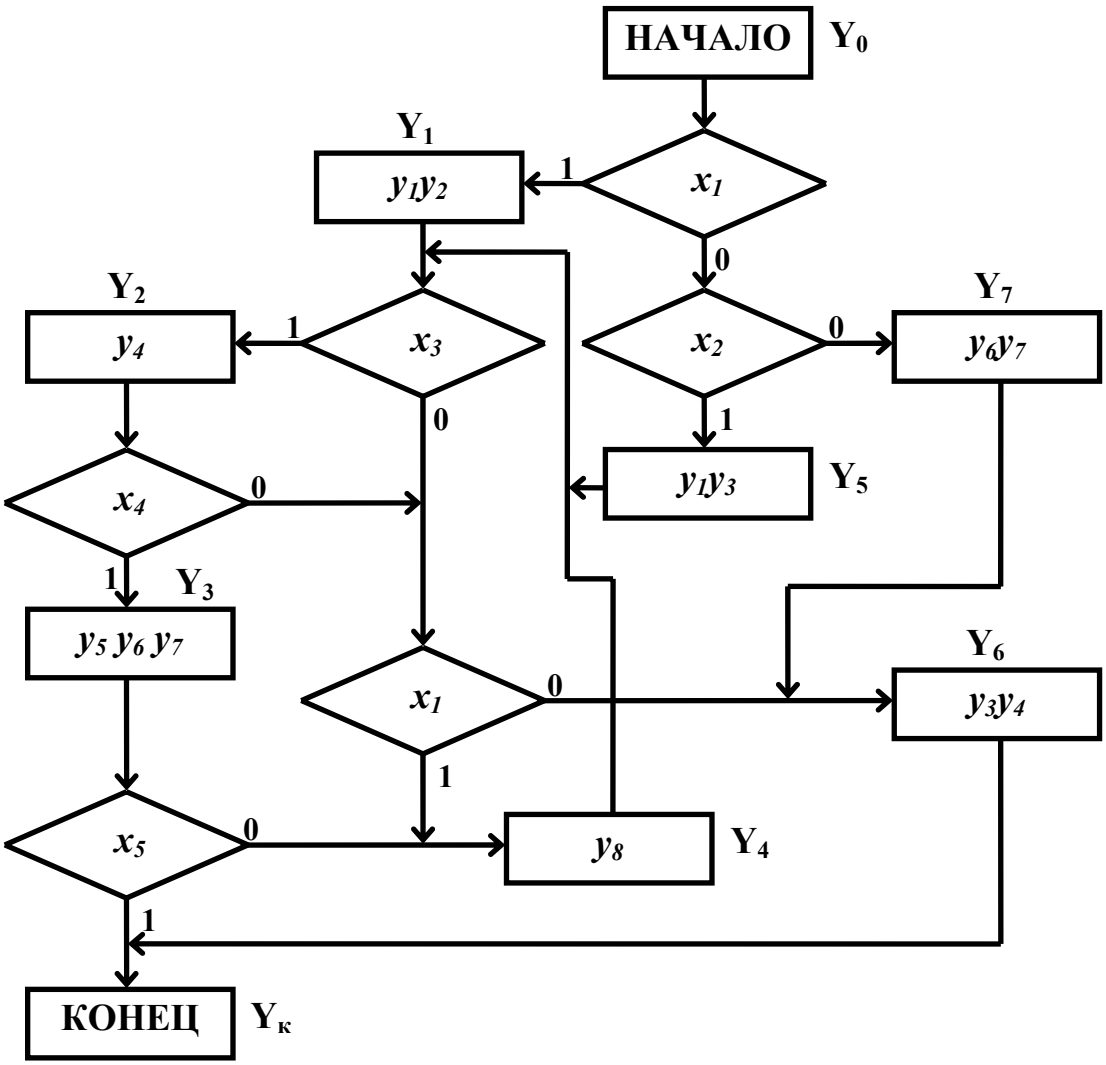


Рис. 70. Пример **ГСА**

Если между вершинам Y_i и Y_j имеется несколько (например **H**) путей, проходящих через условные вершины, то α_{ij} равно дизъюнкции элементарных конъюнкций, соответствующих всем путям, то есть

$б_{ij} = \bigvee_{h=1}^{H} б_{ij}^{h}$, где $б_{ij}^{h}$ – конъюнкция, соответствующая **h** – му пути из Y_i в Y_j.

Таким образом, α_{ij} – функция перехода от оператора (микрокоманды) Y_i к оператору (микрокоманде) Y_j.

Всевозможные наборы значений переменных $\tilde{x}_1,...,\tilde{x}_L$ обозначим через $\Delta_1,...,\Delta_{2^L}$. Определим процесс выполнения **ГСА**, начиная с оператора Y_0 (начальный оператор), на произвольной бесконечной последовательности наборов Δ_{m1}, ..., Δ_{mg}, ... следующим образом.

Выписываем оператор Y_0.

ШАГ 1. Придаём переменным $\tilde{x}_1,...,\tilde{x}_L$ значения из набора Δ_{m1}. Из множества функций перехода $\alpha_{01}, ..., \alpha_{0T}$ выбираем функцию α_{0i1} $(i_1 \in \{1, ..., T\})$, такую, что $\alpha_{0i1}(\Delta_{m1}) = 1$.

В строчку рядом с Y_0 записываем Y_{i1}

$Y_0\ Y_{i1}$.

