ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Б1.В.05 Теория конечных автоматов

Направление подготовки (специальность)
09.04.01 Информатика и вычислительная техника

Профиль образовательной программы
"Автоматизированные системы обработки информации и управления"

Форма обучения очная

СОДЕРЖАНИЕ

1 Тематическое содержание дисциплины

1.1 Тема 1: «Представление числовой информации в цифровых автоматах» (22 часа).

Перечень и краткое содержание рассматриваемых вопросов.

1. Лекция Представление числовой информации в цифровых автоматах.

1.1. Общие понятия о системах счисления.

Системой счисления называется совокупность правил записи чисел.

Системы счисления подразделяются на позиционные и непозиционные.

Как в позиционных, так и в непозиционных системах счисления используется определенный набор символов - цифр, последовательное сочетание которых образует число.

Непозиционные системы счисления появились раньше позиционных. Они характеризуются тем, что в них символы, обозначающие то или иное число (то есть цифры), не меняют своего значения в зависимости от местоположения в записи этого числа.

Классическим примером такой системы счисления является римская. В ней для записи чисел используются буквы латинского алфавита. При этом буква I означает единицу, V - пять, X - десять, L - пятьдесят, C - сто, D - пятьсот, M - тысячу.

Для получения количественного эквивалента числа в римской системе счисления необходимо просто просуммировать количественные эквиваленты входящих в него цифр. Исключение из этого правила составляет случай, когда младшая цифра находится перед старшей, - в этом случае нужно не складывать, а вычитать число вхождений этой младшей цифры.

Пример:

$DLXXII = 500 + 50 + 10 + 10 + 5 + 1 + 1 = 577.$

$CDXXIX = 500 - 100 + 10 + 10 - 1 + 10 = 429.$

В позиционной системе счисления количество символов в наборе равно основанию системы счисления.

Место каждой цифры в числе называется позицией.

Номер позиции символа (за вычетом единицы) в числе называется разрядом. Разряд называется младшим разрядом. Каждой цифре соответствует определенный количественный эквивалент.

Введем обозначение - запись будет означать количественный эквивалент числа, состоящего из цифр (где) в системе счисления с основанием.

Это число можно представить в виде последовательности цифр.

При этом, конечно, всегда выполняется неравенство.

В общем случае количественный эквивалент некоторого положительного числа в позиционной системе счисления можно представить выражением:

$$A_{(p)} = a_{n-1}a_{n-2}...a_1a_0 \ , \qquad (1.1)$$

где - основание системы счисления (некоторое целое положительное число);

- цифра данной системы счисления, n - номер старшего разряда числа.

Для получения количественного эквивалента числа в некоторой позиционной системе счисления необходимо сложить произведения количественных значений цифр на степени основания, показатели которых равны номерам разрядов (обратите внимание на то, что нумерация разрядов начинается с нуля).

1.2. Системы счисления, используемые в ЦВМ.

Двоичная система счисления.

Набор цифр для двоичной системы счисления – , основание степени .

Количественный эквивалент некоторого целого-значного двоичного числа вычисляется согласно формуле (1.1):

$$A_{(2)} = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + ... + a_1 \cdot 2^1 + a_0 \cdot 2^0 \quad . \qquad (2.1)$$

Наличие этой системы счисления обусловлено тем, что компьютер построен на логических схемах, имеющих в своем элементарном виде только два состояния включено и выключено. Производить счет в двоичной системе просто для компьютера, но сложно для человека.

Например, рассмотрим двоичное число 10100111.

Вычислим десятичный эквивалент этого двоичного числа. Согласно формуле (2), это будет величина, равная следующей сумме:

$$1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \quad .$$

Посчитайте сами, сколько получится.

Сложение и вычитание двоичных чисел (рис.2.1) выполняется так же, как и в других позиционных системах счисления, например, десятичной. Точно так же выполняется заем (перенос) единицы из старшего разряда (в старший разряд).

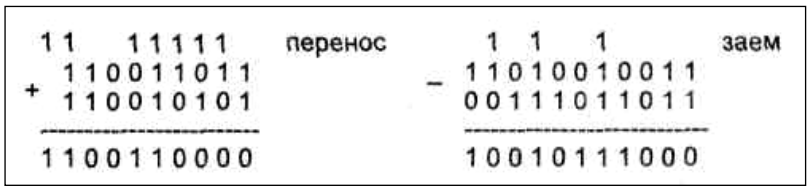


Рис.2.1

Таблица двоек имеет следующий вид (таблица 2.1)

Таблица 2.1 – Талица «двоек»

Степень	Два в указанной степени
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

Шестнадцатеричная система счисления.

Шестнадцатеричная система счисления имеет набор цифр $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ и основание степени $(p) = 16$.

Количественный эквивалент некоторого целого n-значного шестнадцатеричного числа вычисляется согласно формуле (***1***):

$$A_{(16)} = a_{n-1} \cdot 16^{n-1} + a_{n-2} \cdot 16^{n-2} + ... + a_1 \cdot 16^1 + a_0 \cdot 16^0 .$$

К примеру, количественный эквивалент шестнадцатеричного числа $F45ED23C$ равен

$$15 \cdot 16^7 + 4 \cdot 16^6 + 5 \cdot 16^5 + 14 \cdot 16^4 + 13 \cdot 16^3 + 2 \cdot 16^2 + 3 \cdot 16^1 + 12 \cdot 16^0 .$$

Посчитайте сами, сколько получится.

Приведем соответствие двоичных чисел и их десятичных и шестнадцатеричных

эквивалентов (таблица 2.2).

Таблица 2.2 – Представление чисел в различных системах счисления

Десятичное число	Двоичная тетрада	Шестнадцатеричное число
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A, a
11	1011	B, b
12	1100	C, c
13	1101	D, d
14	1110	E, e
15	1111	F, f
16	10000	10

Поначалу запомнить эти соотношения сложно, поэтому полезно иметь под руками некоторую справочную информацию.

Таблица 2.2 содержит представления десятичных чисел из диапазона ***0-16*** в двоичной и шестнадцатеричной системах счисления. Ее удобно использовать для взаимного преобразования чисел в рассматриваемых нами трех системах счисления. Шестнадцатеричная система счисления при вычислениях несколько сложнее, чем двоичная, в частности, в том, что касается правил переносов в старшие разряды (заемов из старших разрядов).

Главное здесь – помнить следующее равенство:

$$(1+F=10)_{16}.$$

Эти переходы очень важны при выполнении сложения и вычитания шестнадцатеричных чисел (рисунок 2.2).

```
 1 1          перенос          1 1          заем
   E F 1 5  1 слагаемое      - B C D 8  уменьшаемое
 + C 1 E 8  2 слагаемое        5 E F 4  вычитаемое
 -----------                  ---------
 1 B 0 F D  результат          5 D E 4  результат
```

Рисунок 2.2 – Операции в шестнадцатеричной системе

Десятичная система счисления.

Десятичная система счисления наиболее известна, так как она постоянно используется нами в повседневной жизни. Данная система счисления имеет набор цифр $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ и основание степени $(p)=10$.

Количественный эквивалент некоторого целого *n*-значного десятичного числа вычисляется согласно формуле (1):

$$A_{(10)} = a_{n-1}\cdot 10^{n-1} + a_{n-2}\cdot 10^{n-2} + \ldots + a_1\cdot 10^1 + a_0\cdot 10^0$$

К примеру, значение числа $A_{(10)} = 4523$ равно:

$$4\cdot 10^3 + 5\cdot 10^2 + 2\cdot 10^1 + 3\cdot 10^0.$$

1.3. Перевод чисел из одной системы счисления в другую.

Одного знания о существовании разных систем счисления мало. Для того чтобы в полной мере использовать их в своей практической работе при программировании на ***ассемблере***, необходимо научиться выполнять взаимное преобразование чисел между тремя системами счисления. Кроме того, дополнительно будут рассмотрены некоторые особенности процессоров ***Intel*** при работе с числами со знаком.

Перевод в десятичную систему счисления.

Перевод в десятичную систему счисления является самым простым. Обычно его производят с помощью так называемого алгоритма замещения, суть которого заключается в следующем: сначала в десятичную систему счисления переводится основание степени p, а затем - цифры исходного числа. Полученная сумма и будет искомым результатом. Неявно при обсуждении двоичной и шестнадцатеричной систем счисления мы производили как раз такое преобразование.

Перевод в двоичную систему счисления.

Перевод из десятичной системы счисления.

Перевод числа в двоичную систему счисления из десятичной выполняется по описанному далее алгоритму.

1. Разделить десятичное число A на 2. Запомнить частное q и остаток a.

2. Если в результате шага 1 частное q не равно 0, то принять его за новое делимое и отметить остаток a, который будет очередной значащей цифрой числа, и вернуться к шагу 1, на котором в качестве делимого (десятичного числа) участвует полученное на шаге 2 частное.

3. Если в результате шага 1 частное q равно 0, алгоритм прекращается. Выписать остатки в порядке, обратном их получению. Получится двоичный эквивалент исходного числа.

К примеру, перевод в двоичную систему счисления числа 247_{10} иллюстрирует ***рис.3.1***. Порядок обхода остатков для получения результата (11110111_2) показан стрелками.

Перевод из шестнадцатеричной системы счисления

Перевод из шестнадцатеричной системы счисления мы уже обсуждали ранее. Суть его заключается в последовательной замене шестнадцатеричных цифр соответствующими двоичными ***тетрадами***, согласно ***табл.2.2***.

К примеру, двоичное число, соответствующее числу $e4d5_{16}$, равно 1110010011010101_2

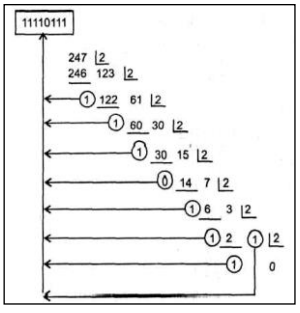


Рис.3.1

Перевод в шестнадцатеричную систему счисления.

Перевод из десятичной системы счисления.

Общая идея алгоритма перевода из десятичной системы счисления в шестнадцатеричную аналогична рассмотренной ранее в алгоритме перевода в двоичную систему счисления из десятичной.

1. Разделить десятичное число A на 16. Запомнить частное q и остаток a.

2. Если в результате шага 1 частное q не равно 0, то принять его за новое делимое, записать остаток и вернуться к шагу 1.

3. Если частное q равно 0, прекратить работу алгоритма. Выписать остатки в порядке, обратном их получению. Получится шестнадцатеричный эквивалент исходного десятичного числа.

К примеру, перевод в шестнадцатеричную систему счисления числа 32767_{10} иллюстрирует ***рис.3.2***. Порядок обхода остатков для получения результата ($7fff_{16}$) показан стрелками.

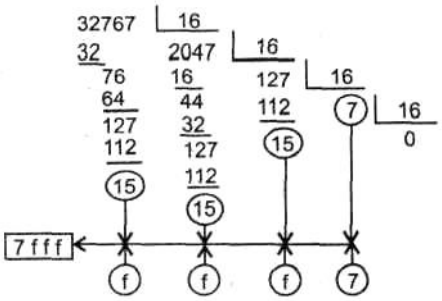


Рис.3.2

Перевод из двоичной системы счисления.

Идея алгоритма состоит в том, что двоичное число разбивается на тетрады, начиная с младшего разряда. Далее каждая тетрада приводится к соответствующему шестнадцатеричному числу, согласно ***Табл. 2.2***.

К примеру, пусть требуется перевести в шестнадцатеричную систему счисления следующее число:

$1110010110101111010110001101100011111010101011012$.

Разобьем его на тетрады:

0111 0010 1101 0111 1010 1100 0110 1100 0111 1010 1010 1101.

По тетрадам приводим последовательности нулей и единиц к шестнадцатеричному представлению:

72d7ac6c7aad.

То есть в результате преобразования мы получим шестнадцатеричное число

72d7ac6c7aad,G.

Перевод дробных чисел.

Программист должен уметь выполнять перевод в различные системы счисления не только целых чисел, но и дробных чисел. Особенно важно это при программировании алгоритмов, использующих операции с плавающей точкой.

Давайте разберемся с наиболее часто используемыми на практике способами перевода дробных чисел. Для этого формулу (4.1) преобразуем к следующему виду (4.3).

Рассмотрим операции перевода чисел на примерах.

Пример 1

Пусть требуется перевести в десятичное представление следующее дробное число, заданное в двоичной системе счисления:

$110100{,}010010112$.

Для перевода используем формулу (4.3):

Не составляет труда вычислить целую часть десятичной дроби:

Для вычисления остальной части выражения удобно использовать ***табл. 4.3***.

Таблица 4.3

Отрицательная степень	Два в указанной степени
1	0,5
2	0,25
3	0,125
4	0,0625
5	0,03125
6	0,015625
7	0,0078125

Вы можете сами продолжить ***табл. 4.3*** значениями отрицательных степеней по основанию числа 2 и в конце концов подсчитать результат перевода в десятичное представление значения

110100,01001011,.

Пример 2

Пусть требуется перевести в десятичное представление следующую дробь, заданную в шестнадцатеричной системе счисления:

Idf2,ale4|6.

Вновь используем формулу (4.3):

Для удобства вычисления дробной части приведем значения отрицательных степеней числа 16 (***табл. 4.4***).

Таблица 4.4

Отрицательная степень	Шестнадцать в указанной степени
1	0,0625
2	0,00390625
3	0,000244140625
4	0,0000152587890625
5	0,0000095367431640625
6	0,000000596046447753906 25
7	0,000000037252902984619140625

Перевод из двоичной системы счисления в шестнадцатеричную и обратно выполняется, как обычно, на основе тетрад и трудности вызывать не должен.

Пример 3

Рассмотрим теперь проблему представления десятичных дробей в двоичной и шестнадцатеричной системах счисления.

Общий алгоритм перевода десятичной дроби в другую систему счисления можно представить следующей последовательностью шагов:

1. Выделить целую часть десятичной дроби и выполнить ее перевод в выбранную систему счисления по алгоритмам, рассмотренным ранее.

2. Выделить дробную часть и умножить ее на основание выбранной новой системы счисления.

3. В полученной после умножения дробной части десятичной дроби выделить целую часть и принять ее в качестве значения первого после запятой разряда числа в новой системе счисления.

4. Если дробная часть значения, полученного после умножения, равна нулю, то прекратить процесс перевода. Процесс перевода можно прекратить также в случае, если достигнута необходимая точность вычисления. В противном случае вернуться к шагу 3.

Пример.

Пусть требуется перевести в двоичную систему счисления десятичную дробь 108,40610.

Сначала переведем целую часть десятичной дроби 108,406]0 в двоичную систему счисления (***рис.4.5***).

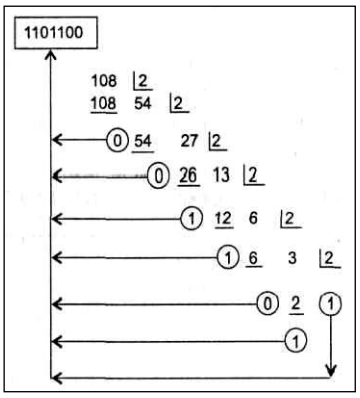


Рис.4.5

Затем переведем дробную часть десятичного числа 108,4061 (***рис.4.6***) по приведенному ранее алгоритму.

Результат перевода следующий:

108,40610= 1101100,011001111.

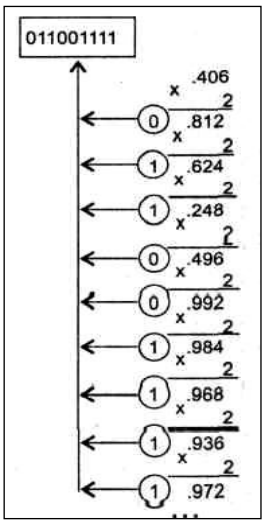


Рис.4.6

При переводе дробного числа из десятичной системы счисления в шестнадцатеричную целесообразно предварительно перевести число в двоичную систему, а затем двоичное представление разбить на тетрады отдельно до разделительной запятой и после запятой.

