НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
АГРАРНЫЙ УНИВЕРСИТЕТ

ИНЖЕНЕРНЫЙ ИНСТИТУТ

А.М. Сажнёв
И.С. Тырышкин

Цифровые устройства и микропроцессоры

Учебное пособие

Новосибирск
2015

УДК 681.325.5-181.8(07)

Составители: А.М. Сажнёв, канд. техн. наук, доц;

И.С. Тырышкин, канд. техн. наук, доц.

Рецензент: Г.М. Симаков, д-р техн. наук, проф. (НГТУ)

Цифровые устройства и микропроцессоры: учеб. пособие/А.М. Сажнев, И.С. Тырышкин; Новосиб. гос. аграр. ун-т. Инженер. ин-т. – Новосибирск: ИЦ «Золотой колос», 2015. – 119 с.

Содержит сведения о логических и арифметических основах цифровых устройств. Рассматриваются начала синтеза комбинационных и последовательных логических схем. Приводится схемотехника отдельных узлов, на основе которых базируются современные вычислители, микропроцессоры и микроконтроллеры, применяемые в устройствах автоматики. Приводятся сведения о микропроцессорных системах и их взаимодействии с внешними устройствами, а также вопросы для самоподготовки.

Рекомендовано к изучению по дисциплинам «Программируемые системы управления» и «Микропроцессорные устройства управления» для студентов очной и заочной форм обучения Инженерного института по направлению подготовки бакалавриата 35.03.06 – Агроинженерия (профиль Электрооборудование и электротехнологии в АПК).

Утверждено и рекомендовано к изданию учебно-методическим советом Инженерного института (протокол № 37 от 24 марта 2015 г.).

Новосибирский государственный
аграрный университет, 2015

Введение

Цифровые вычислительные машины прочно вошли в нашу жизнь и находят применение во все больших сферах человеческой деятельности. Они используются и для расчетов по заданным математическим формулам и для управления механизмами, устройствами и транспортом. При этом, управляемое устройство обычно может находиться в двух состояниях включено или выключено. Лампа светится или нет, электрический двигатель включен или нет, механизм работает или выключен. Использование двоичных сигналов позволило увеличить чувствительность и помехоустойчивость аппаратуры управления, позволяет хранить большие объёмы информации и менять алгоритмы обработки – перепрограммировать устройства.

В природе все процессы непрерывные (аналоговые). Аналоговый сигнал есть непрерывная функция непрерывного времени. Это значит, что в произвольный момент времени можно найти значение сигнала с любой точностью. Цифровые машины работают с цифровыми сигналами. Цифровой сигнал есть дискретная функция дискретного времени nT (рис.1), где n номер такта (n = 1,2,3,4,...), T - шаг дискретизации по времени.

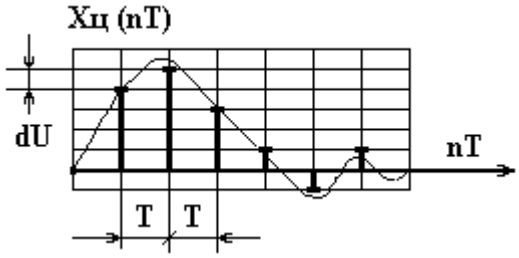


Рисунок 1 – Цифровой сигнал

Здесь сигнал принимает только фиксированные значения, определяемые шагом квантования по амплитуде dU (связано с разрядностью счета или точностью представления числа).

Быстродействие цифровых вычислительных машин определяется шагом дискретизации по времени, а точность представления сигнала – числом уровней квантования. Конечно, это недостатки цифровых вычислителей, но уже существуют машины с быстродействием до 100 млрд оп /с, а точность – до 40 десятичных разрядов, что достаточно для решения большинства технических задач, возлагаемых на вычислительную технику.

В зависимости от назначения все цифровые машины (электронные вычислительные машины - ЭВМ) условно делят на:

1) микрокалькуляторы;

2) микро ЭВМ – это вычислитель, встроенный в прибор, стиральную машину, автомобиль или другое устройство;

3) персональные компьютеры (ПК). Первые ПК появились в 1975 г., когда фирма APPLE выпустила их в количестве 575 шт. В 2001 г. каждая семья в США имела по два ПК. На их основе создают автоматизированные системы управления

(например, диспетчерские пульты или системы продажи билетов).

4) универсальные ЭВМ;

5) быстродействующие ЭВМ, на их основе строятся вычислительные центры (кустовые и региональные) для решения сложных управленческих задач. Например, гидрометеослужба, системы наблюдения за воздушным пространством, управление флотами и др.;

6) сверхбыстродействующие ЭВМ престижные ЭВМ для решения специальных задач.

Особую группу вычислительных устройств составляют микропроцессоры (МП). МП это большая интегральная схема (ИС) с программируемой логикой работы. МП самостоятельного значения не имеет (у него нет памяти и устройств ввода/ вывода информации). МП может работать только в составе микропроцессорной системы (МПС). Разрядность обрабатываемых данных составляет 4,8,10,12,16,32,64 бит (двоичных разрядов). Тактовая частота работы составляет единицы ГГц. На базе МП строятся любые современные вычислители и устройства логического управления - контроллеры.

1 ЛОГИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННО_ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

1.1 Основные понятия

Алгебра логики - один из раздел математической логики. Её создателем является англичанин Джорж Буль (1815 - 1864), поэтому алгебру логики называют булевой алгеброй.

Начальным понятием булевой алгебры является высказывание.

Высказывание - это некоторое предложение, о котором можно утверждать истинно оно или ложно. Высказывание обозначают буквой (идентификатором). Например, два высказывания;

X_1 = < Москва - столица России > $X_1 = 1$ – истина

X_2 = < Луна больше Земли > $X_2 = 0$ – ложь

Если высказывание истинно, то его обозначают единицей, если ложно – нулём.

Логическая переменная – некоторая переменная величина **X,** которая может принимать одно из двух значений 0 или 1, то есть быть ложной или истинной X={0,1}.

Логическая функция (булева функция, переключательная функция, функция алгебры логики) n переменных - это функция, которая может принимать одно из двух значений (0 или 1) на некотором наборе этих переменных $F(X_1, X_2,\ldots,X_n) = \{0,1\}$.

Логическая функция задается таблицей истинности.

Таблица истинности – это совокупность всех возможных наборов (комбинаций) логических переменных и значений функции на этих наборах.

Например, логические функции одной переменной n = 1 – тривиальные функции**.**

Таблица 1 – Логические функции одной переменной

F\X	0	1	Название функции
F_1	0	0	Const «0» - абсолютно ложная функция
F_2	0	1	Переменная икс - тождественная функция
F_3	1	0	«Не икс» - отрицание икс - инверсия икс
F_4	1	1	Const «1» - абсолютно истинная функция

Реализация этих функций показана на рисунке 1.1.

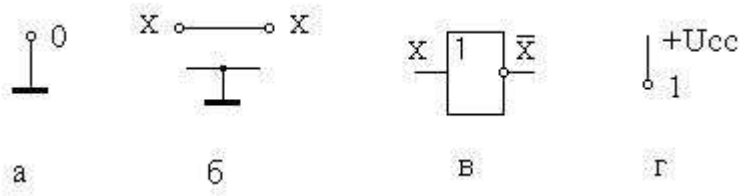


Рисунок 1.1 – Реализация функций одной переменной

Функция F_1 всегда ложна (рис.1.1а), функция F_2 есть сама переменная икс (рис 1.1б), функция F_3 реализуется инвертором (рис.1.1в), функция F_4 всегда истинна (рис.1.1г).

В общем случае, если имеем n независимых логических переменных, то можно составить $2^n = N$ наборов этих переменных, а так как на каждом из наборов функция может принимать значение 0 или 1 , то общее возможное число функций равно $L=2^N$. Так, при n = 1 число наборов N =2, а число функций L = 4.

Рассмотрим логические функции двух переменных n = 2. Они относятся к элементарным функциям. Число наборов переменных равно $N = 2^2 = 4$, а число функций L =16.

На практике имеют простую техническую реализацию и используются не все элементарные функции, а только основные (базисные) функции. Рассмотрим их.

1. Логическое умножение, операция **«И»** – конъюнкция. Выполняется элементом – конъюнктором (рис. 1.2).

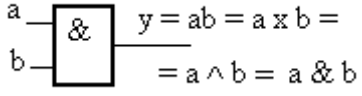


Рисунок 1.2 - Конъюнктор

Его таблица истинности

№\X	a	b	y
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Рисунок 1.3 – Таблица истинности конъюнктора

2. Операция Шеффера «**И – НЕ»** – отрицание конъюнкции. Выполняется элементом Шеффера (рис. 1.4).

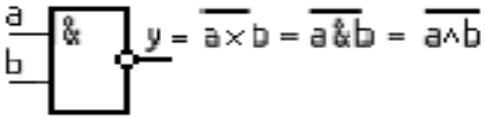


Рисунок 1.4 – Элемент Шеффера

Его таблица истинности

№\X	a	b	y
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

Рисунок 1.5 - Таблица истинности элемента Шеффера

3. Логическое сложение, операция «**ИЛИ»** – дизъюнкция. Выполняется элементом – дизъюнктором (рис.1.6).

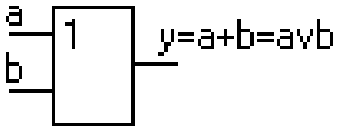


Рисунок 1.6 – Дизъюнктор

Его таблица истинности

№\X	a	b	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Рисунок 1.7 – Таблица истинности дизъюнктора

4. Операция Пирса - отрицание дизъюнкции. Логическое **«ИЛИ – НЕ».** Выполняется элементом Пирса (рис. 1.8).

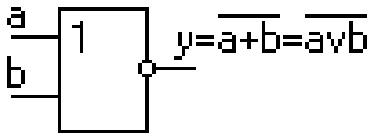


Рисунок 1.8 – Элемент Пирса

Его таблица истинности

№\X	a	b	y
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

Рисунок 1.9 – Таблица истинности элемента Пирса

5. Логическая неравнозначность или сумма по модулю два - **M2**.
Выполняется сумматором по «модулю два» (рис. 1.10). Функция истинна на тех наборах, где число единиц нечетно.

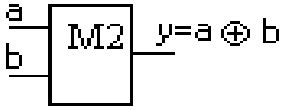


Рисунок 1.10 – Сумматор по модулю два

Его таблица истинности

№\X	a	b	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Рисунок 1.11 – Таблица истинности сумматора по модулю два

Вместе с тем, в литературе встречается функция, так называемая, «исключающее **ИЛИ»,** которая истинна, на тех наборах, где присутствует исключительно одна единица. Операция выполняется элементом «исключающее ИЛИ» (рис.1.11).

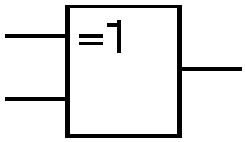


Рисунок 1.11 – Элемент «исключающее ИЛИ»

Его таблица истинности

№\X	a	b	y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Рисунок 1.12 – Таблица истинности элемента «исключающее ИЛИ»

Видно, что таблицы истинности совпадают. Значит для двух переменных функции М2 и =1 – эквивалентны.
Составим таблицу истинности этих функций при числе переменных n=3.

№	a	b	c	М2	=1
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	0
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	0

Рисунок 1.13 – Таблица истинности элементов М2 и =1
для трёх переменных

Видно, что они различаются в последнем наборе. При большем числе переменных это различие возрастает, поэтому функции М2 и =1 нельзя отождествлять.

Графическое изображение и условное обозначение логических элементов регламентируются ГОСТ 2.743-91 ЕСКД. Этот ГОСТ устанавливает следующие геометрические размеры (рис. 1.14).

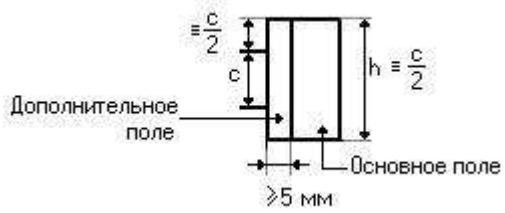


Рисунок 1.14 – Условное изображение логических элементов

Других ограничений на размеры логических элементов ГОСТ не накладывает.

1.2 Аксиомы и основные свойства алгебры логики

Основные свойства алгебры логики базируются на аксиомах и позволяют преобразовывать логические функции. Приведем здесь аксиомы и основные свойства алгебры логики без обсуждения и доказательств. Заметим, что некоторые свойства, в силу их важности, в технической литературе трактуются как законы.

АКСИОМЫ алгебры логики:

0 * 0 = 0	0 + 0 = 0
0 * 1 = 0	0 + 1 = 1
1 * 0 = 0	1 + 0 = 1
1 * 1 = 1	1 + 1 = 1

ЗАКОНЫ алгебры логики:

1. Закон одинарных элементов

1 * X = X	0 * X = 0
1 + X = 1	0 + X = X

2. Законы отрицания:

а) закон дополнительных элементов

$$X + \overline{X} = 1 \quad X * \overline{X} = 0$$

б) двойное отрицание

$$\bar{1} = 0 \quad \bar{0} = 1 \quad \bar{\bar{1}} = 1 \quad \bar{\bar{0}} = 0 \quad \overline{\overline{X}} = X \quad \overline{X} = \overline{X},$$

поэтому отрицание можно переносить из одной части равенства в другую;

в) закон двойственности (правило Моргана):

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

Отрицание дизъюнкции есть конъюнкция отрицаний и наоборот - отрицание конъюнкции есть дизъюнкция отрицаний:

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

Правило справедливо для любого числа переменных.

3. Комбинационные законы.

Они во многом соответствуют обычной алгебре, но есть и отличия:

а) тавтологии **(**многократное повторение**):**

$$X + X + X + X = X$$

$$X * X * X * X = X$$

б) переместительности:

A+B+C+D=A+C+B+D

в) сочетательности:

A+B+C+D=A+(B+C)+D=A+B+(C+D)

г) распределительности:

$X_1(X_2+X_3)= X_1X_2 + X_1X_3$

$X_1+X_2X_3=(X_1+X_2)(X_1+X_3)=$/ докажем это путём раскрытия скобок/=

$= X_1X_1+ X_1X_3+ X_1X_2+ X_2X_3= X_1(1+X_3+X_2)+ X_2X_3= X_1+X_2X_3$

д) правило поглощения (одна переменная поглощает другие**):**

$$X_1+X_1X_2\,X_3 = X_1(1+X_2\,X_3\,) = X_1$$

е) правило склеивания **(**выполняется только по одной переменной**):**

$$A\overline{B}C + ABC = AC(\overline{B} + B) = AC$$

Так же как в обычной математике имеется старшинство операций:

1) действие в скобках;
2) операция с одним операндом (одноместная операция) – НЕ;
3) конъюнкция И;
4) дизъюнкция ИЛИ;
5) сумма по модулю два.

Операции одного ранга выполняются слева направо в порядке написания.

1.3 Понятие базиса

С помощью ограниченного набора элементарных функций можно представить любую, сколь угодно сложную функцию алгебры логики. Такой набор элементарных функций называют **базисом или функционально полным набором.**

Базисов может быть много:

1. И, ИЛИ, НЕ
2. И, НЕ
3. И – НЕ
4. НЕ – И
5. ИЛИ, НЕ
6. ИЛИ – НЕ
7. НЕ – ИЛИ
8. «0», «1», НЕ, $\geq$n и другие

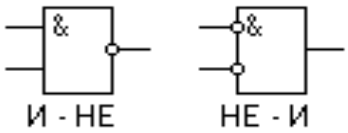


Рисунок 1.15 – Некоторые базисы

Мажоритарный элемент ($\geq$n) имеет нечетное число входов и вырабатывает 1, если число единиц на входе больше чем нулей (правило голосования).

Базис называется избыточным, если исключение одной элементарной функции не приводит к потере функциональной полноты. В противном случае базис называется минимальным. Так, базисы 1,8 – избыточные, а остальные – минимальные.

Используя законы алгебры логики, можно переходить от одного базиса к другому.

Например, пусть имеется элемент ЗИ-НЕ, а необходимо реализовать следующие операции:

1. НЕ;
2. И (для двух переменных);
3. ИЛИ (для двух переменных).

Реализуем эти операции.

1. Операция НЕ получается на основании закона тавтологии (рис.1.16).

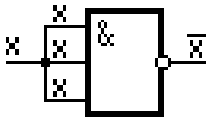


Рисунок 1.16 – Инвертор на элементе Шеффера

2. Операция И получается на основании законов тавтологии и двойного отрицания (рис. 1.17).

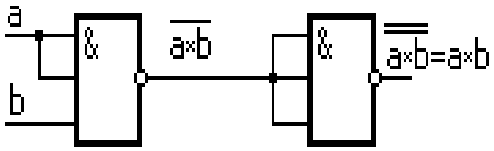


Рисунок 1.17 – Конъюнктор на элементах Шеффера

3. Операция ИЛИ получается на основании правила двойственности $a+b=\overline{\overline{a+b}}=\overline{\overline{a}*\overline{b}}$. Тогда получаем следующую реализацию (рис. 1.18).

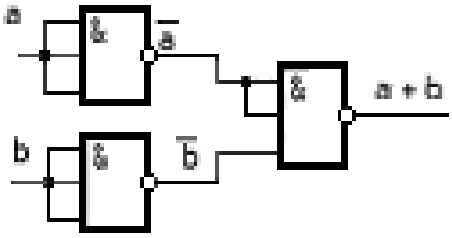


Рисунок 1.18 –Дизъюнктор на элементах Шеффера

1.4 Формы представления функций алгебры логики

Функции алгебры логики могут быть заданы различными способами:

- таблицей истинности;
- в аналитической форме;
- в числовой форме.

Таблица истинности уже была рассмотрена. В ней все наборы логических переменных следуют строго в порядке возрастания их двоичного номера и нумеруются целыми числами от 0 до 2^n-1, где n – число переменных функции.

Если функция имеет значения на всех наборах, то она называется полностью определенной.

При аналитической записи используются так называемые нормальные формы.

Для лучшего понимания материала введем некоторые понятия:

* терм - компонент выражения;

* ранг терма - число переменных, входящих в терм;

* элементарная дизъюнкция - дизъюнктивный терм или макстерм это дизъюнкция произвольного числа попарно независимых переменных. Например,

$(\bar{a}+\bar{b}+c+d)$ - **макстерм 4-го ранга**

$X_1+\overline{X_2}+\overline{X_3}$ - **макстерм 3-го ранга**

$(a+\bar{b}+c+\bar{a})$ – это не макстерм, т.к. переменные a и $\bar{a}$ попарно зависимые;

* элементарная конъюнкция конъюнктивный терм или минтерм - конъюнкция произвольного числа попарно независимых переменных. Например,

$X_1X_2X_3$ - минтерм 3-го ранга

$adc\bar{d}$ – это не минтерм, так как переменные d и $\bar{d}$ зависимые.

Для аналитической записи функций используют две формы:

1) дизъюнктивную нормальную форму – ДНФ;
2) конъюнктивную нормальную форму – КНФ.

ДНФ это дизъюнкция минтермов различного ранга:

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

КНФ это конъюнкция макстермов различного ранга:

$$f(X_1X_2X_3X_4)=(X_1+\overline{X_2}+X_3)(\overline{X_1}+\overline{X_2}+X_3+X_4)(X_1+X_2)$$

Если все термы, входящие в нормальную форму, имеют одинаковый и максимальный ранг, равный числу переменных функции n, то такая форма называется совершенной. При этом минтерм называется констинтуентой (составляющей) единицы (КЕ), а макстерм конституентой нуля (КН).

$F(a,b,c)=\bar{a}bc+abc+a\bar{b}c+ab\bar{c}$ - это СДНФ

$F(a,b,c,d)=(a+b+\bar{c}+d)(\bar{a}+b+\bar{c}+d)(\bar{a}+\bar{d}+\bar{c}+d)$ - это СКНФ

Таким образом, совершенная дизъюнктивная нормальная форма есть дизъюнкция конституент единицы, а СКНФ есть конъюнкция конституент нуля.

Совершенные формы составляются по таблице истинности функции. СДНФ составляется по такому правилу: для каждого набора переменных, на котором функция истинна, записывают минтерм ранга n , в котором с отрицанием берутся переменные, имеющие нулевые значения на данном наборе. Все минтермы объединяют дизъюнктивно.

Пусть, например, имеем произвольную функцию трёх переменных, заданную такой таблицей истинности (рис. 1.19).

№\X	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Рисунок 1.19 – Таблица истинности произвольной функции

Для номеров наборов N= 1, 3, 6 и 7 получаем следующую СДНФ:

$$f(a,b,c) = \bar{a}\bar{b}c + \bar{a}bc + ab\bar{c} + abc$$

СКНФ также записывают по таблице истинности по правилу: для каждого набора переменных, на котором функция ложна, записывают макстерм ранга n, в котором с отрицанием берутся переменные, имеющие единичные значения на данном наборе. Все макстермы объединяют конъюнктивно. Тогда для этой же функции, для номеров наборов N = 0, 2, 4 и 5, получаем СКНФ:

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

Очевидно, что СДНФ и СКНФ полностью дуальны.

Для компактной записи функций используют числовую форму, в которой задаются только номера наборов. Числовая форма для СДНФ:

$$f(a,b,c) = V(1,3,6,7)$$

Числовая форма для СКНФ:

$$f(a,b,c) = \Lambda(0,2,4,5)$$

1.5 Минимизация функций

1.5.1 Задача минимизации

Пусть требуется построить логическую схему, которая реализует следующую таблицу истинности (рис. 1.20)/

№\X	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Рисунок 1.20 – Таблица истинности для логической схемы

Для реализации схемы запишем СДНФ:

$$f(a,b,c) = \bar{a}\bar{b}c + \bar{a}bc + ab\bar{c} + abc$$

Каждый минтерм реализован своим конъюнктором. Инверсии выполнены на входах, чтобы не пользоваться дополнительными инверторами и не загромождать схему. Получаем (рис. 1.21):

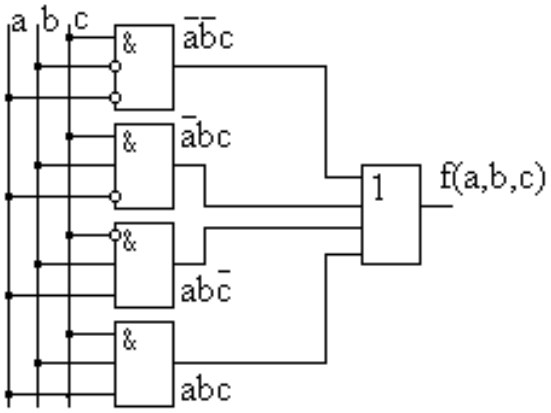


Рисунок 1.21 – Схемная реализация таблицы истинности (рис.1.20)

Сложность логической схемы принято оценивать общим числом входов всех ее элементов, которые условно называют **ценой схемы**. Инверторы не учитываются.

Найдём, чему равна цена нашей схемы:

$$Ц_1 = 12 + 4 = 16$$

Она складывается из числа входов конъюнкторов (элементы первого уровня) и числа входов дизъюнктора (элемент второго уровня).

Число входов элементов 1-го уровня равно числу символов в записи функции. Число входов элементов 2-го уровня равно числу термов в записи функции. То есть цену схемной реализации можно подсчитать сразу по исходной логической формуле.

Используя законы алгебры логики, попытаемся упростить исходную функцию. Для этого вынесем за скобки:

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

Очевидно, что реализация такой формулы значительно проще, ее цена $Ц_2 = 4+2 = 6$.

Логическая формула с наименьшим числом логических связей называется **минимальной.**

Таким образом, получаем минимальную дизъюнктивную нормальную форму (МДНФ) и, соответственно, МКНФ.

Процесс отыскания минимальной формы называется **минимизацией логической функции,** или просто минимизацией.

Минимизировать функции можно тремя методами:

1) расчетным путем, используя законы алгебры логики;

2) графическим путем (метод карт Карно или диаграмм Вейча), используя специальные карты;

3) расчетно-графическим путем (метод Квайна и его модификации).

Расчетный метод мы уже разобрали выше. Метод Квайна используется при числе переменных больше шести, хорошо алгоритмизируется и программируется. На его основе разработаны системы автоматизированного проектирования и различные стандартные программы минимизации логических функций любого числа переменных. Но они не всегда доступны и не оправданны при малом числе независимых переменных. Метод карт Карно хорошо работает при числе переменных меньше шести, прост и удобен для оперативного использования. Тем более, что большинство устройств, с которыми имеет дело разработчик, оперируют именно с малым числом переменных (3…5). Поэтому рассмотрим этот метод подробнее.

1.5.2 Метод карт Карно

Карта Карно (минимизирующая карта) – это развертка некоторой объемной фигуры на плоскости. Карта Карно состоит из клеток, число которых равно числу наборов переменных функции. Каждая клетка соответствует строго определённому набору.

Например, карта Карно одной переменной (рис 1.22):

n = 1, число наборов $N = 2^n = 2$.

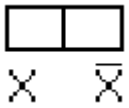


Рисунок 1.22 – Карта Карно одной переменной

Карта Карно двух переменных (рис.1.23): n = 2, число наборов $N = 2^2 = 4$.

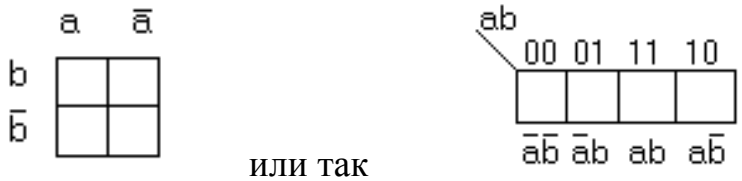


Рисунок 1.23 – Карта Карно двух переменных

Крайние клетки, соответствующие комбинациям 00 и 10 , являются соседними и отличаются одной переменной a.

Переменные в карте могут располагаться произвольно, но **любые соседние по вертикали или по горизонтали клетки могут отличаться не более чем одной переменной.**

Карта Карно трёх переменных (рис.1.24): n = 3, число наборов $N = 2^3 = 8$.

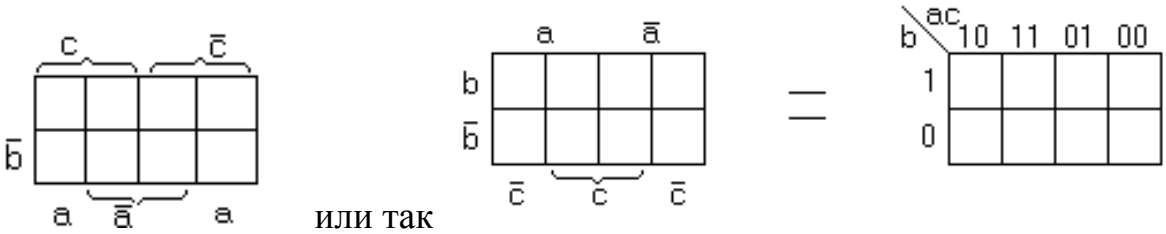


Рисунок 1.24 – Карта Карно трёх переменных

Если имеется функция трёх переменных, заданная следующей таблицей истинности (рис. 1.25):

№\X	A	B	C	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Рисунок 1.25 – Таблица истинности произвольной функции

Тогда соответствующая ей карта Карно выглядит так (рис. 1.26):

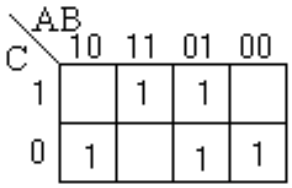


Рисунок 1.26 – Карта Карно функции рис. 1.25

Обычно нули в карту не пишут, а заносят только единицы.
Карта Карно четырёх переменных (рис. 1.27): n = 4, N = 16.