ШАГ 2. Придаём переменным $\tilde{x}_1,...,\tilde{x}_L$ значения из набора Δ_{m2}. Из множества функций перехода $\alpha_{i11}, ..., \alpha_{i1T}$ выбираем функцию α_{i1i2} $(i_2 \in \{1, ..., T\})$, такую, что $\alpha_{i1i2}(\Delta_{m2}) = 1$.

В строчку рядом с Y_{i1} записываем Y_{i2}

$Y_0\ Y_{i1}\ Y_{i2}$ и т. д.

Пусть перед некоторым q – м шагом имеем строчку операторов

$$Y_0\ Y_{i1}\ Y_{i2} ...\ Y_{i_{q-1}}.$$

Если на наборе Δ_{mq} некоторая функция перехода $б_{i_{q-1}i_q}$ равна единице $(i_q \in \{1, ..., T\})$, то в выписанную строчку операторов добавляем Y_{i_q}.

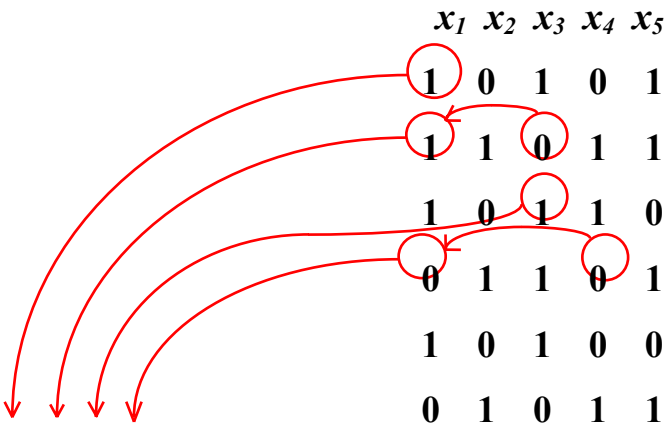
Если оказывается, что $б_{i_{q-1}T+1}(Д_{m_q}) = 1$, то в строчку вслед за $Y_{i_{q-1}}$ записываем Y_{T+1}(конечный оператор), и процесс выполнения **ГСА** завершается.

Строчка $Y_0\ Y_{i1}\ Y_{i2} ...\ Y_{i_{q-1}}\ Y_{i_q}$ называется значением **ГСА** на последовательности наборов $\Delta_{m1}, \Delta_{m2}, ..., \Delta_{m_{q-1}}, \Delta_{m_q}$. Процедура выполнения **ГСА** на заданной последовательности наборов переменных может прекратиться в двух случаях:

– исчерпаны все наборы (заключительным в значении **ГСА** стоит оператор Y_{i_q});

– достигнута конечная вершина (заключительным в значении **ГСА** стоит конечный оператор Y_{T+1}).

Например, значение **ГСА** на по рис.70 на последовательности наборов



равно $Y_0Y_1Y_4Y_2Y_6Y_к$.

Рассмотрим процедуру выполнения ГСА на произвольной последовательности наборов при наличии возвратных условных вершин на примере фрагмента **ГСА**, приведенного на рис. 71.

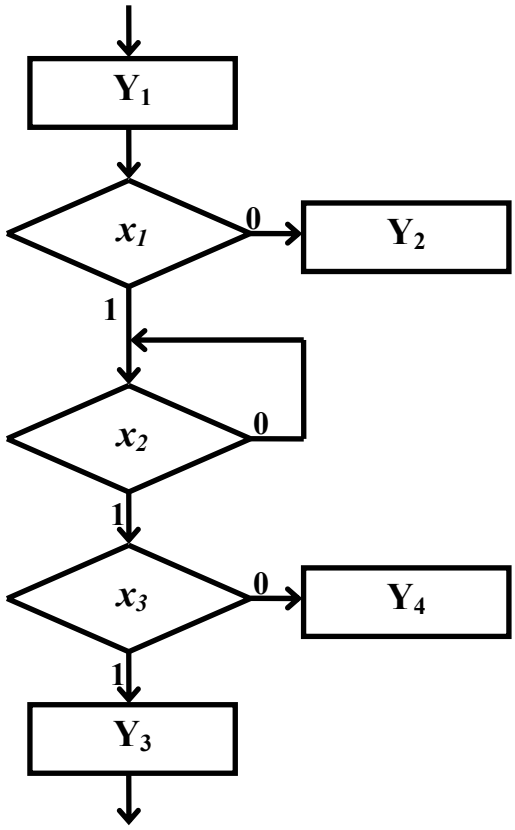


Рис.71. Фрагмент **ГСА** с возвратной условной вершиной

Между операторами Y_1 и Y_3 имеется бесчисленное множество путей, в связи с чем

$$б_{13} = б_{13}^{1} \vee б_{13}^{2} \vee \ldots \vee б_{13}^{n} \ldots = x_1 x_2 x_3 \vee x_1 (\bar{x}_2 x_2) x_3 \vee x_1 \bar{x}_2 (\bar{x}_2 x_2) x_3 \vee$$
$$\ldots \vee x_1 \bar{x}_2 \bar{x}_2 \ldots (\bar{x}_2 x_2) x_3 \vee \ldots = x_1 x_2 x_3 .$$

