Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

РАДИОТЕХНИЧЕКИЙ ФАКУЛЬТЕТ (РТФ)

А. Н. Булдаков

ЦИФРОВЫЕ УСТРОЙСТВА И МИКРОПРОЦЕССОРЫ

Учебное пособие

Томск
2022

УДК [004.3 + 004.318](075.8)
ББК 39.973я73
Б 907

Булдаков А. Н.

Б 907 Цифровые устройства и микропроцессоры : учебное пособие / А. Н. Булдаков. – Томск : ТУСУР, 2022. – 218 с.

Рассмотрены системы счисления и арифметические операции в двоичной системе счисления, логические основы цифровых устройств, синтез и анализ комбинационных схем, основные узлы цифровых устройств и законы их работы, реализация комбинационных схем с использованием электронных элементов и узлов, схемы триггеров, регистров, счетчиков, синтез автоматов, архитектура микропроцессора Intel 8080 и микроконтроллера Intel 8051.

Для студентов направлений 11.03.01 «Радиотехника» по курсу «Цифровые устройства и микропроцессоры» и 11.03.02 «Инфокоммуникационные технологии и системы связи» по курсу «Вычислительная техника».

Рассмотрена на заседании кафедры ТУ, протокол №6 от 12 мая 2022г.

© Булдаков А. Н., 2022
© Оформление.
РТФ, ТУСУР, 2022

Оглавление

Введение

Что такое цифровое устройство? Это такое техническое устройство, которое предназначено для обработки информации, представленной в цифровой форме. При этом используются цифровые технологии, основанные на специальном математическом аппарате.

Первые цифровые устройства (ЦУ) были электромеханическими. Затем, по мере развития электроники и микроэлектроники, ЦУ строились на полупроводниковых элементах, таких как диоды и транзисторы. В дальнейшем удалось «упаковать» на одном кристалле полупроводника несколько диодов и транзисторов. Так появились интегральные микросхемы (МС). Развитие технологии позволило размещать на одном кристалле десятки, затем сотни, тысячи и более полупроводниковых элементов. Это уже были МС низкой, средней и высокой степени интеграции. И, наконец, когда степень интеграции достигла величин 106–109 электронных компонентов на одном кристалле, были созданы микропроцессоры. Микропроцессор – это уже сложное ЦУ в микроэлектронном исполнении, обрабатывающее цифровые данные по алгоритмам, реализованным в программах.

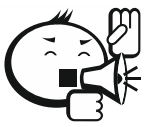
В курсе «Цифровые устройства и микропроцессоры» рассмотрим, из каких элементов состоят ЦУ, как из них строятся узлы, из которых затем создаются сложные структуры для управления объектами различного назначения. Курс основывается на знаниях, полученных из курсов по информатике, электронике, теории цепей.

В главах с первой по девятую данного пособия использовались материалы [1–4]. Для десятой и одиннадцатой глав особо рекомендую книгу [5] и дополнительно к ней [6].

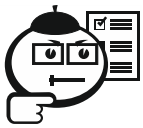
Соглашения, принятые в учебном пособии

Для улучшения восприятия материала в данном учебном пособии используются пиктограммы и специальное выделение важной информации.

Эта пиктограмма означает определение или новое понятие.



Эта пиктограмма означает «Внимание!». Здесь выделена важная информация, требующая акцента на ней. Автор может поделиться с читателем опытом, чтобы помочь избежать некоторых ошибок.



Эта пиктограмма означает задание. Здесь автор может дать указания для выполнения самостоятельной работы или упражнений, сослаться на дополнительные материалы.

Контрольные вопросы по главе

1 Двоичные данные и операции с ними

1.1 Электрические сигналы

Для передачи и переработки информации ее представляют в некоторой форме с использованием различных знаков. В общем случае под информацией понимают совокупность сведений о событиях, объектах или явлениях. Совокупность знаков, содержащих ту или иную информацию, называют сообщением. Сообщение может иметь самое различное содержание, но независимо от этого всегда отображается в виде сигнала.

В качестве сигнала можно использовать любой физический процесс, изменяющийся в соответствии с переносимым сообщением. Мы будем рассматривать только электрические сигналы, физической величиной которых является ток или напряжение. Сигналы формируются изменением (модуляцией) тех или иных параметров амплитуды, фазы, частоты. Сигнал – это средство перенесения информации в пространстве и времени. Для соответствия между сообщением и сигналом, т. е. для обеспечения возможности извлечения сообщения из полученного сигнала, последний следует формировать по определенным правилам. Каждому сообщению должен соответствовать свой сигнал.

Построение сигнала по определенным правилам называют ***кодированием.***

Если сигнал (или сообщение) может принимать любые значения в некотором интервале времени, его называют *непрерывным*, или *аналоговым*. Такой сигнал является непрерывной функцией от времени (на каком-то интервале), даже если сообщение к таковой не относится (рис. 1.1).

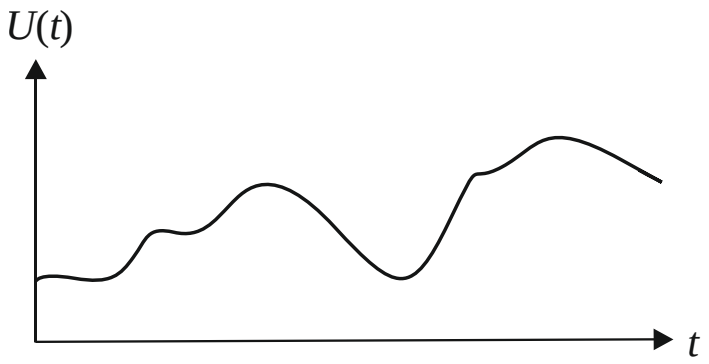


Рис. 1.1 – Аналоговый сигнал

Если сигнал (или сообщение) принимает только некоторые определенные значения из некоторого множества, то такой сигнал называют *дискретным.* На рисунке 1.2 показаны аналоговые сигналы, линейный и нелинейный (красной линией), и дискретные сигналы, соответствующие им (показаны в виде прямоугольников). Здесь на одинаковых промежутках времени (Δt), называемых дискретами времени, значение сигнала не изменяется. Увеличенный фрагмент (рис. 1.2, *в*) хорошо показывает погрешность в соответствиях аналогового и дискретного сигналов.

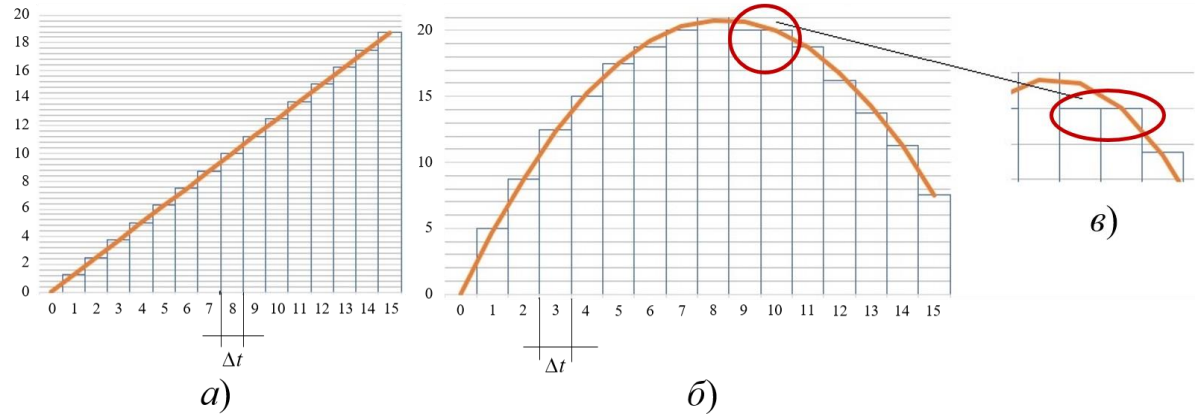


Рис. 1.2 – Дискретный сигнал:
линейно изменяющийся сигнал (*а*); нелинейный сигнал (*б*);
увеличенный фрагмент сигнала (*в*)

Каждая ступенька дискретного сигнала представляется в виде числа, поэтому дискретные сигналы, соответствующие аналоговым, представляют собой массивы числовых данных и могут обрабатываться различными численными методами.

Если дискретный сигнал принимает только два значения (*одно из них обычно обозначают 1, а другое 0*), *то его называют* ***логическим*** *или* ***цифровым сигналом*** (*рис. 1.3, а*). *Устройства, обрабатывающие такие сигналы, называют* ***цифровыми устройствами***.

Логические (*цифровые*) сигналы широко используются при построении устройств автоматики. При математическом описании сигналов подразумевается независимость в обозначении сигнала от истинного значения действующей амплитуды напряжения.

Так, для некоторых ЦУ цифровой сигнал может принимать значение высокого уровня, равное +15 В, для других – равное +5 В. И в том и в другом случае говорят, что сигнал принимает единичное значение, и обозначают его 1. Противоположный сигнал, по уровню близкий к нулевому значению, обозначают 0 и называют нулевым.

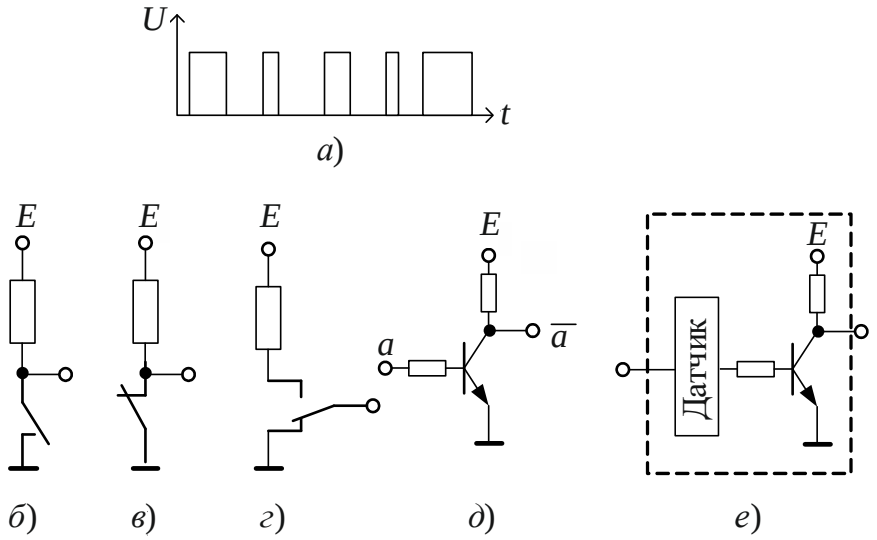


Рис. 1.3 – Цифровой сигнал и его формирование:
диаграмма цифрового сигнала (*а*); нормально разомкнутый контакт (*б*);
нормально замкнутый контакт (*в*); переключающий контакт (*г*);
электронный ключ (*д*); электронный датчик (*е*)

Как может быть сформирован цифровой сигнал, который в качестве входного поступает в ЦУ? Могут быть использованы контактные или электронные элементы. На рисунке 1.3, б показана схема с контактом, который в неактивном состоянии разомкнут. На выходном полюсе будет присутствовать высокий уровень, равный Е питания, т. е. логическая единица. Если на контакт механически нажать, то на выход будет подключена «земля» – логический ноль. При прекращении нажатия на контакт он самостоятельно возвращается в неактивное состояние.

На рисунке 1.3, *в* показана схема, формирующая логические сигналы с помощью нормально замкнутого контакта, который в неактивном состоянии подключает на выход 0, а при нажатии на него выдает 1. На рисунке 1.3, *г* показан переключающий контакт, который в любой момент времени соединяется с одним или другим контактом и фиксируется в таком положении, пока не переключить его.

На рисунке 1.3, *д* показана схема электронного ключа, собранного на транзисторе. При подаче низкого уровня напряжения (0) на его вход транзистор

закрывается и с выхода ключа снимается высокий уровень (1). При подаче на вход высокого уровня (1) транзистор переходит в режим насыщения, т. е. открывается, и с выхода снимается низкий уровень (0).

Во многих современных устройствах используются электронные датчики (рис. 1.3, *е*). В их состав входит сам датчик, преобразующий входную измеряемую величину в электрический сигнал, который поступает на вход транзисторного ключа, формирующий логический сигнал. Этот сигнал показывает, достигла ли измеряемая величина заданного значения или нет. Например, датчик давления, настроенный на какую-то величину давления пара в котле, может показать достижение давления пара какой-то критической величине формированием уровня 0 или 1 на выходе.

Все устройства (рис. 1.3, *а–е*) могут входить в состав разнообразных технических устройств. Они могут переключаться при воздействии механической силы, или воздействия магнитного поля, или других физических явлений.

ЦУ имеют дело с логическими или дискретными сигналами в числовой форме. Для обработки чисел в ЦУ необходимо их представить в понятной для ЦУ форме, т. е. с помощью цифр 0 и 1. Здесь требуется знание перехода от привычной десятичной формы представления числа к другой. Представление чисел в различной форме с использованием цифр определяется системой счисления, т. е. законом записи чисел.

1.2 Системы счисления

Под системой счисления будем понимать способ записи чисел с помощью цифровых знаков. В настоящее время используются позиционные системы счисления, в которых изменения положения цифры в записи числа ведет к изменению самого числа. Позиционной системой счисления (СС) называется такая, которая удовлетворяет следующему равенству:

$$A_{(q)} = a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + ... + a_1 \cdot q^1 + a_0 \cdot q^0 + a_{-1} \cdot q^{-1} + ... + a_{-m} \cdot q^{-m}, \qquad (1.1)$$

где

- $A_{(q)}$ – запись числа в q-ичной СС;
- q – основание СС (целое число, лежащее в диапазоне от 1 до N);
- a_k – значение k-го разряда в записи числа ($a = 0, 1, \ldots, q-1$);
- n – количество целых разрядов в записи числа;

- m – количество дробных разрядов в записи числа (конкретные значения чисел n и m будем называть номером разряда или разрядом числа);
- q^P – вес разряда $(P = n, n-1, \ldots, 1, 0, -1, -2, \ldots, -m)$.

Так, для десятичной (десятеричной) СС $q = 10$, $a_k = 0, 1, 2, \ldots 9$, поэтому числа записываются следующим образом:

$$243{,}98_{(10)} = 2 \cdot 10^2 + 4 \cdot 10^1 + 3 \cdot 10^0 + 9 \cdot 10^{-1} + 8 \cdot 10^{-2} = 200 + 40 + 3 + 0{,}9 + 0{,}08.$$

Изменяя q, можно получать запись одного и того же числа в разных СС. Существуют разные алгоритмы перевода чисел из одной СС в другую. Рассмотрим один из них (*рассмотрим только СС, которые используются в цифровых устройствах – двоичная, восьмеричная и шестнадцатеричная*).

Перевод числа в другую СС производится *отдельно для целой и дробной* части числа. Целая часть переводится делением, а дробная – умножением на основание новой СС.

Перевод целой части:

1) число, которое нужно перевести из одной СС в другую, делится нацело на основание новой СС (q) с записью остатка от деления (остаток не может быть больше $q - 1$);
2) полученное частное снова делится нацело на q;
3) пункт 2 выполняется до тех пор, пока полученное частное не окажется меньше основания новой СС;
4) для получения записи числа в новой СС к последнему частному приписываются справа в обратном порядке все остатки от деления, полученные при выполнении пунктов 1, 2, 3. Например, десятичное число $167{,}36_{(10)}$ так переводится в двоичную СС (рис. 1.4):

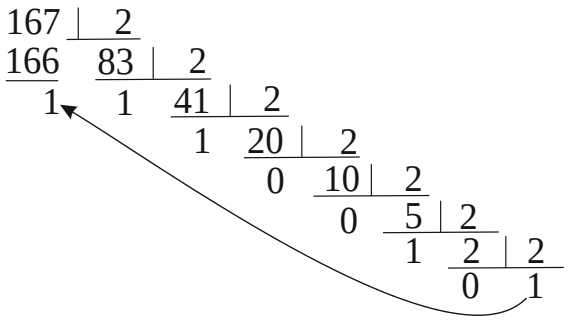


Рис. 1.4 – Перевод десятичного числа в двоичную СС

Результат перевода: $167{,}36_{(10)} = 10100111_{(2)}$.

Дробная часть числа (рис. 1.5):

1) дробная часть умножается на основание новой СС;
2) полученная целая часть отделяется от произведения;
3) пункты 1 и 2 выполняются до тех пор, пока полученное произведение не окажется равным нулю, или до достижения заранее оговоренного количества разрядов;
4) дробная часть в новой СС состоит из последовательности целых частей произведений в порядке их получения.

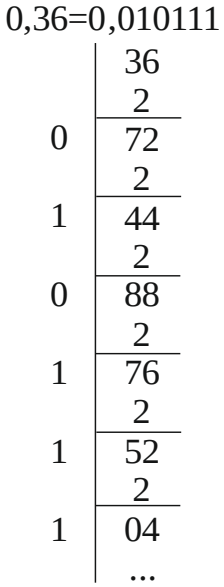


Рис. 1.5 – Перевод дробной части десятичного числа в двоичную СС

К целой части числа приписывается дробная часть, и тогда

$$167{,}36_{(10)} = 10100111{,}010111_{(2)}.$$

Как видно из полученного результата, целая часть переводится точно, а дробная часть отличается от дробной части исходного десятичного числа. Если обратить внимание на формулу 1.1, то очевидно, что дробная часть может приближаться к истинному значению только при увеличении количества разрядов (в пределе к бесконечному). Поэтому при переводах из одной СС в другую необходимо определить такое количество дробных разрядов, которое с достаточной степенью точности представляют исходные числа. Так, в современных компьютерах дробная часть представляется в виде 32 или 64 двоичных разрядов.

В ЦУ кроме двоичной СС применяются восьмеричная и шестнадцатеричная СС. Их основное назначение – это отображение состояния двоичных устройств более коротким числом разрядов.

Восьмеричная СС оперирует восемью цифрами: 0, 1, 2, 3, 4, 5, 6, 7. Пример перевода из десятичной СС в восьмеричную представлен на рисунке 1.6.

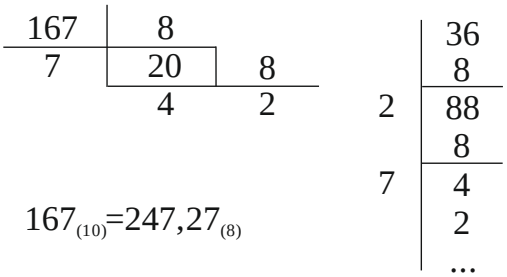


Рис. 1.6 – Перевод десятичного числа в восьмеричную СС

Шестнадцатеричная СС должна использовать 16 цифр, из которых десять – это десятичные цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а шесть цифр обозначаются символами латинского алфавита: A, B, C, D, E, F (a, b, c, d, e, f). При этом остатки от деления и последнее частное, полученные при переводе целой части, и результаты целой части, полученные при умножении дробной части, заменяются на буквенные символы следующим образом:

10 – A, 11 – B, 12 – C, 13 – D, 14 – E, 15 – F. Пример перевода числа 167,36 в шестнадцатеричную СС представлен на рисунке 1.7.

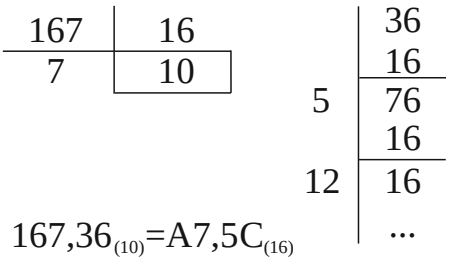


Рис. 1.7 – Перевод десятичного числа в шестнадцатеричную СС

Примеры соответствия чисел в различных СС приведены в таблице 1.1, из которой видно, что чем больше величина числа, тем меньше разрядов нужно для СС с большим основанием.

Таблица 1.1 – Примеры чисел в различных СС

Десятичная	Восьмеричная	Шестнадцатеричная	Двоичная
28	34	1C	11100
493	755	1ED	111101101
16983	41127	4257	100001001010111

При обратном переводе из двоичной, восьмеричной и шестнадцатеричной СС в десятичную можно использовать формулу 1.1:

$$100101{,}101_2 = 1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} =$$
$$= 16 + 2 + 0{,}5 + 0{,}125 = 18{,}625_{10}.$$

$$C9, F1_{16} = C\cdot 16^1 + 9\cdot 16^0 + F\cdot 16^{-1} + 1\cdot 16^{-2} = 12\cdot 16^1 + 9\cdot 16^0 + 1\cdot 16^{-1} + 1\cdot 16^{-2} =$$
$$= 192 + 9 + 0{,}9375 + 0{,}0039 = 201{,}94_{10}.$$

$$175{,}45_8 = 1\cdot 8^2 + 7\cdot 8^1 + 5\cdot 8^0 + 4\cdot 8^{-1} + 5\cdot 8^{-2} = 64 + 56 + 5 + 0{,}5 + 0{,}078 = 125{,}58_{10}.$$

Цифровые устройства, как было сказано ранее, используют двоичную СС. Один разряд двоичного числа, имеющий значение 0 или 1, носит название *бит* (*bit*). Это самая мелкая единица измерения информации. Цифровые и микропроцессорные устройства оперируют более крупными единицами. 8 бит образуют 1 *байт*. Байты объединяются в *слова*, которые, в зависимости от разрядности устройства, состоят из 2, 4 или 8 байт.

1.3 Представление чисел в разрядной сетке цифровых устройств

Цифровые устройства предназначены для обработки информации, записанной в двоичной СС, или, по-другому, в двоичном коде. Над числами выполняются арифметические и логические операции. Числа могут быть целыми и дробными, положительными и отрицательными.

Числа размещаются в разрядной сетке цифрового устройства и все операции выполняются поразрядно над разрядами с одним весом. Разряды нумеруются, начиная с младшего, который имеет номер ноль. Формат такого расположения представлен в таблице 1.2.

Таблица 1.2 – Разрядная сетка 8-разрядного цифрового устройства

7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	1

Числа, которыми оперирует цифровое устройство, могут быть беззнаковыми целыми (табл. 1.2), т. е. все разряды являются значащими. Диапазон чисел, размещаемый в такой разрядной сетке, изменяется

от 0 до 255 ($2^8 - 1$).

В таблице 1.2 записано число 145_{10}.

Если числа являются целыми со знаком, то знак размещается в старшем разряде, причем знак положительных чисел кодируется нулем, отрицательных – единицей. 1,0010001 – отрицательное число, 0,0010001 – положительное число. Между знаком и значащими разрядами ставится запятая, которая называется

знаковой. В таблицах 1.3 и 1.4 показано размещение чисел в разрядной сетке устройства. Знаковый разряд отделен двойной чертой.

Таблица 1.3 – Разрядная сетка цифрового устройства со знаковым разрядом (отрицательное число –17)

7		6	5	4	3	2	1	0
1		0	0	1	0	0	0	1

Таблица 1.4 – Разрядная сетка цифрового устройства со знаковым разрядом (положительное число +17)

7		6	5	4	3	2	1	0
0		0	0	1	0	0	0	1

Диапазон чисел в этом случае изменяется. Половина диапазона из 256 чисел отдается под положительные числа, половина – под отрицательные.

Если в этой разрядной сетке размещаются числа со знаком и с дробной частью, заранее оговаривается, сколько знаков отдается под целую, а сколько – под дробную часть (рис. 1.8). В таком представлении появляются две запятые: одна – знаковая, другая – десятичная. Число записывается так: 1,1001,101 – для отрицательного числа, 0,1011,011 – для положительного.

Рис. 1.8 – Разрядная сетка цифрового устройства с целой и дробной частью

Очевидно, что такое количество дробных разрядов очень невелико для представления чисел с дробной частью. Поэтому такое разделение практически не применяется. Чтобы получить более точное значение дробной части, создается двухбайтовое слово, в котором целая часть – это старший байт со знаковым разрядом. Младший байт содержит только дробную часть числа, и это целое беззнаковое число (рис. 1.9).

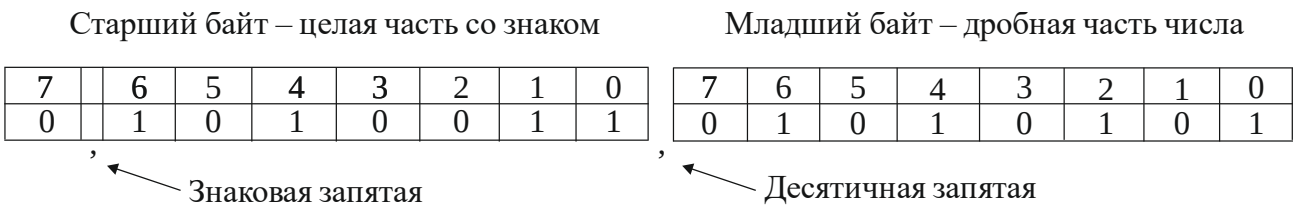


Рис. 1.9 – Двухбайтовое число с дробной частью

При необходимости создаются 4- или даже 8-байтовые слова, в которых большая часть отдается под дробную часть. В таких случаях все обрабатываемые числа представляются в одном формате, т. е. во всех числах десятичная запятая располагается в строго определенном месте между разрядами. Такой формат чисел носит название *числа с фиксированной запятой* (например, $0{,}1010011{,}01010101_2$). В современных ЭВМ все числа при вводе преобразуются так, чтобы они оказались меньше 1. В этом случае все разряды ЭВМ, кроме знакового, оказываются принадлежащими дробной части.

1.4 Арифметические операции в цифровых устройствах

В таблице 1.5 приведено соответствие десятичных чисел цифрам восьмеричной, шестнадцатеричной и двоичной СС.

Таблица 1.5 – Таблица соответствия кодов чисел

Десятичные числа	Двухразрядные числа	Трехразрядные двоичные числа	Четырехразрядные двоичные числа	Восьмеричные цифры	Шестнадцатеричные цифры
0	00	000	0000	0	0
1	01	001	0001	1	1
2	10	010	0010	2	2
3	11	011	0011	3	3
4		100	0100	4	4
5		101	0101	5	5
6		110	0110	6	6
7		111	0111	7	7
8			1000		8
9			1001		9
10			1010		A
11			1011		B
12			1100		C
13			1101		D

14			1110		E
15			1111		F

Выполнение арифметических операций в цифровых устройствах рассмотрим на примере пятиразрядного устройства со знаковым разрядом (рис. 1.10). Десятичная запятая перенесена правее младшего разряда, т. е. в такой разрядной сетке могут размещаться только целые числа.

		3	2	1	0

8	4	2	1

Веса разрядов

Рис. 1.10 – Формат разрядной сетки цифрового устройства

На рисунке 1.10, в верхней строке, приведены веса каждого из разрядов для быстрого перевода чисел из двоичного в десятичное счисление (можно использовать и таблицу 1.4).

Правила сложения двоичных кодов чисел:

- числа a и b подписываются одно под другим так, чтобы разряды с одним весом оказались одно под другим;
- сложение выполняется поразрядно по правилам, указанным в таблице 1.6;
- в операцию сложения вступают все разряды, включая знаковые;
- перенос, возникающий в i-м разряде, прибавляется к сумме $i + 1$-го разряда;
- перенос из старшего разряда теряется.

Таблица 1.6 – Правила сложения двоичных разрядов

a_i	b_i	Перенос p_i	Сумма i-разрядов
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

1.4.1 Прямой код двоичного числа

Если к двоичному коду числа добавить знаковый разряд, то получим прямой код (ПК) числа. Например:

$$0{,}1001\ (\text{ПК}) = 1\cdot 8 + 0\cdot 4 + 0\cdot 2 + 1\cdot 1 = +10;$$
$$0{,}0011\ (\text{ПК}) = 0\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1\cdot 1 = +3;$$
$$1{,}1010\ (\text{ПК}) = 1\cdot 8 + 0\cdot 4 + 1 + 2 + 0\cdot 1 = -10;$$
$$1{,}1110\ (\text{ПК}) = 1\cdot 8 + 1\cdot 4 + 1\cdot 2 + 0\cdot 1 = -14.$$

На рисунке 1.11 приведены примеры сложения положительных и отрицательных чисел, представленных в прямом коде.

Все числа в ПК

a	+10	0	,	1	0	1	0
b	+3	0	,	0	0	1	1
a+*b*	+13	0	,	1	1	0	1

+5	0	,	0	1	0	1
+6	0	,	0	1	1	0
+11	0	,	1	0	1	1

+11	0	,	1	0	1	1
+3	0	,	0	0	1	1
+14	0	,	1	1	1	0

a	+5	0	,	0	1	0	1
b	−2	1	,	0	0	1	0
a+*b*	−7	1	,	0	1	1	1

−7	1	,	0	1	1	1
−6	1	,	0	1	1	0
+13	0	,	1	1	0	1

−1	1	,	0	0	0	1
−9	1	,	1	0	0	1
+10	0	,	1	0	1	0

Рис. 1.11 – Примеры сложения двоичных чисел в ПК

Как видно, при сложении отрицательных чисел получены неверные результаты. Отсюда следует:

В прямом коде можно выполнять сложение только положительных чисел.

1.4.2 Дополнительный код двоичного числа

Для правильного выполнения операции сложения с участием отрицательных чисел разработаны дополнительный и обратный код. В современных микропроцессорных устройствах используется, как правило, дополнительный код (ДК). ДК образуется по следующим правилам:

1) для *положительных* чисел ДК и ПК совпадают;
2) все разряды отрицательного числа (*за исключением знакового*) инвертируются, и к младшему разряду прибавляется единица (*при инвертировании 0 в разряде заменяется на 1, а 1 заменяется на 0*);
3) единица переноса из знакового разряда теряется;

4) сумма, полученная в результате сложения чисел, представленных в ДК, тоже является ДК;
5) если при сложении чисел получается отрицательное число, то чтобы узнать его *истинное* значение, его нужно перевести в ПК, причем для такого перевода следует применить правило из пункта 2.

На рисунке 1.12 приведены примеры перевода отрицательных чисел из ПК в ДК.

−1 1 , 0 0 0 1 ПК 1 , 1 1 1 0 Инвертирование значащих разрядов +1 Прибавление 1 1 , 1 1 1 1 ДК	−7 1 , 0 1 1 1 ПК 1 , 1 0 0 0 Инвертирование значащих разрядов +1 Прибавление 1 1 , 1 0 0 1 ДК
−6 1 , 0 1 1 0 ПК 1 , 1 0 0 1 Инвертирование значащих разрядов +1 Прибавление 1 1 , 1 0 1 0 ДК	−2 1 , 0 0 1 0 ПК 1 , 1 1 0 1 Инвертирование значащих разрядов +1 Прибавление 1 1 , 1 1 1 0 ДК
−9 1 , 1 0 0 1 ПК 1 , 0 1 1 0 Инвертирование значащих разрядов +1 Прибавление 1 1 , 0 1 1 1 ДК	−14 1 , 1 1 1 0 ПК 1 , 0 0 0 1 Инвертирование значащих разрядов +1 Прибавление 1 1 , 0 0 1 0 ДК

Рис. 1.12 – Примеры перевода отрицательных чисел в ДК

На рисунке 1.13 приведены примеры сложения чисел с использованием ДК.

a	+5	0	,	0	1	0	1	
b	−2	1	,	1	1	1	0	ДК
a+*b*	+3	0		0	0	1	1	

Результат положительный ДК совпадает с ПК, перевод не нужен

−7	1	,	1	0	0	1	ДК
−6	1	,	1	0	1	0	ДК
	1	,	0	0	1	1	ДК

Перевод из ДК в ПК

	1	,	1	1	0	0	
						1	
−13	1	,	1	1	0	1	ПК

−1	1	,	1	1	1	1	ДК
−9	1	,	0	1	1	1	ДК
	1	,	0	1	1	0	

Перевод из ДК в ПК

	1	,	1	0	0	1	
						1	
−10	1	,	1	0	1	0	ПК

Рис. 1.13 – Примеры сложения чисел в ДК

1.4.3 Переполнение разрядной сетки

В разрядной сетке цифрового устройства (рис. 1.10) могут быть представлены положительные числа от нуля до $+15_{10}$(0,0000 до 0,1111) и отрицательные от -15_{10} до -1_{10}(1,0001 до 1,1111). Если при сложении чисел результат

превышает +15 или –15, то говорят, что происходит арифметическое переполнение (АП) разрядной сетки цифрового устройства.

Рассмотрим несколько примеров (рис. 1.14).

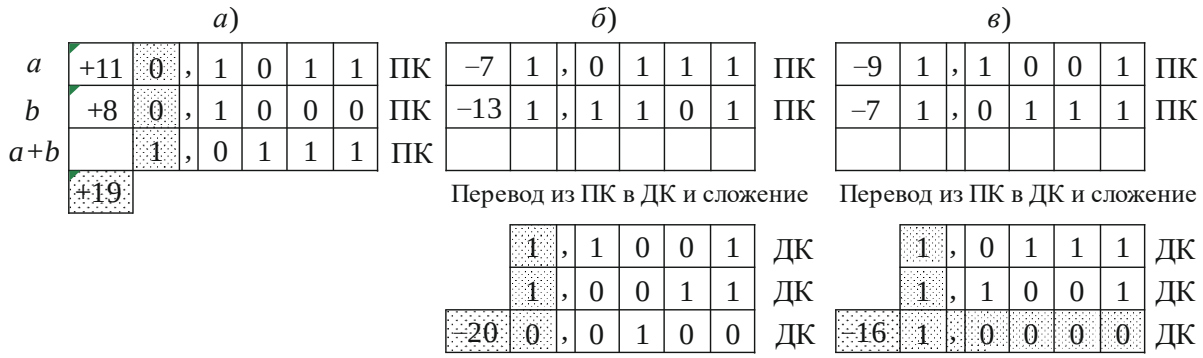
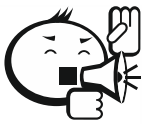


Рис. 1.14 – Арифметическое переполнение разрядной сетки:
переполнение при положительных числах (*а*);
переполнение при отрицательных числах (*б*, *в*)



Правила:

1. Если при сложении *положительных* чисел в знаковом разряде результата *вместо нуля образовалась единица*, то произошло АП. В примере (рис. 1.14, *а*) должна получиться сумма +19, что превышает максимальное число +15, которое может разместиться в разрядной сетке.
2. Если при сложении *отрицательных* чисел в знаковом разряде результата *вместо единицы образовался ноль*, то произошло АП. В примере (рис. 1.14, *б*) должна получиться сумма –20, что превышает максимальное число –15, которое может разместиться в разрядной сетке.
3. В примере (рис. 1.17, *в*) слагаемые имеют отрицательный знак и знак результата тоже отрицателен. Но сумма, которая должна получиться, равна –16, что также превышает возможности разрядной сетки (–15).

Признаком АП в этом случае является то, что *знаки слагаемых и результата отрицательны и во всех значащих разрядах суммы получились нули*.

1.4.4 Арифметические операции вычитания, умножения, сложения

Операция арифметического вычитания A – B может быть заменена на операцию сложения A – B = A + (–B), хотя и существуют алгоритмы вычитания. Вычитание *i*-х разрядов чисел подчиняются правилу (рис. 1.15), по кото-

рому при вычитании единицы из нуля производится заем из старшего $i + 1$-го разряда.

Вычитание A – B

Заем из a_{i+1}	a_i	b_i	Разница i-х разрядов
0	0	0	0
1	0	1	1
0	1	0	1
0	1	1	0

Рис. 1.15 – Правило вычитания двоичных чисел

Пример вычитания двух чисел без знака с целой и дробной частью представлен на рисунке 1.16.

A	1	0	1	0	1	1	1	0	1	,	1	1	= 349,8
B	0	1	0	0	1	0	1	1	1	,	0	1	= 151,3
A – B	0	1	1	0	0	0	1	1	0	,	1	0	= 198,5

Рис. 1.16 – Пример вычитания двоичных чисел

Операция умножения выполняется поразрядным умножением каждого разряда множителя на все разряды множимого, сохранением этого промежуточного результата и сложения всех промежуточных результатов (рис. 1.17).

Множимое A					1	0	1	1	0	1	1		A = 91
Множитель B					0	0	1	1	0	0	1		B = 25
Промежуточные произведения каждого разряда множителя на множимое					1	0	1	1	0	1	1		
				0	0	0	0	0	0	0			
			0	0	0	0	0	0	0				
		1	0	1	1	0	1	1					
	1	0	1	1	0	1	1						
0	0	0	0	0	0	0							
0	0	0	0	0	0	0							Произведение A·B
Сумма всех промежуточных произведений													
0	1	0	0	0	1	1	1	0	0	0	1	1	2275

Рис. 1.17 – Пример умножения двоичных чисел

При перемножении двух n-разрядных чисел получается $2n$-разрядное число. Алгоритмы деления и умножения относятся в микропроцессорной технике к длинным операциям, выполняемым в течение нескольких машинных циклов. Они реализуются относительно сложными алгоритмами, требуют определенных аппаратных ресурсов и реализуются, как правило, программным способом.

1.5 Двоично-десятичные коды

Существует еще одна система кодирования чисел, называемая двоично-десятичной. Ее еще называют кодированием 8-4-2-1. В ней каждая десятичная цифра $0, 1, \ldots 9$ заменяется четырехразрядной *двоично-десятичной декадой* (ДДД). Достоинство такого кодирования заключается в его простоте и точном переводе из десятичной системы счисления в ДДД и обратно. Например:

$$1972{,}68 = 0001\ 1001\ 0111\ 0010,\ 0110\ 1000.$$

Этот способ кодирования удобен тем, что в одном байте помещается две цифры ДДД. Недостатком являются избыточность такого кода, т. к. он требует большего числа двоичных разрядов, и сложность арифметических операций. Дело в том, что при сложении двух ДДД в некоторых разрядах полученная сумма может превысить цифру 9. Поэтому ко всем декадам больше 9 прибавляется корректирующий код 0110. Если и после этого образуются декады больше 9, то они опять корректируются. Это может продолжаться несколько раз в зависимости от длины слагаемых. Поэтому сложение двоично-десятичных чисел в среднем длится дольше, чем при сложении обычных двоичных чисел. Пример сложения представлен на рисунке 1.18.

769,58 =	0111	0110	1001	,	0101	1000
154,12 =	0001	0101	0100	,	0001	0010
923,70 =	1000	1011	1101	,	0110	1010
		Больше 9	Больше 9			Больше 9
	1000	1011	1101	,	0110	1010
Корректирующий код		0110	0110			0110
Сумма	1001	0010	0011	,	0111	0000
	9	2	3	,	7	0

Рис. 1.18 – Пример сложения двоично-десятичных чисел

Сначала числа складываются, а затем выполняется коррекция (рис. 1.18). При сложении длинных чисел может потребоваться применение операции корректирования несколько раз. В результате выполнение операции сложения чисел в ДДК требует больше времени. Кроме того, усложняется аппаратная часть устройства для сложения чисел. Поэтому такое кодирование используется гораздо реже, чем кодирование в двоичной СС.

Контрольные вопросы по главе 1

1. Что нужно для того, чтобы как можно точнее заменить аналоговый сигнал дискретным сигналом?
2. Зависит ли точность перевода десятичных чисел в двоичные от разрядности устройства, хранящего двоичное число?
3. В какой СС запись десятичного числа 64899524567_{10} будет самой короткой?
4. Что нужно сделать, чтобы узнать истинное значение отрицательного числа, представленного в дополнительном коде?
5. Знак одного слагаемого равен 0, второго – –0. Знак результата сложения этих чисел равен 1. Как трактовать полученный результат?
6. Складываются два числа в дополнительном коде. Одно число положительное, второе – отрицательное. В каком случае при этом может возникнуть АП?
7. Перемножаются два шестнадцатиразрядных числа. Какова разрядность результата?
8. Всегда ли потребуется операция коррекции при сложении чисел в ДДК?
9. Если год вашего рождения перевести из десятичной СС в двоичную, то сколько потребуется двоичных разрядов? (Ответить надо без перевода числа).
10. Устройство имеет Q состояний. Каждому состоянию присвоен уникальный двоичный код. Сколько двоичных разрядов потребуется для кодирования?

2 Логические основы цифровых устройств

Математическая модель *цифрового устройства* ***S*** *представляет собой шестикомпонентный вектор* $S=(A,\ Z,\ W,\ \delta,\ \lambda,\ a_1)$, *где*

- $A=\{a_1,\ldots a_m,\ldots a_M\}$ – множество внутренних состояний устройства;
- $a_1 \subseteq A$ – начальное состояние устройства;
- $Z=\{z_1,\ldots z_f,\ldots z_F\}$ – множество входных сигналов;
- $W=\{w_1,\ldots w_g,\ldots w_G\}$ – множество выходных сигналов;
- $\delta:\ A\times Z \to A$ – функция переходов, реализующая отображение произведения множеств $A\times Z$ на A;
- $\lambda:\ A\times Z \to W$ – функция выходов, реализующая отображение произведения множеств $A\times Z$ на W.

Частным случаем являются устройства, для которых множество A является пустым $(A=\varnothing)$. Тогда они описываются только множеством входных сигналов Z, выходных сигналов W и функцией выходов

$$\lambda: Z \to W,$$

которая устанавливает соответствие между элементами множества Z входных сигналов и множества W выходных сигналов. Такие устройства относятся к устройствам первого рода и носят название *комбинационные схемы* (КС). Для них характерно то, что выходной сигнал однозначно определяется только входным сигналом.

2.1 Основы булевой алгебры

В XIX в. английский математик Джордж Буль разработал основные положения алгебры логики, которую называют булевой алгеброй. Только в XX в. соответствующий уровень развития промышленности привел к тому, что этот раздел математики оказался востребованным.

Это произошло в связи с внедрением устройств автоматики, построенных на реле. Реле – это электромеханическое устройство, которое имеет в своем составе контактные группы, замыкающие или размыкающие электрические цепи. Тогда замыкание цепи приводит к тому, что по ней течет электрический ток, а в противном случае ток не течет.

С развитием физических наук и технологий реле были заменены на электронные устройства, которые на своих выходах могли формировать уровни либо высокого, либо низкого напряжения.

В обоих случаях одна из ситуаций могла обозначаться единицей, а другая – нулем.

Булева алгебра оперирует высказываниями, каждое из которых может быть либо истинным – *true* (единица), либо ложным – *false* (ноль) и вводит операции, аксиомы, теоремы для их обработки и преобразований.

Алгебру Буля удобно использовать для описания законов работы цифровых дискретных устройств.

Булевой переменной x *называется переменная, которая может принимать только одно из двух значений: 0 или 1.*

$$\begin{cases} x = 1, \text{ если } x \neq 0, \\ x = 0, \text{ если } x \neq 1. \end{cases}$$

Никаких других значений, кроме 0 или 1, переменная x принимать не может. В дальнейшем для обозначения переменных будем применять латинские буквы с индексами (x_1, x_k, x_5 и т. п.) или просто буквы (a, b, c, d, F, Y).

Булеву переменную по-другому называют *двоичной,* или *логической,* переменной. Операции над этими переменными называют *логическими*.

Определены следующие операции над булевыми переменными:

1. *Логическое произведение,* или *конъюнкция* двух переменных, обозначается:

$$X_1 \& X_2, \qquad X_1 \wedge X_2, \qquad X_1 \cdot X_2, \qquad X_1 X_2$$

и подчиняется следующим правилам (табл. 2.1).

Таблица 2.1 – Таблица истинности операции логического произведения

X_1	X_2	$X_1 \& X_2$
0	0	0
0	1	0
1	0	0
1	1	1

2. *Логическая сумма*, или *дизъюнкция* двух переменных, обозначается

$$a \wedge b \quad \text{или} \quad a + b$$

и подчиняется следующим правилам (табл. 2.2).

Таблица 2.2 – Таблица истинности операции логического сложения

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

3. *Логическое отрицание,* или *инверсия* переменной, обозначается горизонтальной чертой над переменной $\bar{A}$ (в некоторых пакетах прикладных программ инверсия переменной обозначается знаком апострофа или другим знаком около переменной), например:

$$\bar{a} = a' \qquad \bar{b} = |b = \neg b \qquad \bar{d} = \not{d}$$

и подчиняется следующим правилам (табл. 2.3):

Таблица 2.3 – Таблица истинности операции логического отрицания

A	$\bar{A}$
0	1
1	0

Булевы переменные подчиняются следующим аксиомам:

$$0 \& 0 = 0, \quad 0 \& 1 = 0, \quad 1 \& 1 = 1, \quad 0 + 0 = 0, \quad 1 + 1 = 1, \quad 0 + 1 = 1,$$

причем они справедливы и для произвольного числа переменных:

$$0 \cdot 0 \cdot 0 \cdot \ldots \cdot 0 = 0, \quad 1 + 1 + 1 + \ldots + 1 = 1.$$

Для операции инверсии справедливы следующие отношения:

$$\bar{\bar{a}} = a, \qquad \bar{\bar{\bar{b}}} = \bar{b}, \qquad \bar{0} = 1, \qquad \bar{1} = 0,$$

то есть четное число инверсий дает саму переменную, а нечетное – инверсную. В булевой алгебре действуют следующие основные законы:

- *переместительный* для дизъюнкции: $a+b=b+a$;
- *переместительный* для конъюнкции: $ab=ba$;
- *сочетательный* для дизъюнкции: $a+(b+c)=(a+b)+c$;
- *сочетательный* для конъюнкции: $a(bc)=(ab)c$;
- *распределительный* для дизъюнкции: $(a+b)(c+d)=ac+ad+bc+bd$, $a+bc=(a+b)(a+c)$;
- *распределительный* для конъюнкции: $a(b+c)=ab+ac$.

Из аксиом и законов алгебры логики следует ряд важных теорем, свойств и правил, которые полезны при выполнении эквивалентных преобразований:

$$x+x+x+\ldots+x\ =\ x,\ \ x\,x\,x\ldots x\ =\ x,$$

$$a+1=1,\ \ abc+1=1,$$

$$1+x+ab+\bar{a}b\bar{d}+cef\ =1,$$

$$a\ \&\ 0=0,$$

$$0\ \&\ abc\bar{d}\,\overline{aef\bar{d}}=0,\ \ 0\cdot a\cdot b\cdot\bar{c}\cdot(\bar{c}d+abe)=0,$$

$$a+\bar{a}=1,\quad abc+\overline{abc}=1,\ a+a=a,\ \ ab\bar{c}+ab\bar{c}=ab\bar{c},$$

$$a\cdot\bar{a}=0,\ \ abc\cdot\overline{abc}=0,\ a\cdot a\cdot a=a,\ abc\cdot abcd=abcd.$$

Закон склеивания (склеивание происходит по одной переменной, которая в одном случае имеет прямое, а в другом – инверсное значение):

$$ab+a\bar{b}=a,\ \ a\bar{b}cd+abcd=acd,$$

$$(a+b)(a+\bar{b})=a,$$

$$(a+\bar{b}+\bar{c}+\bar{d})(a+\bar{b}+c+\bar{d})=a+\bar{b}+\bar{d}).$$

Закон поглощения:

$$a+ab=a,\ \ a\bar{b}cd+ac=ac,$$

$$a(a+b)=a,\quad ab(ab+a\bar{b}c+abcd+\bar{a}\bar{b}\bar{c})=ab.$$

Теорема де Моргана:

$$\overline{a+b}=\bar{a}\bar{b},\quad \overline{ab}=\bar{a}+\bar{b}.$$

Примеры применения теоремы де Моргана:

$$\overline{a\bar{b}cd}=\bar{a}+bc+\bar{d},\ \ \overline{ab+b\bar{d}c}=\overline{ab}\cdot\overline{b\bar{d}c}=(\bar{a}+\bar{b})(\bar{b}+dc).$$

2.2 Булевы функции. Способы их задания

***Булевой функцией* (*БФ*)**, *или* ***переключательной функцией* (*ПФ*)**, *или* ***функцией алгебры логики* (*ФАЛ*)**, *от* **n** *переменных называется функция* $f(x_1, x_2, x_3,\ldots,x_n)$, *которая на любом наборе своих аргументов может принимать одно из двух значений: 0 или 1.*

Под набором аргументов понимается совокупность значений переменных, каждая из которых может быть равна 0 или 1. Для функции от ***n*** аргументов количество возможных наборов равно 2^n. На них может быть задано $(2^n)^n$ всевозможных БФ.

Способов задания БФ несколько. Одним из самых простых и наглядных является задание таблицей истинности.

Булевы функции f_1 и f_2 от переменных *a*, *b*, *c* заданы таблицей 2.4. Таблицу следует понимать так, что в любой момент времени переменные *abc* могут принимать *любой из наборов*, указанных в строках левой части таблицы. Например, *abc* = 001, или *abc* = 110, или *abc* = 011, и т. д. В правой части таблицы указаны те значения функции, которые она должна принимать на данном наборе аргументов. То есть на наборе *abc* = 001, $f_1 = 0$, $f_2 = 0$, при *abc* = 111 $f_1 = 1$, $f_2 = 0$, а на наборе *abc* = 100 $f_1 = 1$, $f_2 = 1$ и т. д.

Таблица 2.4 – Таблица истинности БФ f_1 и f_2

a	***b***	***c***	f_1	f_2
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

2.3 Совершенные формы БФ

Наборы логических переменных x_1, x_2 … x_n, на которых БФ принимает единичные значения, называют единичными наборами. Наборы логических переменных, на которых БФ принимает нулевые значения $f(x_1, x_2,\ldots, x_n) = 0$, называют нулевыми наборами.

Конъюнкцией называется логическое произведение произвольного числа аргументов. Конъюнкция называется *элементарной*, если она содержит любое количество попарно различимых букв в прямой или инверсной форме. Например, конъюнкции

$$abc, \quad \bar{a}\bar{b}c$$

являются элементарными, а вот конъюнкции

$$a\overline{bc}, \quad a(b+\bar{c})$$

не являются элементарными.

Назовем *рангом R* элементарной конъюнкции количество входящих в нее букв. Две конъюнкции назовем *соседними*, если они зависят от одних и тех же аргументов, имеют одинаковый ранг и отличаются знаком инверсии только одного аргумента, например

$$abc\bar{d} \text{ и } abcd, \ \bar{a}\bar{b}c\bar{d} \text{ и } \bar{a}\bar{b}cd.$$

Длиной функции *L* назовем сумму рангов всех входящих в функцию конъюнкций (т. е. количество букв в формуле).

Дизъюнктивной нормальной формой (ДНФ) БФ назовем дизъюнкцию (логическую сумму) конечного множества попарно различимых элементарных конъюнкций:

$$f_{\text{ДНФ}} = abc + ac\bar{d} + ab\bar{c}d + \bar{a} \qquad (L = 11).$$

Конституентой единицы назовем элементарную конъюнкцию, содержащую *все аргументы*, от которых зависит БФ.

Совершенной дизъюнктивной нормальной формой (СДНФ) БФ назовем ДНФ, каждый член которой является конституентой единицы:

$$f = abcd + ab\bar{c}\bar{d} + abc\bar{d} + \bar{a}bcd + a\bar{b}c\bar{d} + \bar{a}\bar{b}\bar{c}\bar{d} \qquad (L = 24).$$

Дизъюнкцией называется логическая сумма произвольного числа аргументов. Дизъюнкция называется *элементарной*, если она содержит любое количество попарно различимых букв в прямой или инверсной форме. Например, дизъюнкции

$$a+b+c+d, \quad \bar{a}+\bar{b}+d, \quad a+c$$

являются элементарными, а вот дизъюнкции

$$a+bc, \quad ab+cd, \quad a+\overline{c+d}$$

элементарными не являются.

Назовем *рангом R* элементарной дизъюнкции количество входящих в нее букв. Две дизъюнкции назовем *соседними*, если они зависят от одних и тех же

аргументов, имеют одинаковый ранг и отличаются знаком инверсии только одного аргумента. Например:

$$(a+b+c+d) \text{ и } (a+b+\bar{c}+d).$$

Длиной функции L назовем сумму рангов всех входящих в функцию дизъюнкций, т. е. *количество букв в формуле*.

Конъюнктивной нормальной формой (КНФ) БФ назовем конъюнкцию (логическое произведение) конечного множества попарно различимых элементарных дизъюнкций:

$$f_{\text{КНФ}} = (a+\bar{b}+c)(a+\bar{b}+\bar{c}+d)(\bar{b}+\bar{c}+\bar{d}) \qquad (L=10).$$

Конституентой нуля назовем элементарную дизъюнкцию, содержащую *все аргументы*, от которых зависит БФ.

Совершенной конъюнктивной нормальной формой (СКНФ) БФ назовем КНФ, каждый член которой является конституентой нуля:

$$f_{\text{СКНФ}} = (a+b+c)(a+\bar{b}+c)(\text{a}+\bar{b}+\bar{c}) \qquad (L=9).$$

2.4 Переход от табличного способа задания БФ к аналитическому

Пусть БФ f_1 и f_2 заданы таблицей 2.5.

Таблица 2.5 – Таблица истинности функций f_1, f_2

a	***b***	***c***	f_1	f_2
0	0	0	1	0
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

При переходе от табличного способа задания БФ к аналитическому в форме ДНФ или КНФ можно получить только *совершенные* формы (СДНФ и СКНФ). Для получения СДНФ БФ записывается как логическая сумма конституент единицы. Для этого используется следующий алгоритм:

1) поочередно просматриваются все строки таблицы, отмеченные единичным значением функции;
2) конституента единицы записывается как логическое произведение переменных, причем, если переменная в наборе имеет значение 1, то она

записывается в прямом виде, а если 0, то в инверсном, например, СДНФ f_1 будет иметь вид:

$$f_{1\text{СДНФ}} = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c + ab\bar{c} + abc \quad L = 15.$$

БФ f_2 из таблицы 2.5 представим в КНФ. Из таблицы можно получить только СКНФ. БФ записывается как логическое произведение конституент нуля. Для получения аналитического выражения БФ:

1) поочередно просматриваются все строки таблицы, отмеченные нулевым значением функции;
2) конституента нуля записывается как дизъюнкция переменных, причем, если переменная в наборе имеет значение 0, то она записывается в прямом виде, а если 1, то в инверсном, например, f_2 будет иметь вид:

$$f_{2\text{СКНФ}} = (a+b+c)(a+b+\bar{c})(\text{a}+\bar{b}+\bar{c})(\bar{a}+\bar{b}+c)+(\bar{a}+\bar{b}+\bar{c}) \quad L = 15.$$

Следует уяснить, что СДНФ и СКНФ – это просто разные *формы БФ*, и одна и та же БФ может быть записана в любой из них.

Например, пусть БФ *Z* задана таблицей 2.6.

Таблица 2.6 – Таблица истинности функции *Z*

a	***b***	***Z***
0	0	0
0	1	1
1	0	1
1	1	0

Запишем эту функцию в обеих формах:

$$Z_{\text{СДНФ}} = \bar{a}b + a\bar{b}, \quad Z_{\text{СКНФ}} = (a+b)(\bar{a}+\bar{b}).$$

Если в БФ, записанной в форме СКНФ, раскрыть скобки и применить закон поглощения, то получится следующее:

$$Z = (a+b)(\bar{a}+\bar{b}) = a\bar{a} + a\bar{b} + b\bar{a} + b\bar{b} = a\bar{b} + \bar{a}b,$$

т. е. совпадает с СДНФ.

Если функция задана в виде формулы, то может быть восстановлена таблица истинности. Например, задана функция P:

$$P = a\bar{b} + ab\bar{c}.$$

В формуле используются три переменные, значит, БФ зависит от трех аргументов. Вычисляем значения функции для всех наборов переменных от *abc* = 000 до *abc* = 111:

$$P(0,0,0) = 0 \cdot \overline{0} + 0 \cdot 0 \cdot \overline{0} = 0 \cdot 1 + 0 \cdot 0 \cdot 1 = 0 + 0 = 0;$$

$$P(0,0,1) = 0 \cdot \overline{0} + 0 \cdot 0 \cdot \overline{1} = 0 + 0 = 0;$$

$$P(0,1,0) = 0 \cdot \overline{1} + 0 \cdot 1 \cdot \overline{0} = 0 + 0 = 0;$$

$$P(0,1,1) = 0 \cdot \overline{1} + 0 \cdot 1 \cdot \overline{1} = 0 + 0 = 0;$$

$$P(1,0,0) = 1 \cdot \overline{0} + 1 \cdot 0 \cdot \overline{0} = 1 + 0 = 1;$$

$$P(1,0,1) = 1 \cdot \overline{0} + 1 \cdot 0 \cdot \overline{1} = 1 + 0 = 1;$$

$$P(1,1,0) = 1 \cdot \overline{1} + 1 \cdot 1 \cdot \overline{0} = 0 + 1 = 1;$$

$$P(1,1,1) = 1 \cdot \overline{1} + 1 \cdot 1 \cdot \overline{1} = 0 + 0 = 0.$$

Вычисленные значения функции заносятся в таблицу (табл. 2.7).

Таблица 2.7 – Таблица истинности функции *Z*

a	***b***	***c***	***P***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

По таблице 2.7 можно записать СДНФ: $P = a\overline{b}\overline{c} + a\overline{b}c + ab\overline{c}$.

Для функции *R*, заданной в конъюнктивной форме:

$$R = (a + b)(\overline{a} + \overline{b})$$

вычислены значения функции, которые занесены в таблицу 2.8:

$$R(0,0) = (0 + 0)(\overline{0} + \overline{0}) = 0(1 + 1) = 0;$$

$$R(0,1) = (0 + 1)(\overline{0} + \overline{1}) = 1 \cdot 1 = 1;$$

$$R(1,0) = (1 + 0)(\overline{1} + \overline{0}) = 1 \cdot 1 = 1;$$

$$R(1,1) = (1 + 1)(\overline{1} + \overline{1}) = 1 \cdot 0 = 0.$$

Таблица 2.8 – Таблица истинности функции *R*

a	***b***	***R***
0	0	0
0	1	1
1	0	1
1	1	0

2.5 Числовой способ задания БФ

Существует еще один способ задания БФ. Он основан на следующем: если каждой логической переменной поставить в соответствие *разряд* двоичного числа, то каждому *набору* булевых переменных будет соответствовать *двоичное* число, а ему десятичное.

Тогда БФ можно задать перечислением наборов, на которых она принимает только значения, равные единице (для ДНФ), или только значения, равные нулю (для КНФ).

Так БФ f_1 и f_2 из таблицы 2.5 в числовой форме могут быть записаны (*в записи формул используются знаки* Σ *– для ДНФ-сумм* и Π *– произведение для КНФ*):

$$f_{1\text{СДНФ}} = \sum(0,\ 3,\ 5,\ 6,\ 7), \quad f_{2\text{СКНФ}} = \prod(0,\ 1,\ 3,\ 6,\ 7).$$

От числового способа задания можно легко перейти к табличному способу задания и к записи БФ в виде формулы.

2.6 Применение законов склеивания для минимизации БФ

Две БФ называются ***эквивалентными****, если они принимают одинаковые значения на одних и тех же наборах аргументов.*

Одна и та же БФ может быть записана разными формулами. Из нескольких формул предпочтительней та, которая имеет наименьшую длину. Для получения более короткой формулы применяются эквивалентные преобразования с использованием правил склеивания и поглощения. Процедура получения минимальной формулы БФ называется *минимизацией*. Алгоритм минимизации БФ, заданной в форме СДНФ, следующий:

1) склеиванию подлежат только соседние конституенты единицы. В результате склеивания образуется конъюнкция, называемая импликантой. Импликанта, полученная склеиванием двух конъюнкций, ранг которых равен R, имеет ранг $R - 1$;
2) одна и та же конституента единицы может принимать участие в нескольких операциях склеивания;
3) к полученным импликантам опять применяется операция склеивания (до тех пор, пока это возможно);
4) БФ записывается как дизъюнкция полученных импликант.

Исходя из этого для БФ f_1 из таблицы 2.5 (в скобках обозначен номер конъюнкции):

$$f_{1\text{СДНФ}} = \bar{a}\bar{b}\bar{c}(1) + \bar{a}bc(2) + a\bar{b}c(3) + ab\bar{c}(4) + abc(5) \quad (L = 15),$$

- первая конъюнкция не склеивается ни с одной другой;
- вторая склеивается с пятой: $\bar{a}bc + abc = bc$;
- третья – с пятой: $a\bar{b}c + abc = ac$;
- четвертая – с пятой: $ab\bar{c} + abc = ab$.

Полученные конъюнкции больше не могу быть склеены. БФ f_1 запишется как

$$f_1 = \bar{a}\bar{b}\bar{c} + bc + ac + ab \quad (L = 9),$$

т. е. очевидно, что полученная ДНФ короче, чем исходная СДНФ.

Алгоритм минимизации БФ, заданной в форме СКНФ:

1) склеиванию подлежат только соседние конституенты нуля. В результате склеивания образуется дизъюнкция, которая имеет ранг $R - 1$;
2) одна и та же дизъюнкция может принимать участие в нескольких операциях склеивания;
3) к полученным дизъюнкциям опять применяется операция склеивания (до тех пор, пока это возможно);
4) БФ записывается как произведение полученных дизъюнкций.

Минимизация БФ f_2 из таблицы 2.5:

$$f_{2\text{СКНФ}} = (a + b + c)(a + b + \bar{c})(a + \bar{b} + \bar{c})(\bar{a} + \bar{b} + c)(\bar{a} + \bar{b} + \bar{c})$$

1 2 3 4 5

- первая дизъюнкция склеивается со второй: $(a+b+c)(a+b+\bar{c}) = a+b$;
- вторая склеивается с третьей: $(a+b+\bar{c})(a+\bar{b}+\bar{c}) = a+\bar{c}$;
- четвертая – с пятой: $(\bar{a}+\bar{b}+c)(\bar{a}+\bar{b}+\bar{c}) = \bar{a}+\bar{b}$.

К полученным дизъюнкциям операция склеивания уже не применима, поэтому БФ f_2:

$$f_2 = (a+b)(a+\bar{c})(\bar{a}+\bar{b}) \quad (L = 6),$$

т. е. очевидно, что полученная КНФ короче, чем исходная СКНФ. Для f_2 операции склеивания можно выполнить и по-другому:

- первую – со второй, четвертую – с пятой, как и в первом случае;
- а вот третью можно склеить не со второй, а с пятой;

- полученные дизъюнкции не могу быть склеены, поэтому можно записать:

$$f_2 = (a+b)(a+\bar{c})(\bar{b}+\bar{c}) \quad (L=6).$$

Оба варианта функции f_2 имеют одинаковую длину $L = 6$ поэтому они равноценны.

Применение законов поглощения также позволяет минимизировать БФ, заданные не только в совершенной форме. Пусть БФ задана в форме ДНФ:

$$f = ab + a\bar{b} + a\bar{c}\bar{e}g + acd\bar{e} + \bar{a}\bar{c}\bar{d} = a + a\bar{b} + a\bar{c}\bar{e}g + acd\bar{e} + \bar{a}\bar{c}\bar{d} = a + \bar{a}\bar{c}\bar{d}.$$

В данном преобразовании вначале применен закон склеивания, а затем закон поглощения.

Для функции P в форме КНФ:

$$P = (b+\bar{c})(b+c)(b+\bar{a}d)(b+\bar{c}\bar{d})(\bar{a}+\bar{b}\bar{d}) =$$
$$= b(b+\bar{a}d)(b+\bar{c}\bar{d})(\bar{a}+\bar{b}\bar{d}) = b(\bar{a}+\bar{b}\bar{d}) = \bar{a}b.$$

Контрольные вопросы по главе 2

1. Чем отличается конституента единицы от конституенты нуля?
2. В таблице истинности БФ восемь строк. В столбце значений функции четыре единицы и четыре нуля. Каковы длины СДНФ и СКНФ этой БФ?
3. Чему равен ранг конституенты единицы БФ от четырех переменных?
4. Чему равен ранг конституенты нуля БФ от трех переменных?
5. Задана БФ от переменных *a, b, c, d*. Конституента единицы зависит только от переменных *b, c, d*. Может ли такое быть?
6. Какую операцию – поглощения или склеивания – нужно применить для уменьшения длины соседних дизъюнкций?
7. При склеивании двух конституент нуля длина у полученной дизъюнкции оказалась раной четырем. От скольких переменных зависит функция?
8. БФ задана числовым способом. Самое большое число в записи номеров наборов равно 23. Как определить количество переменных, от которых зависит БФ?
9. Какая операция используется для минимизации СДНФ?

10. БФ в форме КНФ от четырех переменных задана числовым способом. В скобках записано шесть чисел. Сколько строк, отмеченных единицами, будет в таблице истинности этой БФ?

3 Синтез комбинационных схем

3.1 Построение комбинационных схем на электронных элементах

С момента появления полупроводниковых приборов и развития технологии микроэлектроники для реализации цифровых устройств широко используются интегральные электронные элементы. Существует большое количество серий (наборов) интегральных микросхем. Каждая серия содержит широкую номенклатуру различных электронных логических элементов, с помощью которых строятся различные цифровые схемы. На рисунке 3.1 представлены обозначения логических элементов на схемах, выполненные в соответствии с ЕСКД (рядом с каждым элементом тонкими линиями приведены обозначения тех же элементов в стандарте, принятом в некоторых странах, в частности США).

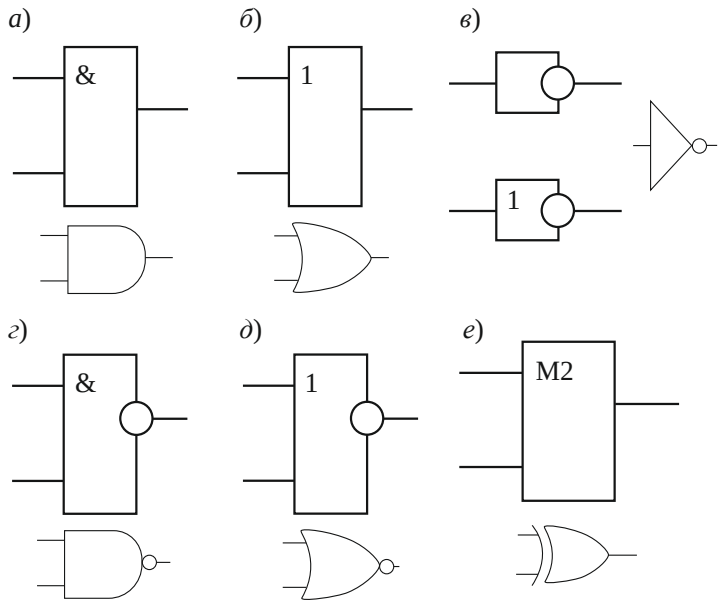
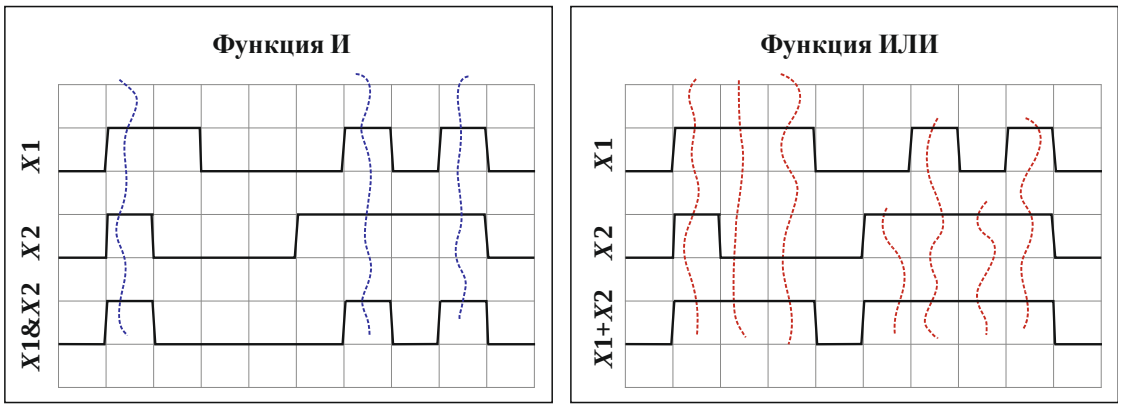


Рис. 3.1 – Обозначения электронных логических элементов: элемент И (*а*); элемент ИЛИ (*б*); элемент НЕ (*в*); элемент И-НЕ (*г*); элемент ИЛИ-НЕ (*д*); элемент «исключающее ИЛИ» (*е*)

В обозначении элементов ставятся значки, показывающие выполняемые ими функции: И – «&», ИЛИ – «1» (*для элемента ИЛИ допускается обозначение* «≥1»), сумма по модулю 2 (*другое ее название – исключающее ИЛИ*) – «M2». Элемент НЕ можно изображать двумя способами.

Инверсный выход элементов НЕ, И-НЕ и ИЛИ-НЕ обозначается кружком. При создании схем ЦУ у любого логического элемента, кроме НЕ, добавляется столько входов, сколько требуется для реализации логической операции, т. е. 2, 3, 4 и т. д. С помощью этих логических элементов *реализуются* любые БФ.

В отличие от формул булевых функций и выражений схемы имеют на входах и выходах электрические сигналы, соответствующие булевым переменным. На рисунке 3.2 приведены диаграммы (осциллограммы) зависимости выходных сигналов элемента И и элемента ИЛИ в зависимости от входных сигналов.



а) *б*)

Рис. 3.2 – Зависимости выходного сигнала от входных:
для элемента И (*а*); для элемента ИЛИ (*б*)

На вход элемента И поступают электрические сигналы, соответствующие переменным X_1 и X_2. Только когда оба сигнала $\left(X_1 \textbf{ И } X_2 = X_1 \cdot X_2\right)$ примут единичное значение, на выходе также образуется единичный сигнал. Эти моменты отмечены синими пунктирными линиями.

Если на любом из входов элемента ИЛИ $\left(X_1 \textbf{ ИЛИ } X_2 = X_1 + X_2\right)$ поступит единичный сигнал (отмечено красной пунктирной линией), то на выходе формируется единица.

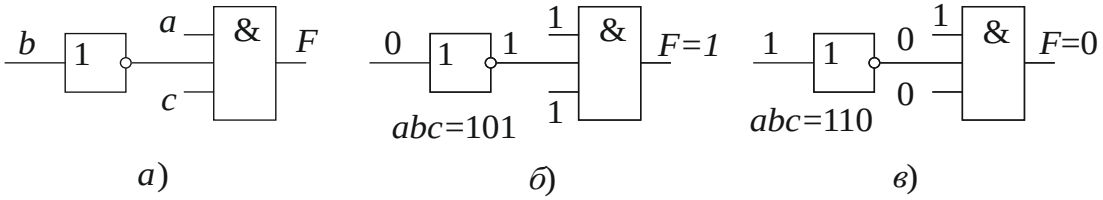
Под *синтезом КС* будем понимать последовательность действий, состоящую из математического описания (задания БФ), вывода формулы, минимизации и построения схемы на логических элементах, т. е. *схемную реализацию БФ*.

Для схемной реализации БФ используются логические элементы И, ИЛИ, НЕ.

Рассмотрим пример реализации простейших функций. Пусть необходимо реализовать БФ, заданные формулами:

$$F = a\bar{b}c, \quad Q = \bar{a} + b + \bar{c}.$$

Функция F представляет собой конъюнкцию, т. е. $F = 1$ только на одном-единственном наборе 101 $(1 \cdot \bar{0} \cdot 1 = 1 \cdot 1 \cdot 1 = 1)$, а на всех остальных наборах она равна 0. Чтобы получить логическое произведение переменных, необходимо использовать элемент И, имеющий три входа. На два входа подаются прямые значения сигналов *a* и *b*, а на третий вход сигнал *c* следует проинвертировать (рис. 3.3, *а*). Для этого в схему включается инвертор, реализующий функцию НЕ.



а) *б*) *в*)

Рис. 3.3 – Реализация функции *F*:
схема (*а*); сигналы на элементе И при входе *abc* = 101 (*б*);
выход при *abc* = 110 (*в*)

На рисунке 3.3, *б* показано значение входных и выходного сигнала в том случае, если переменные примут значения *abc* = 101. Любая другая комбинация сигналов на входах приведет к тому, что один из сомножителей обратится в 0, а значит, и произведение станет равным нулю (на наборе переменных *abc* = 110 функция *F* принимает нулевое значение).

Функция $Q = \bar{a} + b + \bar{c}$ задана в виде дизъюнкции. Она реализуется на элементе ИЛИ с тремя входами (рис. 3.4, *а*). Функция принимает значение $Q = 0$ только на одном наборе, а именно $a = 1,\ b = 0,\ c = 1,$ т. к. только в этом случае выполняется условие $Q = \bar{1} + 0 + \bar{1} = 0 + 0 + 0 = 0$ (рис. 3.4, *б*). При любой другой комбинации сигналов на входе одно из слагаемых становится равным 1, и вся логическая сумма становится равной 1.