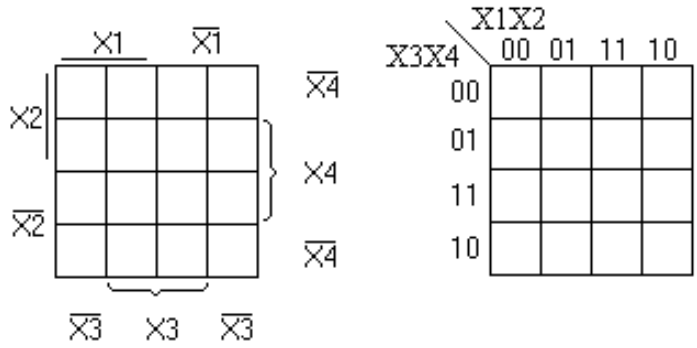


Рисунок 1.27 – Карта Карно функции четырёх переменных

В этих картах переменные расположены по-разному, но они обе правильны, так как любые соседние клетки по горизонтали или вертикали отличаются только одной из переменных. Клетки верхнего ряда соседние с клетками нижнего ряда, а клетки крайнего правого столбца соседние с клетками крайнего левого столбца.

Карта Карно пяти переменных (рис. 1.28): n = 5, N = 32. Это две шестнадцатиклеточных карты, отличающиеся только одной (пятой) переменной.

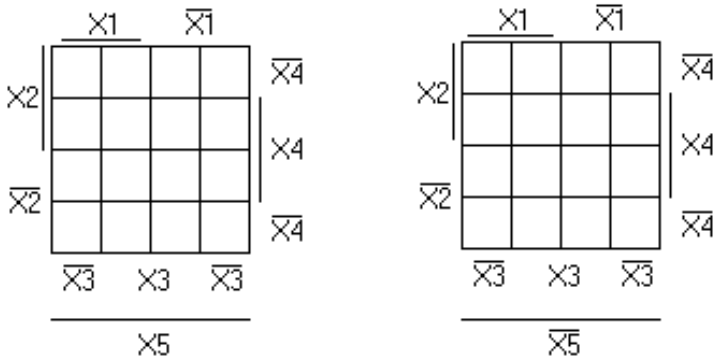


Рисунок 1.28 – Карта Карно функции пяти переменных

Карта Карно шести переменных (рис. 1.29): n = 6, N = 64. Это четыре шестнадцатиклеточных карты.

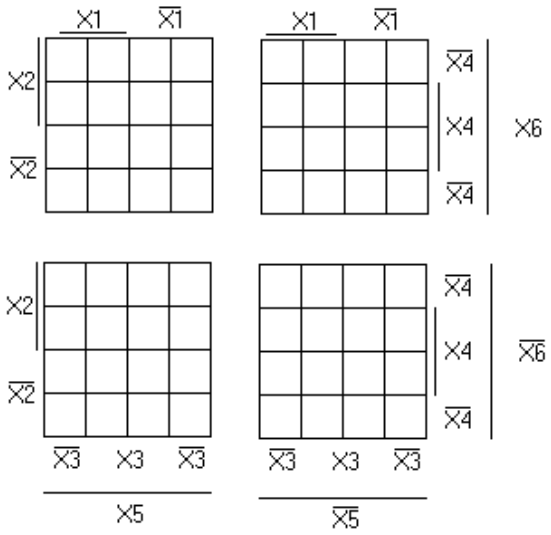


Рисунок 1.29 – Карта Карно функции шести переменных

Здесь любые клетки соседние, если они отличаются только одной переменной.

После заполнения карты единицами из таблицы истинности приступают к минимизации. **Суть минимизации: охватить все единицы карты Карно наименьшим числом кубов наиболее высокого ранга. Из каждого куба выписывают минтерм общих переменных. Минтермы объединяют дизъюнктивно.**

Куб – это прямоугольный или квадратный контур, содержащий клетки только с единицами:

одна единица – куб «0»-го ранга, так как $2^0 = 1$;
две единицы - куб «1»-го ранга, т.к. $2^1 = 2$;
четыре единицы - куб «2»-го ранга, т.к. $2^2 = 4$;
восемь единиц - куб «3»-го ранга, т.к. $2^3 = 8$;
шестнадцать единиц - куб «4»-го ранга, т.к. $2^4=16$ и т.д.

Куб не может содержать другое число единиц и клетки с нулями. Одна и та же единица одновременно может принадлежать нескольким кубам, чтобы ранг куба был наибольшим. Тогда форма будет именно минимальной.

Пример 1.

Найти МДНФ такой функции: F(a,b,c)=V(1,3,6,7) . Составим её таблицу истинности, которая приведена на рис. 1.30.

№\X	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Рисунок 1.30 – Таблица истинности функции

Заполняем карту Карно.

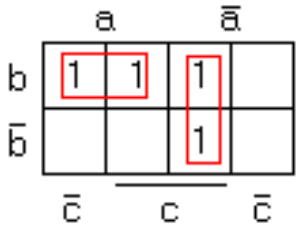


Рисунок 1.31 – Карта Карно функции рис. 1.30

Здесь следует провести два куба первого ранга и составить МДНФ:

$$f_{\min} = ab + \bar{a}c$$

Запишем СДНФ исходной функции и преобразуем её по законам алгебры логики $f = \bar{a}\bar{b}c + \bar{a}bc + ab\bar{c} + abc = \bar{a}c + ab$. Получим тот же результат.

Представим карту Карно этой же функции по-другому (рис.1.32)

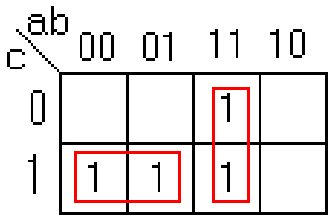


Рисунок 1.32 – Карта Карно функции рис. 1.30

Видно, что результат такой же: $f_{\min} = ab + \bar{a}c$.

Пример 2.

Минимизировать функцию трёх переменных: F(a,b,c)= ∧(0,4,5).

Начинаем с составления таблицы истинности (рис. 1.33).

№\X	a	b	c	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Рисунок 1.33 – Таблица истинности

Карта Карно будет такой:

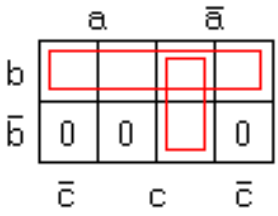


Рисунок 1.34 – Карта Карно функции (рис. 1.33)

И соответствующая минимальная форма $F_{\min} = b + \bar{a}c$. Здесь первый куб содержит четыре единицы (ранг куба равен r = 2). Число переменных в минтерме равно (n – r), где n = 3.

n – r = 3 – 2 = 1 – количество переменных из первого куба.

Из второго куба n – r = 3 – 1 = 2.

Составим СДНФ и выполним склеивание минтермов (конституент единицы) каждого с каждым:

$$f(a,b,c) = \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + ab\bar{c} + abc +$$
$$= \bar{a}c + \bar{a}b + b\bar{c} + bc + ab = \bar{a}c + b + b = \bar{a}c + b$$

Первый склеиваем со вторым , затем с третьим, с четвёртым и т.д. Далее второй склеиваем с третьим, с четвёртым и т.д. Все минтермы пройдут через склеивание. После первого склеивания выполняется второе и т.д. пока оно возможно. Напомним, что минтермы для склеивания могут отличаться только одной переменной.

Пример 3.

Минимизировать функцию двух переменных:

$f_1 = \overline{X_1 * X_2 + \overline{X_1} * \overline{X_2}} + \overline{X_1} * \overline{X_2} =$ / приведем логическую формулу к нормальному виду

(СДНФ)/ $= \overline{X_1 * X_2} * \overline{\overline{X_1} * \overline{X_2}} + \overline{X_1} * \overline{X_2} = \left(\overline{X_1} + \overline{X_2}\right) * \left(X_1 + X_2\right) + \overline{X_1} * \overline{X_2} =$

$= \overline{X_1} * X_1 + \overline{X_1} * X_2 + X_1 * \overline{X_2} + X_2 * \overline{X_2} + \overline{X_1} * \overline{X_2} = \overline{X_1} * X_2 + X_1 * \overline{X_2} + \overline{X_1} * \overline{X_2}$

Составим карту Карно (рис. 1.35).

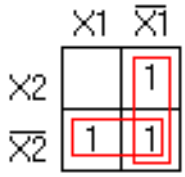


Рисунок 1.35 – Карта Карно функции

По которой получаем $f_{\min} = \overline{X_1} + \overline{X_2}$. Этот же результат можно получить склеиванием минтермов.

Пример 4.

Минимизировать функцию четырёх переменных, карта Карно которой представлена на рис. 1.36.

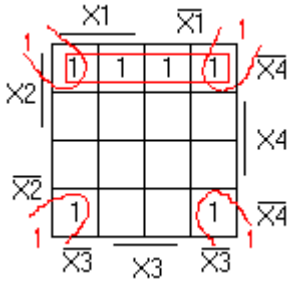


Рисунок 1.36 – Карта Карно исходной функции

Проводим два контура второго ранга и получаем

$$f_{\min} = \overline{X_3} * \overline{X_4} + X_2 * \overline{X_4}$$

Цена схемы равна Ц = 4 + 2 = 6.

Можно в карте Карно объединить нули, но при этом получаем инверсную функцию: $\overline{f_{\min}} = X_4 + \overline{X_2} * X_3$. Здесь проведены два куба – третьего и второго ранга. Цена схемы получается меньше. Ц = 2 + 2 = 4. Её реализация на произвольных элементах имеет вид (рис.1.37).

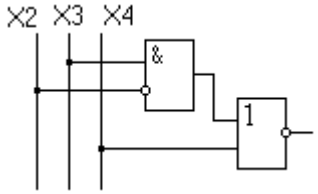


Рисунок 1.37 – Схемная реализация функции (рис. 1.36)

Отрицание можно перенести в правую часть, что не отражается на цене.

$$f_{\min} = \overline{X_4 + \overline{X_2} * X_3}$$

Чем меньше цена, тем лучше. Поэтому минимизировать по карте Карно следует и по единицам, и по нулям. К реализации принимают формулу с наименьшей ценой.

Пример 5.

Имеем функцию пяти переменных, заданную картой Карно (рис. 1.38).

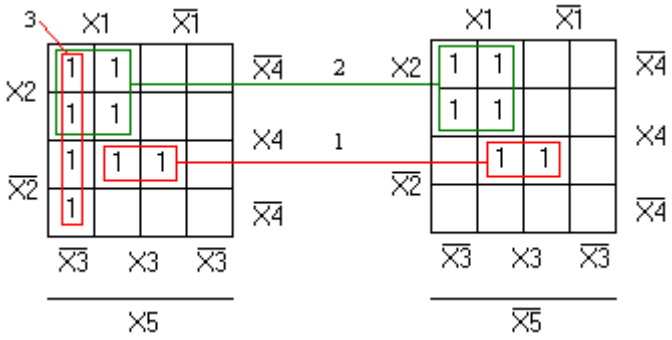


Рисунок 1.38 – Карта Карно исходной функции

Здесь проводим три куба : 1 – куб второго ранга, 2 – куб третьего ранга, 3 – куб второго ранга. Записываем минимальную форму: $f_{\min} = \overline{X_2} X_3 X_4 + X_1 X_2 + X_1 \overline{X_3} X_5$.

Находим цену Ц = 8 + 3 = 11. Желающие могут найти цену схемы после объединения нулей.

1.5.3. Минимизация не полностью определенных функций

Имеется ряд функций, значение которых на некоторых наборах неопределенно или нас просто не интересует. Такие наборы называются **запрещенными** и

используются для минимизации, дополняя функцию нулями или единицами так, чтобы провести куб более высокого ранга.

Пусть, например, имеем функцию трёх переменных, заданную такой таблицей истинности (рис.1.39).

№\X	a	b	c	F
0	0	0	0	*
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	*
6	1	1	0	1
7	1	1	1	0

Рисунок 1.39 – Таблица истинности не полностью определённой функции

Здесь символом * обозначены запрещённые комбинации входных переменных. Требуется найти минимальную форму.

Если не использовать запрещённые наборы, то карта Карно и минимальная форма будут следующими (рис. 1.40).

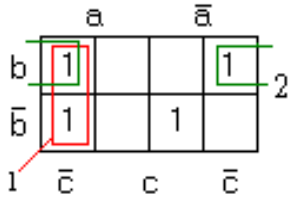


Рисунок 1.40 – Карта Карно функции рис. 1.39

$$f_{\min} = a\bar{c} + b\bar{c} + \bar{a}\bar{b}c$$

Цена схемы равна Ц = 7+ 3 = 10. Если на запрещённых наборах функцию дополнить единицами, то карта Карно принимает вид (рис. 1.41)

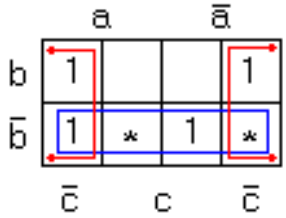


Рисунок 1.41 – Карта Карно функции (рис. 1.39), дополненная единицами

Минимальная форма $f_{\min} = \bar{b} + \bar{c}$. Цена Ц = 2. Схемная реализация получается значительно проще.

1.6 Синтез логических схем

1.6.1 Синтез схем с одним выходом

Синтез комбинационных схем с одним выходом включает следующие этапы:

1. Кодирование входных и выходных переменных и переход от словесного описания работы устройства к таблице истинности.
2. Получение СДНФ.
3. Минимизация функции.
4. Перевод минимальной формы в заданный базис.
5. Составление логической схемы.

Рассмотрим все эти этапы на примере.

Имеются три датчика, выходные сигналы которых двоичные числа.

Используя элементы Шеффера 2И-НЕ, обеспечить индикацию на выходе, если по меньшей мере два из трёх входных сигналов единичны.

Выполняем кодировку и составляем таблицу истинности (рис.1.42).

№\X	a	b	c	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Рисунок 1.42 – Таблица истинности исходной функции

Заполняем карту Карно (рис. 1.43) и получаем минимальную форму, которую с помощью двойного отрицания переводим в базис И – НЕ.

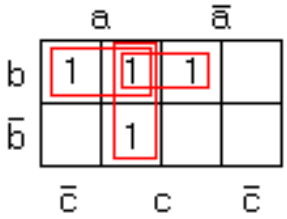


Рисунок 1.43 – Карта Карно функции (рис. 1.42)

Записываем МДНФ:

$$f_{\min} = ab + bc + ac = \overline{\overline{ab + bc + ac}} = \overline{\overline{ab} * \overline{bc} * \overline{ac}}$$

Схемная реализация этой функции выглядит следующим образом (рис. 1.44).

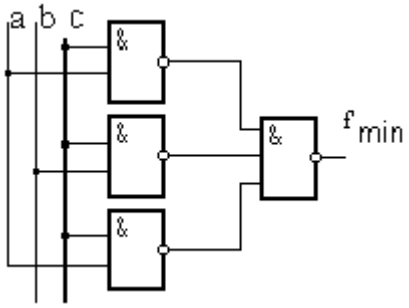


Рисунок 1.44 – Схемная реализация функции (рис. 1.42)

Однако эта схема не отвечает условиям задачи, так как использует трёхвходовой элемент Шеффера. Требуется преобразовать функцию под двухвходовые элементы. Снова воспользуемся двойным отрицанием, которое не меняет значения функции, но объединяет по два входа:

$$f_{\min} = \overline{\overline{ab} * \overline{bc} * \overline{ac}} = \overline{\overline{\overline{ab} * \overline{bc}} * \overline{ac}}$$

Последнее выражение реализуется уже на двухвходовых элементах (рис. 1.45).

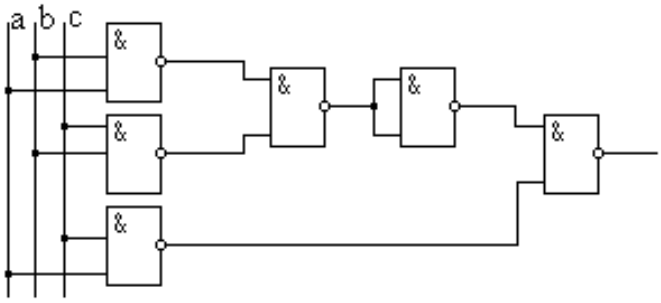


Рисунок 1.45 – Схемная реализация функции (рис. 1.42)
на двухвходовых элементах

Условия задачи выполнены.

1.6.2 Синтез схем с несколькими выходами

Если логическая схема имеет n входов (n - число независимых переменных) и по условию задачи должна иметь k выходов, то каждый из k выходов описывают своей функцией алгебры логики:

$$y_1 = f_1(X_1 X_2 ... X_n)$$
$$y_2 = f_2(X_1 X_2 ... X_n)$$
$$.....$$
$$y_k = f_k(X_1 X_2 ... X_n)$$

Эта система функций называется **системой собственных функций** и описывает так называемый логический (n,k) – полюсник. Этапы синтеза (n,k) – полюсников повторяют этапы синтеза схем с одним выходом, только каждую функцию минимизируют отдельно, хотя известно, что совместная минимизация дает лучшие результаты.

Пусть, например, требуется синтезировать (2 , 5) – полюсник, заданный

такой таблицей истинности:

№\X	a	b	Y_0	Y_1	Y_2	Y_3	Y_4
0	0	0	1	0	1	0	1
1	0	1	0	0	0	1	0
2	1	0	1	1	0	1	1
3	1	1	1	0	0	0	0

Рисунок 1.46 – Таблица истинности (2,5) – полюсника

Составляем систему собственных функций и минимизируем каждую из них:

$$y_0 = \bar{a} * \bar{b} + a * \bar{b} + a * b = \bar{b} + a$$

$$y_1 = a * \bar{b}$$

$$y_2 = \bar{a} * \bar{b}$$

$$y_3 = a * \bar{b} + \bar{a} * b$$

$$y_4 = \bar{a} * \bar{b} + a * \bar{b} = \bar{b}$$

На основании этих выражений составляем схему (рис. 1.47).

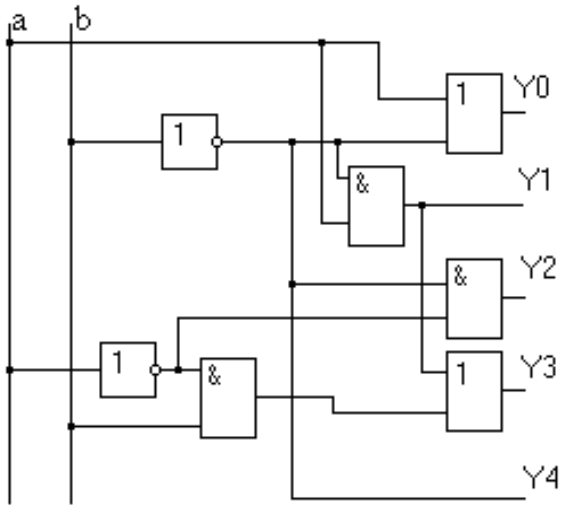


Рисунок 1.47 – Схемная реализация функции (рис. 1.46)

1.6.3 Скобочная форма функций алгебры логики

Пусть в результате минимизации получена следующая минимальная форма (МДНФ):

$$f = X_5 * X_6 + X_2 * X_3 * X_4 * X_6 + X_1 * X_3 * X_4 * X_6$$

Построим схему, реализующую эту функцию (рис. 1.48).

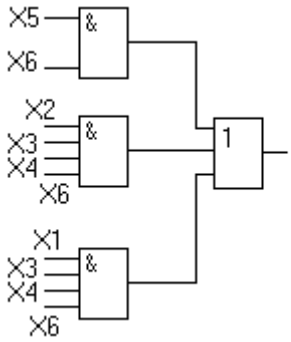


Рисунок 1.48 – Схемная реализация исходной функции

Цена этой схемы равна Ц=10+3=13. Время, через которое сигнал появляется на выходе (задержка) $t_{зад} = 2\tau_{срабат}$, где $\tau_{срабат}$ время прохождения сигнала через один элемент.

Функция, полученная в результате решения задачи минимизации, не является абсолютно минимальной и допускает дальнейшее уменьшение цены путем вынесения за скобку общих множителей (так называемая скобочная форма):

$$f = X_6\left(X_5 + X_3 * X_4\left(X_2 + X_1\right)\right)$$

Построим схему, реализующую эту функцию (рис. 1.49).

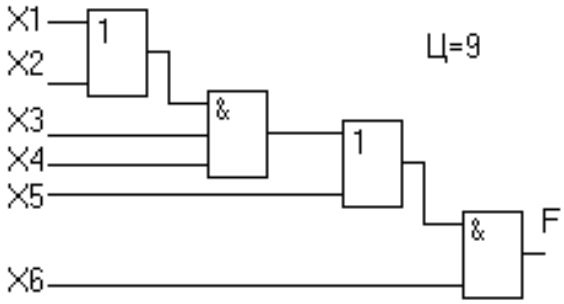


Рисунок 1.49 – Схемная реализация скобочной формы функции

Общее число входов получилось меньше, но время задержки увеличилось и равно $t_{зад} = 4\tau_{срабат}$. Возросла «глубина» схемы и увеличилось время прохождения сигнала. Поэтому, наиболее быстродействующие схемы – схемы, построенные по ДНФ.

Контрольные вопросы

1. Что такое алгебра логики?
2. Что такое таблица истинности?
3. Что такое карта Карно?
4. Что такое синтез логических схем?

2 АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННО-ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

2.1 Системы счисления

Система счисления (Ссч) – это совокупность правил записи чисел цифровыми знаками. Они бывают позиционные и непозиционные (например, римская Ссч).

В вычислительной технике используются только позиционные системы счисления.

Число в любой позиционной Ссч можно представить в виде последовательности цифр:

$$A = a_n a_{n-1} ... a_1 a_0 , b_{-1} b_{-2} ... b_{-k} \ ,$$

где a_i, b_i – цифры данной системы счисления.

Или в виде формулы разложения:

$$A = a_n p^n + a_{n-1} + ... + a_2 p^2 + a_1 p^1 + a_0 p^0 + b_{-1} p^{-1} + b_{-2} p^{-2} + ... b_{-k} p^{-k} ,$$

где p – основание системы счисления (количество различных цифр в Ссч);

p^i – вес единицы данного разряда.

В ЭВМ используются системы счисления с основаниями p = 10, 2, 8, 16. Рассмотрим эти системы счисления.

Десятичная Ссч.

p = 10 Разрешённые цифры (0,1,2,3,4,5,6,7,8,9). Число можно представить так: $A_{10} = 247{,}56_{10} = 2*10^2 + 4*10^1 + 7*10^0 + 5*10^{-1} + 6*10^{-2}$. Веса соседних разрядов влево и вправо от запятой различаются в десять раз (p = 10):

…1000 100 10 1 , 1/10 1/100 1/1000 …

Двоичная Ссч.

p = 2 Разрешённые цифры (0,1) . Число представляется так:

$$A_2 = 101110{,}101_2 = 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = 46{,}625$$

Веса соседних разрядов влево и вправо от запятой различаются в два раза (p = 2)

… 32 16 8 4 2 1, 1/2 1/4 1/8 …

Восьмеричная Ссч.

p = 8 Разрешённые цифры (0,1,2,3,4,5,6,7) . Число представляется так

$$A_8 = 125{,}46_8 = 1*8^2 + 2*8^1 + 5*8^0 + 4*8^{-1} + 6*8^{-2} = 64 + 16 + 5 + 0{,}5 + \frac{6}{64} = 85{,}6_{10}$$

Веса соседних разрядов влево и вправо от запятой различаются в восемь раз (p = 8) … 4096 512 64 8 1 , 1/8 1/64 1/512 ...

Шестнадцатеричная Ссч.

p = 16 Разрешённые цифры (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). Число представляется так:

$$A_{16} = 2AF,C4_{16} = 2*16^2 + 10*16^1 + 15*16^0 + 12*16^{-1} + 4*16^{-2} = 512 + 160 + 15 + \frac{12}{16} + \frac{4}{256} = 687,76_{10}$$

Веса соседних разрядов влево и вправо от запятой различаются в шестнадцать раз (p=16).

…4096 256 16 1, 1/16 1/256 1/4096 ...

Вполне очевидно, что для записи одного и того же числа в разных системах счисления требуется разное количество разрядов. Основание системы счисления во всех системах счисления записывается одинаково: 10.

2.2 Перевод чисел из одной системы в другую

Перевод целых чисел и правильных дробей выполняется по разным правилам. В действительном числе переводят отдельно целую и дробную части.

Перевод целых чисел

Для перевода необходимо исходное число разделить на основание новой системы счисления до получения целого остатка, который является младшим разрядом числа в новой системе счисления. Полученное частное снова делим на основание и так до тех пор, пока частное не станет меньшего нового основания. Все операции выполняются в исходной Ссч.

Например, перевод из десятичной в двоичную и восьмеричную системы счисления.

Возьмём десятичное число A = 124:

A = 124 = 1111100
делением **на 2**

124	2						
124	62	2					
0	62	31	2				
	0	30	15	2			
		1	14	7	2		
			1	6	3	2	
				1	2	1	
					1		

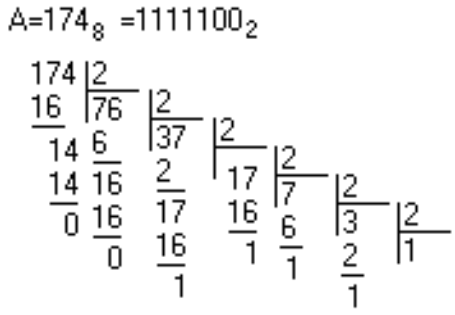
10→8
$A=124_{10}=174_8$

124	8	
8	15	8
44	8	1
40	7	
4		

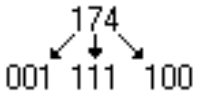
Аналогично переходим $10 \rightarrow 16$ $A = 124_{10} = 7C_{16}$

124	16
112	7
12	

Перевод $8 \rightarrow 2$ также выполняем делением:

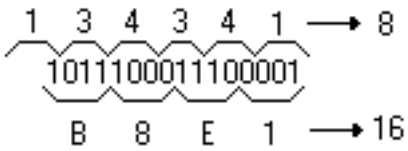


Однако выполнять деление в восьмеричной системе не всегда удобно (подсознательно мы делим в десятичной Ссч!!), поэтому используют более простое правило: каждую восьмеричную цифру представляют триадой двоичных:



Аналогично делают обратный перевод, т.е. двоичное число разбивают на триады относительно крайнего правого разряда (или запятой) и каждую триаду представляют восьмеричной цифрой. Перевод $16 \rightarrow 2$ выполняют с помощью тетрад

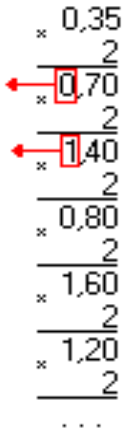
Перевод из двоичной системы счисления в восьмеричную и шестнадцатеричную и обратно выполняется так же:



Перевод правильных дробей

Этот перевод выполняют умножением исходного числа на основание новой системы счисления. Целая часть произведения является старшим разрядом числа в новой системе счисления. Дробную часть произведения снова умножают на основание системы счисления и т.д. Все операции выполняют в исходной Ссч.