Так как логические условия, соответствующие всем путям, начиная со второго, равны нулю, α_{13} фактически определяется первым путём. В связи с этим в дальнейшем учитываются лишь те пути через возвратные вершины, которые проходят через выход, не соединённый с её входом. Таким образом, на рис.71 имеется лишь один путь вида

$$Y_i p_{i1}^{li1} \ldots p_{ir}^{lir} \ldots p_{iR}^{liR} Y_j \qquad \textbf{(A)}$$

из вершины Y_1 в вершину Y_3. Тогда

$$б_{12} = \bar{x}_1, \quad б_{13} = x_1 x_2 x_3, \quad б_{14} = x_1 x_2 \bar{x}_3 ; \qquad \textbf{(C)}$$

Пусть в процессе выполнения **ГСА** на некоторой последовательности наборов выполнился оператор Y_1, после чего в этой последовательности стоят наборы значений переменных x_i

101,
111.

Из выражений (**C**) видно, что ни одна функция перехода из оператора Y_1 на наборе **101** не равна единице, поэтому в строку рядом с Y_1 записывается пустой оператор Y_0

$$\ldots Y_1 Y_0$$

и выполняется переход к следующему набору (**111**), на котором проверяется значение функции перехода из того же оператора Y_1**.** На наборе **111** α_{13}**=1**, а поэтому рядом с Y_0 записывается оператор Y_3

$$\ldots Y_1 Y_0 Y_3 \ldots$$

Если бы и на этом наборе ни одна из функций перехода не была бы равна единице, то рядом с Y_0 был бы записан ещё один пустой оператор Y_0 и так до тех пор, пока на каком–то наборе не появилось значение некоторой функции перехода, равной единице.

Такое выполнение **ГСА** соответствует тому, что при значениях переменных, при которых ни одна функция перехода из последнего выполненного оператора не равна единице, управляющее устройство не выдаёт сигналов на выполнение микроопераций в операционном блоке (выполняется микрокоманда Y_0) и тем самым реализуется ожидание в работе устройства.

3.6. Синтез микропрограммных автоматов по граф–схеме алгоритма

3.6.1. Синтез микропрограммного автомата Мили

Конечный автомат, реализующий микропрограмму работы дискретного устройства, называется микропрограммным автоматом.

Синтез микропрограммного автомата Мили по граф – схеме алгоритма осуществляется в два этапа:

– получение отмеченной **ГСА**;

– построение графа автомата.

На этапе получения отмеченной **ГСА** входы вершин, следующих за операторными, отмечаются символами $\mathbf{a_1}$, $\mathbf{a_2}$, ... по следующим правилам:

– символом $\mathbf{a_1}$ отмечаем вход вершины, следующей за начальной и вход конечной вершины;

– входы всех вершин, следующих за операторными, должны быть помечены;

– если вход вершины помечается, то только одним символом;

– входы различных вершин, за исключением конечной, помечаются различными символами.

Для расстановки отметок потребуется конечное число символов. Для определённости входы вершин помечаются символами $\mathbf{a_1}, \dots, \mathbf{a_M}$.

Результатом выполнения первого этапа является отмеченная **ГСА**, которая служит основой для выполнения второго этапа – переходу к графу автомата. Применение первого этапа к граф–схеме автомата на рис.70 даёт отмеченную **ГСА**, изображённую на рис.72.

Если идти от одной отметки $\mathbf{a_m}$ к другой отметке $\mathbf{a_s}$ в направлении ориентации дуг **ГСА**, выписывая содержимое лежащих на этом пути вершин, то каждому такому пути можно поставить в соответствие слово в алфавите:

$$\{x_1^{l_1}, \dots, x_1^{l_L}, \mathbf{Y_1}, \dots, \mathbf{Y_T}\},$$

где $l_l =$ $\begin{cases} \mathbf{1} \text{ при выходе из условной вершины по 1 в соответствующем пути,} \\ \mathbf{0} \text{ при выходе из условной вершины по 0.} \end{cases}$

Таким образом, $x_l^0 = \bar{x}_l$; $x_l^1 = x_l$, $l = 1, \dots, L$. Для обозначения того, что выписанное слово соответствует пути по **ГСА** из $\mathbf{a_m}$ в $\mathbf{a_s}$, это слово ограничивается слева и справа символами $\mathbf{a_m}$ и $\mathbf{a_s}$ соответственно.

Практический интерес представляют слова двух видов:

$$\mathbf{a_m} x_{m1}^{l_{m1}} \dots x_{mR}^{l_{mR}} \mathbf{Y_t a_s} \quad (\mathbf{a_m}, \mathbf{a_s} \in \{\mathbf{a_1}, \dots, \mathbf{a_M}\}); \qquad \text{(D)}$$

$$\mathbf{a_m} x_{m1}^{l_{m1}} \dots x_{mR}^{l_{mR}} \mathbf{a_1} \quad (\mathbf{a_m} \in \{\mathbf{a_2}, \dots, \mathbf{a_M}\}). \qquad \text{(E)}$$

Соответствующие этим словам пути в граф – схеме называются путями перехода. В пути вида (D) возможен случай **R=0**, то есть

$$\mathbf{a_m Y_t a_s}.$$

Таким образом, путь вида (D) – это путь в **ГСА** из одной отметки $\mathbf{a_m}$ в другую отметку $\mathbf{a_s}$ (допустимо $\mathbf{a_m}=\mathbf{a_s}$), содержащий точно одну операторную вершину. Путь вида (E) – это путь из некоторой отметки $\mathbf{a_m}$ в отметку $\mathbf{a_1}$ (недопустимо $\mathbf{a_m}=\mathbf{a_1}$), проходящий точно через все условные вершины.