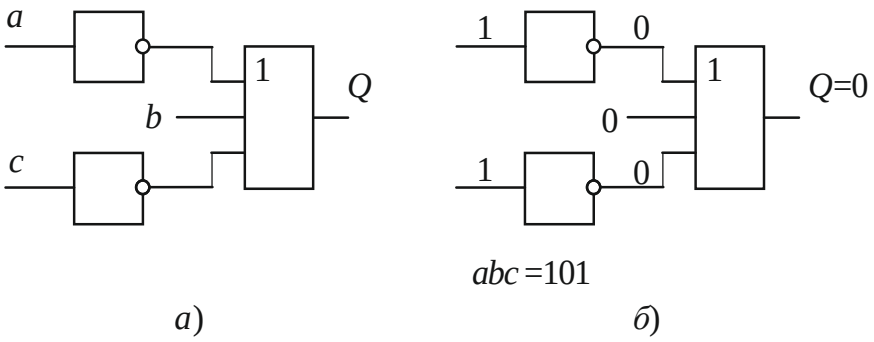


Рис. 3.4 – Реализация функции Q:
схема (*а*); сигналы на элементе ИЛИ при входе, abc = 101 (*б*)

Рассмотрим примеры реализации более сложных БФ. В таблице 3.1 заданы БФ f_1 и f_2. Реализуем на логических элементах f_1 в форме СДНФ (рис. 3.5), f_2 – в форме СКНФ (рис. 3.6).

Таблица 3.1 – Таблица истинности функций f_1 и f_2

Номер набора	$X_1X_2X_3$	f_1	f_2
0	000	1	1
1	001	0	1
2	010	0	0
3	011	1	0
4	100	1	1
5	101	0	1
6	110	1	0
7	111	0	0

$$f_1 = \overline{X_1}\,\overline{X_2}\,\overline{X_3} + \overline{X_1}X_2X_3 + X_1\overline{X_2}\,\overline{X_3} + X_1X_2\overline{X_3} \quad (L=12).$$

$$f_2 = (X_1 + \overline{X_2} + X_3)(X_1 + \overline{X_2} + \overline{X_3})(\overline{X_1} + \overline{X_2} + X_3)(\overline{X_1} + \overline{X_2} + \overline{X_3}) \quad (L=12).$$

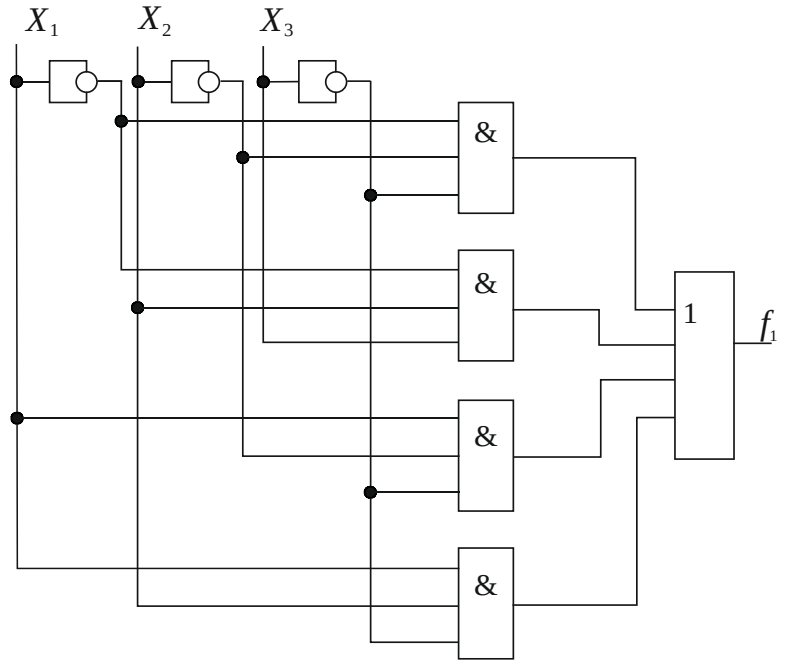


Рис. 3.5 – Реализация функции f_1, заданной в форме СДНФ

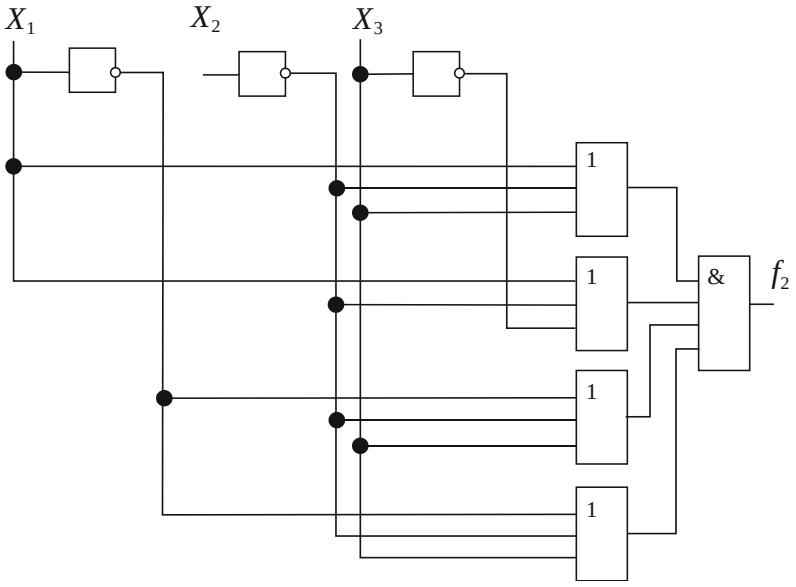


Рис. 3.6 – Реализация функций f_2 в форме СКНФ

Важной характеристикой электронной схемы является количество оборудования, необходимое для реализации БФ. Введем понятие цены схемы. Под ценой схемы C будем понимать количество оборудования, т. е. количество логических элементов, входящих в данную схему. Цена схемы определяется как сумма цен всех элементов, причем *цена* одного элемента – это *количество* его *входов*. Такая оценка, конечно, является условной. *При подсчете цены схемы не учитываются элементы НЕ, обеспечивающие инвертирование входных переменных.* Цены для схем (рис. 3.5, 3.6) равняются $C_1 = 16$, $C_2 = 16$ соответственно.

Если f_1 минимизировать, применив операцию склеивания, то в результате получится БФ с меньшей длиной:

$$f_1 = \overline{X_1} X_2 X_3 + \overline{X_2}\,\overline{X_3} + X_1 \overline{X_3} \quad (L = 7).$$

Схема, реализующая f_1 после минимизации, представлена на рисунке 3.7. Цена этой схемы C = 10.

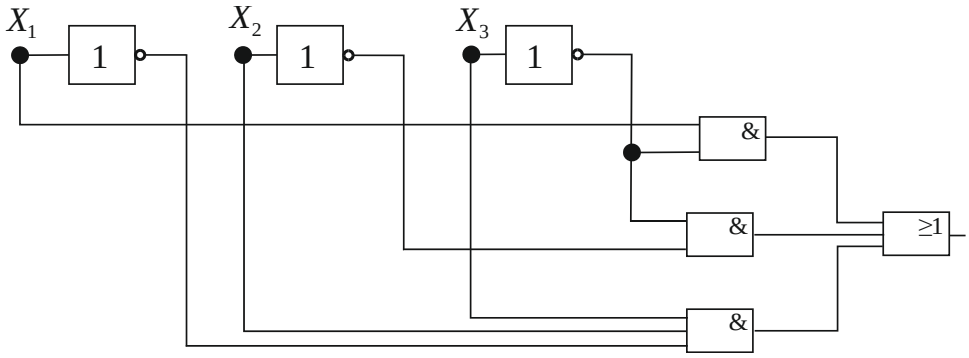


Рис. 3.7 – Реализация функций f_1 после минимизации

Отсюда следует вывод, что чем короче формула БФ, тем экономичнее будет схема, реализующая БФ, с точки зрения количества электронных компонентов. То есть сначала БФ необходимо минимизировать и только потом строить схему.

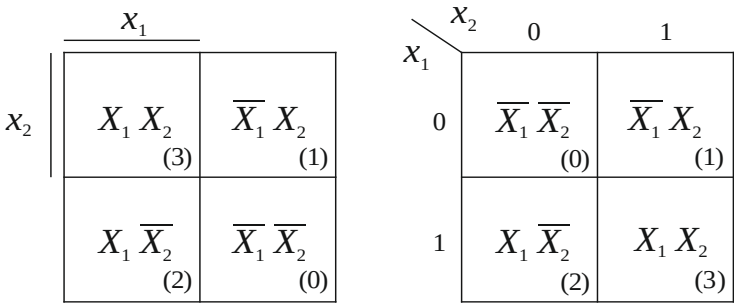
3.2 Минимизация БФ с помощью карт Карно – Вейча

Минимизация БФ с использованием закона склеивания часто представляет собой довольно громоздкую процедуру, особенно при увеличении числа аргументов. Это связано с тем, что приходится искать различные комбинации для склеивания, выбирая из них наиболее короткий. Поэтому были разработаны другие методы минимизации. Они, как правило, также громоздки, хотя и дают отличные результаты. Практически все они хорошо алгоритмируются и реализуются в виде компьютерных программ. Рассмотрим более простой метод минимизации, который можно условно назвать графическим. Этот метод был предложен независимо друг от друга М. Карно и Э. В. Вейчем и называется по их именам. В нем все наборы переменных БФ располагаются в таблицах, называемых картами Карно и картами Вейча. Их удобно использовать для минимизации БФ от 2, 3, 4 и реже от 5 и 6 переменных.

Карты Карно – Вейча для БФ от n переменных представляют собой таблицы (*n-мерное пространство, развернутое на плоскости*), содержащие 2^n клеток, каждая из которых соответствует одному-единственному набору аргумен-

тов БФ. Карты Карно отличаются от карт Вейча только разметкой, т. е. тем, как идентифицируется каждая клетка карты.

Разметка карты Вейча для БФ от двух аргументов (рис. 3.8, *а*) осуществляется булевыми переменными, отмечающими столбцы и строки карты.



а) *б*)

Рис. 3.8 – Карты от двух переменных: карта Вейча (*а*); карта Карно (*б*)

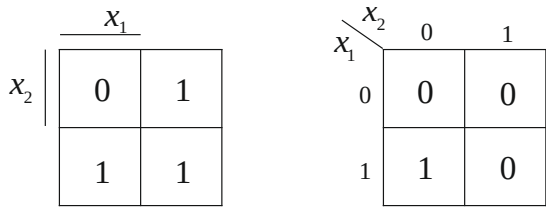
Так, переменная X_1 *покрывает* обе клетки левого столбца, а переменная $\overline{X_1}$ (ее не принято показывать на карте, чтобы не загромождать ее) покрывает обе клетки правого столбца. Переменная X_2 покрывает две клетки верхней строки таблицы, а переменная $\overline{X_2}$ покрывает две клетки нижней строки таблицы. Таким образом, каждая клетка карты имеет имя, составленное из набора переменных, покрывающих эту клетку. На карте Вейча (рис. 3.8, *а*) и карте Карно (рис. 3.8, *б*) показаны имена каждой клетки, а в скобках приведены номера десятичных кодов, соответствующих наборам аргументов, покрывающих эту клетку.

Отличие карт Карно в том, что на ней переменная, отмечающая строку или столбец карты, обозначается 0, если она принимает инверсное значение $\overline{X_1}$, или 1, если переменная принимает прямое значение X_1.

Если рассмотреть любые две соседние клетки карт, то очевидно, что конъюнкции, записанные в них, являются соседними (не только физически, но и логически) и к ним возможно применение операции склеивания. На картах операция склеивания проводится геометрически – склеиваемые клетки карты обводятся контуром.

Нанесение БФ на карту заключается в расстановке значений БФ в соответствующие клетки карты. Если БФ задана ДНФ, то в соответствующие клетки карты заносятся 1, а если КНФ, то 0.

Пусть заданы две БФ: $Z_1 = \sum(0,\ 1,\ 2)$ и $Z_2 = P(0,\ 1,\ 3)$. Результат нанесения на карту Вейча (Z_1) и карту Карно (Z_2) представлен на рисунке 3.9.



а) *б*)

Рис. 3.9 – Функции Z_1 и Z_2, нанесенные на карту Вейча (*а*) и карту Карно (*б*)

Минимизация с помощью карт Карно – Вейча осуществляется в соответствие со следующим алгоритмом:

1) соседние клетки карты, отмеченные 1 для СДНФ или 0 для СКНФ, покрываются контурами, имеющими форму квадрата или прямоугольника;
2) контур может покрывать 2^k клеток, где k = 0, 1, 2 … n (n – количество переменных, от которых зависит БФ), т. е. 1, 2, 4, 8 и т. д. клеток;
3) *все клетки,* отмеченные 1 для СДНФ или 0 для СКНФ, должны быть покрыты контурами;
4) одна и та же клетка карты может входить в несколько контуров;
5) количество контуров должно быть минимально, а их размер максимальным;
6) каждому контуру присваивается свое имя.

БФ записывается как логическая сумма (СДНФ) или логическое произведение (СКНФ) имен контуров.

Имена контуров для ДНФ составляются по следующему алгоритму:

1) все переменные для данного контура просматриваются поочередно;
2) если для всех клеток контура переменная принимает прямое значение, то в имени контура на ее месте записывается 1;
3) если для всех клеток контура переменная принимает инверсное значение, то в имени контура на ее месте записывается 0;
4) если для одних клеток контура переменная принимает прямое значение 1, а для других инверсное 0, то переменная является *независимой*, и в имени контура на ее месте записывается крест X;
5) конъюнкция, соответствующая имени контура, записывается как произведение переменных, причем, если в имени контура на месте пере-

менной стоит 1, то переменная записывается в прямом виде, а если на месте переменной стоит 0, то в конъюнкции на ее месте записывается инверсное значение переменной, независимая переменная, отмеченная X, из конъюнкции исключается.

На рисунке 3.10 показаны обозначения имен клеток и всех возможных контуров покрытия. Так, имена контуров записываются независимо от формы задания булевой функции, т. е. и для СДНФ, и для СКНФ имена контуров одинаковы.

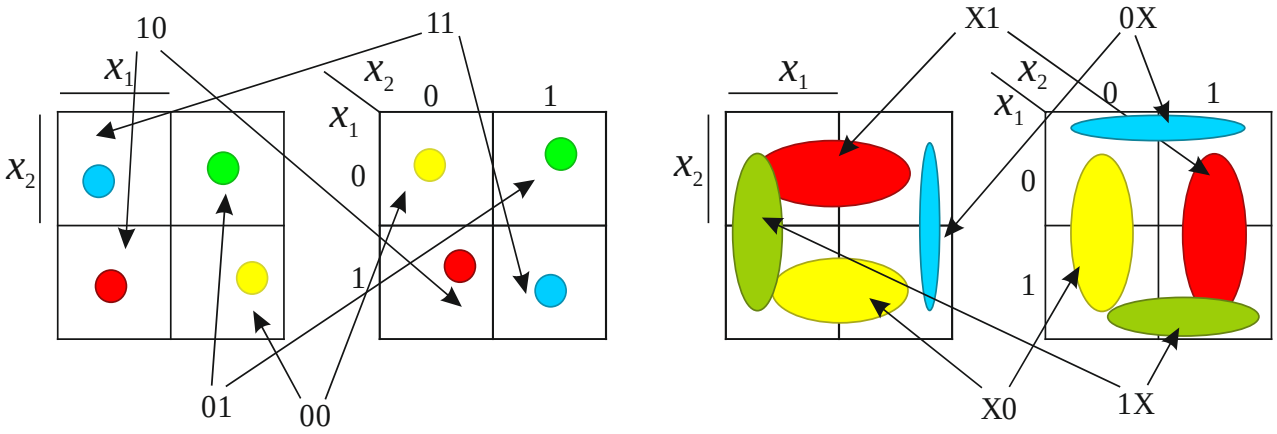


Рис. 3.10 – Имена каждой клетки и двухклеточных контуров

Для КНФ алгоритм получения имен контуров следующий:

1) все переменные для данного контура просматриваются поочередно;
2) если для всех клеток контура переменная принимает прямое значение, то в имени контура на ее месте записывается 1;
3) если для всех клеток контура переменная принимает инверсное значение, то в имени контура на ее месте записывается 0;
4) если для одних клеток контура переменная принимает прямое значение, а для других инверсное, то переменная является независимой и в имени контура на ее месте записывается крест X;
5) дизъюнкция, соответствующая имени контура, записывается как сумма переменных, причем, если в имени контура на месте переменной стоит 1, то переменная записывается в инверсном виде, а если на месте переменной стоит 0, то на ее месте записывается прямое значение переменной. Переменная, отмеченная как X, из записи дизъюнкции исключается.

Пример

- Для БФ, заданной в СДНФ, имена контуров в формуле функции запишутся следующим образом:

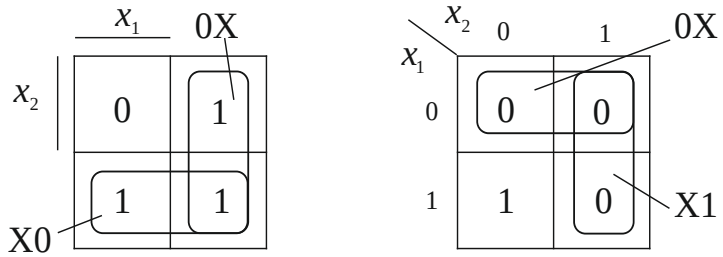
$$1X1X = ac,\quad 1001 = a\bar{b}\bar{c}d,\quad 10X0 = a\bar{b}\bar{d},\quad X01 = \bar{b}c,\quad 101 = a\bar{b}c;$$

- для БФ, заданной в СКНФ, имена контуров в формуле функции запишутся следующим образом:

$$1X1X = \bar{a} + \bar{c},\quad 1001 = \bar{a} + b + c + \bar{d},\quad 10X0 = \bar{a} + b + d,$$
$$X01 = b + \bar{c},\quad 101 = \bar{a} + b + \bar{c}.$$

. . . . . .

Применение алгоритма построения контуров показано на рисунке 3.11.



а) *б*)

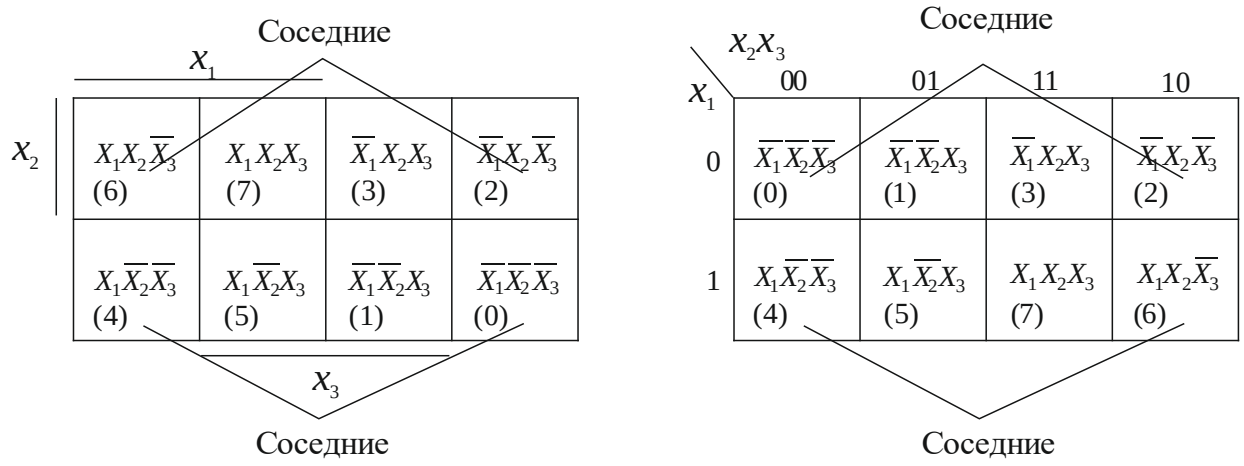
Рис. 3.11 – Покрытие контурами функций: Z_1 (*а*); Z_2 (*б*)

Функции Z_1 и Z_2 после минимизации имеют вид:

$$Z_1 = \overline{X_1}\overline{X_2},\qquad Z_2 = X_1\overline{X_2}.$$

Карты для БФ трех аргументов имеют по восемь клеток (2^3). Они имеют вид, показанный на рисунке 3.12. Внутри каждой клетки представлены наборы переменных, покрывающих эту клетку, и десятичные коды, соответствующие числовому эквиваленту набора аргументов.

Особенностью этих карт является то, что их можно сворачивать, как цилиндр, по широкой стороне (4 и более клеток). Это основано на том, что клетки левого и правого столбцов карт отмечены соседними конъюнкциями (рис. 3.12, *а*, *б*).



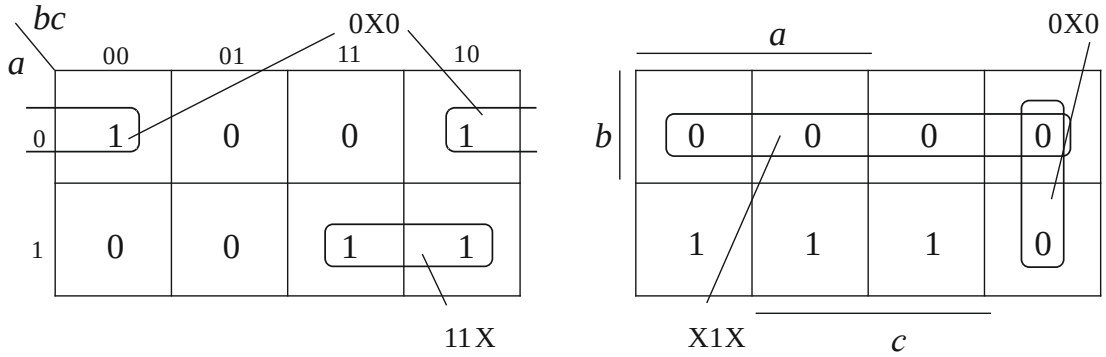
а) *б*)

Рис. 3.12 – Карты для трех переменных: карта Вейча (*а*); карта Карно (*б*)

Рассмотрим пример двух функций от трех переменных:

$$Y_1 = \sum(0,\ 2,\ 6,\ 7) \quad \text{и} \quad Y_2 = P(0,\ 2,\ 3,\ 6,\ 7).$$

Первая задана ДНФ и нанесена на карту Карно, а вторая задана КНФ и нанесена на карту Вейча (рис. 3.13).



а) *б*)

Рис. 3.13 – Минимизация БФ: Y_1 (*а*); Y_2 (*б*)

БФ Y_1 покрывается двумя контурами по две клетки в каждом, Y_2 – двумя контурами из четырех и двух клеток. Если записать эти БФ в СДНФ и СКНФ соответственно, то получатся следующие формулы:

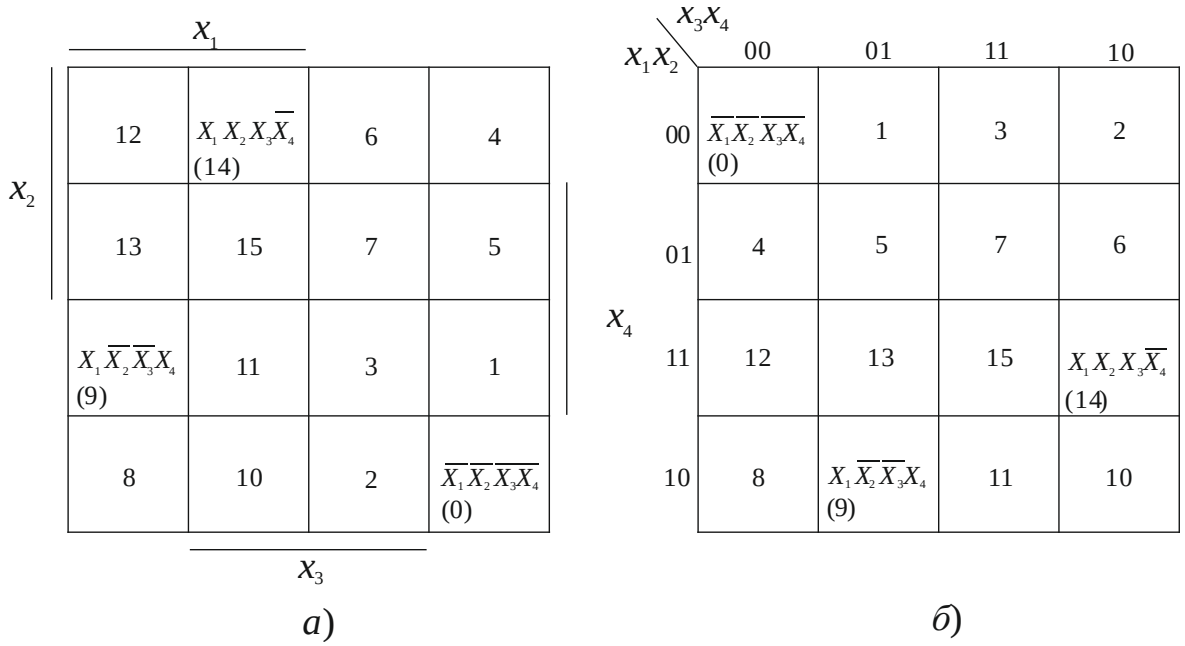
$$Y_1 = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + ab\bar{c} + abc,$$

$$Y_2 = (a+b+c)(a+\bar{b}+c)(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+c)(\bar{a}+\bar{b}+\bar{c}).$$

После минимизации формулы этих БФ принимают вид:

$$Y_1 = \bar{a}\,\bar{c} + ab \qquad Y_2 = \bar{b}(a+c)$$

Карты для БФ от четырех аргументов содержат $2^4 = 16$ клеток (рис. 3.14). Имена клеток карт нанесены в виде десятичных чисел, соответствующих наборам переменных.



а) *б*)

Рис. 3.14 – Карты для четырех переменных: карта Вейча (*а*); карта Карно (*б*)

Карты от четырех переменных можно сворачивать как по вертикали, так и по горизонтали. Причем это можно делать одновременно (*это фигура из четырехмерного пространства, развернутая в двумерной плоскости*), поэтому четыре угловые клетки карт образуют один контур из четырех клеток. Минимизируем булевы функции $S_1 = \sum(0,\ 2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10)$ с помощью карты Карно (рис. 3.15, *а*) и $S_2 = P(0,\ 1,\ 3,\ 4,\ 5,\ 7,\ 8,\ 9,\ 10,\ 12,\ 13)$ с помощью карты Вейча (рис. 3.15, *б*).

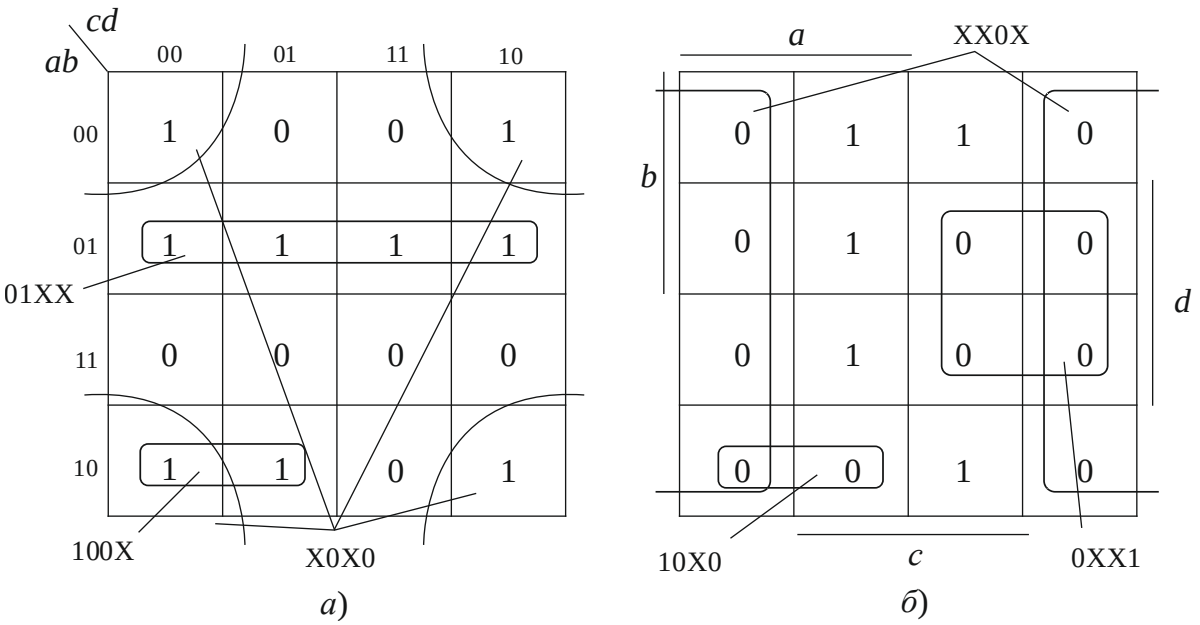


Рис. 3.15 – Покрытие функций S_1 и S_2 контурами: S_1 (*а*); S_2 (*б*)

$$S_1 = \bar{a}b + a\bar{b}\bar{c} + \bar{b}\,\bar{d},$$

$$S_2 = c(\bar{a} + b + d)(a + \bar{d}).$$

На рисунке 3.16 приведена карта Карно для пяти переменных.

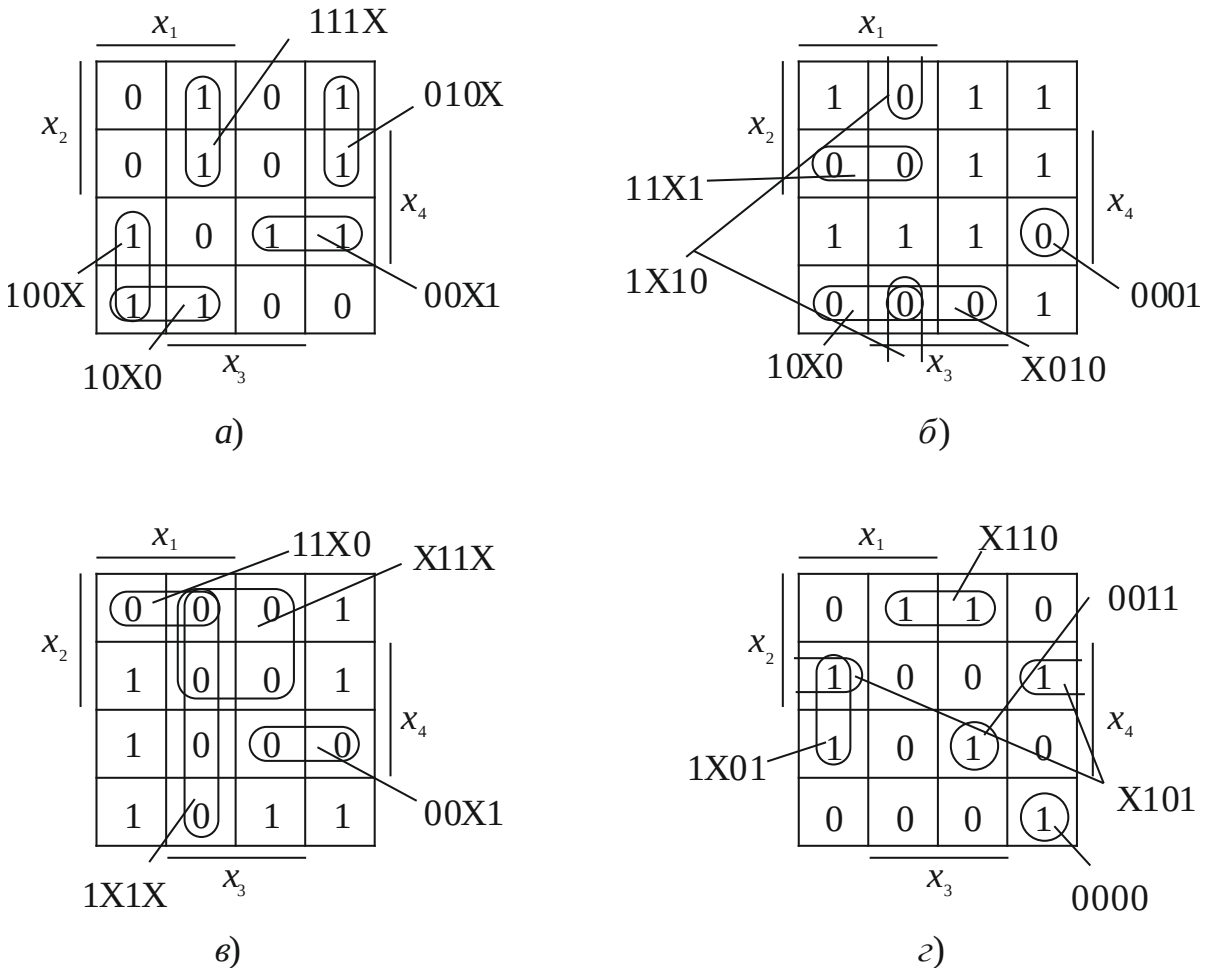
x_1x_2 \ $x_3x_4x_5$	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

Рис. 3.16 – Разметка карты Карно для БФ от пяти аргументов

Сложность применения карт Карно и Вейча от пяти переменных заключается в том, что соседними являются столбцы, расположенные не только рядом, но и на расстоянии друг от друга. На рисунке 3.16, например, это столбцы, отмеченные переменными $X_3X_4X_5$ – 001 и 101, 011 и 111. Для карт от шести переменных появляются такие же соседние строки. Поэтому карты Карно и Вейча

чаще применяются для минимизации БФ от 2, 3 и 4 переменных. А булевы функции от пяти и более переменных минимизируются другими способами.

На рисунках 3.17 и 3.18 приведены примеры БФ, нанесенных на карты Вейча и Карно, и показан результат их возможной минимизации.



а) *б*) *в*) *г*)

Рис. 3.17 – Примеры минимизации булевых функций с помощью карт Вейча: P_1 (*а*); P_2 (*б*); P_3 (*в*); P_4 (*г*)

1) $P_1 = \sum(1,\ 3,\ 4,\ 5,\ 8,\ 9,\ 10,\ 14,\ 15),\ \ P_1 = a\bar{b}\bar{c} + a\bar{b}\bar{d} + \bar{a}\bar{b}d + \bar{a}b\bar{c} + abc;$

2) $P_2 = P(1,\ 2,\ 8,\ 10,\ 13,\ 14,\ 15),$

$P_2 = (\bar{a} + \bar{b} + \bar{d})(\bar{a} + \bar{c} + d)(\bar{a} + b + d)(b + \bar{c} + d)(a + b + c + \bar{d});$

3) $P_3 = P(1,\ 3,\ 6,\ 7,\ 10,\ 11,\ 12,\ 14,\ 15),$

$P_3 = (\bar{a} + \bar{c})(a + b + \bar{d})(\bar{b} + \bar{c})(\bar{a} + \bar{b} + d);$

4) $P_4 = \sum(0,\ 3,\ 5,\ 6,\ 9,\ 13,\ 14),\ \ P_4 = a\bar{c}d + \bar{a}\bar{b}\bar{c}\bar{d} + b\bar{c}d + \bar{a}\bar{b}cd + bc\bar{d}.$

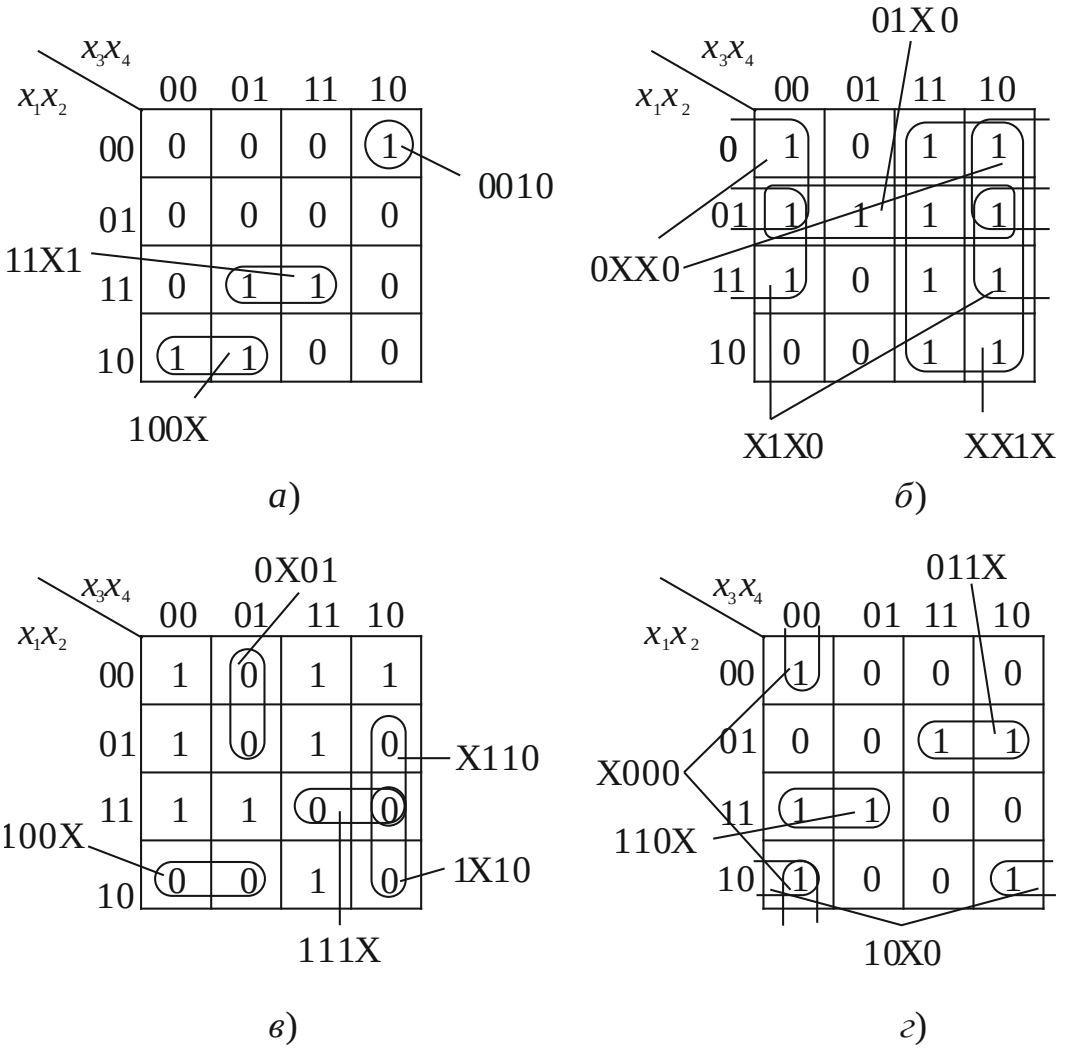


Рис. 3.18 – Примеры минимизации булевых функций с помощью карт Карно: F_1 (*а*); F_2 (*б*); F_3 (*в*); F_4 (*г*)

1) $F_1 = \sum(2,\ 8,\ 9,\ 13,\ 15),\ F_1 = abd + a\bar{b}\bar{c} + \bar{a}\bar{b}cd;$

2) $F_2 = \sum(0,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 10,\ 11,\ 12,\ 14,\ 15),\ F_2 = \bar{a}\bar{d} + b\bar{d} + c + \bar{a}b;$

3) $F_3 = P(1,\ 5,\ 6,\ 8,\ 9,\ 10,\ 14,\ 15),$

$F_3 = (\bar{a} + b + c)(\bar{a} + \bar{c} + d)(\bar{b} + \bar{c} + d)(\bar{a} + \bar{b} + \bar{c})(a + c + \bar{d});$

4) $F_4 = \sum(0,\ 6,\ 7,\ 8,\ 10,\ 12,\ 13),\ F_4 = \bar{b}\bar{c}\bar{d} + ab\bar{c} + a\bar{b}\bar{d} + \bar{a}bc.$

3.3 Функционально полные системы БФ

Систему БФ $\{f_1, f_2, \ldots, f_n\}$ называют полной, если любая БФ может быть выражена суперпозицией функций $f_1, f_2, \ldots, f_n$.

Функции, полученные применением суперпозиции, отличаются от исходных функций, и их свойства отличаются от свойств исходных функций. Среди БФ есть такие, которые обладают основными свойствами. Доказано, что число функций в полной системе не превышает четырех.

Базисом называют полную систему функций алгебры логики, с помощью которых любую БФ можно представить как суперпозицию функций базиса. Минимальный базис состоит из такого набора функций, что исключение любой из них превращает этот набор в неполную систему функций. Известны и широко применяются несколько базисов, состоящих из элементарных функций.

Базис, который называют основным, включает три функции: И, ИЛИ, НЕ – и, соответственно, реализуется тремя логическими элементами.

Базис И-НЕ включает в себя только одну функцию И-НЕ и один логический элемент. Его еще называют универсальным. БФ элемента этого базиса:

$$f = \overline{ab}.$$

На основе этого элемента производятся различные цифровые устройства. Чтобы реализовать любую функцию в этом базисе, необходимо провести преобразования БФ в данный базис. Для этого используются теоремы де Моргана. Например:

$$f_1 = x_1\overline{x}_2 + \overline{x}_1 x_2 = \overline{\overline{x_1\overline{x}_2 + \overline{x}_1 x_2}} = \overline{\overline{x_1\overline{x}_2} \cdot \overline{\overline{x}_1 x_2}}.$$

Таким образом, в окончательной формуле присутствуют только операции И-НЕ. Схема, реализующая эту БФ, приведена на рисунке 3.19.

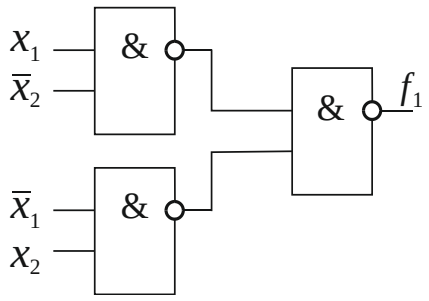


Рис. 3.19 – Реализация БФ в базисе И-НЕ

Базис ИЛИ-НЕ включает в себя одну функцию ИЛИ-НЕ и только один логический элемент также называется универсальным. БФ элемента этого базиса:

$$f = \overline{a + b}.$$

Чтобы реализовать любую функцию в этом базисе, необходимо провести преобразования БФ. Преобразование формулы в базис ИЛИ-НЕ:

$$f_2 = x_1\bar{x}_2 + \bar{x}_1 x_2 = \overline{\overline{x_1\bar{x}_2}} + \overline{\overline{\bar{x}_1 x_2}} = \overline{\bar{x}_1 + x_2} + \overline{x_1 + \bar{x}_2}.$$

Схема, реализующая эту БФ, приведена на рисунке 3.20.

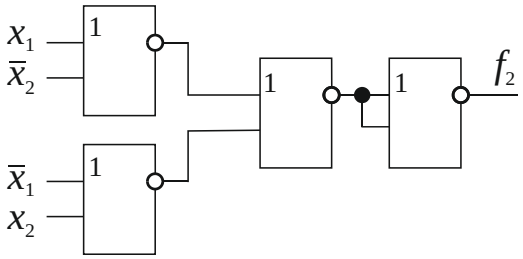


Рис. 3.20 – Реализация БФ в базисе ИЛИ-НЕ

Рассмотрим особенности базисов И-НЕ и ИЛИ-НЕ. Элемент И-НЕ описывается функцией. Таблица истинности этой функции приведена в таблице 3.2.

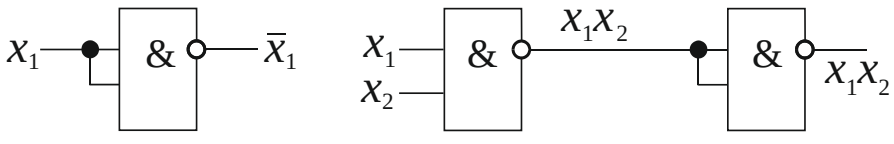
Таблица 3.2 – Таблица истинности функции *F*

X_1	X_2	F
0	0	1
0	1	1
0	1	1
1	1	0

Если на оба входа элемента И-НЕ подать одну и ту же переменную (первая и четвертая строки таблицы), то в результате получается функция:

$$f = \bar{X}_1 \cdot \bar{X}_1 = \bar{X}_1,$$

т. е. элемент И-НЕ становится инвертором (рис. 3.21, *а*). При последовательном соединении двух элементов И-НЕ, как показано на рисунке 3.21, *б*, схема выполняет функцию И. Таким образом, базис И-НЕ содержит как бы три элемента: И-НЕ, И, НЕ.



а) *б*)

Рис. 3.21 – Соединение элементов базиса И-НЕ:
инвертор (НЕ) (*а*); элемент И на двух элементах И-НЕ (*б*)

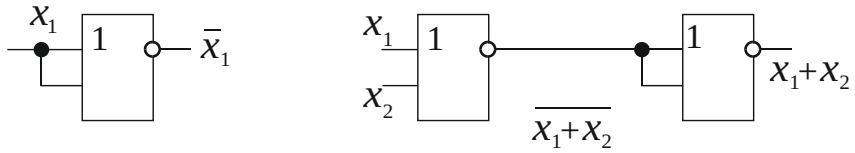
Элемент ИЛИ-НЕ описывается функцией $f = \overline{X_1 + X_2}$.

Таблица истинности этой функции приведена в таблице 3.3.

Таблица 3.3 – Таблица истинности функции F

X_1	X_2	F
0	0	1
0	1	0
0	1	0
1	1	0

Если на оба входа элемента ИЛИ-НЕ подать одну и ту же переменную, то получим функцию: $f = \overline{X_1 + X_1} = \overline{X_1},$ т. е. элемент ИЛИ-НЕ становится инвертором (рис. 3.22, *а*). При последовательном соединении двух элементов ИЛИ-НЕ, как показано на рисунке 3.22, *б*, элемент ИЛИ-НЕ преобразуется в элемент ИЛИ. Таким образом, базис ИЛИ-НЕ содержит как бы три элемента: ИЛИ-НЕ, ИЛИ, НЕ.



а) *б*)

Рис. 3.22 – Соединение элементов базиса ИЛИ-НЕ:
инвертор (НЕ) (*а*); элемент ИЛИ на двух элементах ИЛИ-НЕ (*б*)

Рассмотрим примеры реализации функций в различных базисах.

Функция D (ДНФ) в базисе И-НЕ сначала минимизируется, затем переводится в базис, а потом реализуется (рис. 3.23).

$$D = \sum(1,\ 2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 14) = x_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_4 + x_2x_3\bar{x}_4 + \bar{x}_1x_2 =$$

$$= \overline{\overline{x_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_4 + x_2x_3\bar{x}_4 + \bar{x}_1x_2}} = \overline{\overline{x_1\bar{x}_2\bar{x}_3\bar{x}_4} \cdot \overline{\bar{x}_1x_4} \cdot \overline{x_2x_3\bar{x}_4} \cdot \overline{\bar{x}_1x_2}}.$$

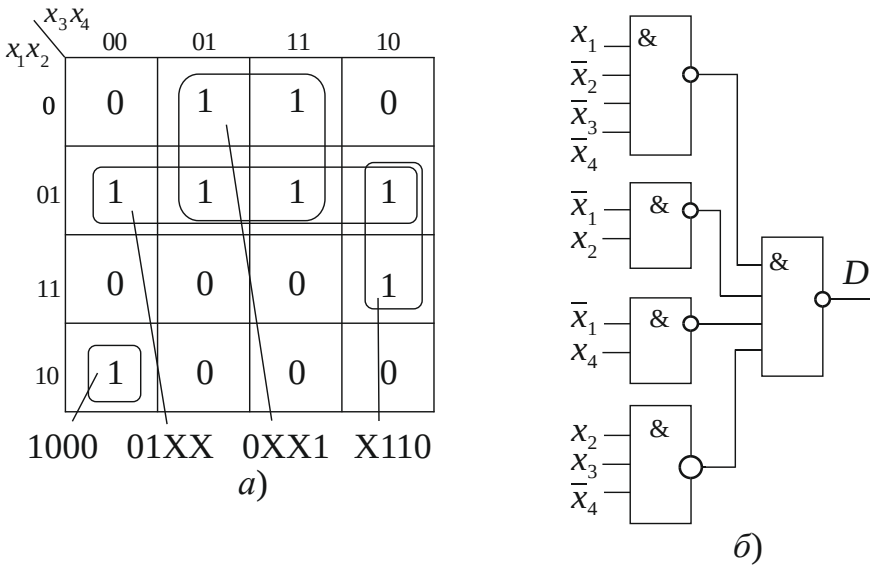


Рис. 3.23 – Функция D – карта Карно (*а*) и схема в базисе И-НЕ (*б*)

Функция K (ДНФ) в базисе И-НЕ (рис. 3.24):

$$K = \sum(0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10,\ 15) = \overline{x}_1 + x_2x_2x_3 + \overline{x}_2\overline{x}_4 =$$

$$= \overline{\overline{\overline{x}_1 + x_2x_2x_3 + \overline{x}_2\overline{x}_4}} = \overline{x_1 + \overline{x_2x_2x_3} + \overline{\overline{x}_2\overline{x}_4}}.$$

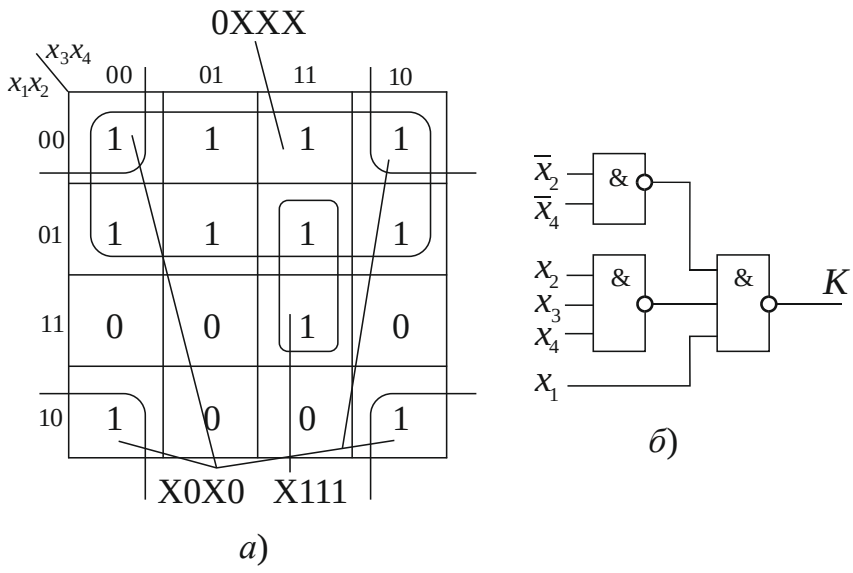
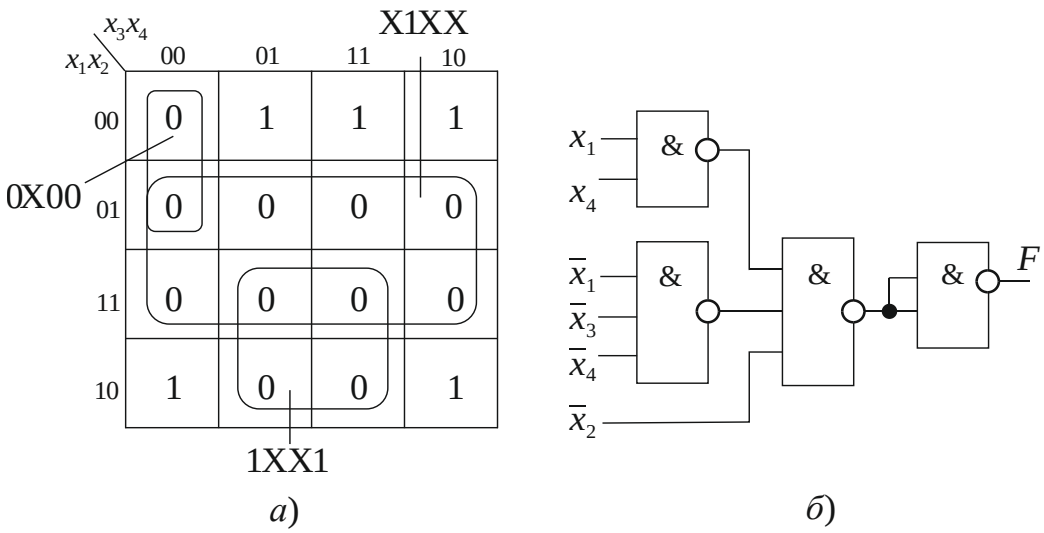


Рис. 3.24 – Функция K – карта Карно (*а*) и схема в базисе И-НЕ (*б*)

Функция F (КНФ) в базисе И-НЕ (рис. 3.25):

$$F = \prod(0,\ 4,\ 5,\ 6,\ 7,\ 9,\ 11,\ 12,\ 13,\ 14,\ 15) = \overline{x}_2(\overline{x}_1 + \overline{x}_4)(x_1 + x_3 + x_4) =$$

$$= (\overline{\overline{x}_2})(\overline{\overline{x}_1 + \overline{x}_4})(\overline{x_1 + x_3 + x_4}) = \overline{x}_2\overline{x_1x_4}\,\overline{\overline{x}_1\overline{x}_3\overline{x}_4}.$$



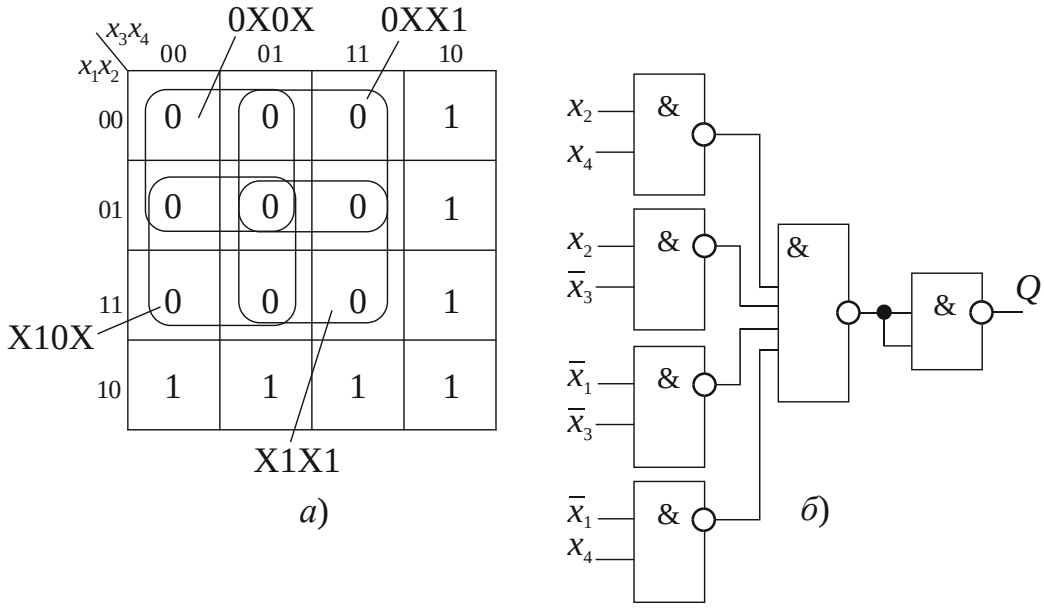
а) *б*)

Рис. 3.25 – Функция *F* – карта Карно (*а*) и схема в базисе И-НЕ (*б*)

Функция *Q* (КНФ) в базисе И-НЕ (рис. 3.26):

$$Q = \prod(0,\ 1,\ 3,\ 4,\ 5,\ 7,\ 12,\ 13,\ 15) = (\overline{x}_2 + \overline{x}_4)(\overline{x}_2 + x_3)(x_1 + x_3)(x_1 + \overline{x}_4) =$$

$$= \overline{\overline{(\overline{x}_2 + \overline{x}_4)}}\ \overline{\overline{(\overline{x}_2 + x_3)}}\ \overline{\overline{(x_1 + x_3)}}\ \overline{\overline{(x_1 + \overline{x}_4)}} = \overline{x_2 x_4}\ \overline{x_2 \overline{x}_3}\ \overline{\overline{x}_1 \overline{x}_3}\ \overline{\overline{x}_1 x_4}.$$



а) *б*)

Рис. 3.26 – Функция *Q* – карта Карно (*а*) и схема в базисе И-НЕ (*б*)

Функция *R* в базисе ИЛИ-НЕ (рис. 3.27):

$$R = \prod(2,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 15) =$$

$$= (\overline{x}_1 + x_3)(x_2 + \overline{x}_3 + x_4)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_4) =$$

$$= \overline{\overline{(\overline{x}_1 + x_3)(x_2 + \overline{x}_3 + x_4)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_4)}} =$$

$$= \overline{\overline{x}_1 + x_3} + \overline{x_2 + \overline{x}_3 + x_4} + \overline{x_1 + \overline{x}_2 + \overline{x}_3} + \overline{\overline{x}_1 + \overline{x}_4}.$$

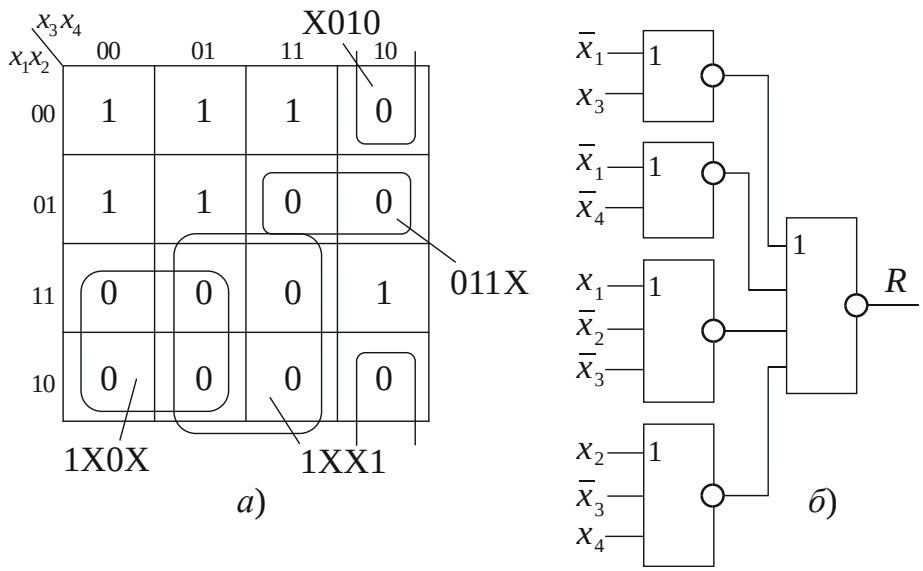


Рис. 3.27 – Функция *R* – карта Карно (*а*) и схема в базисе ИЛИ-НЕ (*б*)

Функция *S* (КНФ) в базисе ИЛИ-НЕ (рис. 3.28):

$$S = \prod(0,\ 1,\ 3,\ 5,\ 7,\ 9,\ 11,\ 13,\ 14,\ 15) = \overline{x}_4(x_1 + x_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) =$$

$$= \overline{\overline{x}_4(x_1 + x_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)} = \overline{x_4 + \overline{x_1 + x_2 + x_3} + \overline{\overline{x}_1 + \overline{x}_2 + \overline{x}_3}}.$$

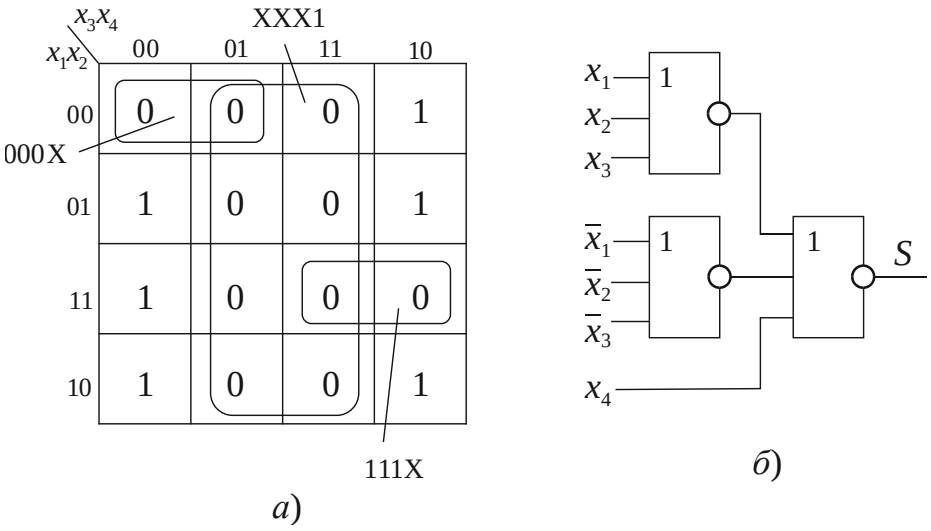


Рис. 3.28 – Функция *S* – карта Карно (*а*) и схема в базисе ИЛИ-НЕ (*б*)

Функция *G* (ДНФ) в базисе ИЛИ-НЕ (рис. 3.29):

$$G = \sum(0,\ 2,\ 3,\ 4,\ 5,\ 6,\ 8,\ 10,\ 11,\ 12,\ 14) = \overline{x}_1 x_2 \overline{x}_3 + \overline{x}_4 + \overline{x}_2 x_3 =$$

$$= \overline{\overline{x}_1 x_2 \overline{x}_3} + \overline{\overline{x}_2 x_3} + \overline{x}_4 = \overline{x}_4 + \overline{x_1 + \overline{x}_2 + x_3} + \overline{x_2 + \overline{x}_3}.$$

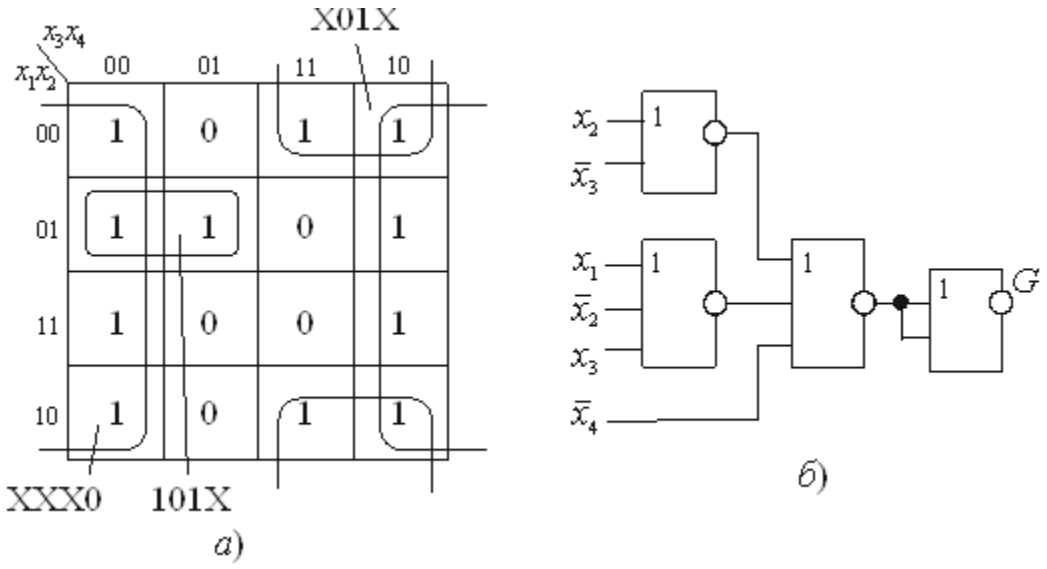
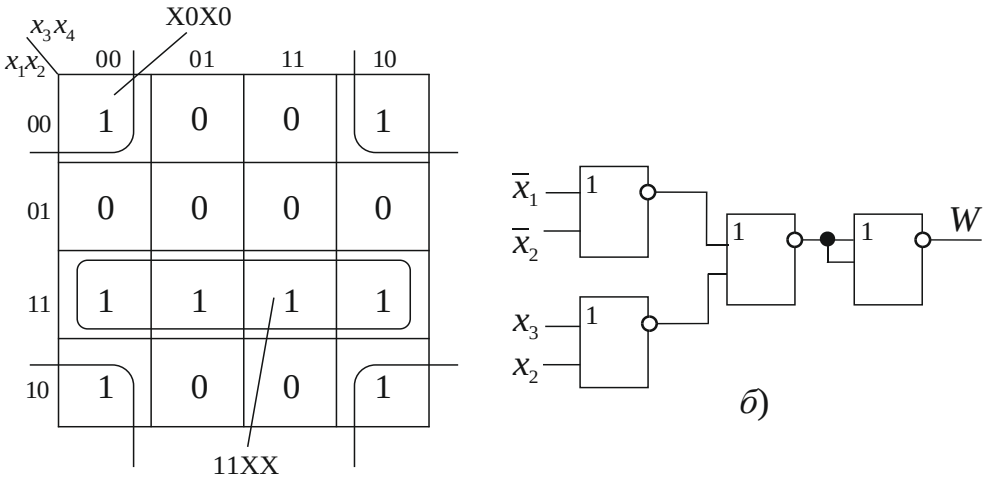


Рис. 3.29 – Функция *G* – карта Карно (*а*) и схема в базисе ИЛИ-НЕ (*б*)

Функция W (ДНФ) в базисе ИЛИ-НЕ (рис. 3.30):

$$W = \sum(0,\ 2,\ 8,\ 10,\ 12,\ 13,\ 14,\ 15) = x_1x_2 + \overline{x}_2\overline{x}_3 =$$

$$= \overline{\overline{x_1x_2} + \overline{\overline{x}_2\overline{x}_3}} = \overline{\overline{x}_1 + \overline{x}_2} + \overline{x_2 + x_3}.$$



а) *б*)

Рис. 3.30 – Функция W – карта Карно (*а*) и схема в базисе ИЛИ-НЕ (*б*)

3.4 Неполностью определенные БФ

Если значение БФ на некоторых наборах не определено, то такая БФ называется *неполностью определенной*. В таблице истинности значения БФ на этих наборах обозначаются крестом (X), в отличие от определенных значений, обозначаемых 0 или 1. Например, неполностью определенная БФ f_1 задана таблицей истинности (табл. 3.4).

Таблица 3.4 – Таблица истинности неполностью определенной БФ

	X_1	X_2	X_3	X_4	f_1
0	0	0	0	0	X
1	0	0	0	1	X
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	X
9	1	0	0	1	1
10	1	0	1	0	X
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Из таблицы 3.4 видно, что на 0, 1, 8, 10-м наборах данная функция не определена. При задании неполностью определенных БФ числовым способом неопределенные наборы заключаются в дополнительные скобки, например, для f_1, заданной СДНФ $f_1 = \sum(2,\ 4,\ 9,\ 11(0,\ 1,\ 8,\ 10))$, а при задании этой же функции в СКНФ $f_1 = P(3,\ 5,\ 6,\ 7,\ 12,\ 13,\ 14,\ 15(0,\ 1,\ 8,\ 10))$.

При минимизации неполностью определенных БФ с помощью карт Карно – Вейча алгоритм образования контуров дополняется следующими правилами:

- при образовании контура, клетки карты, отмеченные X, т. е. неопределенным значением, в любой момент времени, *если это выгодно для образования контура большего размера*, могут считаться отмеченными 1 для СДНФ или 0 для СКНФ;
- любой контур не должен покрывать *только* клетки, отмеченные неопределенными значениями X;
- не все клетки, отмеченные X, могут быть использованы при образовании контуров.

На рисунке 3.31 показана карта Карно, на которую нанесена БФ f_1 и проведена ее минимизация.

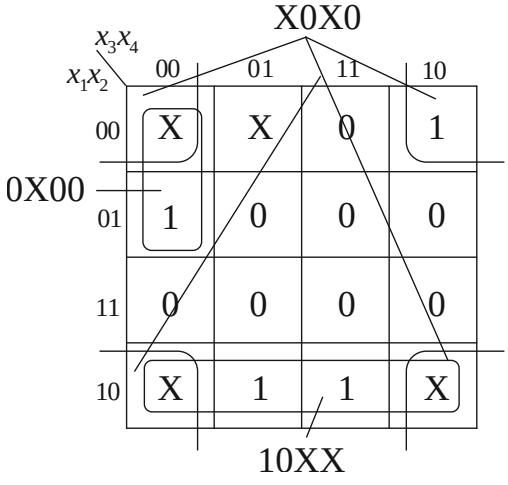


Рис. 3.31 – Минимизация БФ f_1

Функция (рис. 3.31) покрывается тремя контурами и описывается следующей формулой:

$$f_1 = \overline{X_1}\,\overline{X_3}\,\overline{X_4} + X_1\overline{X_2} + \overline{X_2}\,\overline{X_4}.$$

На рисунке 3.32 приведены примеры неполностью определенных БФ S_i и проведена их минимизация (используются карты Карно и Вейча).

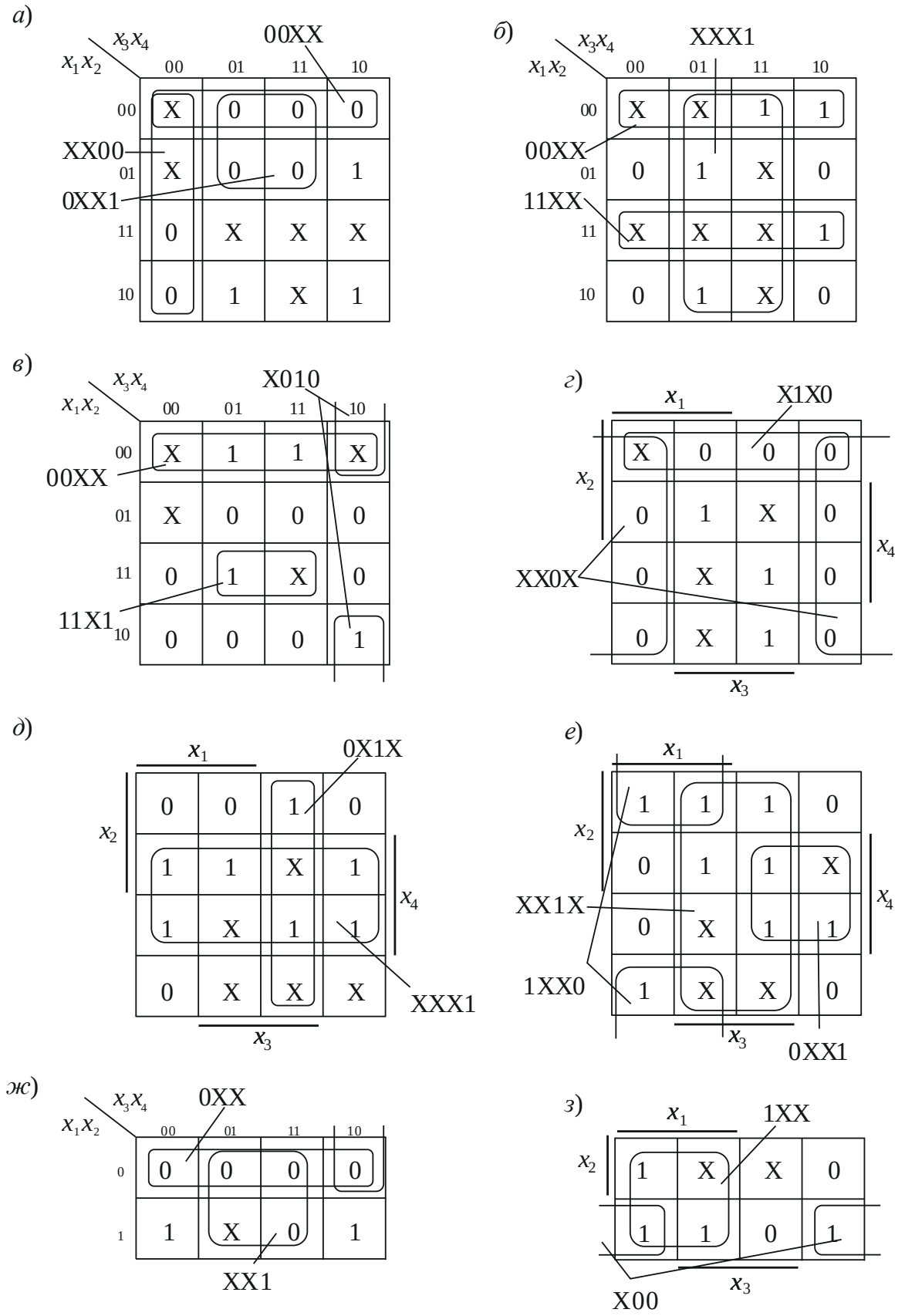


Рис. 3.32 – Минимизация неполностью определенных БФ

1) $S_1 = P\left(1,\ 2,\ 3,\ 5,\ 7,\ 8,\ 12\ \left(0,\ 4,\ 11,\ 13,\ 14,\ 15\right)\right)$ (рис. 3.32, *а*)

$\left(L = 6,\ C = 9\right),\ S_1 = (X_3 + X_4)(X_1 + \overline{X_4})(X_1 + X_2)$;

2) $S_2 = \sum(2,\ 3,\ 5,\ 9,\ 14\ (0,\ 1,\ 7,\ 11,\ 12,\ 13,\ 15))$ (рис. 3.32, *б*)
$(L = 5,\ C = 7),\ S_2 = X_4 + X_1X_2 + \overline{X_1}\,\overline{X_2};$

3) $S_3 = \sum(1,\ 3,\ 10,\ 13\ (0,\ 2,\ 4,\ 15))$ (рис. 3.32, *в*) $(L = 8,\ C = 11)$,
$S_3 = \overline{X_1}\,\overline{X_2} + X_1X_2X_4 + \overline{X_2}X_2\,\overline{X_4};$

4) $S_4 = P(0,\ 1,\ 4,\ 5,\ 6,\ 8,\ 9,\ 13,\ 14\ (7,\ 10,\ 11,\ 12))$ (рис. 3.32, *г*)
$(L = 3,\ C = 4),\ S_4 = X_3(\overline{X_2} + X_4);$

5) $S_5 = \sum(1,\ 3,\ 5,\ 6,\ 9,\ 13,\ 15\ (0,\ 2,\ 7,\ 10,\ 11))$ (рис. 3.32, *д*)
$(L = 3,\ C = 4),\ S_5 = X_4 + \overline{X_1}X_3;$

6) $S_6 = \sum(1,\ 3,\ 6,\ 7,\ 8,\ 12,\ 14,\ 15\ (2,\ 5,\ 10,\ 11))$ (рис. 3.32, *е*)
$(L = 5,\ C = 7),\ S_6 = X_3 + X_1\overline{X_4} + \overline{X_1}X_4;$

7) $S_7 = P\ (0,\ 1,\ 2,\ 3,\ 7\ (5))$ (рис. 3.32, *ж*) $(L = 2,\ C = 2),\ S_7 = X_1\overline{X_3};$

8) $S_8 = \sum(0,\ 4,\ 5,\ 6\ (3,7))$ (рис. 3.32, *з*) $(L = 3,\ C = 4),\ S_8 = X_1 + \overline{X_2}\,\overline{X_3}.$

3.5 Скобочные формы БФ

Тупиковые ДНФ и КНФ могут быть не самыми короткими формулами БФ. Например, формула $f = ab + a\bar{c}$ имеет длину $L = 4,$ но если вынести за скобку переменную *a*, то получим $f = a(b + \bar{c})$ и длина формулы уменьшается до $L = 3.$ Такая форма БФ носит название *скобочной*.