Перевод $10 \rightarrow 2$ $A = 0{,}35_{10} = 0{,}01011$



В общем случае перевод правильных дробей является бесконечным. Число разрядов в новой системе можно найти исходя из одинаковой точности представления чисел в разных системах счисления.

Одинаковая точность – одинаковые веса младших разрядов чисел.

$$P^{-n_p} = q^{-n_q},$$

где p и q - основания старой и новой Ссч соответственно.

Например, для десятичного числа 0,35 вес младшего разряда 1/ 100 = 0.01, а для двоичного числа 0,01011 вес младшего разряда 1/ 32 ≅ 0,03.

Берем логарифм по основанию p :

$$-n_p = -n_q \log_p q, \text{ откуда находим } \quad n_q = \frac{n_p}{\log_p q}.$$

Тогда при $p = 10$ $q = 2$ получаем

$$n_2 = \frac{n_{10}}{\lg_2} = \frac{n_{10}}{0{,}3}$$

Если $n_{10} = 2$, то в новой Ссч количество разрядов равно

$$n_2 = \frac{2}{0{,}3} = 7,$$

а при $n_{10} = 3$, $n_2 = \frac{3}{0{,}3} = 10$. Для восьмеричной и шестнадцатеричной Ссч имеем

$$n_8 = \frac{n_{10}}{\lg 8} = \frac{n_{10}}{0{,}9} \qquad n_{16} = \frac{n_{10}}{1{,}2}$$

Из десятеричной в восьмеричную Ссч переходим по тому же правилу:

$10 \to 8$ $\qquad A = 0{,}35_{10} = 0{,}263_8 = 2*8^{-1} + 6*8^{-2} + 3*8^{-3} = 0{,}25 + \frac{6}{64} + \frac{3}{256} \cong 0{,}35$

Аналогично можно делать перевод в любую систему счисления. Перевод из $8 \to 2 \to 16$ и назад выполняется с помощью триад и тетрад, которые отмеряют от запятой.

2.3 Арифметические операции в различных системах счисления

Арифметические операции в различных системах счисления выполняют на основании таблиц сложения и умножения.

Двоичная Ссч. Таблицы сложения и умножения.

0+0=0	0*0=0
0+1=1	0*1=0
1+0=1	1*0=0
1+1=10	1*1=1

Возьмём два десятичных числа и выполним сложение, вычитание и умножение.

$A = 5_{10} = 101_2$

$B = 3{,}5_{10} = 11{,}1_2 = 1*2^1 + 1*2^0 + 1*2^{-1} = 2 + 1 + 0{,}5 = 3{,}5$

A+B=101 0
11,1
1000,1 $= 1*2^3 + 1*2^{-1} = 8{,}5_{10}$

A-B=101 0
11,1
001,1 $= 1*2^0 + 1*2^{-1} = 1{,}5_{10}$

A*B= 101
11,1
101
101
101
10001,1 $= 1*2^4 + 1*2^0 + 1*2^{-1} = 17{,}5_{10}$

Восьмеричная Ссч.
Таблицы сложения и умножения (рис. 2.1 и 2.2 соответственно).

+	1	2	3	4	5	6	7
1	2	3	4	5	6	7	10
2	3	4	5	6	7	10	11
3	4	5	6	7	10	11	12
4	5	6	7	10	11	12	13
5	6	7	10	11	12	13	14
6	7	10	11	12	13	14	15
7	10	11	12	13	14	15	16

Рисунок 2.1 – Таблица сложения восьмеричных чисел

*	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	10	12	14	16
3	3	6	11	14	20	22	25
4	4	10	14	20	24	30	34
5	5	12	17	24	31	36	43
6	6	14	22	30	36	44	52
7	7	16	25	34	43	52	61

Рисунок 2.2 – Таблица умножения восьмеричных чисел

Выполним сложение, вычитание и умножение взятых нами чисел.

$A = 5_{10} = 5_8$

$B = 3{,}5_{10} = 3{,}4_8 = 3*8^0 + 4*8^{-1} = 3 + \frac{4}{8} = 3{,}5$

A+B=5,0
3,4
10,4 $= 1*8^0 + 4*8^{-1} = 8{,}5_{10}$

A-B=5,0
3,4
1,4 $= 1*8^0 + 4*8^{-1} = 1{,}5_{10}$

A*B=5,0
3,4
3,4
5
21,4 $= 2*8^1 + 1*8^0 + 4*8^{-1} = 17{,}5$

Теперь в Ссч с основанием $P = 16$

$A = 5_{10} = 5_{16}$

$B = 3{,}5_{10} = 3{,}8_{16}$

A+B=5,0
3,8
8,8 $= 8*16^0 + 8*16^{-1} = 8{,}5$

A-B=5,0
3,8
$1{,}8_{16} = 1{,}5_{10}$

A*B=3,8
5
11,8 $= 1*16^1 + 1*16^0 + 8*16^{-1} = 16 + 1 + 0{,}7 = 17{,}5$

Надо помнить, что при переносе в старший разряд уходит число единиц, равное основанию Ссч, а при заёме из старшего разряда приходит число единиц, также равное основанию Ссч.

2.4 Формы представления чисел

Наименьшая мера информации, используемая в цифровой технике 1 бит (один двоичный разряд). Производными от этой единицы являются:

1 байт = 8 бит

1К = 2^{10} бит = 1024 бит («Кило»)

1М = 2^{10} К = 1024 К («Мега»)

1Г = 2^{10} М = 1024 М («Гига»)

Наряду с этим используются и другие единицы: слово, полуслово, двойное слово (слово может быть 1, 2, 4, 8 или др., число байт), поле (до 256 байт).

В ЭВМ используются две формы представления чисел: с фиксированной запятой (естественная форма) и с плавающей запятой (показательная форма).

Форма с фиксированной запятой

Используется для записи целых чисел.

Имеются две разновидности: знаковые и беззнаковые числа (рис. 2.3).

Рисунок 2.3 – Форматы целых чисел

Минимальное число, которое может быть представлено в этом формате: 0 и 1. Максимальное число равно

$A_{max} = 2^{15} - 1 = 32767_{10}$ со знаком

$2^{16} - 1 = 65535_{10}$ без знака

Форма с плавающей запятой

Число представляют в виде:

$$N = \pm m * p^{\pm q},$$

где m - мантисса числа - правильная дробь;

p - основание Ссч;

q - порядок числа.

Если мантисса находится в пределах $\frac{1}{p} \le m < 1$, то говорят, что число **нормализовано.**

$0{,}26 * 10^{15}$ нормализованное число;

$0{,}026 * 10^{16}$ не нормализованное.

Нормализацию чисел машина выполняет автоматически и работает только с нормализованными числами.

Классический формат числа с плавающей запятой представлен на рис. 2.4.

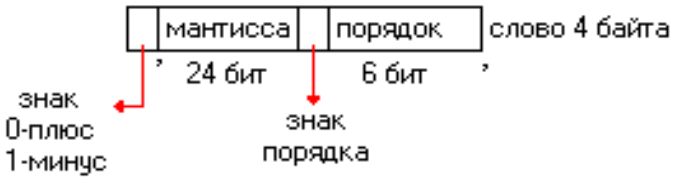


Рисунок 2.4 – Формат с плавающей запятой

Здесь запятая в мантиссе фиксирована перед старшим разрядом, а порядок – целое число (запятая фиксирована после младшего разряда). В разрядной сетке запятые нигде не стоят, они подразумеваются в определённом месте.

Максимальное число, которое может быть записано в этом формате:

$A \cong 1 * 2^{(2^6-1)} \cong 2^{63} \cong 10^{19}$, так как $10^x = 2^{63}$, откуда $X = 63 \lg 2 = 63 * 0{,}3 \cong 19$.

Минимальное нормализованное число $A_{\min} = 2^{-1} * 2^{-(2^6-1)} = 2^{-1} * 2^{-63} = 2^{-64} \cong 10^{-19}$.

В этом формате имеется одно замечательное число - нуль. Истинный нуль - когда во всех разрядах мантиссы стоят нули, и машинный (нормализованный) нуль, когда число меньше минимально представимого в данном формате, он может быть как положительным, так и отрицательным (рис. 2.5).

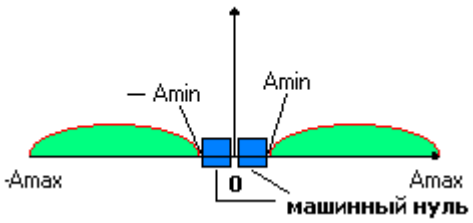


Рисунок 2.5 – Диапазон чисел формата с плавающей запятой

Существует большое количество различных форматов с плавающей запятой. Это осложняет перевод программ с одной машины на другую, поэтому в 1985г. был принят международный стандарт IEEE-754, который оговаривает четыре формата с плавающей запятой.

Базовый одинарный формат

Слово длиной четыре байта, в котором используется смещённый порядок (характеристика) числа (рис. 2.6).

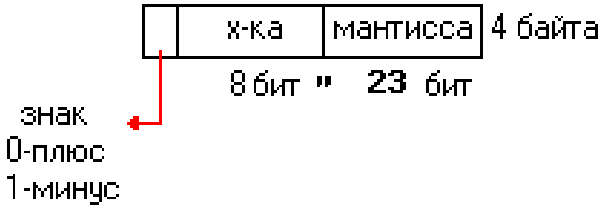


Рисунок 2.6 – Базовый одинарный формат с плавающей запятой

Запятые для характеристики и мантиссы фиксированы. Характеристика – целое число без знака, для неё запятая расположена после младшего разряда. Для мантиссы запятая расположена перед старшим разрядом.

X = 128 + q – смещённый порядок ($2^n - 1 = 255$ - максимальное число в восьми битах). Характеристика числа всегда положительна:

$$X \geq 0$$

$$q = 127 \qquad X = 255$$

$$q = -128 \qquad X = 0$$

Это упрощает выполнение арифметических операций, так как не надо проверять знак порядка. В этом формате

$$A_{\max} \approx 1 * 2^{127} \approx 10^{38}$$

$$10^x = 2^{127}$$

$$X = 127 \lg 2 = 127 * 0.3 = 38$$

Поэтому диапазон чисел $N = \pm 10^{\pm 38}$.

По сравнению с классическим форматом диапазон чисел значительно шире, но мантисса числа на один бит меньше. Поэтому точность представления должна быть хуже, но так как число всегда нормализовано, то первую единицу после запятой не хранят, а подразумевают, поэтому точность чисел такая же, как в классическом формате. В этом формате используется так называемая **скрытая единица мантиссы**.

Базовый двойной формат

Здесь слово длиной восемь байт (рис. 2.7). Смещение порядка составляет 1024.

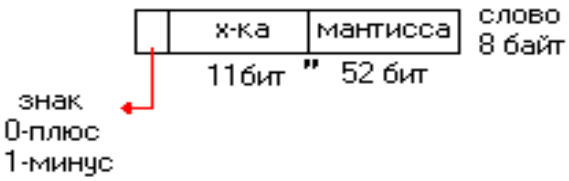


Рисунок 2.7 – Базовый двойной формат

Характеристика X = 1024 + q. Порядок может находиться в пределах

$$-1024 \leq q \leq 1023$$

Диапазон чисел следующий:

$$A_{\max} = 1 * 2^{1024}$$
$$10^{x} = 2^{1024}$$
$$X = 1024 \lg 2 = 1024 * 0.3 = 308$$
$$N \cong \pm 10^{\pm 308}$$

В базовых форматах значение характеристики, равное нулю, соответствует нулевому числу, а значение характеристики, равное максимуму, соответствует бесконечности.

Мантисса длинной 24 бита соответствует точности представления числа 6 – 7 десятичных цифр. Мантисса длинной 52 бита соответствует точности представления 16 – 17 десятичных цифр.

Имеются также и расширенные форматы, но их мы не рассматриваем.

2.5 Машинные коды

Независимо от формы записи чисел с фиксированной или плавающей запятой все числа в ЭВМ представляются в виде специальных кодов – прямом, обратном или дополнительном.

Прямой код используется для хранения чисел в памяти и выполнения операции умножения.

Обратный и дополнительный коды используются для сложения положительных и отрицательных чисел.

Рассмотрим машинные коды на примере чисел с фиксированной запятой.

Прямой код:

$$[A]_{пр} = 0 \quad a_n a_{n-1} \ \ldots \ a_1 a_0 \quad A \geq 0$$

$$[A]_{пр} = 1 \quad a_n a_{n-1} \ \ldots \ a_1 a_0 \quad A \leq 0$$

Знак плюс кодируют нулём, а знак минус единицей. Знак числа обязательно должен быть в любом машинном коде.

Например:

Число	Прямой код
+ 1101	01101
- 1101	11101
+ 0,1101	01101
- 0,1101	11101
- 0,0000	10000
+ 0,0000	00000

Запятая в коде не пишется. Число нуль в прямом коде имеет двойное изображение – положительное и отрицательное.

Обратный код:

$$[A]_{обр} = 0 \quad a_n a_{n-1} \ \dots \ a_1 a_0 \quad A \geq 0,$$

$$[A]_{обр} = 1 \quad \overline{a_n}\,\overline{a_{n-1}} \ \dots \ \overline{a_1}\,\overline{a_0} \quad A < 0,$$

где $\overline{a_i} = 1 - a_i$ – дополнение числа до 1 (инверсия разрядов двоичного числа).

Например,

Число	Обратный код
+ 1101	01101
- 1101	10010
- 0,1101	10010
+ 0,0000	00000

Дополнительный код:

$$[A]_{доп} = 0 \quad a_n a_{n-1} \ \dots \ a_1 a_0 \quad A \geq 0,$$

$$[A]_{доп} = 1 \quad \overline{a_n}\,\overline{a_{n-1}} \ \dots \ \overline{a_1}\,\overline{a_0} \quad +00...01 \quad A < 0,$$

где $\overline{a_i} = 1 - a_i$ – дополнение числа до 1 (инверсия разрядов двоичного числа).

Дополнительный код числа это обратный код плюс единица в младший разряд.

Например:

Число	Дополнительный код
+1101	01101
- 1101	10011
- 1100	10100

Дополнительный код правильной дроби – это дополнение числа до основания системы счисления. $A + [A]_{доп} = 10$, где 10 – основание системы счисления.

Дополнительный код n – разрядного целого отрицательного числа есть результат вычитания этого числа из единицы с (n+1) нулями. Так, для числа A = - 1101 (n = 4) $[A]_{доп} = 100000 - 1101 = 10011$.

Для положительных чисел прямой, обратный и дополнительный коды совпадают.

2.6 Операции над числами в машинных кодах

Операции с фиксированной запятой

Сложение чисел

В вычислительной технике, благодаря машинным кодам, операция вычитания заменяется операцией сложения с числом обратного знака.

$$A - B = A + (-B)$$

Сложение в обратном коде

Пусть даны два числа A и B. Надо найти их сумму.

$$A = -0{,}1101$$

$$B = +0{,}0010$$

Выполним сложение в обратном коде. При этом биты знака участвуют в сложении наравне со значащими разрядами.

$$[A]_{ОБР} = {}_{+}10010$$

$$[B]_{ОБР} = 00010$$

$$[C]_{ОБР} = \overline{10100}$$

Ответ: $C = -0{,}1011$

Теперь сложим два других числа:

$$A = -0{,}1101$$

$$B = -0{,}0010$$

$$[A]_{ОБР} = {}_{+}10010$$

$$[B]_{ОБР} = 11101$$

$$1 \leftarrow {}_{+}01111$$

$$\longrightarrow 1$$

При сложение в обратном коде единица переноса из старшего (знакового) бита добавляется в младший разряд результата имеет место так называемый циклический перенос. Результат сложения $[C]_{ОБР} = 10000$, а ответ C = - 0,1111.

Сложение в дополнительном коде

Выполним сложение тех же чисел:

$$A = -0{,}1101$$

$$B = +0{,}0010$$

$$[A]_{ДОП} = 10011$$

$$[B]_{ДОП} = 00010$$

$$[C]_{ДОП} = 10101 \quad \text{дополнительный код результата}$$

$$1$$

$$[C]_{ОБР} = \overline{10100}$$

Теперь возьмём другие числа:

$$A = -0{,}1101$$

$$B = -0{,}0010$$

$$[A]_{доп} = 10011$$

$$[B]_{доп} = 11110$$

$$[C]_{доп} = 1 \leftarrow 10001$$

В дополнительном коде единица переноса из знакового бита отбрасывается. Тогда $[C]_{ОБР} = 10000$, а ответ равен C = - 0,1111.

При сложении обязательно выравнивание разрядов слагаемых нулями (не кодов!) Для отрицательных чисел эти выравнивающие нули превращаются в единицы при инвертировании. Знаки чисел (крайние левые биты кодов) обязательно находятся один под другим.

Аналогично складывают и целые числа. Например, сложить в разрядной сетке 1 байт два числа C = A + B, где A= + 9_{10} и B = - 7_{10}. Разместим их в фиксированной разрядной сетке – в восьми битах:

[A]пр = 0 0 0 0 1 0 0 1
[B]пр = 1 0 0 0 0 1 1 1

[A]доп = 0 0 0 0 1 0 0 1
[B]доп = 1 1 1 1 1 0 0 1
[C]доп = 0 0 0 0 0 0 1 0
[C]пр = 0 0 0 0 0 0 1 0

В результате сложения получилось положительное число, поэтому других преобразований ответ не требует.

Умножение двоичных чисел

Умножение выполняется по тем же правилам , что и десятичное умножение, то есть перемножаются модули чисел, а знак результата получается сложением по модулю двух знаков сомножителей.

Известно, что произведение двух n-разрядных чисел есть число 2n-разрядное:

$$n * n = 2n$$

Возьмем два целых четырёхразрядных двоичных числа:

$$A = -0101$$
$$B = +0010 \quad (\text{n} = 4)$$
$$C = A * B$$

Знак произведения $\text{signC} = \text{signA} \oplus \text{signB} = 1 \oplus 0 = 1$. Перемножаем модули

$[A]_{ПР} = 10101$
$[B]_{ПР} = 00010$

$$\begin{array}{r} \times\ 0101 \\ 0010 \\ \hline 0000 \\ 0101 \\ 0000 \\ 0000 \\ \hline 00001010 \end{array}$$

n=4 n=4

При умножении целых чисел в качестве результата берутся младшие n-разрядов. При этом в старших n-разрядах должны быть нули. Если это не так, то имеет место переполнение разрядной сетки машины. Для расширения разрядной сетки нужно добавить слева нули. Теперь умножим два дробных числа:

$A = -0{,}101 (n = 3)$

$B = 0{,}010$

$SignC = SignA \oplus SignB = 1 \oplus 0 = 1$

$$\begin{array}{r} \times \begin{array}{r} 101 \\ 010 \end{array} \\ \hline 000 \\ 101 \\ 000 \\ \hline 0{,}001010 \end{array} \longrightarrow C_0 = 0{,}001_2 = 1 * 2^{-3} = 0{,}125$$

n=3 n=3

При умножении дробных чисел в качестве результата выбираются старшие n-разрядов. При этом происходит округление по правилу: если в (n+1) разряде была единица, то к ответу добавляется единица в младший разряд, если в (n+1) разряде был ноль, то к ответу ничего не добавляется. В примере имеем ноль, поэтому ничего не добавляем. Проверим ответ в десятичной системе:

$A = 1 * 2^{-1} + 1 * 2^{-3} = 0.625 \qquad B = 1 * 2^{-2} = 0.25 \qquad C_0 = A * B = 0.625 * 0.25 = 0.156$

$$\Delta = 0.156 - 0.125 = 0.031 \quad - \textit{ абсолютная ошибка}$$

$$\delta = \frac{\Delta}{C_0} = \frac{0.031}{0.156} = 20\% \quad - \textit{относительная погрешность}$$

Округление при умножении дробных чисел вносит значительную погрешность (операции же с целыми числами выполняются абсолютно точно!)

Погрешность, возникающая при умножении дробных чисел, равна

$$\Delta = \pm \frac{1}{2} * 2^{-n} = \pm 2^{-1} * 2^{-n} = \pm 2^{-(n+1)}$$

Чтобы не потерять точность при вычислениях, нужно расширить разрядную сетку путём добавления нулей справа.

Операции с плавающей запятой

Операции над числами с плавающей запятой выполняются по тем же правилам. Мантиссы отрицательных чисел вступают в операцию в обратном или дополнительном коде. Перед сложением чисел сначала выравнивают порядки – приводят число к большему порядку и затем складывают мантиссы. Полученный результат нормализуют. Для умножения чисел перемножают мантиссы по правилам двоичной арифметики, а порядки складывают.

В качестве примера рассмотрим сложение двух действительных десятичных чисел A = 3,23 и B = - 14,85 в классическом формате с плавающей запятой, используя дополнительный код . C = A + B = ?

Решение начнём с перевода чисел в двоичную систему счисления.

$A = 3{,}23_{10} = + 11, 0011101_2 = + 0{,}110011101 * 2^{10}$

$B = - 14{,}85_{10} = - 1110{,}1101100_2 = - 0{,}11101101100 * 2^{100}$

Количество разрядов после запятой в ненормализованном двоичном числе равно $n_2 = n_{10} / 0{,}3 = 2 / 0{,}3 = 7$.

Занесём эти числа в разрядную сетку 4 байта

[A]пр = 0 11001110100000000000000 0 000010
[B]пр = 1 11101101100000000000000 0 000100

Приводим операнды к большему порядку ($4_{10} = 100_2$)

[A]пр = 0 00110011101000000000000 0 000100

Записываем дополнительные коды операндов

[B]обр = 1 00010010011111111111111 0 000100
[B]доп = 1 00010010100000000000000
[A]доп = 0 00110011101000000000000
[C]доп = 1 01000110001000000000000 – результат сложения
[C]обр = 1 01000110000111111111111
[C]пр = 1 10111001111000000000000 0 000100

Ответ в двоичной системе счисления: $C = -0{,}10111001111 * 2^{100} = -1011{,}1001111_2 = -(2^3 + 2^1 + 2^0 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7}) \approx -11{,}617_{10}$.

Точный ответ $C_0 = -14{,}85 + 3{,}23 = -11{,}62$. Появилась погрешность за счёт перевода чисел в двоичную систему счисления.

2.7 Двоично-десятичная система кодирования

Все операции в ЭВМ выполняются в двоичной системе счисления. Однако, человеку удобно вводить информацию и получать результаты вычислений в десятичной системе счисления. Для этого используются, так называемые, двоично-десятичные коды. В них один десятичный разряд представляют четырьмя двоичными разрядами (тетрадой). При помощи четырех бит можно закодировать шестнадцать различных символов (цифр), поэтому используются не все возможные комбинации . Существует много разных систем кодирования [10], но наиболее широко применяется код прямого замещения - код 8–4–2–1 (это веса двоичных разрядов влево от запятой). Составим таблицу соответствия двоично-десятичного кода и десятичных цифр.

Двоично-десятичный код				Десятичная цифра
8	4	2	1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Остальные комбинации двоичного кода являются лишними (запрещенными). Запишем пример двоично-десятичного кода:

1258 = 0001 0010 0101 1000
589 = 0000 0101 1000 1001

Сложение двоично-десятичных чисел

Сложение двоично-десятичных чисел производится по правилам двоичной арифметики, с учетом переносов. Пусть имеем два десятичных числа A и B. Требуется найти сумму C = A + B.

В каждой тетраде выполняем сложение трёх чисел: двух слагаемых и переноса из предыдущего разряда, т.е. $a_n + b_n + p_{n-1.}$ При этом возможны такие ситуации:

1) $a_n + b_n + p_{n-1} < 10$. A = 14 B = 23 C = A + B = 37

A = 0001 0100
B = 0010 0011
C = 0011 0111 $\Rightarrow$ 37 Ответ получился верный.

2) $a_n + b_n + p_{n-1} > 15$. A = 47 B = 39 C = A + B = 86

A = 0100 0111
B = 0011 1001
C = 1000 $\leftarrow$0000 $\Rightarrow$ 80 Ответ получился неверный, так как был перенос из младшей тетрады в старшую. Тетрада переполняется числом 16, т.е. единица межтетрадного переноса уносит в старшую тетраду 16 , а не 10 единиц как в десятичной системе счисления (шесть лишних единиц!) Поэтому результат необходимо скорректировать путём добавки + 6. Выполним коррекцию

C = 1000 0000
0000 0110 - коррекция (+ 6)
ответ 1000 0110 $\Rightarrow$ 86 Ответ правильный.

3) $10 \le a_n + b_n + p_{n-1} \le 15$ A = 47 B = 36 C = A + B =83

A = 0100 0111
B = 0011 0110
C = 0111 1101 $\Rightarrow$ Ответ неверный, хотя и нет межтетрадного переноса, но имеется запрещённая комбинация.

Необходимо вызвать искусственное переполнение тетрады путём добавки +6. Выполним коррекцию

C = 0111 1101
0000 0110 - коррекция (+ 6)
ответ 1000$\leftarrow$0011 $\Rightarrow$ 83 Теперь ответ правильный.

Внимание! Коррекция результата выполняется только один раз, поэтому межтетрадный перенос при коррекции не требует ещё одной коррекции.

Таким образом, при сложении двоично-десятичных чисел выполняется коррекция результата по правилу: если был межтетрадный перенос (переполнение тетрады) или получилась запрещённая комбинация, то к этой тетраде добавляется + 6 (0110).

Ещё один пример. A = 479 B = 128 C = A + B = 607

```
  0100  0111  1001
+ 0001  0010  1000
-------------------  – выполняем коррекцию
  0101  1010  0001

  0000  0110  0110
-------------------
  0110  0000  0111 – результат верный
```

В старшей тетраде коррекция 0, в средней +6 (запрещённая комбинация), в младшей тетраде +6 (был перенос).

2.8 Переполнение разрядной сетки машины

При сложении чисел одинакового знака может возникнуть переполнение разрядной сетки. Признаком переполнения является отличие знака результата от знаков слагаемых. Пусть, например, требуется сложить два числа:

$$A = -1101 \qquad [A]_{доп} = 10011$$

$$B = -1001 \qquad [B]_{доп} = 10111$$

$$C = A + B \qquad [C]_{доп} = 01010$$

Складывали два отрицательных числа , а ответ – положительный. Чтобы не было переполнения необходимо расширить разрядную сетку. Запишем числа A и B в пяти битах:

$$A = -01101 \qquad [A]_{доп} = 110011$$

$$B = -01001 \qquad [B]_{доп} = 110111$$

$$C = A + B \qquad [C]_{доп} = 101010$$

$$[C]_{обр} = 101001$$

$$C = -10110$$

.