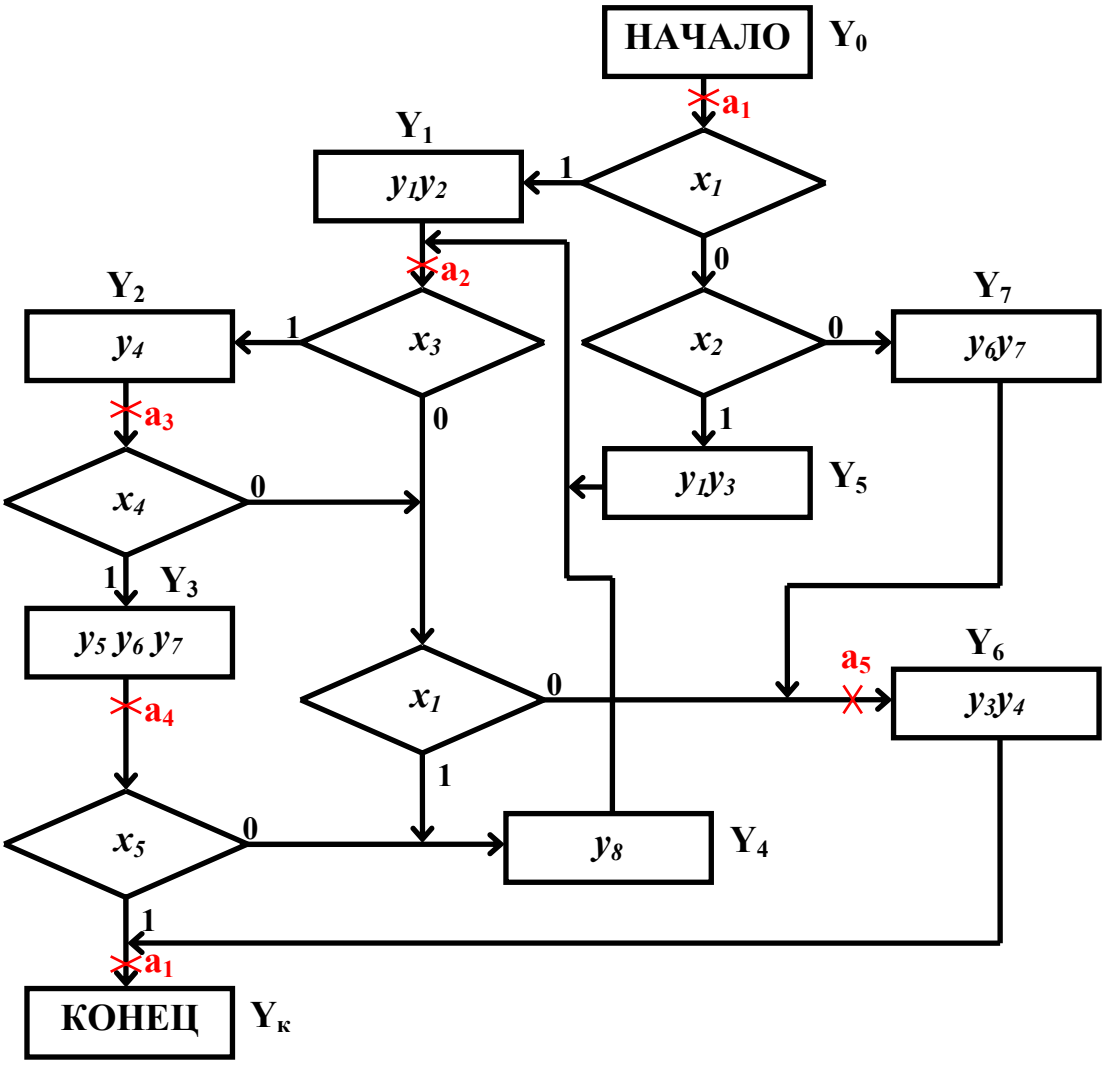


Рис. 72. Пример отмеченной **ГСА** при синтезе автомата Мили

Каждому пути (или слову) типов (**D**) или (**E**) ставится в соответствие конъюнкция

$$\mathbf{X}(a_m, a_s) = \&_{r=1}^{R} x_{mr}^{lmr}.$$

Для краткости эти пути обозначаются:

$$a_m\ X(a_m,a_s)\ Y(a_m,a_s)\ a_s\ ;$$

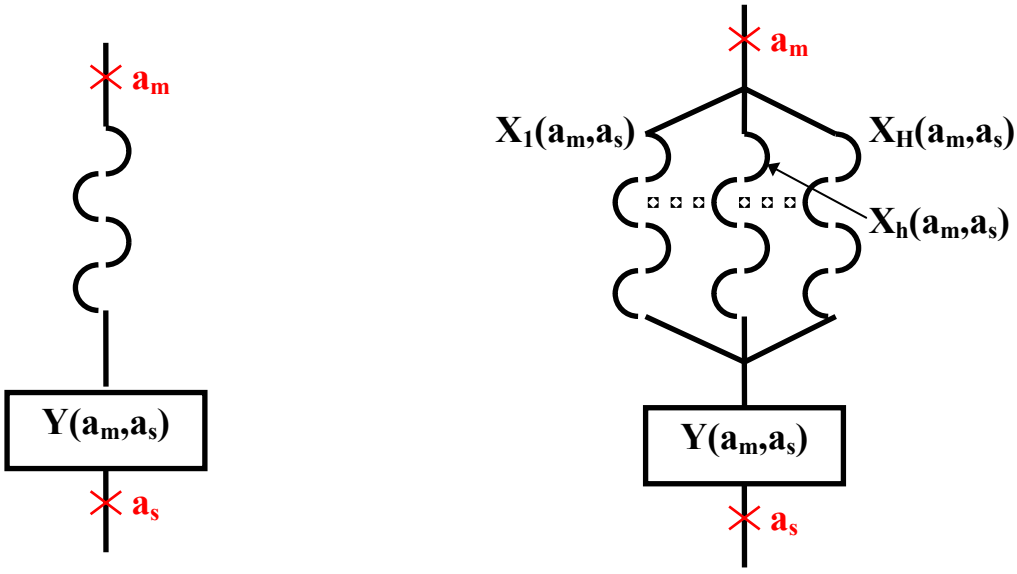
$$a_m\ X(a_m,a_1)\ a_1,$$

где $Y(a_m,a_s) = Y_t$.

Схематично пути перехода из a_m в a_s изображается так, как показано на рис.73, где волнистой линии соответствует путь через условные вершины.

Если между a_m и a_s имеется несколько параллельных путей (рис.73б) вида $a_m\ X_h(a_m,a_s)\ Y(a_m,a_s)\ a_s$ **(h=1, ... , H),** проходящих через одну и ту же операторную вершину, то указанному множеству путей соответствует дизъюнкция

$\mathbf{X(a_m,a_s)} = \bigvee_{h=1}^{H} \mathbf{X_h(a_m,a_s)}$, а само множество путей обозначается, как и ранее, $\mathbf{a_m}$ $\mathbf{X(a_m,a_s)}$ $\mathbf{Y(a_m,a_s)}$ $\mathbf{a_s}$ и называется обобщённым путём перехода.



а) б)

Рис.73. Условное графическое изображение путей перехода между $\mathbf{a_m}$ и $\mathbf{a_s}$

а) – один путь;

б) – H путей.

После получения отмеченной **ГСА** строится граф автомата Мили **S**, состояниями которого будут $\mathbf{a_1, \dots, a_M}$, причём $\mathbf{a_1}$ – начальное состояние. Переходы и выходные сигналы в этом автомате определяются следующим образом:

– находятся все пути перехода в отмеченной **ГСА**. Если при некотором **r (r=1, ... , R)** имеется несколько вхождений символа x_r^{lr} в путь перехода, то все символы x_r^{lr}, кроме одного, из этого пути удаляются. Если при некотором **r (r=1, ... , R)** в путь перехода входят как x_r, так и $\overline{x}_r$, то такой путь перехода из дальнейшего рассмотрения исключается;

– каждому пути перехода по отмеченной **ГСА** микропрограммного автомата вида $\mathbf{a_m}$ $\mathbf{X(a_m,a_s)}$ $\mathbf{Y(a_m,a_s)}$ $\mathbf{a_s}$ $\mathbf{(a_m,a_s \in \{a_1, \dots, a_M\})}$ ставится в соответствие переход автомата **S** из состояния $\mathbf{a_m}$ в состояние $\mathbf{a_s}$ под действием входного сигнала $\mathbf{X(a_m,a_s)}$ с выдачей выходного сигнала $\mathbf{Y(a_m,a_s)}$;

– каждому пути перехода вида $\mathbf{a_m}$ $\mathbf{Y(a_m,a_s)}$ $\mathbf{a_s}$ $\mathbf{(a_m,a_s \in \{a_1, \dots, a_M\})}$ ставится в соответствие переход автомата **S** из состояния $\mathbf{a_m}$ в состояние $\mathbf{a_s}$, где $\mathbf{a_m}$ и $\mathbf{a_s}$ определяются так же, как и раньше. Входной сигнал на этом переходе также равен конъюнкции содержимого условных вершин на этом пути. Так как множество условных вершин на этом пути пусто, а конъюнкция пустого множества переменных равна единице, то

соответствующий входной сигнал считается равным **1**, а выходной сигнал, как и ранее, **$Y(a_m,a_s)$;**

– каждому пути перехода вида a_m $X(a_m,a_1)$ a_1 ставится в соответствие переход из a_m в a_1. При этом a_m и входной сигнал определяются как и раньше, а выходной сигнал полагается равным Y_0 (пустой оператор).

В результате получается граф автомата Мили **S**, имеющий столько же состояний, сколько символов потребовалось для отметки входов вершин **ГСА**. Граф автомата Мили, соответствующий примеру на рис. 72, приведён на рис. 74.

Особенность микропрограммных автоматов, синтезированных описанным выше способом, состоит в том, что приписанные дугам графа входные сигналы являются элементарными конъюнкциями тех переменных, которые входят в соответствующие пути перехода, причём ранг (длина) этих конъюнкций существенно меньше числа **L** всех входных переменных.

3.6.2. Синтез микропрограммного автомата Мура

Синтез автомата Мура по граф–схеме алгоритма также состоит из двух этапов:

– получение отмеченной **ГСА**;

– построение графа автомата.