Разбиение на тетрады двоичных разрядов целой части производят от запятой в сторону старших разрядов. Неполную старшую тетраду дополняют слева нулями.

Разряды дробной части, напротив, разбивают на тетрады от запятой вправо к младшим разрядам. Если последняя тетрада неполная, то ее дополняют нулями справа.

На ***рис.4.7*** показан процесс перевода того же дробного десятичного числа (108,40610) в эквивалентное шестнадцатеричное представление.

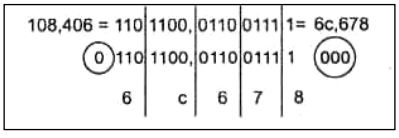


Рис.4.7

2. Практическое занятие. Представление чисел в различных системах счисления.

2.1. Представление чисел в двоичной системе.

2.2. Представление чисел в восьмеричной системе счисления.

2.3. Представление чисел в шестнадцатеричной системе счисления

2.4. Представление чисел в двоично-десятичной системе.

3. Практическое занятие. Перевод чисел из одной системы счисления в другую.

3.1. Перевод чисел из десятичной системы в двоичную, восьмеричную и шестнадцатеричную и обратно.

3.2. Перевод дробных чисел.

3.3. Перевод чисел со знаком.

1.2. Тема 2: «Методы логического описания электронных схем» (32 часа).

Перечень и краткое содержание рассматриваемых вопросов.

1. Лекция. Основы алгебры логики и формы представления ее функций.

1.1. Основные понятия алгебры логики.

Аппарат алгебры логики (алгебры Буля, Булевой алгебры) был создан в XIX в. английским математиком Дж. Булем и применяется для формального описания цифровых автоматов.

Основным понятием алгебры логики является *высказывание*.

Высказывание – некоторое предложение, о котором можно утверждать, что оно истинно или ложно.

Любое высказывание можно обозначить символом x и считать, что $x = 1$, если высказывание истинно, а $x = 0$ – если высказывание ложно.

Логическая (булева) переменная – такая величина x, которая может принимать только два значения (0 или 1).

Высказывание абсолютно истинно, если соответствующая ей логическая величина принимает значение $x = 1$ при любых условиях.

Высказывание абсолютно ложно, если соответствующая ей логическая величина принимает значение $x = 0$ при любых условиях.

Логическая функция (функция алгебры логики, ФАЛ) – функция $f(x1,x2,...,x_n)$, принимающая значение, равное 0 или 1 на наборе логических переменных $x1,x2,...,x_n$.

Логические функции от одной переменной представлены в табл.3.1.

Таблица 3.1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	1	0	0	1
1	1	0	1	0

Функция $f_1(x)$ является абсолютно истинной (константа единицы); функция $f_2(x)$ – абсолютно ложная (константа нуля).

Функция $f_3(x)$ – тождественная функция, а функция $f_4(x)$ называется функцией логического отрицания (функция НЕ, функция инверсии). Математически функция логического отрицания

$$f_4(x)=\neg x=\overline{x} \ .$$

В табл.3.2 представлены 5 из 16 наиболее часто употребляемых на практике логических функций от двух переменных.

Функция $f_1(x1,x2)$ называемая функцией дизъюнкции (функция логического сложения, функция ИЛИ), истинна тогда, когда истинны или $x1$, или $x2$, или обе переменные. Алгебраическое обозначение функции:

$$f_1(x1,x2) = x1 + x2 = x1 \ V \ x2 \ .$$

Таблица 3.2

$x1$	$x2$	$f_1(x1,x2)$	$f_2(x1,x2)$	$f_3(x1,x2)$	$f_4(x1,x2)$	$f_5(x1,x2)$
0	0	0	0	1	1	0
0	1	1	0	1	0	1
1	0	1	0	1	0	1
1	1	1	1	0	0	0

Функция $f_2(x1,x2)$ – функция конъюнкции (функция логического умножения, функция И) истинна только тогда, когда истинны $x1$ и $x2$.

Алгебраическое обозначение функции:

$$f_2(x1,x2) = x1 \cdot x2 = x1x2 = x1 \wedge x2 = x1 \,\&\, x2 .$$

Функция $f_3(x1,x2)$ называется "штрих Шеффера" (функция Шеффера). Эта функция ложна только тогда, когда $x1$ и $x2$ истинны. Алгебраическое обозначение функции:

$$f_3(x1,x2) = x1 \mid x2 .$$

Если сравнить функции $f_2(x1,x2)$ и $f_3(x1,x2)$ (см. табл.3.2), то видно, что функция Шеффера является инверсией функции И. Следовательно, функцию Шеффера можно еще назвать функцией И-НЕ. Алгебраическое обозначение можно записать так:

$$f_3(x1,x2) = x1 \mid x2 = \overline{x1x2} .$$

Функция $f_4(x1,x2)$ называется "стрелка Пирса" (функция Пирса, функция Вебба). Эта функция истинна только тогда, когда $x1$ и $x2$ ложны.

Алгебраическое обозначение функции:

$$f_4(x1,x2) = x1 \downarrow x2 = x1 \,\text{O}\, x2 .$$

Если сравнить функции $f_1(x1,x2)$ и $f_4(x1,x2)$, то видно, что функция Пирса является инверсией функции ИЛИ. Следовательно, функцию Пирса можно еще назвать функцией ИЛИ-НЕ. Алгебраическое обозначение функции можно записать так:

$$f_4(x1,x2) = x1 \downarrow x2 = x1 \,\text{O}\, x2 = \overline{x1+x2} .$$

Функция $f_5(x1,x2)$ называется функцией сложения по модулю 2 (функция неравнозначности, функция разноименности). Функция истинна, когда истинны или $x1$ или $x2$ в отдельности. Алгебраическое обозначение этой функции:

$$f_5(x1,x2) = x1 \oplus x2 .$$

1.2. Аксиомы и свойства алгебры логики.

Используя основные положения алгебры логики, подставляя вместо булевой переменной x ее возможные значения, нетрудно убедиться в справедливости следующих аксиом:

1. $x = \overline{\overline{x}}$ – закон исключения двойного отрицания.

2.	3.	4.
$x + 0 = x$	$x \cdot 0 = 0$	$x \oplus 0 = x$
$x + 1 = 1$	$x \cdot 1 = x$	$x \oplus 1 = \overline{x}$
$x + x = x$	$x \cdot x = x$	$x \oplus x = 0$
$x + \overline{x} = 1$	$x \cdot \overline{x} = 0$	$x \oplus \overline{x} = 1$

Дизъюнкция, конъюнкция и функция сложения по модулю 2 обладают рядом свойств, аналогичных свойствам обычных арифметических операций сложения и умножения.

1) свойство ассоциативности (сочетательный закон):

$$x1 + (x2 + x3) = (x1 + x2) + x3 ;$$
$$x1 \cdot (x2 \cdot x3) = (x1 \cdot x2) \cdot x3 ;$$
$$x1 \oplus (x2 \oplus x3) = (x1 \oplus x2) \oplus x3 .$$

2) свойство коммутативности (переместительный закон):

$$x1 + x2 = x2 + x1 ;$$
$$x1 \cdot x2 = x2 \cdot x1 ;$$

$$x1 \oplus x2 = x2 \oplus x1 .$$

3) свойство дистрибутивности (распределительный закон):

$$x1 \cdot (x2 + x3) = x1 \cdot x2 + x1 \cdot x3 ;$$
$$x1 + x2 \cdot x3 = (x1 + x2) \cdot (x1 + x3) ;$$
$$x1 \cdot (x2 \oplus x3) = x1 \cdot x2 \oplus x1 \cdot x3 .$$

Правила "склеивания":

$$x1\,x2 + x1\,\overline{x2} = x1 ; \qquad (x1 + x2) \cdot (x1 + \overline{x2}) = x1 . \qquad (3.1)$$

Законы де Моргана, с помощью которых возможно выразить конъюнкцию через дизъюнкцию и отрицание или дизъюнкцию через конъюнкцию и отрицание имеют следующий вид:

$$\overline{x1 \cdot x2} = \overline{x1} + \overline{x2} \quad ,$$
$$\overline{x1 + x2} = \overline{x1} \cdot \overline{x2} \quad .$$

Следствия из законов де Моргана:

$$x1 \cdot x2 = \overline{\overline{x1} + \overline{x2}} \quad ,$$
$$x1 + x2 = \overline{\overline{x1} \cdot \overline{x2}} \quad . \qquad (3.2)$$

Законы де Моргана справедливы для любого числа переменных.
Законы поглощения:

$$x1 + (x1 \cdot x2) = x1 ;$$
$$x1 \cdot (x1 + x2) = x1 .$$

Законы обобщенного поглощения:

$$x1 \cdot x2 + \overline{x1} \cdot x3 + x2 \cdot x3 = x1 \cdot x2 + \overline{x1} \cdot x3 ;$$
$$(x1 + x2) \cdot (\overline{x1} + x3) \cdot (x2 + x3) = (x1 + x2) \cdot (\overline{x1} + x3) .$$

Порядок выполнения логических операций:
1. Инверсия отдельных переменных.
2. Операции в скобках и операции со знаком отрицания.
3. Конъюнкция.
4. Дизъюнкция.

1.3. Табличное и аналитическое представление ФАЛ.

В табличном способе задания функций алгебры логики (ФАЛ) функция задается в виде таблицы, в которой для каждого набора значений переменных указывается значение самой логической функции. Табл.3.3 иллюстрирует табличный способ задания ФАЛ от трех переменных.

Таблица 3.3

x1	*x2*	*x3*	*f(x1,x2,x3)*
0	0	0	1
0	0	1	1
0	1	0	0

0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Этот способ показателен и может быть применен для записи ФАЛ от любого числа переменных. Однако при задании ФАЛ от большого числа переменных таблицы получаются слишком большие, т.к. число строк в такой таблице составляет 2^n, где n - число переменных.

При аналитической форме записи ФАЛ задается в виде объединения термов. Различают два вида термов - дизъюнктивные термы (макстермы) и конъюнктивные термы (минтермы).

Макстерм - терм, связывающий все переменные, представленные в прямой и инверсной форме, знаком дизъюнкции.

Например,

$$\Phi = \overline{x1} + x2 + x3 .$$

Минтерм - терм, связывающий все переменные, представленные в прямой и инверсной форме, знаком конъюнкции.

Например,

$$F = \overline{x1} \cdot x2 \cdot \overline{x3} .$$

Ранг терма равен числу переменных, входящих в данный терм.

Теорема 1. Любая таблично заданная ФАЛ, кроме абсолютно ложной, может быть представлена аналитическим способом в виде объединения минтермов:

$$f(x1,x2,...,xn) = F1 + F2 + ... + F_m , \qquad (3.3)$$

где m – число наборов переменных, на которых функция равна 1.

ФАЛ, заданная в табл.3.3 записанная в виде объединения минтермов:

$$f(x1,x2,x3) = \overline{x1}\cdot\overline{x2}\cdot\overline{x3}+\overline{x1}\cdot\overline{x2}\cdot x3+\overline{x1}\,x2\,x3 + x1\cdot x2\cdot\overline{x3}+x1\cdot x2\cdot x3.. \qquad (3.4)$$

Прямые следствия из теоремы 1:

– если функция равна 1, то в правой части аналитической записи (3.3) найдется хотя бы один минтерм равный 1;

– если функция равна 0, то в правой части соотношения (3.3) нет ни одного минтерма равного 1.

Обратные следствия из теоремы 1:

– если какой-либо минтерм равен 1, то функция равна 1;

– если какой-либо минтерм равен 0, то функция может быть равна 1.

Теорема 2. Любая таблично заданная ФАЛ, кроме абсолютно истинной, может быть представлена аналитическим способом в виде объединения макстермов:

$$f(x1,x2,x3) = \Phi1 \cdot \Phi2 \cdot ... \cdot \Phi_k , \qquad (3.5)$$

где k – число наборов переменных, на которых функция равна 0.

ФАЛ, заданную в табл.3.3, можно записать так:

$$f(x1,x2,x3) = (x1 + \overline{x2} + x3)\cdot(\overline{x1} + x2 + x3)\cdot(\overline{x1} + x2 + \overline{x3}). \qquad (3.6)$$

Прямые следствия из теоремы 2:

– если функция равна 0, то в правой части соотношения (3.5) найдется хотя бы один макстерм, равный 0;

– если функция равна 1, то в правой части соотношения (3.5) все макстермы равны 1.

Обратные следствия из теоремы 2:

– если какой-либо макстерм равен 0, то функция равна 0;

– если какой-либо макстерм равен 1, то функция может быть равна 0.

Нормальные и совершенные нормальные формы записи ФАЛ.

Нормальная дизъюнктивная форма (НДФ) аналитической записи ФАЛ – это объединение знаком дизъюнкции минтермов разного ранга. Например,

$$f(x1,x2,x3,x4) = x1 \cdot \overline{x3} + x1 \cdot \overline{x2} \cdot x4 .$$

Здесь аналитическая запись ФАЛ содержит два минтерма рангов 2 и 3.

Нормальной конъюнктивной формой (НКФ) записи ФАЛ называется объединение знаком конъюнкции макстермов разного ранга.

Например,

$$f(x1,x2,x3,x4) = \overline{x1} \cdot (x1 + \overline{x4}) \cdot (x1 + \overline{x2} + \overline{x3} + x4) .$$

Здесь аналитическая запись ФАЛ содержит три макстерма рангов 1, 2 и 4.

Нормальные дизъюнктивная и конъюнктивная формы не дают однозначного представления о функции, т.е. одна и та же ФАЛ может быть представлена несколькими нормальными формами. Кроме этого, по нормальной форме записи не видно, на каких наборах переменных функция равна 0, а на каких – 1.

От этих недостатков свободны совершенные нормальные формы (СНФ) – совершенная нормальная дизъюнктивная форма (СНДФ) и совершенная нормальная конъюнктивная форма (СНКФ).

Признаки СНФ :

– в СНФ нет одинаковых термов;

– в СНФ ни один терм не содержит двух одинаковых переменных;

– в СНФ ни один терм не содержит одну и ту же переменную в прямой и инверсной форме;

– в СНФ все термы имеют одинаковый ранг, который равен числу переменных функции.

Примером СНДФ может служить соотношение (3.4), а примером СНКФ - соотношение (3.6).

Для преобразования аналитической записи ФАЛ из нормальной формы в СНФ необходимо воспользоваться правилом склеивания (3.1). Записав его в обратном порядке, получим

$$x1 = x1 \cdot x2 + x1 \cdot \overline{x2} \quad . \qquad (3.7)$$

Таким образом в минтерм ранга 1, не содержащий переменных кроме *x1,* можно ввести недостающие переменные до получения минтема максимального для данной функции ранга.

Пример 3.1. Логическую функцию

$$f(x1,x2,x3) = x1 + x2 \cdot \overline{x3} + + \overline{x1} \cdot \overline{x2} \cdot x3$$

преобразовать в СНКФ.