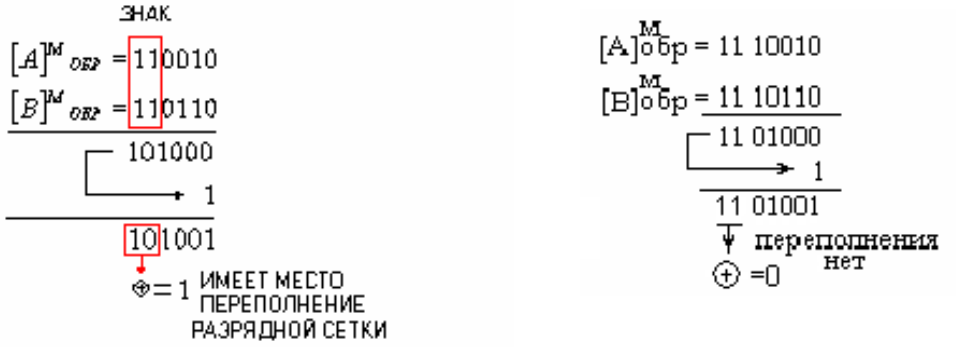
Теперь ответ верный. В вычислителях для контроля переполнения следят за переносом в знаковый разряд и из него. Если оба переноса имеют место или оба отсутствуют, то переполнения нет. Если есть хотя бы один из переносов, то имеет место переполнение разрядной сетки.

В некоторых машинах для контроля переполнения используют так называемые модифицированные коды: прямой, обратный, дополнительный. В этих кодах под знак числа отводится по два бита. После сложения эти знаковые биты складывают между собой по модулю два. Если результат сложения равен 0, то нет переполнения, если результат сложения равен 1, то переполнение есть.

Проверим это правило на примере обратного модифицированного кода. Сложим два числа:

$$A = -1101$$

$$B = -1001$$



Таким образом, модифицированные коды удобны для использования, хотя и занимают на один бит больше памяти.

2.9 Контроль информации

Рассмотренные ранее алгоритмы выполнения операций дают правильный результат, если не будет сбоев в работе машины. Если произойдет сбой, то ответ будет неправильный и пользователь никогда об этом не узнает, поэтому в первых вычислительных машинах всегда выполнялся двойной счет. В современных машинах существуют специальные системы контроля, которые позволяют определить наличие ошибки и даже автоматически ее исправить. Вообще система контроля это совокупность средств и методов, обеспечивающих проверку правильности работы устройства, обнаружение и исправление ошибок.

Простейшим методом контроля является проверка на четность (проверка паритета). Исходное число дополняют битом четности, в который заносят 0 или 1 с тем, чтобы общее количество единиц было четным (рис.2.8).

Число	Бит паритета	Код
1010	0	10100
1000	1	10001
0010	1	00101

Рисунок 2.8 – Добавление бита паритета

Этот метод позволяет выявить только одну ошибку. Кодировка и проверка выполняются автоматически с помощью сумматоров по модулю два (рис. 2.9).

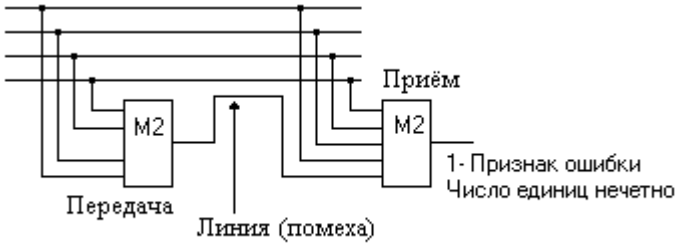


Рисунок 2.9 – Проверка паритета в месте приёма

Для автоматического исправления ошибки этот метод видоизменяют – добавляют несколько контрольных разрядов по типу матрицы (рис. 2.10).

M_1	M_2	M_3	K_1
M_4	M_5	M_6	K_2
M_7	M_8	M_9	K_3
K_4	K_5	K_6	

Рисунок 2.10 – Добавление битов паритета в матрицу

Здесь обозначено: M – информационные разряды
K – контрольные разряды

N=M+K=9+6=15 – общее число разрядов кода.

Проверку на четность выполняют по строкам и по столбцам. Одиночная ошибка сразу выявляется и исправляется (рис. 2.11).

0	1	1	0
1	0	1	0
0	$0 \rightarrow 1$	0	1 ?
1	0 ?	0	

Рисунок 2.11 – Исправление ошибки в матрице

Кодирование по методу четности - довольно мощный и надежный способ защиты информации. Такие узлы находят применение на практике.

Для более длинных посылок или обнаружения двух и более ошибок используют и более сложные коды, например, коды Хемминга.

2.10 Представление алфавитно-цифровой информации

С помощью двоичных знаков можно записывать, хранить и обрабатывать не только числа, но и любую символьную информацию. Для этого каждому символу необходимо присвоить свой код. С помощью кода, состоящего из n – разрядов, можно представить 2^n различных символов. В 1963 г. был введён 7-разрядный код ASCII (American Standard Code for Information Interchange американский стандартный код для обмена информацией). Он позволял кодировать 128 символов. На основе этого кода построен отечественный код КОИ-7 и восьмиразрядные коды КОИ-8 и ДКОИ (двоичный код обработки информации), в

которых каждый символ кодируется одним байтом. Последующие модификации кода ASCII также 8-разрядные (байтовые). Поэтому можно закодировать 256 различных символов. Например, разряды байта кода ДКОИ распределяются следующим образом:

Рисунок 2.12 – Признаки символов в байте кода ДКОИ

Старшая тетрада (зона) носит служебный характер и указывает тип символа. Код непосредственно символа заносится в младшую тетраду. В других кодах (КОИ-8 и ASCII) кодировка зоны отличается от приведённой на рисунке 2.12, но смысл зоны сохраняется – признак символа.

Любая информация устройством ввода преобразуется в код и хранится в памяти машины в виде слов:

Рисунок 2.13 – Формат слова (четыре байта)

Размер слова может быть различным : 2 байта, 4 байта, 8 байт и др.

Любая деловая информация содержит цифр больше, чем букв, поэтому для записи чисел используются специальные форматы: двоичные и десятичные форматы целых чисел (с фиксированной запятой) и форматы с плавающей запятой. Кратко ознакомимся с ними.

Десятичный формат. Используется для записи целых чисел при этом знак числа записывают в зону младшего байта. Это так называемый десятичный распакованный формат – формат с зоной. Он приведён на рисунке 2.14.

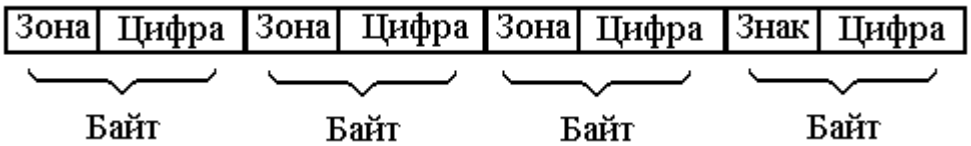


Рисунок 2.14 – Десятичный распакованный формат

Очевидно, что память используется не рационально, так как хранить одинаковую зону для каждой цифры неразумно, поэтому используют десятичный упакованный формат. Он приведён на рисунке 2.15.

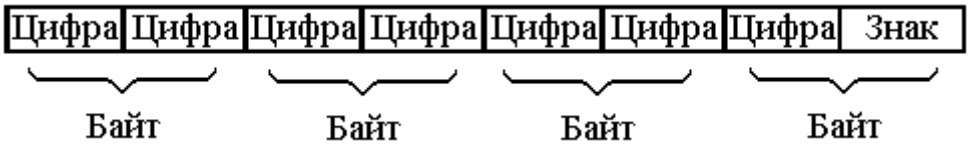


Рисунок 2.15 - Десятичный упакованный формат

Здесь код знака числа содержится в младшей тетраде младшего байта. Так записываются положительные и отрицательные числа. Память используется более рационально, но это **только десятичные числа.**

Двоичный формат. Используется для записи чисел с фиксированной и с плавающей запятой. Формат с фиксированной запятой для целых чисел приведен на рисунке 2.16.

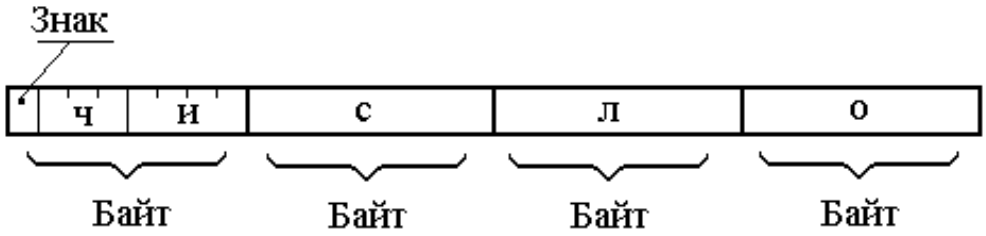


Рисунок 2.16 - Двоичный формат для целых чисел

Знак кодируется как обычно: 0 – плюс, 1 – минус. Число записывается в двоичной системе счисления и каждый байт представляют парой шестнадцатеричных цифр. Максимальное положительное число, представимое в этом формате:

0111 1111 ….1111 или

7 F FF FF FF $_{(16)}$

Отрицательные числа в этом формате представляют в дополнительном коде. Так, например, числа +50 и -50 (десятичные) будут представлены в следующем виде :

00 00 00 32 (+50)

FF FF FF CE (- 50)

Числа +10 и – 10 (десятичные) записываются так:

00 00 00 0A (+10)

FF FF FF F6 (-10)

Какое наибольшее (по модулю) отрицательное число можно записать в этом формате? Для положительных чисел это $2^n - 1$, а для отрицательных 2^n. Возьмём, например, n = 3 . Тогда 0 111 (или $2^3 - 1 = +7$). Дополнительный же код для числа - 7 = - 111 равен 1 001, то есть дополнение не равно нулю и

можно ещё уменьшить отрицательное число с тем, чтобы дополнение стало равно нулю. Значит в трёх битах можно записать число минус 8, а его дополнительный код равен 1 000.

Поэтому в формате четыре байта представимые целые числа лежат в диапазоне от -2^{31} до $+2^{31}-1$, то есть диапазон несимметричный:

$$-2147483648 \le N \le +2147483647$$

$$-80\ 00\ 00\ 00 \le N \le +7F\ FF\ FF\ FF$$

В двоичном формате с плавающей запятой положительные числа записываются в нормализованном виде (см. п. 2.4) в прямом коде, а отрицательные числа (мантиссы) записываются в дополнительном коде. Характеристики чисел всегда положительны. Например, десятичные числа + 50 и - 50 в формате с плавающей запятой будут записаны так :

$A = \pm 50 = \pm 32_{16} = \pm 0011\ 0010_2 = \pm 0,110010\ 2^{+0110}$

Характеристика числа $X = 128 + q = 126 + 6 = 134_{10} = 86_{16} = 1000\ 0110_2$.

Число + 50 :

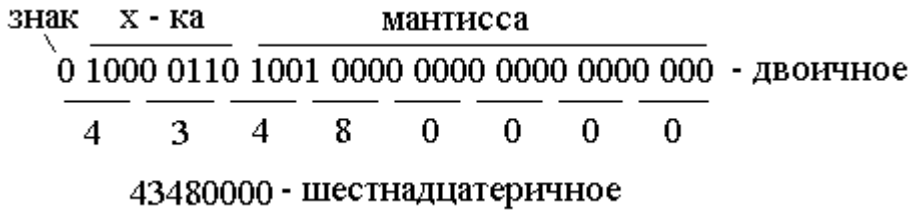


Рисунок 2.17 – Число +50 в формате с плавающей запятой и скрытой единицей мантиссы

Число -50 :

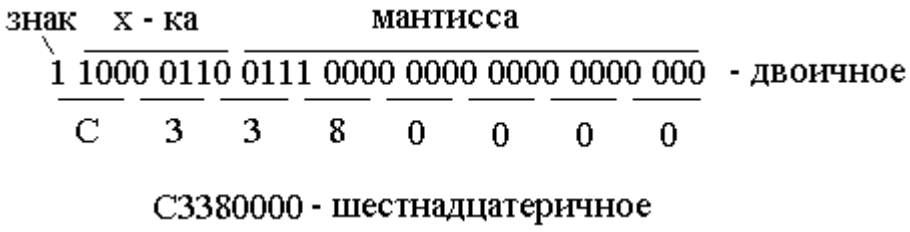


Рисунок 2.18 – Число - 50 в формате с плавающей запятой и скрытой единицей мантиссы

Аналогично можно записать любое другое число.

Контрольные вопросы

1. Что такое система счисления?
2. Каковы особенности арифметических операций в различных системах счисления?
3. Что такое машинные коды?
4. Что такое алфавитно-цифровое кодирование?

3 КОМБИНАЦИОННЫЕ УСТРОЙСТВА

Комбинационные устройства состоят из логических элементов. Выходной сигнал в любой момент времени определяется состоянием входов и пропадает после снятия входных сигналов. Наиболее известным представителем такого класса схем является **дешифратор**.

3.1 Дешифратор и шифратор

Дешифратор (декодер) служит для преобразования n-разрядного позиционного двоичного кода в единичный выходной сигнал на одном из 2^n выходов. Потому полный декодер имеет n-входов и 2^n-выходов.

Составим таблицу истинности декодера при n = 2 (рис. 3.1).

№	a	b	Y0	Y1	Y2	Y3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

Рисунок 3.1 – Таблица истинности декодера

Составим систему собственных функций (ФАЛ для каждого выхода):

$$y_0 = \bar{a} * \bar{b}$$

$$y_1 = \bar{a} * b$$

$$y_2 = a * \bar{b}$$

$$y_3 = a * b$$

По этой системе несложно построить схему дешифратора, которая представляет собой четыре двухвходовых конъюнктора (рис. 3.2):

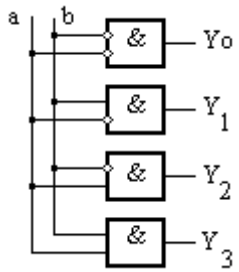


Рисунок 3.2 – Схемная реализация декодера

Условное обозначение декодера показано на рис. 3.3.

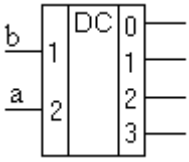


Рисунок 3.3 – Условное обозначение декодера

Декодеры выпускают в виде отдельных микросхем. Например, ИМС К155ИД3 полный декодер 4*16 (рис.3.4).

Рисунок 3.4 – Декодер К155ИД3

Обратите внимание, что четырёхразрядное двоичное число abcd подаётся на входы по старшинству (согласно своему весу).

Декодер можно синтезировать с управляющим входом, например, входом V. Для этого составим таблицу истинности (рис. 3.5).

№	V	a	b	Y0	Y1	Y2	Y3
0	1	0	0	1	0	0	0
1	1	0	1	0	1	0	0
2	1	1	0	0	0	1	0
3	1	1	1	0	0	0	1
4	0	-	-	0	0	0	0

Рисунок 3.5 – Таблица истинности декодера с входом V

Таблица истинности составлена так, что при V=1 декодер работает как обычно, а при V=0 на всех выходах будут нули. Система собственных функций:

$$y_0 = V * \bar{a} * \bar{b}$$

$$y_1 = V * \bar{a} * b$$

$$y_2 = V * a * \bar{b}$$

$$y_3 = V * a * b$$

По этим выражениям нетрудно построить схему (рис. 3.6).

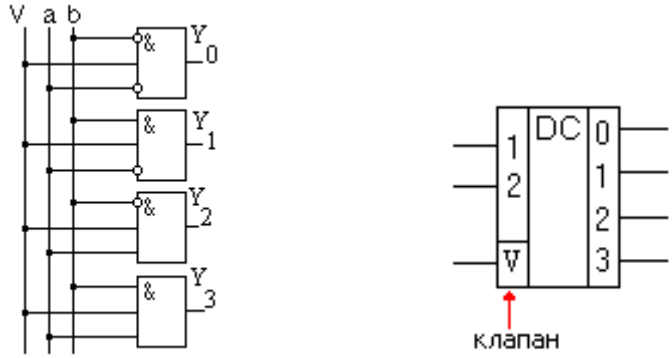


Рисунок 3.6 – Декодер с управляющим входом V (valve - клапан)

Декодер является простейшей схемой, но на его основе создают другие, более сложные комбинационные устройства.

Шифратор (кодер) выполняет функцию, обратную декодеру, то есть преобразует непозиционный (унитарный) двоичный код в n - разрядный позиционный.

Составим таблицу истинности шифратора при n = 2 (рис. 3.7).

№	Z_0	Z_1	Z_2	Z_3	a	b
0	1	0	0	0	0	0
1	0	1	0	0	0	1
2	0	0	1	0	1	0
3	0	0	0	1	1	1

Рисунок 3.7 – Таблица истинности кодера

Синтезируем шифратор. Для этого запишем систему собственных функций:

$$a = \overline{Z_0} * \overline{Z_1} * Z_2 * \overline{Z_3} + \overline{Z_0} * \overline{Z_1} * \overline{Z_2} * Z_3$$
$$b = \overline{Z_0} * Z_1 * \overline{Z_2} * \overline{Z_3} + \overline{Z_0} * \overline{Z_1} * \overline{Z_2} * Z_3 \quad ,$$

согласно которой составим схему (рис. 3.8).

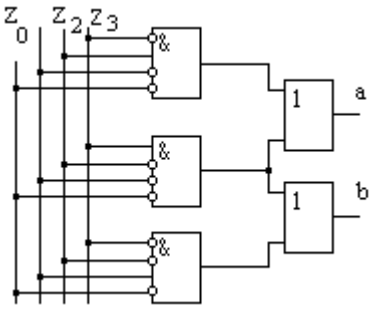


Рисунок 3.8 – Схема шифратора при n = 2

Условное обозначение шифратора и пример микросхемы приведены на рис. 3.9.

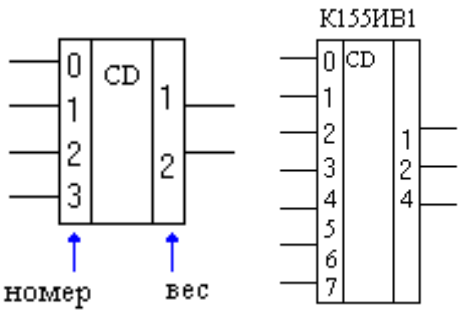


Рисунок 3.9 – Условное обозначение шифратора

Шифраторы имеются во многих сериях микросхем (К555ИВ3, 533ИВ2).

3.2 Мультиплексор и демультиплексор

Мультиплексор (коммутатор) это многовходовая комбинационная схема, служит для коммутации одного из 2^n информационных входов на выход под действием n управляющих (адресных) сигналов.

Составим схему мультиплексора при n = 2 (рис. 3.10).

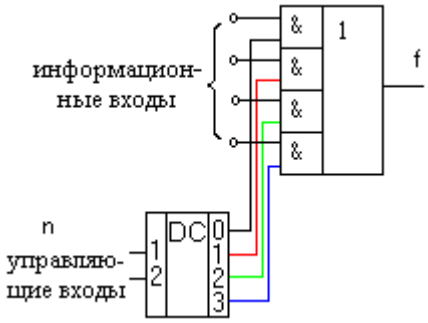


Рисунок 3.10 – Схема мультиплексора

Мультиплексор реализует дизъюнкцию элементарных конъюнкций и является универсальным устройством.

$$f = \bigvee_{i=0}^{2^n-1} X_i * Z_i \quad ,$$

где Z_i – информационный сигнал, X_i – сигналы с выхода декодера.

Условное обозначение мультиплексора представлено на рис. 3.11а.

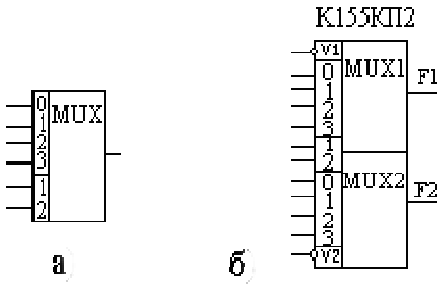


Рисунок 3.11 – Условное обозначение и пример микросхемы

Мультиплексоры выпускают как отдельные микросхемы. Например, сдвоенный четырёхканальный мультиплексор К555КП2 (рис.3.11 б), который имеет общий декодер на оба канала.

Промышленность выпускает мультиплексоры с числом адресных входов 2, 3, 4. Если этого недостаточно, то используют их каскадное включение.

Пусть требуется создать 16-канальный мультиплексор из 4-канальных. Значит адрес должен быть четырёхразрядным: $X_1\ X_2\ X_3\ X_4$ ($2^4 = 16$). Этот адрес подаётся на входы декодеров определённым образом (рис. 3.12).

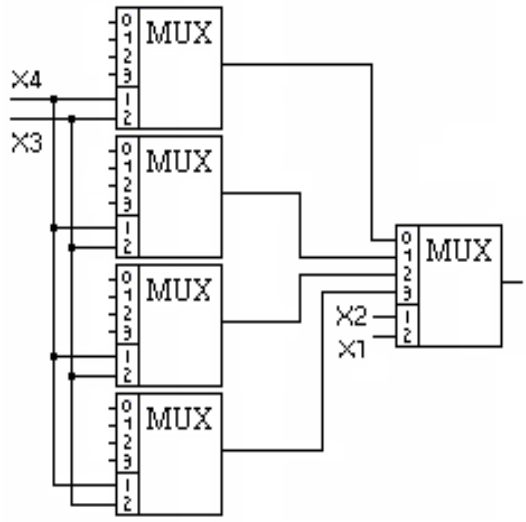


Рисунок 3.12 – Мультиплексор на 16 каналов (16 в 1)

Так как мультиплексор выполняет дизъюнкцию элементарных конъюнкций, то с его помощью можно реализовать любые функции алгебры логики. Пусть, например, задана функция алгебры логики F такой таблицей истинности (рис. 3.13).

№	a	b	c	F	D
0 1	0 0	0 0	0 1	0 1	C
2 3	0 0	1 1	0 1	1 0	$\overline{C}$
4 5	1 1	0 0	0 1	0 0	0
6 7	1 1	1 1	0 1	1 1	1

Рисунок 3.13 – Таблица истинности и входные сигналы D

Возьмем 4-канальный мультиплексор. Старшую переменную подаем на старший вход декодера, на младший вход декодера поступает следующая переменная. Младшая переменная (С) подаётся на информационные входы, но в зависимости от значения самой ФАЛ. Так, на вход № 0 , определяемый переменными a,b (00), подадим сигнал С, поскольку значения F и С совпадают. На вход № 1 (01) – сигнал «не С», поскольку значения F и С противоположны. На вход № 2 (10) – сигнал «0», поскольку значения F не зависят от С и равны нулю. На вход №3 (11) – 1. Тогда реализация исходной ФАЛ будет такой (рис. 3.14).

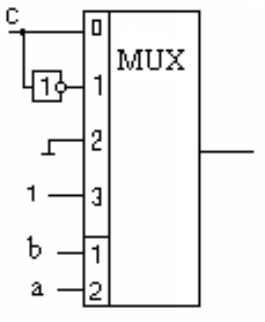


Рисунок 3.14 – Реализация ФАЛ на мультиплексоре

С точки зрения логики безразлично, какие переменные подавать на адресные входы, а какую переменную на информационные. Реализуем схему. по переменной «a».

Здесь номер информационного входа задают переменные «bc», а значение ФАЛ надо сравнивать с переменной «a». Тогда получим другую реализацию ФАЛ на мультиплексоре (рис. 3.15).

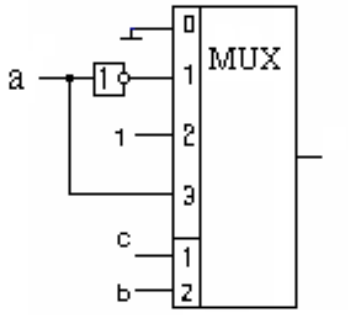


Рисунок 3.15 – Второй вариант реализации ФАЛ

Демультиплексор выполняет функцию, обратную мультиплексору, то есть коммутирует один информационный вход на один из 2^n выходов под действием n управляющих (адресных) сигналов. Составим схему демультиплексора при n = 2 (рис. 3.16).

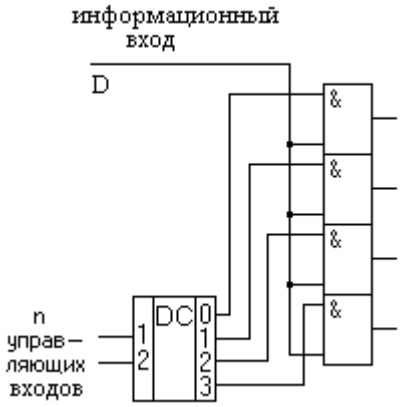


Рисунок 3.16 – Демультиплексор на 4 канала

Очевидно, что если на информационный вход (D) подать 1, то это будет дешифратор в чистом виде. Вход D можно использовать как клапан (V) (рис.3.17).

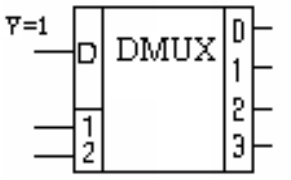


Рисунок 3.17 – Обозначение демультиплексора

Микросхемы так и называются дешифратор – демультиплексор. Например, ИМС КР531ИД14.

3.3 Сумматоры

Сумматор – это узел ЭВМ, предназначенный для сложения кодов двоичных чисел. Сумматоры делятся на последовательные (накапливающие) и параллельные (комбинационные). Накапливающие сумматоры имеют низкое быстродействие, поэтому они рассматриваться не будут. В комбинационных сумматорах слагаемые поступают на входы одновременно, а на выходе получается код суммы. После снятия слагаемых результат пропадает. Эти устройства не обладают памятью и строятся на логических элементах.

Составим таблицу истинности устройства для сложения двух одноразрядных чисел a и b (рис. 3.18).

№	a	b	p	s
0	0	0	0	0
1	0	1	0	1
2	1	0	0	1
3	1	1	1	0

Рисунок 3.18 – Таблица истинности для сложения двух цифр

Здесь p – перенос в старший разряд, s – значение суммы. Устройство, реализующее эту таблицу истинности, называют двоичным полусумматором. Его можно синтезировать по ФАЛ для каждого из выходов:

$$p = a * b$$

$$s = a * \bar{b} + \bar{a} * b$$

Составим схему на произвольных элементах (рис. 3.19).

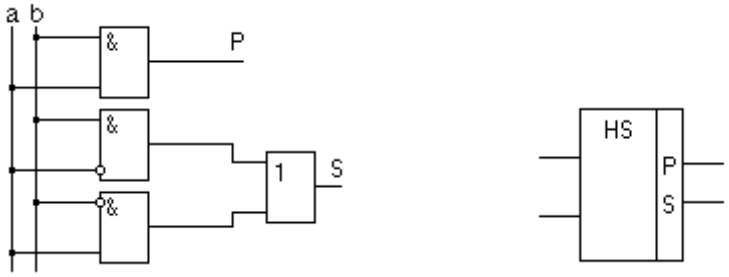


Рисунок 3.19 – Схемная реализация и условное обозначение полусумматора

При сложении многоразрядных чисел необходимо складывать три двоичных цифры в каждом разряде: два слагаемых и единицу переноса из предыдущего разряда Pi-1. Наличие этой единицы переноса несколько меняет таблицу сложения двоичных чисел (рис. 3.20).