Рассмотрим пример построения скобочной формы функции, заданной ДНФ:

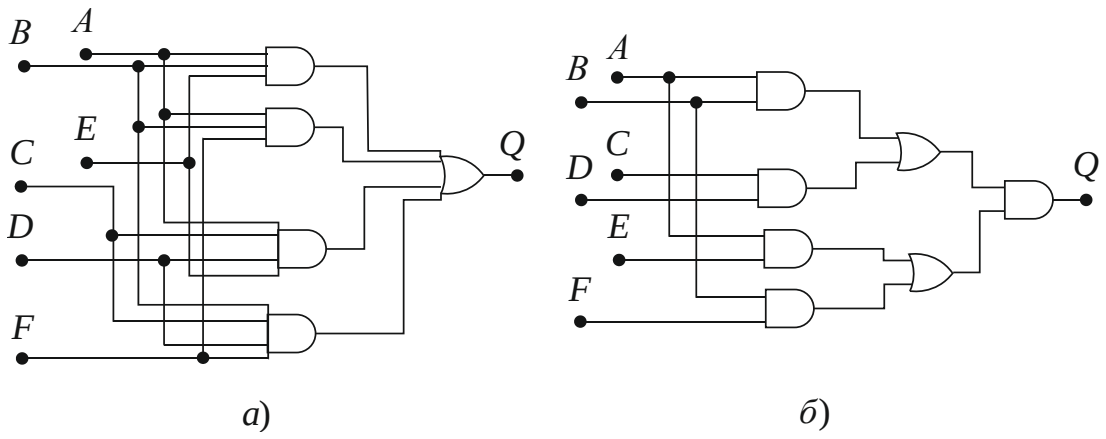
$$Q = ABE + ABF + ACDE + BCDF.$$

Преобразуем функцию вынесением за скобки общих сомножителей:

$$Q = ABE + ABF + ACDE + BCDF = AB(AE + BF) + CD(AE + BF) =$$
$$= (AE + BF)(AB + CD).$$

Цена схемы для исходной функции $C = 14,$ для функции после преобразования в скобочную форму $C = 8.$

На рисунке 3.33 приведены КС, реализующие обе функции.



а) *б*)

Рис. 3.33 – Реализация функции *Q*:
в форме ДНФ (*а*); в скобочной форме (*б*)

Контрольные вопросы по главе 3

1. Какие логические элементы используются при синтезе комбинационных схем?
2. Может ли элемент И-НЕ иметь два выхода?
3. Если на карте Карно имеются клетки, отмеченные только 1 и X, то чему равно значение БФ?
4. Сколько независимых переменных в имени контура, состоящем из двух клеток?
5. Сколько независимых переменных в имени контура, состоящем из четырех клеток?
6. Сколько независимых переменных в имени контура, состоящем из восьми клеток?
7. Какой закон используется для перевода БФ, записанной в базисе И, ИЛИ, НЕ, в базис И-НЕ?
8. Какой закон используется при преобразовании формулы БФ, заданной в СДНФ, в скобочную форму?
9. При минимизации БФ с помощью карт возможно ли получение нескольких минимальных формул?
10. БФ нанесли на карту Карно и карту Вейча. После минимизации были получены две отличающиеся БФ с одинаковой длиной формулы. Возможна ли такая ситуация?

4 Анализ комбинационных схем

Часто возникают ситуации, когда есть готовое устройство, а закон его работы неизвестен. Или есть схема устройства, а какую функцию оно выполняет, также неизвестно. В этом случае осуществляется анализ работы устройства. Рассмотрим, как проводится анализ работы комбинационной схемы. Задача анализа комбинационных схем состоит в нахождении закона работы, то есть восстановлении таблицы истинности или формулы БФ, описывающей эту комбинационную схему. Анализ проводят для того, чтобы определить функциональные свойства КС, или для проверки правильности функционирования КС, или же для того, чтобы определить работоспособность схемы при повреждениях некоторых его элементов.

При анализе КС можно попытаться определить, насколько экономично была спроектирована КС и возможно ли ее упрощение. Это можно проверить минимизацией восстановленного в результате анализа закона работы БФ. Отдельной задачей является выяснение поведения устройства в переходных режимах и нарушения работы именно в эти периоды. Анализ КС проводят в два этапа.

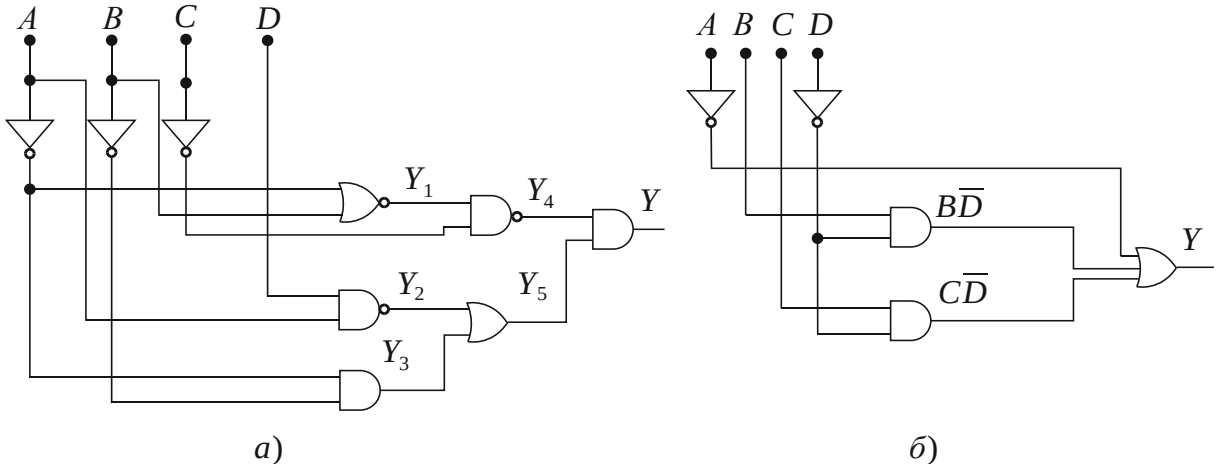
1. Из имеющейся принципиальной схемы устройства удаляют все несущественные или вспомогательные элементы, которые не влияют на логику работы КС, а предназначены для обеспечения надежности или устойчивости, или дополнительных возможностей схемы. Например, удаляются инверторы, которые используются для получения инверсных значений переменных. Тогда схема будет состоять только из элементов, выполняющих логические функции.

2. На основе полученной таким образом схемы восстанавливается БФ, которая и подвергается анализу. При анализе работы устройства, выполненного на электронных элементах, функция записывается непосредственно по самой схеме. Такие действия можно выполнить по следующему алгоритму:

- выходу каждого логического элемента приписывается какое-либо имя;
- выходная функция каждого элемента записывается в терминах переменных на ее входах;
- последовательно от входов к выходам описывается вся схема;
- функция преобразуется в базис И, ИЛИ, НЕ и представляется в ДНФ;

- при необходимости функция доопределяется до СДНФ и минимизируется;
- формируются выводы о соответствии полученной БФ и работы анализируемого устройства.

Рассмотрим последовательность работы на примере схемы, представленной на рисунке 4.1.



а) *б*)

Рис. 4.1 – Комбинационная схема и функция *Y*:
исходная КС (*а*); минимизированная КС (*б*)

Выход каждого элемента (рис. 4.1, *а*) обозначим своей функцией. Запишем функцию каждого элемента, выведем БФ конечного элемента *Y*.

$$Y_1 = \overline{\bar{A} + B}, \qquad Y_2 = \overline{AD}, \qquad Y_3 = \overline{A}\,\overline{B}, \qquad Y_4 = \overline{Y_1 \cdot \bar{C}},$$

$$Y_5 = Y_2 + Y_3, \qquad Y = Y_4 \cdot Y_5.$$

Преобразуем полученную формулу, минимизируем ее и построим схему (рис. 4.1, б). Теперь подставим все в *Y* и выполним все преобразования функции *Y*:

$$Y = Y_4 \cdot Y_5 = \overline{Y_1 \cdot \bar{C}} \cdot (Y_2 + Y_3) = \overline{\overline{\bar{A} + B \cdot \bar{C}}} \cdot (\overline{AD} + \overline{A}\,\overline{B}) =$$

$$= \overline{A}\,\overline{B}\,\overline{\bar{C}}\,(\overline{AD} + \overline{A}\,\overline{B}) = (\overline{A} + B + C)(\overline{A} + \overline{D} + \cancel{\overline{A}\,\overline{B}}) =$$

$$= \overline{A} + B(\overline{A} + \overline{D}) + \mathrm{C}(\overline{A} + \overline{D}) = \overline{A} + \cancel{\overline{A}\,\overline{B}} + BD + \cancel{\overline{A}\,\overline{C}} + C\overline{D} =$$

$$= \overline{A} + B\overline{D} + C\overline{D}.$$

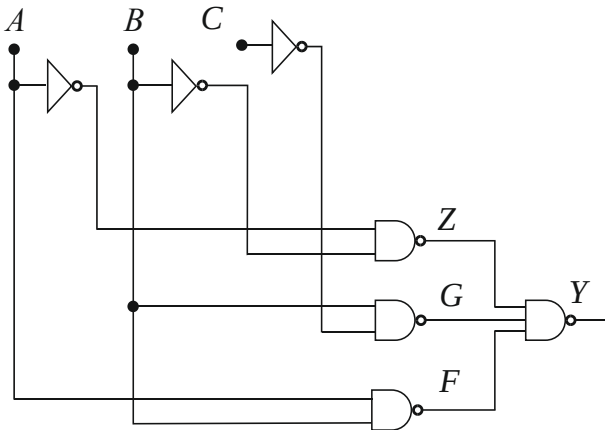
Очевидно, что исходная схема (рис. 4.1, *а*) спроектирована избыточно и ее можно заменить эквивалентной ей (рис. 4.1, *б*).

Закон схемы можно представить и в виде таблицы истинности. Для этого составляют таблицу, в левой части которой записаны все возможные наборы аргументов. Затем вычисляется значение выходного сигнала схемы для каждого

набора переменных и результат заносится в таблицу. При этом можно использовать следующие очевидные свойства элементов *А*. Для электронных схем:

- 0 на любом входе схемы И приводит к появлению 0 на выходе, независимо от сигналов на любых других входах;
- 1 на любом входе схемы ИЛИ приводит к появлению 1 на выходе, независимо от сигналов на любых других входах;
- 0 на любом входе схемы И-НЕ приводит к появлению 1 на выходе, независимо от сигналов на любых других входах;
- 1 на любом входе схемы ИЛИ-НЕ приводит к появлению 0 на выходе, независимо от сигналов на любых других входах.

Например, задана КС (рис. 4.2). Составим для нее таблицу истинности.



№	*ABC*	*Z*	*G*	*F*	*Y*
0	000	0	1	1	1
1	001	0	1	1	1
2	010	1	0	1	1
3	011	1	1	1	0
4	100	1	1	1	0
5	101	1	1	1	0
6	110	1	0	0	1
7	111	1	1	0	1

Рис. 4.2 – Комбинационная схема и таблица истинности

Обозначим выходы элементов *Z, G, F, Y* и составим таблицу истинности:

- функцию *Z* реализует элемент И-НЕ, на выходе которого будет 1, если *A* = 1 или *B* = 1 (строки со второй по 7);
- функцию *G* реализует элемент И-НЕ, на выходе которого будет 1, если *B* = 0 или *C* = 1 (строки 0, 1, 3, 4, 5, 7);
- функцию *F* реализует элемент И-НЕ, на выходе которого будет 1, если *A* = 0 или *B* = 0 (строки 0, 1, 2, 3, 4, 5);
- функцию *Y* реализует элемент И-НЕ, на выходе которого будет 1 в тех строках, где или *Z* = 0 или *G* = 0, или *F* = 0 (строки 0, 1, 2, 6, 7).

При решении таких задач промежуточные столбцы (*Z, G, F*) в таблице истинности могут быть опущены, а вычисление выходных сигналов производится прямо на схеме. На рисунке 4.3 для примера рассмотрены состояния выхода схемы для нескольких наборов переменных $ABC = \{010, 100, 101, 111\}$. Резуль-

тат совпадает со значениями функции Y в таблице истинности. Если минимизировать эту функцию, то получим формулу:

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

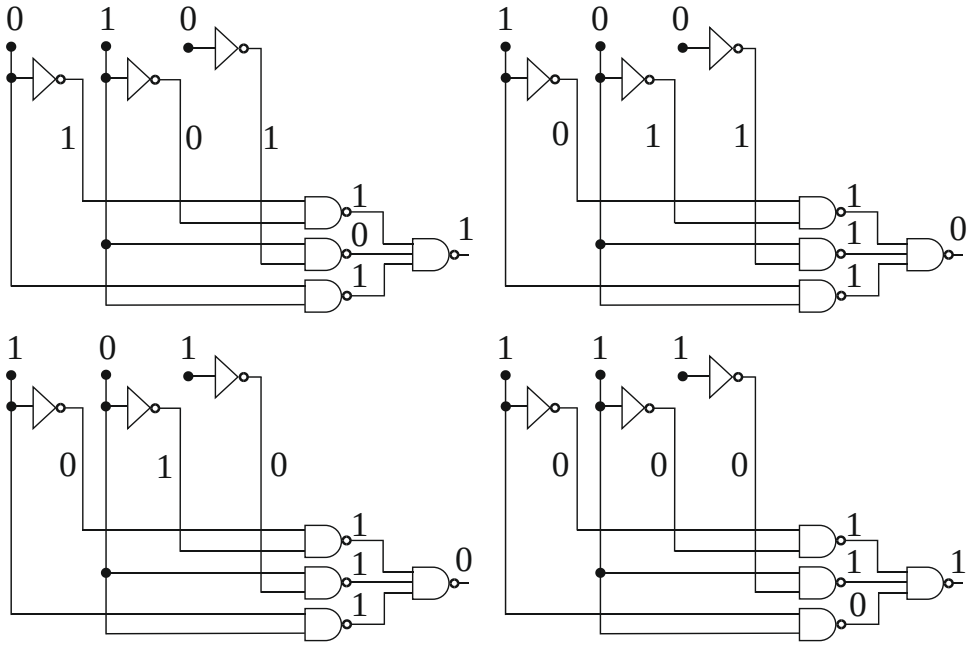
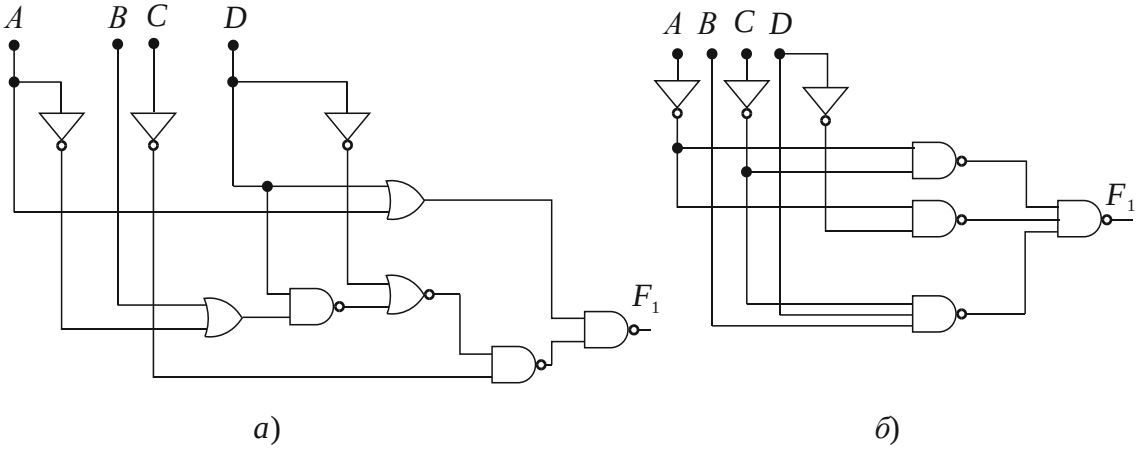


Рис. 4.3 – Значения выхода схемы для некоторых наборов

В результате анализа получена эквивалентная функция, записанная в базисе И, ИЛИ, НЕ. Поскольку цены схем одинаковы, то можно сказать, схема (рис. 4.2) спроектирована без избыточности.

На рисунках 4.4–4.6 приведены схемы устройств. Таблицы истинности схем сведены в таблице 4.1. Полученные функции минимизированы и построены новые схемы на основе в форме ДНФ в различных базисах.



а) *б*)

Рис. 4.4 – Преобразование схемы для F_1:
исходная схема (*а*); минимизированная схема в базисе И-НЕ (*б*)

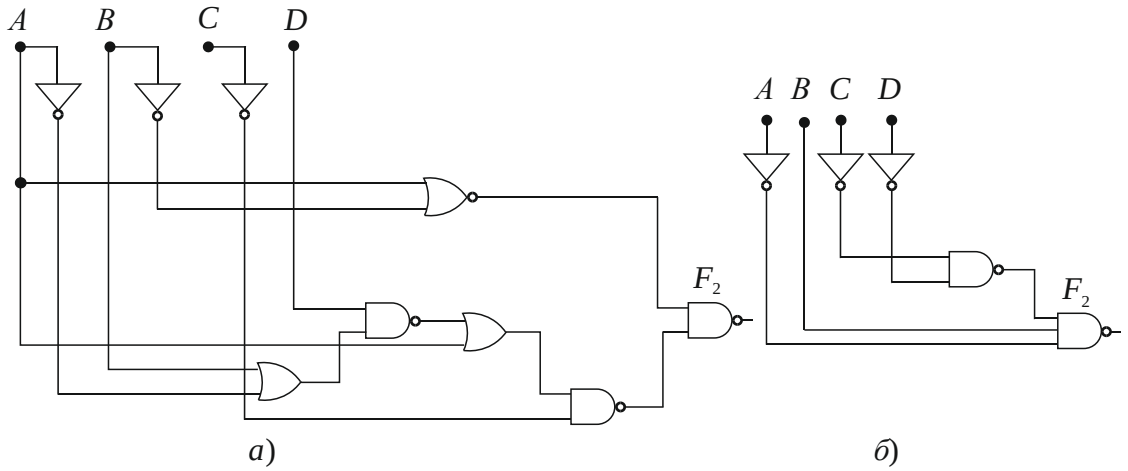


Рис. 4.5 – Преобразование схемы для F_2:
исходная схема (*а*); минимизированная схема в базисе И-НЕ (*б*)

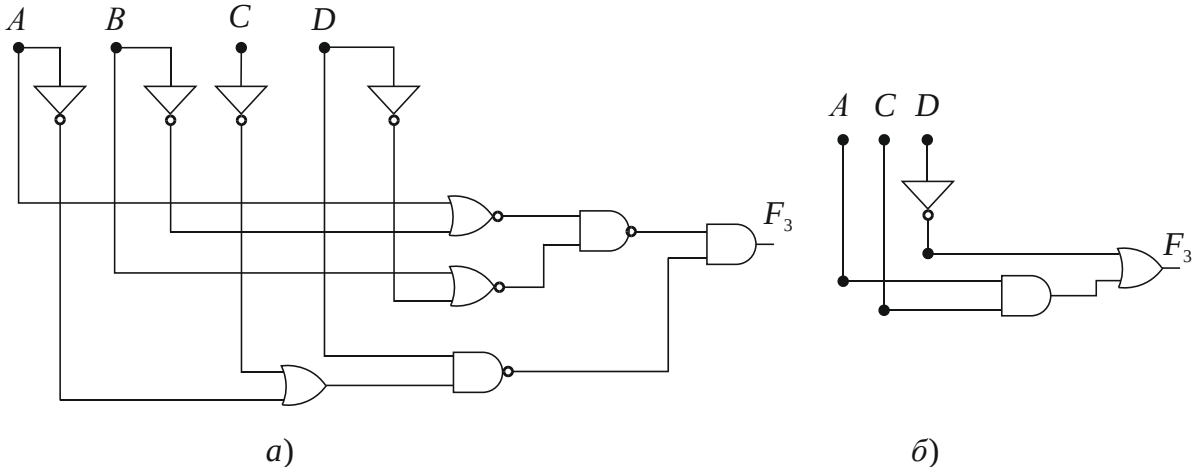


Рис. 4.6 – Преобразование схемы для F_3:
исходная схема (*а*); минимизированная схема в базисе И, ИЛИ, НЕ (*б*)

Таблица 4.1 – Таблица истинности для F_1, F_2, F_3,

№	***ABCD***	F_1	F_2	F_3
0	0000	1	1	1
1	0001	1	1	0
2	0010	1	1	1
3	0011	0	1	0
4	0100	1	1	1
5	0101	1	0	0
6	0110	1	0	1
7	0111	0	0	0
8	1000	0	1	1
9	1001	0	1	0
10	1010	0	1	1
11	1011	0	1	1
12	1100	0	1	1
13	1101	1	1	0
14	1110	0	1	1
15	1111	0	1	1

В следующем примере (рис. 4.7) приведена схема устройства. Выведены значения выходной функции (элементы *DD*5) и приведена эквивалентная схема (выход – элемент *DD*6).

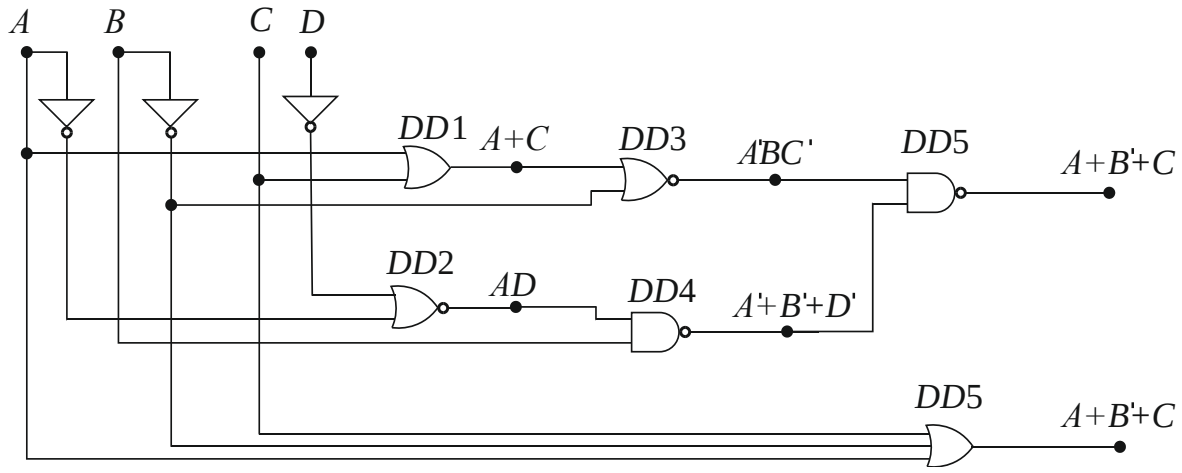


Рис. 4.7 – Анализ схемы и ее минимизация. Пример 1

Продемонстрируем вывод выходной функции схемы (*DD*5):

$$DD1:\ A+C;$$

$$DD2:\ \overline{\overline{A}+\overline{D}}=AD;$$

$$DD3:\ \overline{DD1+\overline{B}}=\overline{(A+C)+\overline{B}}=\overline{A}B\overline{C};$$

$$DD4:\ \overline{ADB}=\overline{A}+\overline{B}+\overline{D};$$

$$DD5:\ \overline{\overline{A}B\overline{C}\cdot(\overline{A}+\overline{B}+\overline{D})}=\overline{\overline{A}B\overline{C}}+\overline{(\overline{A}+\overline{B}+\overline{D})}=A+\overline{B}+C+ABD=A+\overline{B}+C.$$

На рисунках 4.8, 4.9 изображены еще две схемы, для которых приведены значения выходных функций.

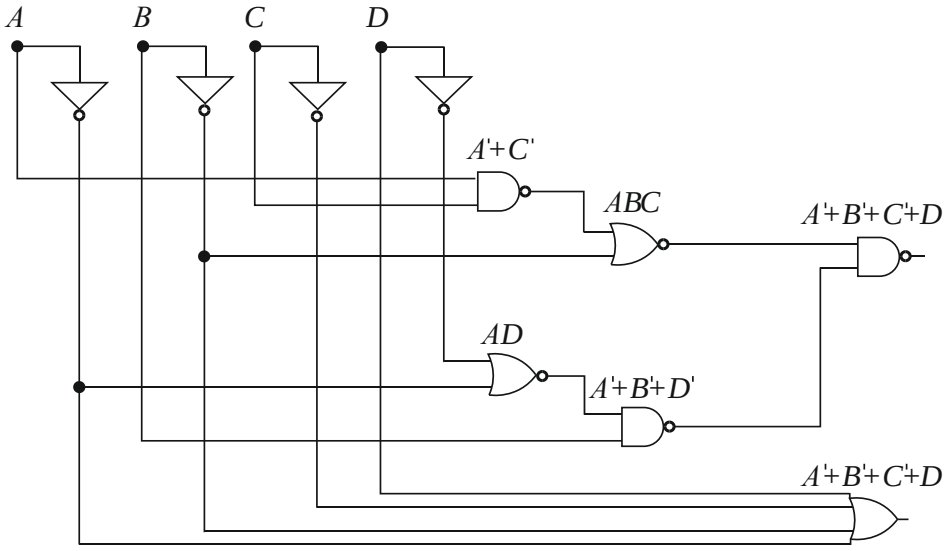


Рис. 4.8 – Анализ схемы и ее минимизация. Пример 2

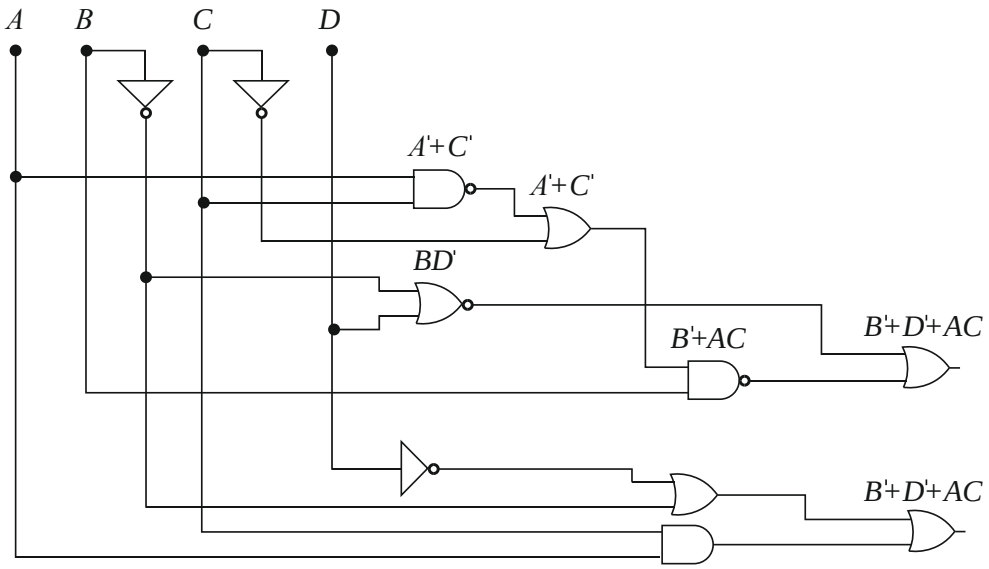


Рис. 4.9 – Анализ схемы и ее минимизация. Пример 3

Если комбинационная схема реализована в виде прибора, то БФ этой КС можно восстановить с помощью тестирующего оборудования, которое может формировать сигналы, соответствующие всевозможным наборам аргументов. Устройство подключается к КС, и снимается диаграмма его работы, по которой строится таблица истинности БФ. На рисунке 4.10 приведена диаграмма работы, снятая с КС, реализующей БФ *W*. Задающее устройство формирует последовательно наборы переменных *ABCD* от 1111 до 0000. На диаграмме показаны значения наборов, на которых выходной сигнал *W* принимает единичные значения.

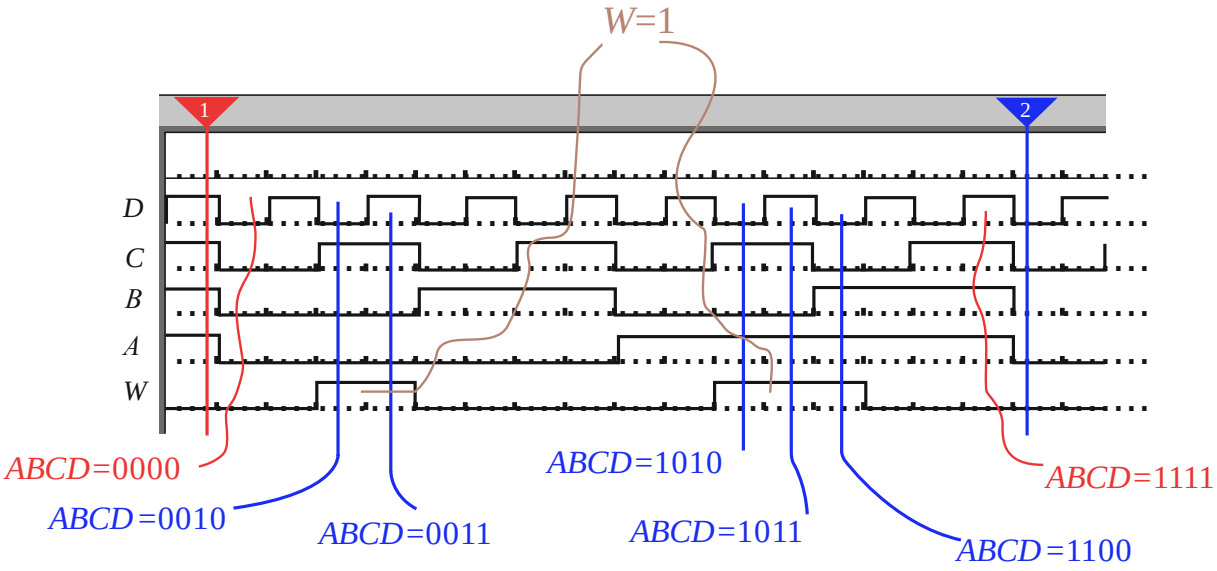


Рис. 4.10 – Диаграмма КС, реализующей функцию *W*

По диаграмме можно записать БФ в форме ДНФ числовым способом:

$$W = \sum(2,\ 3,\ 10,\ 11,\ 12) = \bar{b}c + ab\bar{c}\bar{d}.$$

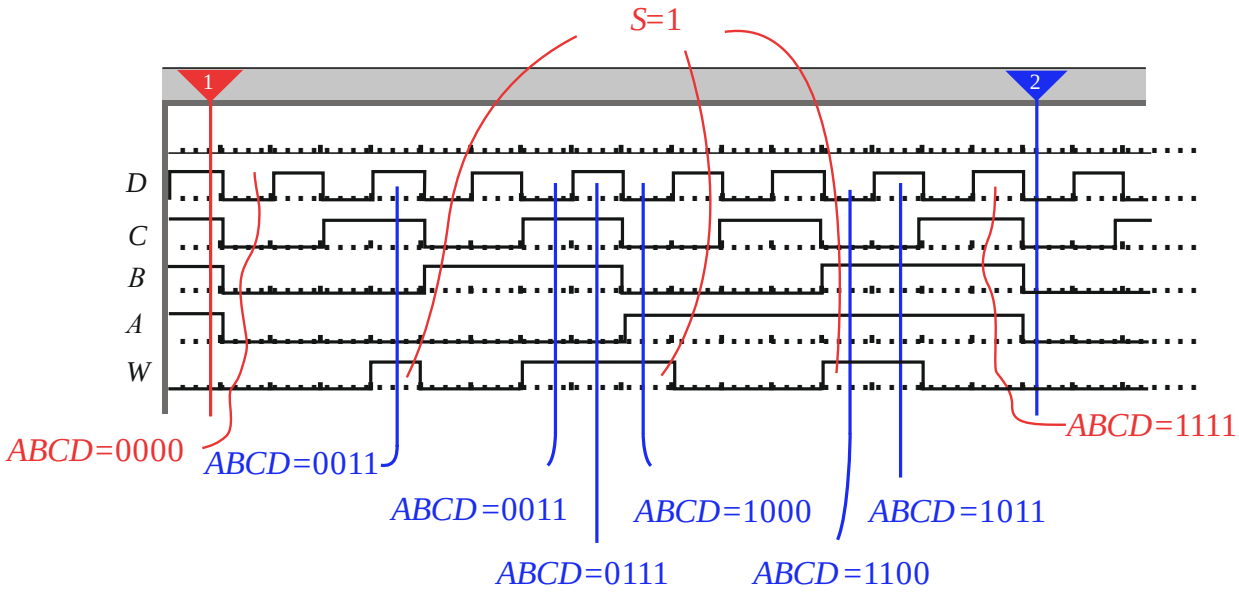


Рис. 4.11 – Диаграмма КС, реализующей функцию *S*

КС, диаграмма которой приведена на рисунке 4.11, реализует БФ:

$$S = \sum(3,\ 6,\ 7,\ 8,\ 12,\ 13) = ab\bar{c} + a\bar{c}\bar{d} + \bar{a}cd + \bar{a}bc.$$

Контрольные вопросы по главе 4

1. Всегда ли возможно при анализе КС восстановить таблицу истинности?
2. Назовите, какую функцию реализует логический элемент *DD*3 на рисунке 4.7.
3. Как зависит выходной сигнал на выходе элемента И-НЕ от нулевого сигнала на одном из его входов?
4. Как зависит выходной сигнал на выходе элемента ИЛИ-НЕ от единичного сигнала на одном из его входов?
5. Что показывает диаграмма, приведенная на рисунке 4.11?
6. Всегда ли в результате преобразования формулы, описывающей работу КС, получается эквивалентная ей функция, меньшая по длине, чем исходная?
7. Будут ли учитываться вспомогательные элементы при анализе КС?
8. Как при анализе КС отличить вспомогательные элементы от основных?

9. Можно ли по диаграмме работы КС определить форму булевой функции ДНФ или КНФ?
10. При анализе КС восстановили таблицу истинности, а она не совпадает с диаграммой работы. В каком случае это может произойти?

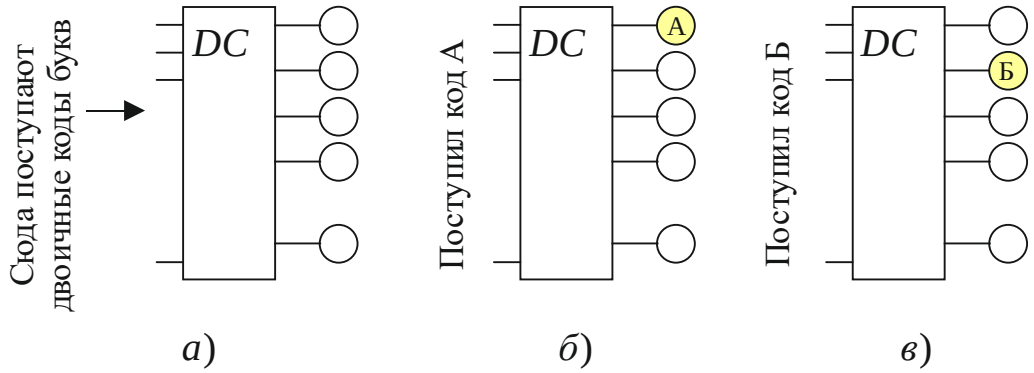
5 Узлы цифровых устройств

Узлы цифровых устройств – это функционально законченные устройства, в состав которых входит некоторое количество (от единиц до нескольких сотен) логических элементов. Узлы выполняют функции, которые широко используются при построении сложных цифровых устройств. Рассмотрим законы функционирования этих устройств и их схемы.

5.1 Дешифраторы

***Дешифратор* (*декодер*)** *– это устройство, которое преобразует двоичный n-разрядный код в 2n-разрядный унитарный двоичный код. Унитарным кодом называется такой двоичный код, у которого состояние одного двоичного разряда противоположно состоянию всех остальных разрядов, т. е. если все разряды равны 0, то только один разряд равен 1.*

Например, дешифратор кодов букв. Из информатики известно, что каждому символу алфавита присвоен свой код, отличный от кодов других символов. На рисунке 5.1, *а* представлен дешифратор кодов букв в момент, когда на входах присутствует код, не присвоенный ни одной букве. На рисунке 5.1, *б, в* показана ситуация, когда на вход дешифратора подаются коды букв А и Б. Из рисунка видно, что в каждом случае только на одном из выходов появляется активный сигнал, а все остальные выходы неактивны (это и есть унитарный код).



а) *б*) *в*)

Рис. 5.1 – Пример работы дешифратора:
при отсутствии кода символа (*а*); при подаче
кода символа А (*б*); при подаче кода символа Б (*в*)

Дешифраторы на схеме обозначаются латинскими буквами *DC* (декодер). Разрядность дешифратора считается по числу входных переменных. Так, при $n = 2$, дешифратор называется двухразрядным, при $n = 3$ – трехразрядным и т. д. Если *DC* имеет $2n$ выходов, то он называется полным, в противном случае – неполным.

Например, двухразрядный полный *DC* при значении входов 00, 01, 10, 11 может формировать на выходах один из следующих кодов: 1000, 0100, 0010, 0001. Такой *DC* называется *DC* с прямыми выходами. Если *DC* формирует на выходе коды 0111, 1011, 1101, 1110, то он называется инверсным. На схемах дешифраторы обозначают так, как показано на рисунке 5.2.

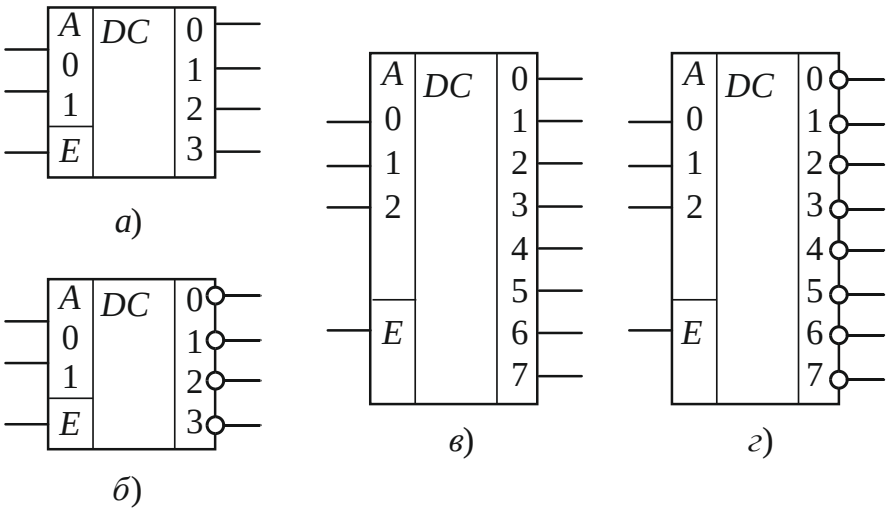


Рис. 5.2 – Обозначения дешифраторов:
полный управляемый двухразрядный *DC* с прямыми выходами (*а*);
полный управляемый двухразрядный *DC* с инверсными выходами (*б*);
полный управляемый трехразрядный *DC* с прямыми выходами (*в*);
полный управляемый трехразрядный *DC* с инверсными выходами (*г*)

Активный выход, имеющий противоположное по отношению к остальным выходам значение, называется *возбужденным*. В обозначениях дешифраторов различают несколько полей (рис. 5.2):

- *n*-разрядное адресное поле (обозначено буквой *A*) показывает, сколько разрядов имеет входной код, где цифры 0, 1, 2 показывают вес разряда двоичного кода (0 – младший разряд, 2 – следующий разряд двоичного кода и т. д.);

- выходное поле полного дешифратора (обозначений может не иметь) содержит 2^n выходов. На схемах выходы дешифратора с инверсными выходами обозначаются инверсиями;
- поле *E* (может иметь и другое обозначение, например *W*) – поле управляющего входа, может присутствовать или отсутствовать, и тогда его в обозначении *DC* опускают. Назначение этого входа рассмотрено ниже.

Дешифратор является комбинационной схемой, имеющей несколько выходов, каждый из которых описывается своей БФ. На рисунке 5.3 приведены таблица истинности *DC*, изображенного на рисунке 5.2, *а*, и БФ всех его выходов.

	X_1X_0	Y_0	Y_1	Y_2	Y_3
0	00	1	0	0	0
1	01	0	1	0	0
2	10	0	0	1	0
3	11	0	0	0	1

$$Y_0 = \overline{X}_1 \cdot \overline{X}_0$$
$$Y_1 = \overline{X}_1 \cdot X_0$$
$$Y_2 = X_1 \cdot \overline{X}_0$$
$$Y_3 = X_1 \cdot X_0$$

Рис. 5.3 – Таблица истинности и БФ выходов двухразрядного дешифратора с прямыми выходами (в формате ДНФ)

При поступлении на вход дешифратора двоичного кода 00 возбуждается выход 0, при поступлении кода 01 возбуждается выход 1, при 10 на входах возбуждается выход 2 и при 11 – выход 3.

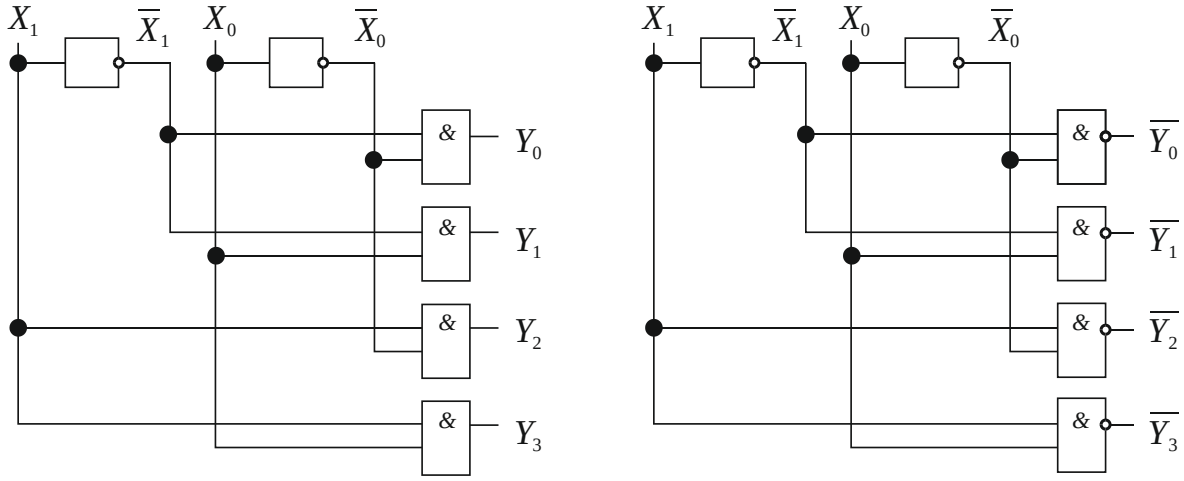
В таблице истинности *DC*, в любом *k*-м столбце (рис. 5.3) только одна единица, а в таблице (рис. 5.4) только один ноль. То есть функция Y_k фактически описывается конституентой единицы для дешифратора с прямыми выходами и конституентой нуля для дешифратора с инверсными выходами.

	X_1X_0	Y_0	Y_1	Y_2	Y_3
0	00	0	1	1	1
1	01	1	0	1	1
2	10	1	1	0	1
3	11	1	1	1	0

$$Y_0 = X_1 \cdot X_0 = \overline{\overline{X_1 \cdot X_0}} = \overline{\overline{X}_1 \cdot \overline{X}_0}$$
$$Y_1 = X_1 \cdot \overline{X}_0 = \overline{\overline{X_1 \cdot \overline{X}_0}} = \overline{\overline{X}_1 \cdot X_0}$$
$$Y_2 = \overline{X}_1 \cdot X_0 = \overline{\overline{\overline{X}_1 \cdot X_0}} = \overline{X_1 \cdot \overline{X}_0}$$
$$Y_3 = \overline{X}_1 \cdot \overline{X}_0 = \overline{\overline{\overline{X}_1 \cdot \overline{X}_0}} = \overline{X_1 \cdot X_0}$$

Рис. 5.4 – Таблица истинности и БФ выходов двухразрядного дешифратора с инверсными выходами (в формате КНФ)

В соответствии с функциями (рис. 5.3, 5.4) построены схемы дешифраторов. Они представлены на рисунке 5.5.

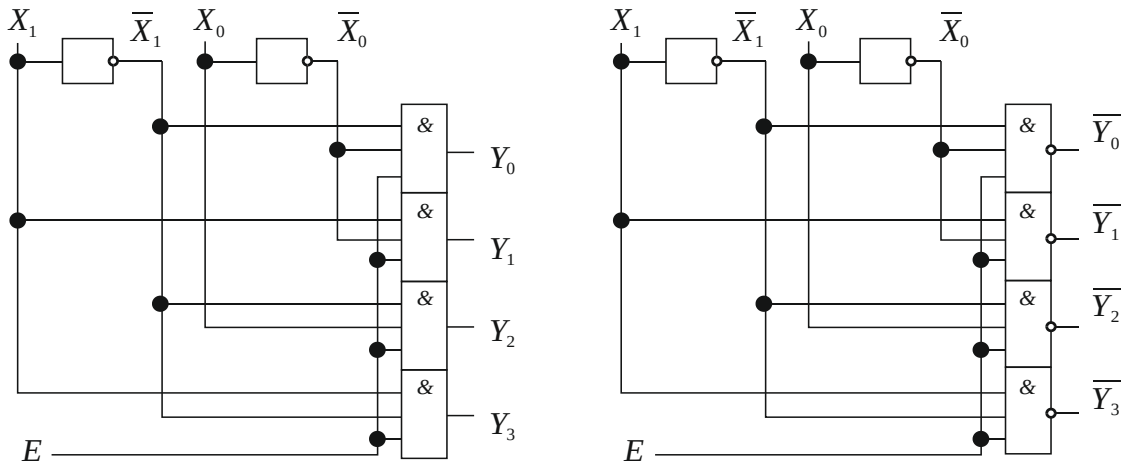


а) *б*)

Рис. 5.5 – Схемы двухразрядных дешифраторов:
с прямыми выходами (*а*); с инверсными выходами (*б*)

Введем еще одну переменную E, на которую умножим булеву функцию каждого выхода DC с прямыми выходами:

$$Y_0 = E\overline{X_0}\,\overline{X_1}, \quad Y_1 = EX_0\overline{X_1}, \quad Y_2 = E\overline{X_0}X_1, \quad Y_3 = X_0X_1.$$



а) *б*)

Рис. 5.6 – Схемы управляемых дешифраторов:
с прямыми выходами (*а*); с инверсными выходами (*б*)

При $E = 0$ все выходы $Y_0 = Y_1 = Y_2 = Y_3 = 0$, а при $E = 1$ функции выходов принимают значения в зависимости от входного кода. Вход E называют управляющим входом или входом стробирования. При $E = 0$ дешифратор «выключается», т. е. все выходы неактивны, а при $E = 1$ «включается» в работу. Такие

дешифраторы называют управляемыми. Схемы управляемых дешифраторов представлены на рисунке 5.6.

В цифровых устройствах узлы строятся так, чтобы на основе устройств с небольшим количеством разрядов можно было строить многоразрядные устройства. Например, трехразрядный дешифратор можно построить на основе двухразрядных.

Задача 5.1. Построить дешифратор трехразрядного двоичного кода на двухразрядных управляемых дешифраторах.

Для решения задачи сначала рассмотрим таблицу истинности трехразрядного полного неуправляемого дешифратора (табл. 5.1).

Таблица 5.1 – Таблица истинности трехразрядного дешифратора

	X_1	X_2	X_3	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Запишем систему булевых функций выходов дешифратора:

$$Y_0 = \overline{X_1}\,\overline{X_2}\,\overline{X_3},\quad Y_1 = \overline{X_1}\,\overline{X_2}X_3,\quad Y_2 = \overline{X_1}X_2\,\overline{X_3},\quad Y_3 = \overline{X_1}X_2X_3,$$

$$Y_4 = X_1\,\overline{X_2}\,\overline{X_3},\quad Y_5 = \overline{X_1}X_2\,\overline{X_3},\quad Y_6 = X_1X_2\,\overline{X_3},\quad Y_7 = X_1X_2X_3.$$

Раскрасим таблицу для большего удобства.

Таблица 5.2 – Таблица истинности трехразрядного дешифратора

	X_1	X_2	X_3	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Анализ таблицы 5.2 показывает, что переменные X_2 и X_3 в верхней и нижней части таблицы принимают одни и те же значения. Отсюда можно сделать вывод, что если взять два двухразрядных управляемых дешифратора, то из них

можно построить один трехразрядный дешифратор. Для этого на адресные входы A_0 и A_1 двухразрядных полных управляемых дешифраторов (рис. 5.7) следует подать переменные X_3 и X_2 соответственно. При поступлении на входы одинаковых комбинаций X_2 и X_3 у обоих дешифраторов будут возбуждаться одни и те же выходы.

Теперь следует разделить работу дешифраторов так, чтобы один реализовывал верхнюю половину таблицы (табл. 5.2), а другой – нижнюю. Если на управляющий вход дешифратора $DC1$ подать переменную X_1 через инвертор, а на вход $DC2$ прямое значение переменной X_1, то

- при $X_1 = 0$ $DC2$ отключится, т. е. $Y_4 = Y_5 = Y_6 = Y_7 = 0$, а выходы $DC1$ будут возбуждаться в соответствие с кодами X_2, X_3;
- при $X_1 = 0$, наоборот, отключится $DC1$ $Y_0 = Y_1 = Y_2 = Y_3 = 0$, $DC2$ будет работать в соответствие с кодами X_2, X_3. Таким образом, будет реализована таблица истинности (табл. 5.2).

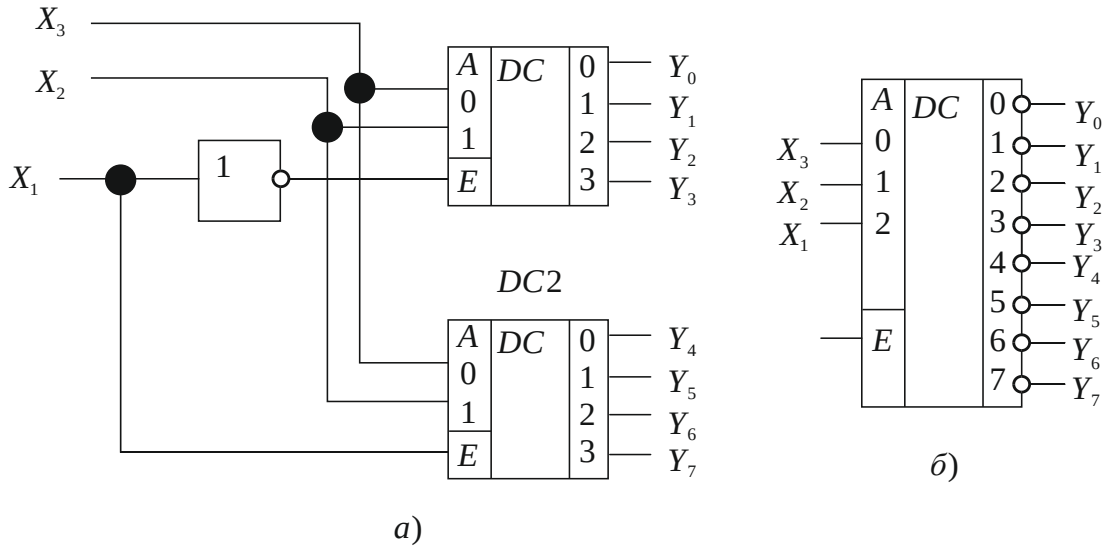


Рис. 5.7 – Схема и обозначение трехразрядного дешифратора:
схема трехразрядного дешифратора,
собранного из двухразрядных дешифраторов (*а*);
обозначение трехразрядного дешифратора на схемах (*б*)

На рисунке 5.8 показана диаграмма входных и выходных сигналов трехразрядного дешифратора. На входы дешифратора подаются сигналы, соответствующие переменным логических сигналов *a*, *b*, *c*. Источник формирует сигналы периодически и создает последовательно все комбинации двоичного кода от 000 до 111. Входы дешифратора и его выходы подключаются к многоканальному осциллографу, который разворачивает появление сигналов во времени. На рисунке сделано несколько отметок (вертикальные линии), отмечающих

код на входе дешифратора и его выходы. Так, в первом периоде отмечена комбинация переменных *abc* = 011, на которую только выход Y_3 откликается высоким уровнем выходного сигнала ($Y_3 = 1$). Все остальные выходы в это время остаются равными нулю, что соответствует таблице истинности. Во втором периоде отмечается комбинация *abc* = 001, на которую откликается выход Y_1, и комбинация *abc* = 101 с откликом на Y_5.

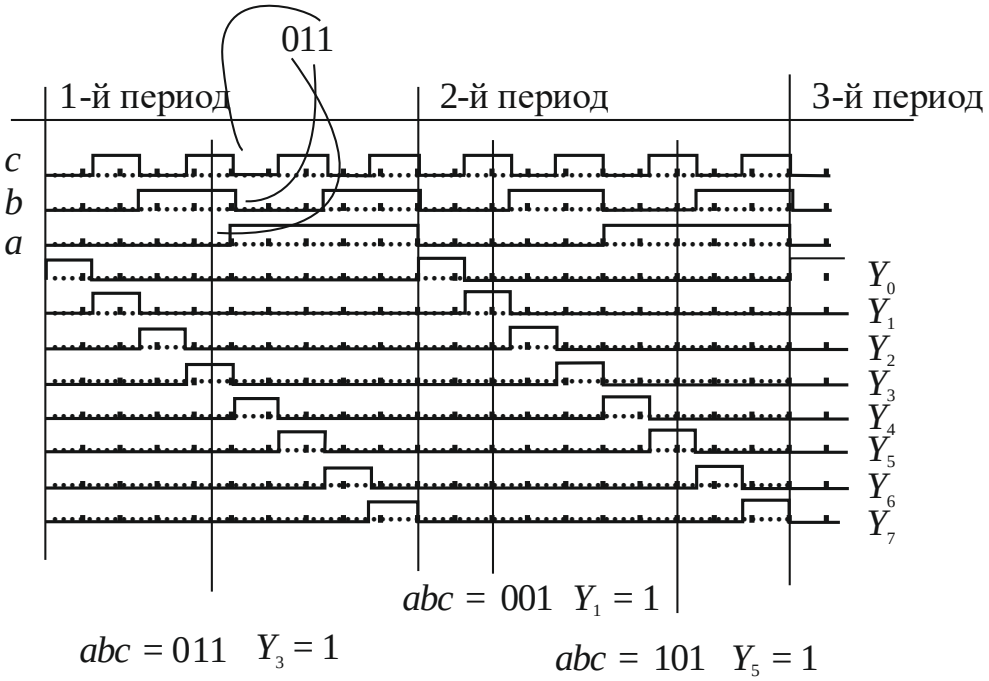


Рис. 5.8 – Диаграмма работы трехразрядного дешифратора с прямыми выходами

На рисунке 5.9 показана диаграмма работы трехразрядного дешифратора с инверсными выходами. У такого дешифратора активный выходной сигнал принимает нулевое значение, в то время как остальные выходы равны единице.

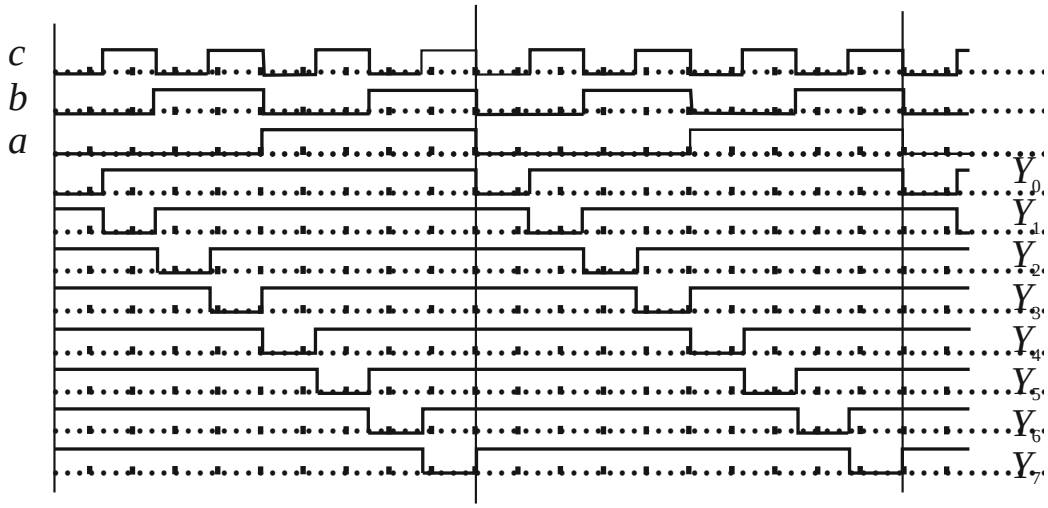


Рис. 5.9 – Диаграмма работы трехразрядного дешифратора
с инверсными выходами

Задача 5.2. Построить схему четырехразрядного дешифратора двоичного кода на дешифраторах меньшей разрядности. Для решения задачи с помощью двухразрядных дешифраторов покажем таблицу истинности (табл. 5.3).

Таблица 5.3 – Таблица истинности четырехразрядного дешифратора

	*DC*1					
	X_1	X_2	X_3	X_4		
0	0	0	0	0	*DC*2	Y_0
1	0	0	0	1		Y_1
2	0	0	1	0		Y_2
3	0	0	1	1		Y_3
4	0	1	0	0	*DC*3	Y_4
5	0	1	0	1		Y_5
6	0	1	1	0		Y_6
7	0	1	1	1		Y_7
8	1	0	0	0	*DC*4	Y_8
9	1	0	0	1		Y_9
10	1	0	1	0		Y_{10}
11	1	0	1	1		Y_{11}
12	1	1	0	0	*DC*5	Y_{12}
13	1	1	0	1		Y_{13}
14	1	1	1	0		Y_{14}
15	1	1	1	1		Y_{15}

Из таблицы видно, что для переменных X_3 и X_4 таблицы совпадают. Переменные X_1 и X_2 принимают одинаковые значения в каждой группе из четырех клеток. Если взять двухвходовый дешифратор и на адресные входы подать X_1 и X_2, то четыре его выхода реализуют часть таблицы, обозначенную *DC*1, как показано на рисунке 5.10. Для реализации реакции на переменные X_3 и X_4 потребуется четыре двухвходовых управляемых дешифратора. На их адресные входы подаются одни и те же переменные X_3, X_4, а на входы управления подключается один из выходов *DC*1. Таким образом, данная схема реализует таблицу 5.3. Для работы *DC*1 на его вход *E* необходимо подать постоянный сигнал, равный логической единице.

Такие схемы называется двухступенчатыми. В них первая ступень реализует первую часть таблицы истинности, а вторая – следующую.

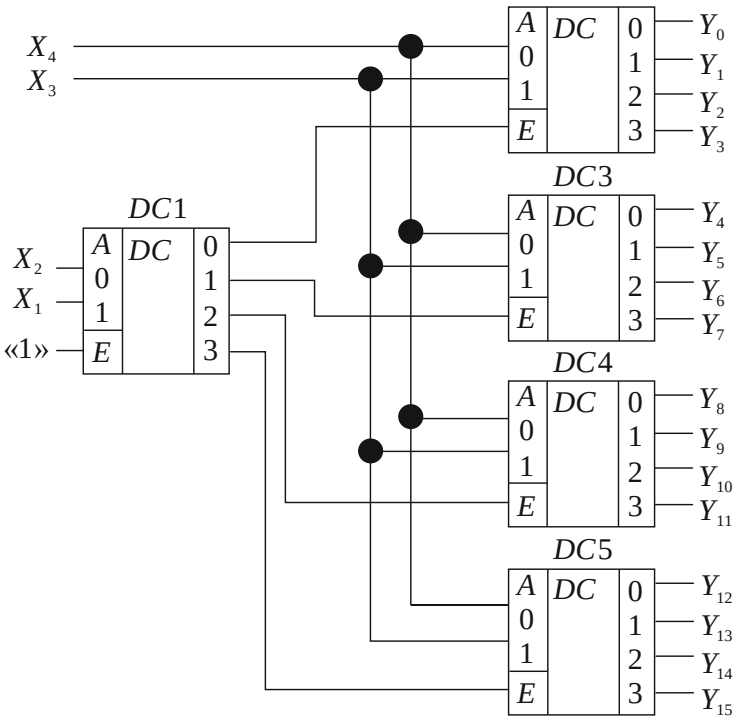


Рис. 5.10 – Четырехразрядный дешифратор,
построенный на двухразрядных дешифраторах

Эту задачу можно решить и с использованием трехразрядных дешифраторов (рис. 5.11). Таблицу истинности покажем немного по-другому (табл. 5.4). Переменная $X_1 = 0$ включает $DC1$ и отключает $DC2$ (все его выходы равны нулю). При $X_1 = 1$ инвертор отключает $DC1$ и включает в работу $DC2$. Работающий дешифратор формирует выходной сигнал, равный единице, только на одном выходе в зависимости от того, какой входной код поступает на адресные входы X_2, X_3, X_4.

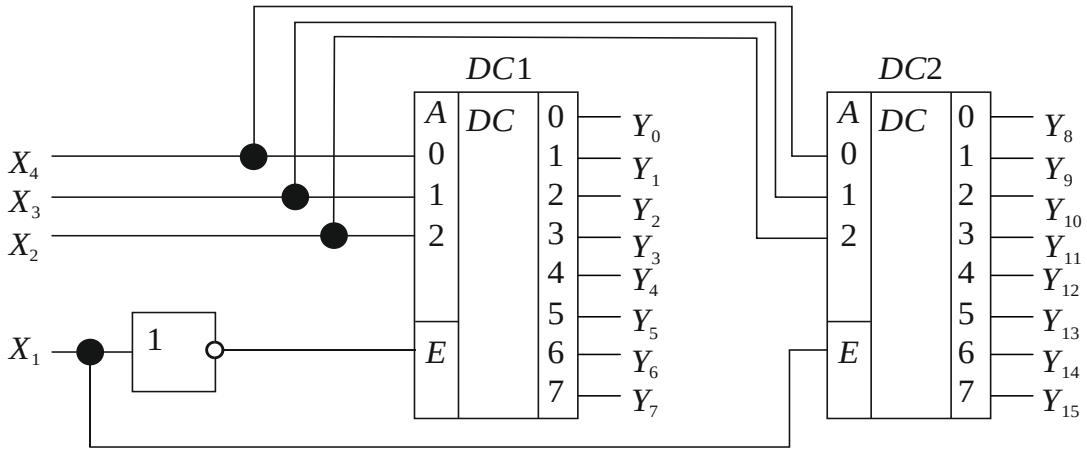


Рис. 5.11 – Четырехразрядный дешифратор,
построенный на трехразрядных дешифраторах

Таблица 5.4 – Таблица истинности
четырехразрядного дешифратора с прямыми выходами

	X_1	X_2	X_3	X_4	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
12	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
14	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

5.2 Шифраторы

Шифратор (кодер) – это устройство (обратное дешифратору), преобразующее унитарный n-разрядный входной код в заданную двоичную m-разрядную кодовую комбинацию $(m \le n)$. Различные шифраторы для одного и того же входного кода могут выдать различные выходные кодовые комбинации, т. е. зашифровывать символы. Шифраторы используются во многих устройствах, например, в различных клавиатурах, в которых при нажатии любой одной клавиши должен быть сформирован соответствующий ей двоичный код. Шифраторы синтезируются по таблице истинности, которая задает систему из m булевых функций.

Задача 5.3. Синтезировать схему шифратора, работающего по следующему закону:

- шифратор имеет пять входов, на которые поступают логические сигналы от клавиш клавиатуры;
- шифратор имеет три выхода, на которых образуется двоичный код в соответствие с таблицей истинности (табл. 5.5);
- в любой момент времени только один из входных сигналов может иметь активный единичный уровень;

- схему шифратора выполнить на элементах И-НЕ.

Таблица 5.5 – Таблица истинности шифратора

№ входа	Логическая переменная	Значения выходных сигналов		
		Y_3	Y_2	Y_1
0	$X_1 = 1$	1	0	1
1	$X_2 = 1$	0	1	0
2	$X_3 = 1$	1	0	0
3	$X_4 = 1$	0	1	1
4	$X_5 = 1$	1	1	1
5	$X_1 = X_2 = X_3 = X_4 = X_5 = 0$	0	0	0

Составим БФ выходов и преобразуем их к базису И-НЕ:

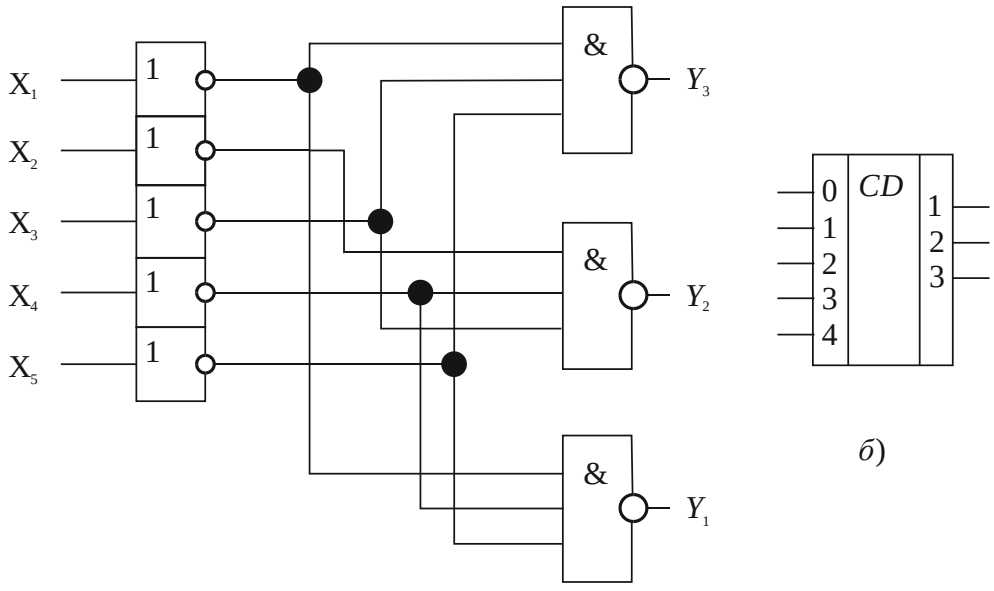
$$Y_1 = X_1 + X_4 + X_5, \quad Y_2 = X_2 + X_4 + X_5, \quad Y_3 = X_1 + X_3 + X_5;$$

$$Y_1 = \overline{\overline{X_1 + X_4 + X_5}} = \overline{\overline{X_1}\,\overline{X_2}\,\overline{X_3}};$$

$$Y_2 = \overline{\overline{X_2 + X_4 + X_5}} = \overline{\overline{X_2}\,\overline{X_4}\,\overline{X_5}};$$

$$Y_3 = \overline{\overline{X_1 + X_3 + X_5}} = \overline{\overline{X_1}\,\overline{X_3}\,\overline{X_5}}.$$

На основе этих функций построим комбинационную схему устройства (рис. 5.12, *а*).



а) *б*)

Рис. 5.12 – Шифратор: схема шифратора (*а*);
обозначение шифратора на схеме (*б*)

5.3 Преобразователи кодов

Преобразователь кода – это устройство, которое преобразует *n*-разрядный двоичный код в *m*-разрядный двоичный код. Вполне очевидно, что простейший преобразователь кода можно построить, последовательно соединив дешифратор и шифратор. Дешифратор преобразует входной *n*-разрядный двоичный код в унитарный код разрядностью 2^n, а шифратор, имеющий 2^n входов, преобразует его в *m*-разрядный двоичный выходной код. Другой способ построения преобразователя кода состоит в построении таблицы истинности, в которой устанавливается соответствие между входным и выходным кодами, т. е. задается система *m* булевых функций от *n* переменных. И наконец, для преобразователя кода можно использовать постоянные запоминающие устройства или программируемые логические матрицы.

Задача 5.4. Синтезировать преобразователь трехразрядного двоичного кода в код Грея. Код Грея – это двоичный код, у которого любые две соседние комбинации имеют отличие только в одном разряде.

Этот код широко используется в различных системах автоматики. Кстати, он используется при разметке карт Карно. Для получения кода Грея разработано несколько способов. Например, находится сумма по модулю 2 ($f = a\bar{b} + \bar{a}b$) двоичного числа с ним самим, сдвинутым на один разряд вправо (крайний справа разряд сдвинутого числа отсекается) (рис. 5.13).