Решение. НДФ содержит три минтерма ранга 1, 2 и 3. В первом минтерме не хватает переменных *x2* и *x3*, во втором - одной переменной *x1*. Для того чтобы ввести недостающие переменные, воспользуемся соотношением (3.7):

$$x1 = x1 \cdot x2 + x1 \cdot \overline{x2}$$

Из минтерма ранга 1 получились два минтерма ранга 2, в каждом из которых недостает переменной *x3*:

$$x1 \cdot x2 = x1 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot \overline{x3};$$
$$x1 \cdot \overline{x2} = x1 \cdot \overline{x2} \cdot x3 + x1 \cdot \overline{x2} \cdot \overline{x3}.$$

Таким образом, вместо минтерма *x1* получим:

$$x1 = x1 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot \overline{x3} + x1 \cdot \overline{x2} \cdot x3 + x1 \cdot \overline{x2} \cdot \overline{x3}.$$

Для второго минтерма

$$x2 \cdot \overline{x3} = x1 \cdot x2 \cdot \overline{x3} + \overline{x1} \cdot x2 \cdot \overline{x3}.$$

Записав вместо первого и второго минтермов исходной ФАЛ правые части полученных соотношений, получим

$$f(x1,x2,x3) = x1 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot \overline{x3} + x1 \cdot \overline{x2} \cdot x3 + x1 \cdot \overline{x2} \cdot \overline{x3} +$$
$$+ x1 \cdot x2 \cdot \overline{x3} + \overline{x1} \cdot x2 \cdot \overline{x3} + \overline{x1} \cdot \overline{x2} \cdot x3.$$

Однако эта запись не является СНДФ, поскольку в ней содержится два одинаковых минтерма (второй и пятый). Воспользовавшись аксиомой алгебры логики $x + x = x$ можно исключить один из этих минтермов.

Ответ: $f(x1,x2,x3) = x1 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot \overline{x3} + x1 \cdot \overline{x2} \cdot x3 + x1 \cdot \overline{x2} \cdot \overline{x3} +$
$+ \overline{x1} \cdot x2 \cdot \overline{x3} + \overline{x1} \cdot \overline{x2} \cdot x3.$

Для преобразования КНФ в СКНФ необходимо воспользоваться вторым из правил склеивания (3.1), записав его в обратном порядке:

$$x1 = (x1 + x2) \cdot (x1 + \overline{x2}).$$

2. Лекция. Методы минимизация функций алгебры логики.

Минимальная форма представления ФАЛ - форма, которая содержит минимальное количество термов и минимальное количество переменных в термах, т.е. минимальная форма не допускает никаких упрощений.

Если сравнивать в смысле минимальности различные формы представления ФАЛ, то очевидно, что нормальные формы экономичнее совершенных нормальных форм.

Комбинационная схема, реализующая ФАЛ и построенная по минимальной форме записи ФАЛ, дает минимальные аппаратурные затраты, уменьшение потребляемой энергии, уменьшение габаритов устройства и т.д.

2.1. Аналитический метод минимизации ФАЛ.

Так как минимальная форма записи ФАЛ не допускает никаких упрощений, то, например, функция $f(x1,x2,...,x_n) = x1 + x2$ является минимальной формой и, наоборот, функция $f(x1,x2,...,xn) = x1 + \overline{x1} \cdot x2$ может быть упрощена, если к этому выражению применить распределительный закон, т.е.

$$x1 + \overline{x1} \cdot x2 = (x1 + \overline{x1}) \cdot (x1 + x2) = x1 + x2.$$

Следовательно, упрощение сложных логических выражений может быть осуществлено по основным законам и аксиомам, изложенным выше.

2.2. Метод неопределенных коэффициентов.

В теореме Жегалкина [21, 22] говорится, что любая булева функция может быть представлена многочленом вида

$$f(x1,x2,...,x_n) = k_0 + k_1 \cdot x1 + k_2 \cdot x2 + ... + k_n \cdot x_n + \\ + k_{n+1} \cdot x1 \cdot x2 + k_{n+2} \cdot x1 \cdot x3 + k_{n+3} \cdot x1 \cdot x4 + ... + k_{n+m} \cdot x1 \cdot x2 \cdot ... \cdot x_n ,$$

где k_i - коэффициенты, принимающие значение 0 или 1.

Теорема Жегалкина дает возможность представить любую логическую функцию в виде полиномов разной степени. Например, ФАЛ *f(x1,x2,x3)* можно записать в следующей нормальной дизъюнктивной форме:

$$f(x1,x2,x3) = k_1^1 \cdot x1 + k_1^0 \cdot \overline{x1} + k_2^1 \cdot x2 + k_2^0 \cdot \overline{x2} + k_3^1 \cdot x3 + k_3^0 \cdot \overline{x3} + \\ + k_{12}^{00} \cdot \overline{x1} \cdot \overline{x2} + k_{12}^{01} \cdot \overline{x1} \cdot x2 + k_{12}^{10} \cdot x1 \cdot \overline{x2} + k_{12}^{11} \cdot x1 x2 + k_{13}^{00} \cdot \overline{x1} \cdot \overline{x3} + k_{13}^{01} \cdot \overline{x1} \cdot x3 + \\ + k_{13}^{10} \cdot x1 \cdot \overline{x3} + k_{13}^{11} \cdot x1 \cdot x3 + k_{23}^{00} \cdot \overline{x2} \cdot \overline{x3} + k_{23}^{01} \cdot \overline{x2} \cdot x3 + k_{23}^{10} \cdot x2 \cdot \overline{x3} + k_{23}^{11} \cdot x2 \cdot x3 + \\ + k_{123}^{000} \cdot \overline{x1} \cdot \overline{x2} \cdot \overline{x3} + k_{123}^{001} \cdot \overline{x1} \cdot \overline{x2} \cdot x3 + k_{123}^{010} \cdot \overline{x1} \cdot x2 \cdot \overline{x3} + k_{123}^{011} \cdot \overline{x1} \cdot x2 \cdot x3 + \\ + k_{123}^{100} \cdot x1 \cdot \overline{x2} \cdot \overline{x3} + k_{123}^{101} \cdot x1 \cdot \overline{x2} \cdot x3 + k_{123}^{110} \cdot x1 \cdot x2 \cdot \overline{x3} + k_{123}^{111} \cdot x1 \cdot x2 \cdot x3, \qquad (4.3)$$

где k_{ijh}^{lnm} - неопределенные коэффициенты, принимающие значение 0 или 1 и подбираемые так, чтобы получающаяся после этого НДФ была минимальной.

Так как функция может принимать на наборах переменных значения 0 или 1, то на основании уравнения (4.3) можно получить [21, 22] уравнения (4.4):

$$\begin{aligned}
k_1^0 + k_2^0 + k_3^0 + k_{12}^{00} + k_{13}^{00} + k_{23}^{00} + k_{123}^{000} &= f_0(0,0,0); \\
k_1^0 + k_2^0 + k_3^1 + k_{12}^{00} + k_{13}^{01} + k_{23}^{01} + k_{123}^{001} &= f_1(0,0,1); \\
k_1^0 + k_2^1 + k_3^0 + k_{12}^{01} + k_{13}^{00} + k_{23}^{10} + k_{123}^{010} &= f_2(0,1,0); \\
k_1^0 + k_2^1 + k_3^1 + k_{12}^{01} + k_{13}^{01} + k_{23}^{11} + k_{123}^{011} &= f_3(0,1,1); \\
k_1^1 + k_2^0 + k_3^0 + k_{12}^{10} + k_{13}^{10} + k_{23}^{00} + k_{123}^{100} &= f_4(1,0,0); \qquad (4.4) \\
k_1^1 + k_2^0 + k_3^1 + k_{12}^{10} + k_{13}^{11} + k_{23}^{01} + k_{123}^{101} &= f_5(1,0,1); \\
k_1^1 + k_2^1 + k_3^0 + k_{12}^{11} + k_{13}^{10} + k_{23}^{10} + k_{123}^{110} &= f_6(1,1,0); \\
k_1^1 + k_2^1 + k_3^1 + k_{12}^{11} + k_{13}^{11} + k_{23}^{11} + k_{123}^{111} &= f_7(1,1,1).
\end{aligned}$$

Метод неопределенных коэффициентов наиболее применим для дизъюнктивной формы и практически неприменим для конъюнктивной формы.

К недостаткам метода относится его громоздкость; число строк в таблице равно 2n, а число колонок - (2n - 1).

Преимущество - метод хорошо поддается алгоритмизации.

2.3. Метод минимизирующих карт Карно.

Метод карт Карно основан на графическом представлении ФАЛ и предназначен для минимизации функций от небольшого числа переменных.

Карта Карно представляет собой развертку на плоскости ФАЛ, представленной в геометрической форме. При этом вершины n-мерного куба представляются клетками карты, координаты которых совпадают с координатами соответствующих вершин куба. Карта заполняется так же, как таблица истинности – в клетках, соответствующих набору переменных. Ставятся значения функции.

На рис.4.1 показаны примеры карт Карно для двух (рис.4.1,а), трех (рис.4.1,б) и четырех (рис.4.1,в) переменных.

При минимизации, для получения минимальной ДНФ следует объединять («склеивать») по 2, 4, 8, 16 и т.д. (по 2^k, где $k = 1, 2, 3,\ldots$) клеток, содержащих 1. При этом клетки, лежащие на границах карты, также являются соседними по отношению друг к другу.

Для получения минимальной КНФ по тем же правилам объединяют клетки, содержащие 0.

Каждое объединение дает терм, содержащий лишь те переменные, которые не изменяются в данном объединении.

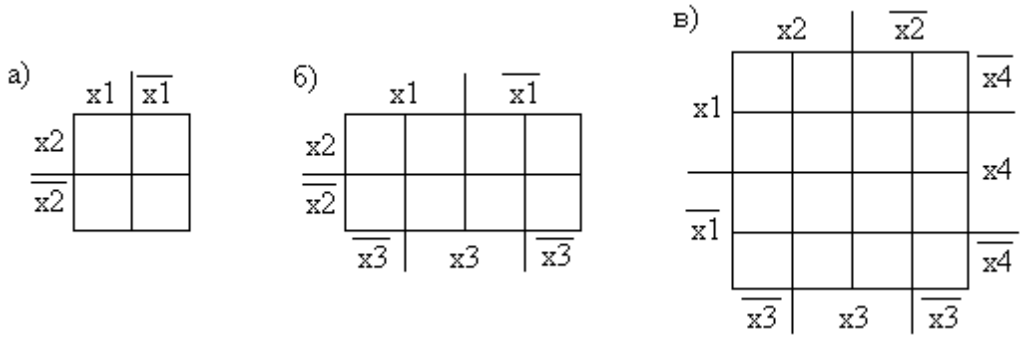


Рис.4.1. Минимизирующие карты

2.4. Минимизация не полностью определенных ФАЛ.

Не полностью определенная логическая функция *n* переменных - функция, заданная на числе наборов, меньших 2^n. Если количество неопределенных наборов *m*, то путем различных доопределений можно получить 2^m различных функций. Обычно набор переменных, на котором функция *f(x1,x2,...,xn)* не определена отмечается звездочкой *. Для этих наборов значение ФАЛ может быть произвольным.

Задача доопределения функции сводится к тому, что из 2^m возможных вариантов доопределения необходимо выбрать тот, который позволит получить минимальную форму записи ФАЛ. Для доопределения ФАЛ от небольшого числа переменных целесообразно воспользоваться картами Карно, поскольку они обеспечивают максимальную простоту и наглядность процесса доопределения. *0* и *1*, подставляемые вместо звездочек, располагают таким образом, чтобы получить как можно меньше объединений, и чтобы в каждом объединении присутствовало возможно большее число единиц (для ДНФ) или нулей (для КНФ).

3. Лекция. Реализация функций алгебры логики.

3.1. Функциональные логические элементы.

Техническим аналогом булевой функции является комбинационная схема, выполняющая соответствующее этой функции преобразование информации. Уровни напряжения шин, соответствующие принятому в схеме представлению сигналов *0* и *1*, рассматриваются как технические аналоги функции константы **0** и константы *1*.

Элементарные логические операции над двоичными переменными реализуются схемами, которые называются логическими элементами. Число входов логического элемента соответствует числу аргументов воспроизводимой им булевой функции.

На рис.3.4 приведены условные графические обозначения схем, реализующих логические функции базиса Дж. Буля.

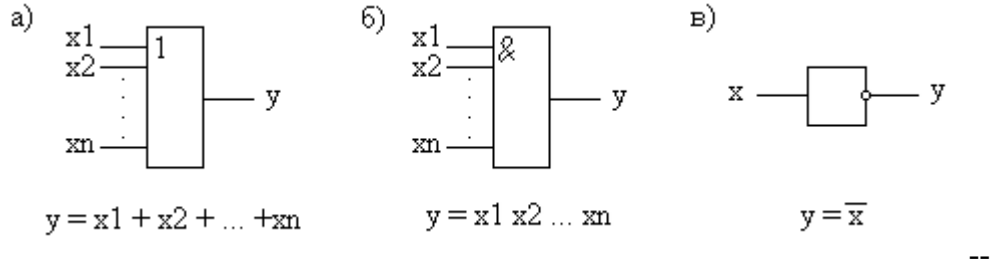


Рис.3.4. Условные графические обозначения элементов
ИЛИ (а), И (б), и НЕ (в)

Условные графические обозначения схем, реализующих логические функции Шеффера, Пирса и сложение по модулю 2, представлены на рис.3.5.

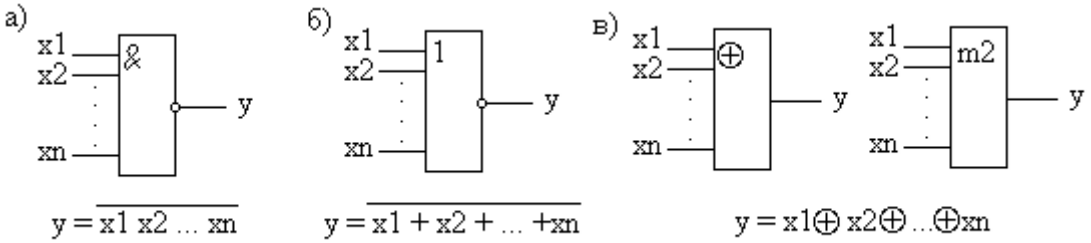


Рис.3.5. Условные графические обозначения элементов Шеффера (а),
Пирса (б) и сложения по модулю 2 (в)

В литературе элемент Шеффера часто называют элементом И-НЕ, а элемент Пирса – элементом ИЛИ-НЕ. Например, обозначение 4И-НЕ свидетельствует о том, что это четырехвходовой элемент Шеффера.

Подобно тому, как сложная булева функция может быть получена суперпозицией более простых функций, так и комбинационная схема строится из элементарных схем – логических элементов. Подача выходного сигнала *y* одной схемы на вход x_i другой схемы соответствует подстановке в логические функции в качестве аргументов других логических функций.

Схему, показывающую связи между различными логическими элементами, где сами элементы представлены условными обозначениями, называют функциональной схемой.

Если при реализации функции возникает необходимость в инверсных значениях некоторых входных переменных, то соответствующие входы отмечаются кружком.

Выход отмечается кружком, если функция реализуется с инверсией.

На рис.3.6 показан логический элемент, реализующий функцию $f(x1,x2,x3)=\overline{x1\cdot\overline{x2}\cdot x3\cdot\overline{x4}}$.

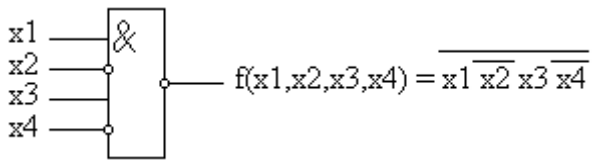


Рис.3.6. Условное обозначение элемента, реализующего
функцию $f(x1,x2,x3)=\overline{x1\cdot\overline{x2}\cdot x3\cdot\overline{x4}}$

Набор логических элементов для построения логических комбинационных схем является функционально полным, если реализуемые этими элементами логические функции образуют функционально полную систему функций.

3.2. Синтез схем с одним выходом.

Схемы с несколькими входами и одним выходом - наиболее простые схемы. Основная сложность синтеза состоит в том, чтобы найти выражение для выходной функции в заданном базисе.