№	a_i	b_i	P_{i-1}	P_i	S_i
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

Рисунок 3.20 – Таблица истинности для сложения трёх цифр

Система собственных функций:

для суммы: $S_i = \bar{a}_i * \bar{b}_i * P_{i-1} + \bar{a} * b * \bar{P}_{i-1} + a_i * \bar{b}_i * \bar{P}_{i-1} + a_i * b_i * P_{i-1}$

для переноса (рис. 3.21).

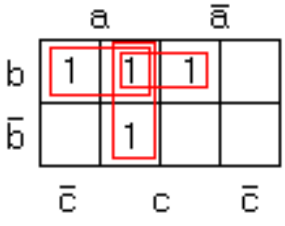


Рисунок 3.21 – Карта Карно для цепи переноса

Минимальная форма по этой карте $P_i = a_i * b_i + a_i * P_{i-1} + b_i * P_{i-1}$.

Уравнение для S_i не минимизируется. Устройство, реализующее эти ФАЛ, называется сумматор (полный сумматор). Он имеет три входа и два выхода. Цена сумматора по уравнениям составляет Ц = 25. Путем совместной минимизации уравнений S_i и P_i удается снизить цену до 20 и в таком виде выпускаются микросхемы сумматоров. Например, К155ИМ1 – полный одноразрядный сумматор (рис. 3.22).

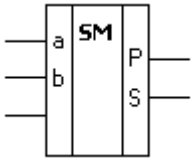


Рисунок 3.22 – Полный сумматор

Для сложения многоразрядных чисел сумматор составляют из одноразрядных. Пусть требуется сложить два четырёхразрядных двоичных числа: A и B.

$$A = a_3 a_2 a_1 a_0$$

$$B = b_3 b_2 b_1 b_0$$

Составим схему сумматора (рис. 3.23).

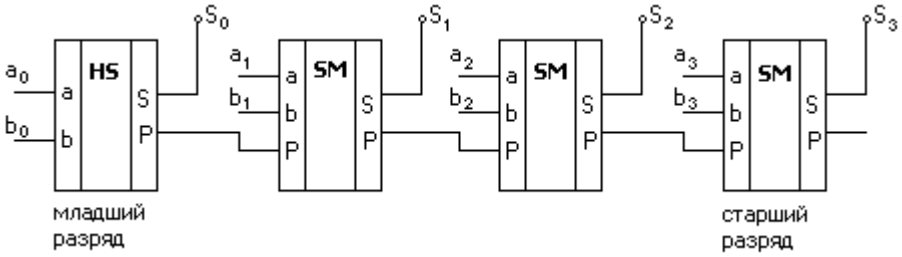


Рисунок 3.23 – Многоразрядный сумматор

Получился многоразрядный сумматор с последовательным переносом**.** Такие сумматоры выпускают в виде отдельных микросхем. Например, ИМС К155 ИМ3 четырёхразрядный сумматор с последовательным переносом. Время сложения чисел определяется временем распространения переноса и равно 55 нс (для четырёх разрядов). С ростом числа разрядов быстродействие сумматора уменьшается, так как цепь переноса последовательная.

Вспомним формулу переноса:

$$P_i = a_i * b_i + a_i * P_{i-1} + b_i * P_{i-1}$$

Найдём эти переносы:

$$P_0 = a_0 * b_0$$

$$P_1 = a_1 * b_1 + a_1 * P_0 + b_1 * P_0 = a_1 * b_1 + a_1 * a_0 * b_0 + b_1 * a_0 * b_0$$

$$P_2 = a_2 * b_2 + a_2 * P_1 + b_2 * P_1 = \cdots$$

Видно, что имея только слагаемые, можно формировать перенос в любом разряде, не дожидаясь его появления в предыдущем разряде, причём с помощью только двухуровневой схемы (один слой конъюнкторов и один дизъюнктор). Такая схема называется **схемой ускоренного переноса** (параллельного переноса). Она может быть встроена в сумматор (сумматор с параллельным переносом) или выпускаться отдельно. Например, ИМС К155 ИМ6 – четырёхразрядный сумматор с параллельным переносом. Время сложения чисел равно 27 нс.

При большом числе разрядов сложность схемы ускоренного переноса сильно возрастает. Поэтому сумматор разбивают на группы по 4 или 8 разрядов. Внутри группы выполняют параллельный перенос, а между группами параллельный или последовательный. Такие сумматоры называют сумматорами с групповым переносом.

Многоразрядный сумматор условно обозначают так (рис. 3.24).

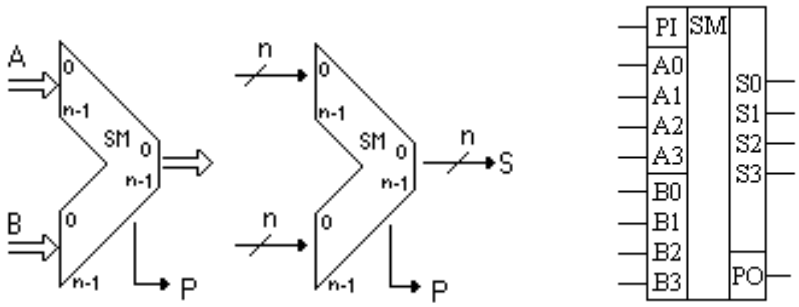


Рисунок 3.24 – Условное обозначение многоразрядного сумматора

С помощью сумматоров можно не только складывать, но и вычитать двоичные числа. При использовании дополнительных кодов операцию вычитания двух положительных чисел заменяют операцией суммирования положительного и отрицательного чисел, при этом получение дополнительного кода числа является элементарной операцией. Для этого необходимо проинвертировать число и прибавить к нему в младший разряд 1.

Схема вычитателя числа A из числа B приведена на рисунке 3.25, а схема вычитателя числа B из числа A на рисунке 3.26.

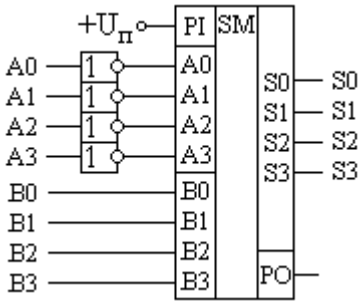


Рисунок 3.25. Схема вычитателя числа A из числа B

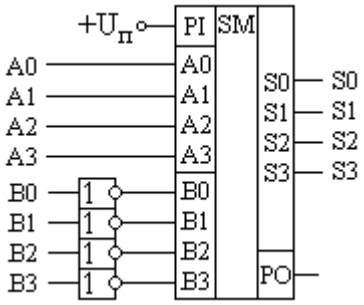


Рисунок 3.26. Схема вычитателя числа B из числа A.

Схема инкремент/декремент

Возьмём три полусумматора и соединим их следующим образом (рис.3.27).

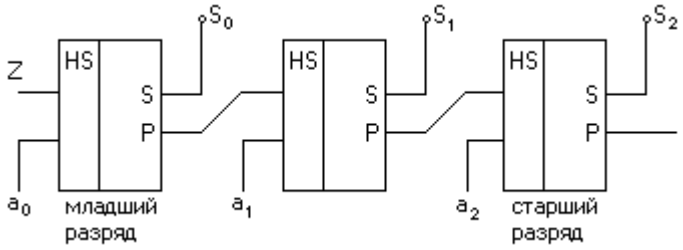


Рисунок 3.27 – Схема инкремент / декремент

Подавая на управляющий вход Z ноль или единицу, проанализируем состояние выхода при различных входных сигналах (рис. 3.28).

Z	Входной код a_2	a_1	a_0	Выходной код S_2	S_1	S_0
0	1	1	0	1	1	0
0	0	0	1	0	0	1
1	1	1	0	1	1	1
1	0	0	1	0	1	0

Рисунок 3.28 – Соответствие сигналов схемы инкремент / декремент

Если на вход Z поступает 0, то число на выходе будет без изменений. Если на вход Z подать 1, то эта единица добавляется к младшему разряду числа (инкремент +1).

Если числа на входе и выходе проинвертировать, то мы получаем схему декремент (декремент -1).

	A_2 A_1 A_0			S_2 S_1 S_0	
Число на входе	1 1 0			1 0 1	получился ответ
Выполняем инверсию	↓			↑	
	0 0 1	инкремент	→	0 1 0	

Эта схема самостоятельного значения не имеет, но широко используется как составная часть арифметико-логических устройств.

3.4 Преобразователи кодов

Наиболее широко преобразователи кодов известны применительно к цифровым индикаторам. Например, преобразователь четырехразрядного позиционного двоичного кода в десятичные цифры.

Имеется семисегментный индикатор и с его помощью требуется высветить десять цифр (рис. 3.29).

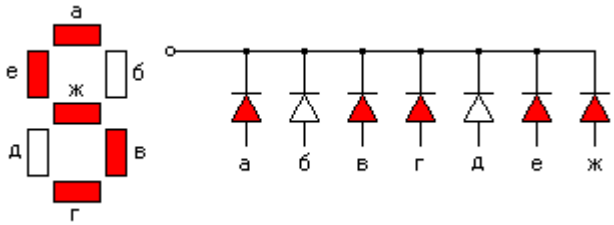


Рисунок 3.29 – Обозначение индикатора

Очевидно, что двоичный код должен иметь не менее четырёх разрядов ($2^4 = 16$, что больше 10).

Составим таблицу истинности работы такого преобразователя (рис. 3.30).

Цифра	Код 8-4-2-1				а	б	в	г	д	е	ж
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Рисунок 3.30 – Таблица преобразования для индикатора

Несложно составить систему собственных функций для всех выходов, т.е. СДНФ, минимизировать её и составить схему. Так и делаем. Преобразователь кода обозначаем следующим образом (рис. 3.31).

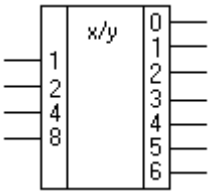


Рисунок 3.31 – Условное обозначение преобразователя кода

Промышленность выпускает преобразователи кодов, например, К155ПР6 и К155ПР7 – преобразователи двоичного кода в двоично-десятичные цифры и обратно.

Второе, более широкое применение преобразователей кодов это аппаратная защита информации, передаваемой по открытым каналам связи (рис. 3.32).

Рисунок 3.32 – Аппаратное преобразование кода

Выполняем перестановку передаваемых битов в прямом преобразователе, а на приёмном конце устанавливаем преобразователь с обратным законом перестановки. Получается верный код, хотя в линии код неверный. Степень защищённости такого способа невысокая, так как мало число перестановок (n!), поэтому используют комбинацию DC – CD (рис. 3.33).

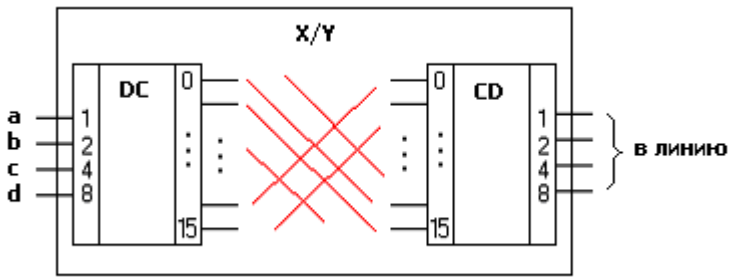


Рисунок 3.33 – Аппаратное преобразование кода DC - CD

Здесь число перестановок значительно больше и равно $2^n!$, так как у декодера 2^n выходов.

На приёмном конце устанавливаем узел с обратным законом соединений. Такой узел находит применение на практике.

3.5 Шинный формирователь

Вспомним базовый элемент ТТЛ (рис. 3.34).

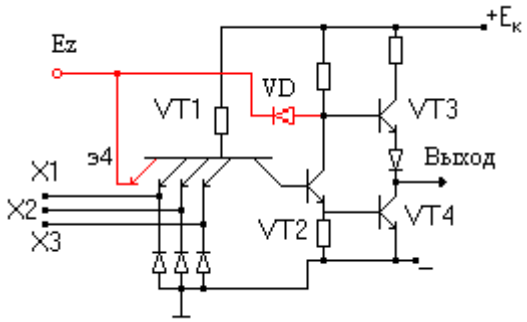


Рисунок 3.34 – Схема базового элемента ТТЛ

Эта схема выполняет операцию ЗИ-НЕ (операцию Шеффера). Когда сигнал на любом из входов X1, X2 или X3 равен нулю, то VT1 открыт, VT2 закрыт, VT4 закрыт. VT3 открыт и на выходе будет 1. Если на входах все единицы или ничего не подано, то ток базы VT1 течет в базу VT2, открывает его и насыщает , VT4 открыт и насыщен, VT3 закрыт. Таким образом, выходная шина подключается то к плюсу источника Ек, то к его минусу, и потенциал на ней либо ноль, либо единица.

Изменим схему. Добавим диод VD и еще один эмиттер э4 в транзисторе VT1, вход обозначим Ez. Если на вход Ez ничего не подавать, что эквивалентно подаче 1, то он не влияет на работу логического элемента. Если на вход Ez подать 0, то есть заземлить его, то транзистор VT1 будет открыт по эмиттеру э4, при этом будут заперты VT2 и VT4, потенциал базы VT3 также будет равен 0, поэтому VT3 также будет заперт, то есть выходная шина логического элемента будет оторвана от плюса и от минуса источника. Она находится в так называемом высокоимпедансном (третьем состоянии или Z-состоянии).

Элементы с тремя состояниями обозначают треугольником (рис. 3.35).

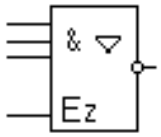


Рисунок 3.35 – Элемент с тремя состояниями

Тристабильные элементы можно соединять параллельно по выходу, как показано на рис. 3.36.

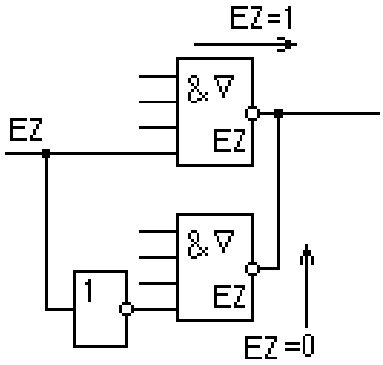


Рисунок 3.36 – Включение тристабильных элементов

Элементы с Z-состоянием используются для двунаправленной передачи данных (рис. 3.37).

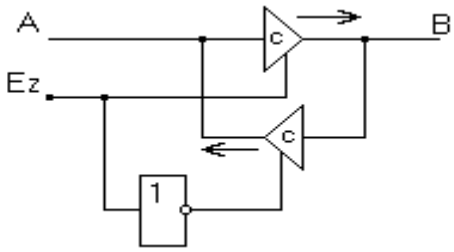


Рисунок 3.37 – Двунаправленная передача данных

Они выполняют функцию буферизации сигнала, то есть усиление его по мощности, поэтому их иногда называют магистральные усилители или шинные формирователи.

Рассмотрим функциональную схему шинного формирователя К580ВА86. Это восьмиразрядная интегральная схема средней степени интеграции, выполненная по ТТЛ – технологии (рис. 3.38).

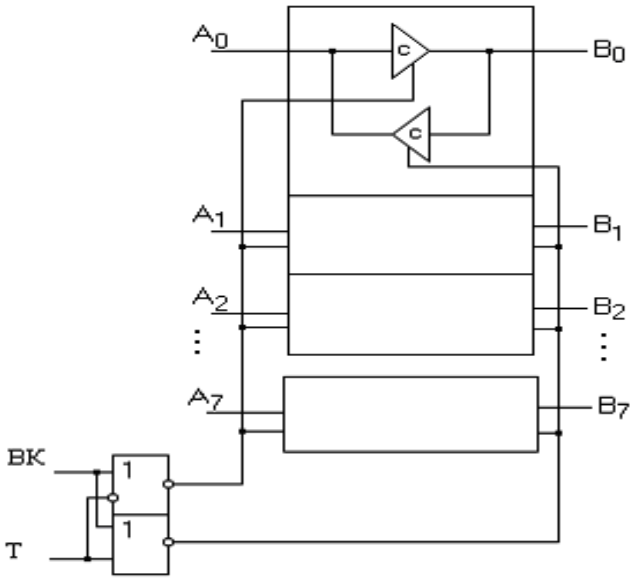


Рисунок 3.39 – Шинный формирователь

Вход ВК – выбор кристалла. Когда сигнал ВК = 1, все элементы находятся в третьем состоянии . Шина А отключена от шины В. Если сигнал ВК=0, то тогда при Т = 1 – направление передачи информации от А к В, а при Т =0 от В к А.

С помощью таких формирователей к одной обшей шине можно поочерёдно подключать множество различных устройств, что и используется в микропроцессорах и системах.

Контрольные вопросы

1. Что такое шифратор и дешифратор?
2. Что такое мультиплексор и демультиплексор?
3. Что такое сумматор?
4. Что такое шинный формирователь?

4 ПОСЛЕДОВАТЕЛЬНОСТНЫЕ УСТРОЙСТВА

Последовательностные устройства – это устройства с памятью. В них выходной сигнал определяется не только текущим состоянием входа, но и рядом предыдущих значений. Простейшее последовательностное устройство это триггер. Его особенностью является способность бесконечно долго находится в одном из двух устойчивых состояний. Приняв одно состояние за ноль, другое за единицу, можно считать, что триггер хранит один бит информации.

Триггер характеризуют:

- число входов - $P \le 3$;
- число выходов - один Q (хотя можно получить и $\overline{Q}$);
- число внутренних состояний два Q и $\overline{Q}$;
- характеристическое уравнение $Q(t+1) = \delta(Z(t), Q(t))$,

где t - текущее время;

t+1 - следующий момент времени;

Z(t) - входной сигнал в текущий момент времени t;

Q(t) - состояние триггера в текущий момент времени;

Q(t+1) - состояние триггера, в которое он перейдет в момент времени (t+1).

Триггеры классифицируют по ряду признаков:

1) по логическому функционированию (RS-триггер это триггер с установочными входами; Т-триггер это счетный триггер; D-триггер это триггер передачи (задержки); JK и DV-триггеры это универсальные триггеры; RST, TV и другие это комбинированные триггеры);

2) по способу записи информации в триггер (синхронные и асинхронные). В синхронных триггерах добавлен управляющий вход С, который не является логическим. Сигнал на его входе разрешает перейти триггеру в нужное состояние;

3) по числу ступеней триггеры делятся на одноступенчатые (однотактные) и двухступенчатые (двухтактные).

Общее количество триггеров очень велико: $W = 5^{2^P}$, где p – число информационных входов, если p=1, то W=25, если p=2, то W=625. Но это теоретическое число, так как теоретически триггер может находиться в одном из пяти возможных состояний: Q, $\overline{Q}$, * , 1 ,0 (* неопределённое состояние). Большинство из этих триггеров не имеют физического смысла. Например, при P = 1 в настоящее время синтезированы и применяются только два типа. При P = 2 имеют физический смысл только 24, из них синтезировано 8, среди которых два универсальных. С их помощью решаются все возникающие технические задачи. В других триггерах потребности пока не возникало.

4.1 Асинхронные триггеры

Основным триггером, на котором базируются все остальные, является RS-триггер. Он имеет два логических входа: R - установка 0 (от слова Reset), S - установка 1 (от слова Set).

Простейший RS-триггер состоит из двух логических элементов, охваченных перекрёстной, положительной обратной связью (рис. 4.1).

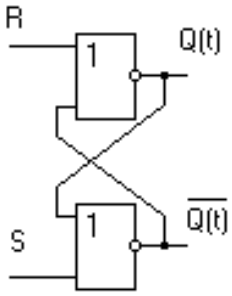


Рисунок 4.1 – Простейший триггер

Один из выходов называют прямым, а другой – обратным. Состояние триггера определяется состоянием прямого выхода.

Составим таблицу истинности RS-триггера, учитывая, что имеем три независимых переменных R, S, Q(t) (рис. 4.2).

Такт t			Такт t+1	Пояснения
R	S	Q(t)	Q(t+1)	
0	0	0	0	Режим хранения информации R=S=0
0	0	1	1	
0	1	0	1	Режим установки единицы S=1
0	1	1	1	
1	0	0	0	Режим установки нуля R=1
1	0	1	0	
1	1	0	*	R=S=1 запрещенная комбинация
1	1	1	*	

Рисунок 4.2 – Таблица истинности простейшего триггера

Характеристическое уравнение триггера запишем по единичным значениям сигнала Q (t + 1). Для этого заполним карту Карно (рис. 4.3).

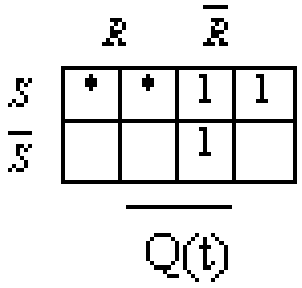


Рисунок 4.3 – Карта Карно

Получаем минимальную форму $Q(t+1) = S + \overline{R} * Q(t)$ характеристическое уравнение RS-триггера.

Такой триггер условно изображается в следующем виде (рис. 4.4).

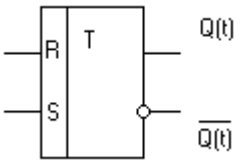


Рисунок 4.4 – Обозначение RS - триггера

Очевидно, что по характеристическому уравнению триггера его схему можно составить в любом базисе. Составим схему на элементах 2И-НЕ. Для этого в уравнении нужно избавиться от дизъюнкции. Применим двойное отрицание

$$Q(t+1) = \overline{\overline{S + \overline{R} * Q(t)}} = \overline{\overline{S} * \overline{\overline{R} * Q(t)}}$$

С учетом того, что сигналы Q(t) и Q(t + 1) одна и та же физическая точка схемы, но в разные моменты времени, эти точки соединяем, как показано на рис. 4.5.

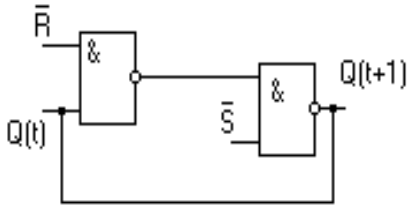


Рисунок 4.5 – Построение триггера

Получился так называемый дуальный триггер, в котором эффективным значением входного сигнала является нуль (рис. 4.6).

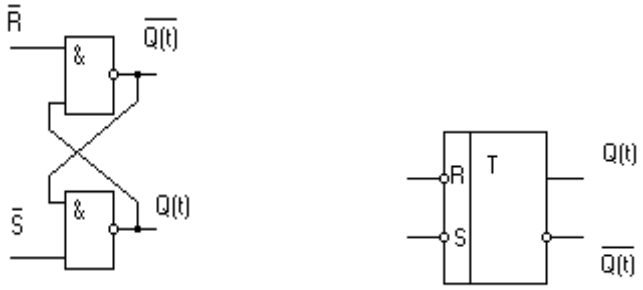


Рисунок 4.6 – Триггер RS на элементах 2И-НЕ

В нем комбинация входных сигналов R=S=0 является запрещенной, а R=S=1 режим хранения. Это триггер с инверсным управлением.

Разновидностью RS-триггера являются: S-триггер, который при R = S = активному уровню переходит в единичное состояние; R - триггер, который при R = S = активному уровню переходит в нулевое состояние; E - триггер, который при R = S = активному уровню сохраняет предыдущее состояние Q(t+1)=Q(t).

Составим словарь переходов RS-триггера, который показывает, какие сигналы следует подавать на входы, чтобы перевести триггер в нужное состояние. Его заполняют на основании таблицы истинности. В нем всегда четыре строки (для любых триггеров). Словарь RS- триггера приведён на рис. 4.7.

Q(t)	R	S	Q(t+1)
0	-	0	0
0	0	1	1
1	1	0	0
1	0	-	1

Рисунок 4.7 – Словарь переходов RS-триггера

Здесь прочерк означает безразличное состояние входа.

Т-триггер. Счетный триггер (от слова topple - переключать). Имеет один информационный вход Т. Функционирует согласно следующей таблице истинности (рис. 4.8).

Такт t Такт t+1

T	Q	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Рисунок 4.8 – Таблица истинности Т-триггера

Когда сигнал на входе Т=0, то триггер сохраняет своё состояние. Когда на входе Т=1, то триггер меняет состояние на противоположное. Условное изображение Т-триггера показано на рис. 4.9.

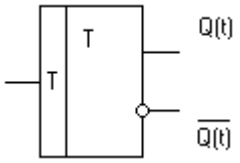


Рисунок 4.9 – Условное изображение Т-триггера

Характеристическое уравнение $Q(t+1) = \overline{T} * Q(t) + T * \overline{Q(t)} = T \oplus Q(t)$.

Счетный триггер выполняет сложение по модулю два входного сигнала и внутреннего состояния триггера Q (t).

Составим схему Т-триггера на основе RS-триггера. Для этого требуется создать комбинационную схему (КС), преобразующую сигнал Т в сигналы R и S, как показано на рис. 4.10.

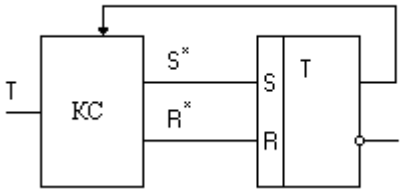


Рисунок 4.10 – Схема для синтеза Т - триггера

Воспользуемся словарём переходов RS-триггера и проставим в таблице истинности (рис. 4.11) такие сигналы $R^* S^*$, чтобы обеспечить требуемые переходы RS-триггера

Такт t				Такт t+1
T	Q	R^*	S^*	Q(t+1)
0	0	-	0	0
0	1	0	-	1
1	0	0	1	1
1	1	1	0	0

Рисунок 4.11 – Таблица истинности для синтеза Т-триггера

По единичным значениям сигналов R^* и S^* запишем логические функции $R^* = T * Q$ и $S^* = T * \overline{Q}$, по которым легко составить схему (рис. 4.12).

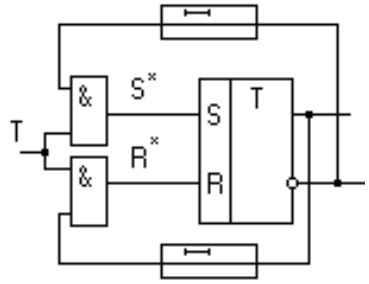


Рисунок 4.12–Схемная реализация Т-триггера на основе RS-триггера

Элементы задержки в цепи обратной связи на время, равное длительности сигнала Т, необходимы, так как логические функции не учитывают фактор времени. Сигнал с выхода не должен поступить на вход раньше, чем закончится сигнал Т. Такая задержка в реальных триггерах обеспечивается путём блокировки цепи обратной связи на время действия входного сигнала. Таким образом, получили счётный триггер. Словарь переходов Т-триггера (рис. 4.13).

Q(t)	T	Q(t+1)
0	0	0
0	1	1
1	1	0
1	0	1

Рисунок 4.13 – Словарь переходов Т - триггера

JK-триггер. Триггер имеет два информационных входа: J - установка единицы (Jerk – включение) и K - установка нуля (Kill - выключение).

В этом триггере, в отличие от RS-триггера, нет запрещённых комбинаций входных сигналов. Комбинация J = K = 1 переводит триггер в противоположенное состояние.

Составим таблицу истинности JK-триггера (рис. 4.14) и реализуем его на RS-триггере.