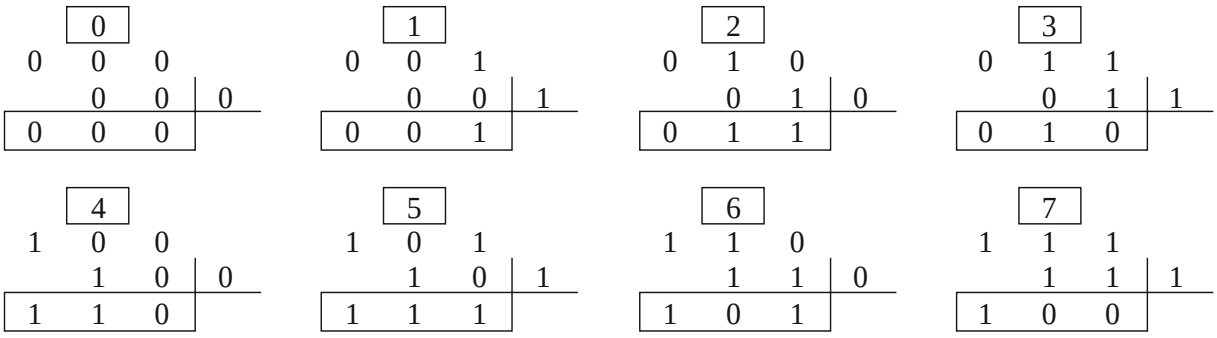


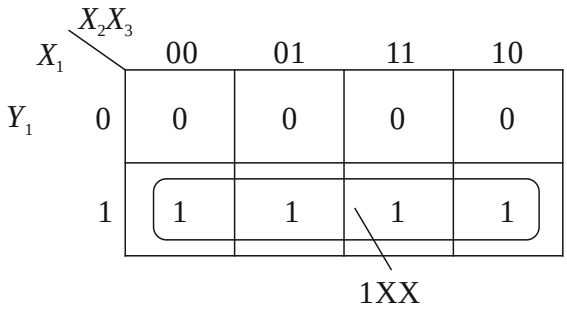
Рис. 5.13 – Преобразование двоичного кода в код Грея

Составим таблицу истинности для преобразователя, нанесем функции выходов на карты Карно (рис. 5.14). После минимизации функции будут иметь вид:

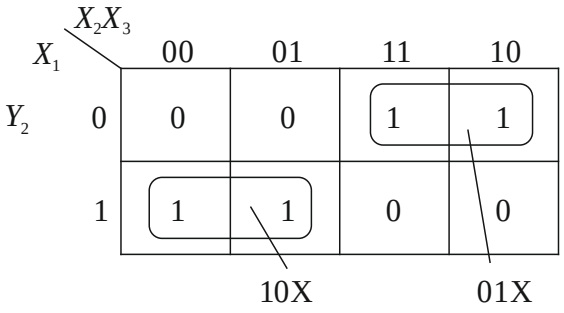
$$Y_1 = X_2, \quad Y_2 = X_1\overline{X_2} + \overline{X_1}X_2, \quad Y_3 = X_2\overline{X_3} + \overline{X_2}X_3.$$

	Двоичный код	Код Грея
	$X_1X_2X_3$	$Y_1Y_2Y_3$
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

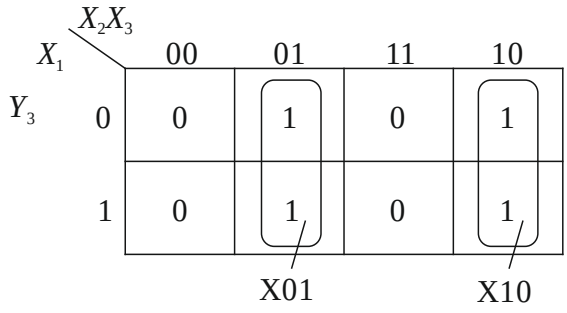
а)



б)



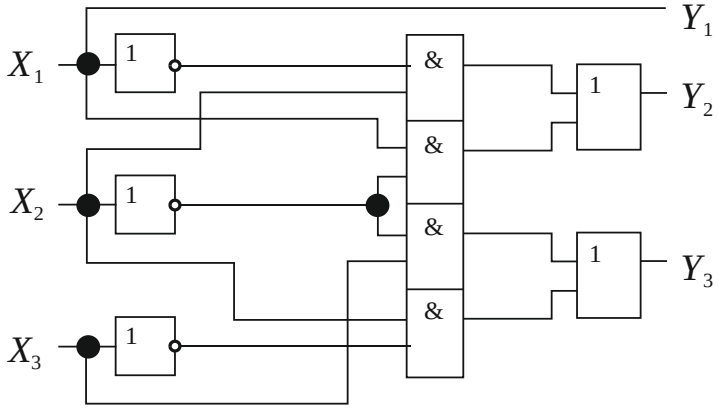
в)



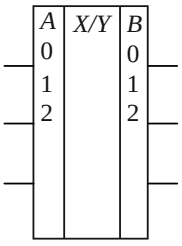
г)

Рис. 5.14 – Таблица истинности и карты Карно для преобразователя кодов:
таблица истинности преобразователя двоичного кода в код Грея (*а*);
карта Карно функции Y_1 (*б*); карта Карно функции Y_2 (*в*);
карта Карно функции Y_3 (*г*)

Схема преобразователя приведена на рисунке 5.15, *а*. На рисунке 5.15, *б* показано обозначение преобразователей кодов на схемах.



а)



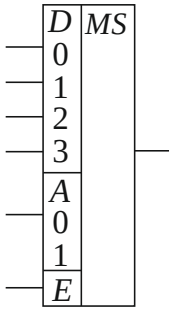
б)

Рис. 5.15 – Преобразователь в код Грея:
схема преобразователя (*а*);
обозначение преобразователей кода на схемах (*б*)

5.4 Мультиплексоры

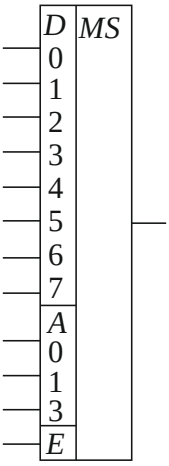
Мультиплексором (коммутатором) называется устройство, имеющее n адресных входов, 2^n входов данных (информационных) и один выход и выполняющее передачу данных (переключение, коммутацию) с одного из информационных входов на выход, в зависимости от двоичного кода, поданного на адресные входы. То есть, указав адрес источника сигнала, можно передать его значение на единственный выход мультиплексора. Количество информационных входов может быть равно 4, 8, 16, и тогда мультиплексоры называют 4 на 1, 8 на 1 или 16 на 1 соответственно. Мультиплексоры могут иметь управляющий или стробирующий вход. Таблицы истинности и обозначение мультиплексоров приведены на рисунке 5.16.

E	A_1	A_0	f
0	X	X	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3



а)

E	A_2	A_1	A_0	f
0	X	X	X	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7



б)

Рис. 5.16 – Таблицы истинности и обозначение управляемых мультиплексоров: мультиплексор 4 на 1 (*а*); мультиплексор 8 на 1 (*б*)

В мультиплексорах имеется поле адреса (A), на входы которого подается двоичный код. Значение этого кода определяет, какой из информационных входов поля D переключается на выход f. Поле E включает в себя один или несколько управляющих сигналов. В таблице истинности мультиплексоров при $E = 0$ выходной сигнал f всегда равен нулю, поэтому значения переменных A_0, A_1 и A_2 безразличны (обозначены X).

Таким образом, функцию выхода мультиплексора 4 на 1 можно записать в следующем виде:

$$f = E \cdot D_0 \cdot \overline{A_0} \cdot \overline{A_1} + E \cdot D_1 \cdot A_0 \cdot \overline{A_1} + E \cdot D_2 \cdot \overline{A_0} \cdot A_1 + E \cdot D_3 \cdot A_0 \cdot A_1.$$

Если вынести E за скобку, то получим такую формулу:

$$f = E \cdot (D_0 \cdot \overline{A_0} \cdot \overline{A_1} + D_1 \cdot A_0 \cdot \overline{A_1} + D_2 \cdot \overline{A_0} \cdot A_1 + D_3 \cdot A_0 \cdot A_1).$$

При $E = 1$ и комбинации в поле адреса $A = 00$ на выход переключается информационный вход D_0, при $A = 01$ – вход D_1, при $A = 10$ – вход D_2, при $A = 11$ – вход D_5.

Из анализа формулы следует, что функция f мультиплексора – это ДНФ. Комбинации переменных A_0 и A_1 в формуле повторяют конъюнкции выходов двухразрядного дешифратора с прямыми выходами. Конъюнкция дополнительно умножается на E и на значение D_k соответствующего информационного входа $k = 0, 1, 2, 3$. Тогда функцию мультиплексора можно представить так, как на рисунке 5.17.

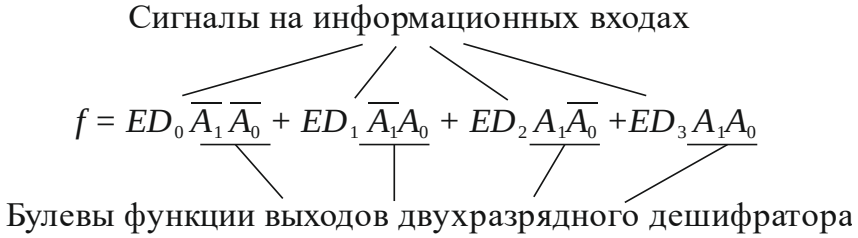


Рис. 5.17 – БФ мультиплексора 4 на 1

Таким образом, мультиплексор состоит из дешифратора для выбора информационного входа, набора элементов И для реализации конъюнкций и элемента ИЛИ, который объединяет выходы элементов И. Схема мультиплексора 4 на 1 представлена на рисунке 5.18.

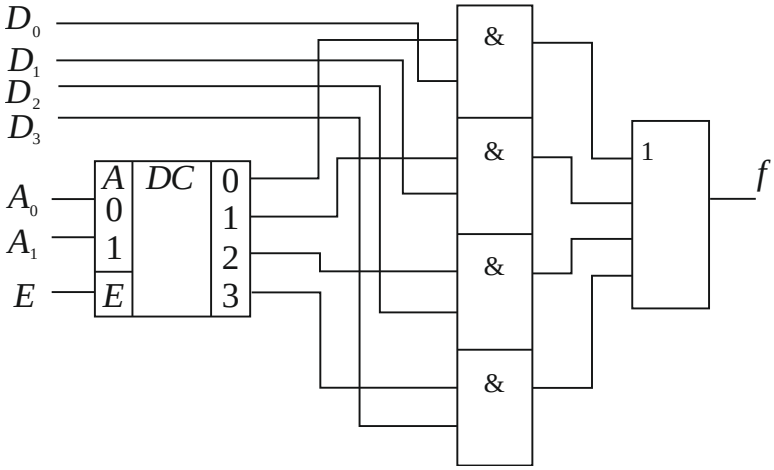


Рис. 5.18 – Схема управляемого мультиплексора 4 на 1

Для неуправляемого мультиплексоров 8 на 1 БФ выхода имеет вид:

$$f = D_0 \cdot \overline{A_2} \cdot \overline{A_1} \cdot \overline{A_0} + D_1 \cdot \overline{A_2} \cdot \overline{A_1} \cdot A_0 + D_2 \cdot \overline{A_2} \cdot A_1 \cdot \overline{A_0} + D_3 \cdot \overline{A_2} \cdot A_1 \cdot A_0 +$$
$$+ D_4 \cdot A_2 \cdot \overline{A_1} \cdot \overline{A_0} + D_5 \cdot \overline{A_2} \cdot A_1 \cdot \overline{A_0} + D_6 \cdot A_2 \cdot A_1 \cdot \overline{A_0} + D_7 \cdot A_2 \cdot A_1 \cdot A_0.$$

Схема мультиплексора представлена на рисунке 5.19.

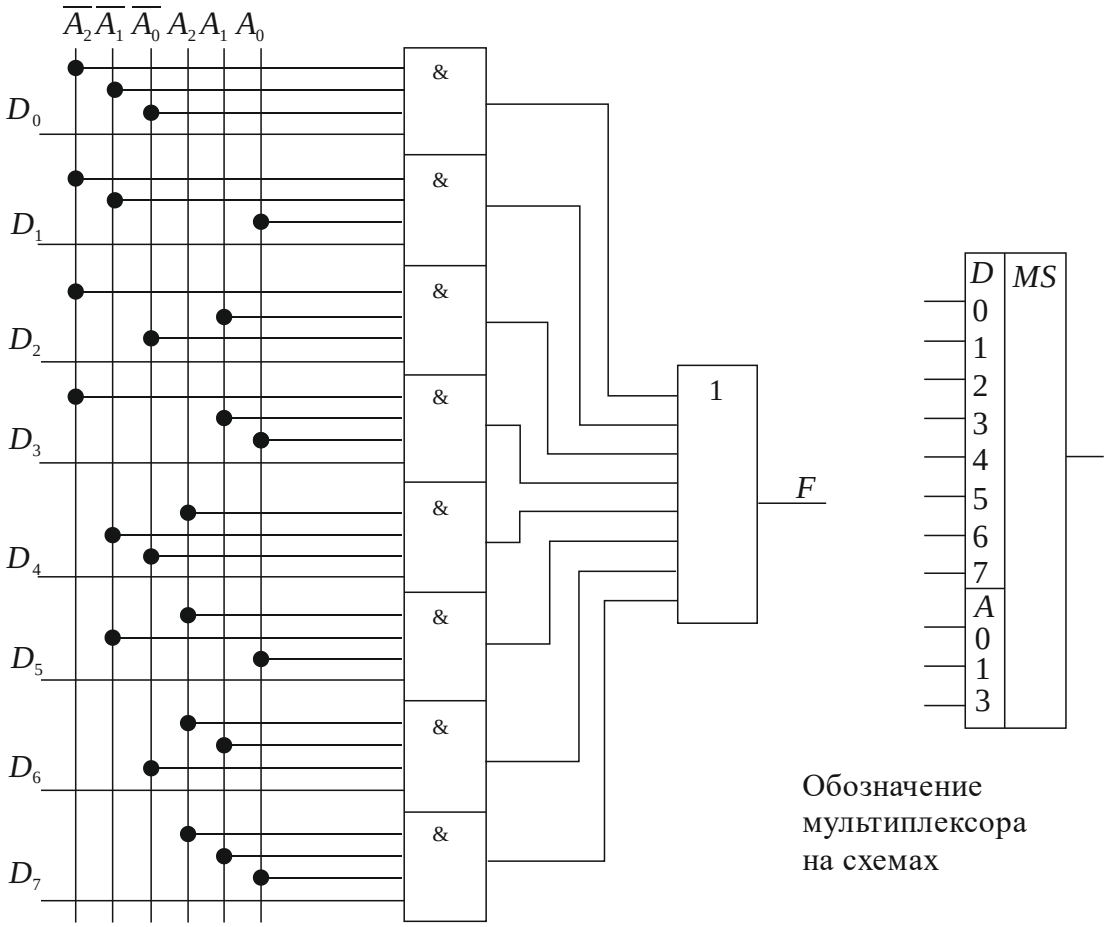


Схема мультиплексора 8 на 1

Рис. 5.19 – Схема управляемого мультиплексора 8 на 1 и его обозначения на схемах

При комбинации на адресных входах $A = 000$ на выход переключается информационный вход D_0, при $A = 001$ – вход D_1, при $A = 110$ – вход D_6, при $A = 111$ – вход D_7 и т. д. В мультиплексор 8 на 1 входит трехразрядный дешифратор, собранный на 8 схемах И, схема ИЛИ на 8 входов.

Эти мультиплексоры предназначены для передачи одноразрядных сигналов. Для передачи двоичных четырехразрядных кодов с двух направлений на одно используются мультиплексоры, построенные по другим схемам.

Пример такого мультиплексора приведен на рисунке 5.20.

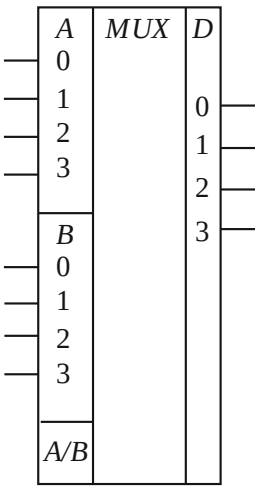
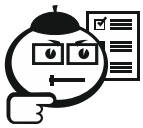


Рис. 5.20 – Мультиплексор для передачи четырехразрядных кодов с двух направлений на одно

На входы полей $A_0 \ldots A_3$ и $B_0 \ldots B_3$ подаются четырехразрядные двоичные коды. Сигнал на входе управления A / B переключает на выход коды с одного из направлений A или B.

Используя управляемые мультиплексоры меньшей разрядности, можно строить мультиплексоры большей разрядности. На рисунке 5.21 представлены схемы построения мультиплексора 16 на 1 из мультиплексоров 4 на 1 и мультиплексоров 8 на 1.



Самостоятельно разберитесь с работой этих схем.

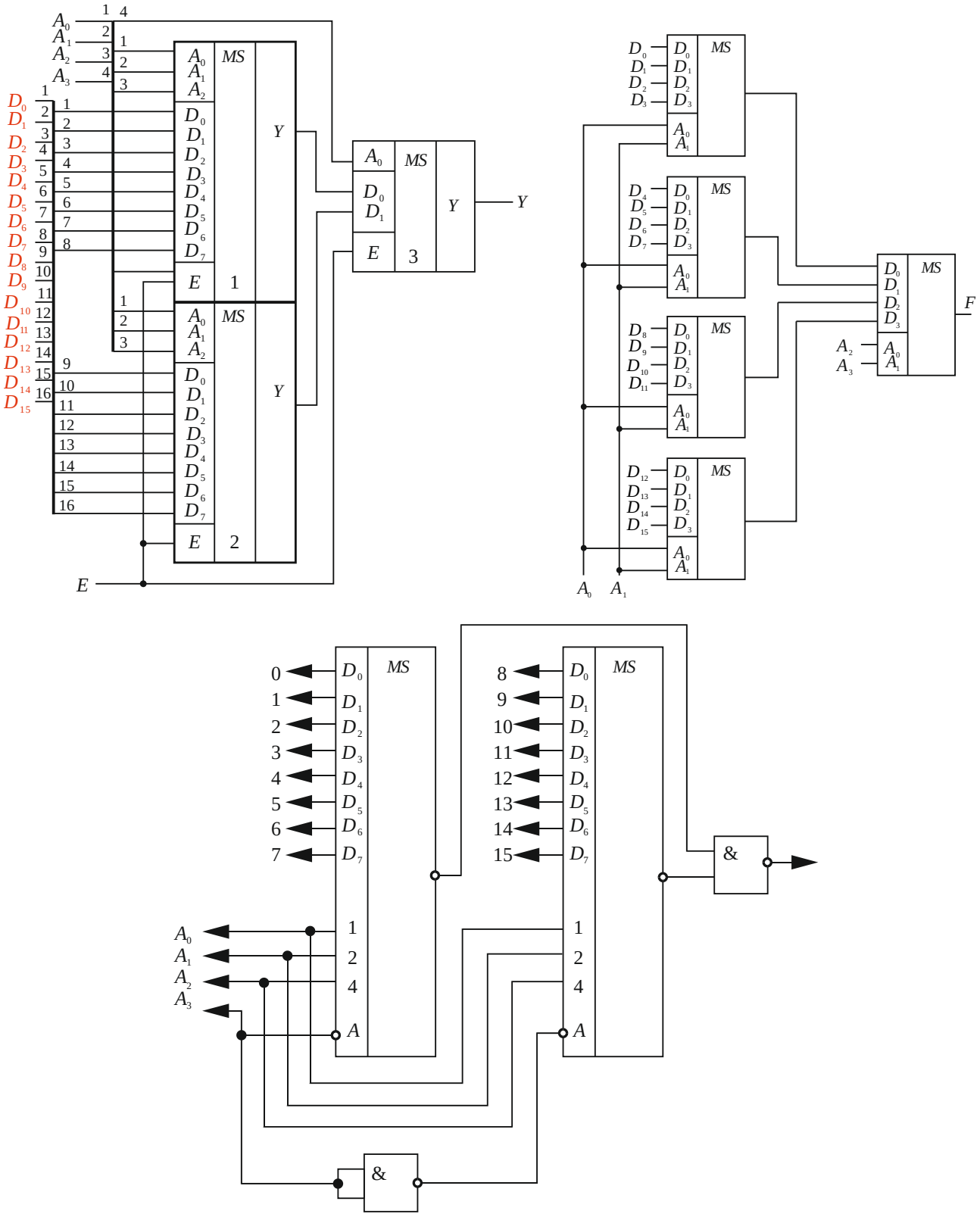


Рис. 5.21 – Схемы построения мультиплексоров 16 на 1

На рисунке 5.22 приведены обозначения на схемах различных мультиплексоров.

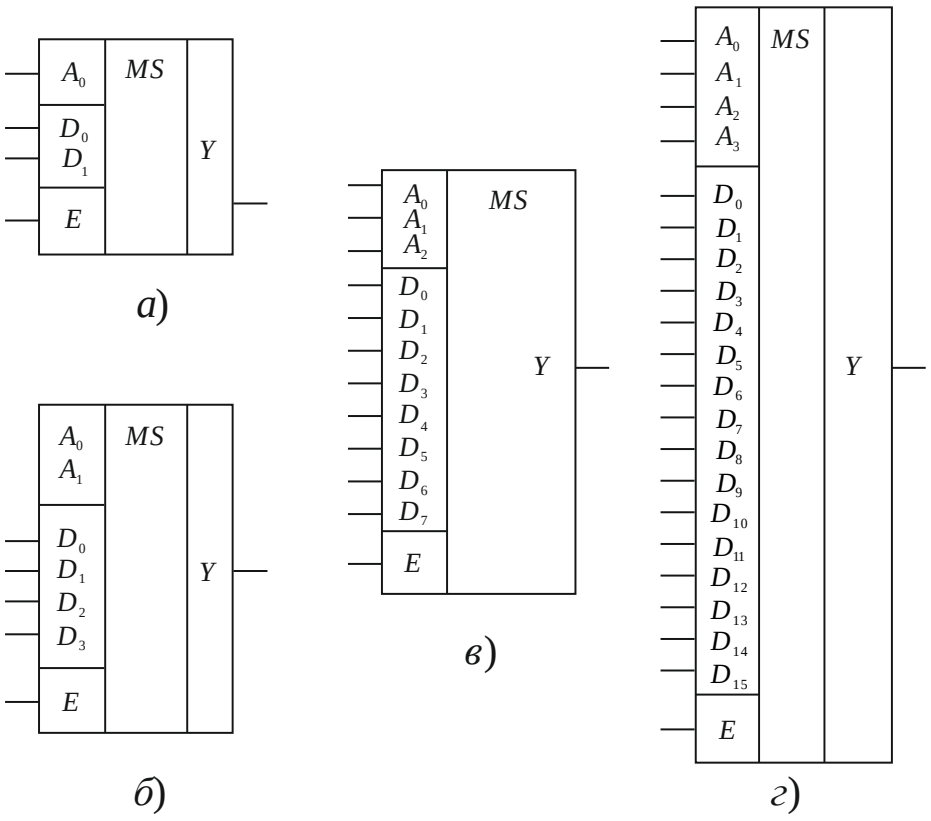


Рис. 5.22 – Обозначения мультиплексоров на схемах: мультиплексор 2 на 1 (*а*); мультиплексор 4 на 1 (*б*); мультиплексор 8 на 1 (*в*); мультиплексор 16 на 1 (*г*)

5.5 Демультиплексоры

Демультиплексор – это устройство, обратное мультиплексору, т. е. единственный входной сигнал коммутируется (передается) на одно из нескольких направлений, в зависимости от кода на адресных входах. Обозначение демультиплексоров на схемах приведено на рисунке 5.23.

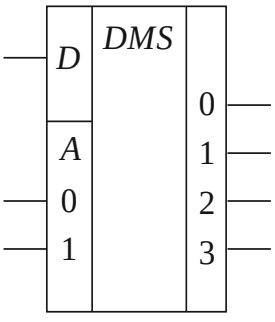


Рис. 5.23 – Демультиплексор 1 на 4

По аналогии с мультиплексорами демультиплексоры можно называть 1 на 4, 1 на 8 и т. д. Таблица истинности демультиплексора 1 на 4, БФ его выходов и схема приведены на рисунке 5.24.

A_1	A_0	f_0	f_1	f_2	f_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

а)

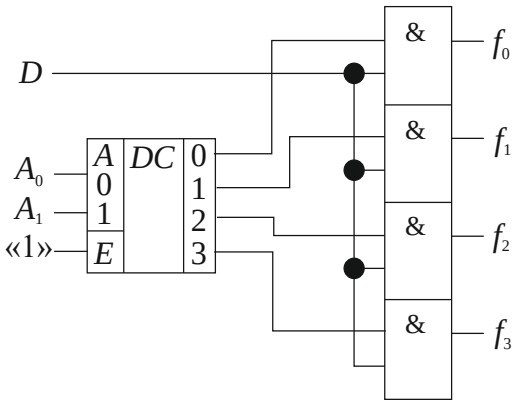
$f_0 = D\overline{A_1}\,\overline{A_0}$

$f_1 = D\overline{A_1}A_0$

$f_2 = DA_1\overline{A_0}$

$f_3 = DA_1A_0$

б)

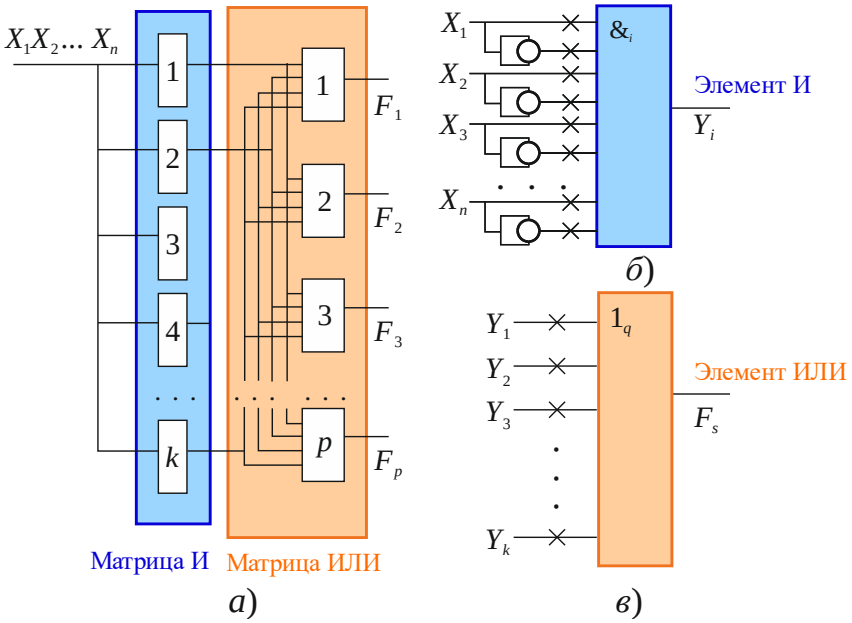


в)

Рис. 5.24 – Демультиплексор 1 на 4:
таблица истинности (*а*); БФ выходов (*б*); схема демультиплексора (*в*)

5.6 Программируемые логические матрицы

Программируемая логическая матрица (ПЛМ, PLM, PAL) – это регулярная программируемая полупроводниковая структура, предназначенная для реализации систем булевых функций от n переменных, заданных в ДНФ. ПЛМ содержит матрицу И и матрицу ИЛИ. Матрица И содержит k схем И, каждая из которых имеет 2^n входов. Они подключены параллельно ко всем элементам И. Матрица ИЛИ содержит p k-входовых схем ИЛИ, также подключенных параллельно ко всем элементам ИЛИ. В таком случае ПЛМ называют матрицей «$k \times p$ для n переменных». Ее структура приведена на рисунке 5.25.



а) *б*) *в*)

Рис. 5.25 – Структура программируемой логической матрицы:
общая структура (*а*); элемент И (*б*); элемент ИЛИ (*в*)

Каждый из k элементов И может реализовать свою конъюнкцию Y от n переменных, причем с помощью встроенных элементов НЕ в конъюнкцию входит прямое и инверсное значение каждой переменной. Выходы всех элементов И поступают на входы каждого элемента ИЛИ. Элемент схемы, обозначенный крестиком (Х), позволяет разорвать свою цепь. Таким образом любой элемент И может реализовывать только свою конъюнкцию от нескольких аргументов, отключая все ненужные входные цепи.

Выходы всех элементов И подключены к входам всех элементов ИЛИ. Те конъюнкции, которые не нужны для реализации БФ, формируемой именно этим элементом ИЛИ, исключаются разрывом цепи с помощью элемента, обозначенного Х. Такая ПЛМ может реализовать P булевых функций (F) от числа аргументов не более n. Во многих ПЛМ реализован инверсный выход.

Программирование ПЛМ состоит в отключении ненужных связей в матрицах И и ИЛИ. ПЛМ могут иметь и инверсные выходы.

На рисунке 5.26 представлена схема незапрограммированной диодной (диод обозначен крестиком) ПЛМ для реализации четырех БФ от пяти переменных. ПЛМ содержит восемь схем И (одна из них обведена овалом) и четыре схемы ИЛИ (одна из них обведена овалом). Каждая из схем И реализует БФ:

$$Y = X_1 \cdot \overline{X_1} \cdot X_2 \cdot \overline{X_2} \cdot X_3 \cdot \overline{X_3} \cdot X_4 \cdot \overline{X_4} \cdot X_4 \cdot \overline{X_4} \cdot X_5 \cdot \overline{X_5}.$$

Очевидно, что такое произведение тождественно равно нулю. Если нужно получить конъюнкцию $Y = \overline{X_1} X_2 \overline{X_3}$, то следует исключить переменные $X_1 \overline{X_2} X_3 X_4 \overline{X_4} X_5 \overline{X_5}$. Это производится программированием на устройстве, называемом программатором, с помощью элементов, обозначенных Х. Элемент, обозначенный Х, называется перемычкой, которая и уничтожается программатором. При этом разрывается цепь, т. е. электрический сигнал не входит в логический элемент.

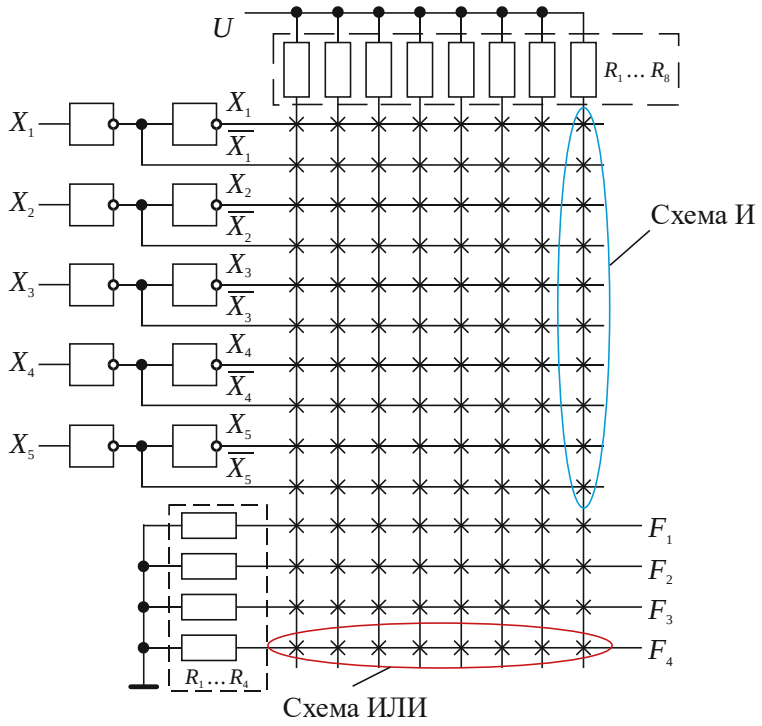


Рис. 5.26 – Пример структуры ПЛМ

На рисунке 5.27 показан фрагмент одной схемы И матрицы до программирования конъюнкции и после него.

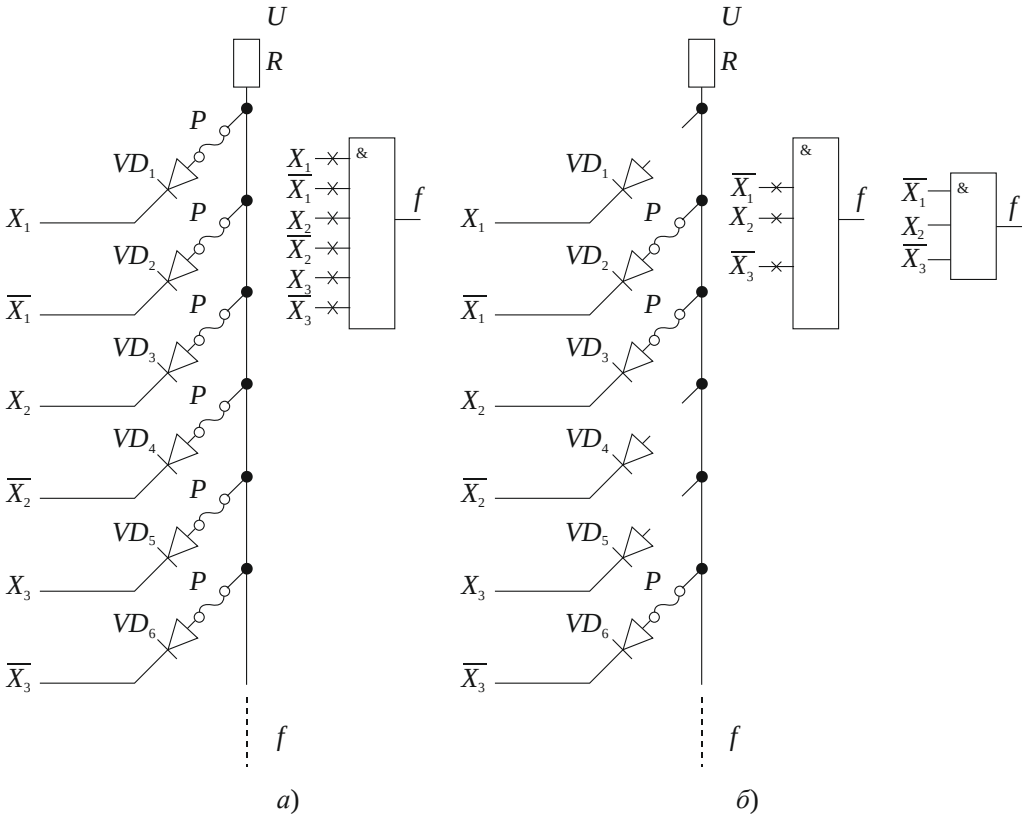


Рис. 5.27 – Фрагмент одного элемента И ПЛМ:
полный (незапрограммированный) элемент И (*а*);
запрограммированный элемент И на три входа (*б*)

На рисунке 5.28 приведен фрагмент схемы ИЛИ, входящей в состав ПЛМ. Схема имеет четыре входа. Входным сигналом для матрицы ИЛИ является выходной сигнал, пришедший с соответствующей шины f матрицы И. Элемент ИЛИ реализует дизъюнкцию $F = f_0 + f_1 + f_2 + f_3$.

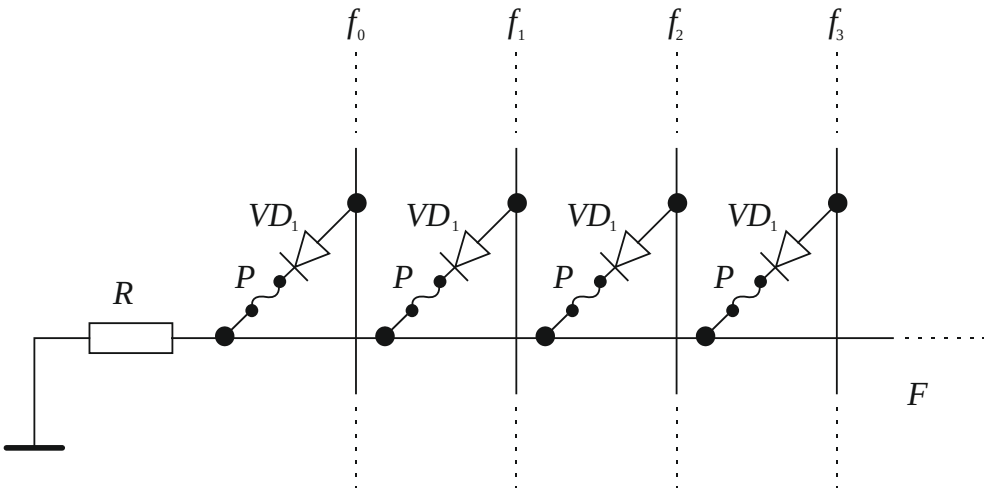


Рис. 5.28 – Фрагмент одного элемента ИЛИ ПЛМ

Если необходимо удалить какую-то переменную f из дизъюнкции, то программатор проводит удаление соответствующей перемычки.

Задача 5.5. Пусть задана система из четырех БФ:

$$Y_1 = X_1\overline{X_2}X_3\overline{X_5} + X_1X_2X_5 + \overline{X_2}\,\overline{X_3}\,\overline{X_4},$$

$$Y_2 = \overline{X_1}\,\overline{X_3}\,\overline{X_5} + X_1\overline{X_2}X_3\overline{X_5} + X_2X_3,$$

$$Y_3 = \overline{X_2}\,\overline{X_3}\,\overline{X_4} + \overline{X_1}X_2\overline{X_3}X_4X_5 + X_2X_3,$$

$$Y_4 = \overline{X_1}\,\overline{X_3}\,\overline{X_5} + X_1\overline{X_2}X_3\overline{X_5} + X_2X_3 + \overline{X_1}\,\overline{X_4}\,\overline{X_5}.$$

На рисунке 5.29 показана ПЛМ, запрограммированная на реализацию этой системы. В тех узлах схемы, в которых перемычки удалены, знаки «Х» отсутствуют (цепи разорваны). Обратите внимание на то, что одинаковые конъюнкции в разных функциях реализуются только один раз. Это дает экономию элементов в матрице И.

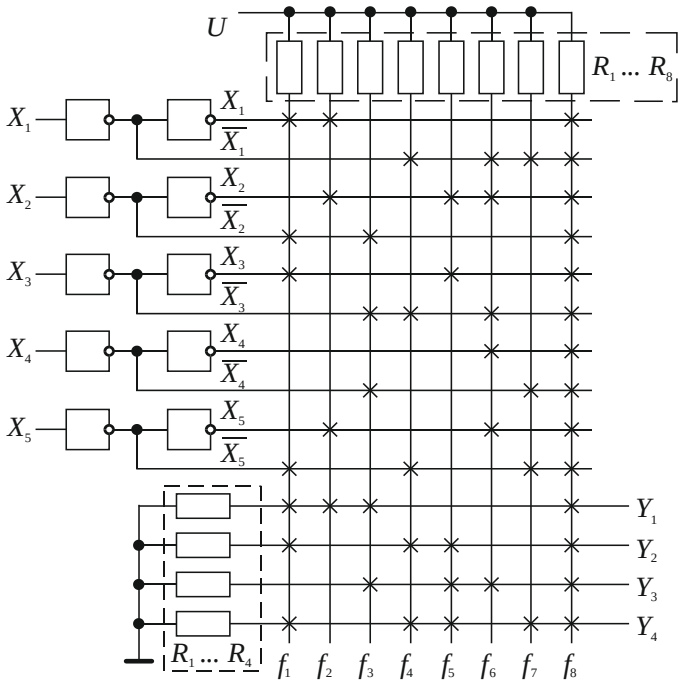


Рис. 5.29 – ПЛМ, запрограммированная на систему БФ

Задача 5.6. На ПЛМ для пяти переменных реализовать систему БФ:

$$Y_1 = X_1X_2\overline{X_3} + X_1X_2X_3 + X_1\overline{X_2}X_3,$$

$$Y_2 = X_1X_2X_3 + X_2\overline{X_4}\,\overline{X_5} + X_1X_5,$$

$$Y_3 = X_1X_2\overline{X_3} + X_1X_2X_3 + \overline{X_3}\,\overline{X_4}\,\overline{X_5},$$

$$Y_4 = X_1X_5 + \overline{X_3}\,\overline{X_4}\,\overline{X_5}.$$

На рисунке 5.30 представлена реализация этой системы БФ на ПЛМ (рисунок упрощен).

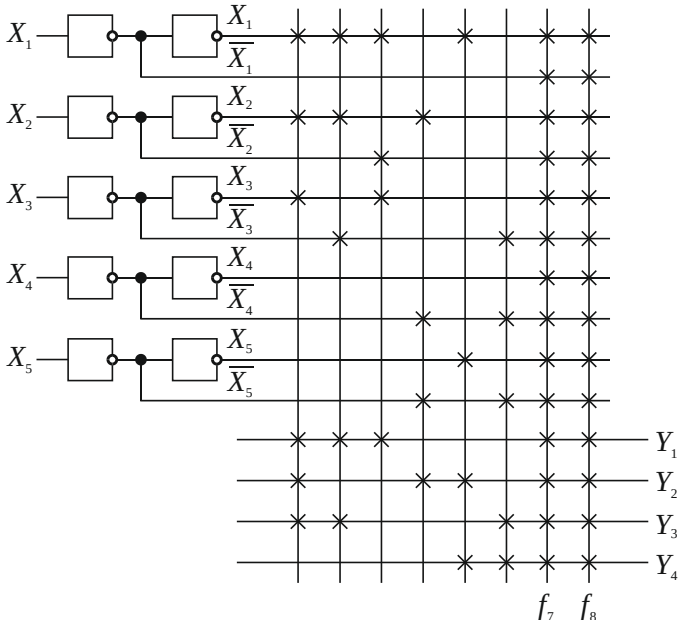


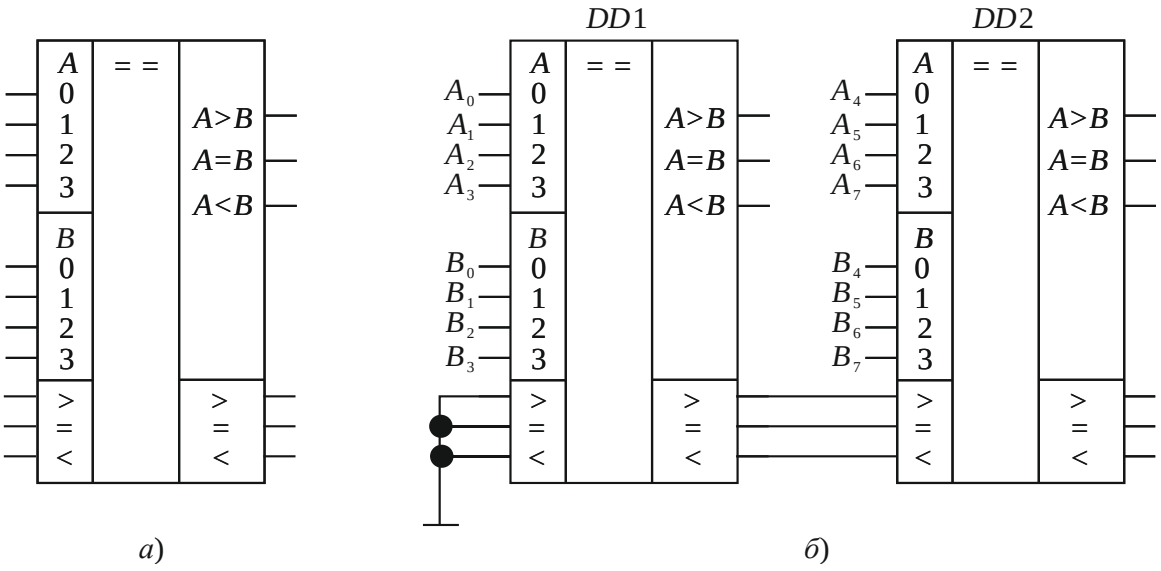
Рис. 5.30 – Реализация системы БФ на ПЛМ

Такие ПЛМ нашли применение в реализации схем микропроцессоров и в программируемых логических интегральных схемах (ПЛИС), на которых строятся сложные цифровые устройства.

5.7 Схемы сравнения

Схема сравнения, или числовой компаратор, предназначена для выяснения *отношения* между *кодами двух чисел*, то есть выясняется, равны ли числа, а если нет, то какое из них большее.

Обозначение схем сравнения на схемах показано на рисунке 5.31, *а*. На схему сравнения поступают четырехразрядные коды чисел *A* и *B*. Выходы схемы единичным сигналом показывают результат сравнения. Дополнительные входы и выходы, обозначенные как <, >, =, предназначены для возможности наращивания схемы в случае сравнения чисел с разрядностью больше, чем четыре. На рисунке 5.31, *б* показано соединение двух компараторов для сравнения восьмиразрядных кодов чисел.



а) *б*)

Рис. 5.31 – Схема сравнения:
обозначение схемы сравнения (*а*);
увеличение разрядности схемы сравнения (*б*)

5.8 Сумматоры

Сумматор – это устройство для арифметического сложения кодов двоичных чисел. Сложение чисел производится поразрядно. Структурная схема четырехразрядного сумматора представлена на рисунке 5.32. Здесь *A* и *B* представляют собой четырехразрядные числа. При сложении чисел между разрядами производится перенос.

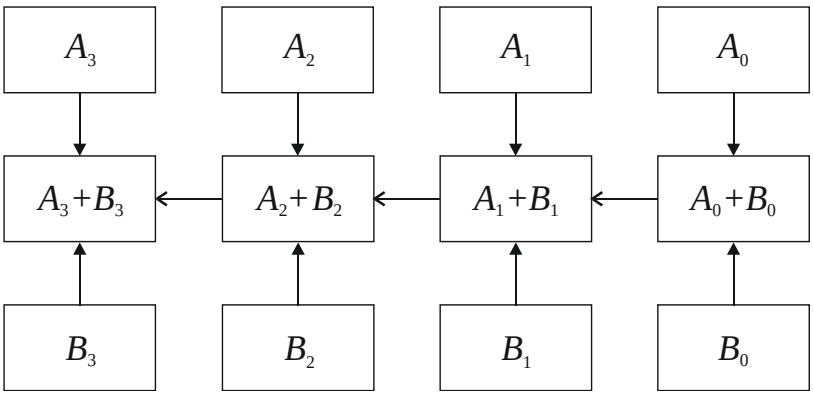


Рис. 5.32 – Сумматор для сложения четырехразрядных двоичных чисел.

Арифметическое сложение подчиняется правилу, показанному в таблице 5.6.

Таблица 5.6 – Правила сложения двоичных чисел

A_i	B_i	$P_i + 1$	$A+B$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Если одноименные разряды обоих чисел равны 1, то значение их суммы равно 0 и возникает перенос (P_i+1), который прибавляется к сумме старших разрядов. В примерах на рисунке 5.33 разряды, из которых произошел перенос, выделены.

	Двоичный код числа				Десятичный код числа
A	0	1	0	1	5
B	0	0	1	0	2
$A+B$	0	1	1	1	7

	Двоичный код числа				Десятичный код числа
A	0	0	1	1	3
B	1	0	0	1	9
$A+B$	1	1	0	0	12

A	0	0	1	1	3
B	0	1	1	0	6
$A+B$	1	0	0	1	9

A	1	0	0	0	8
B	0	0	1	0	2
$A+B$	1	0	1	0	10

Рис. 5.33 – Примеры сложения двоичных чисел

Таким образом, устройство для сложения многоразрядных чисел состоит из ячеек двух типов:

- для сложения самых младших разрядов ячейка должна иметь два входа для приема нулевых разрядов операндов A_0 и B_0, выход S_0 для формирования значения суммы разрядов и выход P_0 для значения переноса в следующий разряд;
- для сложения остальных разрядов каждая *k*-я ячейка должна иметь три входа для переменных A_k, B_k, переноса из предыдущего разряда P_{k-1}, два выхода для формирования значения суммы разрядов S_k и выход P_k для значения переноса в старший разряд.

Таблица истинности для нулевого разряда сумматора и карты Карно функций S_0 и P_0 приведены на рисунке 5.34.

Значения младших разрядов		Перенос	Сумма
A_0	B_0	P_0	S_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

а)

A_0 \ B_0	0	1
0	0	1
1	1	0

б)

A_0 \ B_0	0	1
0	0	0
1	0	1

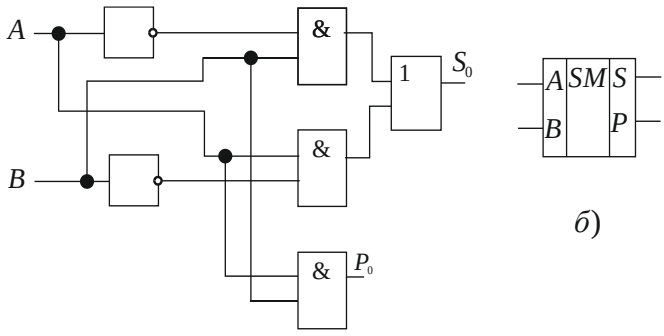
в)

Рис. 5.34 – Сумматор младших разрядов:
таблица истинности (*а*); функция суммы (*б*); функции переноса (*в*)

Из карт Карно следует:

$$S_0 = A_0\overline{B_0} + \overline{A_0}B_0, \quad P_0 = A_0B_0.$$

Устройство, описываемое этими функциями, называется *полусумматором*. Схема полусумматора и его обозначение на схемах показано на рисунке 5.35.



а) *б*)

Рис. 5.35 – Полусумматор:
схема полусумматора (*а*); обозначение на схемах (*б*)

Для сложения остальных разрядов используется *полный сумматор*. Таблица истинности полного сумматора и карты Карно приведены на рисунке 5.36. Схема полного сумматора приведена на рисунке 5.37.

Перенос из предыдущего разряда	Значения младших разрядов		Перенос в следующий разряд	Сумма
P_{i-1}	A_i	B_i	P_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

а)

S_i

P_{i-1} \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

P_i

P_{i-1} \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

б)

Рис. 5.36 – Полный сумматор:
таблица истинности (*а*); карты Карно для суммы и переноса (*б*)

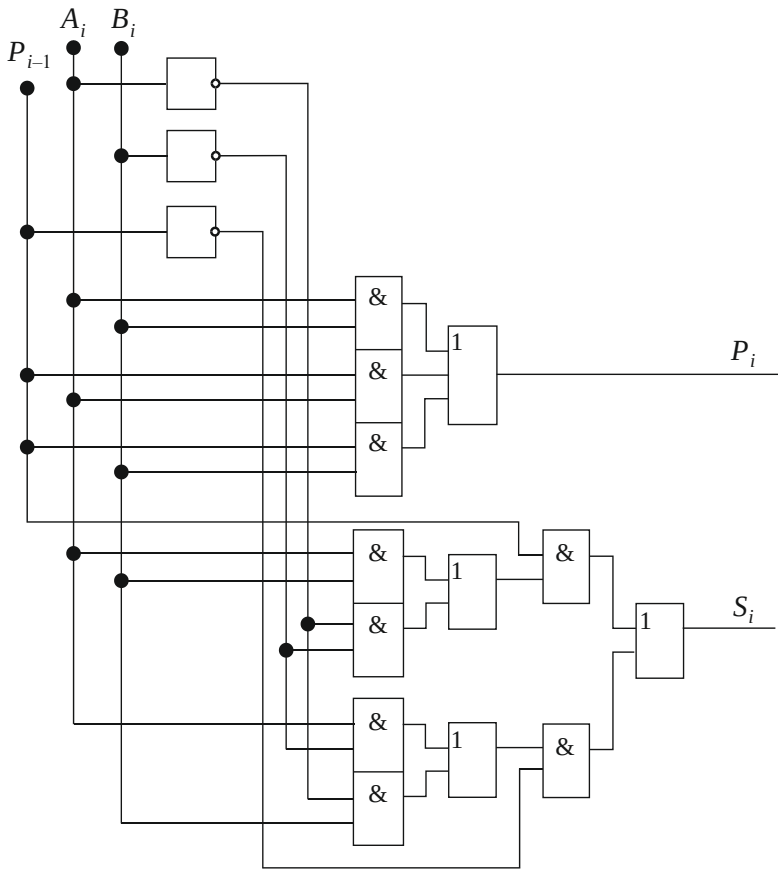
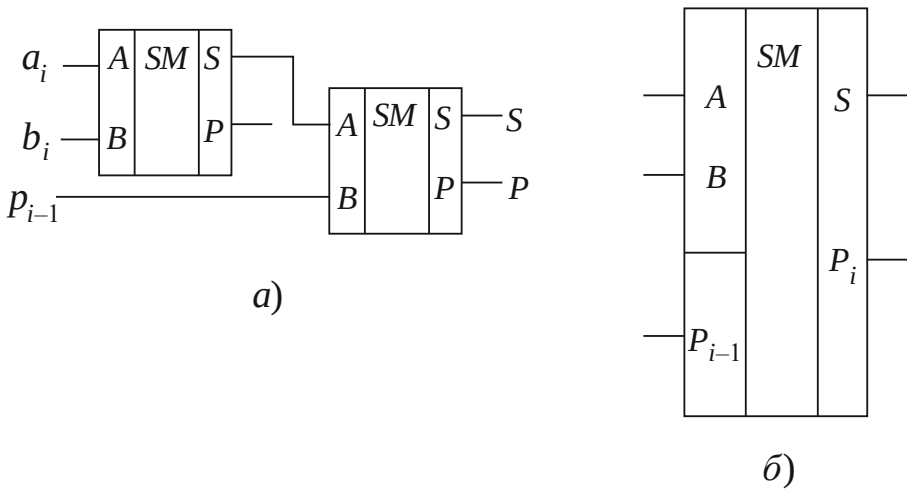


Рис. 5.37 – Схема полного сумматора

Полный сумматор может быть собран из двух полусумматоров. Такая схема сумматора приведена на рисунке 5.38, *а*, а его обозначение – на рисунке 5.38, *б*.



а)

б)

Рис. 5.38 – Полный сумматор из двух полусумматоров:
схема соединения двух полусумматоров (*а*);
обозначение полного сумматора (*б*)

Для сложения n-разрядных чисел нужно использовать один полусумматор для нулевого разряда и $n - 1$ полных сумматоров для остальных разрядов. На рисунке 5.39 приведена схема четырехразрядного сумматора.

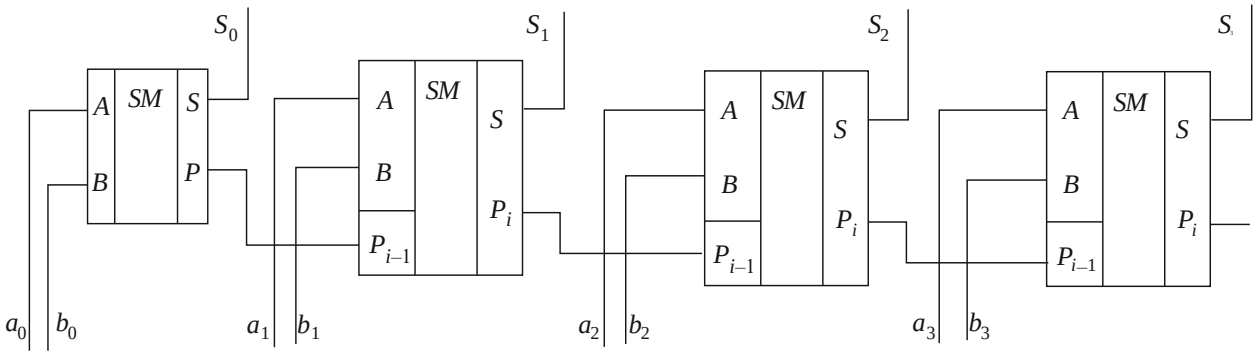


Рис. 5.39 – Схема сумматора для сложения двух четырехразрядных чисел

5.9 Арифметико-логическое устройство

Арифметико-логическое устройство (АЛУ) предназначено для выполнения арифметических и логических операций над числами. Само АЛУ представляет собой сложную комбинационную схему.

Один из возможных вариантов выполнения схемы АЛУ для операций над четырехразрядными числами представлен на рисунке 5.40.

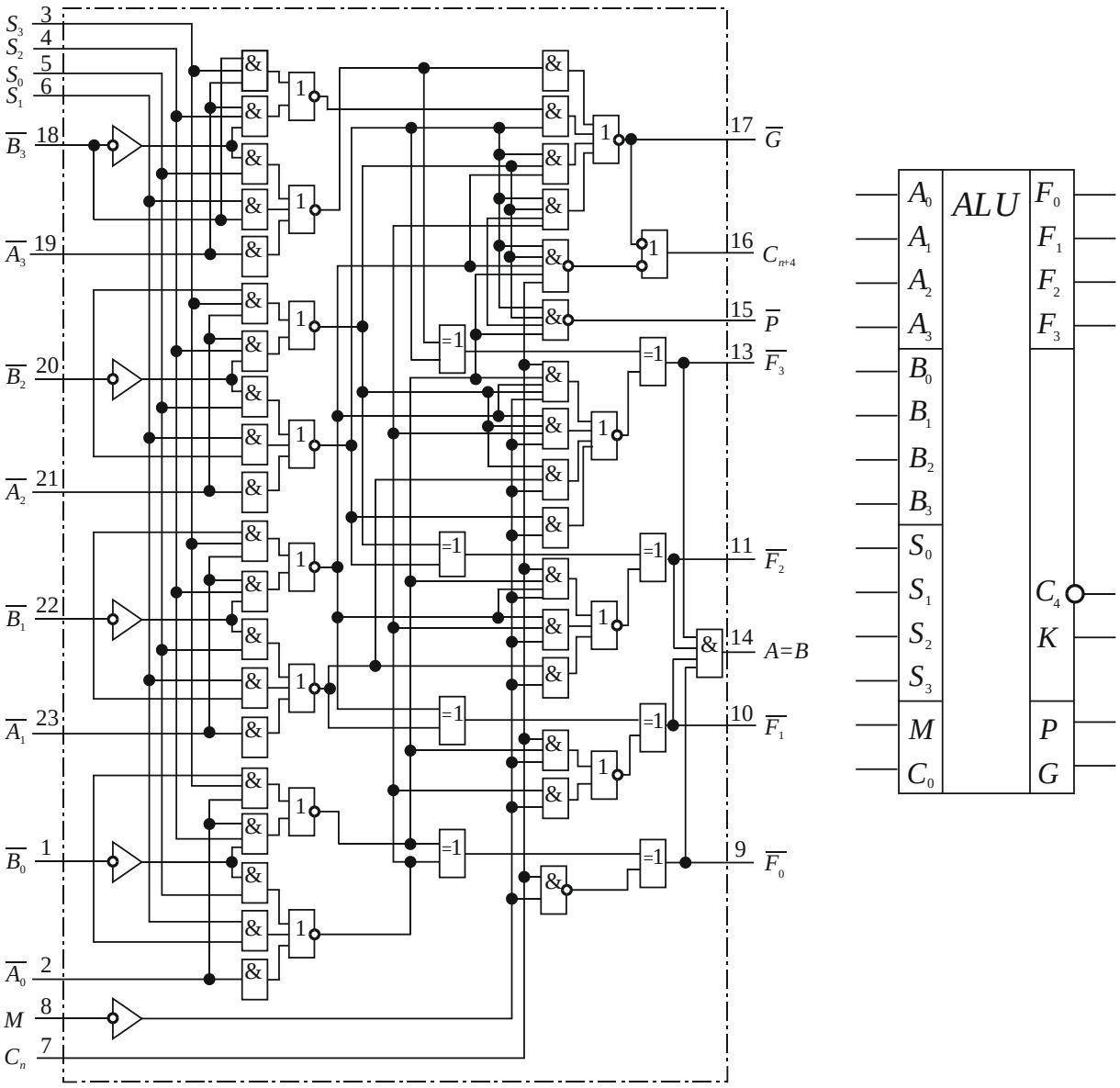


Рис. 5.40 – Схема четырехразрядного АЛУ

АЛУ имеет 4 входа (*A*) для подачи кода одного операнда, 4 входа (*B*) для второго операнда, 4 входа (*S*), на которые подается код выполняемой операции, и несколько управляющих входов. Имеется 4 выхода (*F*) для вывода результата операции и несколько вспомогательных. Вход управления *M* определяет вид операции – логическая или арифметическая. Четырехразрядный код на входах S_0, S_1, S_2, S_3 задает одну из 16 логических или 16 арифметических операций. Вход C_0 и выход *C* служат для приема или передачи единицы переноса и используются для увеличения разрядности АЛУ (кратно четырем разрядам).

5.10 Схемы с третьим состоянием. Шины

Говоря о цифровых устройствах, мы приняли определение о том, что булева переменная и соответствующий ей электрический сигнал могут иметь

только значения ноль или единица. Но при проектировании сложных цифровых устройств возникла необходимость иметь на выходах некоторых логических схем сигнал, имеющий некоторое промежуточное значение. Такие схемы получили название схем с третьим состоянием, оно же *Z*-состояние, оно же *высокоимпедансное* состояние (*High Impedance State* – высокое выходное сопротивление электронного логического элемента).

Третье состояние позволяет соединять напрямую выходы нескольких электронных элементов. В этом состоянии сопротивление между выходом и «землей» становится очень большим и выход элемента не оказывает никакого влияния на подключенные к нему выходы других элементов. Это аналогично тому, что выход элемента как бы отключается от схемы. Такое включение применяется там, где несколько источников сигналов по очереди подключаются к входам одного или нескольких приемников, не мешая друг другу. Схема И-НЕ с *Z*-состоянием выхода приведена на рисунке 5.41, *а*, а ее условное обозначение – на рисунке 5.41, *б*.

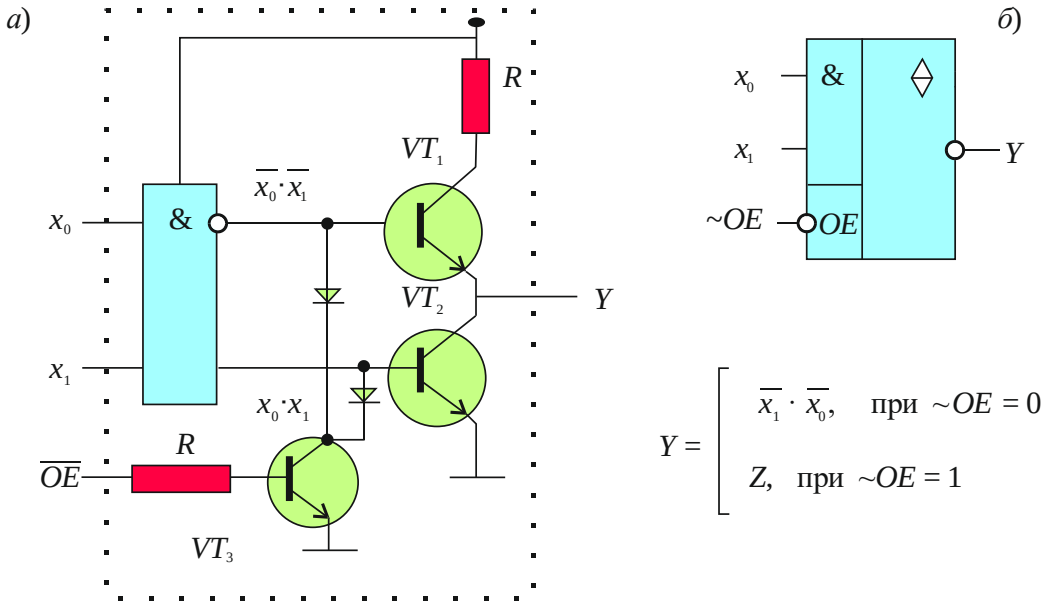


Рис. 5.41 – Схема, реализующая третье (*Z*) состояние

Если сигнал *OE* = 0, транзистор VT_3 заперт, диоды не оказывают влияния на выходы логического элемента И. Схема работает как обычный элемент И-НЕ. При *OE* = 1 транзистор VT_3 откроется до насыщения и на базах транзисторов VT_1 и VT_2 потенциал опустится примерно до нуля, запирая их. Выход *Y* окажется отключенным от внутренней логической схемы. Сопротивления переходов коллектор – эмиттер VT_1 и VT_2 имеют значения в мегомах и напряжение в средней точке $\approx 0{,}5\,E_{\text{пит}}$. Ток, текущий через эту цепь, имеет значение в мик-

роамперах. На схемах такие элементы обозначаются ромбом с поперечной чертой или буквой *Z*. Таким образом, на выходе такого устройства можно получить сигнал низкого уровня (логический 0), сигнал высокого уровня (логическая 1) и сигнал половинной амплитуды (*Z* или третье состояние).

Шина как физическое устройство является частью цифрового устройства. В то же время на графических схемах устройств шина является вспомогательным средством, предназначенным для упрощения графического изображения.

Итак, шины на схемах – это линия, включающая в себя *множество линий цепей*, соединяющих входы и выходы *устройств и элементов*, расположенных на схеме. Представьте себе схему, состоящую из нескольких десятков или сотен элементов, соединенных линиями цепей для передачи сигналов. На изображении таких схем придется рисовать сотни или тысячи линий. Разобраться в таких схемах и проанализировать работу устройства становится очень затруднительно.

На схемах шина представляет собой линию. Ей можно присвоить какой-либо идентификатор (имя) – цифровой или буквенный, например, назвать шину *A* или дать ей имя A_1. В шину можно в любом месте ввести провод цепи с выхода какого-то элемента, присвоив ему свой идентификатор, чаще всего дают его номер в шине. В любом месте шины этот провод можно вывести из шины и подключить его к какому-либо входу элемента. На схеме можно организовать любое количество шин со своими идентификаторами. Шины можно разделять и объединять.

На рисунке 5.42 изображена схема управляющего устройства с применением шины.

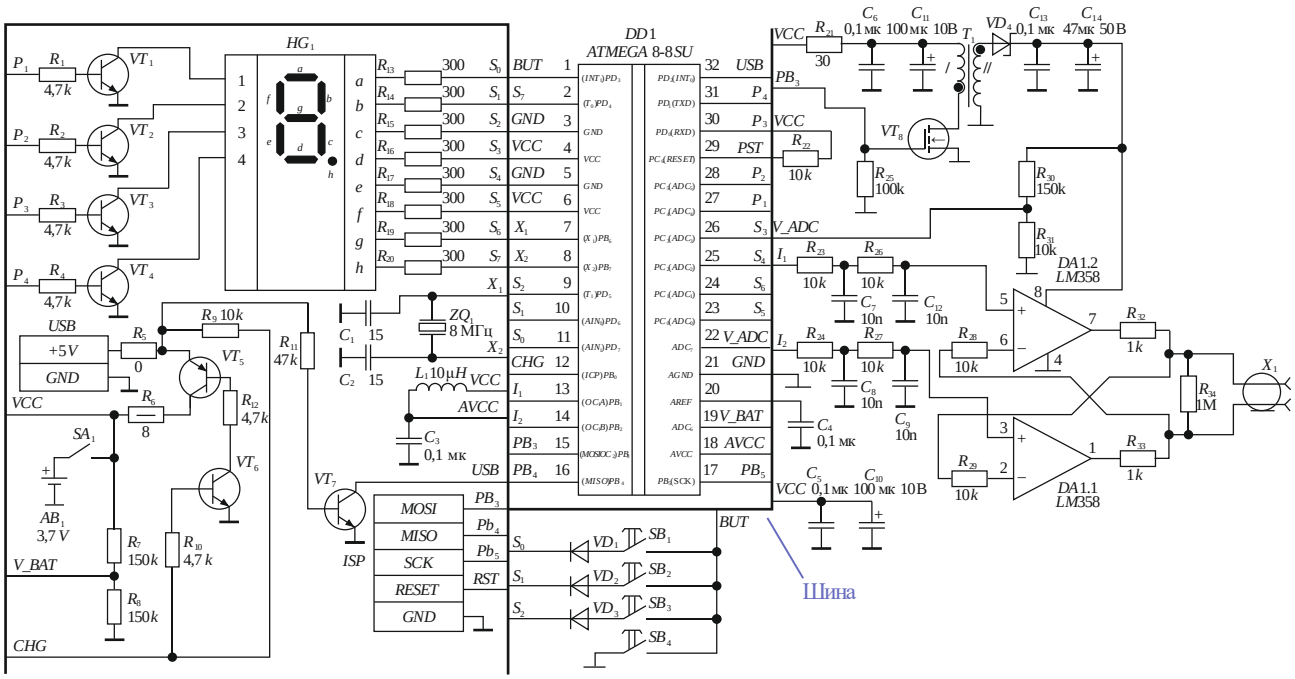
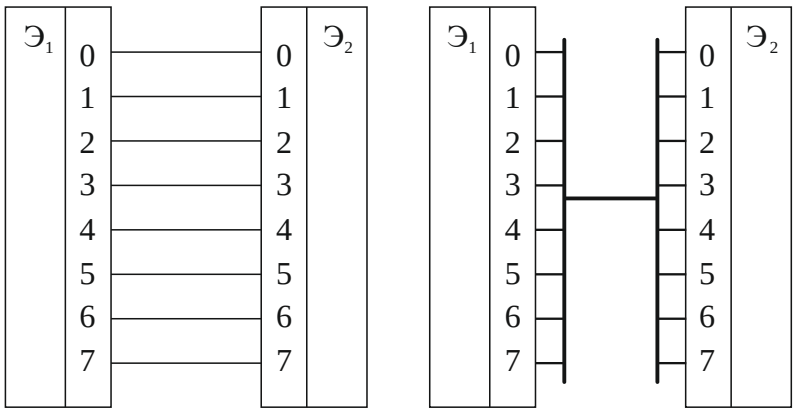


Рис. 5.42 – Схема устройства с применением шины

Обратите внимание на то, что шина позволяет компактно нарисовать схему. Выходные сигналы входят в шину со своими именами. Например, с выхода «*a*» элемента HG_1 сигнал через резистор R_{13} поступает в шину с именем S_0. Из шины он выходит с тем же именем и поступает на вывод 11 микроконтроллера *DD*1 и через диод VD_1 поступает на переключатель SB_1. Если бы не было шины, то пришлось бы через лист схемы тянуть две линии.

Таким образом, ***шины на схемах*** *– это просто средство уменьшения количества линий цепей, связывающих различные устройства.*

Теперь о физических шинах. Шины в цифровых устройствах – это совокупность проводников для передачи электрических сигналов. Простейшая шина – это одна цепь, соединяющая выход одного элемента со входом другого. Чаще, конечно, в шине присутствуют множество проводников. На рисунке 5.43, *а* показано соединение выходов элемента $Э_1$ со входами $Э_2$, что означает передачу данных только в одном направлении. На рисунке 5.43, *б* показано, как это будет выглядеть на графической схеме устройства. Такая шина называется однонаправленной.



а) *б*)

Рис. 5.43 – Однонаправленная шина:
физическое соединение устройств (*а*); обозначение шины на схеме (*б*)

В сложных цифровых устройствах с большим количеством элементов возникает вопрос обмен большими потоками данных. Для решения этой задачи используют двунаправленные шины, называемые *магистралями*. Например, как на рисунке 5.42. Они включают в себя не только наборы проводников, но и специальные схемы. Например, в персональных компьютерах на материнских платах имеется несколько магистралей, управляют которыми специализированные микропроцессоры (микроконтроллеры). В магистралях по одному и тому же проводу электрический сигнал может передаваться то в одну, то в другую сторону в режиме разделения времени. Вот в таких шинах и удобно использовать схемы с тремя состояниями. Специально для таких случаев разрабатываются различные цифровые узлы, имеющие выходы с Z-состоянием. Шинам присваивают имена и часто указывают их разрядность в скобках около имени шины.