Для любой ФАЛ можно синтезировать схему, соответствующую заданной функции. Однако в зависимости от выбранной системы логических элементов (базиса) задача синтеза имеет различные решения. Полученные схемы с минимальным количеством элементов требуют нахождения минимальной формы записи ФАЛ.

Рассмотрим в качестве примера синтез комбинационной схемы, реализующей таблично заданную функцию, представленную в табл.4.1.

Пример 4.6. Синтезировать схему, реализующую ФАЛ

$F = \overline{x1} \cdot \overline{x2} + \overline{x1} \cdot x3 + x1 \cdot x2$, минимизированную в примере 4.1, в базисах И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ.

Решение. Функция $F = \overline{x1} \cdot \overline{x2} + \overline{x1} \cdot x3 + x1 \cdot x2$ представлена в булевом базисе (в базисе И,ИЛИ,НЕ) и для ее реализации потребуется два элемента НЕ для инвертирования входных переменных $x1$ и $x2$, три двухвходовых схемы логического умножения и одна трехвходовая схема логического сложения. Реализация ФАЛ в базисе И, ИЛИ, НЕ представлена на рис.4.5.

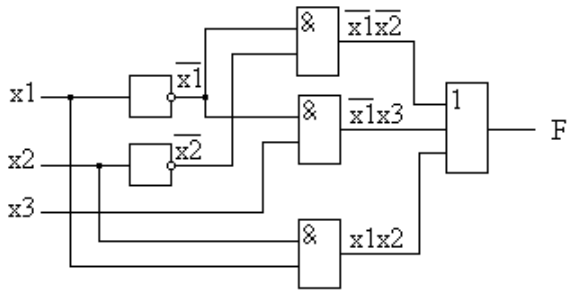


Рис.4.5. Логическая схема на элементах И, ИЛИ, НЕ

Функция в базисе И-НЕ:

$$F = \overline{x1} \cdot \overline{x2} + \overline{x1} \cdot x3 + x1 \cdot x2 = \overline{\overline{x1} \cdot \overline{x2}} \cdot \overline{\overline{x1} \cdot x3} \cdot \overline{x1 \cdot x2} \ .$$

Логическая схема в базисе И-НЕ будет содержать два инвертора, две двухвходовые схемы И-НЕ (2И-НЕ) и одну трехвходовую схему И-НЕ (3И-НЕ). Схема представлена на рис.4.6.

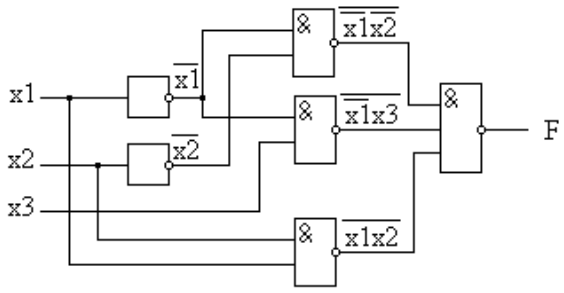


Рис.4.6. Логическая схема на элементах И-НЕ

Перевод в базис ИЛИ-НЕ:

$$F = \overline{x1} \cdot \overline{x2} + \overline{x1} \cdot x3 + x1 \cdot x2 = \overline{\overline{x1+x2}+\overline{x1+\overline{x3}}+\overline{\overline{x1}+\overline{x2}}} \ .$$

Логическая схема в базисе ИЛИ-НЕ будет содержать три инвертора, три схемы 2ИЛИ-НЕ и одну - 3ИЛИ-НЕ (рис.4.7).

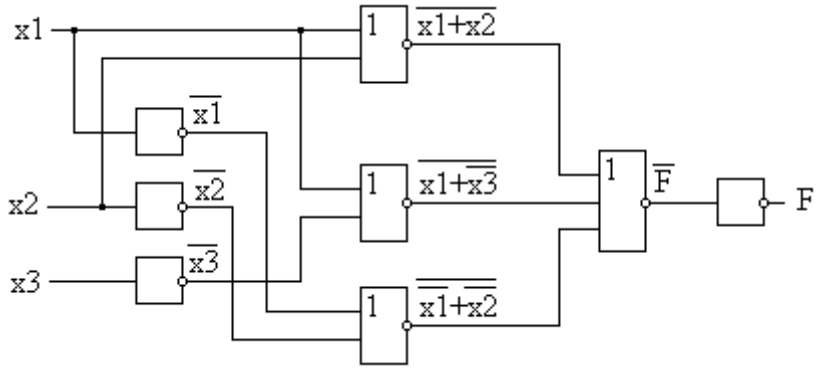


Рис.4.7. Логическая схема на элементах ИЛИ-НЕ

Ответ: Логические схемы на рис.4.5, 4.6, 4.7.

4. Практическое занятие. Логические функции и способы их задания.
4.1. Логические функции и их суперпозиция.
4.2. Способы задания логических функций.
4.3. Переход от структурной формулы к логической схеме

5. Практическое занятие. Минимизация логических функций.
5.1. Аналитический метод. Метод Квайна.
5.2. Метод диаграмм Вейча.
5.3. Карты Карно.

6. Практическое занятие. Синтез комбинационных схем.
6.1. Синтез комбинационных схем с одним выходом.
6.2. Синтез комбинационных схем с множественным выходом.

1.3. Тема 3: «Основные комбинационные узлы цифровых вычислительных машин» (22 часа).

Перечень и краткое содержание рассматриваемых вопросов.

1. Лекция. Основные комбинационные узлы цифровых вычислительных машин.

Рассмотрим основные комбинационные узлы ЭВМ. К ним относятся дешифраторы, шифраторы, мультиплексоры, демультиплексоры, одноразрядные сумматоры, полусумматоры, комбинационные счетчики, формирователи и сдвигатели.

В основе синтеза всех комбинационных схем лежит канонический метод синтеза, рассмотренный в предыдущем разделе. Таким образом, все логические схемы, синтезированные в предыдущем разделе, являются комбинационными схемами.

Основными характеристиками комбинационных схем являются сложность и быстродействие. Сложность схемы оценивается количеством оборудования, составляющего схему (числом корпусов микросхем). Быстродействие оценивается максимальной задержкой сигнала при прохождении его от входа схемы к выходу. Задержка обычно кратна числу элементов, через которые проходит сигнал от входа к выходу.

Рассмотрим дешифраторы, шифраторы, мультиплексоры и демультиплексоры, сумматоры и полусумматоры.

1.1.Дешифратор. Шифратор.

Дешифратором называется комбинационная схема с несколькими входами и выходами, преобразующая традиционный позиционный двоичный код в унитарный код.

Унитарным кодом называется код, состоящий из m двоичных переменных, из которых в каждый момент времени только одна переменная имеет значение 1, а остальные – равны 0.

В общем случае дешифратор с n входами имеет 2^n различных значений и каждому из этих значений соответствует сигнал 1 на одном из выходов дешифратора; 3-входовый дешифратор (рисунок 1) функционирует в соответствии с таблицей истинности (таблица 1).

По таблице истинности дешифратора (табл. 1), представляющей собой систему из восьми ФАЛ, легко построить логическую схему дешифратора в любом из базисов.

В некоторых случаях функция дешифратора ограничивается выделением только $m < 2^n$ входных наборов из общего числа $M = 2^n$. При этом предполагается, что $M - m$ входных наборов не существуют. Дешифраторы с числом выходов $m < 2^n$ называются неполными.

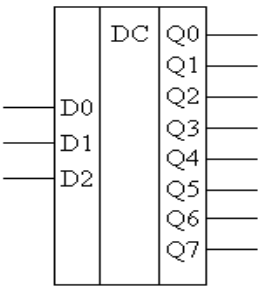


Рисунок 1 – Условное обозначение дешифратора

Таблица 5.1

D2	D1	D0	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Шифратор – комбинационная схема, предназначенная для преобразования унитарного кода в двоичный позиционный код. Таким образом, шифратор реализует преобразование обратное функции дешифратора.

На функциональных схемах шифраторы изображаются как показано на рисунке 2. Таблица истинности шифратора приведена в таблице 2.

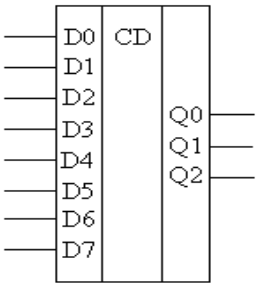


Рисунок 2 – Условное обозначение шифратора

Таблица 5.2

D0	D1	D2	D3	D4	D5	D6	D7	Q2	Q1	Q0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

1.2.Мультиплексор. Демультиплексор.

Мультиплексором называется комбинационная схема, имеющая $n+2^n$ входов и один выход, где n – число адресных входов, а 2^n – число информационных входов мультиплексора. Адреса представляются в двоичном коде и им присваивается номер *i*. Каждому адресу с номером *i* соответствует свой информационный вход A_i, сигнал с которого при данном адресе проходит на выход. Основным назначением мультиплексора является коммутация 2^n входных сигналов на один выход. На рисунке 3 представлено обозначение мультиплексора, имеющего 4 информационных входа. Таблица истинности этого мультиплексора представлена в таблице 3.

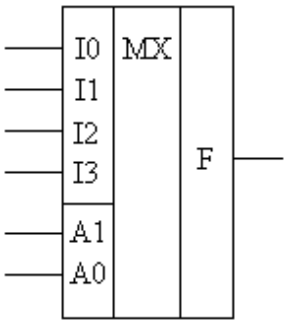


Рисунок 3 – Условное обозначение мультиплексора

Таблица 3

A1	A0	Выход F
0	0	I0 → F
0	1	I1 → F
1	0	I2 → F
1	1	I3 → F

Логическая схема мультиплексора состоит из дешифратора и вентильной части (рисунок 4).

Кроме основного назначения (коммутации сигналов), мультиплексоры могут быть использованы, например, для синтеза схем, выполняющих любую ФАЛ. В этом случае на информационные входы подаются не изменяющиеся во времени сигналы 0 и 1. Считывание данных сигналов производится подачей соответствующих сигналов на адресные входы. На рисунке 5 приведена реализация ФАЛ, заданной в таблице 1.

Если в нескольких однотипных мультиплексорах объединить одноименные управляющие входы, то будем иметь возможность коммутировать не один входной сигнал из *n* возможных, а целую группу входных сигналов (слово).

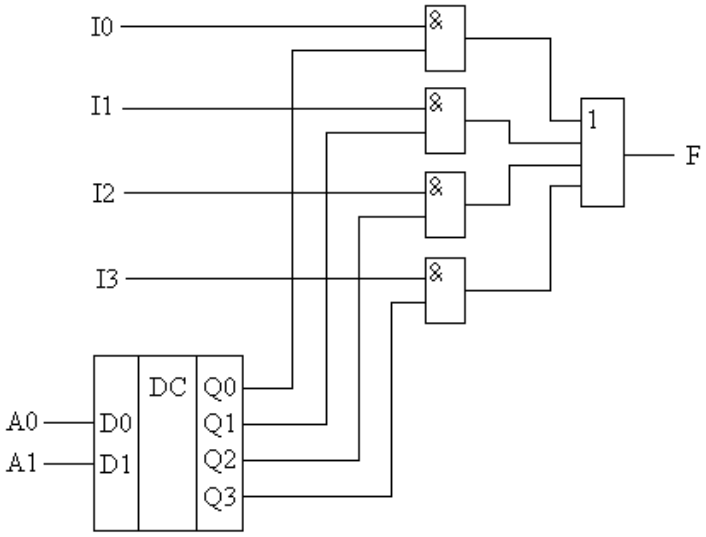


Рисунок 4 – Логическая схема мультиплексора

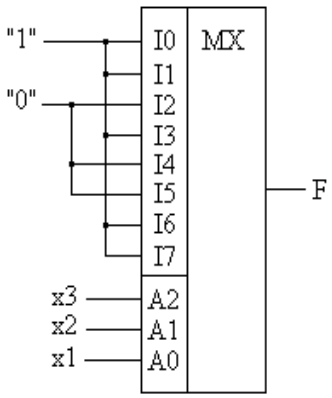
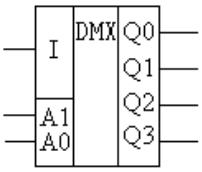


Рисунок 5 – Реализация ФАЛ на мультиплексоре

Демультиплексоры выполняют функцию, обратную мультиплексорам, т.е. производят коммутацию одного информационного входного сигнала на 2^n выходов, где n - число адресных входов. Условное графическое обозначение демультиплексора, предназначенного для коммутации сигнала от одного источника информации на один из четырёх приемников, показано на рисунке 6. Логическая схема, реализующая функцию демультиплексора, представлена на рисунке 7.

Рисунок 6 – Условное обозначение демультиплексора



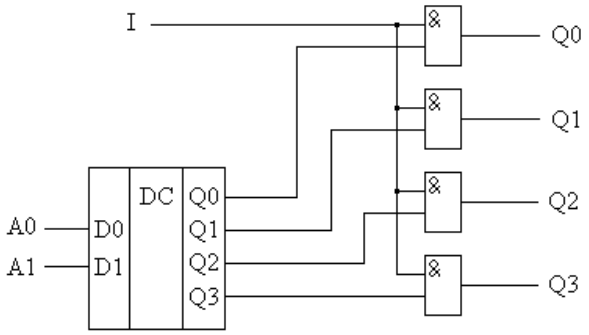


Рисунок 7 – Логическая схема демультиплексора

1.3. Одноразрядный сумматор и полусумматор.

Одноразрядные сумматор и полусумматор предназначены для сложения двоичных разрядов.

Полусумматор представляет собой комбинационную схему, имеющую два входа и два выхода. Условное графическое обозначение полусумматора приведено на рисунке 8,а, где *A* и *B* – слагаемые двоичные цифры, каждая из которых может принимать значение 0 или 1; *S* и *P* – выводы, на которых появляется результат сложения. Вывод *S* предназначен для выдачи суммы слагаемых *A* и *B*, а вывод *P* – для выдачи переноса в следующий *(i+1)*-й разряд. Полусумматор функционирует в соответствии с таблицей истинности (табл. 4).

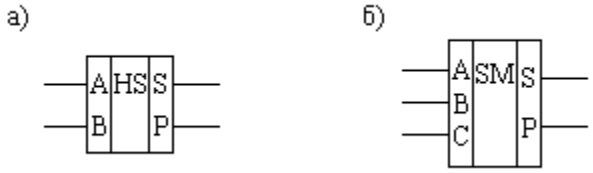


Рисунок 8 – Условное графическое обозначение
полусумматора HS(а) и сумматора SM(б)

Таблица 5.4

A	*B*	*P*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Таблица 5.5

A	*B*	*C*	*P*	*S*
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

В соответствии с табл. 4 функции выходов *S* и *P* имеют вид

$$P = A \cdot B\ ;$$

$$S = \overline{A} \cdot B + A \cdot \overline{B} \quad или \quad S = \overline{A \cdot B} \cdot (A + B) \quad .$$

Полный одноразрядный двоичный сумматор предназначен для построения многоразрядных сумматоров и имеет три входа и, следовательно, предназначен для сложения трёх двоичных цифр. Условное графическое обозначение полного одноразрядного двоичного сумматора приведено на рисунке 8,б, а таблица истинности, в

соответствии с которой функционирует сумматор, – в таблице 5. *A* и *B* – двоичные разряды чисел. *C* – входной перенос в данный разряд из предыдущего разряда. Это разделение входов сумматора условно, т.к. все три входа *A*, *B* и *C* имеют одинаковый вес. На выходе *S* появляется разряд суммы, а на выходе *P* – разряд переноса в следующий разряд.