Такт t					Такт t+1
J	K	Q(t)	R^*	S^*	Q(t+1)
0	0	0	-	0	0
0	0	1	0	-	1
0	1	0	-	0	0
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	-	1
1	1	0	0	1	1
1	1	1	1	0	0

Рисунок 4.14 – Таблица истинности JK - триггера

По единичным значениям сигнала Q(t+1) составим карту Карно (рис. 4.15).

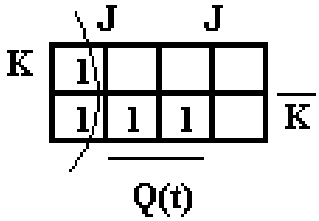


Рисунок 4.15 – Карта Карно JK-триггера

И получим характеристическое уравнение $Q(t+1) = J * \overline{Q(t)} + \overline{K} * Q(t)$.

Построим схему JK-триггера на основе RS- триггера (рис. 4.16), для этого воспользуемся словарем перехода RS- триггера и в таблицу истинности запишем требуемые сигналы R^* и S^*.

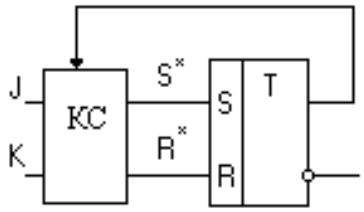


Рисунок 4.16 – JK- триггер на основе RS - триггера

Получаем систему собственных функций:

$$R^* = \overline{J} * K * Q(t) + J * K * Q(t) = K * Q(t)$$

$$S^* = J * \overline{K} * \overline{Q}(t) + J * K * \overline{Q}(t) = J * \overline{Q}(t)$$

И по ней составляем схему (рис. 4.17).

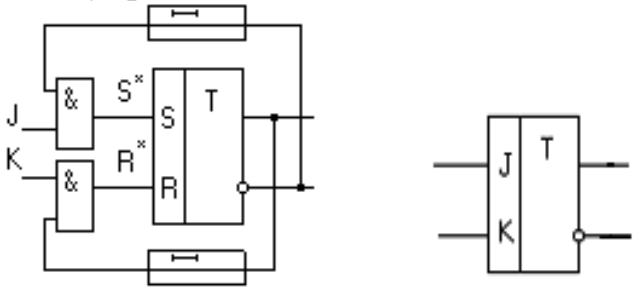


Рисунок 4.17 – Схема и обозначение JK - триггера

Словарь переходов JK-триггера (рис. 4.18).

Q(t)	J	K	Q(t+1)
0	0	-	0
0	1	-	1
1	-	1	0
1	-	0	1

Рисунок 4.18 – Словарь переходов JK-триггера

Из словаря переходов видно, что JK-триггер предоставляет большие возможности по комбинации входных сигналов, чем RS - триггер, поэтому схемные решения на его основе получаются более компактными.

Триггер Шмитта

Имеет одно внутреннее состояние, которое заранее известно, и поэтому он не может хранить информацию. Триггер используется для формирования сигналов с крутыми фронтами из сигналов произвольной формы. Его схему представляют так (рис. 4.19).

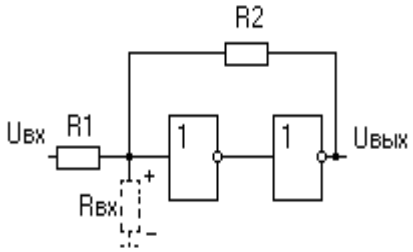


Рисунок 4.19 – Триггер Шмитта на инверторах

Простейший триггер Шмитта это соединение двух инверторов, охваченных обратной связью. Рассмотрим его амплитудную характеристику (рис. 4.20).

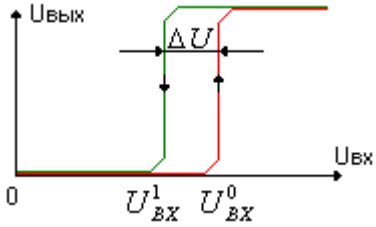


Рисунок 4.20 – Амплитудная характеристика триггера Шмитта

Переключение схемы при снижении входного напряжения происходит при меньшем уровне $U_{вх}$, так как входной ток поддерживается ещё и единичным выходным сигналом. ΔU – напряжение гистерезиса. Величина гистерезиса зависит от соотношения резисторов и при $R_1 \approx 0{,}1R_2$ обычно находится в пределах 1 … 1,8 вольт.

В условном графическом обозначении элемента показывают гистерезис (рис. 4.21).

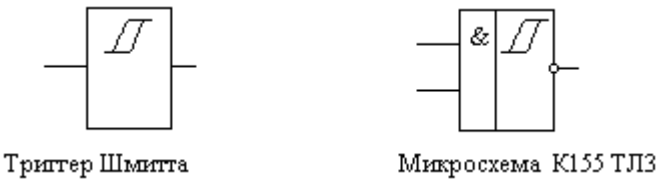


Рисунок 4.21 – Условное обозначение триггера Шмитта

На выходе триггера Шмитта формируются сигналы с короткими фронтами из входных сигналов любой сложной формы (рис 4.22).

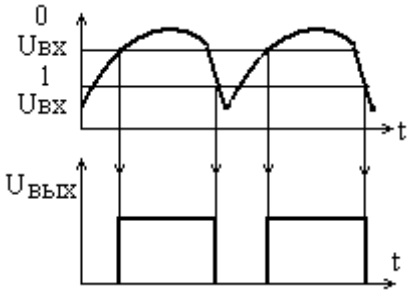


Рисунок 4.22 – Работа триггера Шмитта

Такие элементы используют для создания пороговых устройств, в системах автоматики и при построении помехоустойчивых комбинационных схем.

4.2 Синхронные триггеры

В них добавлен управляющий вход, который не является логическим, он только разрешает перейти триггеру в нужное состояние. Словари переходов и таблицы истинности при этом не меняются. Например, синхронный RS- триггер (рис. 4.23).

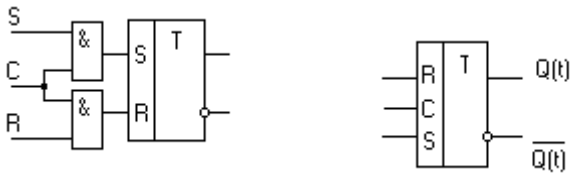


Рисунок 4.23 – Синхронный RS-триггер

Синхронный JK-триггер (рис. 4.24).

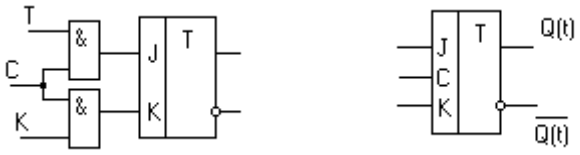


Рисунок 4.24 – Синхронный JK-триггер

Объединение входов J и K даёт T-вход (рис. 4.25), то есть получаем синхронный счётный триггер.

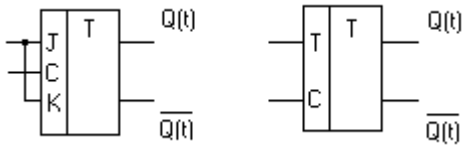


Рисунок 4.25 – Синхронный Т-триггер

D-триггер. Триггер имеет один информационный вход D (Delay – задержка, Drive – передача). D-триггеры бывают только **синхронные.** Состояние информационного входа D передается на выход под действием синхроимпульса.

Составим таблицу истинности D-триггера и синтезируем его на базе RS-триггера (рис. 4.26).

Такт t				Такт t+1
D	Q(t)	R^*	S^*	Q(t+1)
0	0	-	0	0
0	1	1	0	0
1	0	0	1	1
1	1	0	-	1

Рисунок 4.26 – Таблица истинности D-триггера

Состояние входа D передается на выход без учета внутреннего состояния триггера. Его характеристическое уравнение Q (t+1) = D$*\overline{Q(t)}$ + D$*$ $Q(t)$ = D.

Построим D-триггер на основе RS-триггера (рис. 4.27) . Поступаем аналогично, как в JK и Т-триггерах.

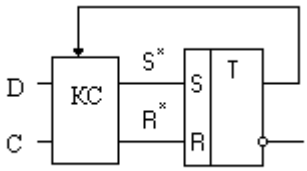


Рисунок 4.27 – Синтез D-триггера на основе RS-триггера

Записываем по единицам:

$$R^* = \overline{D} * Q(t) + \overline{D} * \overline{Q(t)} = \overline{D}$$

$$S^* = D * \overline{Q(t)} + D * Q(t) = D$$

Составляем схему триггера (рис. 4.28).

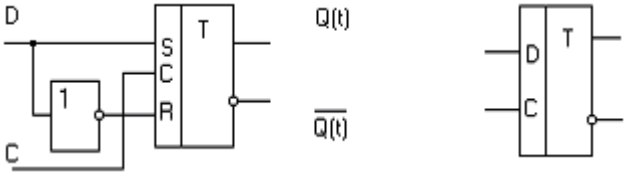


Рисунок 4.28 – D-триггер и его изображение

Назначение D-триггера передавать входной сигнал. При этом происходит задержка во времени, но она зависит от момента появления сигнала на входе D и может быть различной. Проследим это на эпюрах (рис. 4.29).

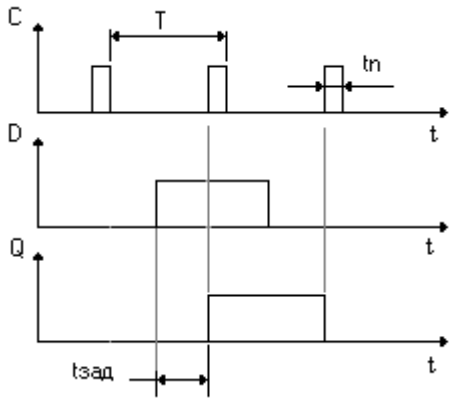


Рисунок 4.29 – Задержка сигнала в D-триггере

Максимальное время задержки равно длительности паузы между синхроимпульсами.

$$T_{ЗАД} \le (T - t_n)$$

DV-триггер. Это универсальный триггер. Он имеет два информационных входа D и V (valve - клапан), получается из D-триггера путём добавления конъюнктора на входе (рис.4.30). Входы V и C эквивалентны.

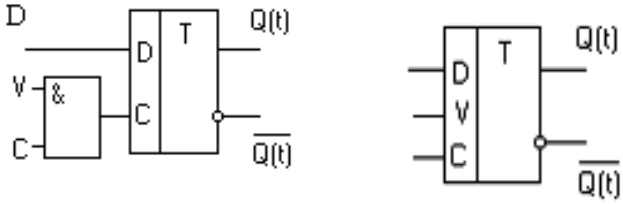


Рисунок 4.30 – DV - триггер

В отличие от JK-триггера здесь комбинация входных сигналов D=V=1 переводит триггер в единичное состояние, то есть этот триггер имеет другие функциональные возможности. Составим его таблицу истинности (рис. 4.31).

Такт t			Такт t+1
D	V	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рисунок 4.31 – Таблица истинности DV-триггера

Составим характеристическое уравнение по карте Карно (рис. 4.32).

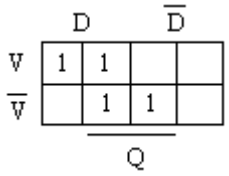


Рисунок 4.32 – Карта Карно DV-триггера

Получаем $Q(t+1) = D*V + \overline{V}*Q(t)$.

Если D = V = 1, то Q (t +1) = 1 , при D = V = 0, Q (t +1) = Q (t).

Составим словарь переходов DV-триггера на основании таблицы истинности (рис. 4.33).

Q(t)	D	V	Q(t+1)
0	0	-	0
	-	0	
0	1	1	1
1	0	1	0
1	1	-	1
	-	1	

Рисунок 4.33 – Словарь переходов DV-триггера

Видно, что DV-триггер предоставляет другие возможности по выбору управляющих сигналов, в отличие от JK-триггера, поэтому его использование в ряде случаев более предпочтительно.

Двухтактные триггеры (MS-триггеры)

Двухтактный триггер состоит из двух ступеней – двух триггеров: первая ступень M (master хозяин), а вторая ступень S (slave помощник). Тип двухтактного триггера определяется типом триггера первой ступени. Возьмём двухтактный RS-триггер (рис. 4.34).

Если сигнал C=1, то первая ступень находится в режиме приема информации, а вторая в режиме хранения, так как сигнал синхронизации на её входе равен нулю. Если C=0, то первый триггер переходит в режим хранения, а второй в режим приема информации и копирует состояние первого триггера. Именно в этот момент информация появляется на выходе триггера (Q).

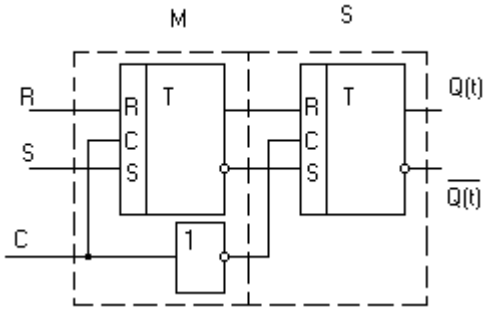


Рисунок 4.34 – Двухтактный RS-триггер

Двухтактный триггер обозначается двумя буквами Т (рис. 4.35).

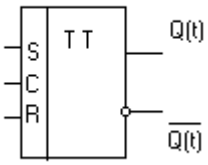


Рисунок 4.35 – Изображение двухтактного RS-триггера

Зачем нужны двухтактные триггеры? Во-первых, они имеют высокую помехоустойчивость, а во-вторых, с помощью двухтактного D-триггера можно задержать сигнал на время, равное периоду синхронизации (в однотактных только на время паузы). Это поясняется эпюрами рис. 4.36 . Здесь время задержки $t_{ЗАД} \le T$.

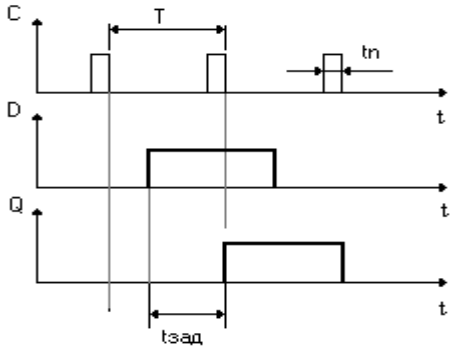


Рисунок 4.36 – Задержка сигнала в двухтактном D-триггере

Имея двухтактный D-триггер, несложно получить асинхронный Т-триггер (рис. 4.37).

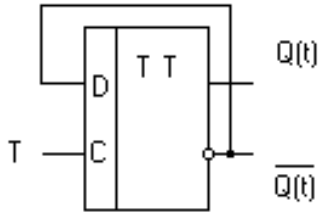


Рисунок 4.37 – Асинхронный Т-триггер на основе двухтактного D-триггера

Вообще любой триггер можно синтезировать на любом другом типе триггера, воспользовавшись словарём переходов последнего.

4.3 Способы управления триггерами

В зависимости от того, какая часть синхросигнала вызывает опрокидывание триггера, все триггеры делят на статические и динамические. Статические триггеры управляются уровнем сигнала - это потенциальные триггеры. Они опрокидываются, когда уровень синхросигнала выше некоторого порога $U_{пор}$ (рис.4.38).

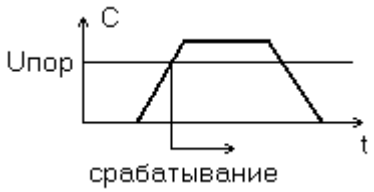


Рисунок 4.38 – Синхросигнал триггера

Такие триггеры воспринимают любую информацию со своих входов, если $C \geq U_{пор}$, то есть триггер «прозрачен» для входной информации. В некоторых случаях это неудобно, поэтому применяют триггеры с динамическим управлением, в которых информация воспринимается только во время перехода синхросигнала $0 \rightarrow 1$ или наоборот. Все остальное время информационные входы блокированы, как в двухтактных триггерах ($0 \rightarrow 1$ прямое динамическое управление; $1 \rightarrow 0$ обратное динамическое управление).

Обозначения различных синхровходов показаны на рис. 4.39.

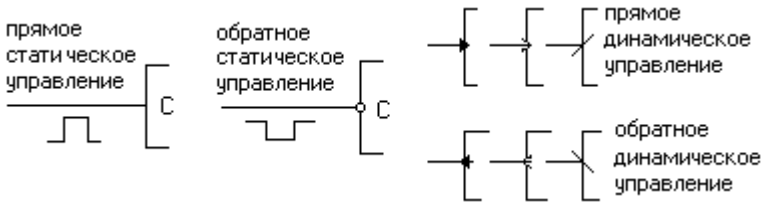
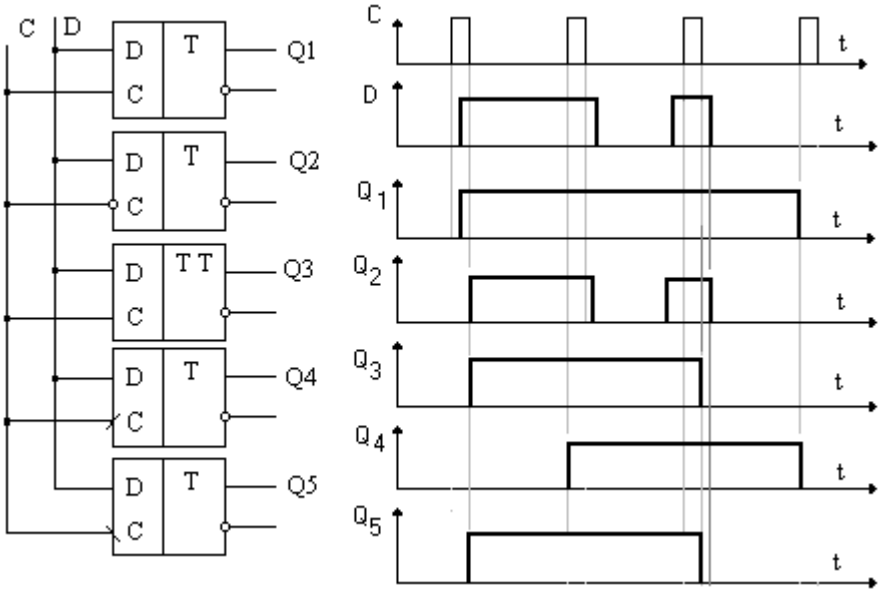


Рисунок 4.39 – Обозначения синхровходов триггеров

Синхровход в двухтактных триггерах всегда обозначается как статический.

Рассмотрим влияние типа управления на выходные сигналы триггера. Для этого возьмем пять D-триггеров с различными типами синхровходов, как показано на рис. 4.40.



а б

Рисунок 4.40 – Влияние типа управления на выходные сигналы D-триггера

Видно, что реакция триггера с обратным динамическим управлением и двухтактного триггера одинакова. Очевидно, что помехоустойчивость триггеров с динамическим управлением выше, чем статических. В динамических триггерах помеха может пройти на выход только если она по времени накладывается на фронт синхроимпульса.

Контрольные вопросы

1. Что такое RS-триггер и JK-триггер?
2. Что такое T-триггер и D-триггер?
3. Что такое триггер Шмита, DV-триггер, MS-триггер?
4. Какие существуют способы управления триггерами?

5 ЭЛЕМЕНТЫ ЦИФРОВЫХ УСТРОЙСТВ

5.1 Уровни представления вычислительных устройств

Вычислительные устройства относятся к классу сложных устройств. В зависимости от минимальной неделимой единицы различают четыре уровня их представления:

1. Уровень электрических схем. Здесь минимальной неделимой единицей является радиокомпонент (диод, резистор, конденсатор, транзистор и т.д.). Для анализа и синтеза этих схем используют дифференциальные уравнения для токов и напряжений, законы Кирхгофа. Такой уровень представления допустим, когда число компонентов не превышает 10 … 20.

2. Уровень логических схем. Здесь минимальной неделимой единицей являются логические и запоминающие элементы, каждый из которых содержит до тысячи радиокомпонентов. Для анализа и синтеза таких схем используют язык булевой алгебры. Этот уровень описания допустим при числе элементов в пределах 200.

3. Уровень операционных схем. Операционная схема – это совокупность устройств и связей между ними с точностью до микроопераций. Микрооперация – это элементарная функциональная операция, выполняемая за один такт под действием одного управляющего сигнала. Минимальный неделимый элемент – операционный элемент (сумматор, дешифратор, мультиплексор, регистр, счётчик и другие). Операционные схемы описывают языком микроопераций.

4. Уровень структурных схем. Структурная схема – это совокупность устройств, выполняющих законченные функции по приему, переработке и хранению информации. Она показывает комплектацию машины, архитектуру (стиль постройки) и все основные связи.

5.2 Структура цифрового устройства

Для анализа и синтеза любое цифровое устройство удобно представить в виде исполнительной и управляющей частей (рис. 5.1).

Операционный блок состоит из операционных элементов, набор которых может быть одним и тем же для выполнения разных алгоритмов обработки.

Под действием кода операции управляющий блок вырабатывает последовательность управляющих сигналов X_i в течение необходимого числа тактов, порождающих в операционном блоке нужную последовательность микроопераций.

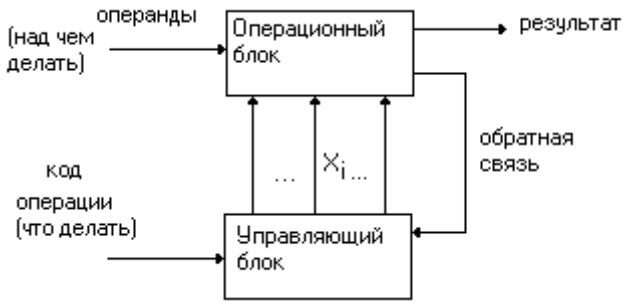


Рисунок 5.1 – Структура цифрового устройства

Набор микроопераций в каждом вычислителе свой и колеблется от нескольких единиц до нескольких десятков. Наиболее часто используются такие микрооперации:

1. Установка <слово> : = <число>
2. Передача <слово> : = <слово1>
3. Счет <слово>: = <слово> ± 1
4. Сложение <слово> : = <слово1> + <слово2>
5. Дизъюнкция <слово> : = <слово1> V <слово2>

6. Конъюнкция <слово> : = <слово1> ∧ <слово2>

7. Инверсия <слово> : = $\overline{<слово1>}$ (поразрядная инверсия двоичных разрядов)

8. Логическая неравнозначность (или сумма по модулю 2)

<слово>: = <слово1> ⊕ <слово2>

9. Логическая равнозначность

<слово> := <слово1> ∞ <слово2>

где ∞ подобие.

10. Сдвиг < слово > : = Rm < слово1 > R – сдвиг вправо на m бит

Lm < слово1 > L – сдвиг влево на m бит

Эти и другие микрооперации выполняются операционными элементами.

5.3 Операционные элементы

Наряду с рассмотренными ранее комбинационными операционными элементами (сумматор, дешифратор, мультиплексор, преобразователь кода, цифровой компаратор и др.) имеется большая группа последовательностных операционных устройств, таких как регистр, счетчик, арифметико-логические устройства, и др. Наиболее широко распространёнными из них являются регистры.

5.3.1 Регистры

Регистр представляет собой набор триггеров, охваченных общей цепью синхронизации. Триггеры называют разрядами регистра.

По способу ввода / вывода информации различают регистры:

1) параллельные (регистры хранения, информация вводится и выводится одновременно по всем разрядам);

2) последовательные (регистры сдвига, информация, бит за битом «проталкивается» через регистр и выводится так же последовательно, бит за битом);

3) комбинированные (параллельный ввод, последовательный вывод или наоборот).

По способу представления информации регистры делятся на однофазные и парафазные.

В однофазном регистре информация представляется в прямом или обратном (инверсном) виде, в парафазных и в том, и в другом виде одновременно, то есть в информации всегда имеются нули и единицы.

На рис. 5.2 изображён трёхразрядный регистр хранения на RS-триггерах (это парафазный регистр, так как есть Q и $\overline{Q}$).

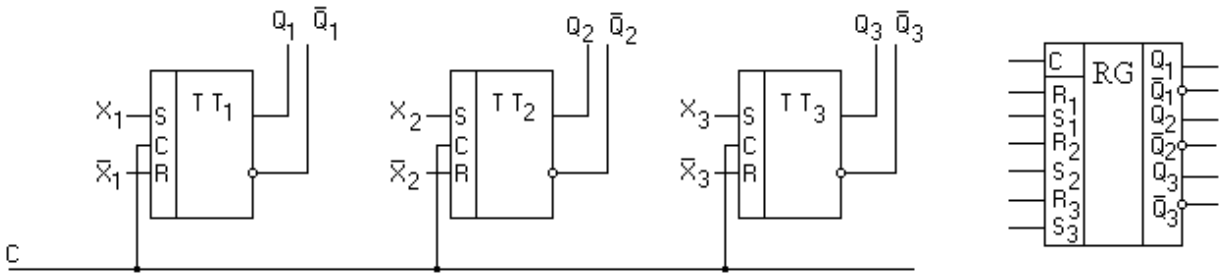


Рисунок 5.2 – Парафазный регистр хранения

Регистр обозначают идентификатором (имя регистра) RG [1 – 18] , P1[1 – 6] . Здесь [...] количество разрядов.

Совокупность всех входов образует входную шину, а выходов – выходную шину. Условное обозначение регистра показано на рис. 5.3.

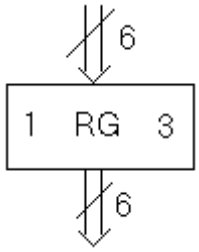


Рисунок 5.3 – Условное обозначение регистра

На операционных схемах регистры и их соединение изображают так, как показано на рис. 5.4.

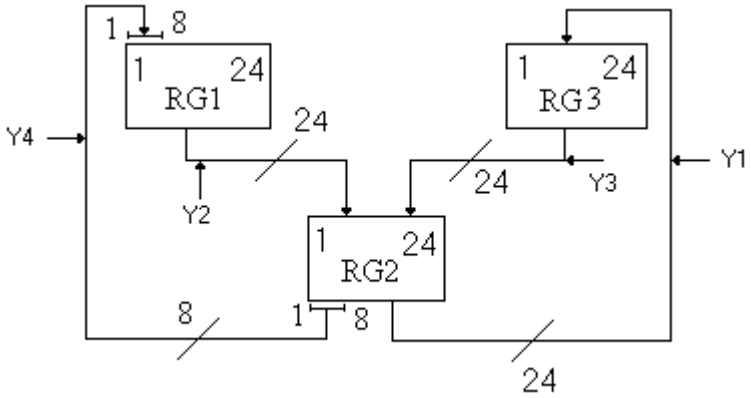


Рисунок 5.4 – Соединение регистров хранения

В этой операционной схеме с тремя 24-разрядными регистрами выполняются следующие микрооперации передачи:

Y1: RG3: = RG2

Y2: RG2: = RG1

Y3: RG2: = RG3

Y4: RG1[1-8] : = RG2 [1-8]

Если часть разрядов в регистре имеют самостоятельное значение, то они называются субрегистром или подрегистром (RG2[1-8] – подрегистр регистра RG2[1-24]). Регистры хранения могут выполнятся на триггерах любых типов.

Синтез регистров хранения

Возьмем два четырёхразрядных регистра хранения на JK-триггерах и соединим их между собой с помощью конъюнкторов (рис. 5.5).