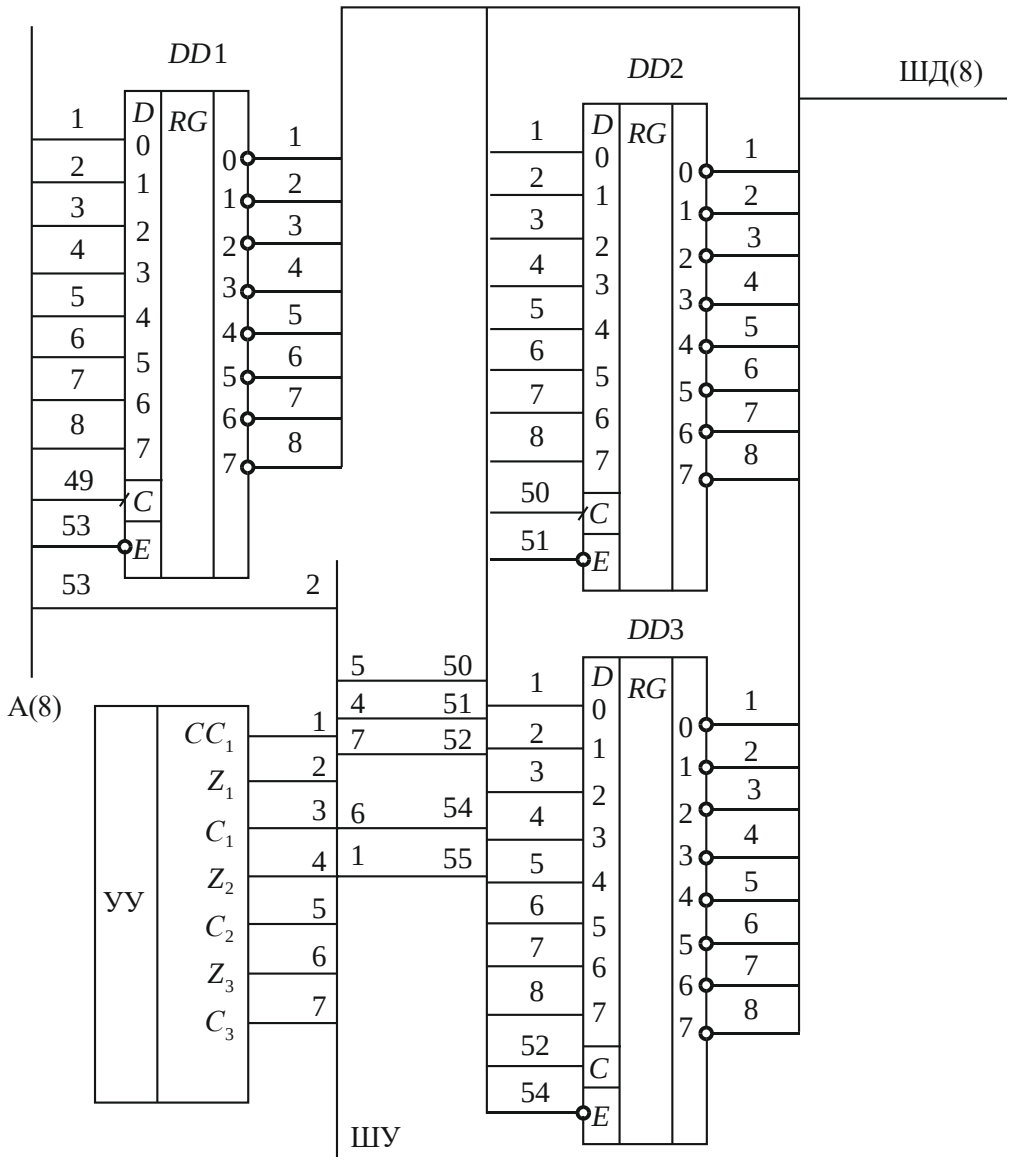


Рис. 5.44 – Фрагмент схемы с шинами

На рисунке 5.44 представлен фрагмент схемы с шинами. Восьмиразрядная однонаправленная шина А(8) приходит из той части схемы, которая невидна. Однонаправленная шина управления (ШУ) формируется из сигналов устройства управления (УУ). Элементы *DD*1, *DD*2, *DD*3 (все с выходами *Z*) являются источниками и приемниками данных и могут по двунаправленной шине ШД(8) обмениваться данными между собой (за исключением *DD*1) и другими устройствами, к которым эта шина подключена на другой части схемы. Из *DD*1 данные по ШД(8) могут быть считаны в *DD*2 или в *DD*3. В то же время *DD*2 и *DD*3 могут обмениваться данными между собой. Если же в устройстве используются элементы, не имеющие *Z*-выхода, то применяются специализированные элементы, называемые передатчиками (шинными формирователями), предназначенные для

однонаправленных шин. Для двунаправленных шин используют приемопередатчики.

Для организации однонаправленной шины можно использовать шинный формирователь. Например, на рисунке 5.45 представлена и схема формирователя, и его обозначение на схемах. Если два таких формирователя включить встречно, то можно создать двунаправленную шину.

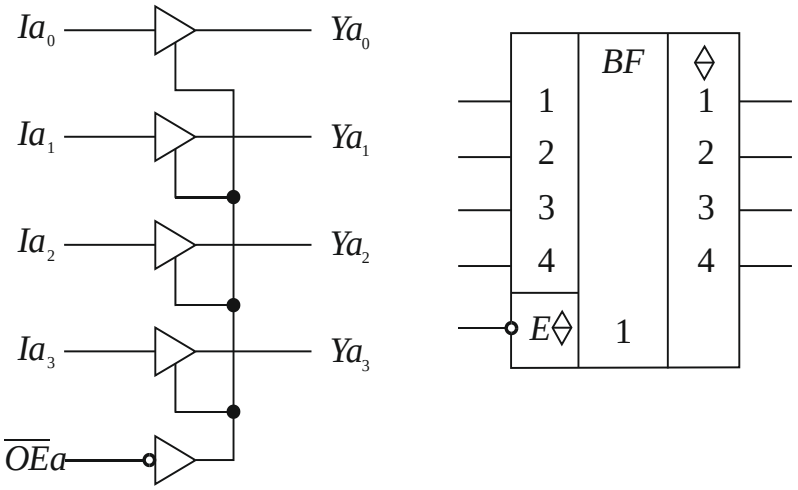


Рис. 5.45 – Четырехразрядный однонаправленный шинный формирователь

Для создания двунаправленных шин разработаны многоразрядные формирователи. На рисунке 5.46 представлена схема одного разряда и обозначение на схемах восьмиразрядного шинного формирователя.

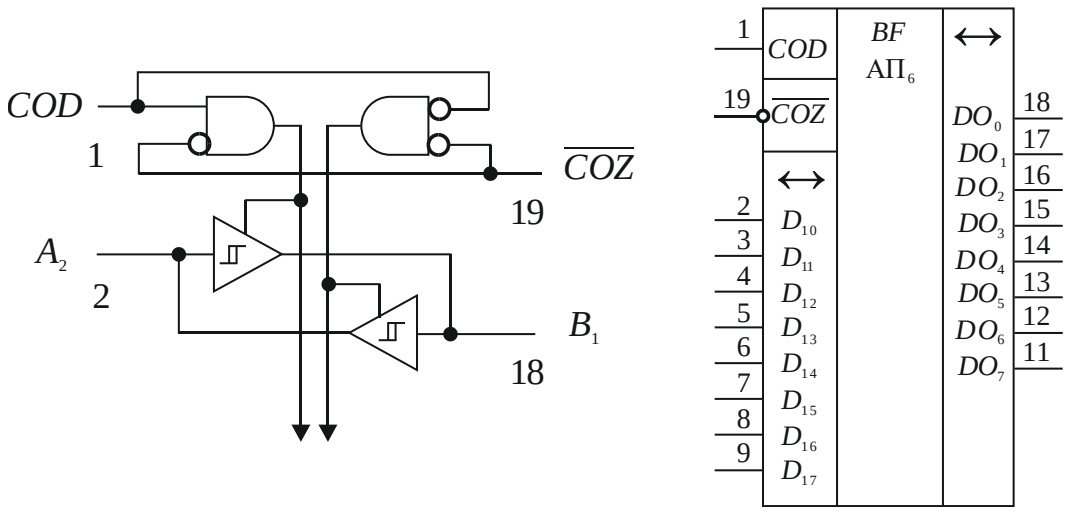


Рис. 5.46 – Восьмиразрядный двунаправленный шинный формирователь

5.11 Запоминающие устройства

5.11.1 Оперативные запоминающие устройства

Оперативное запоминающее устройство, оперативная память (ОЗУ, *RAM*) – это устройство, позволяющее хранить большие объемы данных, причем содержимое памяти может быть считано или изменено в любой нужный момент времени.

Существует большое количество схем, реализующих память, но мы рассмотрим только ОЗУ с произвольным доступом, структурная схема которого приведена на рисунке 5.47.

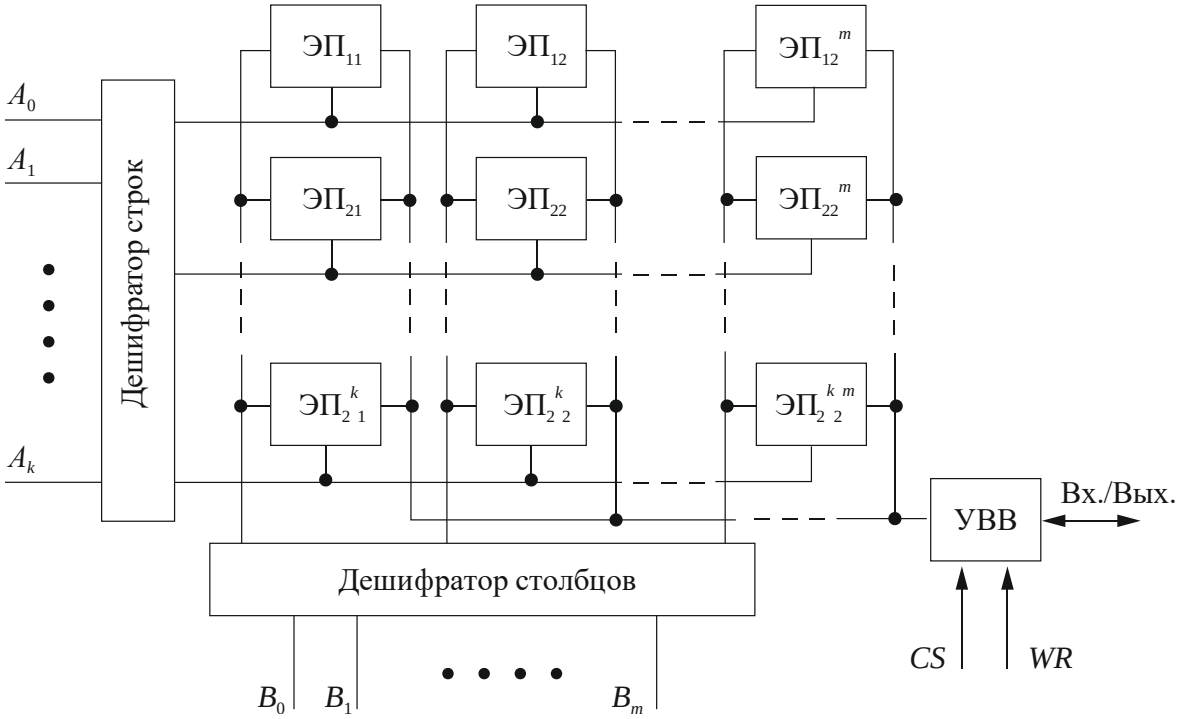


Рис. 5.47 – Структурная схема ОЗУ

ОЗУ состоит из ячеек памяти (элементов памяти – ЭП), размещенных упорядоченно по строкам и столбцам матрицы. Каждый ЭП может хранить от одного до нескольких битов данных, чаще всего 1 или 8 бит (1 байт). Все ЭП пронумерованы числами от нуля до *N*. Номер ЭП называется его *адресом*. Адрес представляется двоичным числом, каждое из которых соответствует номеру ЭП. Адрес ЭП дешифрируется дешифраторами строк и столбцов матрицы, на пересечении которых и находится искомый ЭП. УВВ – устройство ввода/вывода для управления процессом обращения к ОЗУ. Сигнал *CS* для УВВ является разрешающим. Сигнал *WR* при высоком уровне является сигналом чтения (*RD* – *read* – чтение) данных из ОЗУ, а при низком – записью (*write* – за-

пись) данных в ОЗУ. В некоторых ОЗУ могут присутствовать оба сигнала, *WR* и *RD*. ОЗУ имеет следующие основные характеристики:

- разрядность ЭП;
- емкость – количество ЭП;
- быстродействие – за какое время можно обратиться к ЭП, чтобы прочитать данные из ячейки или записать данные в ЭП.

ОЗУ работает с двумя циклами – записи и чтения.

Чтение (рис. 5.48):

1) внешнее устройство (ВУ), обратившееся к ОЗУ, передает на входы адреса двоичный код адреса (*ADR*), который дешифрируется дешифраторами строк и столбцов, выбирая один-единственный ЭП;
2) с задержкой по отношению к адресу ВУ выдает на УВВ разрешающий сигнал *CS* – выбор кристалла, который снимает *Z*-состояние с выходных цепей шины данных ОЗУ (включает кристалл микросхемы ОЗУ в работу);
3) сигнал *RD* (*read* – чтение) высокого уровня, переключает выходы выбранного ЭП на выходы УВВ, передавая данные (*DATA*) ВУ;
4) после этого ВУ должно снять *ADR* и *CS*. Цикл чтения завершен.

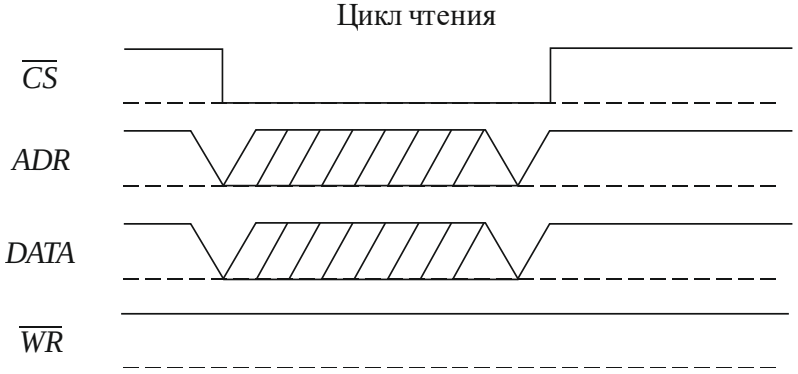


Рис. 5.48 – Цикл чтения из ОЗУ

Запись (рис. 5.49):

1) ВУ передает на входы адреса двоичный код адреса, который дешифрируется дешифраторами строк и столбцов;
2) ВУ передает на входы УВВ код данных (*DATA*);
3) с задержкой по отношению к адресу ВУ выдает на УВВ разрешение *CS*;
4) с задержкой по отношению к *CS* ВУ формирует сигнал записи *WR* (*Write* – запись), по которому в ЭП записывается код данных;

5) снимаются *WR*, *DATA*, *CS*. Цикл записи окончен.

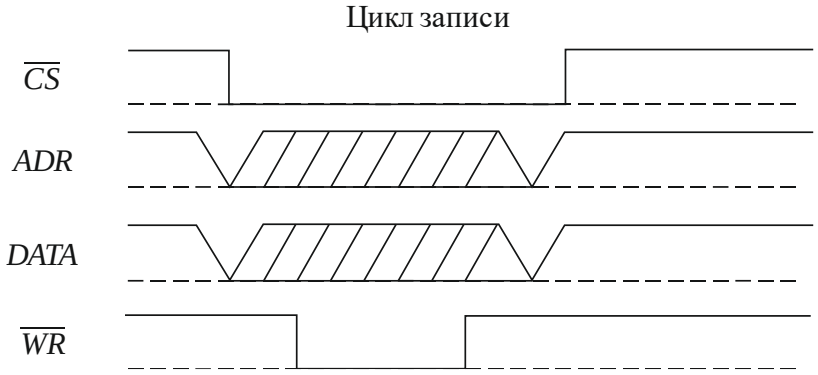


Рис. 5.49 – Цикл записи в ОЗУ

Различают ОЗУ статические и динамические. Статические менее емкие и более быстродействующие. Динамические имеют меньшее быстродействие, но в то же время гораздо более емкие, т. к. схема одного ЭП значительно проще.

Память, построенная в соответствие со структурой (рис. 5.47), называется памятью с произвольным доступом, потому что к любой ячейке памяти можно обратиться (получить доступ) в любой момент времени, задав нужный адрес.

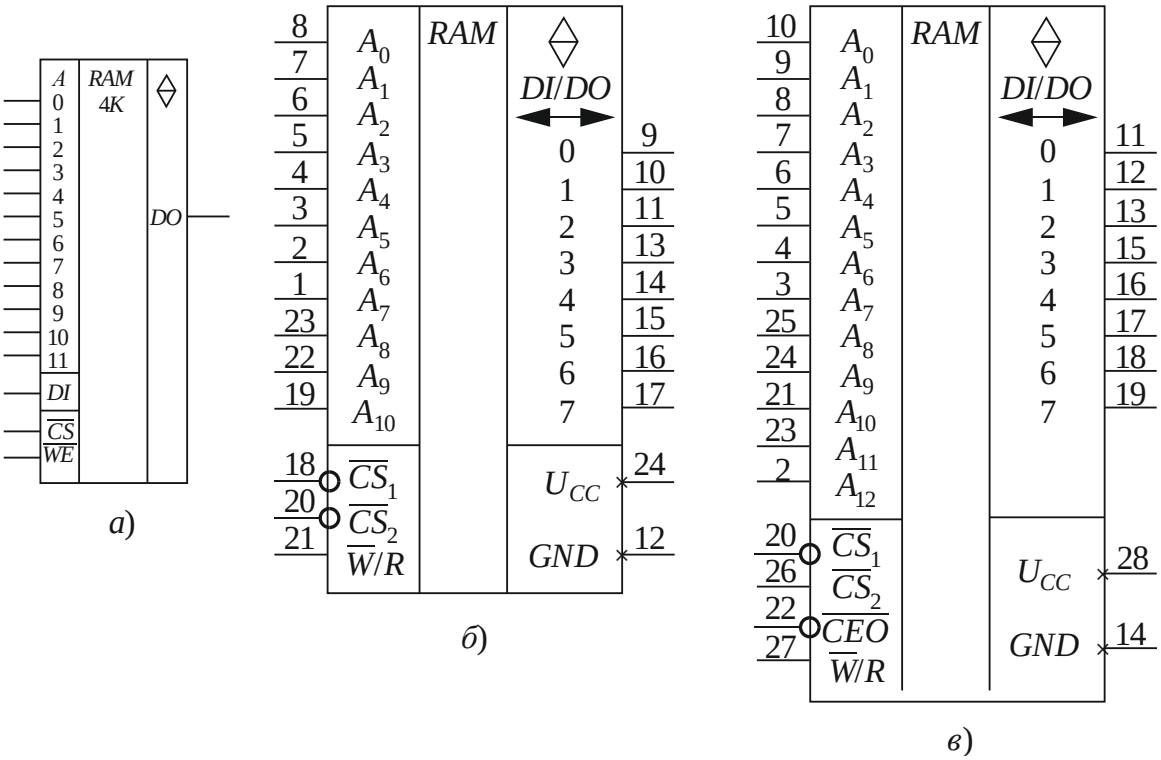


Рис. 5.50 – Обозначение микросхем ОЗУ на схемах:
ОЗУ емкостью 1 Кбит (*а*); ОЗУ емкостью 1 Кбайт (*б*);
ОЗУ емкостью 8 Кбайт (*в*)

На рисунке 5.50 приведены обозначения некоторых микросхем ОЗУ, которые используются при создании схем цифровых устройств. Используя эти микросхемы и управляющие сигналы *CS*, можно строить ОЗУ большой емкости.

5.11.2 Постоянные запоминающие устройства

Полупроводниковые постоянные запоминающие устройства (ПЗУ, *ROM*) предназначены для хранения редко изменяемых данных, чаще всего констант, программ микроконтроллеров и т. п. Поэтому данные в ПЗУ записываются или только один раз или обновляются очень редко. Основная работа ПЗУ заключается в чтении из них нужных в данное время данных. Процесс записи данных в ПЗУ называется программированием или прошивкой, а записанный код называется зашитым. Программирование выполняется с помощью программаторов (как и ПЛМ).

ПЗУ имеет примерно такую же структуру, как и ОЗУ с произвольным доступом, но в качестве ячеек памяти выступают другие элементы. По сути, они и не являются ячейками памяти. ПЗУ имеют почти такие же характеристиками, как и ОЗУ. Следует заметить, что по сравнению с ОЗУ эти устройства обладают существенно большим временем обращения к ним и гораздо меньшей емкостью.

Полупроводниковые ПЗУ делятся на несколько видов:

1) однократно программируемые – пользователь может записать данные только один раз;
2) масочные – программируются на заводе-изготовителе ПЗУ и из них можно только читать данные;
3) многократно программируемые – пользователь может стирать старые данные и записывать новые несколько раз (до нескольких десятков, реже сотен раз). Они делятся:
    - на ПЗУ со стиранием старых данных с помощью ультрафиолетового облучения;
    - ПЗУ с электрическим стиранием старых данных.

Отдельной строкой следует назвать современные ПЗУ, изготовленные по *flesh*-технологии (флеш). Они имеют относительно большую емкость и неплохое быстродействие режимов записи и чтения, конечно, уступая в этом современным ОЗУ. Но по своей структуре они наиболее близки к структуре ОЗУ.

На рисунке 5.51 приведена структура незапрограммированного диодного ПЗУ. ПЗУ состоит из матрицы диодов с элементами, дающими возможность

разорвать цепь. Слова данных, располагаются в строках матрицы. Для выбора строки, т. е. нужных данных, используется дешифратор, на входы которого подается двоичный код адреса нужной строки. Управляющий узел (УУ) принимает извне два сигнала. Инверсный сигнал *CS* (выбор кристалла) разрешает работу ПЗУ (если 0), или запрещает ее (если 1). Сигнал *P* дает разрешение на программирование матрицы данных. Выходной буфер состоит из элементов, согласующих внутренние сигналы ПЗУ с сигналами внешних устройств. Чаще всего эти элементы имеют *Z*-состояние выходов.

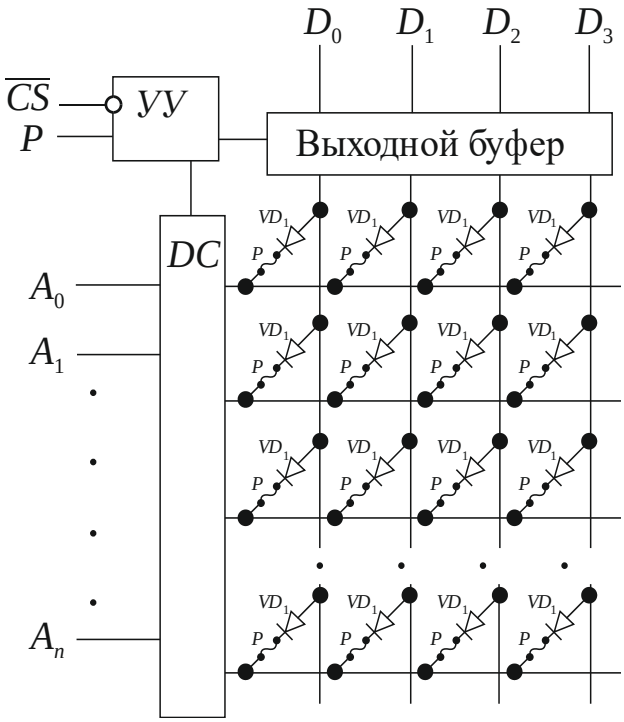


Рис. 5.51 – Структура диодного ПЗУ

При программировании элементы *P* можно удалять, и тогда в строке матрицы «зашивается» заданный двоичный код. При работе ПЗУ в режиме чтения на вход *CS* подается 0, а на адресные входы код адреса. Только один выход дешифратора адреса возбуждается, и в выходной буфер этой строки передается код «зашитых» в ней данных.

На рисунке 5.52 показан пример прошивки ПЗУ. При подаче нулевого адреса, равного 000…0, возбудится нулевой выход дешифратора, и в выходной буфер считается код в $D_0D_1D_2D_3 = 1010$. При подаче адреса, равного единице, – код 1001. При подаче самого старшего адреса 11…1 считается код 0111.

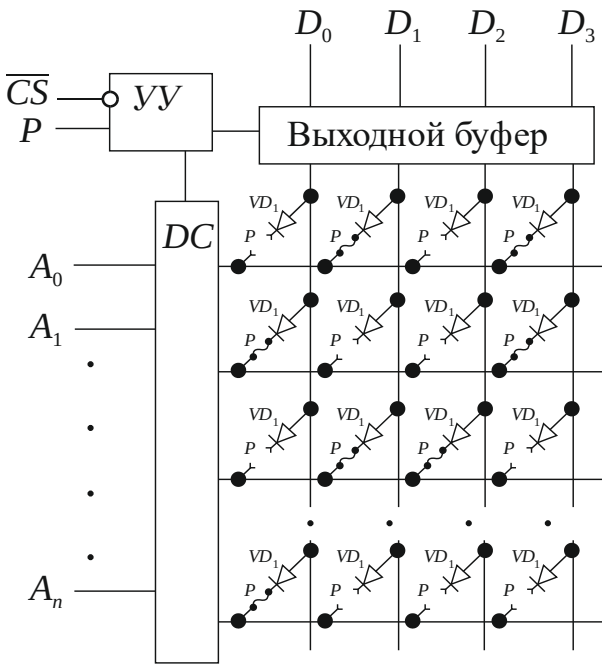
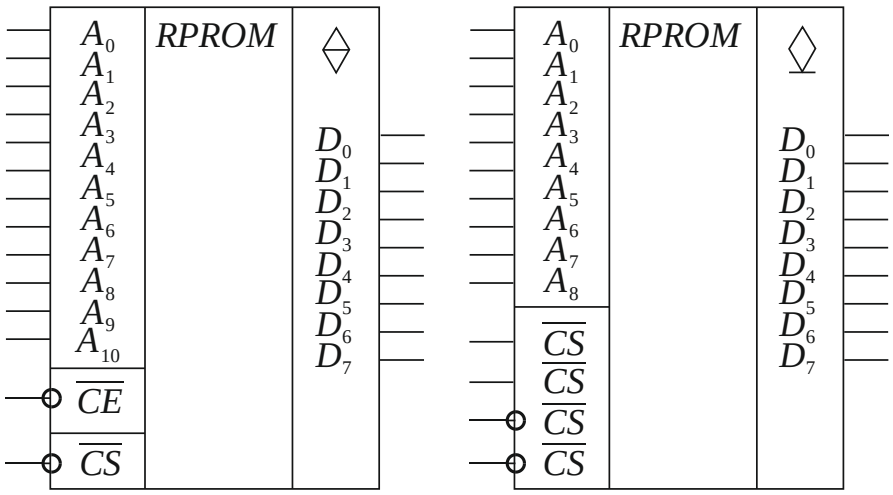


Рис. 5.52 – Пример прошивки диодного ПЗУ

На рисунке 5.53 показано обозначение ПЗУ на схемах.



а) *б*)

Рис. 5.53 – Обозначение ПЗУ на схемах:
ПЗУ емкостью 2 Кбайт (*а*); ПЗУ емкостью 512 байт (*б*)

Контрольные вопросы по главе 5

1. Можно ли один из выходов ПЛМ соединить с ее входом?
2. Может ли ПЛМ иметь на выходе схемы с Z-состоянием?
3. Что такое магистраль?
4. Какие данные хранят в ПЗУ?

5. Может ли дешифратор с инверсными выходами иметь один выход?
6. Какими свойствами должны обладать устройства, на которых можно строить магистрали передачи данных?
7. Какой функцией описывается мультиплексор 8 на 1?
8. В основе описания функций шифратора лежит ДНФ или КНФ?
9. На схемах сравнения можно выполнять операцию сравнения чисел без знака. Так ли это?
10. Функция любого выхода дешифратора может быть реализована только в базисе И, ИЛИ. Верно ли это утверждение?

6 Синтез КС с использованием узлов цифровых устройств

6.1 Синтез КС с использованием дешифраторов

В булевых функциях узлов уже присутствуют логические операции входных переменных. Это позволяет реализовывать БФ на основе этих узлов.

Задача 6.1. С помощью дешифратора синтезировать схему, реализующую БФ:

$$F_1 = \sum(0,\ 2,\ 4,\ 5,\ 7).$$

Запишем эту БФ в СДНФ:

$$F_1 = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1x_2x_3.$$

Вспомним систему БФ трехразрядного дешифратора:

$$Y_0 = \overline{x}_1\overline{x}_2\overline{x}_3,\quad Y_1 = \overline{x}_1\overline{x}_2x_3,\quad Y_2 = \overline{x}_1x_2\overline{x}_3,\quad Y_3 = x_1\overline{x}_2x_3,$$
$$Y_4 = x_1\overline{x}_2\overline{x}_3,\quad Y_5 = \overline{x}_1x_2\overline{x}_3,\quad Y_6 = x_1x_2\overline{x}_3,\quad Y_7 = x_1x_2x_3.$$

Выходные функции дешифратора Y_0, Y_2, Y_4, Y_5, Y_7 полностью совпадают с конъюнкциями, входящими в F_1. Таким образом, *тривиальная* реализация БФ заключается в простом логическом сложении сигналов с соответствующих выходов дешифратора. Чтобы реализовать заданную БФ, к дешифратору нужно добавить схему, выполняющую логическое сложение сигналов с этих выходов, т. е. схему ИЛИ (рис. 6.1).

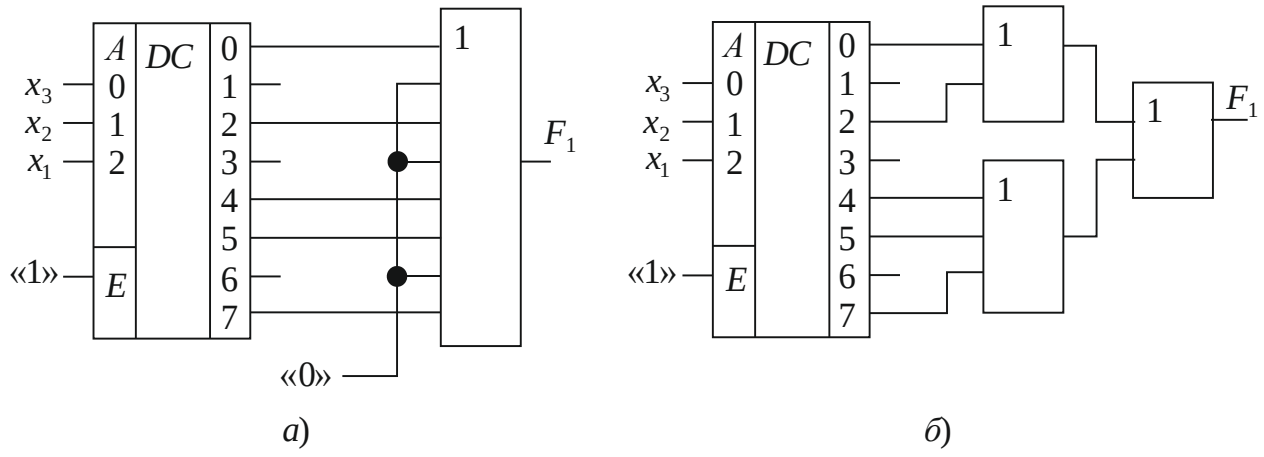


Рис. 6.1 – Схема реализации F_1 на управляемом дешифраторе:
с использованием схемы ИЛИ на 8 входов (*а*);
с использованием нескольких схем ИЛИ (*б*)

Комбинационные схемы, соответствующие функции F_1, представлены на рисунке 6.1, *а*. В схеме использовалась схема ИЛИ на 8 входов. На неиспользуемые входы подается нулевой сигнал. На рисунке 6.1, *б* приведена схема с каскадным включением схем ИЛИ с двумя и тремя входами. Такая схема используется в тех случаях, когда логические элементы, используемые в схеме, имеют ограниченное число входов.

Неиспользуемые выходы дешифраторов ни к чему не подключаются и остаются «висящими в воздухе».

Задачу можно решить с использованием двухразрядных дешифраторов (рис. 6.2). Из них строится трехразрядный дешифратор, соответствующие выходы которого собираются на схемах ИЛИ. В этой схеме элемент ИЛИ на пять входов реализован иначе, чем на рисунке 6.1, *б*.

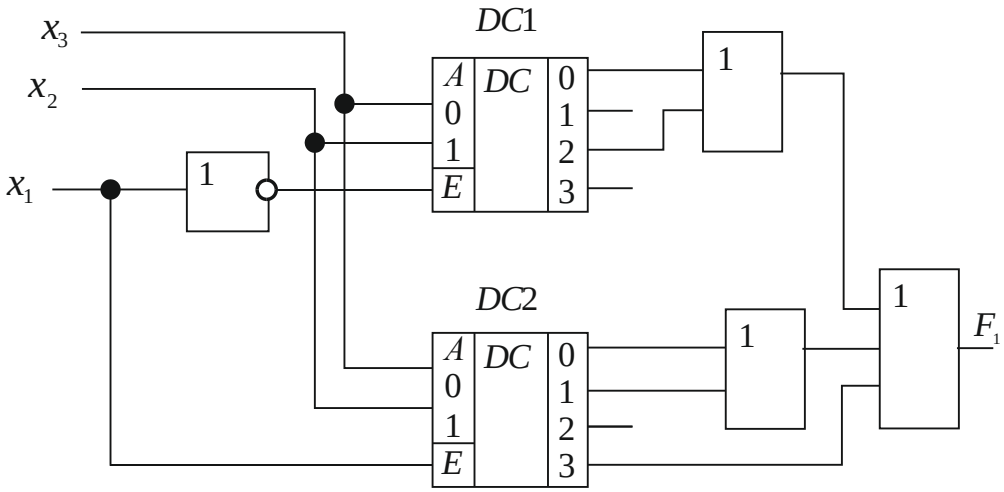


Рис. 6.2 – Реализация БФ F_1 на двухразрядных дешифраторах

Задача 6.2. С помощью трехразрядного дешифратора с инверсными выходами синтезировать схему, реализующую БФ:

$$F_2 = \sum(3, 4, 5, 6, 8, 10, 11);$$

$$F_2 = \overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1\overline{x}_2x_3x_4.$$

Любой выход дешифратора с инверсными выходами реализует конъюнкцию с инверсией. В соответствии с заданием необходимо преобразовать функцию с использованием теоремы де Моргана следующим образом:

$$F_2 = \overline{\overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1\overline{x}_2x_3x_4};$$

$$F_2 = \overline{\overline{x}_1\overline{x}_2x_3x_4} \cdot \overline{\overline{x}_1x_2\overline{x}_3\overline{x}_4} \cdot \overline{\overline{x}_1x_2\overline{x}_3x_4} \cdot \overline{\overline{x}_1x_2x_3\overline{x}_4} \cdot \overline{x_1\overline{x}_2\overline{x}_3\overline{x}_4} \cdot \overline{x_1\overline{x}_2x_3\overline{x}_4} \cdot \overline{x_1\overline{x}_2x_3x_4}.$$

Четырехразрядный дешифратор построен из двух трехразрядных дешифраторов. В качестве выходной схемы используется стандартная схема И-НЕ с восемью входами. На неиспользуемый вход схемы И-НЕ подается сигнал, равный логической единице (рис. 6.3).

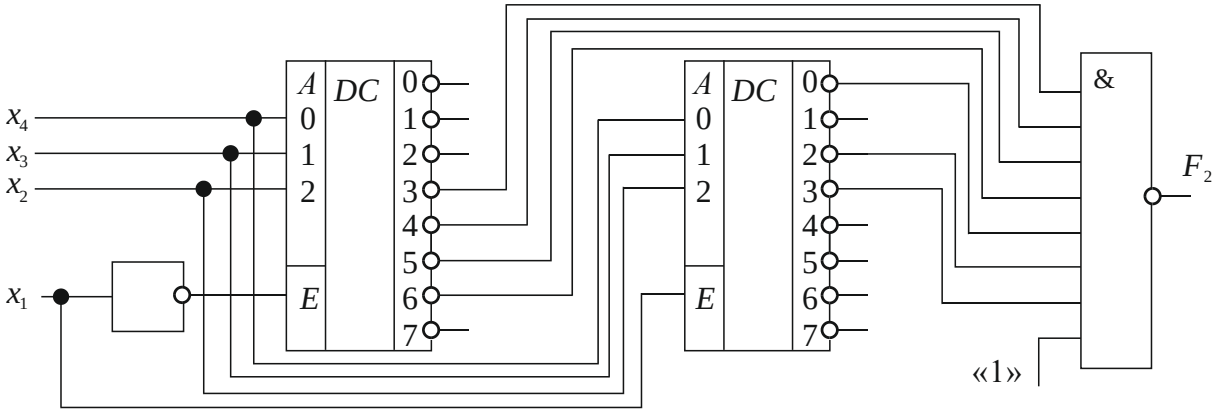


Рис. 6.3 – Реализация функции F_2

Задача 6.3. С помощью трехразрядного дешифратора синтезировать схему, реализующую БФ:

$$Z = \prod(1,\ 2,\ 5,\ 6,\ 7,\ 8,\ 12,\ 13,\ 15).$$

Эту задачу можно решить двумя способами:

1. Поскольку функция задана в КНФ, то задачу можно переформулировать: $Z = \sum(0,\ 3,\ 4,\ 9,\ 10,\ 11,\ 14)$, т. е. задать БФ в форме ДНФ, при этом указываются те наборы аргументов, на которых БФ принимает единичные значения. В этом случае удобно строить схему на дешифраторах с прямыми выходами, как и в предыдущих задачах (рис. 6.4, *а*).

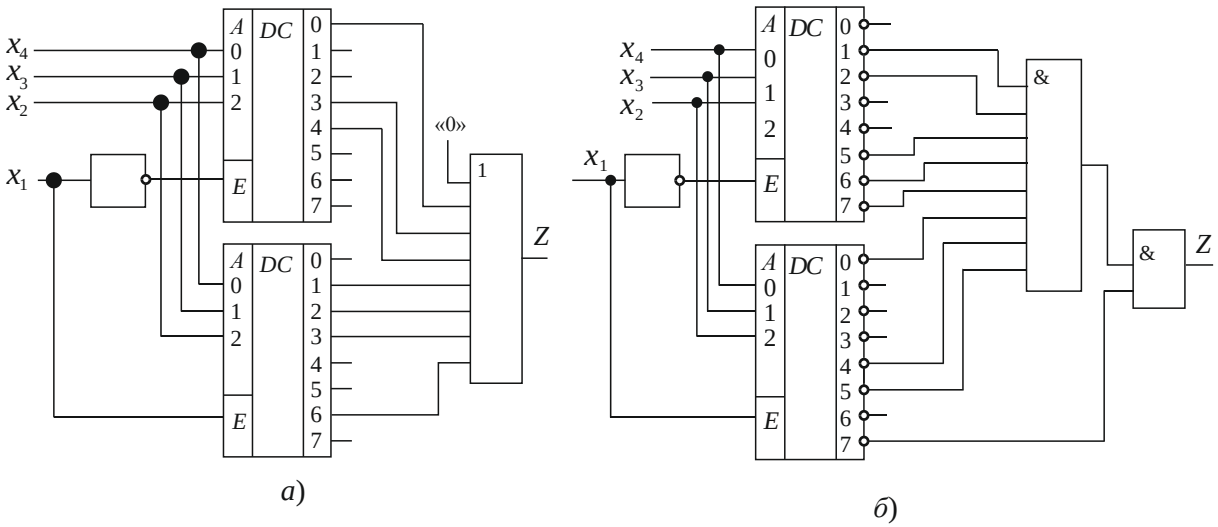


Рис. 6.4 – Реализация функции Z:
на дешифраторе с прямыми выходами (*а*);
на дешифраторе с инверсными выходами (*б*)

2. Записать БФ в форме СКНФ и выполнить преобразования по переводу в базис И-НЕ:

$$Z = (x_1 + x_2 + x_3 + \overline{x}_4)(x_1 + x_2 + \overline{x}_3 + x_4)(x_1 + \overline{x}_2 + x_3 + \overline{x}_4)(x_1 + \overline{x}_2 + \overline{x}_3 + x_4)$$
$$(x_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4)(\overline{x}_1 + x_2 + x_3 + x_4)(\overline{x}_1 + \overline{x}_2 + x_3 + x_4)(\overline{x}_1 + \overline{x}_2 + x_3 + \overline{x}_4)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4);$$

$$Z = \overline{\overline{(x_1 + x_2 + x_3 + \overline{x}_4)}}\,\overline{\overline{(x_1 + x_2 + \overline{x}_3 + x_4)}}\,\overline{\overline{(x_1 + \overline{x}_2 + x_3 + \overline{x}_4)}}\,\overline{\overline{(x_1 + \overline{x}_2 + \overline{x}_3 + x_4)}}$$
$$\overline{\overline{(x_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4)}}\,\overline{\overline{(\overline{x}_1 + x_2 + x_3 + x_4)}}\,\overline{\overline{(\overline{x}_1 + \overline{x}_2 + x_3 + x_4)}}\,\overline{\overline{(\overline{x}_1 + \overline{x}_2 + x_3 + \overline{x}_4)}}\,\overline{\overline{(\overline{x}_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4)}};$$

$$Z = \overline{\overline{x}_1\overline{x}_2\overline{x}_3x_4} \cdot \overline{\overline{x}_1\overline{x}_2x_3\overline{x}_4} \cdot \overline{\overline{x}_1x_2\overline{x}_3x_4} \cdot \overline{\overline{x}_1x_2x_3\overline{x}_4} \cdot \overline{\overline{x}_1x_2x_3x_4} \cdot$$
$$\cdot\overline{x_1\overline{x}_2\overline{x}_3\overline{x}_4} \cdot \overline{x_1x_2\overline{x}_3\overline{x}_4} \cdot \overline{x_1x_2\overline{x}_3x_4} \cdot \overline{x_1x_2x_3x_4}.$$

Каждая конъюнкция с инверсией в формуле описывает функцию одного из выходов дешифратора с инверсными выходами. Для реализации функции Z с использованием дешифратора с инверсными выходами в схему включена схема И, выполняющая логическое произведение всех конъюнкций (рис. 6.4, *б*).

6.2 Синтез КС с использованием мультиплексоров

Булева функция управляемого мультиплексора 4 на 1 имеет вид:

$$f = ED_0\overline{x}_2\overline{x}_1 + ED_1\overline{x}_2x_1 + ED_2x_2\overline{x}_1 + ED_3x_2x_1,$$

где адресная переменная обозначена через x.

Анализ функции (при $E = 1$) показывает, что в каждую конъюнкцию входит конституента единицы из переменных адреса, т. к. в состав мультиплексора входит дешифратор. Кроме того, мультиплексор реализует логическую сумму конъюнкций переменных. Отсюда очевидна реализация БФ: если в любой мо-

мент времени переменные x_0x_1 принимают одно из четырех значений (00, 01, 10, 11), то, чтобы получить необходимое значение выхода, нужно подать на входы D_0, D_1, D_2, D_3 соответствующее постоянное значение.

Задача 6.4. С помощью управляемого мультиплексора синтезировать схему, реализующую БФ $Q = \sum(1, 2)$.

Запишем функцию Q в форме СДНФ:

$$Q = \bar{x}_2 x_1 + x_2 \bar{x}_1.$$

Если в формуле мультиплексора принять $E = 1$, $D_0 = D_3 = 0$, $D_1 = D_2 = 1$, то получим следующее:

$$f = ED_0\bar{x}_2\bar{x}_1 + ED_1\bar{x}_2x_1 + ED_2x_2\bar{x}_1 + ED_3x_2x_1 =$$
$$= 1\cdot 0\cdot \bar{x}_2\bar{x}_1 + 1\cdot 1\cdot \bar{x}_2x_1 + 1\cdot 1\cdot x_2\bar{x}_1 + 1\cdot 0\cdot x_2x_1 = \bar{x}_2x_1 + x_2\bar{x}_1 = Q.$$

Отсюда вытекает схема реализации функции Q (рис. 6.5).

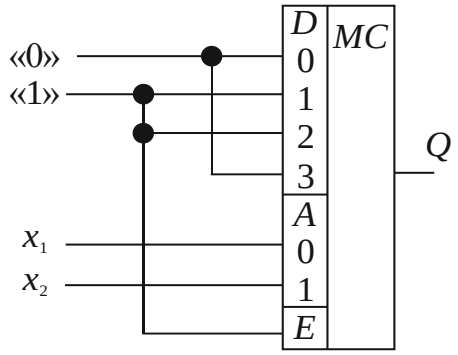


Рис. 6.5 – Реализация функции Q на мультиплексоре

Если переменные x_1x_2 принимают значения 00 или 11, то мультиплексор коммутирует на выход значения сигнала с входов 0 или 3, т. е. нулевые значения. Если x_1x_2 принимают значения 01 или 10, то на выход поступает сигнал высокого уровня с входов 1 или 2.

Задача 6.5. С помощью управляемого мультиплексора 8 на 1 синтезировать схему, реализующую БФ:

$$Z = \sum(1,\ 3,\ 4,\ 5,\ 7,\ 9,\ 12,\ 14,\ 15).$$

Функция Z зависит от четырех переменных, поэтому следует использовать два мультиплексора 8 на 1 с управляющими входами и собрать из них мультиплексор 16 на 1. Для реализации функции, по аналогии с задачей 6.4, на входы 1, 3, 4, 5, 7, 9, 12, 14, 15 этого мультиплексора нужно подать единицу, а на остальные входы – ноль. Решение задачи представлено на рисунке 6.6.

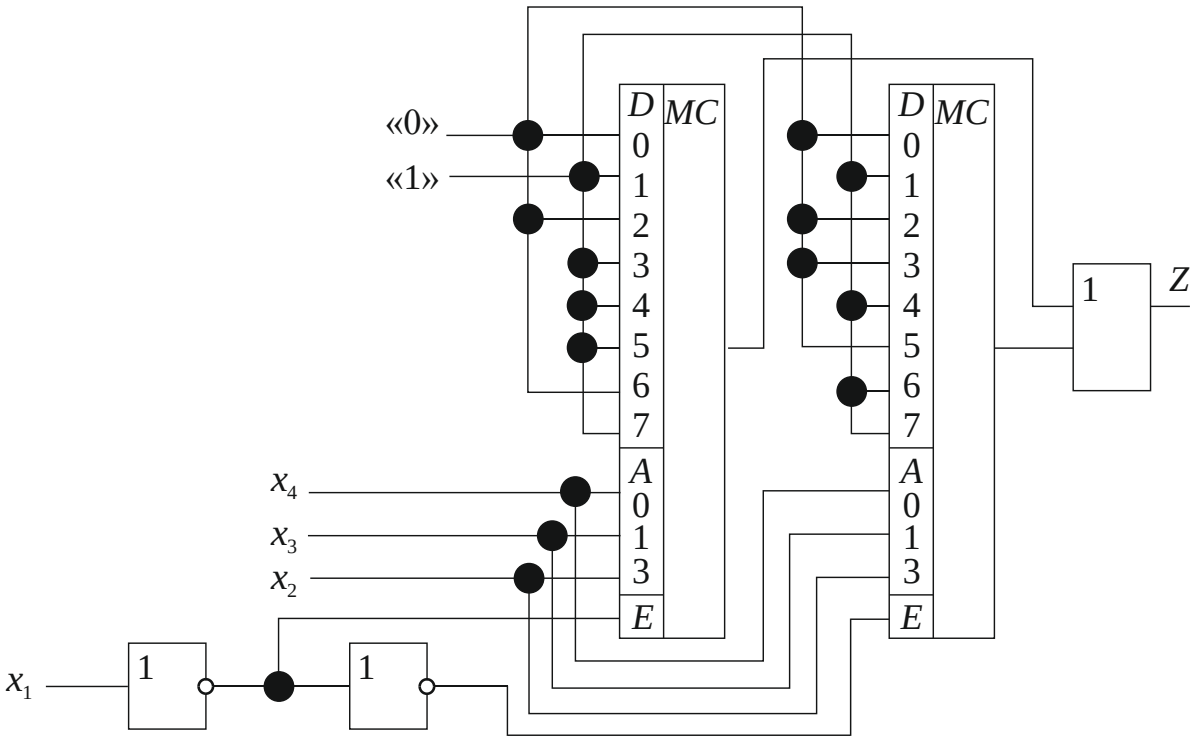


Рис. 6.6 – Реализация функции Z на мультиплексорах 8 на 1

Реализация БФ в задачах 6.4 и 6.5 была тривиальной. Эти задачи можно решить более экономичными способами, применяя разложение функций по переменным.

6.3 Разложение булевых функций

Любая БФ может быть преобразована к виду:

$$f(x_1, x_2, \ldots x_n) = x_1 \cdot f(x_2, \ldots x_n) + \overline{x}_1 \cdot f(x_2, \ldots x_n).$$

Например, разложим функцию f по переменной x_1:

$$f = x_1\overline{x}_2 + \overline{x}_1 x_2 = x_1(1 \cdot \overline{x}_2 + 0 \cdot x_2) + \overline{x}_1(0 \cdot \overline{x}_2 + 1 \cdot x_2) = x_1\overline{x}_2 + \overline{x}_1 x_2.$$

При вынесении за функцию переменной x_1 вместо нее в каждой конъюнкции ставится 1, при прямом значении переменной и 0 при инверсном значении. При вынесении за функцию инверсной переменной x_1 вместо нее в каждой конъюнкции ставится 0, при прямом значении переменной и 1 при инверсном значении.

В общем случае разложение может быть проведено для любого количества переменных k, где $k \le n$, и тогда БФ может быть представлена в виде:

$$f(x_1, x_2, \ldots x_n) = \overline{x}_1\overline{x}_2 \ldots \overline{x}_k f_0 + \overline{x}_1\overline{x}_2 \ldots \overline{x}_{k-1}x_k f_1 + \ldots + x_1 x_2 \ldots x_k f_{2^k-1},$$

где f_k – БФ, получаемая из исходной подстановкой в нее набора переменных, равного значению k. Например, разложение функции f по двум переменным:

$$f = \overline{x}_1\overline{x}_2 + x_1x_3 + x_2x_4 = \overline{x}_1\overline{x}_2 f_0 + \overline{x}_1x_2 f_1 + x_1\overline{x}_2 f_2 + x_1x_2 f_3 =$$

$$\text{разложение по } x_1 \text{ и } x_2 = \left\{\begin{array}{ll} x_1x_2 = 00, & f_0 = 1\cdot 1 + 0\cdot x_3 + 0\cdot x_4 = 1, \\ x_1x_2 = 01, & f_1 = 1\cdot 0 + 0\cdot x_3 + 1\cdot x_4 = 1\cdot x_4, \\ x_1x_2 = 10, & f_2 = 0\cdot 1 + 1\cdot x_3 + 0\cdot x_4 = x_3, \\ x_1x_2 = 11, & f_3 = 0\cdot 0 + 1\cdot x_3 + 1\cdot x_4 = x_3 + x_4 \end{array}\right\} =$$

$$= \overline{x}_1\overline{x}_2\cdot 1 + \overline{x}_1x_2\cdot x_4 + x_1\overline{x}_2\cdot x_3 + x_1x_2\cdot(x_3 + x_4).$$

Теперь *каждая конъюнкция* функции обязательно зависит от x_1 и x_2, чего нет в исходной функции. Разложение БФ используется в тех случаях, когда БФ должна быть представлена так, чтобы в каждой конъюнкции присутствовали необходимые переменные.

Рассмотрим ту часть функции f, разложенной по переменным x_1, x_2, которая повторяет общую функцию мультиплексора 4 на 1:

$$f = \overline{x}_1\overline{x}_2 + \overline{x}_1x_2x_4 + x_1\overline{x}_2x_3 + x_1x_2\cdot(x_3 + x_2).$$

$$f_{\text{мультиплексора}} = ED_0\overline{x}_2\overline{x}_1 + ED_1\overline{x}_2x_1 + ED_2x_2\overline{x}_1 + ED_3x_2x_1.$$

Очевидно, что для равенства этих функций необходимо выполнение:

$$E = 1, \quad D_0 = 1, \quad D_2 = x_4, \quad D_2 = x_3, \quad D_3 = x_3 + x_4.$$

Схема реализации функции *f* на мультиплексоре 4 на 1 изображена на рисунке 6.7. Для реализации дизъюнкции $x_3 + x_4$ в схему потребовалось ввести элемент ИЛИ.

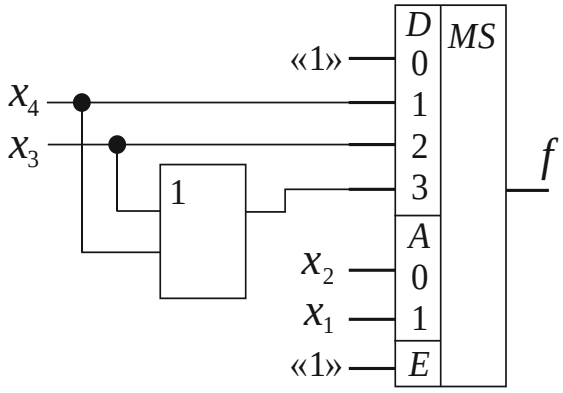


Рис. 6.7 – Реализация функции *f* при разложении по двум переменным

Задача 6.6. Синтезировать схему, реализующую БФ:

$$Z_1 = \sum(0,\ 1,\ 3,\ 6),$$

с помощью неуправляемого мультиплексора 4 на 1.

Запишем функцию в СДНФ и разложим ее по переменным x_1 и x_2:

$$Z_1 = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1x_2\overline{x}_3;$$
$$x_1x_2 = 00 \to F_0 = \overline{x}_3 + x_3 + 0 + 0 = 1;$$
$$x_1x_2 = 01 \to F_1 = 0 + 0 + x_3 + 0 = x_3;$$
$$x_1x_2 = 10 \to F_2 = 0 + 0 + 0 + 0 = 0;$$
$$x_1x_2 = 11 \to F_3 = 0 + 0 + 0 + \overline{x}_3 = \overline{x}_3;$$
$$Z_1 = \overline{x}_1\overline{x}_1F_0 + \overline{x}_1x_2F_1 + x_1\overline{x}_2F_2 + x_1x_2F_3.$$

В соответствии с разложением функции на входы мультиплексора D_0, D_1, D_2 и D_3 подаются сигналы, соответствующие функциям F_0, F_1, F_2, F_3 (рис. 6.8).

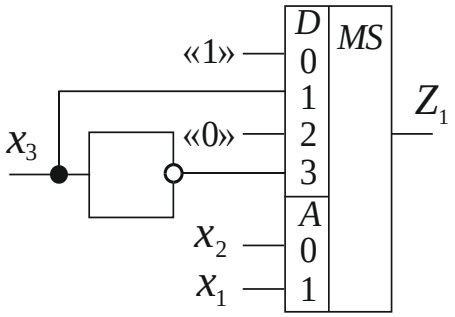


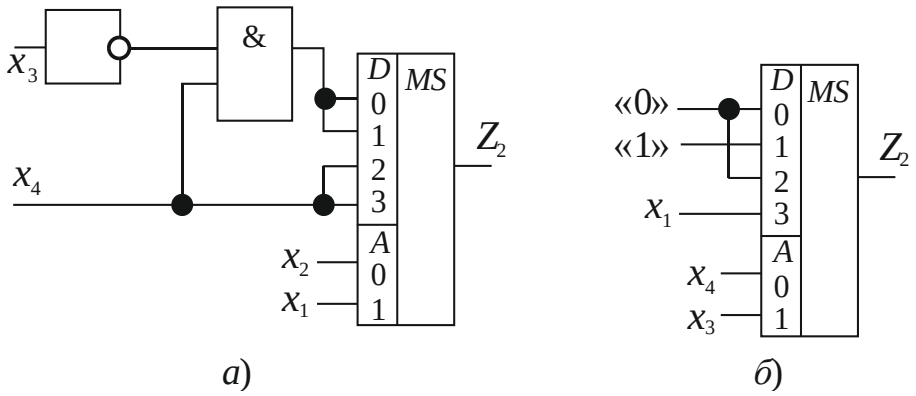
Рис. 6.8 – Реализация БФ Z_1

Задача 6.7. С помощью неуправляемого мультиплексора 8 на 1 синтезировать схему, реализующую БФ $Z_2 = \sum(1,\ 5,\ 9,\ 11,\ 13,\ 15)$.

Разложим функцию по переменным x_1 и x_2:

$$Z_2 = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2\overline{x}_3x_4 + x_1\overline{x}_2\overline{x}_3x_4 + x_1\overline{x}_2x_3x_4 + x_1x_2\overline{x}_3x_4 + x_1x_2x_3x_4;$$
$$x_1x_2 = 00 \to F_0 = \boxed{\overline{x}_3x_4};\quad x_1x_2 = 01 \to F_1 = \boxed{\overline{x}_3x_4};$$
$$x_1x_2 = 10 \to F_2 = \overline{x}_3x_4 + x_3x_4 = \boxed{x_4};\quad x_1x_2 = 11 \to F_3 = \overline{x}_3x_4 + x_3x_4 = \boxed{x_4};$$
$$Z_2 = \overline{x}_1\overline{x}_2(F_0) + \overline{x}_1x_2(F_1) + x_1\overline{x}_2(F_2) + x_1x_2(F_3) =$$
$$= \overline{x}_1\overline{x}_2(\underline{\overline{x}_3x_4}) + \overline{x}_1x_2(\underline{\overline{x}_3x_4}) + x_1\overline{x}_2(\underline{x_4}) + x_1x_2(\underline{x_4}).$$

Схема, реализующая Z_2, представлена на рисунке 6.9, *а*. При таком разложении пришлось ввести дополнительные логические элементы для реализации F_0 и F_1.



а) *б*)

Рис. 6.9 – Реализация БФ Z_1: при разложении по x_1 и x_2 (*а*); при разложении по x_3 и x_4 (*б*)

При разложении функции Z_2 по переменным x_3, x_4 получаем:

$$Z_2 = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2\overline{x}_3x_4 + x_1\overline{x}_2\overline{x}_3x_4 + x_1\overline{x}_2x_3x_4 + x_1x_2\overline{x}_3x_4 + x_1x_2x_3x_4;$$

$$x_3x_4 = 00 \to F_0 = \boxed{0}; \quad x_3x_4 = 01 \to F_1 = \overline{x}_1\overline{x}_2 + \overline{x}_1x_2 + x_1\overline{x}_2 + x_1x_2 = \boxed{1};$$

$$x_3x_4 = 10 \to F_2 = \boxed{0}; \quad x_3x_4 = 11 \to F_3 = x_1\overline{x}_2 + x_1x_2 = \boxed{x_1};$$

$$Z_2 = \overline{x}_1\overline{x}_2(F_0) + \overline{x}_1x_2(F_1) + x_1\overline{x}_2(F_2) + x_1x_2(F_3) =$$

$$= \overline{x}_1\overline{x}_2(0) + \overline{x}_1x_2(1) + x_1\overline{x}_2(0) + x_1x_2(x_1).$$

По сравнению с разложением по x_1 и x_2, это разложение дает лучший результат, и схема становится проще (рис. 6.9, *б*).

Если БФ зависит от значительно большего числа переменных, чем число адресных входов мультиплексора, то схема реализуется каскадным включением мультиплексоров.

Задача 6.8. С помощью мультиплексоров 4 на 1 синтезировать схему, реализующую БФ $P = \sum(0,\ 2,\ 4,\ 6,\ 9,\ 12,\ 13,\ 14,\ 15)$.

Проведем разложение функции по x_1, которая будет подаваться на стробирующие (управляющие) входы мультиплексоров:

$$P = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3x_4 +$$

$$+x_1x_2\overline{x}_3\overline{x}_4 + x_1x_2\overline{x}_3x_4 + x_1x_2x_3\overline{x}_4 + x_1x_2x_3x_4;$$

$$x_1 = 0 \to F_0 = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_2x_3\overline{x}_4 + x_2\overline{x}_3\overline{x}_4 + x_2x_3\overline{x}_4;$$

$$x_1 = 1 \to F_1 = \overline{x}_2\overline{x}_3x_4 + x_2\overline{x}_3\overline{x}_4 + x_2\overline{x}_3x_4 + x_2x_3\overline{x}_4 + x_2x_3x_4;$$

$$P = \overline{x}_1F_0 + x_1F_1 = \overline{x}_1\left(\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_2x_3\overline{x}_4 + x_2\overline{x}_3\overline{x}_4 + x_2x_3\overline{x}_4\right) +$$

$$+x_1\left(\overline{x}_2\overline{x}_3x_4 + x_2\overline{x}_3\overline{x}_4 + x_2\overline{x}_3x_4 + x_2x_3\overline{x}_4 + x_2x_3x_4\right).$$

В свою очередь функции F_0 и F_1 разложим по переменным x_2 и x_3:

$$F_0 = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_2 x_3\overline{x}_4 + x_2\overline{x}_3\overline{x}_4 + x_2 x_3\overline{x}_4;$$

$$x_2x_3 = 00 \to K_0 = \boxed{\overline{x}_4}; \quad x_2x_3 = 01 \to K_1 = \boxed{\overline{x}_4};$$

$$x_2x_3 = 10 \to K_2 = \boxed{\overline{x}_4}; \quad x_2x_3 = 11 \to \boxed{\overline{x}_4};$$

$$F_0 = \overline{x}_2\overline{x}_3(\underline{\overline{x}_4}) + \overline{x}_2 x_3(\underline{\overline{x}_4}) + x_2\overline{x}_3(\underline{\overline{x}_4}) + x_2 x_3(\underline{\overline{x}_4});$$

$$F_1 = \overline{x}_2\overline{x}_3 x_4 + x_2\overline{x}_3\overline{x}_4 + x_2\overline{x}_3 x_4 + x_2 x_3\overline{x}_4 + x_2 x_3 x_4;$$

$$x_2x_3 = 00 \to D_0 = \boxed{x_4}; \quad x_2x_3 = 01 \to D_1 = \boxed{0};$$

$$x_2x_3 = 10 \to D_2 = \overline{x}_4 + x_4 = \boxed{1};$$

$$x_2x_3 = 11 \to D_4 = \overline{x}_4 + x_4 = \boxed{1};$$

$$F_1 = \overline{x}_2\overline{x}_3(\underline{x_4}) + \overline{x}_2 x_3(\underline{0}) + x_2\overline{x}_3(\underline{1}) + x_2 x_3(\underline{1}).$$

Реализация функции P на управляемых мультиплексорах представлена на рисунке 6.10.

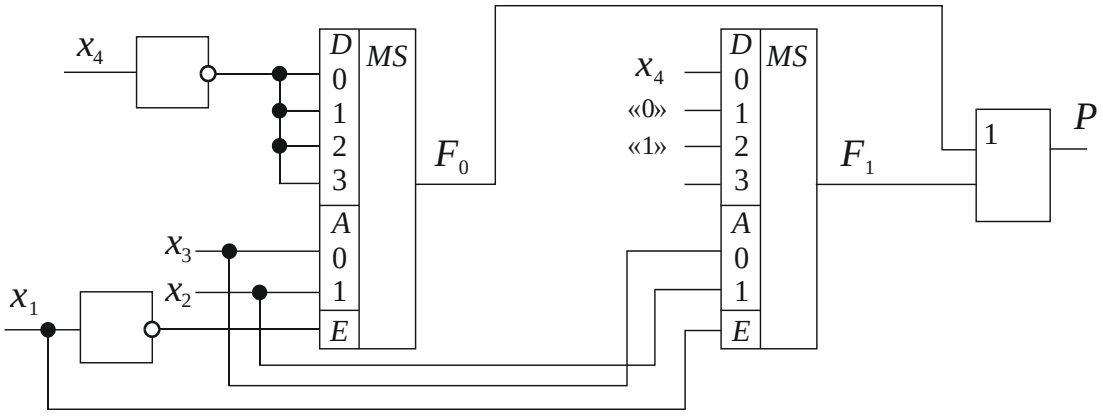
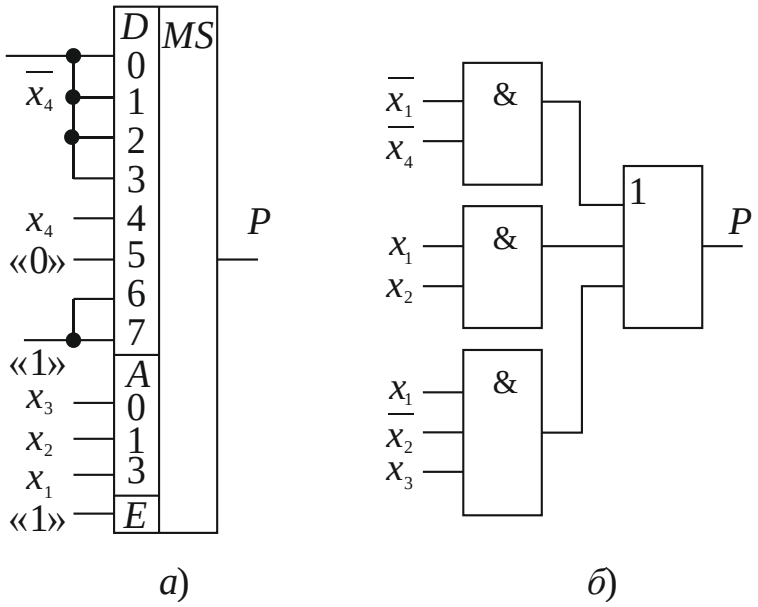


Рис. 6.10 – Реализация функции *P* на управляемых мультиплексорах

При разложении F_0 и F_1 по другим переменным, например x_3 и x_4, получится другой результат и, соответственно, изменится схема. Следовательно, разложение БФ по переменным является комбинаторной задачей. Отсюда вытекает, что следует разложить функцию по разным переменным и выбрать наиболее простой в реализации на мультиплексорах результат.

Если функцию P разложить по переменным $x_1x_2x_3$ и реализовать ее на мультиплексоре 8 на 1, то получим схему, представленную на рисунке 6.11, *а*. При реализация этой функции на дискретных элементах получается схема, представленная на рисунке 6.11, *б*.

Функция одна, результаты реализации разные. Поэтому выбор того, какую схему выбрать для реализации в виде устройства, остается за разработчиком схемы.



а) *б*)

Рис. 6.11 – Реализация БФ P
при разложении по $x_1x_2x_3$ (*а*); на дискретных элементах (*б*)

Контрольные вопросы по главе 6

1. Можно ли на неуправляемом дешифраторе трех переменных реализовать БФ от четырех переменных?
2. Как реализовать БФ, заданную в форме КНФ, с помощью дешифратора?
3. Можно ли на одном дешифраторе реализовать несколько БФ?
4. Можно ли на одном мультиплексоре реализовать несколько БФ?
5. Если булеву функцию от четырех переменных разложить по трем переменным, то какой мультиплексор потребуется для реализации этой БФ?

7 Цифровые последовательностные элементы и устройства

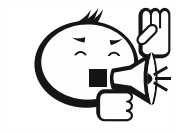
Мы рассматривали комбинационные схемы. Это такие устройства, выходные сигналы которых в текущий момент времени t зависят только от входных сигналов в этот же момент времени. *Последовательностные* цифровые устройства характеризуются тем, что выходные сигналы зависят не только от текущих значений входных сигналов в данный момент времени t_i, но и от последовательности значений входных сигналов, поступивших на входы в предшествующий момент времени t_{i-1}. В состав таких устройств должны входить *элементарные элементы памяти* (*элементарные автоматы*), которые хранят состояние устройства, достигнутое им в результате действия последовательности предыдущих входных сигналов. В качестве таких элементов памяти в электронных цифровых устройствах используются триггеры.

7.1 Асинхронные триггеры

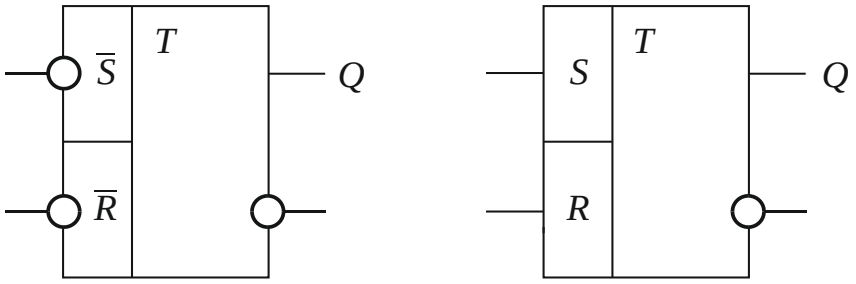
Триггер *– это устройство, имеющее два устойчивых состояния, в каждом из которых он может находиться бесконечно долго. Одно из состояний обозначается единицей, другое – нулем.*

Таким образом, триггер – это устройство, которое может хранить один бит данных. Триггер является элементарным автоматом. На них строятся цифровые устройства с памятью, называемые автоматами. Автомат – это устройство, выполняющее определенную, заранее заданную, последовательность действий.

Простейшим триггером, на основе которого строятся все остальные, является триггер типа *R*-*S* (можно называть и *S*-*R*). На рисунке 7.1 приведены обозначения на схемах триггеров типа *R*-*S*. Выход *Q* называется прямым, а второй выход инверсным. Значение инверсного выхода *всегда должно быть противоположно значению* на прямом выходе.



При анализе работы схем, включающих в себя триггеры, рассуждения всегда проводятся по отношению к прямому выходу.



а) *б*)

Рис. 7.1 – Обозначение триггеров типа *R-S*:
с инверсными входами (*а*); с прямыми входами (*б*)

Триггеры отличаются друг от друга законами управления, которые задаются таблицами переходов (табл. 7.1).

Таблица 7.1 – Таблицы переходов триггеров *R-S*

Триггер с инверсными входами управления		
S'	*R'*	*Q*
1	1	Хранение
0	1	1
1	0	0
0	0	X

Триггер с прямыми входами управления		
S	*R*	*Q*
1	1	X
0	1	0
1	0	1
0	0	Хранение

X обозначает запрещенную комбинацию входных сигналов, хранение – значение выхода триггера не изменяется.

Из таблиц следует, что в неактивном состоянии на входы триггеров всегда должны быть поданы сигналы, соответствующие режиму хранения. Затем внешними схемами организуется проявление сигналов для перехода триггера в 0 или 1, а после этого входы опять переводятся в соответствие с режимом хранения.

Схема триггера *R-S* с инверсными входами управления приведена на рисунке 7.2.

Триггер собран на двух элементах И-НЕ. При единичном значении сигналов на входах *S'* и *R'* триггер будет находиться в том состоянии, в котором он был перед этим (режим хранения). В этом состоянии триггер может находиться

сколь угодно долго, т. е. до тех пор, пока на входе R' или входе S' не появится активный сигнал 0.

Вход S' для установки в единичное состояние получил название от слова set – установка.

Вход R' для установки в нулевое состояние получил название от слова reset – сброс в ноль.

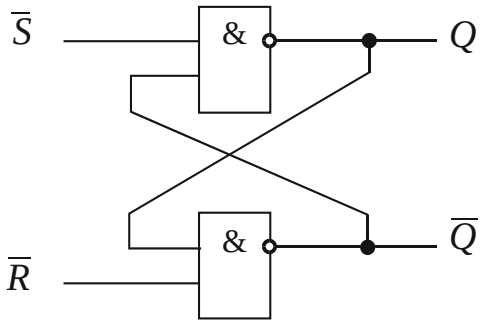


Рис. 7.2 – Схема триггера типа *R-S* с инверсными входами

Запрещенная комбинация входов $S' = R' = 0$ приведет к тому, что на обоих выходах появится единичный сигнал ($Q = 1$, $Q' = 1$), что нарушает понятие прямого и инверсного выходов. Кроме того, точно неизвестно, в какое состояние переключится триггер, если после этого на его входы одновременно подать неактивные сигналы $S' = R' = 1$.

На рисунке 7.3 приведена схема триггера *R-S* с прямыми входами. Триггер строится на элементах ИЛИ-НЕ, для которых активным является единичный входной сигнал. Его появление на любом входе элемента ИЛИ-НЕ обязательно вызывает появление нуля на выходе.

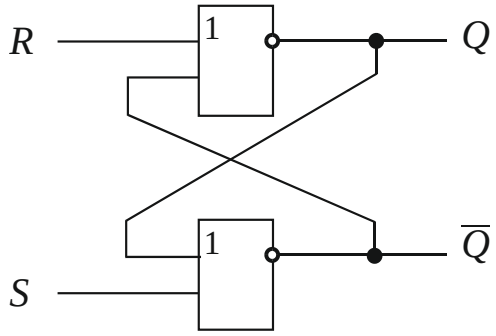


Рис. 7.3 – Схема триггера типа *R-S* с прямыми входами

На рисунке 7.4 приведена диаграмма работы *R-S* триггера с инверсными входами. В начальном состоянии триггер находился в единичном состоянии, а затем переключался, в соответствии с сигналами на его входах.

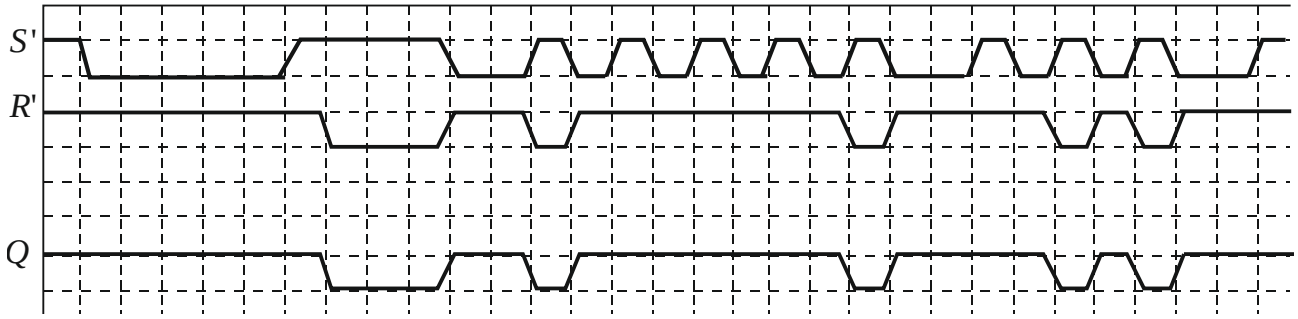


Рис. 7.4 – Диаграмма работы триггера с инверсными входами *R-S*

Триггеры типа *R-S* относятся к *асинхронным* триггерам, потому что появление входных сигналов активного уровня в *любой момент времени* тут же вызывает переход триггера в новое состояние, в соответствии с таблицей переходов.