Функциональная схема полного одноразрядного двоичного сумматора строится на основе функций суммы *S* и переноса *P*, составляемых по таблице истинности (табл. 5):

$$P = A \cdot B + A \cdot C + B \cdot C \ ;$$
$$S = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + ABC =$$
$$= \overline{A}(\overline{B}\,C + B\overline{C}) + A\,\overline{B}\,\overline{C} + ABC = \overline{A}(B \oplus C) + A\,\overline{B}\,\overline{C} + ABC \quad .$$

Наиболее удачные схемы одноразрядных сумматоров, содержащих наименьшее количество элементов и наименьшее количество входов у элементов, получены эмпирическим путём. Одна из таких функциональных схем приведена на рисунке 9. Логические уравнения для значений суммы и переноса в этой схеме следующие:

$$P = A \cdot B + A \cdot C + B \cdot C \ ;$$
$$S = ABC + (A + B + C) \cdot \overline{P}.$$

Как видно из рисунка 9, схема состоит из 9 логических элементов. Общее число входов – 20. В данной схеме ни один логический элемент не нагружается на входы двух других элементов.

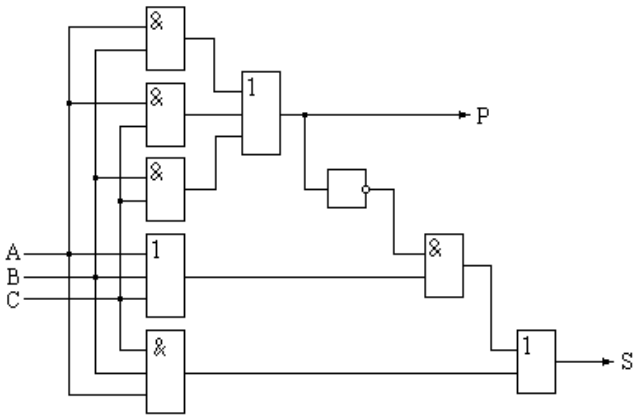


Рис. 9. Функциональная схема одноразрядного сумматора
на элементах "И", "ИЛИ", "НЕ"

Если для реализации сумматора использовать соотношения

$$P = A \cdot B + (A + B) \cdot C \ ;$$
$$S = ABC + (A + B + C) \cdot \overline{P},$$

то у девяти используемых логических элементов общее число входов будет равно 19.

Функциональная схема полного двоичного одноразрядного сумматора, состоящая из двух совершенно одинаковых частей (полусумматоров и схемы "2ИЛИ", объединяющей сигналы с этих частей, представлена на рисунке 10. Полусумматор *HS1* суммирует две цифры слагаемых без учёта переносов из младшего разряда. При этом вырабатывается промежуточный сигнал переноса *p1* и промежуточная сумма чисел *s1*. Полусумматор *HS2* производит суммирование промежуточной суммы *s1* и сигнала переноса *C* из младшего разряда.

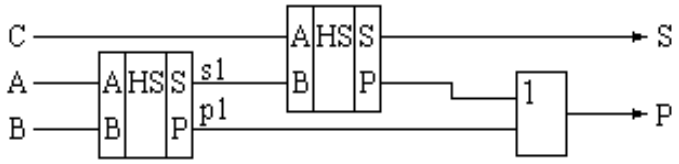


Рисунок 10 – Функциональная схема одноразрядного сумматора, состоящего из двух полусумматоров и схемы "2ИЛИ"

Если в полусумматоре *HS1* сигнал переноса не возник, то он может возникнуть в полусумматоре *HS2* при условии поступления на вход сумматора сигнала переноса из младшего разряда и значения "1" одного из слагаемых. Если же сигнал переноса появился в полусумматоре *HS1*, то сигнал переноса *p1* из полусумматора *HS2* не появится. Общий сигнал переноса сумматора равен логической сумме переносов обоих полусумматоров. Сигнал суммы *S* вырабатывается на выходе полусумматора *HS2*.

Если реализовать схему, изображённую на рисунке 10, на логических элементах "И", "ИЛИ", "НЕ", то общее число элементов будет равно 9. Общее число входов такой схемы равно 16. Таким образом, схема полного одноразрядного двоичного сумматора, состоящего из двух полусумматоров, является наиболее экономичной.

2. Практическое занятие. Реализация схем дешифратора и шифратора.

2.1. Построение комбинационной схемы с множеством выходов в заданном базисе элементов.

2.2. Построение системы ПФ.

2.3. Реализация схемы для семисегментных индикаторов на интегральной микросхеме дешифратора

3. Практическое занятие.Синтез сумматоров.

3.1.Записать аналитическое выражение для выходной функции.

3.2. Построить таблицу истинности для выходной функции F.

3.3. Построить электрическую принципиальную схему в заданном базисе.

1.4. Тема 4: «Основы теории конечных автоматов» (36 часов).

Перечень и краткое содержание рассматриваемых вопросов.

1. Лекция. Основы теории цифровых автоматов.

1.1. Основные понятия и определения.

Термин "автомат" используется в двух аспектах. С одной стороны, автомат – устройство, выполняющее некоторые функции без участия человека (например, ЭВМ). С другой стороны, термин "автомат" – математическое понятие и обозначает математическую модель реальных технических процессов.

В общем случае автомат представляется как "черный ящик" и полностью описывается совокупностью следующих шести объектов [20, 25, 26]:

1) множество входных сигналов X;

2) множество выходных сигналов Y;

3) множество состояний автомата A;

4) начальное состояние автомата a_0 как элемент множества A;

5) функция перехода из одного состояния в другое $f(a, x)$;

6) функция выходов автомата $\varphi(a, x)$.

Автомат называется конечным, если множество его внутренних состояний, входных и выходных сигналов – конечные множества.

Цифровой автомат – устройство, предназначенное для преобразования цифровой информации.

В начальный момент времени t_0 автомат находится в состоянии a_0. В каждый момент времени t, определяемый интервалом дискретности, автомат под воздействием входного сигнала $x(t)$ скачкообразно переходит из состояния $a_i(t)$ в состояние $a_i(t+1)$ и выдает соответствующий выходной сигнал $y(t)$.

Понятие состояние автомата используется для описания систем, выходы которых зависят не только от входных сигналов в данный момент времени, но и от некоторой предыстории, – сигналов, которые поступили на входы системы ранее, т.е. состояние – некоторая память о прошлом.

Цифровые автоматы делят на два класса.

В синхронных автоматах моменты времени, в которые фиксируются состояния автомата, задаются специальным устройством – генератором синхроимпульсов.

В асинхронных автоматах моменты перехода автомата из одного состояния в другое заранее не определены и зависят от ковких-то событий.

В зависимости от закона определения выходных сигналов синхронные автоматы делятся на автоматы Мили и автоматы Мура.

Для автомата Мили закон функционирования определяется следующими соотношениями:

$$\begin{cases} a(t+1) = f(a(t), x(t)), \\ y(t) = \varphi(a(t), x(t)). \end{cases} \tag{6.1}$$

Следующее состояние автомата Мили (состояние в момент времени $t+1$) зависит от состояния автомата в текущий момент времени t и входного сигнала x в текущий момент времени. Выходной сигнал в момент времени t определяется текущим внутренним состоянием и текущим входным сигналом.

Работа автомата Мура описывается соотношениями

$$\begin{cases} a(t+1) = f(a(t), x(t)), \\ y(t) = \varphi(a(t)). \end{cases} \tag{6.2}$$

Следующее состояние автомата Мура есть функция от текущего состояния и текущего входного сигнала, а выходной сигнал в текущий момент времени t зависит только от текущего внутреннего состояния.

1.2. Способы описания автоматов.

Среди зыков описания наиболее распространены таблицы и графы переходов и выходов.

Таблица переходов (выходов) представляет собой таблицу с двойным входом, строки которой нумерованы входными сигналами, а столбцы – состояниями. На пересечении указывается состояние, в которое переходит автомат (в таблице переходов) или выходной сигнал, выдаваемый им (в таблице выходов).

Ниже представлены таблица переходов (табл. 6.1.) и таблица выходов (табл. 6.2.) автомата Мили.

Таблица 6.1

Входной сигнал	Внутреннее состояние			
	a_0	a_1	a_2	a_3
x_1	a_1	a_2	a_3	a_0
x_2	a_3	a_2	a_2	a_1

Таблица 6.2

Входной сигнал	Внутреннее состояние			
	a_0	a_1	a_2	a_3
x_1	y_1	y_1, y_3	y_2	y_3
x_2	y_1	y_2	y_3	y_4

Например, если автомат Мили в текущий момент времени находится в состоянии a_1 и на входе автомата присутствует входной сигнал x_1, то на выходе автомата будут присутствовать выходные сигналы y_1, y_3 (см. табл. 6.2), а следующее состояние a_2 (см. табл. 6.1).

Иногда при задании автомата Мили используют одну совмещенную таблицу переходов и выходов, в которой на пересечении столбца a_i и строки x_j записывают в виде a_k/y_m следующее состояние и выдаваемый выходной сигнал. Совмещенная таблица переходов и выходов автомата Мили, заданного табл. 6.1 и 6.2, представлена в табл. 6.3.

Таблица 6.3

Входной сигнал	Внутреннее состояние			
	a_0	a_1	a_2	a_3
x_1	a_1/y_1	$a_2/y_1,y_3$	a_3/y_2	a_0/y_3
x_2	a_3/y_1	a_2/y_2	a_2/y_3	a_1/y_4

Так как в автомате Мура выходной сигнал зависит только от состояния, автомат Мура задается одній отмеченной таблицей переходов (табл. 6.4), в которой каждому её

столбцу приписан кроме состояния a_i ещё и выходной сигнал, соответствующий этому состоянию. Например, автомат Мура, заданный табл. 6.4, находясь в состоянии a_2, выдает выходной сигнал y_1 и при поступлении входного сигнала x_2 перейдет в следующее состояние – a_1.

Таблица 6.4

Входной сигнал	Выходные сигналы				
	y_2	y_3	y_1	y_4	y_5
	Внутреннее состояние				
	a_0	a_1	a_2	a_3	a_4
x_1	a_1	a_2	a_3	a_4	a_0
x_2	a_3	a_2	a_1	a_0	a_4

Более наглядно и удобно автомат можно задать с помощью графа автомата. Граф автомата – ориентированный граф, вершины которого соответствуют состояниям, а дуги – переходам между ними. Дугам автомата Мили приписывается входной сигнал, обеспечивающий данный переход и выходной сигнал, который при этом выдаётся. При описании автомата Мура в виде графа выходной сигнал, соответствующий данному состоянию, записывается внутри вершины или рядом с ней. На рис. 6.1 приведень графы автомата Мили (рис. 6.1, *а*) и автомата Мура (рис. 6.1, *б*), заданных табл. 6.3 и 6.4.

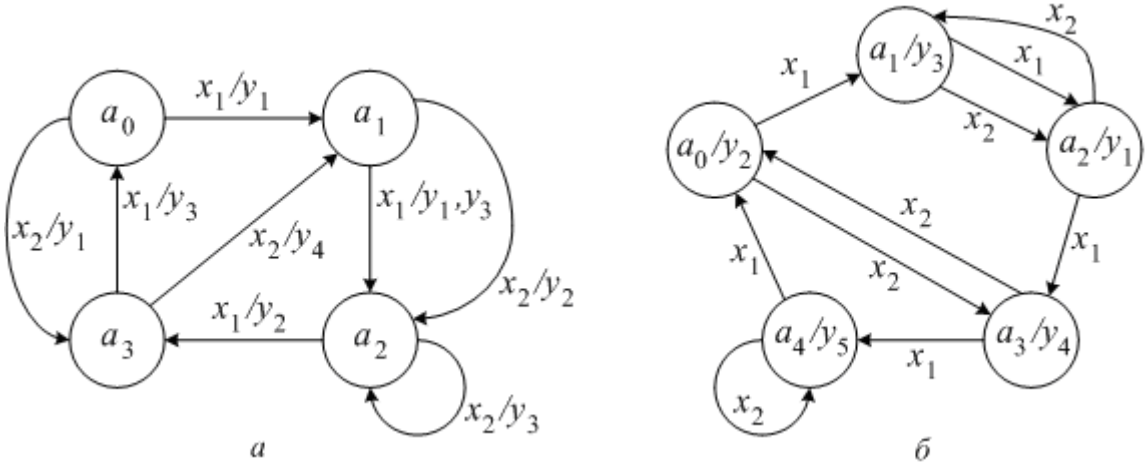


Рис. 6.1. Графы автоматов Мили (*а*) и Мура (*б*)

В графе автомата не должно существовать двух дуг с одинаковыми входными сигналами, выходящих из одной и той же вершины (условие однозначности).

Состояние автомата называется тупиковым, если соответствующая вершина графа не содержит исходящих дуг, но имеет хотя бы одну входящую дугу.

Изолированным состоянием называется такое состояние, которому соответствует вершина графа, не имеющая как входящих, так и исходящих дуг.

2. Лекция. Типовые узлы цифровых вычислительных машин.

2.1. Элементарные автоматы – триггеры.

В большинстве схем современных ЭВМ в качестве элементов памяти применяются элементарные автоматы, представляющие собой физические приборы с двумя устойчивыми состояниями (триггеры).

Для удобства использования в схемах вычислительных устройств триггеры обычно имеют два выхода – прямой Q (называется также *выход* 1) и инверсный $\overline{Q}$ (*выход* 0). Когда состояние триггера отображается парой сигналов Q и $\overline{Q}$, то это значит, что состояние представляется в парафазном коде. Выходные сигналы триггеров определяются внутренними состояниями. Двум внутренним состояниям соответствуют два различных выходных сигнала, которые по существу и позволяют физически различать состояния элементарных автоматов.

Элементарные автоматы (триггеры) переключаются из одного состояния в другое под управлением входных сигналов (сигналов возбуждения), число и воздействие которых на состояние триггера определяется типом триггера.

Триггеры разделяют на два класса – синхронные и асинхронные. Синхронный триггер имеет синхронизирующий вход C для подай сигнала, разрешающего переключение триггера. Если $C = 0$, то триггер сохраняет своё состояние независимо от значений сигналов на его остальных управляющих входах. Триггер переключается только в момент поступления сигнала $C = 1$.

Асинхронный триггер переключается в тот момент, когда изменяется значение сигналов на его входах.

Законы функционирования триггеров задаются таблицами переходов.

В зависимости от набора управляющих входов, которыми могут быть оснащены триггеры, выделяют следующие основные типы триггеров: *RS*-триггеры, *D*-триггеры, *T*-триггеры, *JK*-триггеры.

2.2. RS-триггеры.

Элементарные асинхронные *RS*-триггеры строятся на основе логических схем ИЛИ-НЕ или И-НЕ, выход каждой из которых соединен с одним из входов другой. В простейшем случае для построения триггеров используются двухвходовые логические элементы. В результате указанного соединения в схеми вводится положительная обратная связь и создаются условия возникновения двух устойчивых состояний.

Функциональная структура и условное графическое изображение асинхронных *RS*-триггеров на основе логических элементов 2ИЛИ-НЕ и 2И-НЕ приведены соответственно на рис. 6.2, *а*, *б*, *в*.

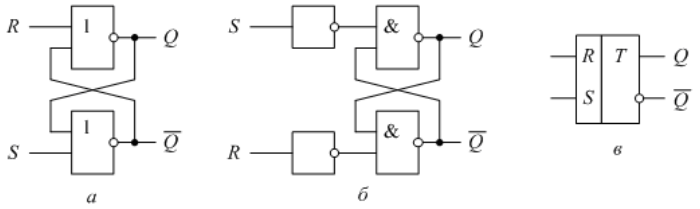


Рис. 6.2. Функциональные схемы асинхронных *RS*-триггеров

на элементах ИЛИ-НЕ (*а*), И-НЕ (*б*), и их условное графическое изображение (*в*)

Асинхронный *RS*-триггер переключается под воздействием сигналов R и S в соответствии с таблицей истинности (табл. 6.5).

Вход S называют входом установки триггера в единичное состояние (*set*), а вход R – входом установки в нулевое состояние или входом сброса (*reset*).