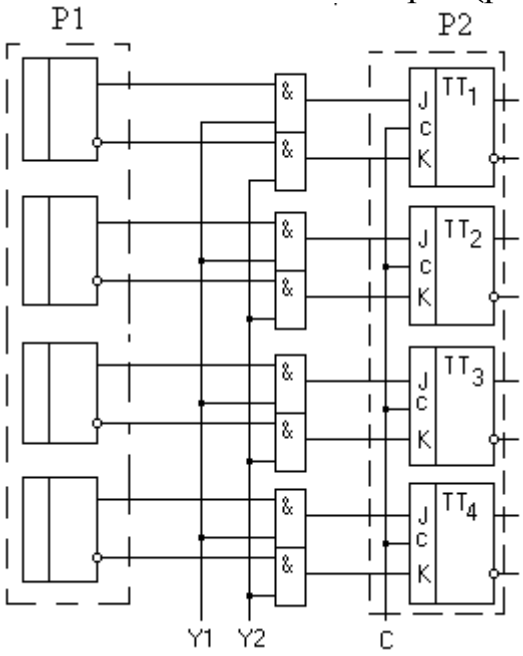


Рисунок 5.5 – Соединение регистров с помощью конъюнкторов

Входы Y1 и Y2 являются дополнительными управляющими входами. На операционных схемах такое соединение регистров условно обозначают следующим образом (рис. 5.6).

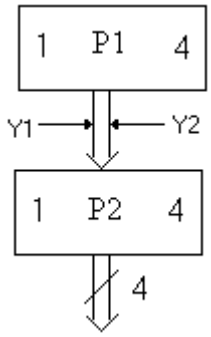


Рисунок 5.6 – Соединение регистров на операционных схемах

Рассмотрим, как будут влиять сигналы Y1 и Y2 на передачу информации с регистра P1 на регистр P2.

Пусть, например, Y1=Y2=0. Тогда P2 находится в режиме хранения своей информации, так как на выходах всех конъюнкторов имеются нули (на входах J = K = 0). Если Y1=1, Y2=0, то на входах K =0 (нижние конъюнкторы), а состояние входов J рассмотрим по таблице (рис. 5.7), в которой состояние регистра определяется прямыми выходами триггеров.

Такт t			Такт t+1
P1	P2	Входы P2	Регистр P2
0	0	0 0	0
0	1	0 0	1
1	0	1 0	1
1	1	1 0	1

Рисунок 5.7 – Состояния входов и выходов регистра P2

Если на входе триггера J = 1, то на выходе P2 тоже будет единица. Таким образом , выполняется микрооперация дизъюнкции над содержимым регистров

Y1: P2 : = P1 V P2

Возьмём другой случай Y1= 0, Y2 = 1 (рис. 5.8).

Такт t			Такт t+1
P1	P2	Входы P2	Регистр P2
0	0	0 1	0
0	1	0 1	0
1	0	0 0	0
1	1	0 0	1

Рисунок 5.8 – Состояния входов и выходов регистра P2

Видно, что выполняется другая микрооперация, а именно – конъюнкция

Y2: P2 : = P1 ∧ P2

Нетрудно видеть, что если Y1=Y2=1, то P1: = P2, то есть выполняется микрооперация передачи.

Мы выполнили анализ регистра, то есть соединили регистры и ответили на вопрос, что будет при таком соединении.

На практике обычно ставится задача не анализа, а синтеза, то есть как соединить регистры между собой, чтобы выполнялась требуемая микрооперация.

Синтез регистра сводится к синтезу одного разряда, так как все остальные разряды идентичны. Пусть, например, требуется синтезировать регистр на Т-триггерах для выполнения следующих микроопераций между входной информацией и содержимым регистра:

1) передача; 2) дизьюнкция (V); 3) конънкция (∧).

Составим схему (рис. 5.9).

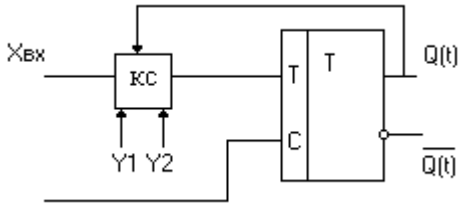


Рисунок 5.9 – Схема для синтеза одного разряда регистра

Здесь КС – комбинационная схема.

Составим таблицу истинности работы нашего устройства, но сначала необходимо произвольно закодировать управляющие сигналы (рис. 5.10).

Y1	Y2	Микрооперация
0	0	Передача
0	1	V
1	1	Λ

Рисунок 5.10 – Кодировка микроопераций

Составляем таблицу истинности для дальнейшего синтеза (рис. 5.11).

Xвх	Y1	Y2	Q(t)	Q(t+1)	T^*
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	*	-
0	1	0	1	*	-
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	*	-
1	1	0	1	*	-
1	1	1	0	0	0
1	1	1	1	1	0

Рисунок 5.11 – Таблица истинности для синтеза регистра

На основании словаря переходов Т-триггера (рис. 5.12) заполняем колонку с сигналом T* - выходным сигналом комбинационной схемы.

Q(t)	T	Q(t+1)
0	0	0
0	1	1
1	1	0
1	0	1

Рисунок 5.12 – Словарь переходов Т - триггера

Составим карту Карно (рис. 5.13) и минимизируем функцию для Т*.

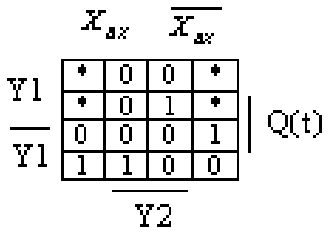


Рисунок 5.13 – Карта Карно для регистра

Получаем минимальную форму:

$$T^* = X_{BX} * \overline{Y_1} * \overline{Q(t)} + \overline{X}_{BX} * \overline{Y_2} * Q(t) + \overline{X}_{BX} * Y_1 * Q(t),$$

по которой составляем комбинационную схему одного разряда регистра (рис. 5.14).

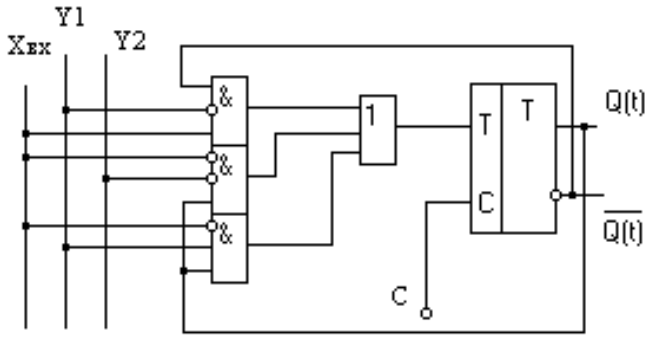


Рисунок 5.14 – Схема одного разряда регистра

По такому алгоритму регистры хранения можно создавать для выполнения различных микроопераций на триггерах любого типа.

Регистры сдвига

Служат для сдвига кода слова, занесённого в регистр. В зависимости от направления сдвига регистры делят на:

1) регистры со сдвигом вправо;
2) регистры со сдвигом влево;
3) реверсивные регистры.

Причём различают ещё и сдвиги арифметические, логические и циклические.

При арифметическом сдвиге смещаются все разряды кода, кроме знакового (крайнего левого бита). При логическом сдвиге смещаются все

разряды кода, включая знаковый бит. При циклическом сдвиге крайние разряды соединены между собой, поэтому выдвигающийся бит помещается на освободившееся место.

При арифметическом и логическом сдвигах выдвигающиеся биты теряются, а освободившиеся места заполняются нулями.

Микрооперация сдвига записывается следующим образом:

Y1: P2:= сдв ЛП (2) P2 - сдвиг логический вправо на два бита содержимого регистра P2;

Y2: P1:= сдв АЛ (1) P1 - сдвиг арифметический влево на один бит содержимого регистра P1;

Y3: RG : = сдв ЦП (1) RG – сдвиг циклический вправо на один бит содержимого регистра RG.

Регистр сдвига наиболее просто выполнить на D-триггерах (рис. 5.15).

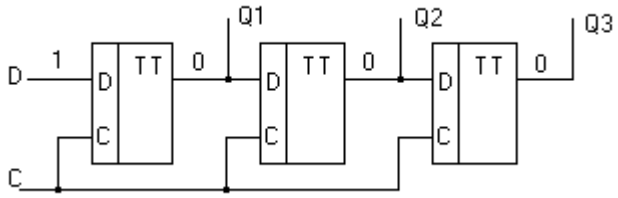


Рисунок 5.15 – Регистр сдвига

С приходом синхроимпульса код смещается вправо на один разряд. Регистр сдвига можно выполнить на JK-триггерах. Для чего организуют D-вход (рис. 5.16).

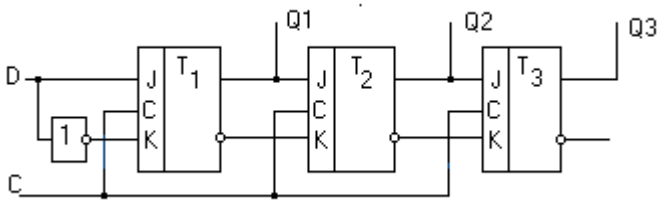


Рисунок 5.16 – Регистр сдвига на JK-триггерах

Так как цепь синхронизации является общей для всего блока (или узла), а микрооперацию сдвига требуется выполнить не всегда, то делают дополнительный вход через конъюнктор (рис. 5.17). При V=1 сдвиг будет выполнен по синхросигналу. При V=0 сдвига не будет.

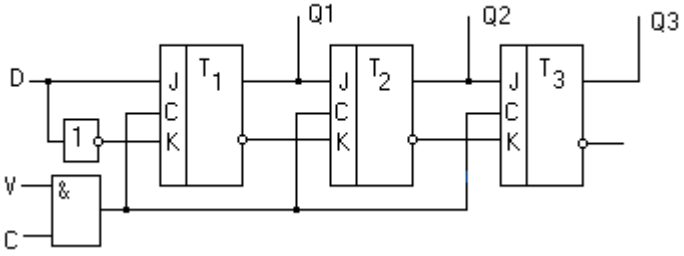


Рисунок 5.17 – Регистр сдвига на JK-триггерах с конъюнктором

Поэтому регистры сдвига удобно выполнять на DV-триггерах (рис. 5.18). На вход V подают сигнал микрооперации сдвига.

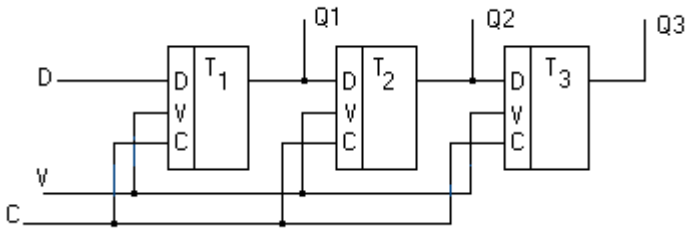


Рисунок 5.18 – Регистр сдвига на DV-триггерах

Регистр сдвига на операционной схеме обозначают так (рис. 5.19).

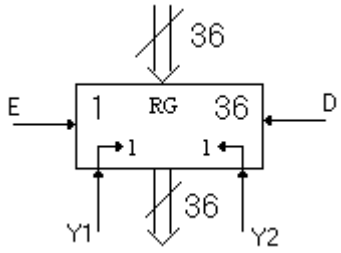


Рисунок 5.19 – Регистр сдвига на операционной схеме

На этом рисунке обозначено: Y1 - микрооперация сдвига вправо на 1 бит; Y2 - микрооперация сдвига влево на 1 бит; Д, Е - дополнительная информация, которая помещается в освобождающиеся разряды. Тогда происходит составление нового слова на регистре:

Y1: P1: = сдв ЛП (1) P2 I E

Y2: P2: = сдв ЛЛ (1) P2 I D

Микрооперация составления нового слова называется конкатенаций.

Регистры сдвига широко используются для организации последовательных АЛУ, преобразований кодов и выпускаются в виде отдельных микросхем (К155ИР1, К155ИР11 и др.).

Если вход и выход регистра сдвига соединить между собой, то код будет непрерывно циркулировать по замкнутому контуру, получается так называемый динамический регистр. Возьмём трёхразрядный регистр сдвига (рис. 5.20).

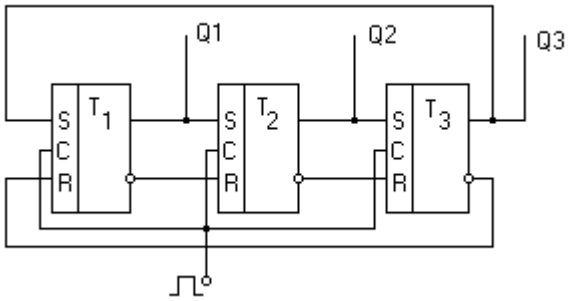


Рисунок 5.20 – Динамический регистр

Пусть в исходном состоянии на регистре записан код 1 0 0 (такт № 0). В каждом последующем такте эта единица будет перемещаться (рис.5.21).

Такт	Q1	Q2	Q3	tцсх<tцц			tцсх>tцц		
0	1	0	0	1	0	0	1	0	0
1	0	1	0						
2	0	0	1	0	0	1			
3	1	0	0						
4	0	1	0	0	1	0	0	1	0
5	0	0	1						
6	1	0	0	1	0	0			
7	0	1	0						
8	0	0	1	0	0	1	0	0	1
9	1	0	0						
10	0	1	0	0	1	0			
11	0	0	1						
12	1	0	0	1	0	0	1	0	0
13	0	1	0						
14	0	0	1	0	0	1			
15	1	0	0						
16	0	1	0	0	1	0	0	1	0
17	0	0	1						
18	1	0	0	1	0	0			
19	0	1	0						

Рисунок 5.21 – Движение кода в динамическом регистре

Видно, что код на регистре повторяется через n-тактов, где n = 3 число разрядов регистра.

Существуют два понятия:

1) период циркуляции кода на регистре, он равен числу разрядов (tцц = n);

2) период цикла схемы это время, через которое код считывается из регистра (t цсх).

В зависимости от соотношения tцц и tцсх возможны три режима работы:

1) tцц = tцсх - режим хранения информации;

2) tцсх < tцц - режим сдвига влево (регистр со сдвигом влево);

3) tцсх > tцц - регистр со сдвигом вправо.

Получился универсальный регистр, но быстродействие его невысокое.

Динамические n – разрядные регистры соединяются в блоки по m штук (как показано на рис. 5.22) и записывают в них слова «поперёк». Всего можно записать n штук m – разрядных слов. Эти слова появляются в некотором сечении регистров через время равное n тактов. Получили так называемую регистровую память, или цифровую линию задержки (ЦЛЗ).

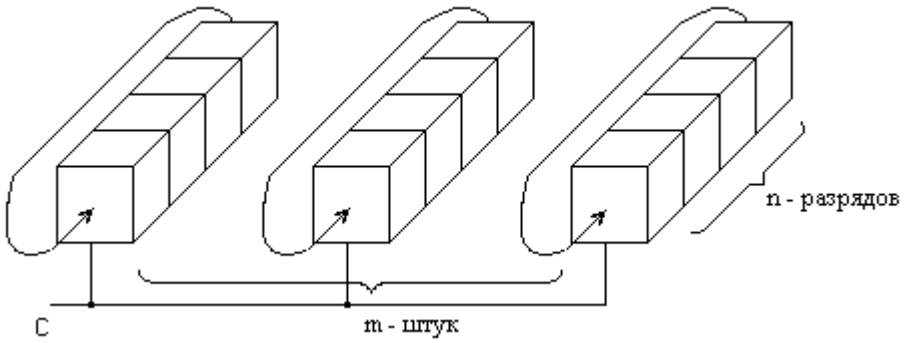


Рисунок 5.22 – Цифровая линия задержки

Для этих целей специально выпускаются регистры. Например, микросхема К144ИР3 – регистр на 64 бит (задержка на 64 такта).

5.3.2 Счетчики

Счетчик – это узел ЭВМ, предназначенный для подсчета числа входных сигналов. Счетчик выполняет микрооперацию счет Сч =Сч ± 1.

Счетчики характеризуются рядом параметров:

1) модуль счета М. Это число устойчивых состояний счетчика. Если М кратно 2^n, где n - число разрядов (триггеров), то счётчик называется двоичный. Иначе счётчик с произвольным модулем (основанием) счёта;

2) емкость Е. Это максимальное число, которое может быть записано в счетчик Е = М – 1;

3) быстродействие или скорость перехода из состояния 11...111 в состояние 00...000 или наоборот.

Счетчики классифицируют по таким признакам:

1) по направлению счета (суммирующие, вычитающие и реверсивные);

2) по способу построения цепи переноса (с последовательным переносом, с параллельным переносом и комбинированные);

3) по способу опрокидывания триггеров (асинхронные и синхронные).

Простейший суммирующий асинхронный двоичный счетчик строится на Т-триггерах. Например, трёхразрядный счетчик (рис.5.23).

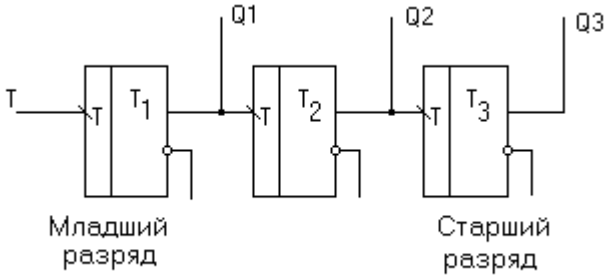


Рисунок 5.23 – Простейший суммирующий счётчик

Диаграмма его работы приведена на рис. 5.24. Из неё видно, что если двоичное число читать снизу вверх, то это число увеличивается на единицу с приходом очередного сигнала на вход T. Модуль счёта M = 8, а емкость E = 7. Если информацию снимать с инверсных выходов триггеров, то мы получаем вычитающий счетчик.

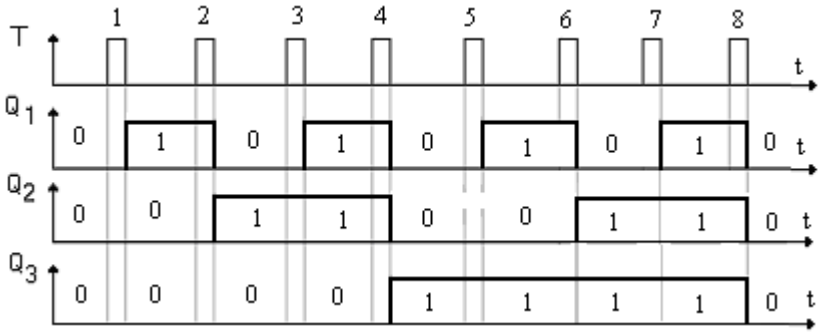


Рисунок 5.24 – Диаграмма работы счётчика

Составим схему такого же счетчика, но на триггерах с прямым типом управления (рис. 5.25).

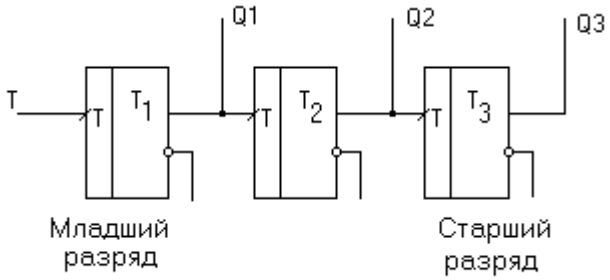


Рисунок 5.25 – Простейший вычитающий счётчик

Здесь триггеры опрокидываются по переднему фронту входного сигнала, поэтому диаграмма работы будет следующая (рис. 5.26).

Получили вычитающий счетчик. Если выходные сигналы снимать с инверсных выходов триггеров, то будет суммирующий счетчик.

Это счетчик с так называемым последовательным переносом, так как перенос распространяется последовательно от разряда к разряду.

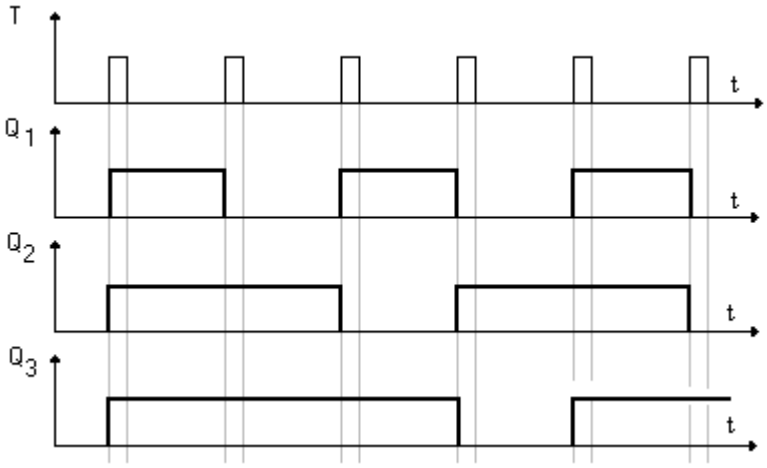


Рисунок 5.26 – Диаграмма работы

Быстродействие такого счётчика определяется временем опрокидывания всех разрядов (рис. 5.27) и примерно равно произведению времени срабатывания одного триггера (tср) и числа разрядов (n).

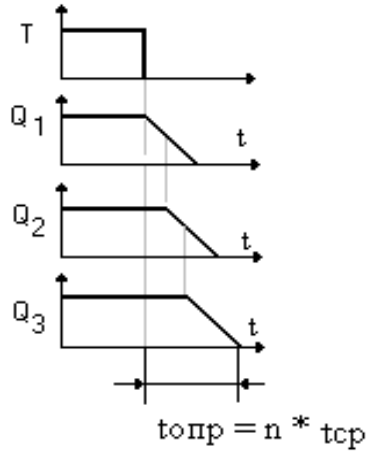


Рисунок 5.27 – Опрокидывание всех разрядов счётчика

Асинхронные счетчики с последовательным переносом имеют низкое быстродействие, поэтому чаще используют синхронные счетчики.

Трёхразрядный синхронный суммирующий счетчик (рис. 5. 28).

Счётчик строится на синхронных Т - триггерах. Счётные импульсы поступают на вход С. Все триггеры опрокидываются одновременно, по сигналу С.

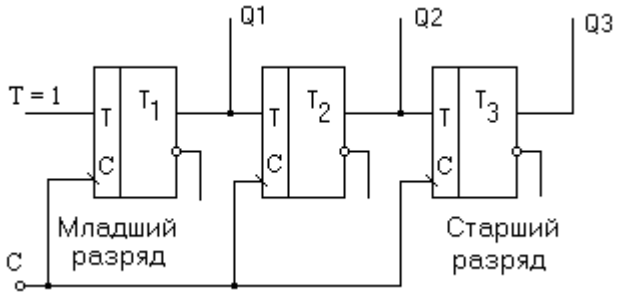


Рисунок 5.28 – Синхронный счётчик

Здесь t опр ≈ t ср – время опрокидывания счётчика не зависит от количества разрядов, но по сути этот счетчик остается счетчиком с последовательным переносом.

Несколько большими возможностями обладают счетчики с параллельным переносом. Трёхразрядный счетчик с параллельным переносом (рис. 5.29).

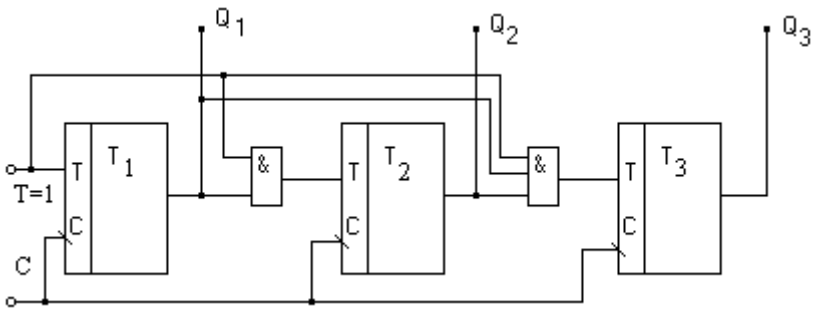


Рисунок 5.29 – Синхронный счётчик с параллельным переносом

Здесь, когда сигнал T = 1, выполняется счет, когда T=0, на выходах всех конъюнкторов будут нули и все триггеры счетчика будут находиться в режиме хранения информации, то есть сигнал T это сигнал микрооперации счета:

T: Сч: = Сч + 1

Быстродействие t опр = t ср.

Счетчик хороший, но с увеличением числа разрядов возрастает сложность конъюнкторов, поэтому многоразрядный счетчик разбивают на группы по 4 или 8 разрядов. Внутри группы делают параллельный или последовательный перенос, а между группами параллельный. Такой счетчик называется счетчиком с групповым переносом. На операционных схемах счетчик обозначают следующим образом (рис.5.30).

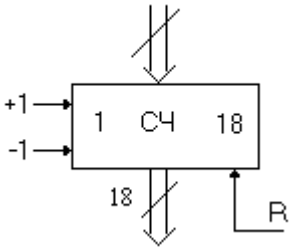


Рисунок 5.30 – Обозначение счётчика на операционных схемах

Здесь + 1 - вход для работы на сложение;

- 1 - вход для работы на вычитание;

R - вход установки всех разрядов счетчика в нуль.

Счетчики выполняются в виде отдельных микросхем. Например, четырёхразрядный двоичный реверсивный счетчик К155ИЕ7 (рис. 5.31).

Рисунок 5.31 – Пример микросхемы счётчика

На этой схеме C1,C2 – синхровходы (C1 для работы на сложение, C2 для работы на вычитание), A1,A2,A3,A4 - входы для начальной записи информации в счетчик, R - сброс всех разрядов счетчика в ноль, L - управление режимом. При L=0 запись информации по входам A. При L=1 выполняется счет. P1 - выход переноса при работе на сложение, P2 - выход единицы заёма при работе на вычитание. Счётчик выполнен на RST-триггерах. Выходы P1 и P2 служат для наращивания разрядности счётчика.

Различные области использования счетчиков требуют различных модулей счета, не кратных 2^n, то есть это счетчики с произвольным модулем счета К.

Построить такие счётчики можно тремя способами.

1. Использование стандартного двоичного счетчика с модулем M = 2^n большим, чем требуемый модуль К.

Допустим, нам нужен счетчик с модулем 17. Тогда требуется пятиразрядный двоичный счетчик (n =5).

$K = 17_{10} = 21_8 = 10001$. На выход подключаем конъюнктор (рис. 5.32).

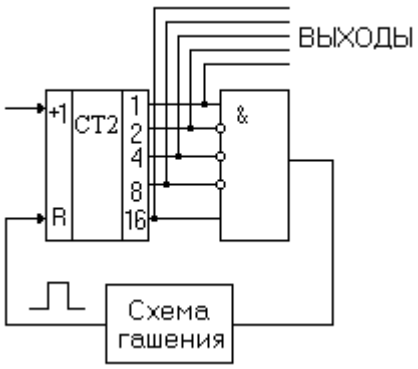


Рисунок 5.32 – Счётчик с произвольным модулем счёта

Когда код на счётчике будет равен К = 10001, на выходе конъюнктора появляется единица, которая запускает схему гашения. Длительность импульса схемы гашения должна быть достаточной для надёжного сброса всех триггеров счётчика, иначе код на счётчике будет неправильным. Для другого модуля К, конъюнктор будет другим (рис. 5.33).