На триггерах типа *R-S* строятся асинхронные автоматы. Их основным недостатком является эффект гонок. Сигналы управления, поступающие на входы триггеров, формируются комбинационными схемами. Комбинационные схемы, формирующие сигналы для входа *R* и входа *S*, строятся из цепочек логических элементов, имеющих различную длину. В результате этого сигналы управления могут запаздывать относительно друг друга, что приведет к неправильному срабатыванию триггеров, т. е. к сбою в работе устройства.

7.2 Синхронные триггеры

Избавиться от эффекта гонок можно в том случае, если заставить триггеры переключаться в тот момент времени, когда значения всех управляющих сигналов гарантированно сформированы. В этом случае в схему вводят генератор синхронизирующих (тактовых) импульсов. Триггеры, работающие под управлением синхроимпульсов, называются синхронными или синхронизированными.

7.2.1 Триггеры типа *D*

D-триггеры бывают разных видов и обозначается на схемах так, как показано на рисунке 7.5.

D-триггер (рис. 7.5, *а*) имеет два инверсных асинхронных входа – *R'* и *S'* – для возможности предварительной установки триггера в нулевое или единичное состояние. На вход *D* подается информационный сигнал 0 или 1, которые по перепаду из 0 в 1 на входе *C* записывается в триггер. Вход *C* называется *синхронизирующим* входом или *синхровходом*, вход *D* называется входом *данных*. Входы *R'-S'* для этого триггера являются *приоритетными* по сравнению с входом *D*.

Закон работы триггера *можно сформулировать так: по активному фронту синхроимпульса* (*в данном случае переход из 0 в 1*) *на входе C данные, находящиеся на входе D, записываются в триггер и появляются на его выходе.*

Этот закон представлен в таблице переходов (табл. 7.2).

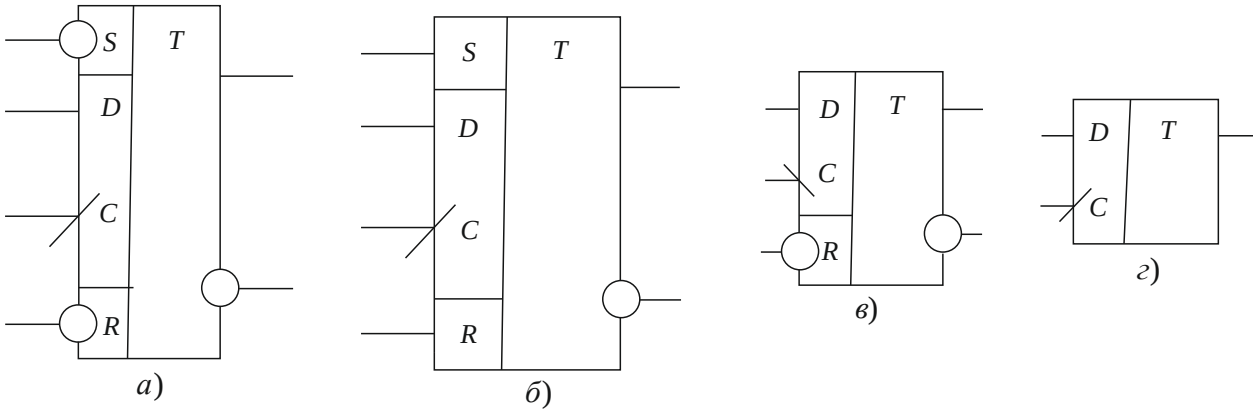


Рис. 7.5 – Виды и обозначения *D*-триггеров

Таблица 7.2 – Таблица переходов *D*-триггера

S	***R***	***D***	***C***	***Q***
0	0	Запрещенная комбинация		
0	1	X	X	1
1	0	X	X	0
1	1	X	0 или 1	*Q*
		0		0
		1		1

X – безразлично 0 или 1, *Q* – не изменяет своего состояния – хранение данных, 0 или 1 – на входе *C* постоянный уровень напряжения.

Надо отметить, что существует несколько схем *D*-триггеров. Есть триггеры без асинхронных инверсных входов *R'* и *S'* (рис. 7.5, *г*), с асинхронными

прямыми входами *R* и *S* (рис. 7.5, *б*), только с одним асинхронным инверсным входом *R* (рис. 7.5, *в*), с синхровходом *C*, реагирующим на перепад сигнала из 1 в 0 (рис. 7.5, *в*).

На рисунке 7.6 приведена диаграмма работы *D*-триггера, реагирующего на перепад синхросигнала из 0 в 1 и с начальным состоянием триггера *Q*, равным 1. На диаграмме хорошо видно, что по каждому синхросигналу (перепад на входе *C* из 0 в 1) на выходе *Q* появляются данные, находящиеся на входе *D*.

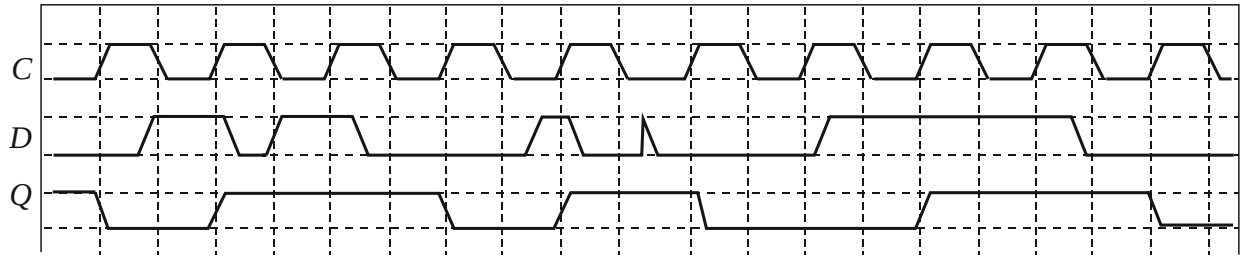


Рис. 7.6 – Диаграмма работы *D*-триггера

7.2.2 Триггеры типа *J-K*

Триггеры типа *J-K* относятся к синхронизируемым триггерам и разрабатывались как усовершенствование *D*-триггера.

Обозначение простого триггера *J-K* и *универсального* триггера *J-K* на схемах показано на рисунке 7.7, *а* и рисунке 7.7, *б* соответственно.

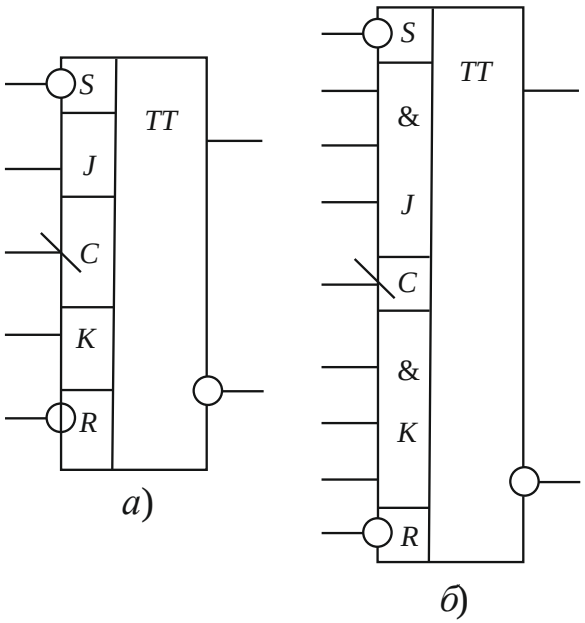


Рис. 7.7 – Обозначение *J-K*-триггера на схемах:
простой *J-K* триггер (*а*); универсальный *J-K* триггер (*б*)

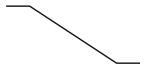
Триггер имеет асинхронные установочные входы S' – для записи единицы, R' – для записи нуля, синхровход C (на перепад из 1 в 0) и информационные входы J и K. Информационные входы J и K могут строиться и как одновходовые, и как трехвходовые схемы И, на которых можно выполнять предварительную обработку входных сигналов. Входы S' и R' имеют приоритет по сравнению с входами J, K.

Если перед перепадом сигнала из 1 в 0 на входе C:

1) вход $J = 1$, $K = 0$, то триггер устанавливается в 1;
2) вход $J = 0$, $K = 1$, то триггер устанавливается в 0;
3) входы $J = K = 1$, то триггер изменяет свое состояние на противоположное;
4) входы $J = K = 0$, режим хранения – триггер остается в том состоянии, в котором находился перед этим.

Таблица переходов триггера представлена в таблице 7.3.

Таблица 7.3 – Таблица переходов J-K-триггера

S	***R***	***J***	***K***	***C***	***Q***
0	0	Запрещенная комбинация			
0	1	X	X	X	1
1	0	X	X	X	0
1	1	X	X	0 или 1	Q
		0	1		0
		1	0		1
		1	1		$\overline{Q}$

X – безразлично 0 или 1, Q – не изменяет своего состояния – хранение данных, 0 или 1 – на входе C постоянный уровень напряжения, C – вход – перепад из 1 в 0 при работе со входами J и K.

На рисунке 7.8 представлена диаграмма работы J-K-триггера, на которой отмечены перепады 1 в 0 синхроимпульсов и переход по нему выхода Q, в зависимости от состояния входов J и K.

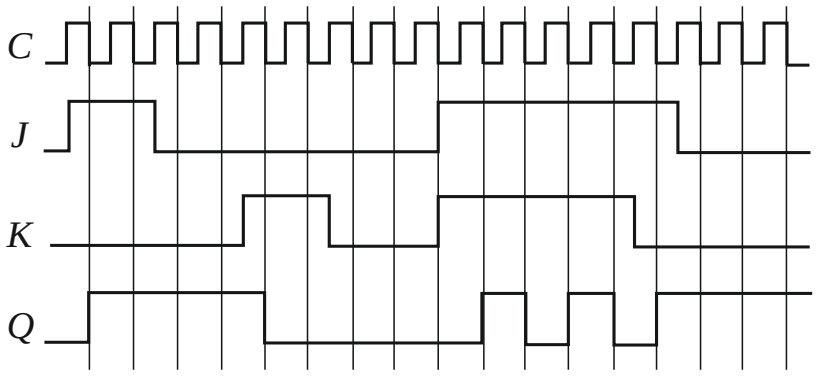
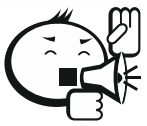


Рис. 7.8 – Диаграмма работы *J-K*-триггера



Следует запомнить: при работе триггера *D* и триггера *J-K* данные на информационные входы *D, J, K* должны быть поданы до того, как на входе *C* появится соответствующий фронт синхроимпульса.

7.3 Регистры

Регистрами называются устройства для хранения данных. Регистры состоят из триггеров, имеющих общие цепи управления. Количество триггеров определяет разрядность регистра. Регистры, выпускаемые в виде микросхем, обычно состоят из 4, 8 разрядов, реже их разрядность может быть произвольной (12, 16). В подавляющем числе случаев регистры строятся на триггерах типа *D*. Регистры не только хранят данные, но и могут выполнять логические операции сдвига.

На рисунке 7.9, *а* представлена схема четырехразрядного регистра с общим сбросом *R'* и обозначение такого регистра на схемах (рис. 7.9, *б*). Регистр состоит из четырех *D*-триггеров, у которых объединены входы *C* и инверсные входы сброса в ноль *R'*. (Инверсные выходы триггеров и входы *S* не используются, поэтому они не выведены за контур схемы.) Данные на входы $D_0..D_3$ должны быть поданы одновременно и после этого с некоторой задержкой на вход *C* подается синхроимпульс записи (перепад напряжения из 0 в 1). Данные записываются в триггеры регистра и появляются на прямых выходах *Q*0..*Q*4. Такие регистры называются регистрами с параллельной записью.

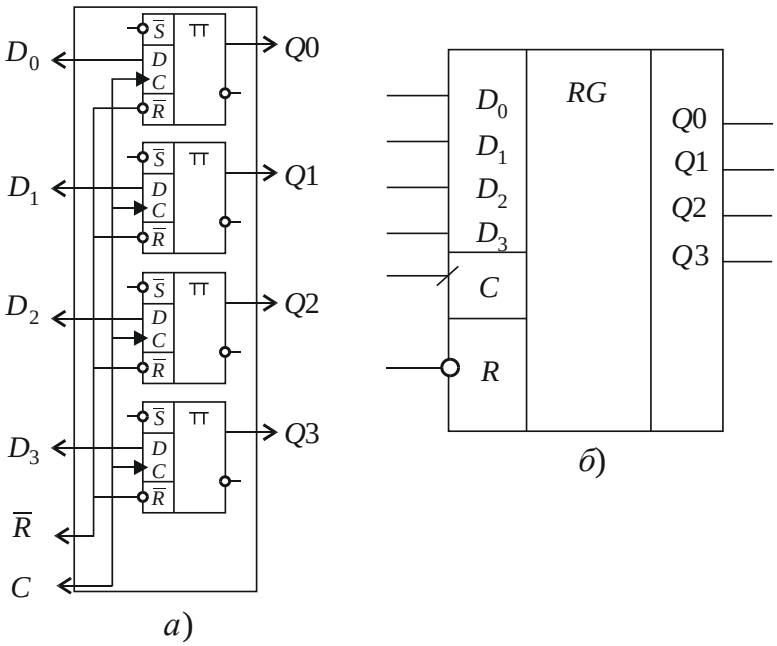


Рис. 7.9 – Четырехразрядный регистр с прямыми выходами
и с параллельной записью: схема регистра (*а*);
обозначение регистра на схемах (*б*)

На рисунке 7.10 приведено обозначение восьмиразрядных регистров с инверсными выходами. На рисунке 7.10, *а* приведено обозначение регистра с синхровходом *C* и общим сбросом в нулевое состояние *R'*. Этот регистр состоит из восьми *D*-триггеров, у которых выведены инверсные выходы. На рисунке 7.10, *б* приведено обозначение регистра с синхровходом *C* и возможностью перевода выходов в состояние *Z* по нулевому сигналу *EZ*.

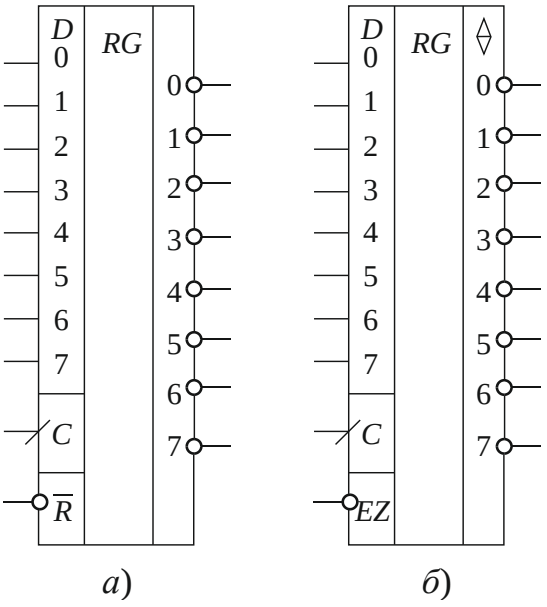


Рис. 7.10 – Восьмиразрядные регистры:
регистр с общим сбросом (*а*); регистр с *Z*-состоянием выходов (*б*)

Третье состояние выходов дает возможность их подключения напрямую к шине.

7.4 Регистры сдвига

Регистр сдвига, или сдвигающий регистр, – это регистр, в котором кроме хранения реализуется операция сдвига данных влево или вправо. Вправо – в сторону старших разрядов числа, влево – в сторону младших. Регистр, в котором реализуются обе операции, называется *реверсивным* сдвигающим регистром.

Операция сдвига выполняется так, как представлено в таблице 7.4, а на рисунке 7.11, *а* изображена схема такого регистра.

Таблица 7.4 – Операция сдвига вправо в четырехразрядном регистре с прямыми выходами

	Сдвиг вправо				
	DR	*Q*0	*Q*1	*Q*2	*Q*3
Исходное состояние регистра	**0**	**0**	**1**	**0**	**1**
Первый сдвиг вправо	0	0	0	1	0
Второй сдвиг вправо	1	1	0	0	1
Третий сдвиг вправо	0	0	1	0	0
Четвертый сдвиг вправо	1	1	0	1	0
Пятый сдвиг вправо	0	0	1	0	1

Для выполнения сдвига вправо выход каждого *i*-го *D*-триггера соединен со входом *D* *i*+1-го *D*-триггера. Вход *DR* предназначен для того, чтобы в триггер нулевого разряда, из которого данные перепишутся в первый разряд, можно было занести данные из внешнего источника. На вход *R'* в начале работы подается импульс сброса, по которому весь регистр переводится в начальное *нулевое* состояние. На входы *C* всех триггеров подается импульс, которым инициируется перепись в *i*+1-й разряд содержимого *i*-го разряда. Операцию сдвига легко проследить по таблице 7.4.

Следует заметить, что содержимое *старшего* разряда *Q*3 при сдвигах выталкивается из регистра и *пропадает*.

Сдвиг влево реализуется по схеме (рис. 7.11, *б*). Здесь выход *i*+1-го разряда подается на вход *D* *i*-го разряда, т. е. данные с выходов старших разрядов передвигаются в сторону младших разрядов. Состояние старшего разряда изменяется в соответствии с содержимым входа *DL*, на который подаются данные от внешнего источника.

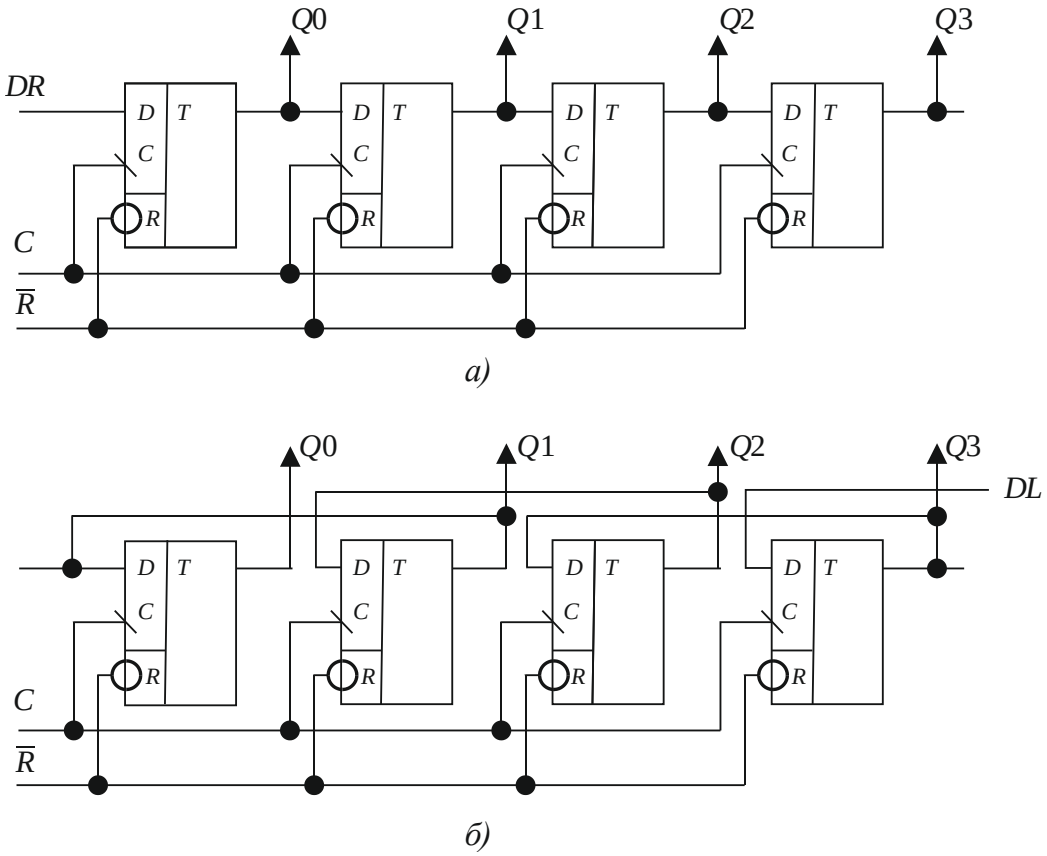


Рис. 7.11 – Четырехразрядный регистр сдвига:
сдвиг вправо (*а*); сдвиг влево (*б*)

В таблице 7.5 приведено выполнение операции сдвига влево. Исходное состояние регистра $Q0 – Q1 – Q2 – Q3 = 1100$, на вход *DL* подан 0.

Таблица 7.5 – Операция сдвига влево в четырехразрядном регистре

	Сдвиг влево				
	*Q*0	*Q*1	*Q*2	*Q*3	*DL*
Исходное состояние регистра	**0**	**1**	**1**	**0**	**0**
Первый сдвиг влево	1	1	0	0	1
Второй сдвиг влево	1	0	0	1	1
Третий сдвиг влево	0	0	1	1	0
Четвертый сдвиг влево	0	1	1	0	0
Пятый сдвиг влево	1	1	0	0	1

В реверсивный регистр добавлены логические цепи, которые коммутируют выходы регистров для подключения их к следующим или предыдущим разрядам (рис. 7.12). Входы сброса в ноль *R* и синхронизации *C* являются общими для всех триггеров. Элемент *DD*1 по единичному сигналу *WL* (влево) разрешает передачу данных из триггера $i+1$-го разряда в i-й разряд. Элемент *DD*2 по единичному сигналу *WR* (вправо) разрешает передачу данных из триггера i-го разряда в $i+1$-й разряд. Очевидно, что сигналы *WR* и *WL* не должны

одновременно быть разрешающими, т. е. $WR = WL = 1$ являются запрещенной комбинацией.

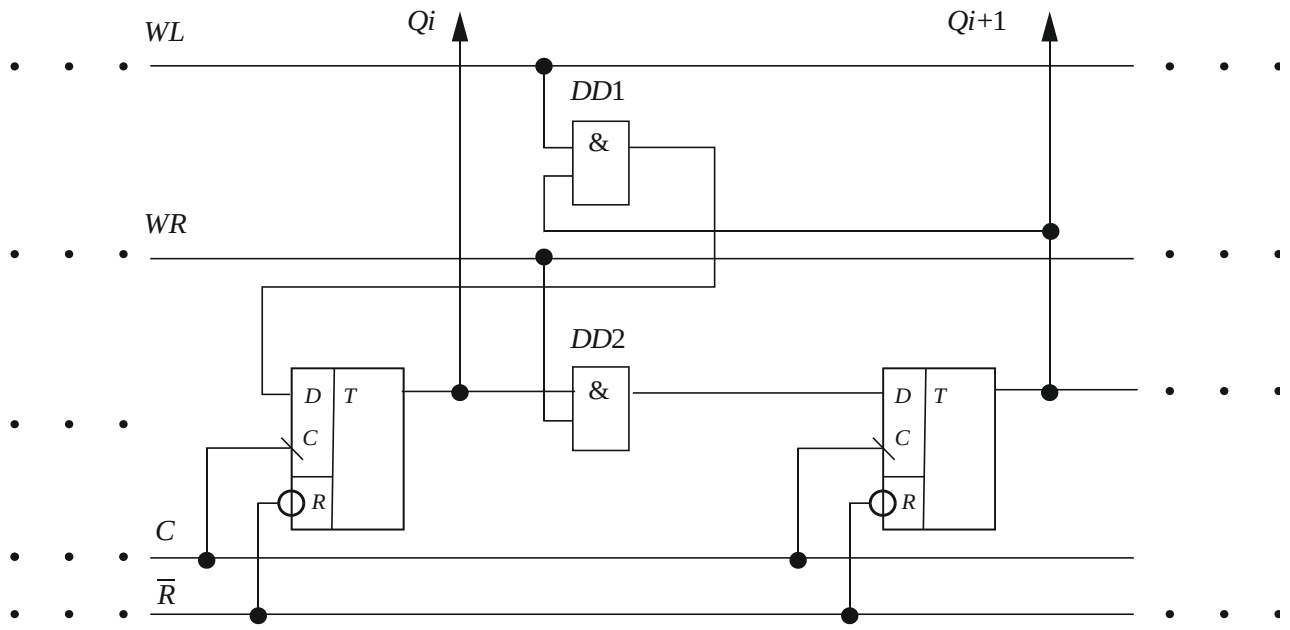


Рис. 7.12 – Схема цепей для осуществления реверсного сдвига

Разработано большое количество схем сдвигающих регистров. Например, на рисунке 7.13 показано обозначение универсального регистра.

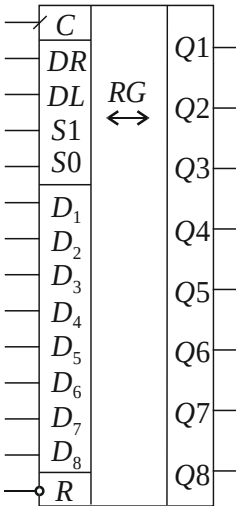


Рис. 7.13 – Универсальный реверсивный сдвигающий регистр

Этот регистр предназначен для выполнения нескольких операций. Вход R – для сброса всех триггеров в нулевое состояние. На вход C подается синхроимпульс с активным фронтом 0–1. На входы $D_1..D_8$ подается код 8-разрядного числа, который параллельно (все сразу) записывается в триггеры регистра по активному фронту на входе C. Входы DR, DL для подачи данных в освобождающиеся триггеры при сдвиге вправо или влево соответственно. И, наконец,

входы управления $S0$, $S1$, задающие четыре режима работы регистра. Режим хранения задается при $S0 = S1 = 0$. В этом режиме состояния триггеров регистра неизменны (можно только сбросить регистр в ноль сигналом сброса R'). Данные с входов $D_1..D_8$ по фронту сигнала C записываются в регистр (режим параллельной записи) при $S0 = S1 = 1$. Режим сдвига влево или вправо возникает при $S0S1 = 01$ или $S0S1 = 10$.

К выходам регистров можно подключать любые цифровые устройства, которые обрабатывают данные, хранящиеся в регистре. Используются регистры как устройства временного хранения данных.

Подключая регистры к сумматору или АЛУ, получают регистровые сумматоры и регистровые АЛУ (РАЛУ). Например, можно создать схему накапливающего сумматора.

В накапливающий сумматор (рис. 7.14) входят регистр суммы, в котором содержится накопленная сумма чисел. Регистр А предназначен для хранения очередного операнда. На сумматор поступает одно число из регистра А и второе число с выхода регистра суммы. НС работает следующим образом:

- регистр суммы обнуляется сигналом «Сброс в 0»;
- в регистр А записывается очередное число;
- в сумматоре появляется сумма чисел из регистра А и содержимого регистра суммы;
- синхроимпульсом записи результат суммирования записывается в регистр суммы;
- следующее число опять записывается в регистр А;
- этот процесс продолжается, и в регистре суммы накапливается сумма чисел, последовательно записываемых в регистр А;
- в нужный момент процесс прекращается и из регистра суммы можно считать результат сложения.

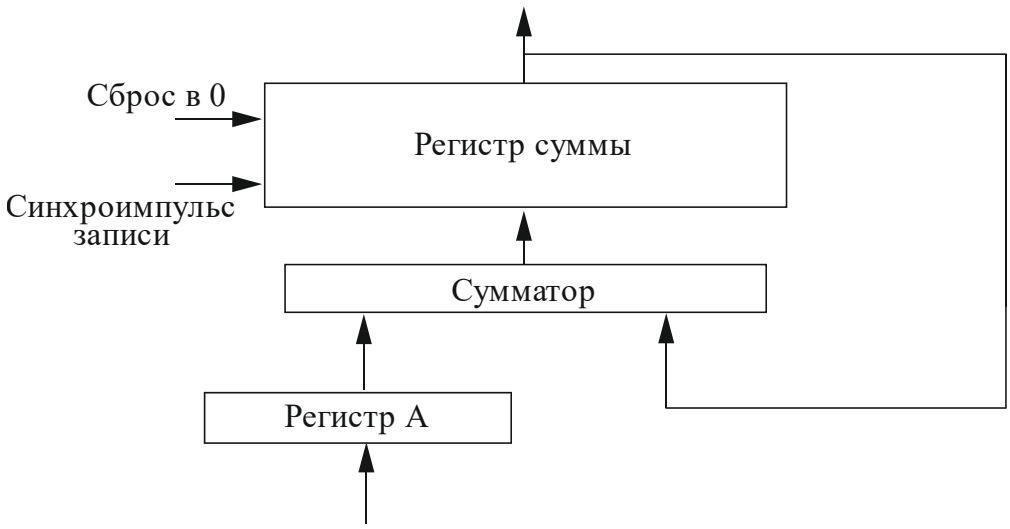


Рис. 7.14 – Структура накапливающего сумматора

Контрольные вопросы по главе 7

1. Какие выходы имеет триггер?
2. На какой вход нужно подать активный сигнал, чтобы триггер установился в единичное состояние?
3. На какой вход нужно подать активный сигнал, чтобы триггер установился в нулевое состояние?
4. Как называется устройство, состоящее из триггеров с общими цепями управления?
5. Какие операции выполняются в реверсивном сдвигающем регистре?
6. Можно ли объединить одноименные выходы регистров, выходные схемы которых могут переводиться в *Z*-состояние?

8 Счетчики

Счетчик *– это устройство, выполняющее операцию счета. Счет выполняется в двоичном коде. Операция счета заключается в прибавлении единицы к числу или вычитанию единицы из числа.*

Устройства, выполняющие операцию счета в двоичном коде, называются двоичными счетчиками. Счетчики, увеличивающие свое состояние на единицу, называются *накапливающими* или *суммирующими*. Счетчики, уменьшающие свое состояние на единицу, называются *вычитающими*. Счетчики, умеющие выполнять обе операции, называются *реверсивными* счетчиками.

Запоминающим элементом счетчика и является триггер. Счетчик должен помнить свое состояние и по приходу управляющего сигнала изменять свое состояние на единицу в большую или меньшую сторону, т. е. счетчик является автоматом.

Количество триггеров в схеме счетчика определяет его разрядность. Если счетчик содержит N разрядов, то он имеет 2^N состояний и максимальное число, которое может хранить счетчик, равно $2^N - 1$.

Накапливающий счетчик в начале работы, как правило, устанавливается в нулевое состояние, т. е. во всех его разрядах записывается ноль. По мере работы состояние увеличивается на единицу до достижения $2^N - 1$. После очередного прибавления единицы счетчик переходит опять в нулевое состояние и формирует сигнал переноса, который информирует о том, что счетчик достиг своего предельного значения.

8.1 Работа двоичного счетчика

Рассмотрим работу трехразрядных счетчиков, накапливающего и вычитающего. Количество состояний каждого счетчика $2^3 = 8$. На рисунке 8.1 показан цикл работы каждого счетчика. Накапливающий начинает с состояния 000 и заканчивает цикл состоянием 111. Затем он опять переходит в начальное состояние и формирует перенос. Вычитающий счетчик, наоборот, начинает с 111 и заканчивает цикл 000 с формированием переноса. Посмотрите, таблицы полностью совпадают с левой частью таблиц истинности БФ от трех переменных. Та-

ким образом, при необходимости счетчик может последовательно формировать сигналы, которые можно интерпретировать как булевы переменные и использовать в качестве входных сигналов для комбинационных схем.

Накапливающий счетчик	
Перенос	Разряды счетчика
0	000
0	001
0	010
0	011
0	100
0	101
0	110
0	111
+1	000

Цикл работы счетчика

а)

Вычитающий счетчик	
Перенос	Разряды счетчика
0	111
0	110
0	101
0	100
0	011
0	010
0	001
0	000
+1	111

Цикл работы счетчика

б)

Рис. 8.1 – Таблицы переходов трехразрядных счетчиков: накапливающий счетчик (*а*); вычитающий счетчик (*б*)

8.2 Счетчик с последовательным переносом

В схеме на рисунке 8.2 генератор *G* формируют импульсы, которые поступают на синхровход *C* триггера типа *J-K*. Триггер включен так, что изменяет свое состояние на противоположное на каждый задний фронт импульса синхрогенератора. *Говорят, что триггер работает в счетном режиме.*

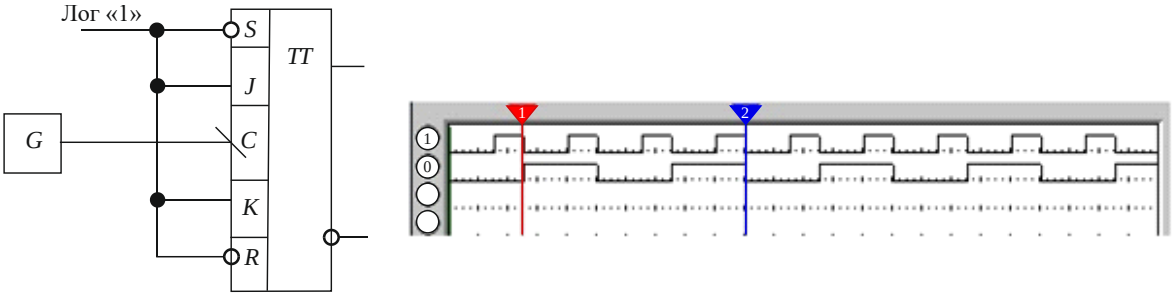


Рис. 8.2 – Триггер типа *J-K* в счетном режиме

Входы *R'* и *S'* инверсные, поэтому на них подается логическая единица с источника. На входы *J* и *K* также подается логическая единица. Из таблицы переходов *J-K* триггера следует, что триггер на каждый синхросигнал на входе *C* будет изменять свое состояние на противоположное. Выход генератора и прямой выход триггера подключены к многоканальному прибору (логический анализатор), с помощью которого удобно рассматривать работу логических

устройств в динамике. Верхний луч логического анализатора показывает импульсы с генератора, а второй – импульсы с прямого выхода триггера. На диаграмме сигналов хорошо видно, что триггер меняет свое состояние на противоположное на каждый задний фронт синхроимпульса, в соответствии с законом его работы. По диаграмме работы видно, что частота импульсов генератора на *C* входе триггера в два раза выше частоты импульсов на выходе триггера. То есть триггер делит входную частоту импульсов на два.

На рисунке 8.3 показана схема, на которой триггер типа *D* включен в счетный режим, т. е. меняет свое состояние на каждый передний фронт синхроимпульса и тоже делит входную частоту на два.

Рис. 8.3 – Триггер типа *D* в счетном режиме

Соединим последовательно три триггера *J-K*, так, чтобы прямой выход предыдущего триггера соединялся со входом *C* следующего триггера, и рассмотрим диаграмму работы (рис. 8.4), на которой выведены импульсы генератора *G* и выходы триггеров *Q*1, *Q*2, *Q*3.

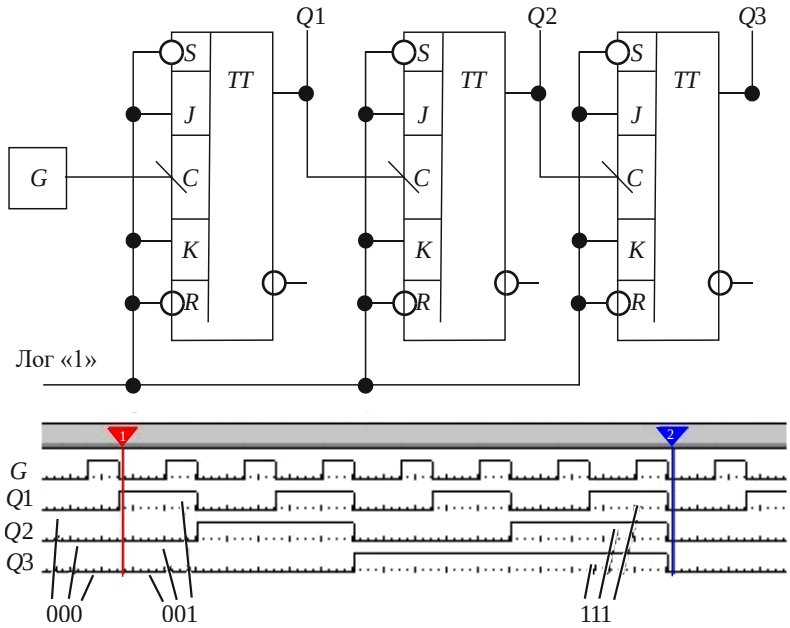


Рис. 8.4 – Трехразрядный накапливающий двоичный счетчик

Из диаграммы видно, что каждый последующий триггер делит частоту с выхода предыдущего триггера на два. Если рассмотреть задний фронт синхроимпульса и провести вертикально линию вниз, то можно прочитать двоичный код из состояния каждого триггера и этот код представляет собой двоичное число, соответствующее количеству импульсов, пришедших на вход первого триггера. Таким образом, эта схема представляет собой накапливающий двоичный счетчик, считающий количество импульсов генератора, пришедших на вход *C*. Состояния счетчика изменяется от 000 в начале цикла (левее красного маркера) до 111 (левее синего маркера). Между двумя вертикальными маркерами разместился один цикл работы счетчика. Цикл повторяется многократно.

Прибавление единицы к очередному состоянию счетчика происходит *переносом*, формируемым задним фронтом с выхода предыдущего триггера *последовательно*, поэтому такая схема называется *счетчиком с последовательным переносом*. Диаграмма работы этого счетчика (рис. 7.4) соответствует таблице 7.1.

На рисунке 8.5 представлена схема трехразрядного двоичного счетчика на *D*-триггерах. Диаграмма работы показывает, что при таком включении триггеров получается вычитающий счетчик.

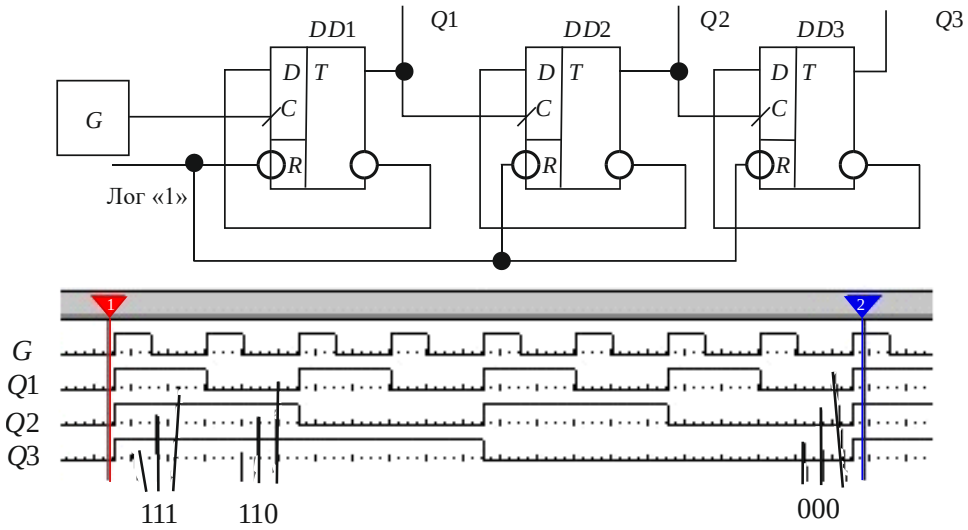


Рис. 8.5 – Трехразрядный вычитающий двоичный счетчик

Главным недостатком счетчиков с последовательным переносом является быстродействие его работы и последовательный набег задержки выходных сигналов. Дело в том, что каждый реальный элемент имеет задержку распространения сигналов t_3, обусловленную свойствами внутренних компонентов. Поэтому выходные сигналы каждого триггера задержаны по отношению к фронту синхросигнала (рис. 8.6).

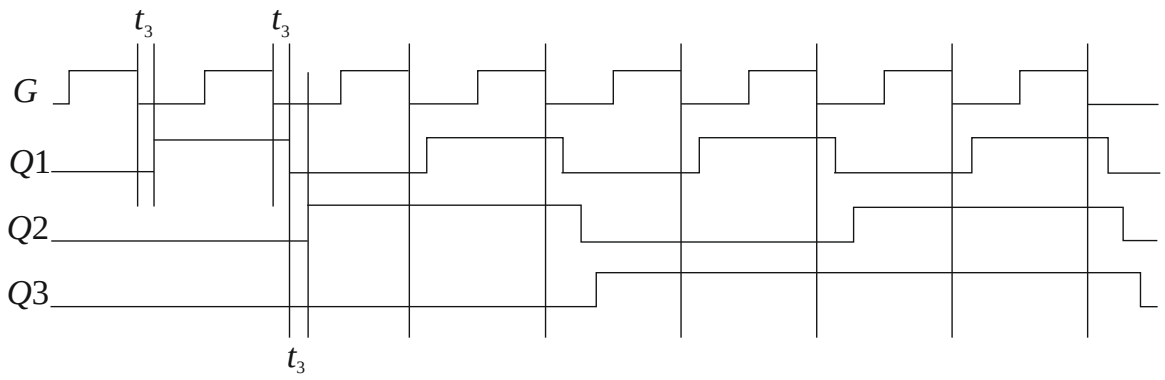


Рис. 8.6 – Эффект гонок в счетчике

После первого счетного импульса выход $Q1$ триггера $DD1$ переходит в противоположное состояние с задержкой $t_з$. Каждый следующий триггер имеет такую же величину задержки. После восьмого импульса состояние счетчика не меняется из $Q3$ $Q2$ $Q1$ = 111 в 000, а проходит ряд промежуточных состояний, таких как 111 – 110 – 100 – 000. Такой эффект называется гонками состояний, и он может нарушить работу всей схемы. Поэтому использование счетчиков с последовательным переносом ограничено.

Следует отметить, что в схемах, использующих триггеры, важно перед началом работы (по включению питания схемы) привести их в начальное состояние. В этом случае используются синхронные триггеры, имеющие асинхронные входы S или R, или один из них. Тогда по сигналу от источника питания сначала происходит установка всех регистров, счетчиков и отдельных триггеров в нулевое или единичное состояние и только потом схема начинает работу. На рисунке 8.7 приведен пример схемы счетчика с возможностью предварительной установки в нулевое состояние.

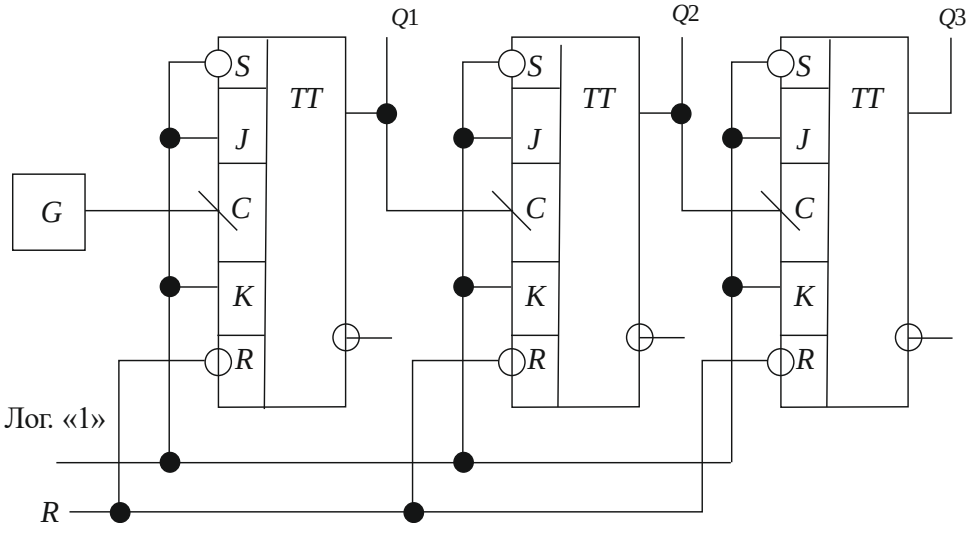


Рис. 8.7 – Двоичный счетчик с возможностью установки в нулевое состояние

8.3 Счетчик с параллельным переносом

Чтобы избавиться от эффекта гонок, в счетчиках применяют схемы с параллельным переносом. В этих схемах синхровходы всех триггеров соединены вместе и синхросигнал подается на все синхровходы одновременно, т. е. триггеры должны изменить состояние в один и тот же момент. Перед приходом активного фронта синхроимпульса на входах триггеров должны быть готовы данные для их перехода в нужное состояние (рис. 8.8).

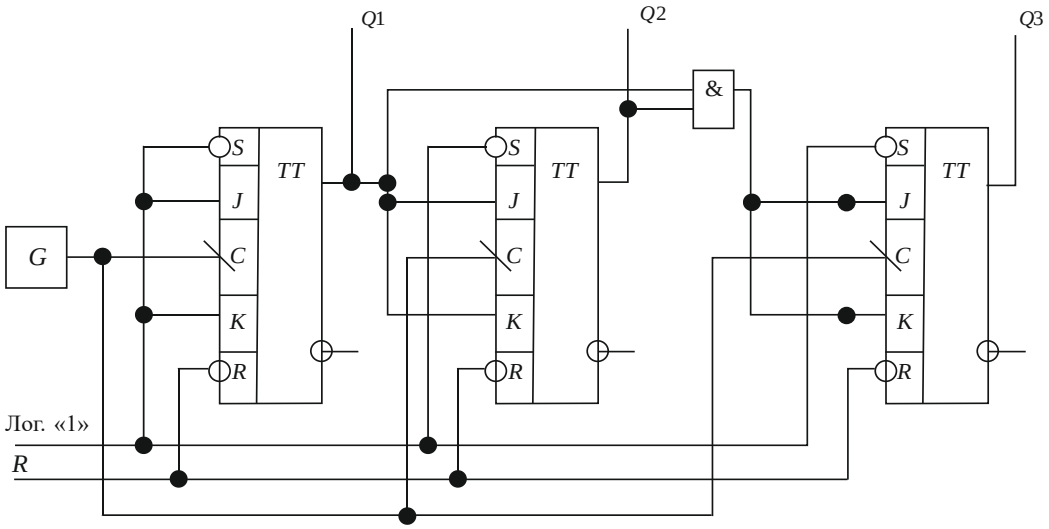


Рис. 8.8 – Двоичный счетчик с параллельным переносом

Для этого используются комбинационные схемы, формирующие необходимые данные. Самое главное в схемах таких счетчиков – это одновременное переключение всех триггеров, у которых на *J* и *K* присутствуют сигналы, соответствующие необходимому переходу триггера. Это исключает гонки. Проанализировать работу такой схемы несложно, что и предлагается сделать самостоятельно.

Существует большая номенклатура счетчиков, выпускаемых промышленностью в виде интегральных микросхем. Для их использования достаточно открыть справочник и прочитать описание счетчика, его обозначение и назначение каждого входа и выхода. Обозначение самого простого четырехразрядного счетчика на схемах показано на рисунке 8.9.

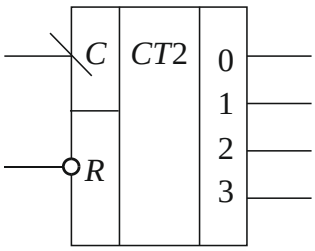


Рис. 8.9 – Четырехразрядный двоичный счетчик

В центральном поле счетчика ставится обозначение *CT* (от *counter* – счетчик), цифра 2 обозначает «двоичный». У счетчика есть вход для сброса в ноль низким уровнем сигнала и символ сброса *R*. Вход синхронизации *C* показывает, на какой фронт реагирует счетчик. Выходы справа показывают номера выходов и их веса (0 – младший разряд, 3 – старший).

8.4 Счетчики с предустановкой

Если в схему счетчика ввести схемы, позволяющие каждый триггер счетчика установить предварительно в 0 или 1, то получится счетчик, в который можно перед счетом записать какое-либо число. Такие счетчики называют счетчиками с предустановкой. В таких счетчиках есть входы двух синхросигналов. Один для синхронизации записи, другой – для счета. Пример обозначения такого счетчика приведен на рисунке 8.10.

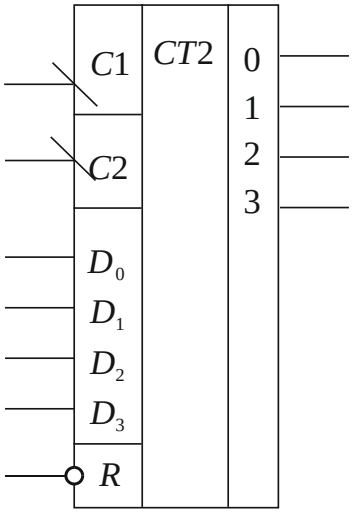


Рис. 8.10 – Обозначение счетчика с предварительной установкой

Счетчик может быть сброшен в нулевое состояние по входу *R*. По синхросигналу *C*1 в него можно записать число со входов D_0, D_1, D_2, D_3. Импульсы, поступающие на вход *C*1, являются счетными.

8.5 Реверсивные счетчики

Счетчики, которые могу выполнять счет с накоплением или убыванием, называются реверсивными (рис. 8.11).

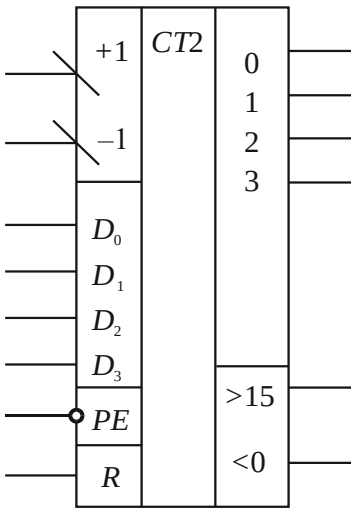


Рис. 8.11 – Обозначение реверсивного счетчика

Входы +1 и –1 предназначены для счета в одну или в другую сторону. Одновременная подача *C*1 и *C*2 запрещена. Входы поля *D* предназначены для подачи кода предварительной записи, которая осуществляется низким уровнем на входе *PE*. Выход > 15 формируют выходной импульс при переходе из 1111 в 0000 при счете с накоплением. Выход < 0 формируют импульс из 0000 в 1111 при счете с убыванием. Они используются как расширители, т. е. позволяют собирать счетчики с большим количеством разрядов (до 8, 12, 16) путем последовательного соединения счетчиков.

На рисунке 8.12 представлен восьмиразрядный счетчик, полученный из двух четырехразрядных.

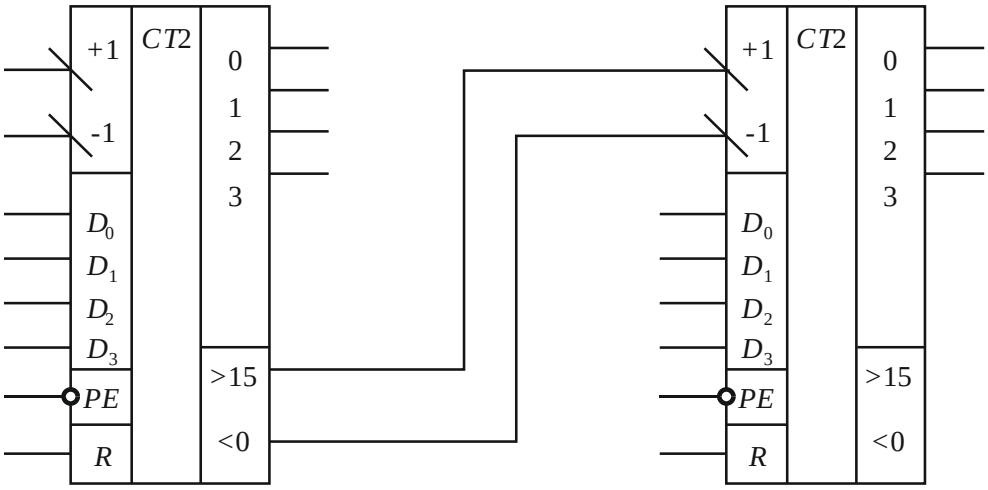


Рис. 8.12 – Восьмиразрядный реверсивный счетчик

8.6 Счетчики – делители частоты

Триггеры счетчика работают в счетном режиме и каждый из них делит входную частоту на два. Поэтому их часто применяют в тех случаях, когда требуется уменьшить частоту импульсов генератора.

Рассмотрим диаграмму работы четырехразрядного накапливающего счетчика (рис. 8.13).

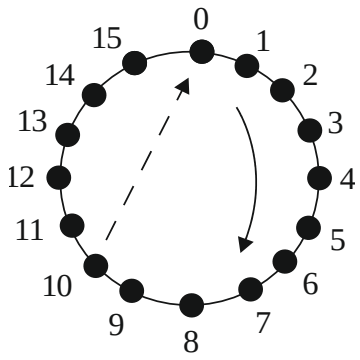


Рис. 8.13 – Диаграмма работы счетчика и счетчика-делителя

Пусть начальное состояние будет нулевым (0000). По мере поступления счетных импульсов счетчик изменяет состояние от 0 (0000) до 15 (1111). Итого счетчик может находиться в одном из 16 состояний. При этом с разрядов счетчика выходят импульсы с частотой в два, четыре, восемь и шестнадцать раз ниже, чем входная частота импульсов генератора (рис. 8.14).

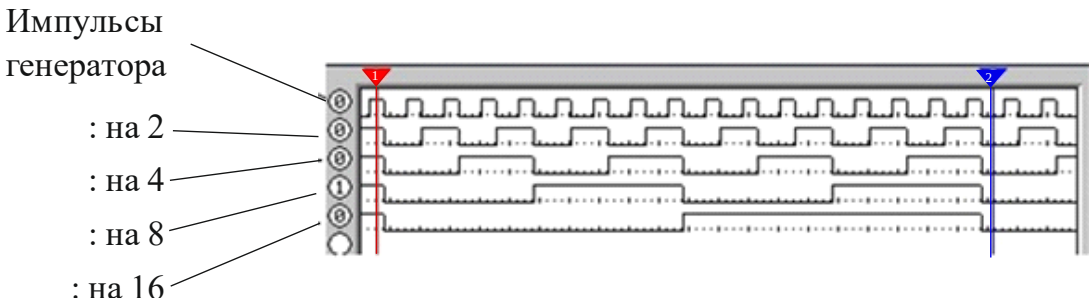


Рис. 8.14 – Диаграмма работы 4-разрядного счетчика

Красный и синий маркеры ограничивают один цикл работы счетчика. Верхний сигнал – это импульсы генератора. В цикле их 16. На выходах 0, 1, 2, 3 счетчика образуются 8, 4, 2, 1 импульса.

Вернемся к рисунку 8.13. Если при достижении состояния 10_{10} (1010_2) счетчик принудительно перевести в начальное состояние (0000), то он пройдет всего десять состояний (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Цикл будет состоять из 10 импульсов генератора. Значит, получается делитель частоты на 10.

В общем, если у счетчика N разрядов, то количество состояний счетчика равно 2^N. Коэффициент деления на K ($K < 2^N$) получится в том случае, когда счетчик принудительно переводится из K-го состояния в нулевое. На рисунке 8.15 приведена схема делителя на 10 и диаграммы его работы. Так как вход R у счетчика инверсный, то для сброса в ноль нужно создать сигнал низкого уровня. Для этого в схеме использован элемент И-НЕ, который выделяет с выходов счетчика комбинацию 10 (1010). Как только на выходах 1 и 4 появятся единицы, на выходе И-НЕ появится нулевой сигнал, сбрасывающий счетчик в нулевое состояние. На осциллограмме цикл работы заключен между красным и синим маркерами. Видно, что на десять импульсов генератора появляется один-единственный импульс на выходе 3, т. е. происходит деление частоты на 10.

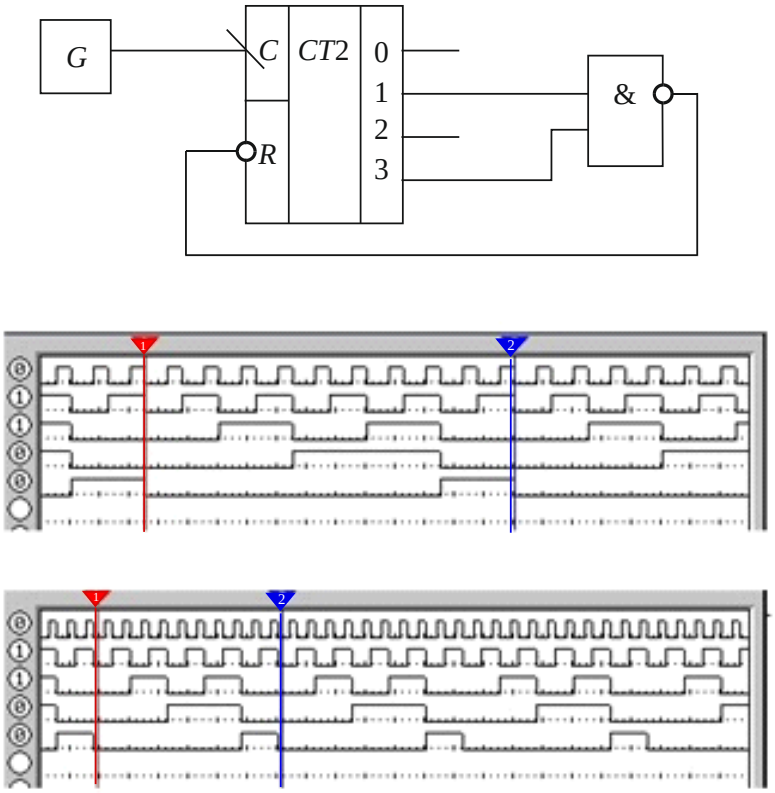


Рис. 8.15 – Диаграмма работы счетчика – делителя на 10

На второй диаграмме для наглядности показаны несколько циклов работы счетчика-делителя. Можно получать схемы счетчиков-делителей с разными коэффициентами деления. На рисунках 8.16 и 8.17 приведены диаграммы работы счетчиков-делителей с различными коэффициентами деления.

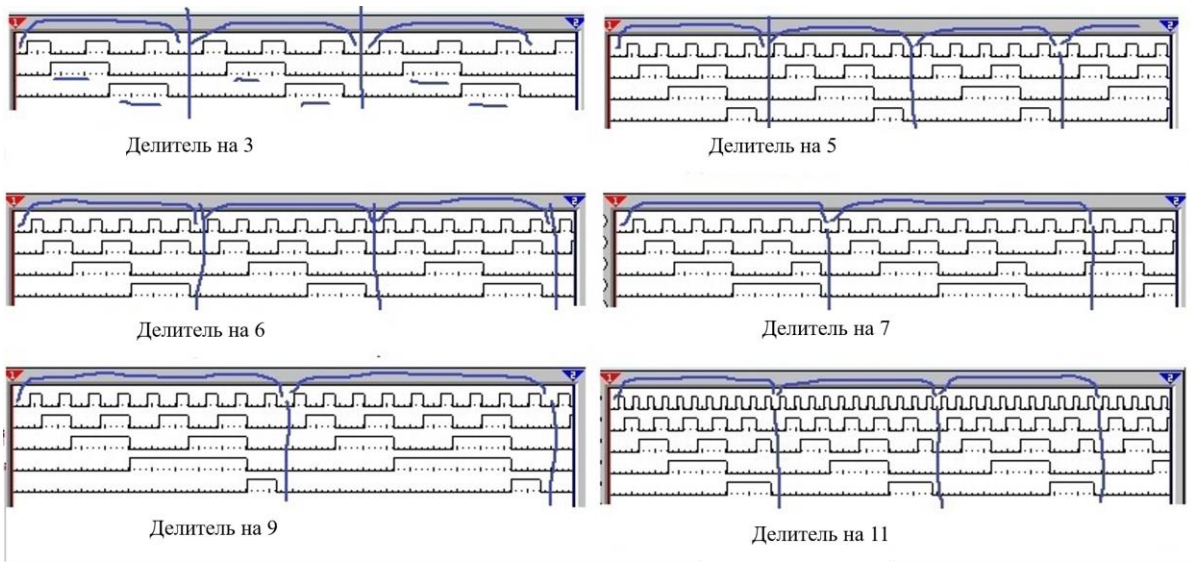


Рис. 8.16 – Диаграмма работы счетчиков – делителей на 3, 5, 6, 7, 9, 11

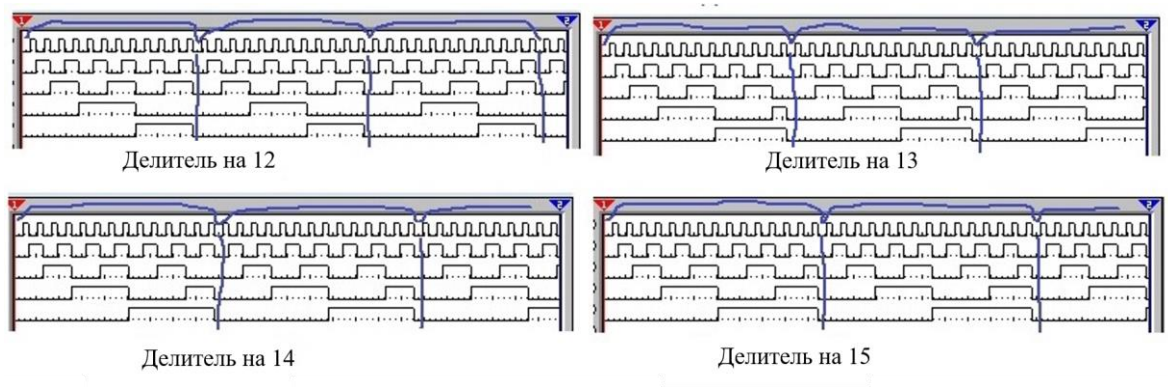


Рис. 8.17 – Диаграмма работы счетчиков – делителей на 12, 13, 14, 15

Микроэлектронная промышленность выпускает широкую номенклатуру счетчиков различного назначения. Обозначения некоторых из них представлены на рисунке 8.18.

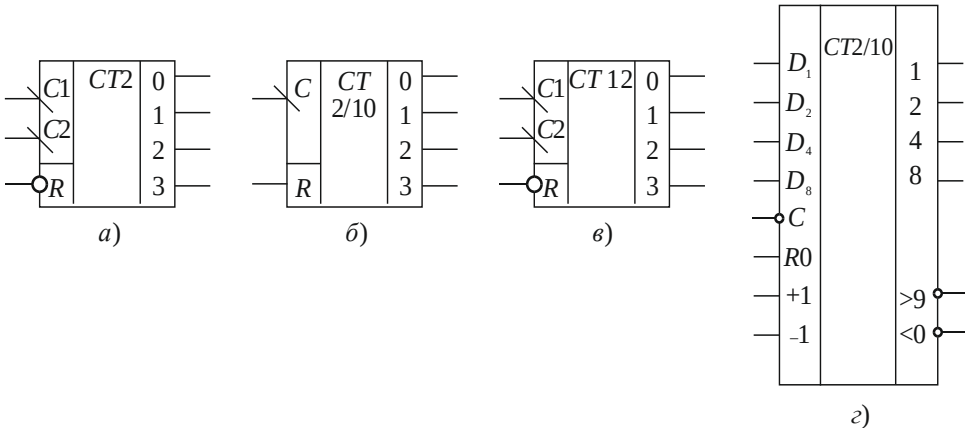


Рис. 8.18 – Счетчики различного назначения:
составной двоичный (*а*); двоично-десятичный (*б*);
составной делитель на 12 (*в*); реверсивный двоично-десятичный (*г*)

Двоичный составной счетчик (рис. 8.18, *а*) состоит из двух частей: делителя на 2 со счетным входом *С*1 и делителя на 8 со счетным входом *С*2. Части счетчика можно использовать по отдельности, как триггер – делитель на 2 и трехразрядный двоичный счетчик. Используя соединение его частей (рис. 8.18, *д*), можно из них построить обычный четырехразрядный накапливающий счетчик.

На рисунке 8.18, *б* представлен счетчик, состоящий из делителя на 2 и делителя на 5. Соединяя обе части, можно получить накапливающий счетчик, имеющий 10 состояний и ведущий счет в двоично-десятичном коде (от 0 до 9).

На рисунке 8.18, *в* представлен счетчик, состоящий из делителя на 2 и делителя на 6. Соединяя обе части, можно получить накапливающий счетчик, имеющий 12 состояний и ведущий счет в двоичном коде (от 0 до 11). Счетчики (рис. 8.18, *б* и рис. 8.18, *в*) создавались специально для создания счетчиков секунд, минут и часов при производстве электронных часов.

На рисунке 8.18, *г* представлен сложный двоично-десятичный реверсивный счетчик с возможностью предустановки. Синхроимпульс C используется для записи двоичного кода с входов D_1, D_2, D_4, D_8 в триггеры счетчика. Входы +1 и –1 используются для счета с накоплением или вычитанием соответственно. Расширяющие выходы (≤ 0, ≥ 9) позволяют передать сигналы переполнения для подключения такого же счетчика при создании счетчика большей разрядности – 8, 12, 16 и т. д.

8.7 Проверка работы КС с помощью счетчика

Двоичные счетчики удобно использовать как формирователи наборов булевых переменных и с их помощью проверять работу КС, реализующих БФ.

Пусть заданы БФ $F = \sum(0,\ 1,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15)$, $P = \prod(0,\ 4,\ 5,\ 6,\ 7)$. Функция F от четырех переменных, P – от трех. После минимизации получаем:

$$F = A + \overline{B}\,\overline{C} + BCD;$$

$$P = \overline{B} \cdot (C + D).$$

Схемы, реализующие БФ, подключим к счетчику и снимем осциллограмму работы БФ. На рисунке 8.19 представлена схема и диаграмма работы. Красный и синий маркеры располагаются на наборах 0000 и 1111 счетчика. Для функции F на диаграмме виден только один цикл, а для функции P – два цик-

ла, т. к. переменные B, C, D за один цикл работы всего счетчика проходят два цикла. Из анализа диаграммы видно, что КС работает в соответствии с заданием.

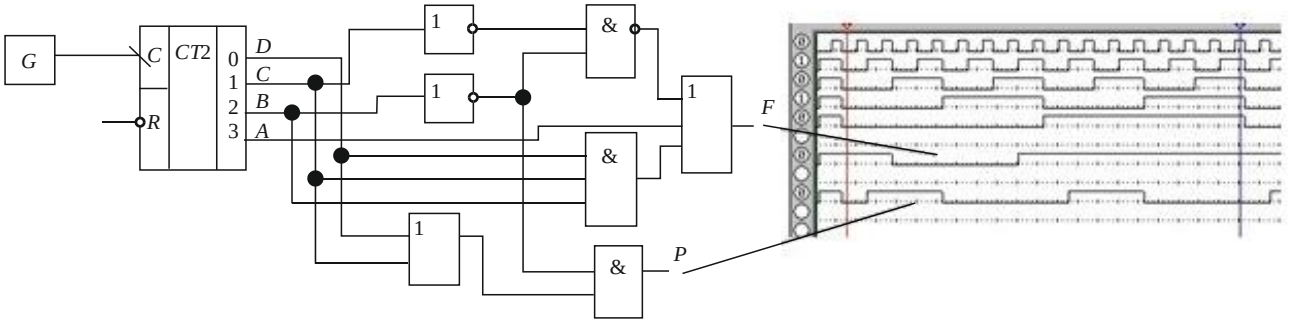


Рис. 8.19 – Диаграмма работы счетчика и КС

Контрольные вопросы по главе 8

1. Сколько состояний у счетчика-делителя с коэффициентом деления *K*?
2. Какую операцию кроме счета может выполнять счетчик?
3. Может ли шестиразрядный двоичный счетчик выполнять функцию счетчика-делителя на 50?
4. Из-за чего возникает эффект гонок в счетчиках?
5. К какой группе триггеров относятся триггеры типа *D* и *J-K*?

9 Автоматы

9.1 Математическая модель цифрового устройства

Вспомним еще раз математическую модель цифрового устройства.

$$S = (A,\ Z,\ W,\ \delta,\ \lambda,\ a_1).$$

Она вводит описание абстрактного автомата S как шестикомпонентного вектора, который включает:

- $A = \{a_1, \ldots a_m, \ldots a_M\}$ – множество внутренних состояний;
- $Z = \{z_1, \ldots z_f, \ldots z_F\}$ – множество входных сигналов;
- $W = \{w_1, \ldots w_g, \ldots w_G\}$ – множество выходных сигналов;
- $\delta: A \times Z \to A$ – функцию переходов, реализующую отображение произведения множеств $A \times Z$ на A;
- $\lambda: A \times Z \to W$ – функция выходов, реализующую отображение произведения множеств $A \times Z$ на W;
- $a_1 \subseteq A$ – начальное состояние.

Рассмотрим работу автомата (A) при следующих условиях:

- $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$ – семь внутренних состояний;
- $Z = \{z_1, z_2, z_3, z_4\}$ – четыре входных сигнала;
- $W = \{w_1, w_2, w_3, w_4\}$ – четыре выходных сигнала.

На рисунке 9.1 представлена диаграмма фрагмента работы автомата. Автоматы работают в сетке времени, которая представляет собой дискретные отсчеты t_0, t_1, t_2, …, называемые машинными тактами. Фрагмент работы автомата поясним диаграммой на рисунке 9.1.

1. Пусть в начальном такте t_0 автомат находился в состоянии a_1 из множества A и на его вход пришел входной сигнал z_1 из множества Z.
2. В соответствии с функцией перехода δ автомат перейдет в следующем такте t_1 в состояние a_2 и в соответствии с функцией λ сформирует выходной сигнал w_4.

3. В такте t_1 автомат находится в состоянии a_2 и на его вход приходит входной сигнал z_4, который заставит в соответствии с функцией перехода δ перейти автомат в такте t_2 в новое состояние a_6 и в соответствии с функцией выхода λ сформировать выходной сигнал w_4.
4. Теперь автомат находится в такте t_2 и ожидает прихода очередного входного сигнала и т. д.

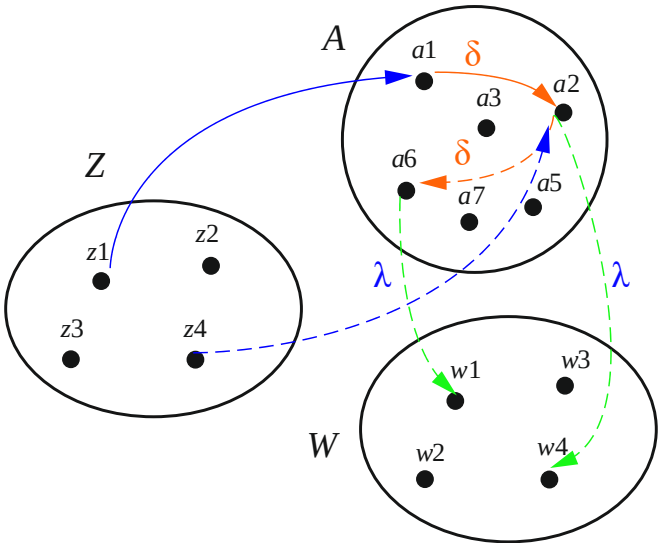


Рис. 9.1 – Диаграмма работы автомата

Таким образом, автомат должен помнить текущее внутреннее состояние a_m и по какому из множества входных сигналов z_f перейти в новое, строго определенное состояние a_k и при этом выдать строго определенный выходной сигнал w_g. Такие автоматы называются детерминированными, т. е. строго определенными. Кроме того, в начале работы автомат должен обязательно находиться в состоянии a_1 (например, любой двоичный счетчик является автоматом).

В автоматах в качестве элементов памяти, хранящих текущее состояние, обязательно должны использоваться элементарные автоматы Мура, в качестве которых выступают триггеры (или другие элементы, имеющие свойство запоминания).

На практике используются два вида автоматов, получивших названия по именам их исследователей.

Автомат Мили задается следующими уравнениями:

$$a(t+1)=\delta\big(a(t),\ z(t)\big),$$
$$w(t)=\lambda\big(a(t),\ z(t)\big),$$
$$t=0,\ 1,\ 2,\ 3,\ \ldots.$$

В момент времени t автомат находится в состоянии $a(t)$ и на его вход приходит входной сигнал $z(t)$. В соответствии с законом выходов λ автомат выдаст выходной сигнал *w*, зависящий от состояния и от входного сигнала, и в соответствии с законом переходов δ перейдет в новое состояние в момент времени $t+1$.

Автомат Мура задается следующими уравнениями:

$$a(t+1)=\delta\big(a(t),\ z(t)\big),$$
$$w(t)=\lambda\big(a(t)\big),$$
$$t=0,\ 1,\ 2,\ 3,\ \ldots.$$

В момент времени t автомат находится в состоянии $a(t)$ и на его вход приходит входной сигнал $z(t)$. В соответствии с законом переходов δ перейдет в новое состояние в момент времени $t+1$. Выходной сигнал этого автомата в соответствии с λ не зависит от входного сигнала, а зависит только от текущего состояния автомата. В дальнейшем будем рассматривать автоматы Мура.