Синхронные (синхронизируемые, стробируемые, тактируемые) RS-триггеры имеют на каждом входе дополнительные схемы совпадения импульса синхронизации C с информационными сигналами сброса и установки.

Таблица 6.5

R	*S*	Q_t	Q_{t+1}	Режим
0	0	0	0	Хране-
0	0	1	1	ние
0	1	0	1	Установ-
0	1	1	1	ка "1"
1	0	0	0	Установ-
1	0	1	0	ка "0"
1	1	0	*	Запрещ.
1	1	1	*	режим

Таблица 6.6

C	*R*	*S*	Q_{t+1}	Режим
0	0	0	Q_t	Хранение
0	0	1	Q_t	
0	1	0	Q_t	
0	1	1	Q_t	
1	0	0	Q_t	Хранение
1	0	1	1	Установка "1"
1	1	0	0	Установка "0"
1	1	1	*	Запрещ. режим

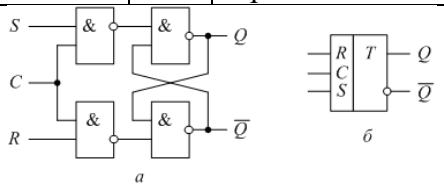


Рис. 6.3. Функциональная схема синхронного *RS*-триггера (*а*) и его условное графическое изображение (*б*)

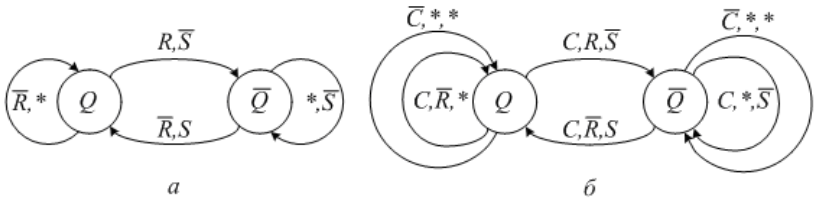


Рис. 6.4. Графы асинхронного (*а*) и синхронного (*б*) *RS*-триггеров

На рис. 6.3 представлены функциональная схема синхронного *RS*-триггера на основе логических элементов И-НЕ (*а*) и его условное графическое изображение. Таблица переходов синхронного *RS*-триггера представлена в табл. 6.6. Графы асинхроного и синхронного представлены соответственно на рис. 6.4, *а* и 6.4, *б*.

2.3. D-триггеры.

D-триггер (триггер задержки) – запоминающий элемент с двумя устойчивыми состояниями и одним информационным входом. Функция перехода D-триггера задается

таблицей истинности (табл. 6.7), а его условное графическое изображение представлено на рис. 6.5.

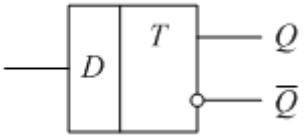


Рис. 6.5. Условное графическое изображение асинхронного*D*-триггера

Таблица 6.7

D	Q_t	Q_{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

Таблица 6.8

C	D	Q_{t+1}	Режим
0	0	Q_t	Хранение
0	1	Q_t	
1	0	0	Установка "0"
1	1	1	Установка "1"

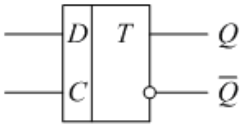


Рис. 6.6. Условное графическое изображение синхронного*D*-триггера

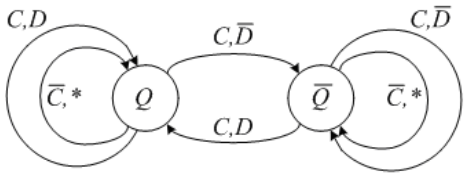


Рис. 6.7. Граф синхронного *D*-триггера

В отличие от *RS*-триггера *D*-триггер имеет только режимы установки 1 и 0. Асинхронный *D*-триггер практически не применяется, так как его выход будет просто повторять входной сигнал. Синхронный *D*-триггер (рис. 6.6) задерживает распространение входного сигнала на время паузы между синхросигналами.

Таблица истинности представлена в табл. 6.8.

Граф синхронного D-триггера приведён на рис. 6.7

2.4. Т-триггеры.

T-триггер (счетный триггер, триггер со счетным входом, дволичним счётчик) – запоминающий элемент, имеющий два устойчивых состояния и изменяющий своё состояние после прихода каждого счётного импульса на вход *T*. *T*-триггер осуществляет операцию суммирования по модулю 2 сигнала состояния триггера *Q* и входного сигнала, поступающего на вход *T*.

Единичный входной сигнал меняет состояние триггера на противоположное, а нулевой – оставляет без изменения.

Условные графические изображения асинхронного и синхронного *T*-триггеров представлены на рис. 6.8.

Рис. 6.8. Условные графические изображения асинхронного (*а*) и синхронного (*б*) *Т*-триггеров

Таблица 6.9

T	Q_t	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 6.10

C	T	Q_{t+1}
0	0	Q_t
0	1	Q_t
1	0	Q_t
1	1	$\overline{Q}_t$

В табл. 6.9 и 6.10 представлены соответственно таблицы переходов асинхронного и синхронного *Т*-триггеров.

2.5. JK-триггеры

Триггеры этого типа отличаются от *RS*-триггеров тем, что при значениях входной информации, запрещенной для *RS*-триггеров, они инвертируют хранимую в них информацию.

JK-триггеры могут быть асинхронными и синхронными. Промышленностью выпускаются только синхронные триггеры.

Функция переходов синхронного *JK*-триггера при изменении входных сигналов задана таблицей истинности (табл. 6.11), а условное графическое обозначение синхронного *JK*-триггера приведено на рис. 6.9.

Так же как вход *S* у *RS*-триггера вход *J* у *JK*-триггера является входом установки триггера в единичное состояние, а вход *K* – входом установки триггера в нулевое состояние (аналогічної входу *R* у *RS*-триггера).

Таблица 6.11

C	J	K	Q_{t+1}	Режим
0	0	0	Q_t	Хранение
0	0	1	Q_t	
0	1	0	Q_t	
0	1	1	Q_t	
1	0	0	Q_t	Зранене
1	0	1	0	Установка "0"
1	1	0	1	Установка "1"

1	1	1	$\overline{Q}_t$	Т-триггер

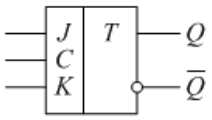


Рис. 6.9. Условное графическое изображение синхронного *JK*-триггера

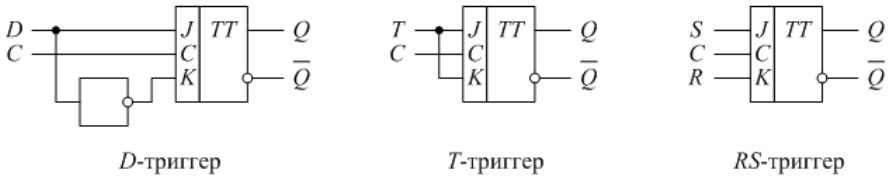


Рис. 6.10. Способы использования *JK*-триггера

JK-триггер является универсальным триггером и удобен тем, что при различных вариантах подключения его входов можно получить схему, функционирующие как *RS*-, *D*- и *T*-триггеры (рис. 6.10).

3. Лекция. Синтез цифровых автоматов.

3.1. Эффект гонок в автоматах.

Переключение автомата в следующее состояние производится сигналами возбуждения, вырабатываемыми в комбинационной части автомата и поступающими на входы триггеров запоминающей части. Значения сигналов возбуждения зависят от значений входных сигналов и состояний триггеров запоминающей части. Сигналы возбуждения вырабатываются в различных цепях комбинационной схемы. Эти цепи, различающиеся по количеству используемых в них логических элементов, характеризуются различным временем задержки сигнала. В результате разброса временных характеристик цепей сигналы возбуждения поступают на входы триггеров с некоторым разбросом во времени. К тому же триггеры имеют различное время переключения. Поэтому триггеры изменяют свои состояния неодновременно. Вследствие этого между триггерами могут начаться "гонки". Тот триггер, который выиграет гонки, т.е. изменит своё состояние раньше, чем другие, может при определённых условиях через цепь обратной связи изменить сигналы на входах остальных триггеров раньше, чем триггеры изменят своё состояние. В результате гонок триггеры могут перейти в состояние, не соответствующее закону функционирования автомата. Таким образом, гонки могут приводить к неправильным переходам.

Наличие гонок в автомате связано со структурными особенностями схем, вырабатывающих сигналы возбуждения, и зависит от вида используемых сигналов и от способа кодирования состояний.

Гонкам подвержены как автоматы, построенные на асинхронных триггерах, так и автоматы на синхронных триггерах, в которых переключение возможно в любой момент при наличии разрешающего потенциала на входе синхронизации. На рис.6.11 синхроимпульсы показаны схематически. Реально же они имеют некоторую длительность. Потенциальный триггер за время, равное длительности синхроимпульса может переключиться несколько раз.

Гонки в автомате могут быть устранены различными способами:

импульсная синхронизация;

использование триггеров, переключающихся по фронту синхроимпульса;

использование двухступенчатой памяти;

противогоночное кодирование.

При использовании импульсной синхронизации длительность синхроимпульса выбирается такой, чтобы за время действия синхроимпульса было возможно только одно переключение триггеров. Недостатком этого способа является то, что с течением времени или при изменении температуры окружающей среды параметры элементов могут изменяться. Это приводит к изменению времени цикла работы автомата и рассогласованию с установленной ранее длительностью синхроимпульса.

Для триггеров переключающихся по фронту, изменение состояния возможно лишь в момент поступления на синхровход либо нарастающего, либо спадающего фронта синхроимпульса.

3.2. Синтез автомата Мура.

Синтез автомата Мура рассмотрим на примере синтеза синхронного двоичного реверсивного счётчика на RS-триггерах, имеющего модуль счёта $M = 3$.

Счётчик является конечным автоматом Мура, поскольку выходной сигнал определяется только внутренним состоянием автомата $y(t) = \varphi(a(t))$. В общем случае конечный автомат строится на базе элементарных автоматов (триггеров), комбинационной части, вырабатывающей сигналы возбуждения триггеров, под действием которых автомат переходит в следующее состояние и комбинационной части вырабатывающей выходные сигналы автомата. Для счётчика комбинационная часть, вырабатывающая выходные сигналы, отсутствует, т.к. выходные сигналы снимаются с выходов триггеров.

Модуль счёта M определяет число различных выходных сигналов счётчика, и следовательно, число внутренних состояний. С приходом каждого синхроимпульса счётчик переходит в следующее состояние (из $a(t)$ в $a(t+1)$) и выдает соответствующий выходной сигнал, т.е. счётчик подсчитывает поступающие на его вход синхроимпульсы. Так как модуль счёта равен 3, то имеем три внутренних состояния и три выходных сигнала, соответствующие двоичным числам 0, 1 и 2.

Поскольку требуется построить реверсивный счётчик, то для управления режимом его работы необходим входной сигнал x. В зависимости от значения x счётчик будет работать либо в суммирующем, либо в вычитающем режиме. Пусть, когда $x = 0$, счётчик работает в режиме суммирования и выдает на выход циклически повторяющиеся сигналы (0, 1, 2, 0, 1, 2, 0, 1, ...), изменяющиеся с приходом каждого синхроимпульса. Если $x = 1$ - счётчик должен считать в обратную сторону (2, 1, 0, 2, 1, 0, 2, 1, ...), т.е. должен работать в вычитающем режиме.

Временная диаграмма работы реверсивного счётчика с модулем счёта $M = 3$ приведена на рис.6.11.

На рис.6.11 отмечены состояния автомата (a_0, a_1 и a_2), соответствующие выходным сигналам (0, 1 и 2).

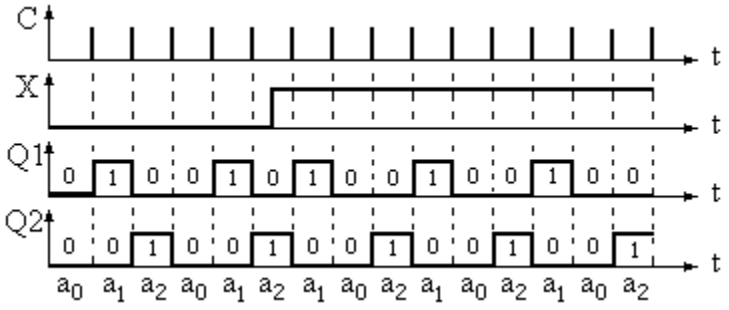


Рис.6.11. Временная диаграмма работы реверсивного счётчика с модулем счёта $M = 3$

По числу внутренних состояний (в данном случае их 3), пользуясь следующим соотношением, определим требуемое число элементарных автоматов:

$$n = \lceil log_2 N \rceil,$$

т.е. требуемое количество триггеров равно ближайшему большему целому от двоичного логарифма числа внутренних состояний *N*.

Для трёх внутренних состояний ближайшее большее целое из $log_2 3$ равно *2*. Таким образом, для реализации счётчика потребуется два RS-триггера.

Обозначив состояния триггеров как *Q2* и *Q1* (*Q2* - состояние старшего триггера, *Q1* - младшего), закодируем внутренние состояния счётчика. Кодировка состояний представлена в табл.6.12.

Выходные сигналы совпадают с внутренними состояниями: $y_0 = a_0$; $y_1 = a_1$; $y_2 = a_2$.

Отмеченная таблица переходов счётчика и граф переходов и выходов приведены соответственно в табл.6.13 и на рис.6.12.

Таблица 6.12

a(t)	*Q2*	*Q1*
a_0	0	0
a_1	0	1
a_2	1	0

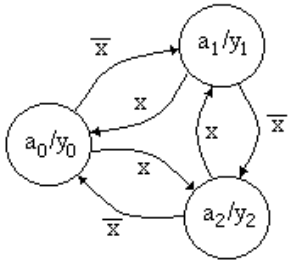


Рис.6.12. Граф переходов и выходов счётчика

Комбинационная схема, обеспечивающая выработку сигналов возбуждения триггеров, строится на основе структурной таблицы автомата Мура (табл.6.14).

Таблица 6.13

Вх. сигнал \ Вн. сост.	y_0	y_1	y_2
	a_0	a_1	a_2
$\bar{x}$	a_1	a_2	a_0
x	a_2	a_0	a_1

Таблица 6.14

Пе-ре-ход	Исх. сост.	Код исх сост.		След. сост.	Код след. сост.		Вх. сиг-нал	Сигналы возбуждения			
		Q2	*Q1*		*Q2*	*Q1*		*R2*	*S2*	*R1*	*S1*
1	a_0	0	0	a_1	0	1	$\bar{x}$	*	0	0	1
2	a_1	0	1	a_2	1	0	$\bar{x}$	0	1	1	0
3	a_2	1	0	a_0	0	0	$\bar{x}$	1	0	*	0
4	a_0	0	0	a_2	1	0	x	0	1	*	0
5	a_1	0	1	a_0	0	0	x	*	0	1	0
	a_2	1	0	a_1	0	1	x	1	0	0	1

Заполняется структурная таблица следующим образом. Например, для заполнения строки, соответствующей первому переходу из исходного состояния a_0 в следующее состояние a_1, в столбцы, соответствующие коду исходного состояния и коду следующего

состояния, записываются коды состояний из табл.6.12. В столбце "Входной сигнал" - значение входного сигнала $\bar{x}$, соответствующего этому переходу. Для того, чтобы обеспечить переход старшего триггера из состояния $Q2 = 0$ в состояние $Q2 = 0$, на входы R и S триггера необходимо подать комбинацию $R2 = 0$, $S2 = 0$; либо комбинацию $R2 = 1$; $S2 = 0$. Следовательно, для того, чтобы сохранить нулевое состояние триггера, в столбце "$S2$" необходимо записать 0, а в столбце "$R2$" - *, т.к. данный переход не зависит от значения сигнала на входе $R2$. Для перехода младшего триггера из состояния $Q1 = 0$ в состояние $Q1 = 1$ на входы младшего RS-триггера необходимо подать сигналы $R1 = 0$, а $S1 = 1$.