На первом из этих этапов начальная, конечная и операторные вершины отмечаются символами a_1 , ... , a_M по следующим правилам:

– символом a_1 помечаются начальная и конечная вершины;

– различные операторные вершины все отмечаются различными символами;

– все операторные вершины должны быть помечены.

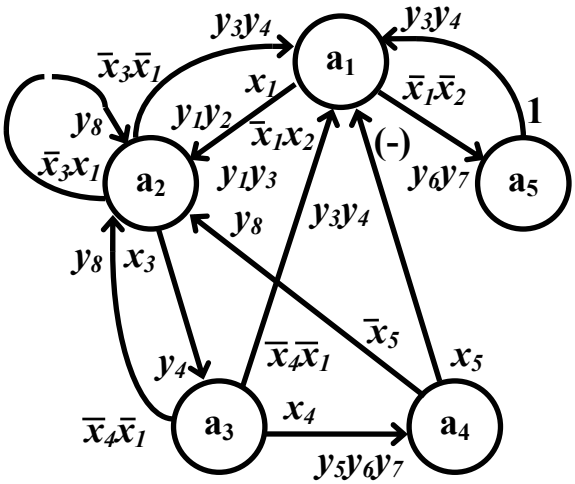


Рис.74. Граф автомата Мили, построенный по **ГСА** рис. 72

Таким образом, при синтезе автомата Мура, в отличие от автомата Мили, помечаются не входы вершин, следующих за операторными, а сами операторные вершины. За счёт начальной и конечной вершин общее количество пометок будет на единицу больше числа операторных вершин в **ГСА**.

В результате применения рассмотренной процедуры получения **ГСА** на рис.75 получается отмеченная **ГСА** микропрограммного автомата по примеру рис.70. По-

строение графа автомата Мура начинается с нахождения в отмеченной **ГСА** путей перехода вида:

$$\mathbf{a}_m x_{m1}^{l_{m1}} \ldots x_{mr}^{l_{mr}} \ldots x_{mR}^{l_{mR}} \mathbf{a}_s \quad (\mathbf{a}_m, \mathbf{a}_s \in \{\mathbf{a}_1, \ldots, \mathbf{a}_M\});$$
$$l_{mr} \in \{0,1\}, \; r = 1, \ldots, R, \; x_{mr}^0 = \overline{x}_{mr}; \; x_{mr}^1 = x_{mr}.$$