9.2 Способы задания автоматов

9.2.1 Табличный способ задания автомата

Существует несколько способов задания абстрактных автоматов, из которых мы рассмотрим табличный и графический способы. Очевидно, что надо каким-то образом перечислить все возможные состояния автомата и все входные сигналы, которые могут поступать в автомат. Также следует задать функции, по которым автомат переходит из одного состояния в другое, и функции формирования выходных сигналов. В табличном способе это достигается построением *таблицы переходов* и *таблицей выходов*. Так как в автомате Мура выходной сигнал зависит только от состояния автомата, то его можно задавать одной таблицей переходов-выходов, которую часто называют *отмеченной таблицей переходов*. На рисунке 9.2, *а* представлена отмеченная таблица переходов полностью определенного автомата.

	w_1	w_2	w_3	w_1
	a_1	a_2	a_3	a_4
z_1	a_2	a_3	a_4	a_1
z_2	a_4	a_1	a_4	a_2
z_3	a_3	a_2	a_1	a_3

а)

	w_1	w_2	w_3	–
	a_1	a_2	a_3	a_4
z_1	a_2	–	a_3	a_1
z_2	–	a_4	a_4	a_3

б)

	–	–	–	–	–	w_1
	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_2	a_3	a_4	a_5	a_6	a_1

в)

Рис. 9.2 – Отмеченные таблицы переходов автомата Мура:
полностью определенный автомат (*а*);
неполностью определенный автомат (*б*); счетчик – делитель на 6 (*в*)

Автомат (рис. 9.2, *а*) работает следующим образом:

- вначале A находится в состоянии a_1, формируя на выходе сигнал w_1, и на его вход может прийти любой из входных сигналов z_1, z_2 или z_3;
- если пришел сигнал z_1, то A перейдет в состояние a_2 и будет выдавать сигнал w_2 все время, пока находится в этом состоянии;
- если же в начальном состоянии a_1 на вход приходит z_2, то автомат перейдет в состояние a_4 и начнет формировать выходной сигнал w_1;
- и так далее.

Если состояние перехода или выходной сигнал в автомате не определены, то автомат называется неполностью определенным. В неполностью определенном автомате (рис. 9.2, *б*) может быть неопределенным переход в другое состояние или не определен выходной сигнал. В остальном работа A не отличается от работы полностью определенного автомата. Этот автомат из состояния a_1 может перейти только в состояние a_2, а из него только в состояние a_4. Причем в состоянии a_4 выходной сигнал автомат не определен.

На рисунке 9.2, *в* для примера представлена таблица переходов для счетчика – делителя на 6. Он имеет 6 состояний, в которые последовательно переходит по единственному входному сигналу z_1 (по синхросигналу от генератора). Выходной сигнал формируется только тогда, когда этот автомат будет находиться в состоянии a_6, из которого опять переходит в состояние a_1.

9.2.2 Графический способ задания автомата

При этом способе A задается графом. Граф автомата – это ориентированный граф. В его *вершинах* записываются *состояния* и соответствующие им *выходные сигналы*. Направленные дуги (стрелки) показывают направление перехода в новое состояние, причем *входной сигнал*, инициирующий этот переход,

записывается у *начала дуги*. Очевидно, что графический и табличный способы задания должны быть взаимно однозначными (по таблице можно нарисовать граф и наоборот). На рисунке 9.3 показаны графы автоматов, которые были перед этим заданы таблицами переходов-выходов (рис. 9.3, *а*, *б*, *в* соответственно).

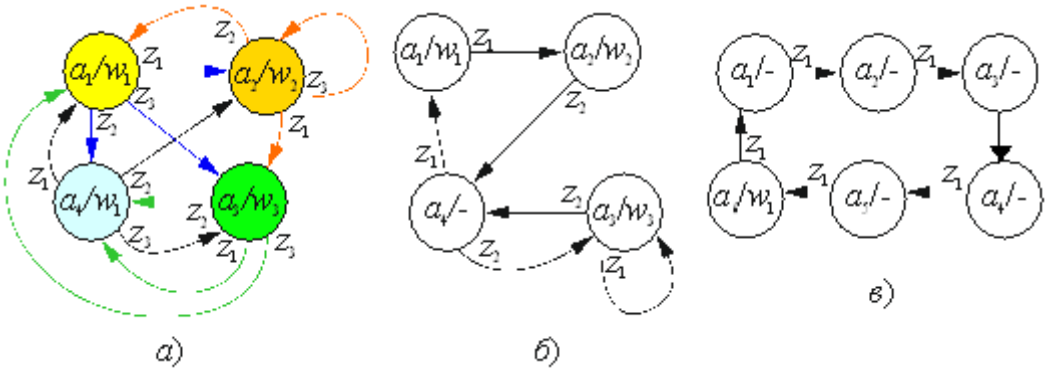


Рис. 9.3 – Графы автомата Мура

Таблицы переходов-выходов и граф автомата полностью задают элементы множеств *A*, *Z*, *W* и функции переходов δ и выходов λ абстрактного автомата. Задание автомата таблицами или графом называется абстрактным синтезом.

9.3 Структурный автомат

После этапа абстрактного синтеза должен следовать следующий этап – структурного синтеза. Результат этого этапа – структурный автомат, т. е. схема, реализующая абстрактный автомат с использованием цифровых элементов и узлов. Структурный автомат должен полностью представить структуру входных, выходных сигналов и внутреннее устройство автомата. Структурный автомат представляется в виде набора элементов памяти, хранящих состояния автомата, и комбинационной схемы, реализующей функции δ и λ (рис. 9.4).

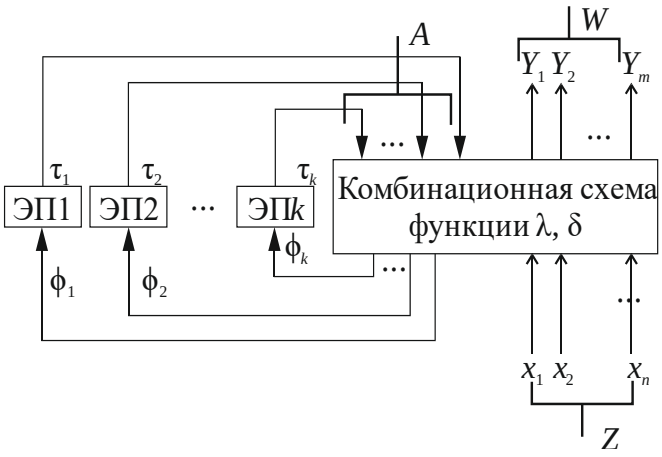


Рис. 9.4 – Структурный автомат

Элементами памяти (ЭП) структурного автомата являются элементарные автоматы Мура, в качестве которых мы будем рассматривать триггеры. Каждое состояние абстрактного автомата из множества A представляется двоичным вектором $\tau_1, \tau_2, \ldots, \tau_k$, значения которых соответствуют состояниям ЭП (триггеров). Любой элемент этого вектора τ_i принимает одно из двух значений – 0 или 1, т. е. является логическим сигналом. Сигналы с выходов ЭП поступают на входы комбинационной схемы (КС). На другие входы КС поступают двоичные входные сигналы, формирующие вектор $x_1, x_2, \ldots, x_n$ (любой x_k может принимать значение либо 0, либо 1), соответствующие сигналам из множества Z. КС формирует выходные сигналы, соответствующие сигналам из множества W, представляя их двоичными векторами $y_1, y_2, \ldots, y_m$. Формирование выходных сигналов реализуется в КС булевыми функциями, определяемыми функцией выходов λ. КС, получив из ЭП двоичный код текущего состояния (из множества A) и двоичный код входного сигнала (из множества Z), формирует булевы функции возбуждения элементов памяти в виде двоичного вектора $\varphi_1, \varphi_2, \ldots, \varphi_k$, который переводит ЭП в новое состояние в соответствии с функцией переходов δ, задаваемой таблицей или графом. Таким образом КС реализует функцию переходов $\varphi = \delta(\tau, x)$ и функцию выходов $y = \lambda(\tau)$.

При синтезе структурного автомата в качестве ЭП используются все виды триггеров *R-S, D, J-K*. Следует, конечно, отметить, что при использовании асинхронных *R-S*-триггеров за счет разной длины логических схем, реализующих функции возбуждения триггеров, возможно появление эффекта гонок. Это может привести к нарушению алгоритма работы автомата, заданном таблицами или графами. Этого можно избежать, используя различные схемотехнические ухищрения, что приводит к усложнению схемы. При использовании синхронизируе-

мых триггеров типа *D* и *J-K* эффект гонок существенно снижается за счет того, что изменение состояний триггеров происходит в строго определенные моменты, определяемые фронтами синхроимпульсов тактового генератора. При этом частота тактового генератора выбирается такой, чтобы за время одного периода следования тактовых импульсов все цепочки логических элементов, формирующих сигналы возбуждения ЭП, гарантированно успели сформировать выходной сигнал.

На рисунке 9.5 показаны две логические схемы формирования функций возбуждения ЭП. Одна из цепей имеет всего один логический элемент, а значит, время распространения сигнала от входа к выходу будет составлять одну единицу. Вторая схема содержит три логических элемента и время распространения будет равно трем. Это значит, что один ЭП при использовании асинхронных триггеров, реагируя на функцию возбуждения, может сменить свое состояние быстрее другого, что может исказить функцию возбуждения другого ЭП. Это приводит к эффекту гонок и нарушает алгоритм работы автомата. Для устранения эффекта гонок используются разные способы. Одним из них является способ, при котором в качестве ЭП используются синхронные триггеры типа *D* или *J-K*. При этом переход автомата из одного состояния в другое осуществляется только по фронту импульса синхронизирующего сигнала тактового генератора. Период *T* следования импульсов генератора выбирается таким, чтобы все, даже самые длинные логические цепи успели сформировать сигналы возбуждения ЭП до прихода фронта синхроимпульса. Это гарантирует правильность переходов автомата в соответствии с таблицей переходов-выходов.

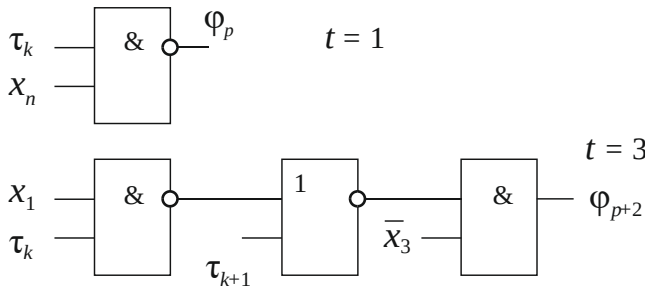


Рис. 9.5 – Схемы формирования сигналов возбуждения

9.4 Проектирование структурного автомата

Проектирование начинается с определения элементов множеств *A*, *Z*, *W* и составления таблицы переходов-выходов или графа автомата. Рассмотрим несколько примеров проектирования структурного автомата.

Задача 9.1. Спроектировать структурный автомат Мура, заданный совмещенной таблицей переходов-выходов (табл. 9.1).

Таблица 9.1 – Таблица переходов-выходов автомата

	w_1	w_2	w_3	w_1
	a_1	a_2	a_3	a_4
z_1	a_2	a_3	a_4	a_1
z_2	a_4	a_1	a_4	a_2
z_3	a_3	a_2	a_1	a_3

1. Кодирование состояний.

Каждому состоянию автомата присваиваются обозначения переменных и двоичный код. Состояний у заданного автомат четыре, т. е. каждому состоянию присваивается двухразрядный двоичный код, а переменным, которые отождествляются с выходами триггеров, присваиваются названия *a, b* (табл. 9.2).

Таблица 9.2 – Кодирование состояний

	Булевы переменные выходов триггеров
	ab
a_1	00
a_2	01
a_3	10
a_4	11

2. Кодирование входных сигналов.

Входных сигналов у автомата три. Каждому из них можно присвоить двухразрядный двоичный код. А можно присвоить и трехразрядный. Примеры приведены в таблице 9.3.

Таблица 9.3 – Кодирование входных сигналов

	Булевы переменные входных сигналов	
	cd	*cde*
z_1	00	100
z_2	01	010
z_3	10	001

3. Кодирование выходных сигналов.

У автомата три выходных сигнала, т. е. каждый из них можно закодировать как минимум двумя битами. Но может быть удобней и трехразрядное кодирование. Пример в таблице 9.4.

Таблица 9.4 – Кодирование выходных сигналов

	Булевы переменные выходных сигналов	
	y_1y_2	$y_1y_2y_3$
w_1	00	100
w_2	01	010
w_3	10	001

4. Выбор элементов памяти.

В качестве ЭП выберем триггеры типа *J-K* с асинхронным входом *R* (для установки ЭП в начальное состояние 00). В таблице 9.5 приведена таблица переходов триггера.

Таблица 9.5 – Переходы триггера *J-K*

$Q(t)$	$Q(T+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Перехода из состояния 0 в состояние 0 можно достичь двумя способами: в первом случае подать комбинацию *JK* = 00, во втором – подать комбинацию *JK* = 01. На вход *J* обязательно подать 0, а на вход *K* можно подать 0 или 1. А раз так, то это можно обозначить неопределенным значением X.

Перехода из состояния 0 в состояние 1 можно достичь двумя способами: в первом случае подать комбинацию *JK* = 10, во втором – подать комбинацию *JK* = 11. То есть на вход *J* обязательно подать 1, а на вход *K* можно подать 0 или 1, поэтому *K* можно обозначить неопределенным значением X.

Перехода из состояния 1 в состояние 0 можно достичь двумя способами: в первом случае подать комбинацию *JK* = 00, во втором – подать комбинацию *JK* = 01. То есть на вход *J* обязательно подать 0, а на вход *K* можно подать 0 или 1, поэтому *J* можно обозначить неопределенным значением X.

Перехода из состояния 1 в состояние 1 также можно достичь двумя способами: в первом случае подать комбинацию *JK* = 00, в втором – подать комбинацию *JK* = 10. То есть на вход *K* обязательно подать 0, а на вход *J* можно подать 1 или 0, поэтому *J* можно обозначить неопределенным значением X.

Сведем эти данные в единую таблицу переходов-выходов (табл. 9.6).

Таблица 9.6 – Таблица переходов-выходов автомата

$a(t)$	Код состояния	$a(t+1)$	Код состояния	Входной сигнал	Код входного сигнала	Вход J_1	Вход K_1	Вход J_2	Вход K_2	Выходной сигнал	Код выходного сигнала
	ab		ab		cde						$y_1y_2y_3$
a_1	00	a_2	01	z_1	001	0	X	1	X	w_1	001
		a_4	11	z_2	010	1	X	1	X		
		a_3	10	z_3	100	1	X	0	X		
a_2	01	a_3	10	z_1	001	1	X	X	1	w_2	010
		a_1	00	z_2	010	0	X	X	1		
		a_2	01	z_3	100	0	X	X	0		
a_3	10	a_4	11	z_1	001	X	0	1	X	w_3	100
		a_4	11	z_2	010	X	0	1	X		
		a_1	00	z_3	100	X	1	0	X		
a_4	11	a_1	00	z_1	001	X	1	X	1	w_1	001
		a_2	01	z_2	010	X	1	X	0		
		a_3	10	z_3	100	X	0	X	1		

В таблице указано состояние автомата, задаваемое переменными ab, в момент автоматного времени t и состояние, в которое автомат перейдет в следующем такте $t+1$ по входному сигналу z, задаваемому переменными cde. Выходной сигнал, зависящий только от состояния автомата, задается двоичным вектором $y_1y_2y_3$. Для перехода автомата из состояния $a(t)$ в $a(t+1)$ на входах J и K необходимо создать соответствующие сигналы. Сигналы J-K формируются в соответствии с функцией переходов $J, K = \delta(a,b,c,d,e)$. Выходные сигналы формируются в соответствии с функцией $Y = \lambda(a,b)$. Для функций выходного сигнала $y_1y_2y_3$ входными сигналами являются состояния автомата, взятые из второго столбца таблицы. Таким образом, таблицу переходов можно рассматривать как таблицу истинности булевых функций J_1, K_1, J_2, K_2, y_1, y_2, y_3. Запишем эти функции числовым способом:

$$J_1(abcde) = \sum\big(2,\ 4,\ 9\ (17,\ 18,\ 20,\ 25,\ 26,\ 28)\big);$$

$$K_1(abcde) = \sum\big(20,\ 25,\ 26\ (1,\ 2,\ 4,\ 9,\ 10,\ 12)\big);$$

$$J_2(abcde) = \sum\big(1,\ 2,\ 17,\ 18\ (9,\ 10,\ 12,\ 25,\ 26,\ 28)\big);$$

$$K_2(abcde) = \sum\big(9,\ 10,\ 25,\ 28\ (1,\ 2,\ 4,\ 17,\ 18,\ 20)\big);$$

$$Y_1(a,b) = \sum(2);\ Y_2 = \sum(1);\ Y_3 = \sum(1,\ 3).$$

После минимизации получим функции:

$$J_1 = \bar{c}e(\bar{b} + \bar{d});$$

$$K_1 = \bar{b}c\bar{d}\bar{e} + b\bar{c}\bar{d}e + b\bar{c}d\bar{e};$$

$$J_2 = \bar{c}\bar{d}e + \bar{c}d\bar{e};$$

$$K_2 = \overline{ac}d\bar{e} + \bar{c}\bar{d}e + ac\bar{d}\bar{e};$$

$$y_1 = a\bar{b};\ y_2 = \bar{a}b;\ y_3 = \bar{a}.$$

5. По полученным функциям можно построить функциональную схему автомата (рис. 9.6).

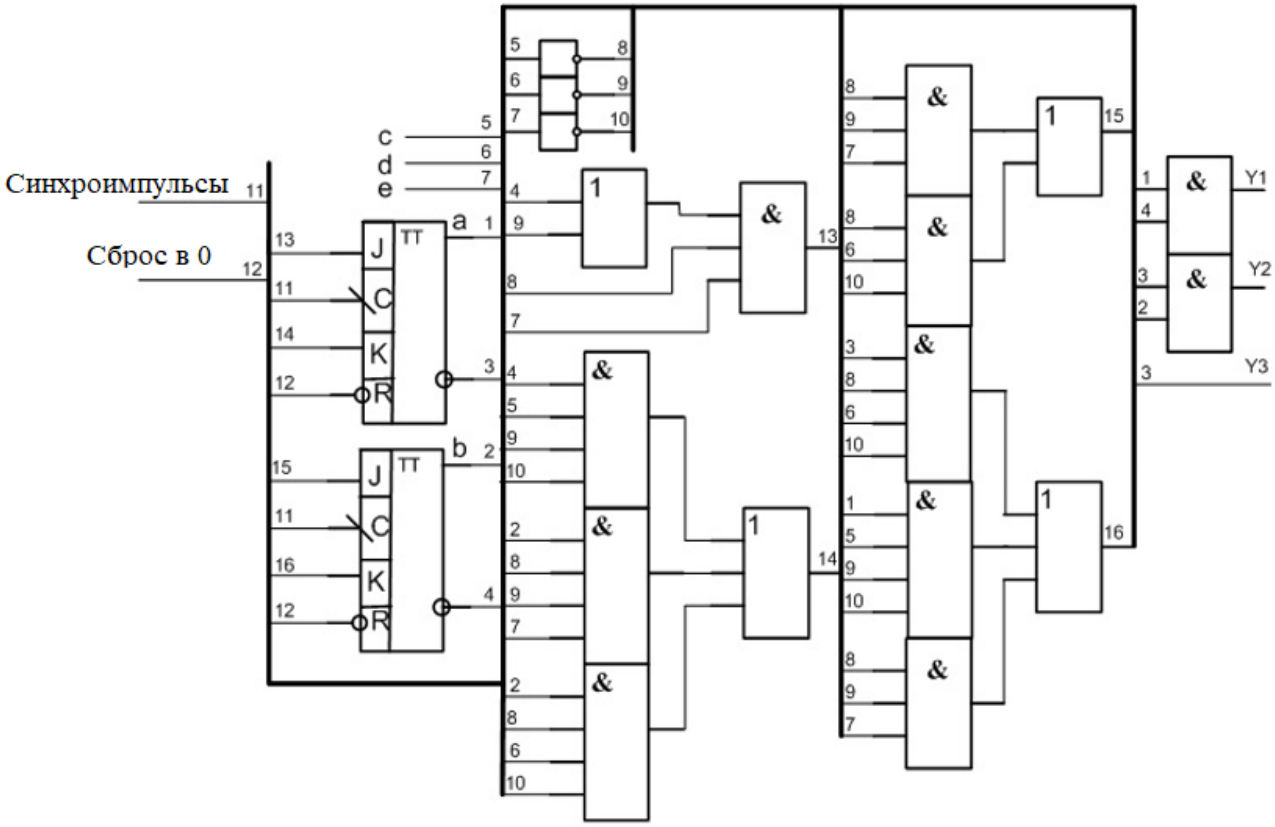


Рис. 9.6 – Функциональная схема автомата

Задача 9.2. Спроектировать генератор четырехразрядных слов. Генератор должен периодически формировать следующую последовательность двоичных кодов:

$$0 – 14 – 7 – 3 – 12 – 9 – 0.$$

В качестве ЭП использовать триггеры *J-K*, комбинационная схема должна быть выполнена на элементах И, ИЛИ, НЕ. Автомат задан графом (рис. 9.7).

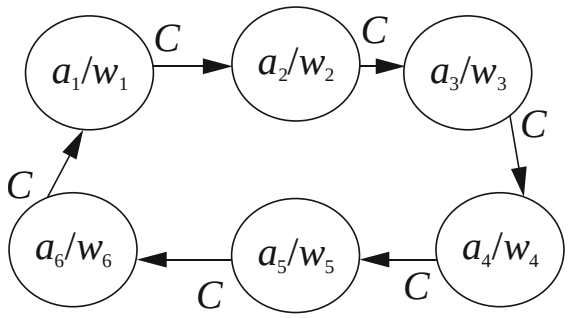


Рис. 9.7 – Граф автомата

1. Кодирование состояний.

Каждому состоянию автомата присваиваются обозначения переменных и двоичный код. Состояний у заданного автомата шесть. Каждому состоянию в соответствии с заданием присваивается четырехразрядный двоичный код, а переменным, которые отождествляются с выходами триггеров, присваиваются названия *a*, *b*, *c*, *d*, например так, как в таблице 9.7.

Таблица 9.7 – Кодирование состояний

	Булевы переменные выходов триггеров
	abcd
a_1	0000
a_2	0100
a_3	1111
a_4	0110
a_5	0010
a_6	0101

2. Кодирование входных сигналов.

У автомата один входной сигнал. Более того, входной сигнал фактически является синхросигналом, тактирующим триггеры. Поэтому считается, что входной сигнал $C = 1$.

3. Кодирование выходных сигналов.

У автомата шесть выходных сигналов. Удобно их закодировать так же, как и состояния триггеров. Это упростит комбинационную схему (табл. 9.8).

Таблица 9.8 – Кодирование выходных сигналов

	Булевы переменные выходных сигналов $y_1y_2y_3y_4$
w_1	0000
w_2	0100
w_3	1111
w_4	0110
w_5	0010
w_6	0101

4. Выбор элементов памяти.

В качестве ЭП заданы триггеры типа *J-K*. Используем триггеры с асинхронным входом *R* для установки ЭП в начальное состояние 0000. В таблице 9.9 приведена таблица переходов триггера. Таблица 9.9 – Переходы триггера *J-K*

***Q*(*t*)**	***Q*(*T*+1)**	***J***	***K***
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Сведем эти данные в единую таблицу переходов-выходов (табл. 9.10).

Таблица 9.10 – Таблица переходов-выходов автомата

***a*(*t*)**	**Код *a*(*t*)**	***a*(*t*+1)**	**Код *a*(*t*+1)**	J_1	K_1	J_2	K_2	J_3	K_3	J_4	K_4	***W***	**Код *W***
	abcd												***abcd***
a_1	0000	a_2	1110	1	X	1	X	1	X	0	X	w_1	0000
a_2	1110	a_3	0111	X	1	X	1	X	0	1	X	w_2	1110
a_3	0111	a_4	0011	0	X	0	X	X	0	X	0	w_3	0111
a_4	0011	a_5	1100	1	X	1	X	X	1	X	1	w_4	0011
a_5	1100	a_6	1001	X	0	X	0	0	X	1	X	w_5	1100
a_6	1001	a_1	0000	X	1	X	1	0	X	X	1	w_6	1001

Функции *J* и *K* всех триггеров зависят от состояний *a*(*t*), при *C* = 1 они фактически зависят от четырех переменных, поэтому нанесем их на карты Карно – Вейча и минимизируем (рис. 9.8 и 9.9). На картах клетки, соответствующие состояниям автомата, отмечены серым цветом. На всех остальных клетках карты ставится знак неопределенности булевой функции.

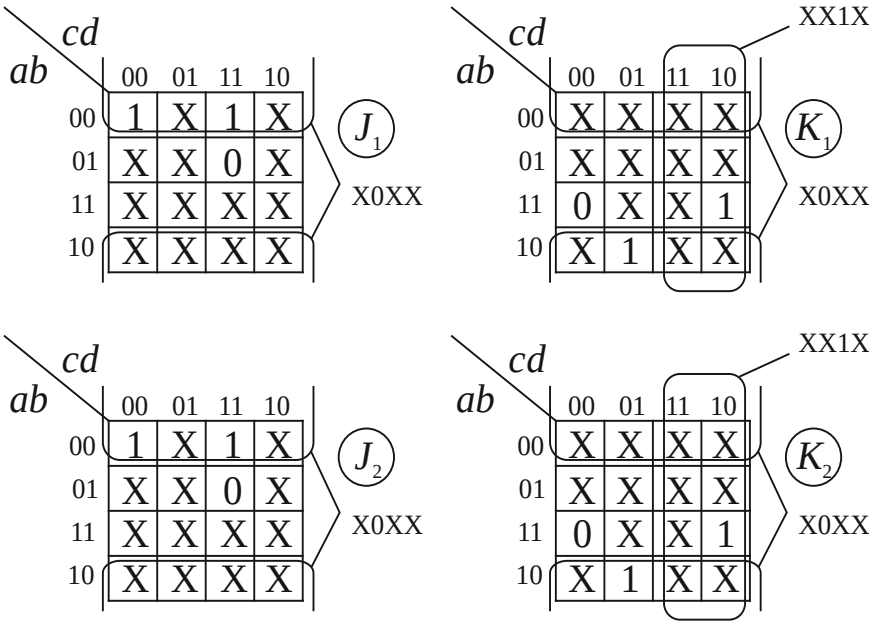


Рис. 9.8 – Карты Карно функций J_1, K_1, J_2, K_2.

$$J_1 = \bar{b},\ K_1 = c + \bar{b},\ \ J_2 = \bar{b},\ K_2 = c + \bar{b}$$

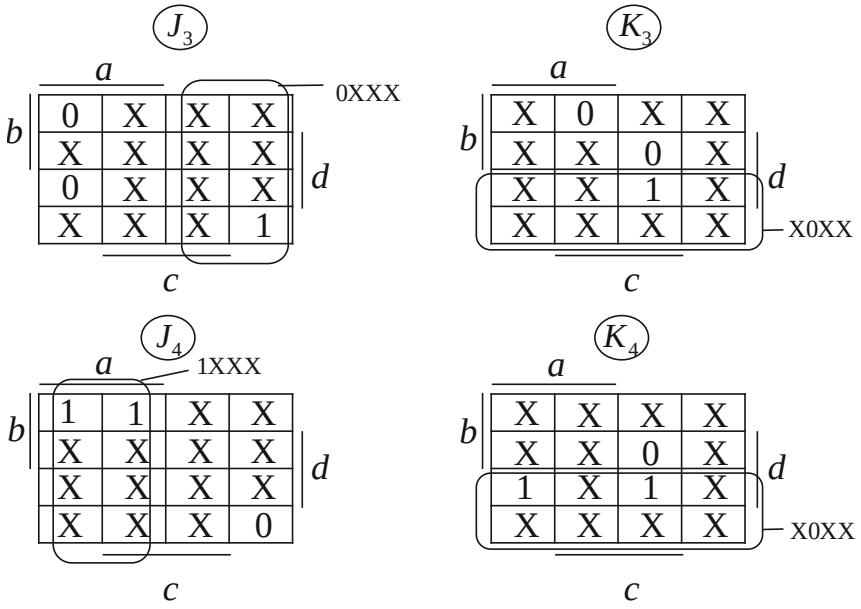


Рис. 9.9 – Карты Вейча функций J_3, K_3, J_4, K_4.

$$J_3 = \bar{a},\ K_3 = \bar{b},\ \ J_4 = a,\ K_4 = \bar{b}$$

Функции выходов формируются выходами триггеров, поэтому вектор $y_1y_2y_3y_4$ совпадает с вектором (кодом) состояния автомата $y_1 = a_1$, $y_2 = b_2$, $y_3 = c_3$, $y_4 = d_4$. На основе всех этих уравнений строим схему (рис. 9.10).

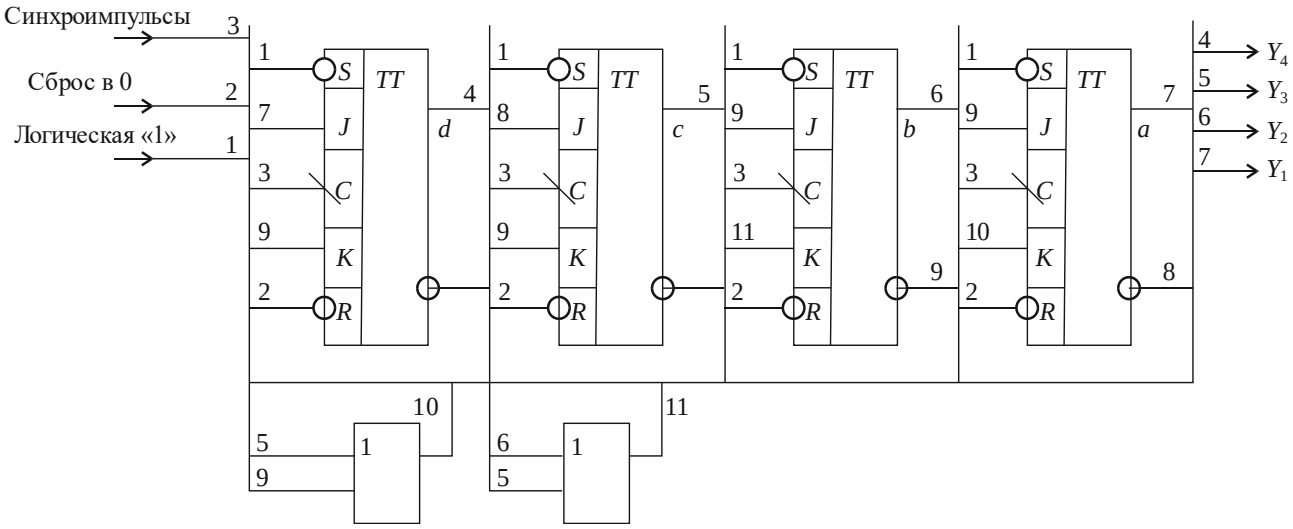


Рис. 9.10 – Функциональная схема автомата

Автомат содержит четыре триггера *J-K* с асинхронными инверсными входами *S* и *R*. Все входные и выходные сигналы соединены в шину. На входы *R* всех триггеров в начале работы автомата должен быть подан сигнал сброса в 0, по которому триггеры установятся в начальной нулевое состояние (0000). На входы *S* всех триггеров подается неактивный единичный сигнал, который никак не влияет на работу триггеров. На входы *C* подается синхронизирующий сигнал, который должен формировать генератор импульсов (на схеме не показан). Выходные сигналы автомата (Y_1, Y_2, Y_3, Y_4) снимаются с прямых выходов триггеров и передаются во внешнее устройство. Функции возбуждения триггеров довольно просты, и только для K_1 и K_2 потребовалось ввести в схему логические элементы ИЛИ.

Задача 9.3. Условия те же, что и в задаче 9.2, но реализовать автомат необходимо на триггерах типа *D*.

Так как условия задачи те же, что и в предыдущем случае, то изменению должны подвергнуться только таблицы 9.9 и 9.10, а все остальные не изменяются. В таблице 9.10 сведем кодировку внутренних состояний и выходных сигналов.

Таблица 9.11 – Таблица кодирования состояний и выходов

	Булевы переменные выходов триггеров		Булевы переменные выходных сигналов
	abcd		$y_1y_2y_3y_4$
a_1	0000	w_1	0000
a_2	0100	w_2	0100
a_3	1111	w_3	1111
a_4	0110	w_4	0110
a_5	0010	w_5	0010
a_6	0101	w_6	0101

В таблице 9.12 приведем закон переходов триггера *D*.

Таблица 9.12 – Переходы триггера *D*

***Q*(*t*)**	***Q*(*T*+1)**	***D***
0	0	0
0	1	1
1	0	0
1	1	1

Сведем эти данные в единую таблицу переходов-выходов (табл. 9.13).

Таблица 9.13 – Таблица переходов-выходов автомата

***a*(*t*)**	**Код *a*(*t*)**	***a*(*t*+1)**	**Код *a*(*t*+1)**	***D*1**	***D*2**	***D*3**	***D*4**	***W***	**Код *W***
	abcd								***abcd***
a_1	0000	a_2	1110	1	1	1	0	w_1	0000
a_2	1110	a_3	0111	0	1	1	1	w_2	1110
a_3	0111	a_4	0011	0	0	1	1	w_3	0111
a_4	0011	a_5	1100	1	1	0	0	w_4	0011
a_5	1100	a_6	1001	1	0	0	1	w_5	1100
a_6	1001	a_1	0000	0	0	0	0	w_6	1001

Функции *D* всех триггеров зависят только от состояний $a(t)$, так как входной сигнал $z = 1$, фактически завися от четырех переменных, поэтому нанесем их на карты Карно – Вейча и минимизируем. На картах клетки, соответствующие состояниям автомата, отмечены серым цветом. На всех остальных клетках карты ставится знак неопределенности булевой функции. Реализуем функции в форме КНФ (рис. 9.11).

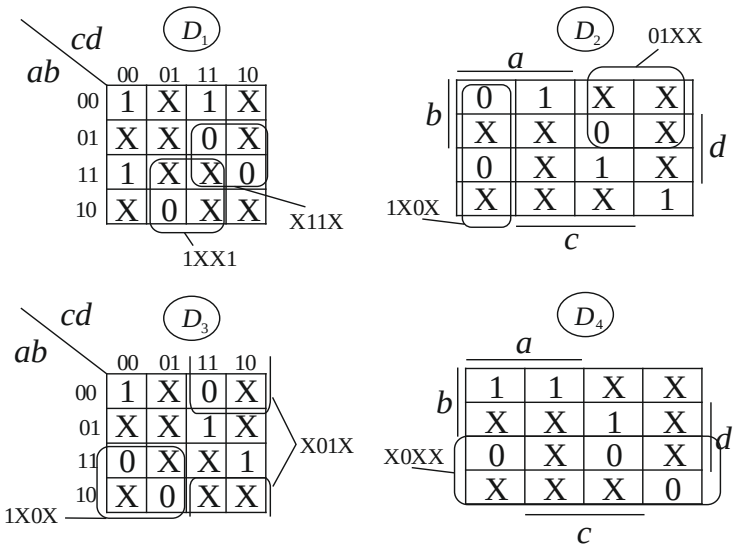


Рис. 9.11 – Карты Карно и Вейча функций D_1, D_2, D_3, D_4.

$$D_1 = (\overline{a} + \overline{c})(\overline{b} + \overline{c}),\ \ D_2 = (\overline{a} + c)(a + \overline{b}),\ \ D_3 = (\overline{a} + c)(b + \overline{c}),\ \ D_4 = b$$

Функции выходов: $y_1 = a_1$, $y_2 = b_2$, $y_3 = c_3$, $y_4 = d_4$. На основе всех этих уравнений строим схему (рис. 9.12).

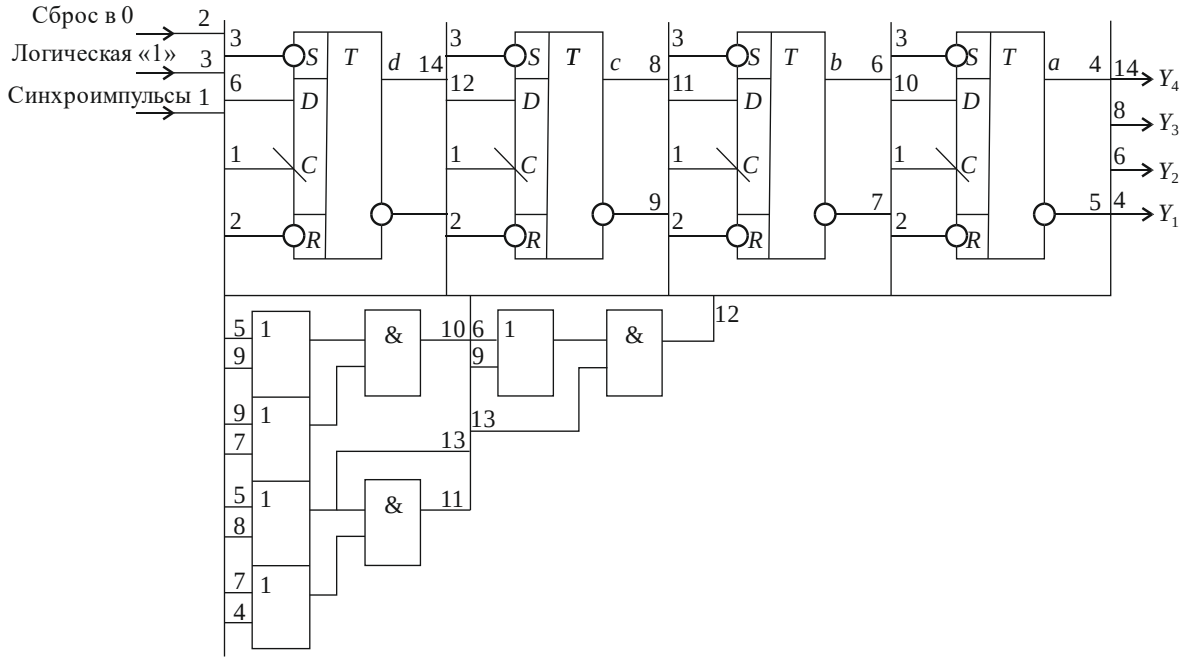


Рис. 9.12 – Функциональная схема автомата на триггерах *D*

Если понадобился счетчик-делитель, а готовых счетчиков нет в наличии, то его можно спроектировать и реализовать на отдельных триггерах.

Задача 9.4. Автомат задан графом (рис. 9.13). Спроектировать счетчик – делитель на 5 на триггерах типа *J-K*, комбинационная схема должна быть выполнена на любых элементах.

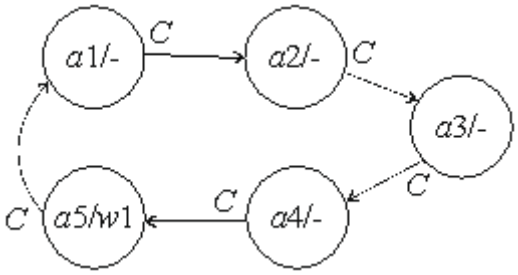


Рис. 9.13 – Граф счетчика – делителя на 5

У этого автомата пять состояний, которые меняются от 0 до 4. В двоичном коде 4 = 100, поэтому потребуется только три триггера. Синтез счетчика-делителя проведем по известному алгоритму.

В таблице 9.14 сведем кодировку внутренних состояний и выходных сигналов.

Таблица 9.14 – Таблица кодирования состояний и выходов

	Булевы переменные выходов триггеров		Булевы переменные выходных сигналов
	abc		y_1
a_1	000	w_1	1
a_2	001		
a_3	010		
a_4	011		
a_5	100		

В таблице 9.15 приведем закон переходов триггера *D*.

Таблица 9.15 – Переходы триггера *J-K*

***Q*(*t*)**	***Q*(*T*+1)**	***J***	***K***
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Сведем эти данные в единую таблицу переходов-выходов таблице 9.16.

Таблица 9.16 – Таблица переходов-выходов автомата

***a*(*t*)**	**Код *abc***	***a*(*t*+1)**	**Код *a*(*t*+1)**	J_1	K_1	J_2	K_2	J_3	K_3	***W***	**Код *W***
a_1	000	a_2	001	0	X	0	X	1	X	-	-
a_2	001	a_3	010	0	X	1	X	X	1	-	-
a_3	010	a_4	011	0	X	X	0	1	X	-	-
a_4	011	a_5	100	1	X	X	1	X	1	-	-
a_5	100	a_6	000	X	1	0	X	0	X	w_1	1

Функции *J-K* всех триггеров зависят от состояний $a(t)$ ($z = 1$), т. е. от трех переменных. Нанесем их на карты Карно и минимизируем. На картах клетки, соответствующие состояниям автомата, отмечены серым цветом. На всех остальных клетках карты ставится знак неопределенности булевой функции. Реализуем функции в форме ДНФ (рис. 9.14).

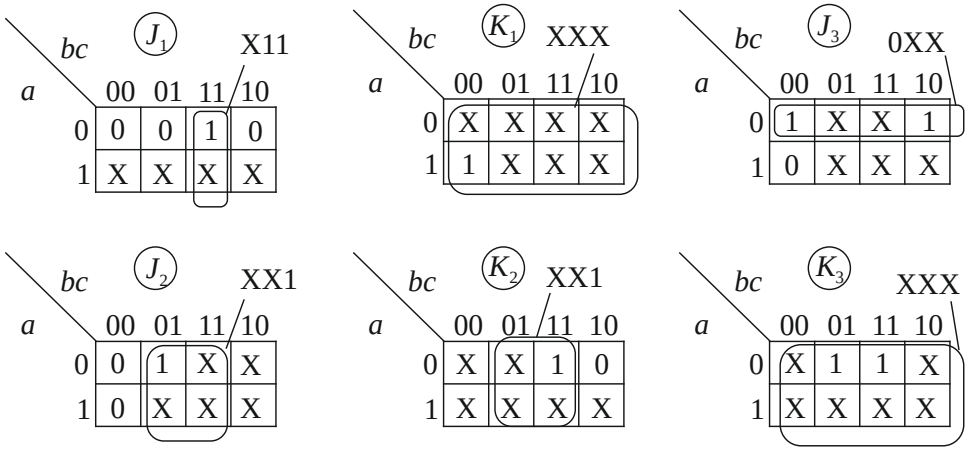


Рис. 9.14 – Карты Карно функций J_1, K_1, J_2, K_2, J_3, K_3.

$$J_1 = bc,\ K_1 = 1,\ \ J_2 = c,\ K_2 = c,\ \ J_3 = \bar{a},\ K_3 = 1,\ y_1 = a$$

На картах функций K_1 и K_3 имеются клетки, отмеченные только единицами и крестами. Эти функции тождественно равны 1. В соответствии с этими функциями построим схему счетчика-делителя (рис. 9.15).

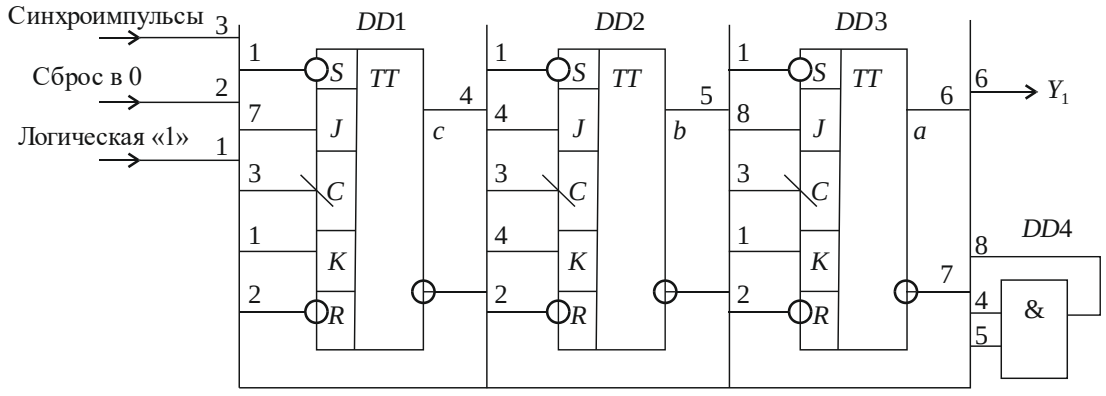


Рис. 9.15 – Функциональная схема счетчика – делителя на 5

На рисунке 9.16 приведена диаграмма работы счетчика-делителя. Диаграммы на рисунке отличаются только длительностью развертки – в первом случае ее частота в два раза выше.

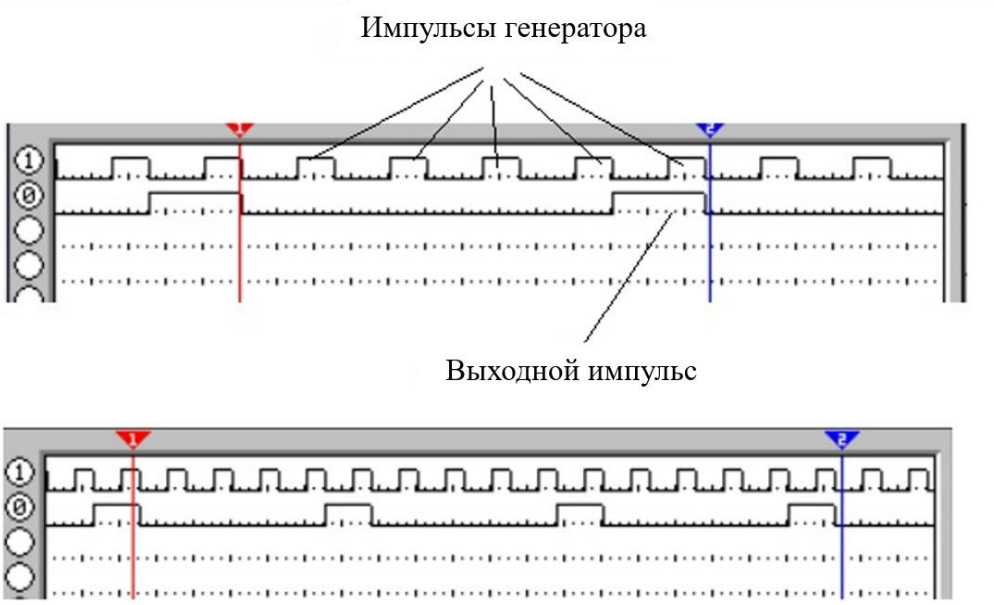


Рис. 9.16 – Диаграмма работы счетчика – делителя на 5

Из диаграммы видно, что на каждые пять импульсов генератора образуется один выходной импульс с прямого выхода триггера *DD*3 (между красным и синим маркерами). На нижней диаграмме показаны четыре периода работы этого счетчика.

Контрольные вопросы по главе 9

1. Чем отличаются абстрактные автоматы Мили и Мура?
2. Что такое абстрактный автомат?
3. Что такое структурный автомат?
4. В таблице переходов-выходов абстрактного автомата Мура указано 21 состояние. Сколько триггеров потребуется при схемной реализации такого автомата?
5. В каких случаях выходные сигналы автомата должны реализовываться в комбинационной схеме?
6. При использовании каких триггеров – *D* или *J-K* – таблица переходов-выходов одного и того же автомата проще?

10 Микропроцессоры. Микроконтроллеры

Цифровые устройства являются неотделимой частью средств вычислительной техники. Цифровые логические элементы и узлы используются вычислительными устройствами, в том числе и электронными вычислительными машинами, как кирпичиками, из которых создаются здания различной конфигурации и назначения.

Поэтому, когда встал вопрос о необходимости создания ЭВМ, то, из чего их строить, не вызывало сомнений. Тем более что построение первых ЭВМ основывалось на принципах, изложенных Д. фон Нейманом, Г. Голдстайном и А. Берксом. Эти принципы возникли на основе опыта, накопленного при разработке первой ЭВМ – ЭНИАК. На основе этих принципов создавались ЭВМ различных поколений.

Основные принципы Д. фон Неймана:

1. В состав ЭВМ должны входить:
   - устройство для выполнения арифметических и логических операций АЛУ;
   - оперативное запоминающее устройство для хранения данных и программ их обработки;
   - устройство управления для управления процессом исполнения программ обработки данных;
   - внешние устройства, предназначенные для ввода в ЭВМ и вывода данных из ЭВМ.
2. Данные кодируются двоичным кодом и могут делиться на группы, называемые словами.
3. Алгоритм обработки данных представляется в виде последовательности управляющих слов, называемых командами и объединенных в программу. Команды программы выполняются последовательно, но должен быть способ для условного перехода к любой части программы.
4. Ячейки памяти для хранения данных и кодов команд имеют номера, называемые адресом ячейки. В любой момент времени можно обратиться к любой ячейке памяти, т. е. использовать ОЗУ с произвольной выборкой.
5. Программы и данные хранятся в одной памяти.

АЛУ ЭВМ может содержать различные узлы. К ним относятся регистры для хранения данных и результатов их обработки, комбинационные схемы, реализующие арифметические, логические операции и операции сдвига, операции инвертирования и т. п. АЛУ и устройство управления (УУ) часто объединяют в одно и называют его *центральным процессором* (*ЦП*). ЦП выполняет все действия для считывания команд из оперативной памяти и их выполнения (рис. 10.1). Три поколения ЭВМ строились в соответствии с принципами фон Неймана и представляли собой довольно громоздкие устройства, занимающие большие площади, энергоемкие, малонадежные и не обладающие высоким быстродействием.

Схема вычислительной машины фон Неймана

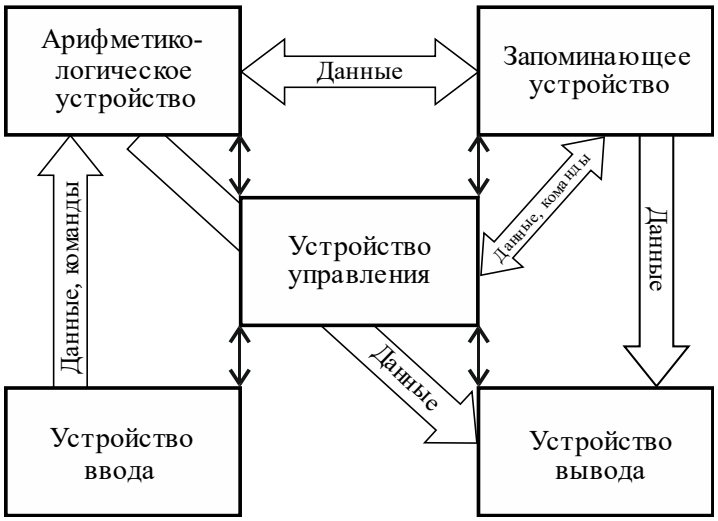


Рис. 10.1 – Структура фоннеймановской ЭВМ

С развитием микроэлектроники и появлением микросхем с высокой плотностью размещения элементов центральный процессор фоннеймановской структуры удалось реализовать на одном кристалле. Так появилось устройство, названное микропроцессором.

Микропроцессор *– программно управляемое электронное устройство для обработки цифровой информации и управления процессом этой обработки, изготовленное с высокой степенью плотности электронных компонентов.*

Разделим микропроцессоры на два класса. К одному отнесем такие, которые обладают возможностями выполнять широкий класс вычислительных задач,

т. е. их можно назвать *универсальными*. Это мощные вычислители, на которых строятся компьютеры, работающие под управлением операционных систем, с широким спектром прикладного программного обеспечения, такого как пакеты прикладных программ, офисные пакеты, сложные игры, тренажеры и т. п. Такие микропроцессоры реализуют центральный процессор ЭВМ, и к ним необходимо подключать различное оборудование для получения полноценной ЭВМ.

Ко второму классу отнесем микропроцессоры, уже имеющие в своем составе все необходимое оборудование – память и развитую систему ввода-вывода. Такие микропроцессоры получили название микро-ЭВМ или микроконтроллеров (МК). МК предназначены для решения узкого круга задач или даже одной задачи, т. е. являются в большей степени *специализированными*. Среди них есть такие МК, которые могут использоваться для решения задач управления широким спектром разнообразного оборудования (елочные гирлянды, бытовые приборы, станки с числовым программным управлением, различные системы воздушных и водных судов, автомобилей и т. д.). Если взять персональный компьютер, то ЦП в нем один, а МК около десятка.

10.1 Структура микропроцессора

Рассмотрим структуру микропроцессора на примере Intel 8080, имеющего фоннеймановскую архитектуру, и на котором строился один из первых персональных компьютеров (рис. 10.2). Это однокристальный, восьмиразрядный цифровой микропроцессор.

Основу составляет внутренняя двунаправленная магистраль, по которой перемещаются все потоки информации.

К АЛУ подключены регистр А (аккумулятор), регистр состояния (регистр флагов), десятичный корректор для операций с числами в двоично-десятичном коде и вспомогательные регистры.

Блок регистров общего назначения (РОН) предназначен для временного хранения восьми- и шестнадцатиразрядных данных.

Регистр команд хранит код выполняемой команды (операции). Дешифратор команд передает в устройство управления данные для формирования машинного цикла выполнения данной команды. Считайте, что для каждой команды устройство управления (система управления и синхронизации) формирует свой машинный цикл.

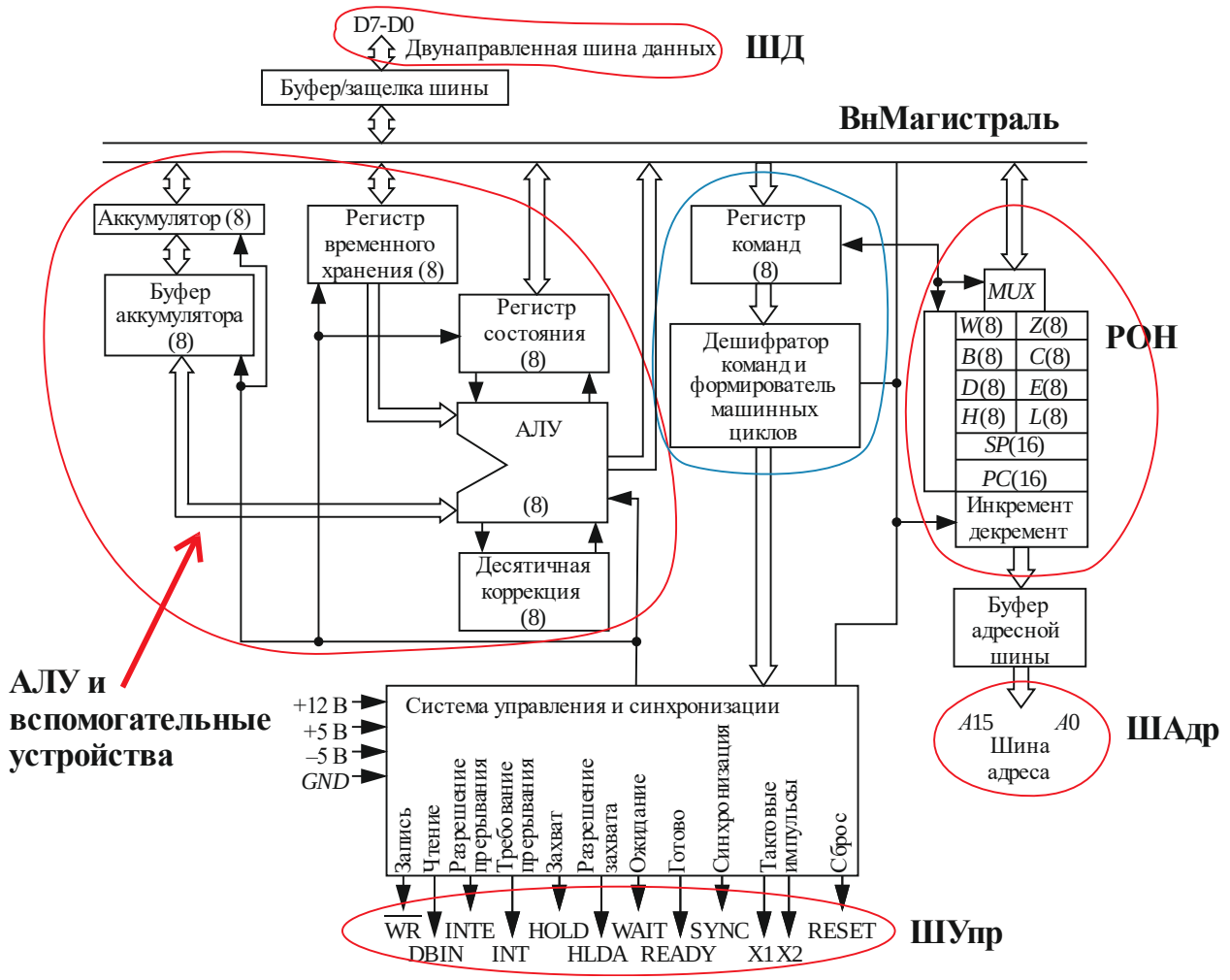


Рис. 10.2 – Структура микропроцессора Intel 8080

Микропроцессор для взаимодействия с внешним миром использует три шины, которые создают системную магистраль ЭВМ.

Шина адреса (ШАдр) – это однонаправленная шина, по которой микропроцессор передает в ОЗУ адрес ячейки памяти или формируется адрес внешнего устройства, к которому обращается программа. Все ячейки ОЗУ и ПЗУ, а также контроллеры внешних устройств имеют свои уникальные адреса. Множество этих адресов создают *адресное пространство.* Микропроцессор формирует на ШАдр двоичный код адреса того внешнего устройства (или ячейки ОЗУ), с которым желает выполнить обмен данными.

Шина данных (ШДан) представляет собой двунаправленную магистраль, по которой числовые данные поступают в микропроцессор или выводятся из него. Все выходы устройств, подключенные к этой шине, могут переключаться в *Z*-состояние.

Шина управления (ШУ) предназначена для формирования некоторого количества управляющих (синхронизирующих) сигналов, управляющих процессом передачи данных.

Таким образом, структура ЭВМ с центральным микропроцессором имеет вид *трехшинной* открытой системы (рис. 10.3). Все шины вместе образуют системную шину. Все устройства подключаются к системной шине параллельно, одинаковым способом. Это унифицирует создание схем подключения к системной шине любого устройства.

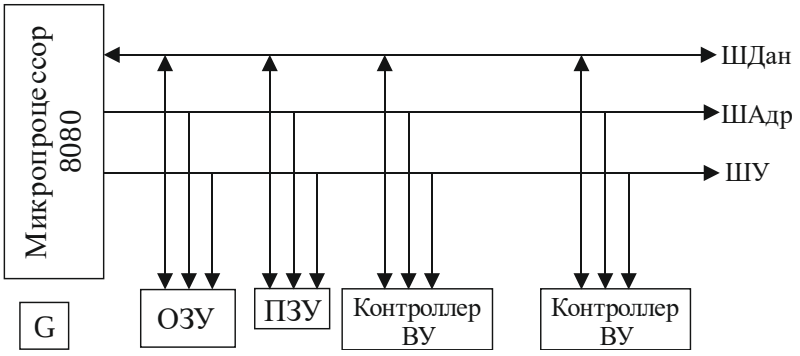


Рис. 10.3 – Трехшинная структура системы с Intel 8080

Устройства управления микропроцессора формируют внутренние машинные циклы, состоящие из нескольких периодов частоты тактового генератора *G*. Любая из команд, входящая в систему команд микропроцессора, выполняется за один или несколько машинных циклов.

Для обмена данными микропроцессор аппаратно формирует цикл чтения или цикл записи. Чтобы обратиться к любому устройству (ОЗУ или ВУ), микропроцессор выставляет на шину адреса (ШАдр) адрес ячейки памяти ОЗУ или адрес ВУ. Для записи данных из микропроцессора он на шину данных одновременно с адресом выставляет данные, затем на ШУ выставляется сигнал записи. Чтобы прочитать данные из ОЗУ или ВУ, сначала на ШАдр выставляется адрес, а затем на ШУ формируется сигнал чтения. Данные из выбранной ячейки ОЗУ или ВУ выставляются на шину данных и попадают в микропроцессор.

С развитием микропроцессорных устройств увеличилась разрядность данных до 16, 32, 64 разрядов. Емкости ОЗУ расширились до мегабайт и гигабайт, что увеличило разрядность шины адреса с 16 до 20–30 разрядов. Это привело к неудобствам в создании печатных плат устройств, т. к. пришлось проводить в разные части платы много проводников для передачи сигналов адреса и данных.

Дальнейшее развитие микропроцессоров привело к изменению их архитектуры, появление кешей, конвейеров для ускорения выполнения команд и многого другого. Вместо трех шин системную магистраль образовали из двух шин: шины адреса/данных (ШАД или ШАдр/Дан) и шины управления (рис. 10.4).

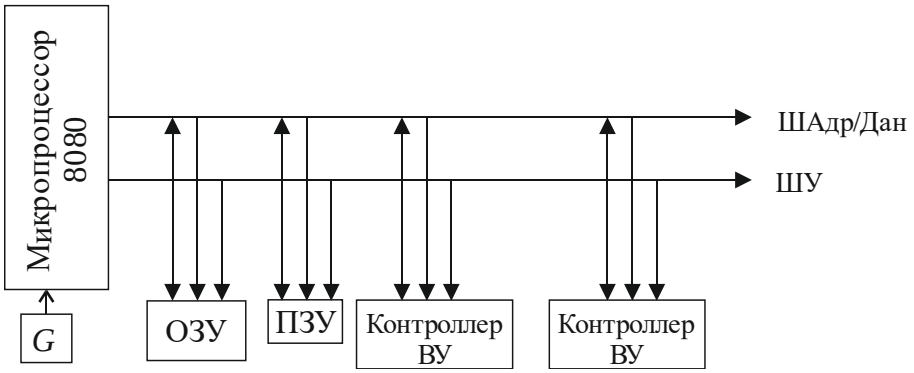


Рис. 10.4 – Двухшинная структура

Отличие в работе от трехшинной структуры заключается в том, что передачи адреса и данных разделили во времени. Сначала передается адрес, а затем он снимается и по этой же шине передаются данные. Разрядность этой шины зависит от заданной величины адресного пространства и разрядности данных.

Архитектура микропроцессоров зависит и от системы команд для обработки данных и методов адресации, т. е. способов, с помощью которых определяется месторасположение данных.

С этой точки зрения существуют две основные архитектуры, по которым строятся микропроцессоры: микропроцессоры с CISC-архитектурой и RISC-архитектурой. CISC-архитектура – это архитектура с полным набором команд. Она предполагает ограниченное количество вспомогательных регистров, много форматов команд различной длины, много методов адресации операндов. Такая архитектура присуща универсальным микропроцессорам.

10.2 Микроконтроллеры

Как показал опыт программирования, многие программисты для решения задач управления из всего множества команд CISC используют только ограниченное их количество. Таким образом произошел переход к RISC-архитектуре. RISC-архитектура использует меньший, по сравнению с CISC, набор команд. При этом в аппаратную часть ввели большее количество регистров с расширенными возможностями по адресации операндов и выполнению арифметических

и логических команд. Современные микроконтроллеры в большинстве своем строятся по RISC-архитектуре, и в построении универсальных микропроцессоров также имеется тенденция к ее использованию.

Рассмотрим класс микропроцессов, называемых микроконтроллерами (МК) или управляющими контроллерами. МК являются специализированными устройствами, имеющими признаки ЭВМ и предназначенными для управления различными устройствами, от самых простых и до сложных систем.

В отличие от универсальных микропроцессоров к управляющим контроллерам, как правило, не предъявляются высокие требования к производительности и программной совместимости. Они не предназначены для выполнения сложных математических расчетов. Основные требования, которые потребители предъявляют к МК, входящим в состав устройств, можно сформулировать следующим образом:

- низкая стоимость;
- высокая надежность;
- высокая степень миниатюризации;
- малое энергопотребление;
- работоспособность в жестких условиях эксплуатации;
- достаточная производительность для выполнения всех требуемых функций.

Выполнение всех этих довольно противоречивых условий одновременно затруднительно, поэтому развитие и совершенствование техники пошло по пути специализации и в настоящее время количество различных моделей управляющих микроконтроллеров чрезвычайно велико.

Однако можно выделить некоторые черты архитектуры и системы команд, общие для всех современных микроконтроллеров:

- так называемая гарвардская архитектура, т. е. раздельные области памяти для хранения команд (программы) и памяти хранения данных;
- интеграция в одном корпусе микросхемы (на одном кристалле) практически всех блоков, характерных для полнофункционального компьютера – процессора, ПЗУ, ОЗУ, устройств ввода-вывода, тактового генератора, системы прерываний и т. д. В русскоязычной литературе подобные устройства часто называются однокристальные ЭВМ (ОЭВМ);
- использование RISC-архитектуры.

Архитектура RISC, помимо упрощенной системы команд, строится так: команды выполняются за короткое время, как правило, за один машинный цикл. Количество общих регистров существенно больше. Количество форматов команд существенно меньше, чем в CISC-процессорах.

Одним из основных отличий МК является гарвардская архитектура. В отличие от фоннеймановской гарвардская архитектура, как уже было сказано, разделяет память на две части – память программ и память данных.

Память программ – это постоянное запоминающее устройство (ПЗУ), в которое «зашивается» заранее подготовленная и отлаженная программа. В процессе работы устройства с МК эта программ изменению не подлежит. В некоторых МК предусмотрена возможность переписи (перепрошивки) программы, если в нее внесены усовершенствования или если МК переносится в совершенно другое устройство со своим алгоритмом работы. Объем памяти программ меняется от единиц до десятков килобайт.

Память данных – это оперативное запоминающее устройство, как правило небольшой емкости – 128 (256) байт. Эту память еще называют блоком *регистров общего назначения*. В ряде МК предусмотрена возможность расширения памяти программ и памяти данных, путем подключения к МК внешних ОЗУ и ПЗУ. При этом МК работает только с внешней памятью программ.

МК выпускают большое количество фирм, а номенклатура и характеристики весьма многообразны:

- четырехразрядные – самые простые и дешевые;
- восьмиразрядные – наиболее многочисленная группа (оптимальное сочетание цены и возможностей), к этой группе относятся микроконтроллеры серии MCS-51 (Intel) и совместимые с ними, PIC (MicroChip), HC68 (Motorola), Z8 (Zilog) и др.;
- шестнадцатиразрядные – MCS-96 (Intel) и др. – более высокопроизводительные, но более дорогостоящие;
- тридцатидвухразрядные – обычно являющиеся модификациями универсальных микропроцессоров, например i80186 или i386EX.

Широко известны микроконтроллеры Atmega, выпускаемые фирмами Microchip, Atmel, используемые платформой Arduino, относящиеся к AVR-контроллерам. Их применяют в радиолюбительской и программистской практике для создания различных игровых моделей и устройств, таких как гирлянды, автомобили, вертолеты, коптеры и т. п.

К сожалению, практически все фирмы в документации не раскрывают структуры микроконтроллеров полностью, предоставляя только укрупненную структурную схему.

В связи с этим мы рассмотрим структуру и работу классического микроконтроллера, с которого и началось их широкое применение в различных областях. Этот микроконтроллер де факто стал основой для разработки всех последующих микроконтроллеров. Микроконтроллер был разработан фирмой Intel, правда, затем он выпускался и многими другими производителями микросхем, в том числе и в России. Фирма Intel присвоила ему индекс 8051. Отечественная разработка получила название КМ1816ВЕ51 или, по-другому, МК51.

10.3 Структура микроконтроллера МК51 (Intel 8051)

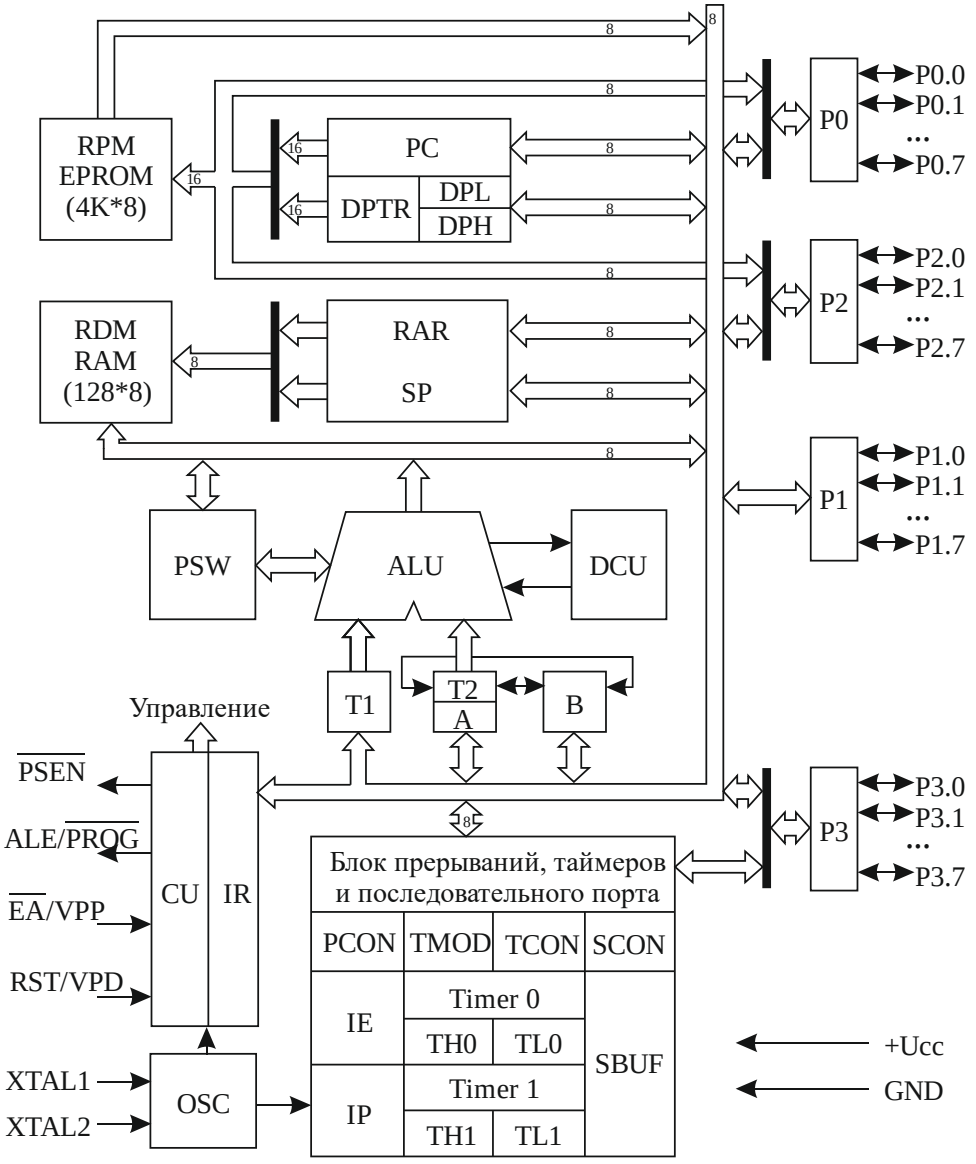


Рис. 10.5 – Структурная схема МК51

Основу МК51 (рис. 10.5) составляет внутренняя двунаправленная восьмиразрядная магистраль данных, по которой байтовая информация передается во все узлы.

Центральной частью МК51 является арифметико-логическое устройство (АЛУ, ALU), в котором выполняются все арифметические и логические команды. С АЛУ непосредственно связаны регистры А и В, двоично-десятичный корректор (DCU) и регистр флагов (PSW). Само АЛУ выполнено в виде комбинационной схемы и вместе с регистрами относится к типу РАЛУ (регистровое АЛУ).

Регистр А (аккумулятор – Acc) является основным регистром, участвующим в выполнении большинства арифметических и логических операций, хотя ряд таких операций выполняется и без его участия.

Регистр В – это вспомогательный регистр. Его основным назначением является участие в выполнении команд умножения и деления, которые аппаратно выполняются над однобайтовыми числами. При умножении $A \cdot B$ старший байт полученного находится в аккумуляторе, а младший – в регистре В. При делении A/B в регистре В размещается *целая часть* деления, а в аккумулятор А складывается *остаток* от деления.

ALU может оперировать четырьмя типами информационных объектов: булевыми (1 бит), цифровыми в двоичном и двоично-десятичном кодах, байтными по 8 бит и адресными по 16 бит. В ALU выполняется 51 различная операция пересылки или преобразования этих данных.