В строке, соответствующей шестому переходу из состояния a_2 в состояние a_1, по сигналу x старший триггер переключается из единичного состояния в нулевое, а младший - из нулевого в единичное. Для выполнения этих переходов на входы надо подать следующие сигналы: $R2 = 1$, $S2 = 0$, $R1 = 0$, $S1 = 1$.

Подобным же образом заполняются и все остальные строки структурной таблицы.

После заполнения структурной таблицы составляются и минимизируются функции возбуждения. Так как следующее состояние зависит от текущего состояния и входного сигнала, (см. соотношения 6.2), то в общем виде функция возбуждения выглядит так:

$$F = [\textit{исх. сост.1}]\cdot[\textit{вх. сигн.1}]+[\textit{исх. сост.2}]\cdot[\textit{вх. сигн.2}]+ \ldots \ . \qquad (6.3)$$

В качестве исходных состояний и входных сигналов выбираются те строки структурной таблицы, для которых значение функции равно 1. Так, для функции $R2$ можно записать:

$$R2 = a_2\cdot\bar{x} + a_2\cdot x \ .$$

Аналогично составляются остальные функции:

$$S2 = a_1\cdot\bar{x} + a_0\cdot x \ ;$$
$$R1 = a_1\cdot\bar{x} + a_1\cdot x \ ;$$
$$S1 = a_0\cdot\bar{x} + a_2\cdot x \ .$$

Раскодировав полученные соотношения, получим:

$$R2 = Q2\cdot\overline{Q1}\cdot\bar{x} + Q2\cdot\overline{Q1}\cdot x \ ;$$
$$S2 = \overline{Q2}\cdot Q1\cdot\bar{x} + \overline{Q2}\cdot\overline{Q1}\cdot x \ ;$$
$$R1 = \overline{Q2}\cdot Q1\cdot\bar{x} + \overline{Q2}\cdot Q1\cdot x \ ;$$
$$S1 = \overline{Q2}\cdot\overline{Q1}\cdot\bar{x} + Q2\cdot\overline{Q1}\cdot x \ .$$

Теперь для составления комбинационной схемы, имеющей меньшие аппаратурные затраты, необходимо минимизировать раскодированные функции возбуждения. Для минимизации можно использовать любой из вышерассмотренных методов минимизации. Воспользуемся методом карт Карно. Для этого по исходным состояниям триггеров и входному сигналу составляются карты для всех четырёх функций возбуждения. Однако несмотря на то, что имеются всего три состояния (a_0, a_1, и a_2), на двух триггерах можно закодировать четыре состояния и, следовательно, в карте Карно будут клетки соответствующие этим несуществующим состояниям (клетки соответствующие комбинациям $Q\cdot Q1\cdot\bar{x}$ и $Q2\cdot Q1\cdot x$). Значения функций для этих комбинаций не определены и для получения минимальной формы записи в эти клетки можно подставить как 0, так и 1. Карты Карно для функций $R2$, $S2$, $R1$ и $S1$ показаны на рис.6.13.

Итак, после минимизации получены следующие соотношения:

$$R2 = Q2 \ ; \qquad S2 = Q1\cdot\bar{x} + \overline{Q2}\cdot\overline{Q1}\cdot x \ ;$$

$$R1 = Q1\ ; \qquad S1 = Q2 \cdot x + \overline{Q2} \cdot \overline{Q1} \cdot \overline{x}\ .$$

Анализируя эти соотношения заметим, что в функциях *S2* и *S1* во вторых минтермах имеется одинаковый множитель $\overline{Q2} \cdot \overline{Q1}$, который можно реализовать на одном логическом элементе.

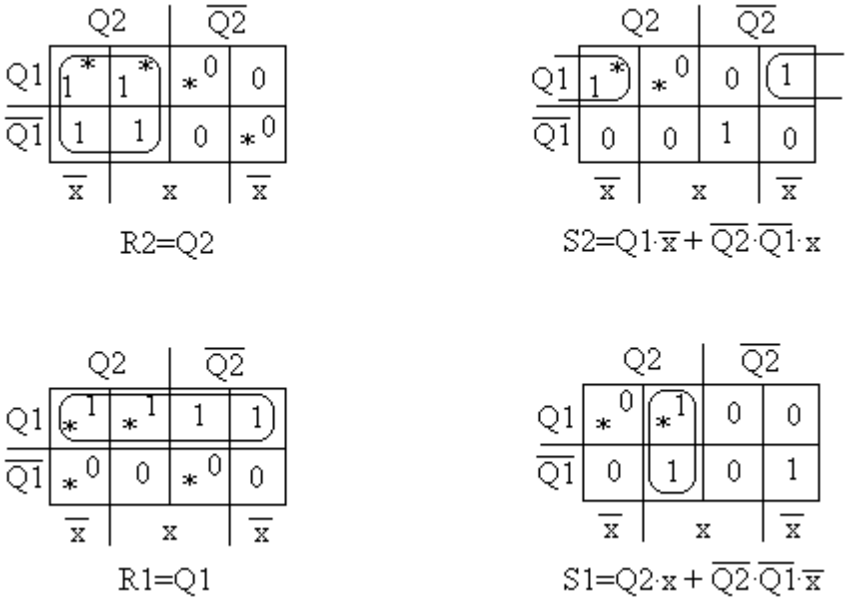


Рис.6.13. Карты Карно для функций возбуждения *R2, S2, R1, S1*

Схема двоичного реверсивного счётчика с модулем счёта *M = 3*, построенная по полученным соотношениям, приведена на рис.6.14.

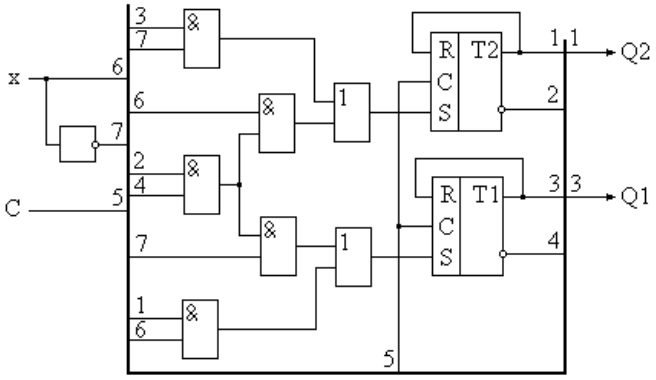


Рис.6.14. Функциональная схема двоичного реверсивного синхронного счётчика на RS-триггерах, имеющего модуль счёта *M = 3*

3.3. Синтез автомата Мили.

Синтез автомата Мили рассмотрим на примере синтеза распределителя импульсов (РИ) на D-триггерах с использованием соседнего кодирования.

Законы функционирования автомата Мили отличаются от законов функционирования автомата Мура тем, что выходной сигнал автомата Мили зависит не только от внутреннего состояния автомата, но и от входного сигнала. Следовательно, основное отличие синтеза автомата Мили заключается в том, что для формирования выходных сигналов должна быть синтезирована комбинационная схема, формирующая эти сигналы. Временная диаграмма работы, по которой необходимо синтезировать автомат, приведена на рис.6.16.

Для определения числа внутренних состояний и числа элементарных автоматов необходимо на временной диаграмме выделить циклы работы автомата для различных наборов входных сигналов. Для временной диаграммы на рис.6.16 выделены следующие циклы работы:

для входного сигнала $x = 0$ цикл состоит из четырёх тактов работы автомата, в каждом из которых автомат находится в одном из своих состояний a_0, a_1, a_2 и a_3 ;

для входного сигнала $x = 1$ цикл работы автомата содержит три такта и, соответственно, три состояния a_0, a_1, a_2.

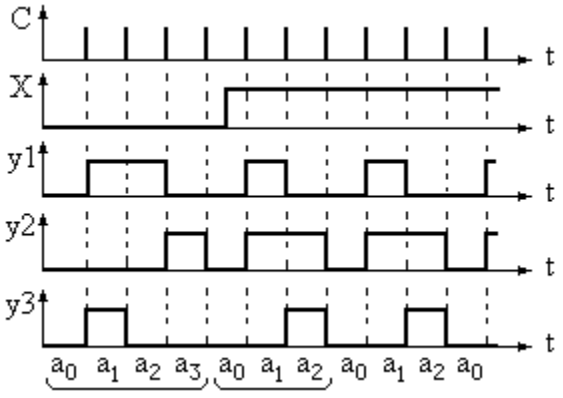


Рис.6.16. Временная диаграмма работы распределителя импульсов

Число элементарных автоматов (D-триггеров в данном случае) определяется по формуле

$$n = \lceil log_2(max\ N_i) \rceil ,$$

где N_i - число внутренних состояний для различных наборов входных сигналов. В данном случае $N_1 = 4$ (при $x = 0$), а $N_2 = 3$ (при $x = 1$) и, следовательно $n = 2$.

Обозначив состояния триггеров *Q2* и *Q1*, закодируем внутренние состояния с использованием соседнего кодирования состояний. Поскольку во втором цикле (для $x = 1$) число состояний нечётно, в этот цикл необходимо ввести дополнительное пустое состояние a_3, для которого выходные сигналы *y1*, *y2* и *y3* равны нулю.

Таблица 6.16

a(t)	*Q2*	*Q1*
a_0	0	0
a_1	0	1
a_2	1	1
a_3	1	0

В результате реальная временная диаграмма будет выглядеть, как показано на рис.6.17.

Соседнее кодирование состояний автомата представлено в табл.6.16.

Табл.6.17 представляет собой совмещенную таблицу переходов и выходов РИ.

Граф переходов и выходов РИ приведен на рис.6.18.

Комбинационные схемы, обеспечивающие выработку сигналов возбуждения триггеров и выходных сигналов, строятся на основе структурной таблицы автомата Мили (табл.6.18).

Заполняется структурная таблица автомата Мили подобно структурной таблице автомата Мура. Отличие состоит в том, что в структурной таблице автомата Мили добавлена колонка выходного сигнала.

В этой колонке записываются выходные сигналы, соответствующие исходным состояниям.

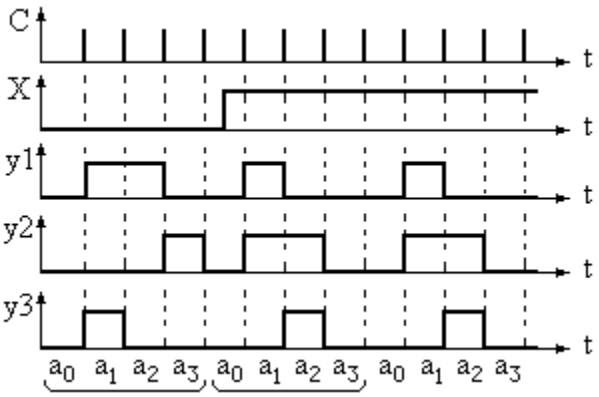


Рис.6.17. Временная диаграмма работы РИ
с соседним кодированием состояний

Вх. сигнал \ Вн. сост.	a_0	a_1	a_2	a_3
$\overline{x}$	$a_1/-$	$a_2/y_1,y_3$	a_3/y_1	a_0/y_2
x	$a_1/-$	$a_2/y_1,y_2$	$a_3/y_2,y_3$	$a_0/-$

Таблица 6.17

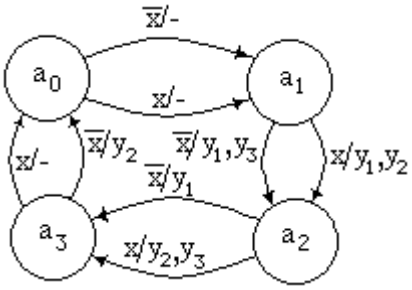


Рис.6.18. Граф переходов и выходов распределителя импульсов

На выходе D-триггера появляется то, что подано на его вход, поэтому в колонки структурной таблицы, соответствующие сигналам возбуждения D-триггеров, переписываются коды *Q2* и *Q1* следующих состояний автомата.

Таблица 6.18

Пе-ре-ход	Исх. сост.	Код исх. сост.		След. сост.	Код след. сост.		Вх. сиг-нал	Вых. сиг-нал	Сигналы возбуждения	
		Q2	*Q1*		*Q2*	*Q1*			*D2*	*D1*
1	a_0	0	0	a_1	0	1	$\overline{x}$	-	0	1
2	a_1	0	1	a_2	1	1	$\overline{x}$	*y1,y3*	1	1
3	a_2	1	1	a_3	1	0	$\overline{x}$	*y1*	1	0
4	a_3	1	0	a_0	0	0	$\overline{x}$	*y2*	0	0
5	a_0	0	0	a_1	0	1	x	-	0	1
6	a_1	0	1	a_2	1	1	x	*y1,y2*	1	1
7	a_2	1	1	a_3	1	0	x	*y2,y3*	1	0
8	a_3	1	0	a_0	0	0	x	-	0	0

Функции возбуждения триггеров автомата Мили составляются аналогично функциям возбуждения автомата Мура (6.3):

$$D2 = a_1 \cdot \bar{x} + a_2 \cdot \bar{x} + a_1 \cdot x + a_2 \cdot x \ ;$$
$$D1 = a_0 \cdot \bar{x} + a_1 \cdot \bar{x} + a_0 \cdot x + a_1 \cdot x \ .$$

Раскодировав эти соотношения и минимизировав их, получим:

$$D2 = \mathrm{Q1} \ ; \qquad D1 = \overline{Q2} \ .$$

Функции выходов автомата Мили строятся согласно соотношению, аналогичному соотношению (6.3):

$$\Phi = [\textit{исх. сост.1}] \cdot [\textit{вх. сигн.1}] + [\textit{исх. сост.2}] \cdot [\textit{вх. сигн.2}] + \ldots \ . \qquad (6.4)$$

Таким образом, для автомата Мили будем иметь:

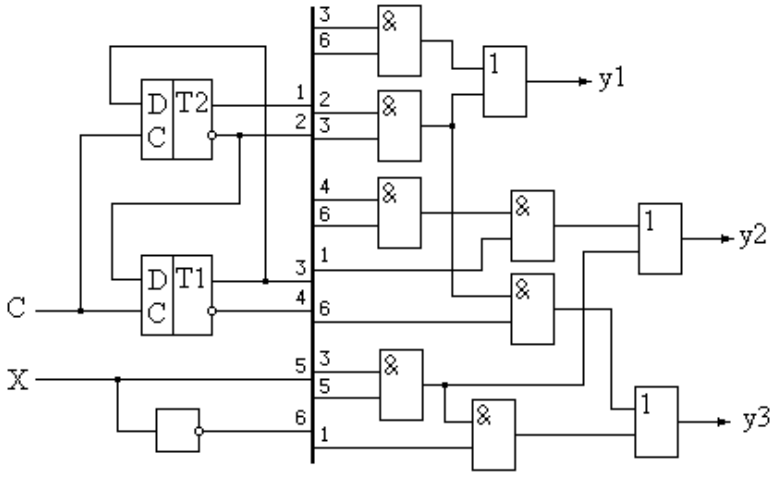
$$y1 = a_1 \cdot \bar{x} + a_2 \cdot \bar{x} + a_1 \cdot x \ ;$$
$$y2 = a_3 \cdot \bar{x} + a_1 \cdot x + a_2 \cdot x \ ;$$
$$y3 = a_1 \cdot \bar{x} + a_2 \cdot x \ .$$

Раскодировав эти соотношения и минимизировав их, получим

$$y1 = Q1 \cdot \bar{x} + \overline{Q2} \cdot Q1 \ ;$$
$$y2 = Q2 \cdot \overline{Q1} \cdot \bar{x} + Q1 \cdot x \ ;$$
$$y3 = \overline{Q2} \cdot Q1 \cdot \bar{x} + Q2 \cdot Q1 \cdot x \ .$$

Анализируя функции выходных сигналов заметим, что второй минтерм функции *y1,* равный $\overline{Q2} \cdot Q1$ входит в первый минтерм функции *y3* в качестве сомножителя, а второй минтерм функции *y2,* равный $Q1 \cdot x$, входит во второй минтерм функции *y3* в качестве сомножителя.