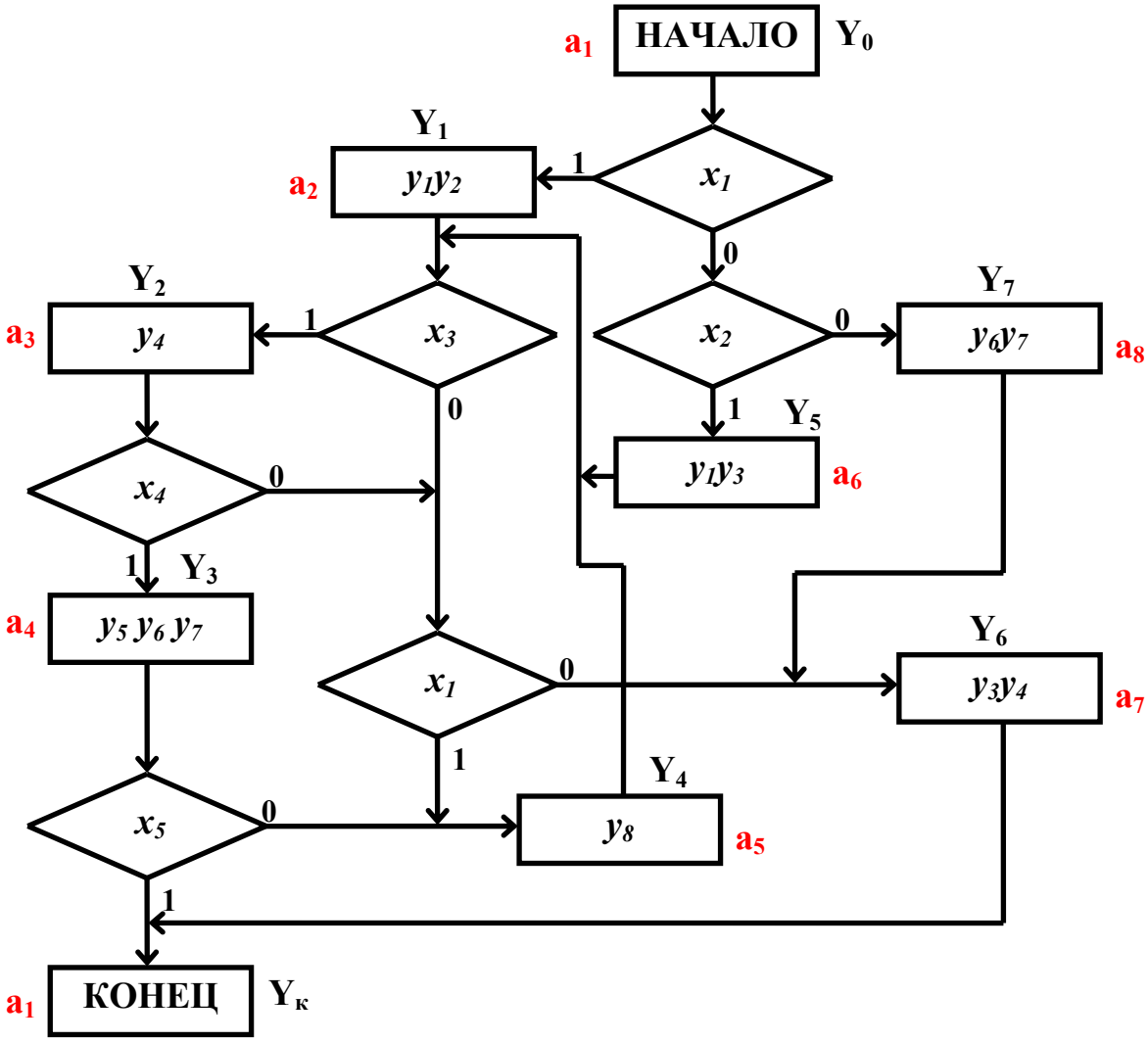


Рис. 75. Пример отмеченной **ГСА** при синтезе автомата Мура

Как и ранее для краткости эти пути обозначаются $\mathbf{a}_m \; \mathbf{X}(\mathbf{a}_m,\mathbf{a}_s) \; \mathbf{a}_s$, причём, если между $\mathbf{a}_m$ и $\mathbf{a}_s$ имеется множество путей вида $\mathbf{a}_m \; \mathbf{X}_h(\mathbf{a}_m,\mathbf{a}_s) \; \mathbf{a}_s$ при **(h=1, ... , H)**, то считается, что этому множеству путей соответствует дизъюнкция $\mathbf{X}(\mathbf{a}_m,\mathbf{a}_s) = \bigvee_{h=1}^{H} \mathbf{X}_h(\mathbf{a}_m,\mathbf{a}_s)$, а само множество путей по–прежнему обозначается $\mathbf{a}_m \; \mathbf{X}(\mathbf{a}_m,\mathbf{a}_s) \; \mathbf{a}_s$ и называется обобщённым путём перехода.

В пути вида $\mathbf{a}_m x_{m1}^{l_{m1}} \ldots x_{mr}^{l_{mr}} \ldots x_{mR}^{l_{mR}} \mathbf{a}_s$, $r = 1, \ldots, R$ не исключено **R=0**, когда между операторными вершинами не расположено ни одной условной вершины и путь $\mathbf{a}_m x_{m1}^{l_{m1}} \ldots x_{mr}^{l_{mr}} \ldots x_{mR}^{l_{mR}} \mathbf{a}_s$ превращается в путь $\mathbf{a}_m \mathbf{a}_s$.

После получения отмеченной **ГСА** стоится граф автомата Мура **S**, состояниями которого являются полученные на предыдущем этапе пометки $\mathbf{a}_1, \ldots, \mathbf{a}_M$. Переходы и выходные сигналы в этом автомате определяются следующим образом:

– каждому пути перехода $\mathbf{a}_m$ $\mathbf{X}(\mathbf{a}_m,\mathbf{a}_s)$ $\mathbf{a}_s$ $(\mathbf{a}_m,\mathbf{a}_s \in \{\mathbf{a}_1, \ldots, \mathbf{a}_M\})$ ставится в соответствие переход автомата **S** из состояния $\mathbf{a}_m$ в состояние $\mathbf{a}_s$ под действием входного сигнала $\mathbf{X}(\mathbf{a}_m,\mathbf{a}_s)$, а пути перехода $\mathbf{a}_m$ $\mathbf{a}_s$ – под действием единичного сигнала;

– при определении выходных сигналов учитывается, что в каждом состоянии независимо от того, откуда в него произошёл переход, выдаётся выходной сигнал, который записан в операторной вершине, отмеченной символом этого состояния.

На рис.76 приведён граф автомата Мура, построенного по отмеченной **ГСА** примера на рис.75.