В командах МК51 используется 11 методов адресации (7 для данных и 4 для адресов). Путем комбинирования в командах операции и метода адресации базовое число из 111 команд расширяется до 255 из 256 возможных при однобайтном коде операции.

Регистр флагов (PSW – слово состояния процессора) отображает ситуации, возникающие при выполнении арифметических, логических, сдвиговых команд, проверки на ноль, паритета (табл. 10.1).

Наиболее часто используется флаг переноса С, который принимает участие и изменяется в процессе выполнения многих операций (сложение, вычитание и сдвиги). Кроме того, флаг переноса обязательно используется в командах с битами.

Флаг переполнения (OV) поднимается (устанавливается в 1) при арифметическом переполнении, если выполняются операции над целыми числами со знаком, представленным в дополнительном коде.

Биты выбора банка регистров (RS0, RS1) изменяются только программой и используются для выбора одного из четырех регистровых банков.

Флаг F0 может использоваться программистом по своему усмотрению, как флаг пользователя.

Флаг AC – флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или займе в битах 3 и 4 аккумулятора. Используется для формирования корректирующего кода при выполнении операций с числами в двоично-десятичном коде.

Флаг P – паритет – устанавливается и сбрасывается только программно. Если при выполнении команды число в аккумуляторе содержит четное или нечетное количество единиц, то этот флаг взводится или сбрасывается.

Таблица 10.1 – Регистр флагов (PSW)

<table>
<tr><th>Символ</th><th>Номер разряда</th><th>Имя и назначение</th></tr>
<tr><td>P</td><td>PSW.0</td><td>Флаг приоритета. Устанавливается и сбрасывается аппаратурно</td></tr>
<tr><td>–</td><td>PSW.1</td><td>Не используется</td></tr>
<tr><td>OV</td><td>PSW.2</td><td>Флаг переполнения. Устанавливается и сбрасывается аппаратурно при выполнении арифметических операций над числами со знаком</td></tr>
<tr><td>RS0–RS1</td><td>PSW.3–PSW.4</td><td>Биты выбора используемого банка регистров
<table>
<tr><th>RS0</th><th>RS1</th><th>Банк</th><th>Границы адресов ОЗУ</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00H–07H</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>08H–0FH</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>10H–17H</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18H–1FH</td></tr>
</table>
</td></tr>
<tr><td>F0</td><td>PSW.5</td><td>Флаг пользователя</td></tr>
<tr><td>AC</td><td>PSW.6</td><td>Флаг вспомогательного переноса</td></tr>
<tr><td>C</td><td>PSW.7</td><td>Флаг переноса. Устанавливается и сбрасывается как аппаратурно, так и программным путем. Этот бит имеет еще одно обозначение – CY</td></tr>
</table>

10.3.1 Память программ и данных

Память программ (ПП) RPM представляет собой ПЗУ, в которое загружается исполняемая МК51 программа. Память данных (ПД) RDM – это оперативное запоминающее устройство, предназначенное для хранения числовых данных. Если ПП и ПД размещены на самом кристалле микроконтроллера, то их называют резидентными (РПП и РПД). Память программ и память данных являются самостоятельными и независимыми друг от друга устройствами, адре-

суемыми различными командами и управляющими сигналами. Объем РПП может составлять 1, 2, 4, 8 Кбайт, реже 32 Кбайт (чем больше память, тем дороже микросхема).

В МК51, как и в других микроконтроллерах, предусмотрена возможность подключения внешней памяти (микросхемы ОЗУ и ПЗУ).

При обращении к *внешней памяти программ* (ВПП) МК51 всегда использует 16-разрядный адрес, что обеспечивает им доступ к 64 Кбайт ПЗУ. Микроконтроллер обращается к ВПП, формируя адрес 16-разрядным счетчиком команд (PC) и двухбайтным регистром DPTR, старший байт которого носит имя DPH, а младший – DPL.

Объем *внешней памяти данных* (ВПД) у МК51 может достигать 64 Кбайт.

Объем *РПД* у МК51 составляет – 128 байт (у других микроконтроллеров может быть и больше). При адресации ячеек памяти можно использовать десятичные (0–127) или шестнадцатеричные (0h–7Fh) обозначения адреса. На рисунке 10.6 представлены адресные пространства РПД.

Адрес байта								
0FFH								
- - -								
030H								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	9	8
20H	7	6	5	4	3	2	1	0
1FH								
	Банк 3							
18H								
17H								
	Банк 2							
10H								
0FH								
	Банк 1							
08H								
07H								
	Банк 0							
00H								

Адрес байта	Адреса битов по разрядам								Имя регистра
	D7	D6	D5	D4	D3	D2	D1	D0	
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
…	…								…
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC(A)
…	…								
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
…	…								…
B8H	-	-	-	BC	BB	BA	B9	B8	IP
…	…								…
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
…	…								…
A8H	AF	-	-	AC	AB	AA	A9	A8	IE
…	…								…
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
…	….								…
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
…	…								…
90H	97	96	95	94	93	92	91	90	P1
…	…								…
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
…	…								…
80H	87	86	85	84	83	82	81	80	P0

Рис. 10.6 – Распределение адресов РПД и битов регистров

Первые 32 байта организованы в четыре банка регистров общего назначения (РОН), обозначаемых, соответственно, банк 0, 1, 2, 3. Каждый из них состоит из восьми регистров, имеющих обозначения (имена) R0, R1, R2, …, R7. В любой момент программе доступен только один банк регистров, номер которого содержится в третьем и четвертом битах *PSW* (табл. 10.1). Программист изменяет в программе содержимое этих битов и обращается к любому другому банку, в котором РОНы также имеют имена R0, R1, R2, …, R7. Это удобно: когда исполняемую программу пишут несколько программистов, то каждый пользуется своим банком. К любой ячейке памяти можно обратиться и по адресу ее байта, например, к R0 третьего банка можно обратиться и по ее адресу 17H (17h), где H (h) означает шестнадцатеричный код (рис. 10.6).

Часть памяти данных представляет собой так называемую битовую область, в ней имеется возможность при помощи специальных битовых команд адресоваться к каждому разряду ячеек памяти. Адрес прямо адресуемых битов может быть записан либо в виде *21.3*, что означает третий разряд ячейки памяти с адресом 21H, либо в виде абсолютного битового адреса *0Ah* (рис. 10.6).

Таблица 10.2 – Адреса внутренних регистров специальных функций

Обозначение	**Имя**	**Адрес (шестнадцатеричный)**
***** ACC	Аккумулятор	0E0H
***** B	Регистр – расширитель аккумулятора	0F0H
***** PSW	Слово состояния программы	0D0H
SP	Регистр – указатель стека	81H
DPTR	Регистр – указатель данных (DPH) (DPL)	83H
		82H
***** P0	Порт 0	80H
***** P1	Порт 1	90H
***** P2	Порт 2	0A0H
***** P3	Порт 3	0B0H
***** IP	Регистр приоритетов	0B8H
***** IE	Регистр маски прерываний	0A8H
TMOD	Регистр режима таймера/счетчика	89H
***** TCON	Регистр управления / статус таймера	88H
TH0	Таймер 0 (старший байт)	8CH
TL0	Таймер 0 (младший байт)	8AH
TH1	Таймер 1 (старший байт)	8DH
TL1	Таймер 1 (младший байт)	8BH
***** SCON	Регистр управления приемопередатчиком	98H
SBUF	Буфер приемопередатчика	99H
PCON	Регистр управления мощностью	87H

Примечание. Регистры, имена которых отмечены знаком (*****), допускают адресацию отдельных бит.

В таблице 10.2 и правой таблице рисунка 10.6 размещены имена и адреса регистров специальных функций (о них пойдет речь ниже), которые управляют работой блоков, входящих в микроконтроллер. К ним в программе тоже можно обращаться и по именам, и по их адресам. К отдельным битам этих регистров можно обращаться, указав имя регистра и через точку номер бита (например, IP.3 означает адрес третьего разряда регистра IP), или по номеру бита – BBh. Некоторые биты имеют также собственные имена.

10.3.2 Организация портов ввода/вывода

Порты ввода/вывода предназначены для осуществления связи микроконтроллеров с внешними устройствами (ВУ). Микроконтроллеры могут иметь разное число портов, обладающих различными свойствами.

В МК51 количество портов – 4. Каждый порт имеет свое название – P0, P1, P2, P3. Они могут адресоваться как регистры специальных функций (рис. 10.6). Разрядность каждого порта 1 байт. При необходимости можно обращаться к любому биту любого порта. Например, P0.0 – нулевой бит порта P0, P2.7 – старший (7) бит порта P2. Все порты двунаправленные с *Z*-состоянием (или схема с открытым коллектором), причем как весь порт, так и его отдельный бит или группа битов можно настроить на прием или передачу данных. Для настройки всего порта или его частей на прием данных из ВУ во все биты порта или в выбранные для ввода биты предварительно программно следует записать единицы.

Порты P0 и P2, кроме простых функций приема/передачи данных, используются при обращении к внешней памяти (ВПД или ВПП). С их помощью реализуется двухшинная системная магистраль, работающая с ВПП, или ВПД, или ВУ. При этом через P0 с разделением во времени вначале передается младший байт адреса внешней памяти, а затем порт переключается на прием или передачу байта данных. Порт P2 содержит старший байт адреса внешней памяти (при 16-разрядном адресе внешней памяти). При использовании 8-разрядного адреса портом P2 можно пользоваться для ввода/вывода информации обычным образом.

Порт P1 используется как порт приема или передачи данных.

Порт P3 кроме обычного ввода/вывода данных используется для формирования и приема специальных управляющих и информационных сигналов взаимодействия с ВУ. Разряды порта (все или частично) при этом выполняют альтернативные функции (табл. 10.3), которые могут быть активированы толь-

ко в том случае, если в соответствующие биты порта P3 предварительно занесены 1. Неиспользуемые альтернативным образом разряды могут работать как обычно.

Таблица 10.3 – Альтернативные функции порта P3

Вывод порта	**Альтернативная функция**
P3.0	RXD – вход последовательного порта
P3.1	TXD – выход последовательного порта
P3.2	INT0 – внешнее прерывание 0
P3.3	INT1 – внешнее прерывание 1
P3.4	T0 – вход таймера-счетчика 0
P3.5	T1 – вход таймера-счетчика 1
P3.6	WR – строб записи во внешнюю память данных
P3.7	RD – строб чтения из внешней памяти данных

По своей структуре каждый разряд порта состоит из триггера-защелки (триггер типа *D*) и дополнительных элементов. Вход *D* триггера подключен к внутренней шине данных (рис. 10.7).

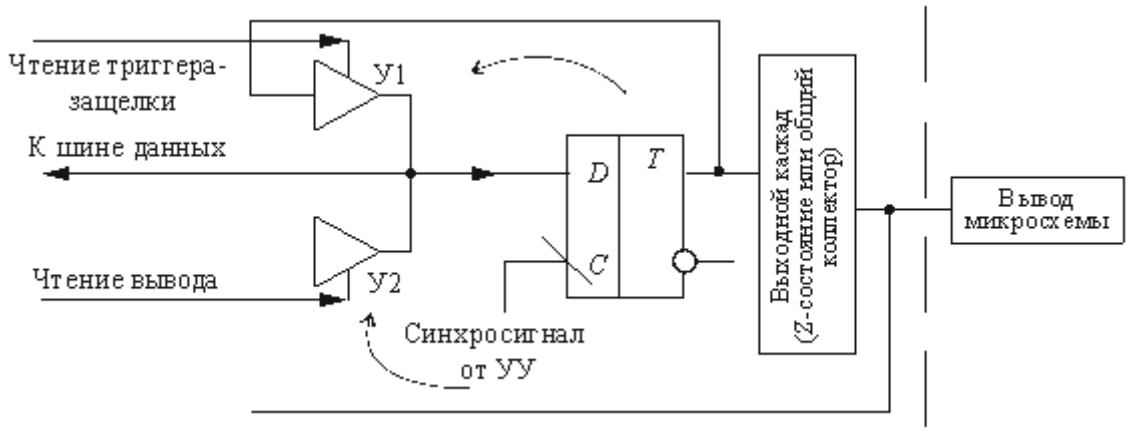


Рис. 10.7 – Структура разряда порта ввода/вывода

Выход триггера через выходной каскад подключен к выводу (пину) микросхемы, который проводами соединяется с входами или выходами внешних устройств. Предусмотрена возможность чтения состояния вывода, на который может быть подан сигнал от ВУ. Усилитель У2 по команде «Чтение вывода» снимает со своего выхода *Z*-состояние и пропускает входной сигнал, сформированный ВУ и поступивший на вывод микросхемы, на шину данных. По команде «Чтение триггера-защелки» выход У1 снимает *Z*-состояние и передает значение выхода триггера на шину данных. Очевидно, что одновременно У1 и У2 срабатывать не могут. Такая схема дает возможность проконтролировать,

что записано в триггер данного разряда и что поступает на этот разряд порта от ВУ. Различными фирмами – изготовителями микросхем этого микроконтроллера схемы портов выполняются по-разному.

На рисунке 10.8 показаны схемы каждого порта (SFR – триггер-защелка) в Intel 8051.

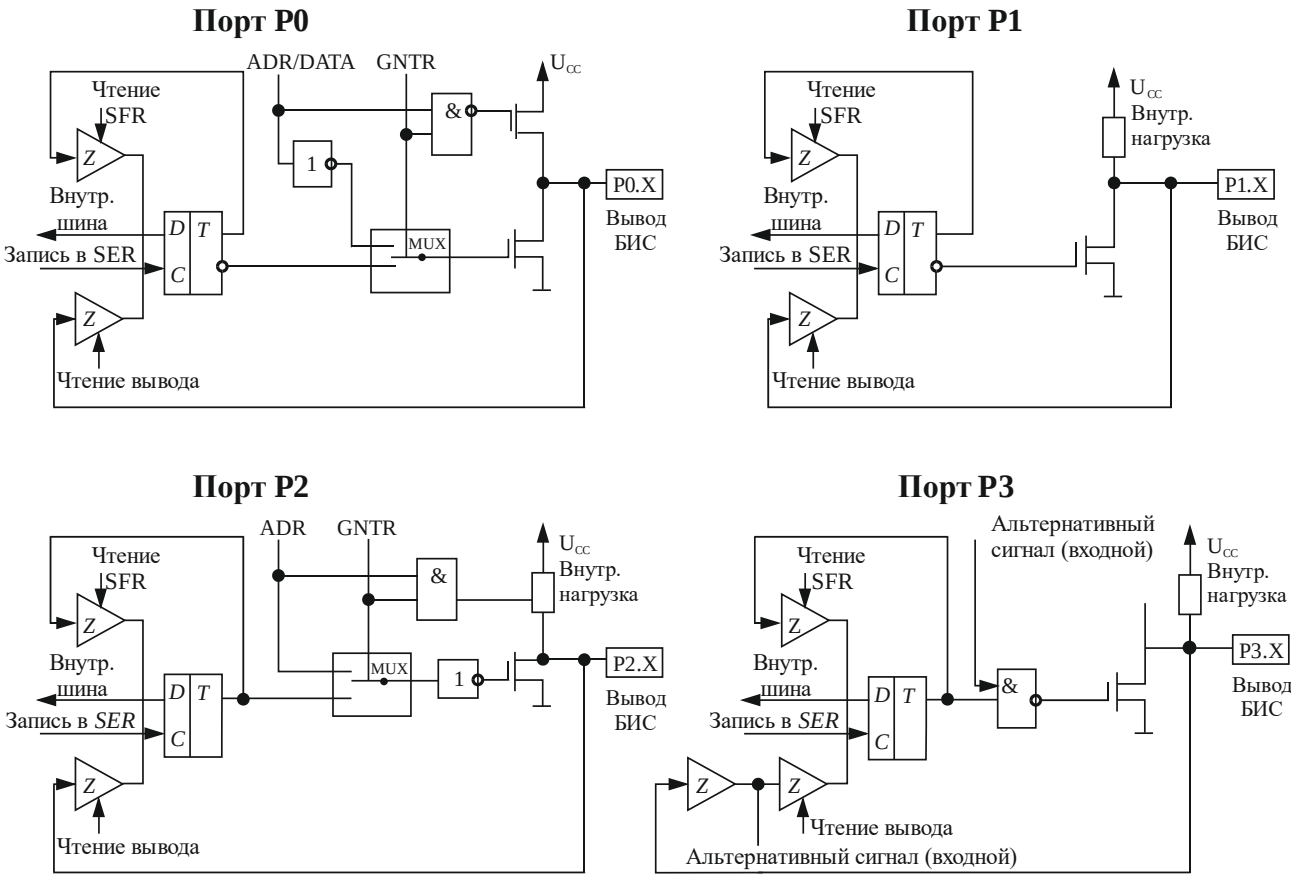


Рис. 10.8 – Схемы портов Intel 8051

Выходные каскады портов ввода/вывода маломощные (до 1,5 мА) и могут передать сигнал со своего выхода только на один вход внешней микросхемы с ТТЛ входами и выходами. Для подключения к выводам портов большей нагрузки к ним следует подключать микросхемы, имеющие более мощные выходы, например микросхемы приемопередатчиков.

10.3.3 Устройство управления и синхронизации

Кварцевый резонатор, подключаемый к внешним выводам микроконтроллера, управляет работой внутреннего генератора, который в свою очередь формирует сигналы синхронизации. Устройство управления (CU, УУ) на основе сигналов синхронизации формирует машинный цикл фиксированной длительности, равной 12 периодам резонатора.

Рис. 10.9 – Машинный цикл УУ

Машинный цикл (рис. 10.9) формируется автоматом (счетчиком – делителем на 12 и комбинационной схемой). Автомат имеет 6 состояний S1, S2, … S6. В каждом состоянии имеется две фазы – P1 и P2. Комбинационная схема, комбинируя состояния и фазы, может создавать множество синхросигналов, управляющих внутренними узлами МК51, и ряд сигналов для синхронизации взаимодействия с ВУ. К ним относятся, например, сигналы альтернативных функций порта P3, сигнал «Строб адреса внешней памяти», сигналы взаимодействия с триггерами-защелками портов, сигнал синхронизации чтения кода очередной команды памяти программ.

Большинство команд микроконтроллера выполняется за один машинный цикл. Некоторые команды, оперирующие с 2-байтными словами или связанные с обращением к внешней памяти, выполняются за два машинных цикла. Только команды деления и умножения требуют четырех машинных циклов. На основе этих особенностей работы устройства управления несложно выполнить расчёт времени исполнения программы или ее части.

На структурной схеме МК51 к устройству управления (CU) примыкает *регистр команд* (*IR*). В нем хранится код выполняемой команды, который дешифрируется и по которому автомат и КС формируют соответствующую последовательность внутренних синхросигналов. Для взаимодействия с ВУ УУ формирует или принимает от ВУ ряд сигналов. Еще ряд сигналов поступает на выводы микросхемы МК51 для поддержания его работы:

- PSEN – разрешение выборки программной памяти (для ВПП);
- PROG – сигнал программирования РПП;
- EA – блокировка работы с внутренней памятью;

- VPP – напряжение программирования;
- RST – сигнал общего сброса;
- VPD – вывод резервного питания памяти от внешнего источника;
- U_{ss} – потенциал общего провода («земли»);
- U_{cc} – основное напряжение питания +5 В;
- X1, X2 – выводы для подключения кварцевого резонатора;
- RST – вход общего сброса микроконтроллера;
- PSEN – разрешение внешней памяти программ (выдается только при обращении к внешнему ПЗУ – ВПП);
- EA – отключение внутренней программной память (уровень 0 на этом входе заставляет микроконтроллер выполнять программу только внешнее ПЗУ, игнорируя внутреннее, если последнее имеется).

10.3.4 Таймеры-счетчики

В МК51 имеется два накапливающих таймера-счетчика (нумеруются как *нулевой* и *первый*), называемых Т/С0 и Т/С1 соответственно (рис. 10.10). Каждый из них состоит из двух байт. К имени каждого байта добавляется индекс (H – старший байт, L – младший байт). Таким образом, для Т/С0 байты имеют имена TH0 и TL0, а для Т/С1 – TH1 и TL1. Каждый Т/С может работать в одном из четырех режимов, в каждом из них схемы подключения Т/С разные. В работе Т/С принимают участие регистры TMOD (регистр конфигурации) и TCON (регистр управления).

Режимы нулевой и первый

В этих режимах таймеры-счетчики соединяются последовательно. В нулевом режиме младшие байты TL0 Т/С0 и TL0 Т/С1 ограничены до пяти разрядов, образуя последовательно со старшим байтом 13-разрядный счетчик. В первом режимах работы оба Т/С имеют 16 разрядов. На рисунке 10.10 показана связь входов управления Т/С0 и Т/С1 с отдельными битами TMOD и TCON и порта Р3.

Счетные импульсы от внутреннего тактового генератора с частотой, деленной на 12, или из внешних устройств через выводы 4 и 5 порта Р3 через переключатель С/Т поступают на контакт Control и далее на счетный вход Т/С. Переключатель С/Т управляется значениями 2 и 6 битами TMOD (0-от внутреннего генератора, 1 – от внешних импульсов).

Контакт Control управляется комбинационной схемой, состоящей из элемента ИЛИ и элемента И. Сигнал Gate (TMOD.3 и TMOD.6) поступает на один из входов элемента ИЛИ через инвертор, т. е. активным является низкий уровень этого сигнала. На второй вход элемента ИЛИ сигналы поступают извне через входы INT0 и INT1 порта P3 (P3.2 и P3.3). Таким образом или Gate должен быть низким (0), или INT – высоким (1). В этом случае на одном из входов элемента И окажется высокий уровень. Высокий уровень на втором входе И формируется сигналами TR0 и TR1 из регистра PCON (PCON.4, PCON.6). Единичный сигнал с выхода схемы И замыкает Control и дает возможность импульсам от переключателя C/T поступать на счетный вход T/C.

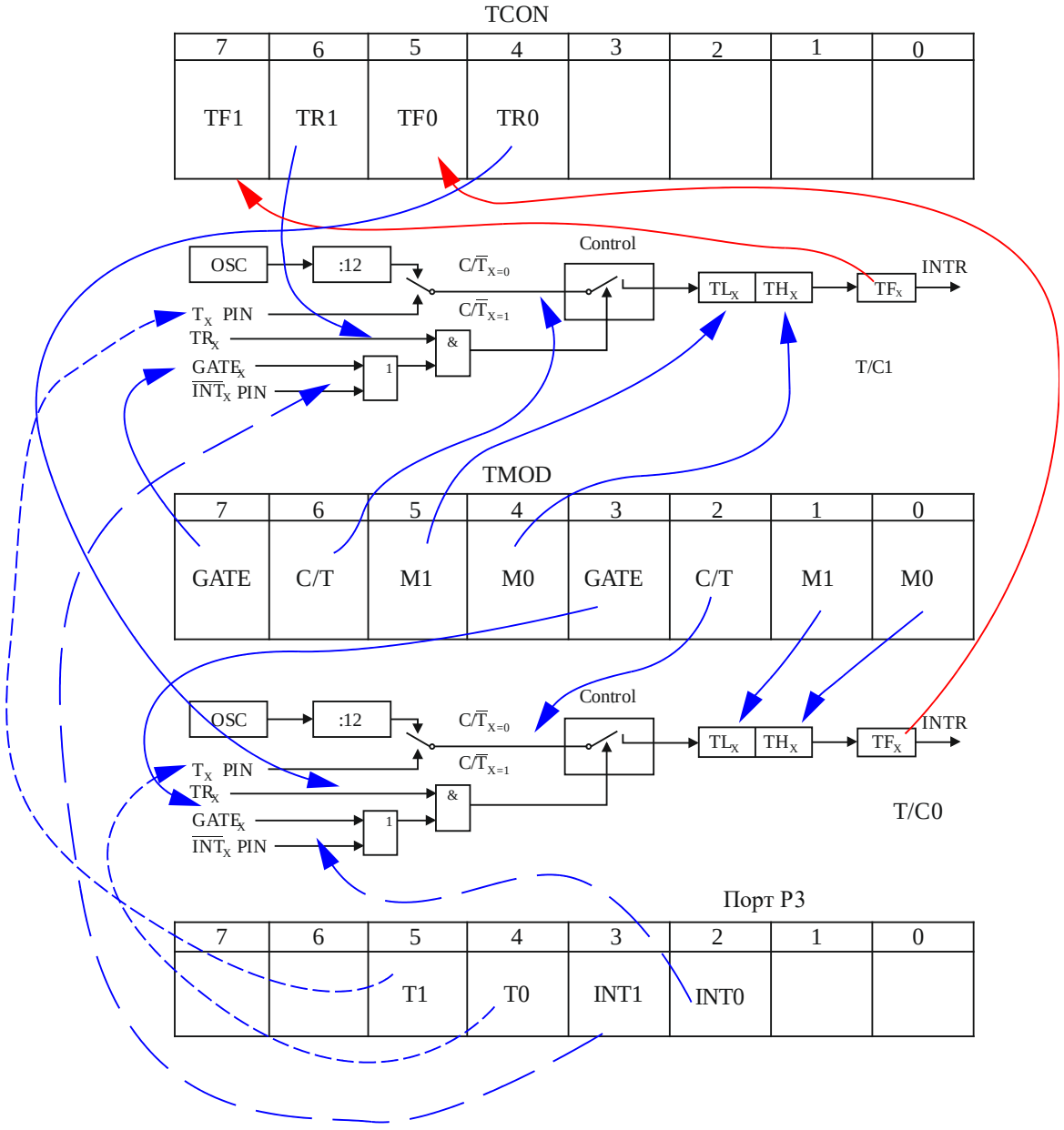


Рис. 10.10 – Структура управления таймерами-счетчиками в режимах 0 и 1

При работе Т/С возникает переполнение (перенос из старшего разряда) когда все разряды счетчика переходят из состояния 1 в 0. Это событие запоминается в триггерах TF0 и TF1 (TCON.5 и TCON.6). При соответствующей настройке эти сигналы вызывают программы обработки этой ситуации (программы обработки прерывания).

В каждый байт любого Т/С можно предварительно занести число. В этом случае Т/С начнет счет не с нуля, а с числа, записанного заранее. При тактовой частоте внутреннего генератора F = 12Мгц на входы Т/С поступают импульсы с частотой F/12 = 1Мгц (период равен 1 мкс). Таким образом, в зависимости от предварительно записанных числа в байты Т/С можно задавать временные отрезки.

Каждый бит регистра TMOD может быть взведен или сброшен программно. Управляя битами TMOD, можно сконфигурировать Т/С на различные режимы работы. Регистр TCON в этом отношении отличается тем, что биты TF0 и TF1 взводятся УУ МК51 при переполнении соответствующего (Т/С0 и Т/С1). Сбросить их можно программно или автоматически при переходе к программе обработки прерывания (об этом речь пойдет ниже). Биты М0 (TMOD.4) и М1 (TMOD.5) управляют режимами работы Т/С1, а, соответственно, М0 (TMOD.0) и М1 (TMOD.1) режимами Т/С0, как указано в таблице 10.4.

Таблица 10.4 – Режимы работы Т/С

М1	**М0**	**Режимы работы Т/С**
0	0	Младший байт Т/С TL0 и TL1 работает как 5-битный счетчик-делитель. Совместно со старшим байтом образуют 13-разрядный Т/С (отключаются три старших разряда в TL0 и TL1)
0	1	Старший и младший байты Т/С включены последовательно и образуют 16-битный таймер-счетчик
1	0	8-битный автоперезагружаемый таймер-счетчик. Старший байт хранит значение, которое должно быть перезагружено в младший байт каждый раз по его переполнению
1	1	Таймер-счетчик 1 останавливается. Таймер-счетчик 0: TLO работает как 8-битный таймер-счетчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1

Как управлять битами? Конечно, программно командами из системы команд МК51.

Режим второй

Во втором режиме могут работать оба Т/С. Управление ими такое же, как в нулевом и первом режимах. Отличается соединением старшего и младшего байтов счетчиков (рис. 10.11). Рассмотрим на примере Т/С0 (Т/С1 работает так же).

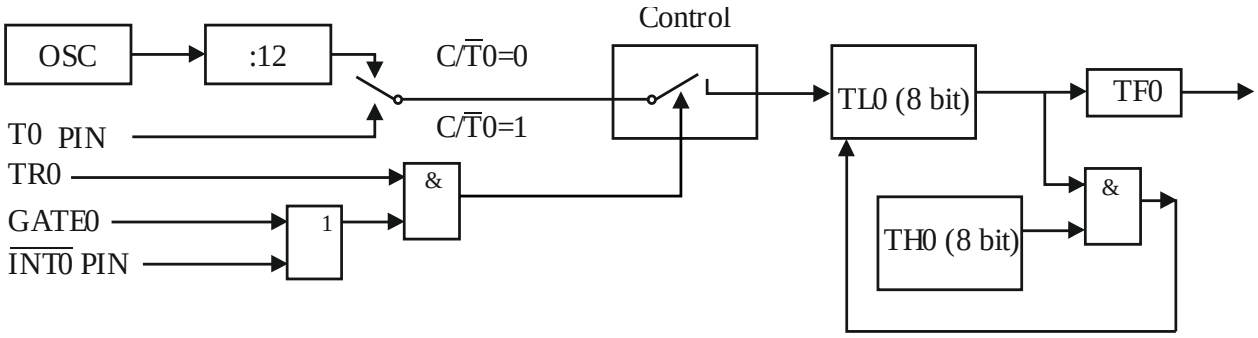


Рис. 10.11 – Второй режим работы Т/С0

Предварительно в старший и младший байты загружается число «data». Таймер-счетчик запускается сигналом TR0 установкой в единицу бита TCON.4 при C/T0 = 0. Т/С0 считает импульсы от внутреннего генератора, с частотой, деленной на 12. Когда в TL0 произойдет переполнение (переход от FFH в 0), то этот сигнал запоминается в триггере TF0 и вызывает передачу содержимого TH0 в TL0, т. е. в TL0 снова заносится то же самое число «data». Таким образом Т/С реализует формирование временного интервала равного «data» · 1 мкс. Так можно программно формировать временные задержки.

Третий режим здесь рассматривать не будем. В последующих модификациях микроконтроллеров семейства 8051 появилось еще несколько счетчиков различного назначения. Может иметься третий таймер-счетчик Т/С2 и (или) блок программных счетчиков PCA, которые могут быть использованы для отсчета временных интервалов.

10.3.5 Канал последовательной передачи данных

Интерфейс канала последовательной передачи данных (последовательный порт – ПП) осуществляет прием и передачу данных от микроконтроллера и к нему последовательным кодом. Связь можно осуществлять с одним или несколькими микроконтроллерами. Объединение нескольких микроконтроллеров можно сделать на основе сети. Сеть микроконтроллеров организуется по архитектуре «Общая шина» (рис. 10.12).

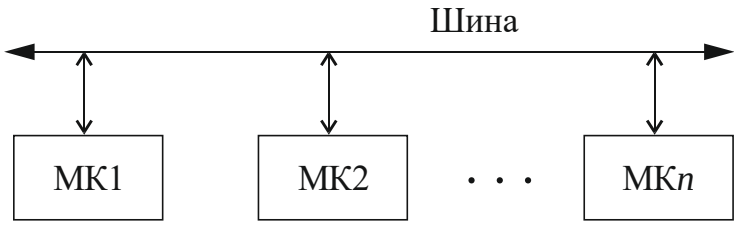


Рис. 10.12 – Структура сети – общая шина

В общей шине один из микроконтроллеров является главным, а остальные подчиненными. Протоколом обмена данными между контроллерами предусмотрен обмен командами от главного к подчиненным и данными между ними. Причем главный микроконтроллер назначает, от какого к какому будет производиться передача данных и в каком объеме.

Структура ПП данных приведена на рисунке 10.13. ПП содержит два сдвиговых регистра. Регистр приемника получает на вход RXD (порт P3.0) побитно данные от другого микроконтроллера в последовательном коде. Сдвиг последовательного кода сопровождается синхроимпульсами. Получив последний бит, приемник формирует бит RI – требование прерывания от последовательного канала. Затем код из регистра приемника пересылается параллельным кодом в буферный регистр приемопередатчика, который в свою очередь может передать его во внутреннюю магистраль данных по командам программы обработки прерывания. При передаче данных из микроконтроллера байт данных передается из внутренней магистрали данных в буферный регистр приемопередатчика и из него параллельно записывается в сдвиговый регистр передатчика. Затем формируются импульсы сдвига, которые выталкивают содержимое сдвигового регистра на выход TXD (P3.1). Когда будут выдвинуты все 8 бит из сдвигового регистра, возникнет требование прерывания от передатчика TI.

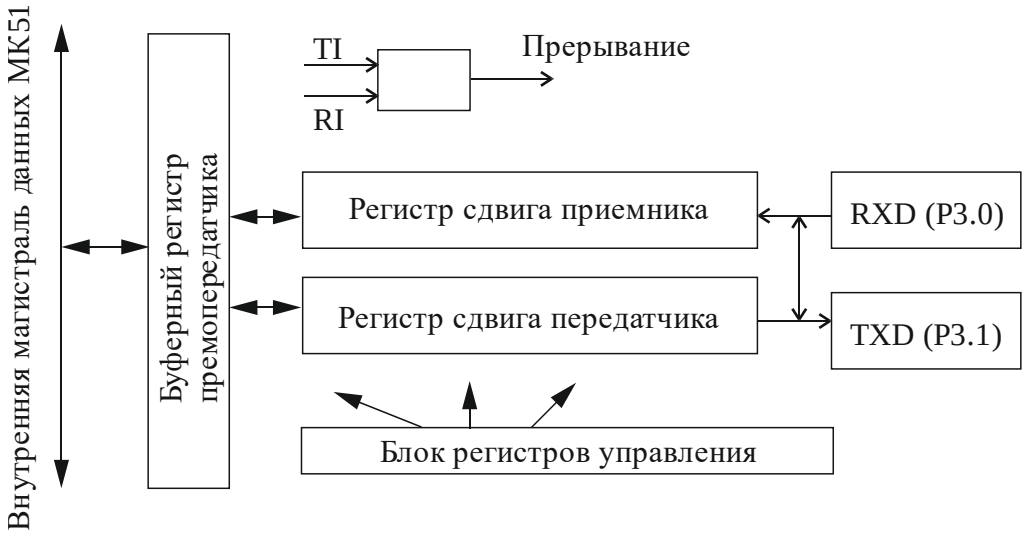


Рис. 10.13 – Структура канала последовательной передачи данных

Блок регистров управления ПП состоит из регистров SCON, PCON.7. В них программно или аппаратно формируются биты, задающие режимы работы ПП. Таких режимов существует четыре.

Режим 0. В этом режиме и на ввод, и на вывод последовательных данных используется RXD, причем как на прием, так и на передачу. По входу RXD происходит прием и передача данных. Сигнал TXD используется для формирования импульсов сдвига с частотой $f_{\text{тактового генератора}}/12$, которые сопровождают биты данных и при приеме, и при передаче.

Режим 1. Каналы приема и передачи работают раздельно. Передаются и принимаются 8 бит данных, старт-бит (нулевой уровень) и стоп-бит (уровень 1). Скорость приема и передачи задается таймером-счетчиком.

Режим 2. Режим такой же, как и предыдущий. Отличие в том, что передаются 11 бит. Старт-бит, 8 бит данных, 9-й бит контроля четности (из PSW.0) и стоп-бит. Частота при приеме-передаче может быть 1/32 или 1/64 тактовой частоты.

Режим 3. Такой же, как и режим 2. Отличие в том, что скорость передачи задается таймером-счетчиком.

Режимы работы задаются регистром SCON (табл. 10.5).

Таблица 10.5 – Регистр управления SCON

<table>
<tr><th>Символ</th><th>Позиция</th><th>Имя и назначение</th></tr>
<tr><td>SM0</td><td>SCON.7</td><td rowspan="2">Биты управления режимом работы приемопередатчика. Устанавливаются/сбрасываются программно
<table>
<tr><th>SM0</th><th>SM1</th><th>Режим работы приемопередатчика</th></tr>
<tr><td>0</td><td>0</td><td>Сдвигающий регистр расширения ввода/вывода</td></tr>
<tr><td>0</td><td>1</td><td>8-битовый приемопередатчик, изменяемая скорость передачи</td></tr>
<tr><td>1</td><td>0</td><td>9-битовый приемопередатчик, фиксированная скорость передачи</td></tr>
<tr><td>1</td><td>1</td><td>9-битовый приемопередатчик, изменяемая скорость передачи</td></tr>
</table></td></tr>
<tr><td>SM1</td><td>SCON.6</td></tr>
<tr><td>SM2</td><td>SCON.5</td><td>Бит управления режимом приемопередатчика. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение 0</td></tr>
<tr><td>REN</td><td>SCON.4</td><td>Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных</td></tr>
<tr><td>TB8</td><td>SCON.3</td><td>Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме 9-битового передатчика</td></tr>
<tr><td>RB8</td><td>SCON.2</td><td>Прием бита 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме 9-битового приемника</td></tr>
<tr><td>TI</td><td>SCON.1</td><td>Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания</td></tr>
<tr><td>RI</td><td>SCON.0</td><td>Флаг прерывания приемника. Устанавливается аппаратно при приеме байта. Сбрасывается программно после обслуживания прерывания</td></tr>
</table>

На скорость передачи дополнительно воздействует бит SMOD из седьмого разряда регистра PCON (PCON.7). Взведение этого бита в единицу удваивает скорость приема и передачи данных. Таймеры-счетчики, используемые при работе ПП, устанавливаются во второй режим работы с перезагрузкой. В таблице 10.6 приведены значения скоростей передачи данных и настроек таймеров-счетчиков.

Таблица 10.6 – Настройка таймера-счетчика T/C1
для разных скоростей работы ПП

Частота приема/передачи	Частота резонатора МГц	Таймер/счетчик 1			
		SMOD	C/T	Режим (MODE)	Перезагружаемое число
Режим 0, макс: 1 МГц	12	X	X	X	X
Режим 2, макс: 375 кГц	12	1	X	X	X
Режим 1, 3: 62,2 кГц	12	1	0	2	0FFH
19,2 кГц	11,059	1	0	2	0FDH
9,6 кГц	11,059	0	0	2	0FDH
4,8 кГц	11,059	0	0	2	0FAH
2,4 кГц	11,059	0	0	2	0F4H

1,2 кГц	11,059	0	0	2	0F4H
137,5 Гц	11,059	0	0	2	1DH
110 Гц	6	0	0	2	72H
110 Гц	12	0	0	1	0FEEBH

Контрольные вопросы по главе 10

1. Если микропроцессор восьмиразрядный, то может ли внутренняя магистраль иметь разрядность 16?
2. Может ли регистр B использоваться в тех командах, которые не связаны с умножением или делением?
3. Чему равен объем РПД МК51?
4. Если к выводам внутреннего генератора МК51 подключить кварцевый резонатор 9 МГц, то сколько импульсов тактовой частоты войдет в машинный цикл?
5. Для чего порты ввода/вывода МК51 могут быть переведены в третье (*Z*) или подобное ему состояние?
6. В каком режиме работают таймеры-счетчики T/C0 и T/C1, в режиме накопления или вычитания?
7. Перечислите режимы работы таймеров-счетчиков.
8. Перечислите режимы работы последовательного канала передачи данных.

11 Система прерываний

11.1 Понятие стека

Стеком *называют специальным способом организованную область памяти данных* (*ОЗУ*). *Стек работает по принципу «последним зашел – первым вышел».*

Наглядный пример стека – коробка, в которую складываются книги одна на другую. Книга, положенная на самый верх, – последняя и только она может быть вынута первой. Это демонстрируется на рисунке 11.1. «Первый» предмет записан в самом начале, затем «второй», «пятый» – последним. Читается сначала «пятый», затем «четвертый» и т. д. «Первый» прочитается последним.

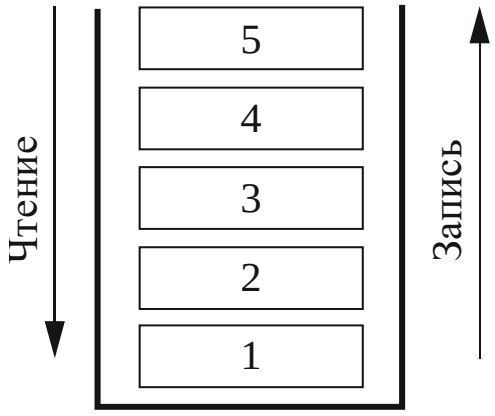


Рис. 11.1 – Организация стека

Стек в микроконтроллерах организуется с использованием аппаратных и программных средств. Стек адресуется с помощью регистра *SP* (*stack pointer* – указатель стека). SP всегда указывает на вершину стека и содержит адрес ячейки ОЗУ, с которой и начинается стек. В МК51 это происходит следующим образом:

1. Предварительно SP должен содержать адрес вершины стека, например 70H.
2. Читается команда «загрузить в стек регистр».
3. SP аппаратно увеличивается на 1 и содержимое регистра записывается в ячейку ОЗУ с адресом 70Р + 1 = 71H.

4. Каждая очередная команда записи в стек выполняется также, таким образом стек растет в сторону увеличения адресов ОЗУ.
5. Если читается команда «прочитать стек», сначала читается содержимое последней ячейки, в которую производилась запись, а потом SP аппаратно уменьшается на 1.

Работать со стеком следует очень аккуратно, чтобы не перемешать данные. Стек практически не используется в программах. Его основное назначение – это работа при вызове подпрограмм и работа в режиме прерываний.

11.2 Общие сведения о прерывании

Рассмотрим, как выполняются программы в МК. МК всегда выполняет какую-либо операцию, т. е. никогда не находится в останове. Счетчик команд (PC) показывает на ячейку памяти программ, в которой находится код очередной операции. Этот код считывается в регистр команд, дешифрируется и передается в УУ МК, который начинает выполнить эту операцию (*текущую*). Сразу после чтения кода текущей команды содержимое PC автоматически увеличивается на единицу и показывает на ячейку, где располагается код *следующей* операции. Таким образом, PC всегда содержит адрес ячейки, в которой располагается *следующая* операция. Главная программа, реализующую алгоритм работы всего устройства, называется *основной программой*.

Во время выполнения основной программы могут возникнуть ситуации (*события*), которые потребуют внимания МК и выполнения какой-то программы, реагирующей на возникшее событие. Событие – это плановая или внеплановая ситуация, возникшая в самом микроконтроллере или в окружающих его ВУ.

Если такие события возникают, то устройство, в котором оно произошло, должно сформировать осведомительный сигнал, который называется *требование прерывания* (ТПР). От внешних устройств он передается на второй или третий разряд порта P3 (*требование прерывания от внешних* устройств INT0 и INT1). Счетчики-таймеры оповещают о переполнении сигналами TF0 и TF1 (*внутреннее требование прерывания от таймеров-счетчиков*). Значит, микроконтроллер должен периодически проверять появление сигналов ТПР и, получив их, перейти к программе, которая обработает это событие. Недостаток такого способа реакции на возникшее событие в том, что МК должен отвлекаться от исполнения основной программы и проверять наличие ТПР. Это не всегда

возможно, особенно при работе управляющим устройством с каким-либо устройством в реальном масштабе времени.

Чтобы обойти это, применяют другой способ. Он заключается в использовании обработки прерываний автоматически с помощью аппаратных и программных средств, которые предусмотрены структурой микроконтроллеров.

Прерывание *– это процесс приостановления выполнения основной программы для выполнения программы, которая не имеет никакого отношения к основной.*

Программа, вызываемая при возникновении того или иного прерывания, называется *программой обработки прерывания* (ПОП). Для каждого из возможных прерываний создается своя ПОП.

Возникновение событий, требующих прерывания, сопровождается сигналом ТПР. Этот сигнал формируется либо внутренними узлами микроконтроллера, либо внешними устройствами. Таким образом прерывания могут быть *внутренними* или *внешними*. Они являются случайными, потому что могут возникнуть в любой момент времени работы основной программы.

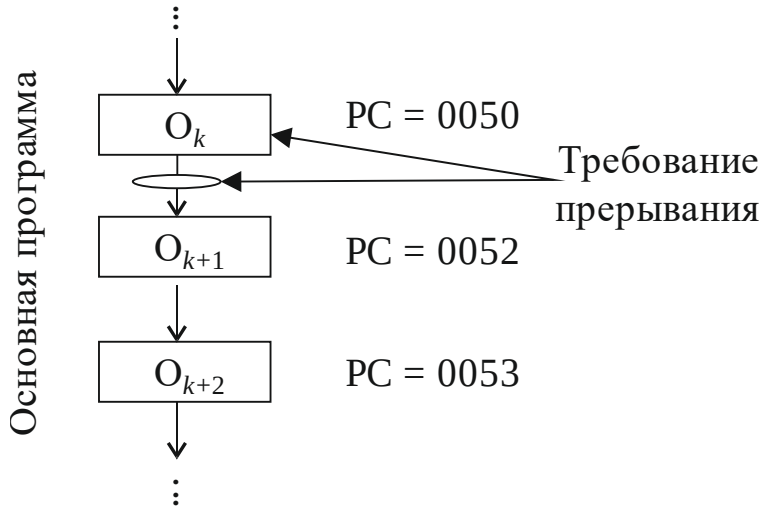


Рис. 11.2 – Появление сигнала ТПР

На рисунке 11.2 показано выполнение основной программой *k*-й операции – O_k, а на ячейку памяти, где хранится код этой команды, показывает счетчик команд PC (в данном случае PC = 0050). Появление сигнала ТПР возможно в любой момент цикла выполнения этой команды.

11.3 Вхождение в режим прерывания

Рассмотрим рисунок 11.3. Считана ячейка памяти 0050, в которой записана операция O_k. РС автоматически увеличивается на 1 и становится равным 0051. Во время исполнения O_k был получен сигнал ТПР. Проверку появления сигнала ТПР (любого) УУ выполняет в состоянии управляющего автомата S5 и фазе P2, т. е. практически в конце выполнения текущей команды.

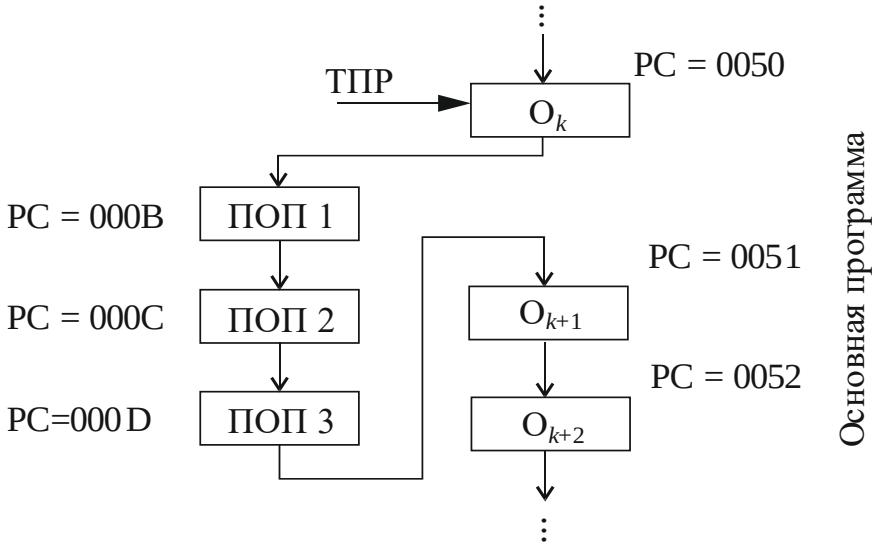


Рис. 11.3 – Процесс прерывания

Чтобы можно было среагировать на прерывание, оно должно быть разрешено. Значит, прерывания могут быть *разрешенными* или *неразрешенными* (*замаскированными*). Если прерывание разрешено, то сначала сохраняется адрес следующей операции основной программы O_{k+1} (0051), а затем в РС заносится адрес, в котором размещена первая команда ПОП (в данном случае это 000B). Адрес O_{k+1} называется *вектором прерванного процесса*. Адрес 000B называется *вектором прерывающего процесса*. Переход к программе обработки прерывания (ПОП) начнется только по завершении операции O_k. Из ячейки с адресом 000B читается первая команда ПОП (ПОП1), затем вторая (ПОП2), затем третья (ПОП3). Третьей (в данном случае) должна быть команда выхода из прерывания, по которой адрес O_{k+1} (0051) возвращается в РС и МК51 продолжит выполнение основной программы. В приведенном примере ПОП содержит всего три операции. Но в зависимости от ситуации это могут быть и более длинные программы.

Рассмотрим более подробно алгоритм вхождения в режим прерывания по рисунку 11.4:

1. Выполнялась операция O_k, считанная из ячейки 0052H, содержимое PC увеличилось на 1 стало равно 0053H (для чтения следующей команды O_{k+1}). Во время выполнения O_k пришел сигнал ТПР. Если прерывание запрещено (замаскировано), то оно игнорируется.
2. В конце машинного цикла O_k (S5P2) проверяется наличие ТПР, и если прерывание разрешено, то содержимое PC (вектор прерванного процесса) автоматически прячется в стек. Двухбайтовые число адреса ячейки памяти записывается так – сначала старший байт PC, затем младший байт PC.
3. ПОП могут содержать любые команды и использовать те же регистры, что и основная программа. Чтобы не потерять значения этих регистров, в том числе и регистра флагов (PSW), их содержимое, но уже программно командой PUSH, необходимо сохранить в стеке.
4. В PC заносится вектор прерывающего процесса (в данном случае 000bH) , т. е. адрес ячейки памяти, в которой находится первая команда ПОП.
5. Выполняется вся ПОП (ПОП1, ПОП2).
6. В конце ПОП следует восстановить содержимое всех регистров, сохраненных в стеке командой POP.
7. Последняя команда ПОП должна быть команда RETI (выход из прерывания), по которой в PC возвращается сохраненное в стеке значение адреса основной программы, т. е. 0053H.
8. Продолжается выполнение основной программы.

В МК51 предусмотрены два прерывания от внешних устройств, инициируемых сигналами INT0 и INT1 (низкими уровнями на входы 2, 3 порта P3), два прерывания по переполнению таймеров-счетчиков TF0 и TF1 (регистр TCON) и прерывание от последовательного канала передачи данных TI/RI (регистр SCON).

В МК51 управление прерываниями осуществляется с помощью регистра масок прерывания IE, регистра приоритетов прерывания IP, регистра TCON и двух младших битов регистра SCON.

Прерывания распределены по приоритету. Это сделано для того, чтобы при одновременном возникновении ТПР от нескольких устройств первым предоставлялось обслуживание прерывания устройству с более высоким приоритетом. По умолчанию в МК51 приоритеты распределены так (по убыванию приоритета):

INT0 → TF0 → INT1 → TF1 → RI/TI,

(внешнее прерывание 0, переполнение T/C0, внешнее прерывание 1, переполнение T/C1, прерывание по приему или передаче байта данных по последовательному каналу).

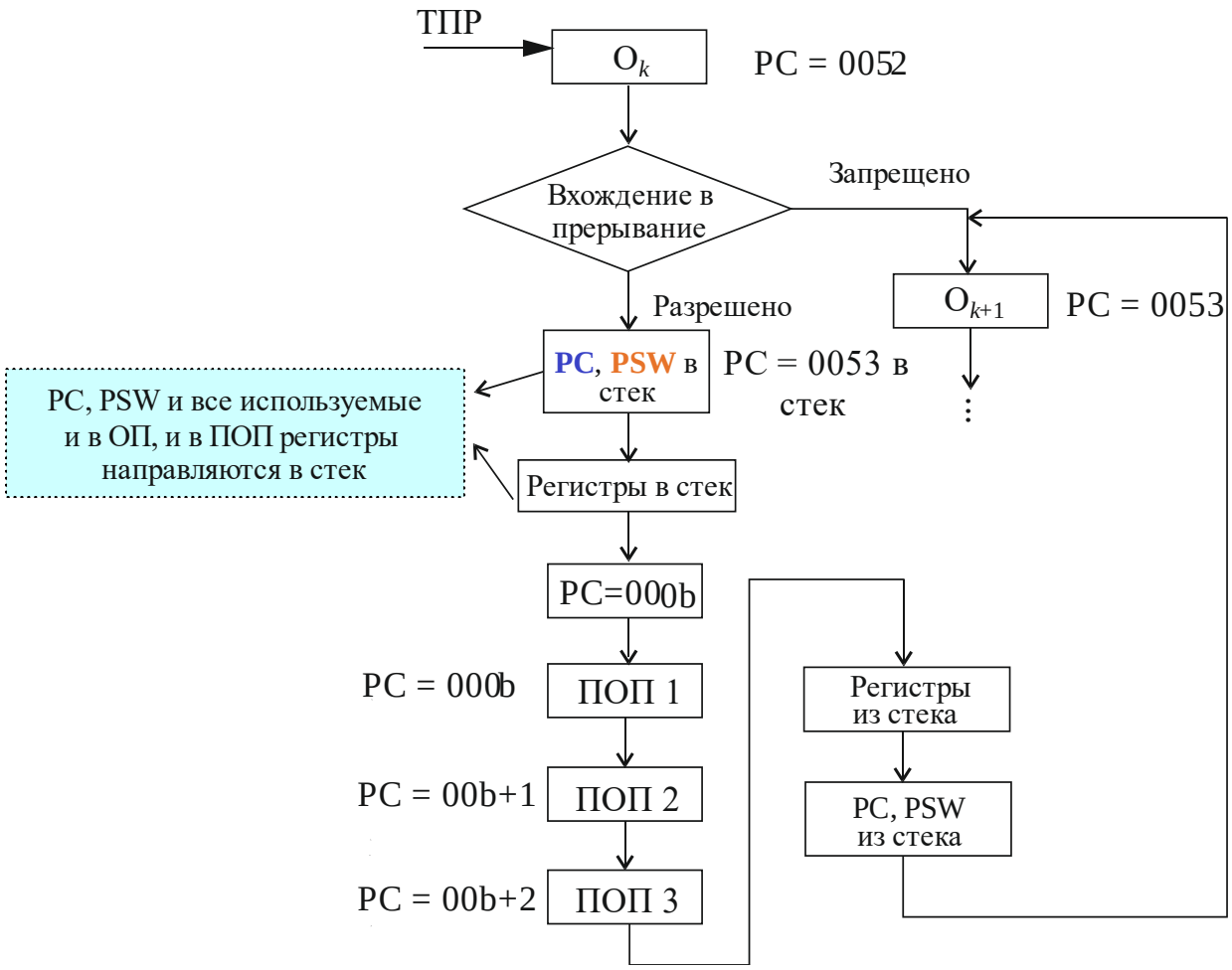


Рис. 11.4 – Вхождение в режим прерывания

Приоритетами можно управлять с помощью регистра IP, все биты которого программно доступны (табл. 11.1).

Таблица 11.1 – Регистр приоритетов прерывания IP

Регистр IP							
7	6	5	4	3	2	1	0
–	–	–	PS	PT1	PX1	PT0	PX0

PX0 – при установке дает высший приоритет внешнему прерыванию 0.

PT0 – при установке дает высший приоритет прерыванию от T/C0.

PX1 – при установке дает высший приоритет внешнему прерыванию 1.

PT1 – при установке дает высший приоритет прерыванию от T/C1.

PS – при установке дает высший приоритет прерыванию от последовательного канала передачи данных.

IP.7, IP.6, IP.5 – не используются.

Для маскирования прерываний введен регистр IE, все биты которого программно доступны (табл. 11.2).

Таблица 11.2 – Регистр масок прерывания IE

Регистр IE							
7	6	5	4	3	2	1	0
EA	–	–	ES	ET1	EX1	ET0	EX0

EA (IE.7) – блокировка прерываний. При установке его в 0 все прерывания маскируются (блокируются, запрещаются), т. е. чтобы разрешить прерывания, надо в этот бит записать 1.

IE.6, IE.5 – не используются.

ES – маскируются прерывания при работе последовательного канала передачи данных.

ET1 – маскируется прерывание от T/C1.

EX1 – маскируется внешнее прерывание 1 (INT1).

ET0 – маскируется прерывание от T/C0.

EX0 – маскируется прерывание 0 (INT0).

Часть регистра TCON, который был рассмотрен при описании работы таймеров-счетчиков, используется и в режиме прерываний (табл. 11.3).

Таблица 11.3 – Регистр управления/статуса таймеров TCON

Регистр TCON							
7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0 – если установить бит в 1, то ТПР по сигналу INT0 будет восприниматься по его перепаду (фронту) из 1 в 0. Если бит сбросить в 0, то сигнал INT0 воспринимается МК51 по низкому уровню. В этом случае в программе обработки прерывания необходимо предусмотреть воздействие на устройство, выставившее этот сигнал, чтобы оно вернуло INT0 в высокий уровень.

IE0 – флаг устанавливается фронтом сигнала INT0, поступившего от внешнего устройства. Этот бит автоматически сбрасывается, как только МК51 переходит к программе обработки прерывания.

IT1 – так же как IT0, только для сигнала INT1.

IE1 – так же как IE0, только для сигнала INT1.

TR0 – пуск T/C0.

TF0 – флаг возникновения переполнения T/C0 (ТПР от T/C0).

TR1 – пуск T/C1.

TF1 – флаг возникновения переполнения T/C1 (ТПР от T/C1).

Прерывания от узлов канала последовательной передачи данных участвуют в системе прерываний двумя битами регистра SCON (рис. 11.4).

Таблица 11.4 – Регистр управления/статуса таймеров SCON

Регистр SCON							
7	6	5	4	3	2	1	0
						TI	RI

TI – флаг устанавливается при передаче данных последовательным кодом по окончании передачи последнего бита. Сбрасывается программно в программе обработки прерывания.

RI – флаг устанавливается при приеме данных последовательным кодом по окончании приема последнего бита. Сбрасывается программно в программе обработки прерывания.

За каждым из пяти прерываний в МК51 жестко закреплен вектор прерываний – адрес ячейки памяти, в которой хранится код первой команды ПОП. В таблице 11.5 показаны адреса всех векторов прерывания МК51.

Таблица 11.5 – Распределение адресов векторов прерывания

Флаг прерывания	**Адрес вектора прерывания**
IE0 (INT0)	0003H
TF0 от T/C0	000BH
IE1 (INT1)	0013H
TF1 от T/C1	001BH
TI/RI	0023H

Как видно из таблицы 11.5, на каждую ПОП выделено по восемь ячеек памяти. Если этого достаточно, то соответствующая ПОП размещается по адресу своего вектора прерывания (табл. 11.6). Если для ПОП места недостаточно, то командой перехода выполняется переход по другому адресу, где располагается первая команда ПОП.

Таблица 11.6 – Распределение адресов памяти программ

Адреса ячеек памяти (16-ричный код)	**Ячейки памяти (1 байт)**	
0000		Старт программы
0001		
0002		
0003		ПОП по INT0 (8 ячеек)
000B		ПОП по T/C0 (8 ячеек)
0013		ПОП по INT1 (8 ячеек)
001B		ПОП по T/C1 (8 ячеек)
0023		ПОП последовательного канала (8 ячеек)
002B		Остальная память
002C		
002D		
…		

Первые три ячейки РПП с адресами 0000, 0001 и 0002 в программах используются для размещения стартовой команды, пересылающей к ячейке РПП, в которой размещается первая команда программы.

11.4 Инициализация МК51

При включении блока питания устройства начинается инициализация МК51, чтобы начать выполнение программы, загруженной в РПП. Для ее начала блок питания должен сформировать единичный сигнал сброса RST. По этому сигналу устанавливаются в нулевое состояние счетчик команд (PC), все регистры функций, в том числе PSW. В указатель стека записывается код семерки (SP = 7). Значения ячеек РПД могут быть неопределенными. По окончании сигнала RST УУ МК51 считывает содержимое ячейки РПП с адресом 00. В ячейках с адресами 0, 1 и 2 может быть размещена команда безусловного перехода (LJMP, AJMP или SJMP) к ячейке, где размещена первая команда основной программы. Если в программе используются все прерывания, то ближайшая ячейка с адресом первой команды основной программы – 30h. После чтения и выполнения первой команды читается вторая и т. д. Пока питание подается на устройство, МК51 все время работает в программном режиме.

Контрольные вопросы по главе 11

1. Что происходит с содержимым регистров при инициализации?
2. Инициализация выполняется программно или аппаратно?
3. Чем формируется сигнал сброса RST?
4. Сколько ячеек памяти программ выделено для обращения к основной программе, выполняемой микроконтроллером?

Заключение

В данном пособии изложены вопросы, связанные с работой цифровых устройств. Рассмотрена двоичная система счисления и операция сложения двоичных чисел. Приведены основные положения булевой алгебры применительно к синтезу и анализу комбинационных схем и их реализации в различных базисах. Перечислены основные узлы цифровых устройств и приведены законы их работы. Приведены примеры использования узлов для проектирования комбинационных схем. Рассмотрена работа схем триггеров и узлов на их основе. Показано, как на основе двоичных счетчиков можно строить схемы счетчиков-делителей. Изложена методика построения автоматов для формирования заданной последовательности выходных слов – генераторов слов. В качестве примера архитектуры микропроцессоров приведена структура микроконтроллера МК-51, описана работа его основных узлов.

Учебное пособие может быть использовано как поддержка при изучении курса «Цифровые устройства и микропроцессоры». Пособие не претендует на всеобъемлющее рассмотрение вопросов, связанных с работой цифровых устройств, и предполагает дальнейшее самостоятельное совершенствование читателей в искусстве схемотехники.

Литература

1. Потехин, В. А. Схемотехника цифровых устройств : учеб. пособие для вузов / В. А. Потехин. – Томск : Изд-во Томск. гос. ун-та систем упр. и радиоэлектроники, 2015. – 501 с.
2. Бунтов, В. Д. Микропроцессорные системы. Ч. 1. Цифровые устройства [Электронный ресурс] : учеб. пособие / В. Д. Бунтов, С. Б. Макаров. – СПб. : Изд-во политехн. ун-та, 2008. – 199 с. – URL: http://window.edu.ru/resource/372/77372/files/MBS2.pdf (дата обращения: 21.02.2022).
3. Брякин, Л. А. Основы схемотехники цифровых устройств [Электронный ресурс] : курс лекций / Л. А. Брякин. – Пенза : Пенз. гос. ун-т, 2005. – URL: http://window.edu.ru/resource/875/36875/files/stup101.pdf (дата обращения: 21.02.2022).
4. Легостаев, Н. С. Микроэлектроника : учеб. пособие / Н. С. Легостаев, К. В. Четвергов. – Томск : ФДО, ТУСУР, 2012. – 252 с.
5. Сташин, В. В. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева. – М. : Энергоатомиздат, 1990. – 224 с.
6. Горюнов, А. Г. Архитектура микроконтроллера Intel 8051 [Электронный ресурс] : учеб. пособие / А. Г. Горюнов, С. Н. Ливенцов. – Томск : Изд-во ТПУ, 2005. – 86 с. – URL: https://portal.tpu.ru/SHARED/Others/_JU_/Teaching/Tab2/MCS51.pdf (дата обращения: 21.02.2022).
7. Сташин В.В. Проектирование цифровых устройств на однокристальных микроконтроллерах : / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева. - М. : Энергоатомиздат, 1990. - 224 с.
8. Трофименко В.Н. Микропроцессорные информационно-управляющие системы связи :Учебное пособие/ В.Н. Трофименко. – Ростов-на-Дону: РГУПС 2019.-120с. – Текст электронный //Лань : электронно-библиотечная система. https;//e.lanbook/com/book/134040

Глоссарий

Автомат – электронный узел, автоматически выполняющий заданный ему алгоритм. Автомат задается множествами входных и выходных сигналов, множеством состояний и функциями переходов и выходов, которые и описывают алгоритм его работы.

Абстрактный автомат – математическая модель автоматического устройства.

Аналоговый сигнал – электрический сигнал, непрерывный в течение промежутка времени.

Анализ комбинационной схемы – процедура, позволяющая выяснить закон функционирования комбинационной схемы.

Арифметико-логическое устройство (*АЛУ*) – электронный узел, выполняющий некоторое количество операций. В них могут входить операции сложения, вычитания, сравнения, сдвига, инвертирования и т. д. Выпускается в виде микросхемы. Входит в состав почти всех микропроцессоров и микроконтроллеров.

Архитектура – набор принципов построения внутренней структуры и интерфейсов взаимодействия с внешними устройствами.

Архитектура фон Неймана – пять принципов, которые были положены в основу построения первых ЭВМ. В настоящее время модифицирована.

Базис – полная система функций алгебры логики, с помощью которых любую БФ можно представить как суперпозицию функций базиса.

Булева алгебра – алгебра, разработанная Джорджем Булем, одним из применений которой является описание закона работы цифрового устройства.

Булева переменная – переменная, принимающая одно из двух значений – «истина» или «ложь», – как правило, обозначаемых как единица или ноль.

Булева функция (*БФ*) – функция, принимающая одно из двух значений «истина» или «ложь» (1 или 0) на любом наборе булевых переменных. Ее также называют переключательной функцией или функцией алгебры логики.

Гарвардская архитектура – предполагает внутреннее разделение памяти на память программ и память данных.

Генератор слов – автомат, как правило, построенный по принципу счетчика, но формирующий на своих выходах заданную последовательность двоичных кодов. Частным случаем одного из генератора слов является счетчик.

Двоичный сумматор – электронный узел, выполняющий операцию арифметического сложения двоичных чисел.

Дешифратор – электронный узел, позволяющий однозначно расшифровать двоичный код, поступивший на его вход.

Дизъюнктивная форма БФ (*ДНФ*) – представление БФ в виде дизъюнкций конъюнкций логических переменных.

Дискретный сигнал – прерывистый электрический сигнал, имеющий значение, взятое из списка разрешенных значений, в течение заданного промежутка времени.

Дополнительный двоичный код – двоичный код, дополняющий прямой код до единицы.

Карта Карно, карта Вейча – карты, применяемые для минимизации булевых функций путем нахождения минимальных покрытий. Количество клеток карт равно количеству наборов переменных булевой функции. В клетке карты записывается значение булевой функции, которое она принимает на этом наборе.

Комбинационная схема (*КС*) – схема, закон работы которой описывается булевой функцией. Комбинационные схемы реализуются с помощью электронных логических элементов или в других устройствах, например, контактных группах.

Конъюнктивная форма БФ (*КНФ*) – конъюнкция дизъюнкций логических переменных.

Машинный такт – период колебаний тактового генератора.

Машинный цикл (*МЦ*) – период, состоящий из определенного количества периодов тактового генератора МП или МК. Машинный цикл формируется устройством управления, которое в каждом цикле создает множество синхронизирующих импульсов, инициирующих выполнение текущей команды.

Минимизация булевых функций – процесс нахождения минимальной формы булевой функции. Применяется при минимизации СДНФ и СКНФ с использованием закона склеивания.

Мультиплексор – электронный узел, позволяющий передать один из множества входных сигналов на его выход.

Микропроцессор (*МП*) – программно управляемое электронное устройство для обработки цифровой информации и управления процессом этой обработки, выполненное с высокой степенью плотности электронных компонентов. Как правило, имеет в своем составе ограниченное число регистров, вся его архитектура направлена на увеличение вычислительной мощности и быстродействия. Имеет развитую систему команд. Используется в качестве центрального процессора персональных компьютеров для решения широкого круга задач. Данные и программы для его обработки располагаются во внешних устройствах.

Микроконтроллер (*МК*) – микропроцессор (*micro controller*), содержащий большое количество внутренних регистров (память данных) и ПЗУ (память программ), развитую систему ввода-вывода, такую как последовательные и параллельные каналы передачи данных, и дополнительное оборудование. Система команд, по сравнению с МП, менее развита. Память данных и программ их обработки располагаются в самом микроконтроллере, поэтому МК часто называют микро-ЭВМ. В основном МК используются как устройства управления различным оборудованием.

Набор булевых переменных – перечисление всех переменных, от которых зависит булева функция, и значение, которое имеет каждая из переменных в данный момент времени.

Неполностью определенная булева функция – такая булева функция, значения которой на некоторых наборах переменных неопределенно, т. е. безразлично, будет ли она равна единице или нулю.

Оперативное запоминающее устройство (*ОЗУ*) – электронный узел, состоящий из множества ячеек, предназначенных для сохранения записанного в них двоичного кода. Каждая ячейка имеет свой номер, называемый адресом. Ячейки могу быть одноразрядными или многоразрядным. При отключении питания данные, записанные в ячейках, пропадают.

Полностью определенная булева функция – такая булева функция, которая имеет одно из двух значений (0 или 1) на любом из наборов переменных.

Порт ввода-вывода – узел в составе МК для организации приема и передачи данных между МК и внешними устройствами. По одним и тем же цепям данные могут перемещаться в обе стороны. Порты могут быть предназначены для параллельной или последовательной передачи данных. Разные МК содержат различное количество портов.

Прямой двоичный код – двоичный код числа, к которому добавлен знаковый разряд.

Преобразователь кодов – электронный узел, выполняющий операцию перекодирования кода числа на его входе в другую кодовую комбинацию на выходе.

Прерывание – процесс приостановки выполнения программы для того, чтобы выполнить какую-то другую программу. Прерывания бывают от внутренних и от внешних устройств (соответственно, внутреннее или внешнее прерывание).

Последовательный порт ввода-вывода – порт, преобразующий байт данных в последовательный код при выводе и последовательный код в байт данных при вводе. Организует обмен данными между МК и внешними устройствами последовательным кодом. С его помощью удобно создавать мультипроцессорные структуры.

Программируемая логическая матрица (*ПЛМ*) – электронный узел, содержащий матрицы элементов И и ИЛИ и предназначенный для реализации системы БФ.

Постоянное запоминающее устройство (*ПЗУ*) – электронный узел, состоящий из множества ячеек, предназначенных для сохранения записанного в них двоичного кода. В отличие от ОЗУ данные в ПЗУ изменяются относительно редко и при пропадании питания не стираются. В микроконтроллерах ПЗУ используются для хранения исполняемых программ.

Регистр – электронный узел, состоящий из множества триггеров, объединенных общими цепями управления. Основное назначение регистров – хранение данных. В то же время регистры могут выполнять и другие операции, например сдвиг данных.

Регистр специальных функций – регистр, каждый бит которого связан с тем или иным узлом МК для их подключения или отключения.

Регистр флагов – регистр, отдельные биты которого хранят качественный результат выполнения команд. Например, возникновение переноса или переполнения разрядной сетки, равенство нулю результата операции и т. п. Этот регистр также называют регистром признаков, регистром слова состояния программы или словом состояния процесса.

Система счисления – способ представления и записи чисел.

Структурный автомат – техническая реализация абстрактного автомата.

Схема сравнения – электронный узел, выполняющий операцию отношения (равенства, неравенства) двух двоичных кодов.

Счетчик (*автомат*) – электронный узел, построенный на триггерах для выполнения операции счета. Счетчики считают в двоичном коде импульсы, поступившие на их счетных вход. Счетчики бывают накапливающими (суммирующими), вычитающими и реверсивными.

Таблица истинности – таблица, в которой перечислены все наборы переменных и значения функции на них.

Таймеры-счетчики (*Т/С*) – в МК накапливающие счетчики, которые используются для подсчета импульсов внутреннего генератора или импульсов, сформированных внешними устройствами. Могут использоваться как формирователи временных задержек. Переполнение таймеров-счетчиков вызывает прерывание в выполнении программ.

Триггер – электронный узел, имеющий два устойчивых состояния, в каждом из которых может находиться бесконечно долго. Триггер является ячейкой, хранящей один бит данных. Триггер имеет два выхода – прямой и инверсный, значения которых всегда противоположны. Триггеры имеют входы управления, сигналами которых переводятся в то или иной состояние.

Узел цифрового устройства – для комбинационных схем представляет собой функционально законченную схем, реализующую одну или несколько булевых функций. При включении в свой состав элементарных автоматов реализует узлы с памятью.

Цифровой сигнал – электрический дискретный сигнал, имеющий только два уровня значений.

Шина – электронный узел, применяемый для построения каналов передачи данных между цифровыми узлами.

Электронный логический элемент – электронное устройство, выполняющее одну из элементарных булевых функций над электрическими логическими сигналами. Технически такие элементы реализуются в виде электронных микросхем, содержащих от одного до нескольких логических элементов. Минимизация булевых функций приводит к экономному использованию логических элементов при реализации комбинационной схемы.

Эффект гонок – результат того, что логические элементы, на которых строится автомат, не являются идеальными и имеют свойство задержки распространения сигнала от входа к выходу, причем каждый элемент имеет свое значение этого параметра. Результатом этого может оказаться сбой автомата.