Учитывая эти соображения построили функциональную схему распределителя импульсов (рис.6.19), реализованного как автомат Мили.



4. Практическое занятие. Синтез автоматов.
4.1. Синтез автомата Мили.
4.2. Синтез автомата Мура.

5. Практическое занятие. Абстрактный синтез управляющего автомата.
5.1. Получение отмеченной ГСА.
5.2. Построение графа-переходов или функционирования автомата.
5.3. Построение таблицы переходов и выходных функций.

6. Практическое занятие. Структурный синтез управляющего автомата.
6.1. Кодирование внутренних состояний.
6.2. Формирование структурной таблицы автомата.
6.3. Формирование функций возбуждения и выходов.
6.4. Построение функциональной схемы управляющего автомата.

1.5. Тема 5: «Концепция операционного и управляющего автоматов» (28 часов).

Перечень и краткое содержание рассматриваемых вопросов.

1. Лекция. Управляющий и операционный автоматы.

1.1. Принцип микропрограммного управления.

Процессор ЦВМ или другое операционное устройство, выполняющее операции над словами информации, можно разделить на две части – операционный (ОА) и управляющий (УА) автоматы (рис. 8.1).

Процессор ЦВМ выполняет заданное множество операций *F* над входными словами *D* с целью вычисления результатов *R*. Каждая операция из множества операций *F* возбуждается соответствующей командой из множества команд *K*. Например, операция "СЛОЖЕНИЕ ЧИСЕЛ С ПЛАВАЮЩЕЙ ТОЧКОЙ" инициализируется командой "ВЫПОЛНИТЬ ОПЕРАЦИЮ СЛОЖЕНИЯ С ПЛАВАЮЩЕЙ ТОЧКОЙ".

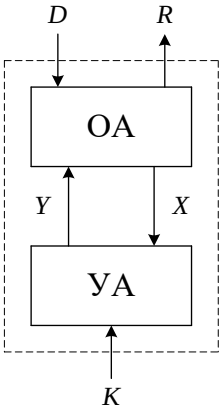


Рис. 8.1. Структура процессора ЦВМ

Выполнение команд организуется на основе принцип микропрограммного управления, который состоит в следующем.

1. Любая операция из множества *F*, реализуемая устройством, рассматривается как сложное действие и разделяется на последовательность элементарных действий над словами информации, называемых микрооперациями. Например: микрооперации "ЗАПИСЬ СЛОВА В РЕГИСТР", "ИНВЕРТИРОВАНИЕ СОДЕРЖИМОГО РЕГИСТРА", "ПЕРЕСЫЛКА ИНФОРМАЦИИ" и т. п.

Каждая микрооперация инициализируется соответствующей микрокомандой из множества мікрокомандо *Y*, вырабатываемых УА. Например: "ВЫПОЛНИТЬ ЗАПИСЬ СЛОВА В РЕГИСТР", "ПРОИНВЕРТИРОВАТЬ СОДЕРЖИМОЕ РЕГИСТРА" и т. д.

2. Для управления порядком следования микроопераций (порядком выдачи управляющим автоматом микрокоманд *Y*) используются логические условия *X*, принимающие значения 1 или 0.

3. Процесс выполнения операций в устройстве описывается в форме алгоритма, представляемого в терминах микроопераций и логических условий и называемого микропрограммой. Микропрограмма определяет порядок проверки логических условий *X* и следования микроопераций, необходимый для получения результата.

Таким образом, УА генерирует последовательность управляющих сигналов *Y*, предписанную микропрограммой и соответствующую значениям логических условий *X*.

Управляющий автомат можно реализовать как автомат с жесткой логикой или как автомат с программируемой логикой.

Формальная запись микропрограммы. Микропрограммный автомат с жесткой логикой. Микропрограммный автомат с программируемой логикой. Блок микропрограммного управления. Операционный автомат.

Структурный базис, принципы функционирования и характеристики операционного

автомата. Микрооперации. Типовые структуры операционных автоматов.

1.2. Граф-схема микропрограммы.

Структура микропрограммы проявляется наиболее чётко, когда микропрограмма представляется в графической форме. Граф-схема микропрограммы (ГСМ) строится с использованием вершин четырёх типов (рис. 8.2) и дуг, связывающих вершины.

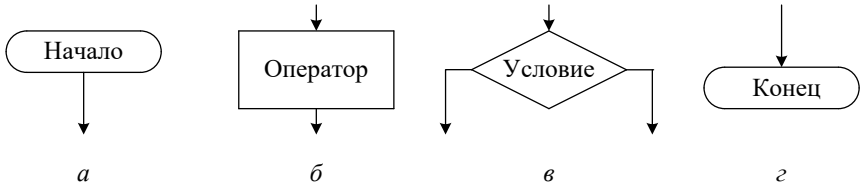


Рис. 8.2. Вершины ГСМ: начальная (*а*), операторная (*б*), условная (*в*) конечная (*г*)

Начальная вершина отмечает начало алгоритма и имеет единственный выход, из которого исходит дуга к первой выполняемой вершине графа.

Операторная вершина определяет действие – микрооперацию, совокупность совместимых микроопераций. В операторную вершину может входить любое, не меньшее одной, число дуг, и из её вершины выходит только одна дуга.

Условная вершина используется для разветвления вычислительного процесса в одном из двух возможных направлений в зависимости от значения проверяемого логического условия. В условную вершину может входить любое число дуг, но выходят всегда две дуги.

Конечная вершина отмечает конец микропрограммы. В конечную вершину может входить любое число дуг.

ГСМ считается корректной, если выполняются следующие условия:

1) ГСМ содержит конечное число вершин, каждая из которых принадлежит к перечисленным типам;

2) имеет одну начальную и одну конечную вершины;

3) выходы и входы вершин соединяются с помощью дуг, направленных от выхода ко входу;

4) каждый выход соединён с одним входом;

5) из любой вершины существует хотя бы один путь к конечной;

6) один из выходов условной вершины может соединяться с её входом, что недопустимо для операторной вершины;

7) в каждой условной вершине записывается один из входных сигналов УА;

8) в каждой операторной вершине записывается оператор (микро-команда из множества микрокоманд).

ГСМ делят на содержательные и закодированные. В содержательной ГСМ в условных и операторных вершинах записываются условия и операторы. Пример содержательной ГСМ представлен на рис. 8.3.

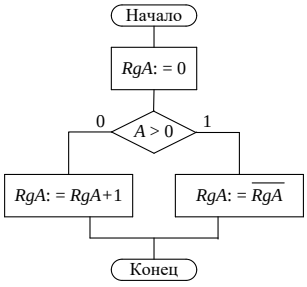


Рис. 8.4. Закодированная ГСМ

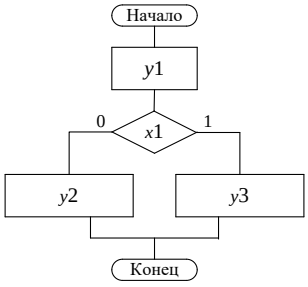


Рис. 8.3.Содержательная ГСМ

Закодированная ГСМ отличается от содержательной тем, что в условных и операторных вершинах записываются символы $x_1, \ldots, x_N$ и $y_1, \ldots, y_M$, отождествляемые с осведомительными и управляющими сигналами УА.

На рис. 8.4 представлена закодированная ГСМ, полученная на основе содержательной ГСМ рис. 8.3.

1.3. Концепция операционного автомата.

Операционным автоматом (ОА) называется устройство, предназначенное для выполнения множества операций *F* (рис. 8.5) над операндами, представляемыми множеством слов *D*, с целью вычисления слов *R*, определяющих значения результатов.

Функция операционного автомата определена следующей совокупностью сведений:

– множеством входных слов *D*, вводимых в автомат в качестве операндах;

– множеством выходных слов *R*, представляющих результаты операций;

– множеством внутренних слов *S*, используемых для представления информации в процессе выполнения операций (промежуточные результаты);

– множеством микроопераций *Y*, реализующих преобразование слов информации;

– множеством логических условий *X*.

Время не является аргументом функции операционного автомата. Функция устанавливает список действий – микроопераций и логических условий, которые может выполнять автомат, но не определяет порядок следования этих действий во времени. Порядок выполнения микроопераций ОА определяется граф-схемой микропрограммы, реализуемой управляющим автоматом.

Микрооперации, выполняемые ОА, подразделяют на следующие классы [14].

1. *Микрооперация установки* – присваивание слову значения константы.

2. *Микрооперация инвертирования* – обеспечение изменения значения слова на инверсное.

3. *Микрооперация передачи* – присваивание слову значения другого слова.

4. *Микрооперация сдвига* – изменение положения разрядов слова по отношению к начальному.

5. *Микрооперация счёта* – обеспечение изменения значения слова на единицу.

6. *Микрооперация сложения* – присваивание слову значение суммы слагаемых.

7. *Бинарные логические микрооперации* – присваивание слову значения, получаемого поразрядным применением операций конъюнкции, дизъюнкции и сложения по модулю 2 к парам соответствующих разрядов слагаемых.

8. *Комбинированные микрооперации* – это микрооперации, не принадлежащие ни к одному из вышеперечисленных классов. Комбинированные микрооперации содержат несколько действий, присущих микрооперациям разных классов. Например, сдвиг с одновременным инвертированием слова.

Одни из используемых в микропрограмме микроопераций могут выполняться параллельно во времени, другие – только последовательно. Свойство совокупности микроопераций, гарантирующее возможность их параллельного выполнения, называется совместимостью. Совместимость микроопераций обусловлена, во-первых, содержанием операторов, представляющих микрооперации, – *функциональная совместимость* – и, во-вторых, структурой ОА, допускающей или исключающей возможность параллельного выполнения нескольких микроопераций, – *структурная совместимость*. В функциональных микропрограммах, описывающих алгоритм выполнения операций безотносительно к структуре ОА, параллельно могут выполняться только те микрооперации, которые обладают свойством функциональной совместимости. Структура ОА вносит ограничения на количество параллельно выполняемых микроопераций. Поэтому возможность параллельного выполнения микроопераций должна определяться исходя из структурной совместимости. Таким образом, если структура ОА не определена,

то совместимыми называются функционально совместимые микрооперации. Если структура задана, то совместимыми называются структурно совместимые микрооперации.

Микрооперации называются *функционально совместимыми*, если они присваивают значения разным словам. Условием совместимости m микроопераций является совместимость каждой пары микроопераций.

Микрооперации называются *структурно несовместимыми*, если из-за ограничений, порождаемых структурой ОА, они не могут быть выполнены совместно – в одном такте автоматного времени. В противном случае микрооперации являются структурно совместимыми. Структурная несовместимость микроопераций связана с использованием микрооперациями общего оборудования, единственность которого исключает возможность совместного выполнения микроопераций.

Основными характеристиками процессоров, а, следовательно, и ОА (см. рис. 8.1) являются быстродействие и затраты оборудования. *Быстродействие устройства* $V=1/\vartheta$ определяется средним временем выполнения операций

$$\vartheta = \sum_{k=1}^{K} p_k \tau_k ,$$

где p_k – вероятность выполнения операции f_k; τ_k – среднее время выполнения операции.

Затраты оборудования в ОА оцениваются суммарной стоимостью элементов, составляющих автомат.

ОА строится на базе логических и запоминающих элементов. Чаще всего используются элементы базиса Шеффера (базис И-НЕ). В качестве запоминающих элементов используются регистры – для хранения слов информации и триггеры – для хранения двоичных переменных.

Наиболее существенное влияние на быстродействие ОА и затраты оборудования оказывает набор микроопераций Y и логических условий X.

Обобщенная структура операционного автомата.

ОА, предназначенные для выполнения арифметических операций (сложения, умножения, вычитания, деления) над числами, включают:

сумматоры, производящие суммирование кодов чисел;

регистры, служащие для приёма, хранения и выдачи чисел в сумматор или другое устройство и передачи их в другие блоки ЭВМ;

логические элементы (сдвигатели, вентили и др.), осуществляющие передачу чисел в соответствующем коде, в зависимости от выполняемых операций и знаков чисел, участвующих в них;

специальные устройства (умножители, делители и др.), предназначенные для ускорения выполнения арифметических операций.

На рис. 10.1 представлена обобщенная структура операционного автомата, предназначенного для выполнения арифметических операций.

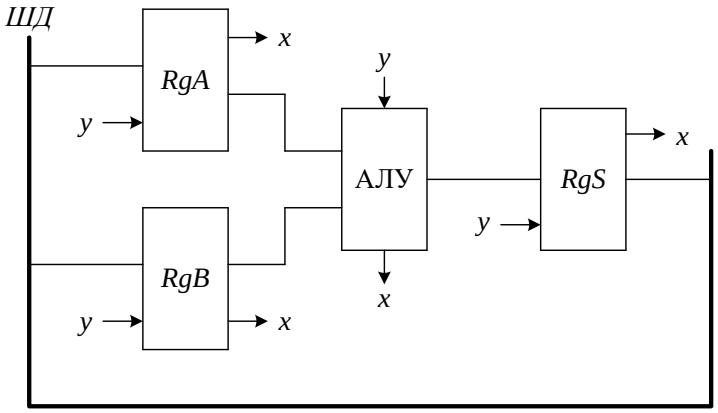


Рис. 10.1. Обобщенная структура ОА

Основу ОА составляют многоразрядное арифметико-логическое устройство (АЛУ) с параллельным вводом и выводом чисел, выполняющее операции суммирования, вычитания, умножения и деления, и три параллельных регистра – два входных и выходной.

Регистры служат не только для хранения, ввода и взводах чисел, но и для непосредственного участия наряду с АЛУ в выполнении таких операций, как деление, умножение и др. Поэтому в схемы регистров и АЛУ включаются цепи сдвигов чисел. При выполнении операций над числами с плавающей точкой часть разрядов арифметических устройств АЛУ и регистров может отводиться для записи порядка и его знака. С помощью соответствующей коммутации эти разряды образуют регистры и АЛУ порядка (в отличие от регистров и АЛУ чисел), над содержимым которых выполняются такие специфические действия, как сравнение, сложение, вычитание порядков и др.

Исходные операнды, над которыми будет выполняться арифметическая операция, поступают с шины данных (ШД) на входы параллельных регистров *RgA* и *RgB*, предназначенных для их хранения. Под действием управляющих сигналов *y*, выдаваемых УА, операнды записываются во входные регистры. Из ОА в УА поступают осведомительные сигналы *x*. В качестве сигналов *x* могут быть следующие сигналы: знаки операндов, отдельные разряды чисел, результаты логических операций над некоторыми разрядами чисел и т.п. УА выдаёт управляющий сигнал, под действием которого АЛУ производит арифметическую операцию. Результат выполнения операции записывается в параллельный регистр результата *RgS* и при наличии соответствующего управляющего сигнала появляется на шине данных.

АЛУ является основной частью ОА и предназначен для выполнения операций над кодами чисел. Обычно АЛУ представляет собой комбинационную схему параллельного действия и значение выдаваемого им результата определяется только комбинациями двоичных сигналов, поступающих на его входы в качестве операндов.

2. Практическое занятие. Синтез операционного автомата.

2.1. Выбор структурного базиса операционного автомата.

2.2. Синтез обобщенной струцтуры.

2.3. Синтез алгоритма работы операционного автомата.