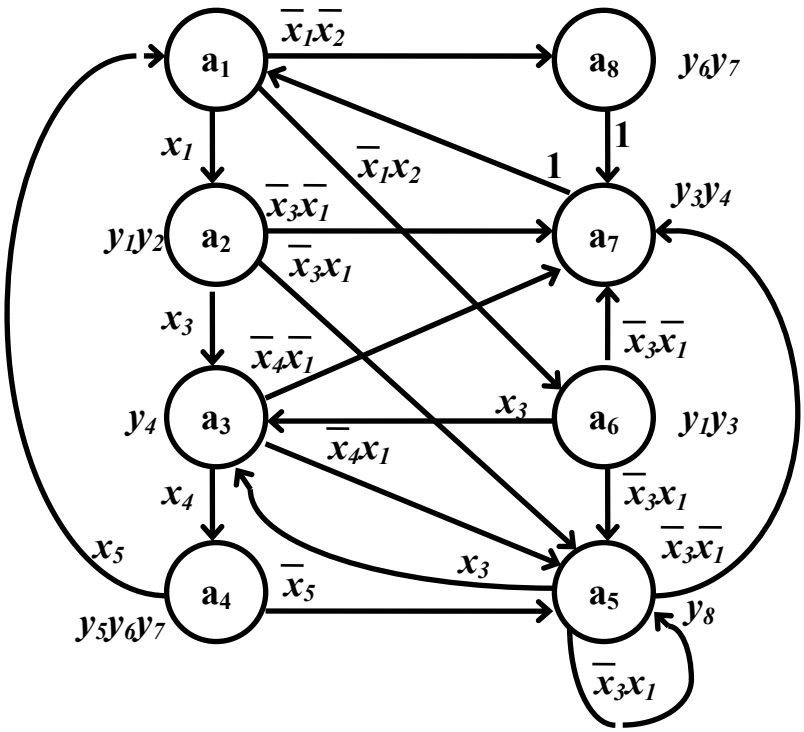


Рис.76. Граф автомата Мура, построенного по примеру **ГСА** рис.75

При использовании графов для описания автоматов с большим количеством состояний и переходов наглядность изображения автомата теряется и оказывается предпочтительнее описывать такие автоматы с помощью таблиц. Исходя из структуры **ГСА** определяются требования к таблице переходов микропрограммного автомата, в которую включаются четыре столбца:

$\mathbf{a}_m$ – исходное состояние;

a_s – состояние перехода;

$X(a_m, a_s)$ – входной сигнал на переходе **(a_m, a_s);**

$Y(a_m, a_s)$ – выходной сигнал на переходе **(a_m, a_s).**

Таким образом, каждому пути перехода соответствует одна строка таблицы переходов. Прямой таблицей переходов микропрограммного автомата называется таблица, в которой последовательно перечисляются все переходы сначала из первого состояния, затем из второго и так далее (пример – табл.43).

Таблица 43

Прямая таблица переходов автомата Мили **S**

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$
a_1	a_2	x_1	y_1y_2
	a_2	$\bar{x}_1x_2$	y_1y_3
	a_5	$\bar{x}_1\bar{x}_2$	y_6y_7
a_2	a_1	$\bar{x}_3\bar{x}_1$	y_3y_4
	a_2	$\bar{x}_3x_1$	y_8
	a_3	x_3	y_4
a_3	a_1	$\bar{x}_4\bar{x}_1$	y_3y_4
	a_2	$\bar{x}_4x_1$	y_8
	a_4	x_4	$y_5y_6y_7$
a_4	a_1	x_5	-
	a_2	$\bar{x}_5$	y_8
a_5	a_1	1	y_3y_4

В ряде случаев оказывается удобным пользоваться обратной таблицей переходов, в которой столбцы обозначены точно также, но сначала записываются все переходы в первое состояние, затем во второе и так далее.

В случае описания автомата Мура можно обойтись всего тремя столбцами, записывая выходной сигнал в прямой таблице рядом с соответствующим ему исходным состоянием, а в обратной – рядом с состоянием перехода. Обратной является табл. 44 для автомата Мура, построенная по рис.75.

Таблицы переходов микропрограммного автомата (прямые или обратные) составляются непосредственно по **ГСА**, когда в эти таблицы заносятся все пути переходов. Поскольку графическое и табличное описания микропрограммных автоматов равноценны, то для заполнения таблиц переходов автомата предварительного составления графа автомата не требуется.

Таблица 44

Обратная таблица переходов автомата Мура

a_m	a_s , $Y(a_s)$	$X(a_m,a_s)$
a_4	a_1 (-)	x_5
a_7		1
a_1	a_2 (y_1y_2)	x_1
a_2	a_3 (y_4)	x_3
a_5		x_3
a_6		$\bar{x}_3x_1$
a_3	a_4 ($y_5y_6y_7$)	$\bar{x}_3x_1$
a_2	a_5 (y_8)	$\bar{x}_5$
a_3		$\bar{x}_3x_1$
a_4		$\bar{x}_3x_1$
a_5		
a_6		
a_1	a_6 (y_3y_4)	$\bar{x}_3\bar{x}_1$ $\bar{x}_4\bar{x}_1$
a_2	a_7 (y_3y_4)	$\bar{x}_3\bar{x}_1$
a_3		$\bar{x}_3\bar{x}_1$
a_5		1
a_6		
a_8		
a_1	a_8 (y_6y_7)	$\bar{x}_1\bar{x}_2$

3.6.3. Минимизация микропрограммных автоматов

Изложенный ранее метод минимизации абстрактных автоматов применяется и для минимизации полностью определённых микропрограммных автоматов.

Если два состояния автоматы Мили совместимы, то под действием одинаковых входных сигналов они выдают одинаковые выходные сигналы. По этой причине для

выявления совместимых состояний микропрограммного автомата Мили необходимо сравнить множество микрокоманд, выдаваемых на переходах из этих состояний.

Для минимизации микропрограммного автомата Мура необходимо предварительно найти разбиение состояний на классы **0** – совместимости. В каждый такой класс попадут состояния, отмеченные одинаковой микрокомандой. Далее процесс минимизации микропрограммного автомата полностью совпадает с процессом минимизации абстрактных автоматов Мили и Мура.

Таким образом, методы минимизации микропрограммных и абстрактных автоматов очень близки. Разница заключается в том, что вместо сравнения абстрактных входных/выходных сигналов сравниваются конкретные булевы функции, соответствующие выходным сигналам микропрограммных автоматов.

Заключение

В рассмотренном выше учебно–методическом пособии теория цифровых автоматов рассмотрена с позиции оценки её практической ценности. Как раздел теории цифровых 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 цифровых 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.

Л И Т Е Р А Т У Р А

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.

ОГЛАВЛЕНИЕ