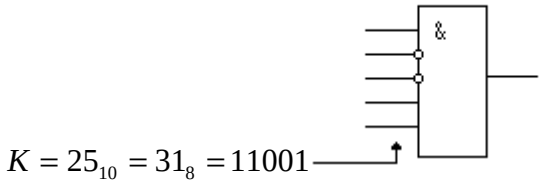


Рисунок 5.33 – Конъюнктор для счётчика

Схема гашения выполняется на RS-триггере, как это показано на рис. 5.34.

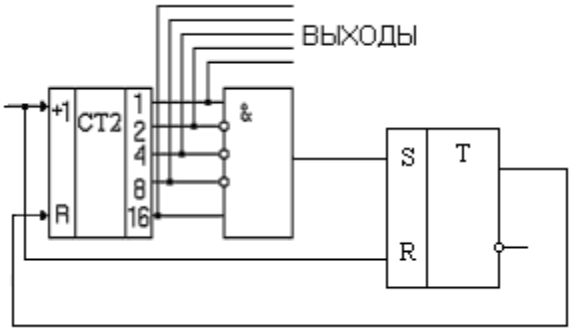


Рисунок 5.34 – Схема гашения на RS-триггере

Сигнал на входе R счётчика будет действовать в течение одного периода счётных импульсов.

Достоинства: используется стандартный счётчик; естественный ход нарастания кода на счетчике.

Недостаток: требуется схема гашения. Невозможность изменения модуля счета без переделки схемы.

2. Использование стандартного счетчика с $M = 2^n$ > К, в который загружают начальное число (исходный код) и работают либо на сложение до естественного переполнения счётчика, либо на вычитание до обнуления счётчика (рис. 5.35), после этого в счетчик снова загружается исходный код.

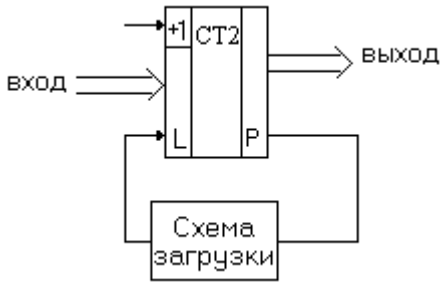


Рисунок 5.35 – Второй способ построения счётчика

Достоинства этого способа: использование штатной цепи переноса; легко программировать модуль и изменять его в процессе работы.

Недостатки: начало счета соответствует некоторому числу, поэтому нет естественного изменения кода; необходима схема загрузки.

3. Прямой синтез счетчика на триггерах заданных типов.

Этот способ используется в специальной аппаратуре, а также для построения устройств управления операционными элементами, где счётчики вырабатывают сигналы микроопераций.

Пусть, например, требуется синтезировать счетчик по произвольному основанию (с модулем M=6).

Порядок смены состояний такой: 001, 010, 011, 100, 110 , 111.

Младший разряд выполнить на RS-триггере, а два других на D и T-триггерах.

Составим таблицу истинности работы счетчика (рис. 5.36).

№	Такт t			Такт t+1			Функции возбуждения		
	Q1	Q2	Q3	Q1	Q2	Q3	T	D	R S
1	0	0	1	0	1	0	0	1	1 0
2	0	1	0	0	1	1	0	1	0 1
3	0	1	1	1	0	0	1	0	1 0
4	1	0	0	1	1	0	0	1	- 0
5	1	1	0	1	1	1	0	1	0 1
6	1	1	1	0	0	1	1	0	0 -

Рисунок 5.36 – Таблица истинности счётчика и функции возбуждения

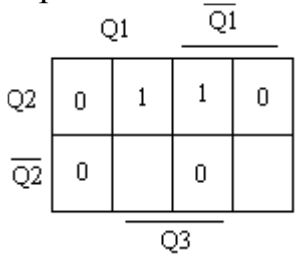
Функции возбуждения триггеров заполним на основании словарей переходов наших триггеров (рис.5.37).

Q(t)	D	T	RS	Q(t+1)
0	0	0	- 0	0
0	1	1	0 1	1
1	0	1	1 0	0
1	1	0	0 -	1

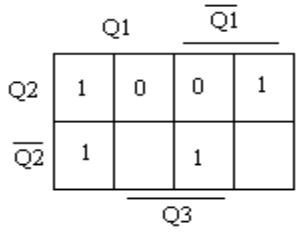
Рисунок 5.37 – Словари переходов триггеров

Составим карты Карно для каждого сигнала возбуждения (рис. 5.38).

T–триггер:



D–триггер:



RS-триггер:

Вход R Вход S

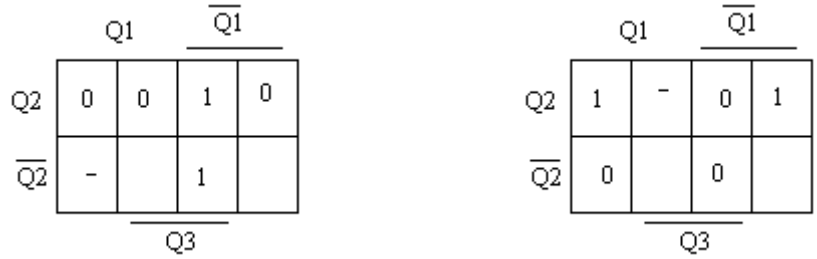


Рисунок 5.38 – Карты Карно для управляющих входов триггеров

На основании этих карт записываем минимальные формы для сигналов управления триггерами $\overline{D} = Q_2 Q_3$ $T = Q_2 Q_3$ $R = \overline{Q_1} Q_3$ $S = Q_2 \overline{Q_3}$, по которым составляем схему счетчика.

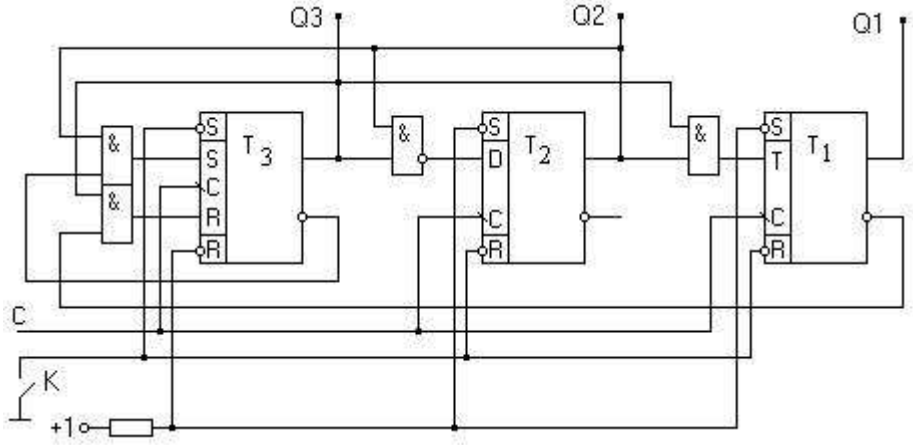


Рисунок 5.39 – Принципиальная схема счётчика на триггерах

Ключ К служит для установки счётчика в начальное состоянии Q1Q2Q3 = 0 0 1 путём его кратковременного замыкания.

5.3.3 Арифметико-логические устройства

Арифметико-логические устройства (АЛУ) служат для выполнения арифметических и логических операций над словами, называемых операндами.

АЛУ современных вычислительных машин выполняют такие группы операций:

- двоичная арифметика с фиксированной запятой;
- двоичная арифметика с плавающей запятой;
- десятичная арифметика;
- логические операции;
- специальная арифметика (нормализация чисел, сдвиги и др.);
- операции над алфавитно-цифровыми полями.

Мини, микро ЭВМ и микропроцессоры все операции выполняют в двоичной системе счисления над числами с фиксированной запятой. Другие группы операций выполняются, как правило, программным способом, то есть с использованием специальных подпрограмм.

По характеру использования элементов АЛУ делят на блочные и многофункциональные.

В **блочных** АЛУ каждая группа операций выполняется отдельным блоком, которые могут работать одновременно. Повышается быстродействие, но возрастают и затраты оборудования.

В **многофункциональных** АЛУ все операции выполняются одними и теми же схемами, которые коммутируются в зависимости от требуемой операции. Поэтому возрастает степень использования оборудования, но быстродействие ниже, чем в блочных АЛУ.

По своим функциям АЛУ является операционным блоком и широко используется для построения арифметических узлов, в частности АЛУ входит в состав любого микропроцессора.

Рассмотрим схему АЛУ для сложения чисел с фиксированной запятой, передаваемых по общей шине. Она приведена на рис. 5.40.

Основой АЛУ является n-разрядный комбинационный сумматор. Операнды поступают по общей входной шине в регистры РА и РВ, которые поочерёдно подключают к шине с помощью тристабильных элементов. Первое слагаемое (уменьшаемое) помещается в регистре РВ. Второе слагаемое (вычитаемое) помещается в регистре Р1, который связан с регистром РА цепями прямой и инверсной передачи кода. Прямая передача используется при сложении, а инверсная при вычитании (выполняется сложение в дополнительном коде). Операция вычитания заменяется сложением: Z = X - Y = X + (-Y), а в младший разряд результата добавляется единица. Результат выдаётся в регистр Р2.

Комбинационная схема (КС) следит за переносами в знаковый разряд и из него и вырабатывает признак результата ПР (флаг): больше нуля (ПР =01), меньше нуля(ПР =10), равно нулю (ПР = 00), переполнение разрядной сетки (ПР =11).

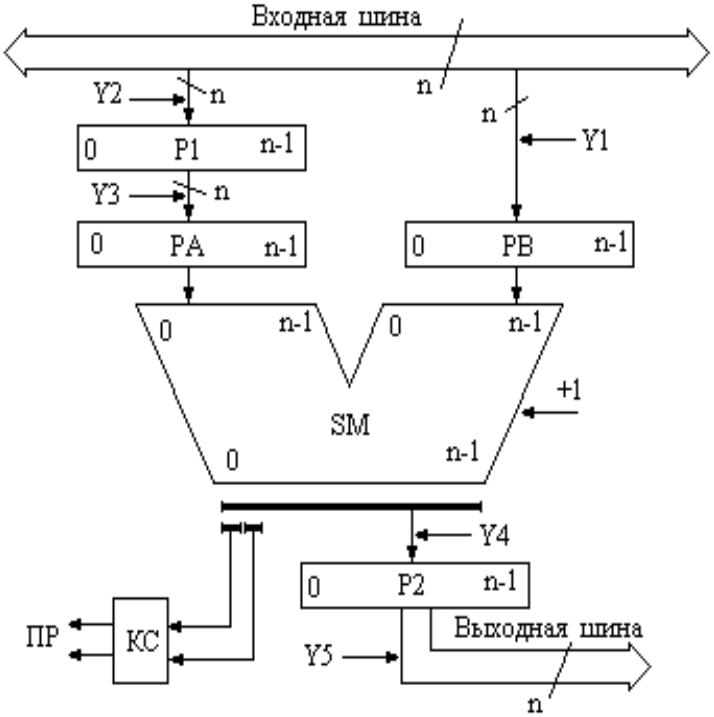


Рисунок 5.40 – Схема АЛУ для сложения чисел

Составим микропрограмму сложения двух чисел.

Y1: РВ = ⟨слово1⟩.

Y2: Р1 = ⟨слово2⟩.

Y3: РА = Если сложение то Р1 иначе $\overline{P1}$.

Y4: Р2 = Если сложение то РА + РВ иначе РА+ РВ + 1.

Y5: Если ПР = 11 то переполнение иначе ⟨вых. шина⟩= Р2.

Конец

Для выполнения операций умножения, деления, логических операций схема АЛУ дополняется другими элементами.

Контрольные вопросы

1. Каковы различия в уровнях представления вычислительных устройств?
2. Что такое регистры хранения и сдвига?
3. Какие существуют счетчики?
4. Что такое арифметико-логическое устройство?

6 ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

6.1 Основные понятия

Память цифровых систем (запоминающие устройства ЗУ) - это совокупность технических средств, предназначенных для приема, хранения и воспроизведения информации, представленной двоичными кодами. Основными характеристиками систем памяти являются:

- информационная емкость (бит или байт, «К» кило 2^{10} =1024, «М» мега 2^{20} =1048576, «Г» гига 2^{30} число близкое к 10^9);
- быстродействие – время от момента обращения к ЗУ до появления требуемой информации на выходе (время доступа) или количество воспроизводимой информации в единицу времени (бит/с);
- энергопотребление – мощность на единицу информационной емкости (Вт/бит).

По назначению все ЗУ делятся на:

1. Внешние ЗУ малого быстродействия. Это сменные магнитные диски (сотни Гбайт, время доступа порядка 5 мс), оптические диски (десятки Гбайт, время доступа порядка 0,5 с). Для сравнения: 1 Мбайт – книга, состоящая примерно из 400 страниц, это около 10^6 букв – одна буква записывается (кодируется) одним байтом.

2. Буферные ЗУ промежуточные, служат для временного хранения информации, подлежащей немедленной обработке или вводу/выводу. Это так называемая КЭШ – память.
3. Постоянные ЗУ (ПЗУ) – память для длительного хранения неизменяемой в процессе работы информации (специальных программ, микропрограмм, констант). ПЗУ – память только для чтения (ROM –read only memory).

4. Оперативные ЗУ (ОЗУ) – память для хранения оперативной информации – операнды, исходные данные, результаты и др. ОЗУ должны допускать запись и считывание информации, это память с произвольным доступом (RAM –random access memory).

В свою очередь ОЗУ делят на:

а) адресные: адрес – номер ячейки, в которой хранится информация (поиск информации производится по адресу);

б) ассоциативные: поиск выполняется не по адресу, а по содержанию этой информации (по тегам) сразу во всех ячейках, что позволяет значительно ускорить процесс поиска и обработки информации;

в) стековые: это тоже безадресная память. Здесь ячейки памяти соединяются последовательно, новое слово как бы «заталкивает» предыдущее слово вглубь (трамвай с одной дверью!) Память типа LIFO (last in first out: последний вошел – первый вышел).

5. Регистровые ЗУ . Это набор регистров, содержимое которых непосредственно используется при обработке информации.

Как правило, характеристики системы памяти находятся в противоречии – чем больше объём, тем ниже быстродействие (рис. 6.1).

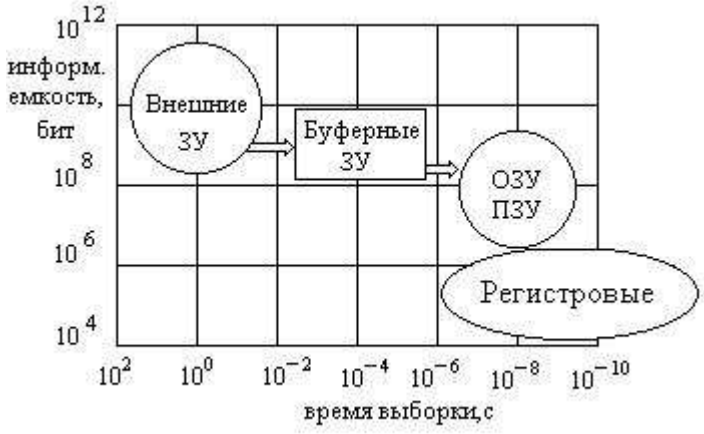


Рисунок 6.1 – Связь объёма и быстродействия памяти

Память вычислителя может содержать не все названные виды памяти. Конфигурация системы памяти бывает различной, в зависимости от назначения ЭВУ. Для повышения производительности пользуются иерархической организацией памяти. Это такая память, которая состоит из ряда взаимосвязанных ЗУ различной емкости и быстродействия. Каждое ЗУ обменивается информацией только с двумя соседними ЗУ.

В современных микро и мини ЭВМ используется исключительно полупроводниковая память на всех уровнях иерархической лестницы. Стоимость памяти достигает половины стоимости всей микро ЭВМ, а её физическая реализация занимает до 70% общего объема!

6.2 Построение памяти требуемого объёма

Несмотря на большое разнообразие типов микросхем памяти (ПЗУ и ОЗУ), все они имеют сходную структуру: накопитель, дешифратор и схему управления чтением/записью.

При построении систем памяти наибольшее распространение получили БИС ЗУ с конфигурацией М х1бит (М = 256,512,1024,2048,4096 …). Хотя БИС с другой конфигурацией и выпускаются, но они применяются реже.

Типовая структура статического МОП ЗУ с организацией 1К х1бит приведена на рис. 6.2.

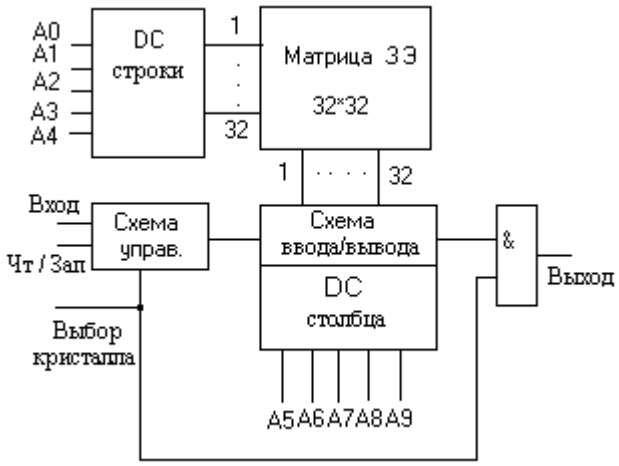


Рисунок 6.2 – Типовая структура статического ЗУ с организацией 1К х 1

Основой ЗУ является матрица запоминающих элементов (ЗЭ). В качестве ЗЭ используются триггеры (статические ОЗУ), емкости затворов МОП структур (динамические ОЗУ) или элементы с односторонней проводимостью (ПЗУ). Здесь размер матрицы 32 х 32 элемента. Адрес десятиразрядный (2^{10} =1к). Первые пять разрядов адреса указывают номер строки, вторые пять – номер столбца. Схема управления переключает режим чтения / записи (для ОЗУ) по внешнему управляющему сигналу. Обращение к данной схеме возможно только при наличии сигнала выбор кристалла. Этот вход нужен для построения блоков памяти большого объема. Особенностью этой системы является независимость входных и выходных данных. Если шина данных двунаправленная, то требуются дополнительные цепи, объединяющие вход и выход. Структурная схема ОЗУ с общими выводами для входных и выходных сигналов приведена на рис. 6.3.

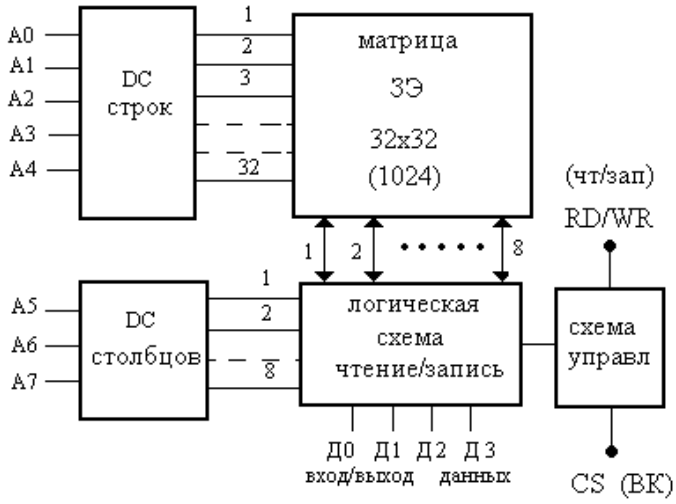


Рисунок 6.3 – Типовая структура статического ЗУ
с организацией 256 х 4 и общей шиной данных

Емкость ЗУ также равна 1024 бит, но организация другая – 256 х 4. Для выбора требуемого четырехразрядного слова используется пятиразрядный DC строк и трехразрядной DC столбцов, выбирающий одно из восьми слов в строке.

Память имеет разрядность, равную или кратную разрядности данных. Для восьмиразрядного МП нужна память с длиной слова восемь бит.

При объединении нескольких микросхем памяти в общий блок происходит наращивание памяти по «горизонтали» или по «вертикали».

“Горизонтальное” наращивание позволяет получить требуемую разрядность памяти при фиксированном количестве слов. Наращивание по “вертикали” обеспечивает увеличение объема памяти, то есть числа хранимых слов при их фиксированной разрядности.

Организация ОЗУ емкостью 256 х 8 бит на основе двух БИС емкостью 256 х 4 бит показана на рис. 6.4. Это наращивание по «горизонтали».

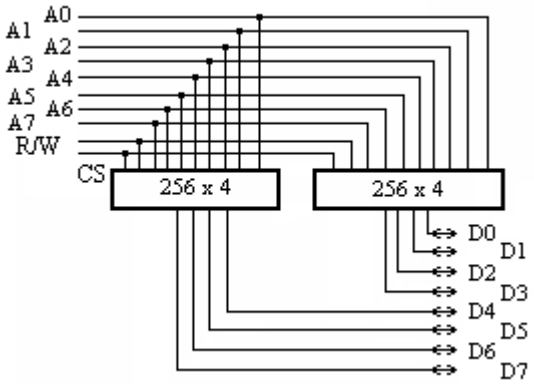


Рисунок 6.4 – Наращивание разрядности памяти

Организация ОЗУ емкостью 4096 х 4 бит (4К х 4) на основе четырех БИС емкостью 1024 х 4 бит (1К х 4) показана на рис. 6.5.

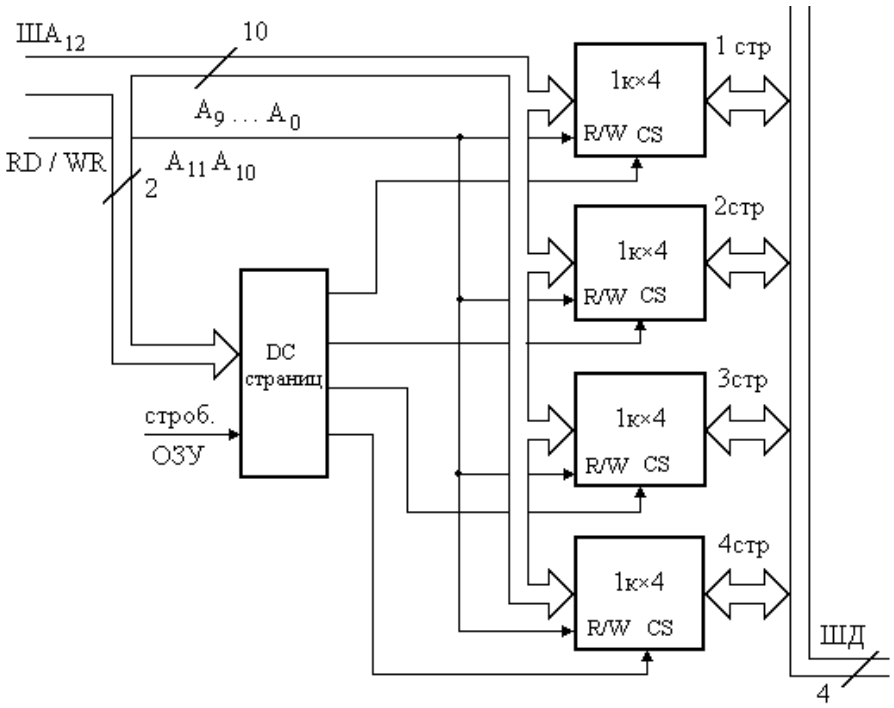


Рисунок 6.5 – Наращивание числа ячеек памяти

Это наращивание по «вертикали». DC страницы выбирает требуемую микросхему в соответствии со старшими битами адреса путём подачи единицы на вход CS (ВК).

Такое наращивание можно осуществлять как в ОЗУ, так и в ПЗУ.
Одна или несколько страниц может быть ОЗУ, другие ПЗУ, и в зависимости от адреса происходит обращение к нужной микросхеме. Выбор страницы поясняет рис. 6.6.

	выбор страницы		адрес в пределах страницы										
	11	10	9	8	7	6	5	4	3	2	1	0	16-ричная
1-ая стр	0	0											000...3FF
2-ая	0	1											400...7FF
3-я	1	0											800...BFF
4-ая	1	1											C00...FFF

Рисунок 6.6 – Двоичные и шестнадцатеричные адреса различных страниц памяти

Следует понимать, что ОЗУ и ПЗУ могут быть изготовлены по разным технологиям.

Контрольные вопросы

1. Каковы основные характеристики запоминающих устройств?
2. Какая связь между быстродействием и объемом запоминающих устройств?
3. Типовая структура статического ЗУ.
4. Как нарастить разрядность и количество ячеек памяти?

7 ОРГАНИЗАЦИЯ ЭЛЕКТРОННО-ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ

7.1 Структура и принцип действия ЭВМ

Современные ЭВМ это машины с гибким программным управлением (используют принцип хранимой в памяти программы). Основу микроЭВМ составляет общая магистраль (шина), к которой подсоединяются все другие устройства. Поэтому современные ЭВМ называют магистрально-модульными (рис. 7.1).

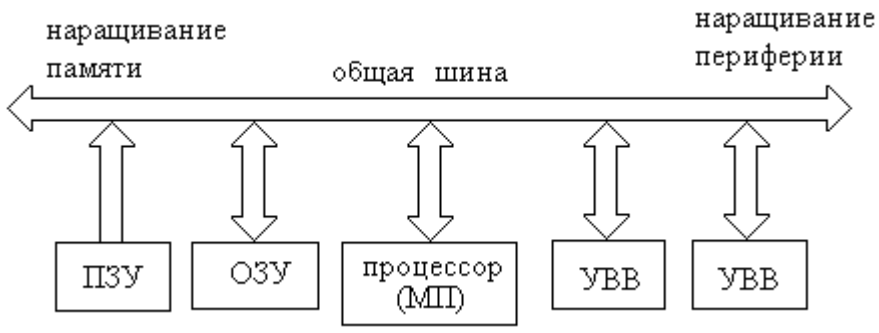


Рисунок 7.1 – Структура микроЭВМ

На рисунке обозначено: ПЗУ – постоянное запоминающее устройство (память программ), ОЗУ – оперативное запоминающее устройство (память данных), УВВ – устройства ввода / вывода информации.

Такая структура эффективна при сравнительно небольшом наборе периферийных устройств (УВВ). Универсальность же их применения может быть обеспечена лишь при высоком быстродействии процессора, что позволяет микроЭВМ обслуживать технологические процессы в реальном масштабе времени. Высокая надежность и низкая стоимость, малые размеры и потребляемая мощность, малое число источников и потребителей информации – это отличительные признаки любой микроЭВМ. Принцип действия ЭВМ (микро ЭВМ) заключается в переработке информации согласно заданному алгоритму. Алгоритм, записанный в форме, воспринимаемой ЭВМ, есть программа. Эта программа с помощью специальной программы транслятора переводится на язык машинных команд.

Команда есть минимальная самостоятельная единица действия ЭВМ. Она должна содержать все указания для выполнения операции – какие действия, над какими операндами и куда поместить результат.

В команде обычно содержатся не сами операнды, а их адреса. Команда состоит из двух частей: операционной и адресной. Операционная часть содержит код операции (КОП). Адресная часть содержит информацию об операндах или их адресах и адрес, куда поместить результат операции. В зависимости от структуры адресной части команды бывают: трёхадресные, двухадресные, одноадресные и безадресные (рис. 7.2).

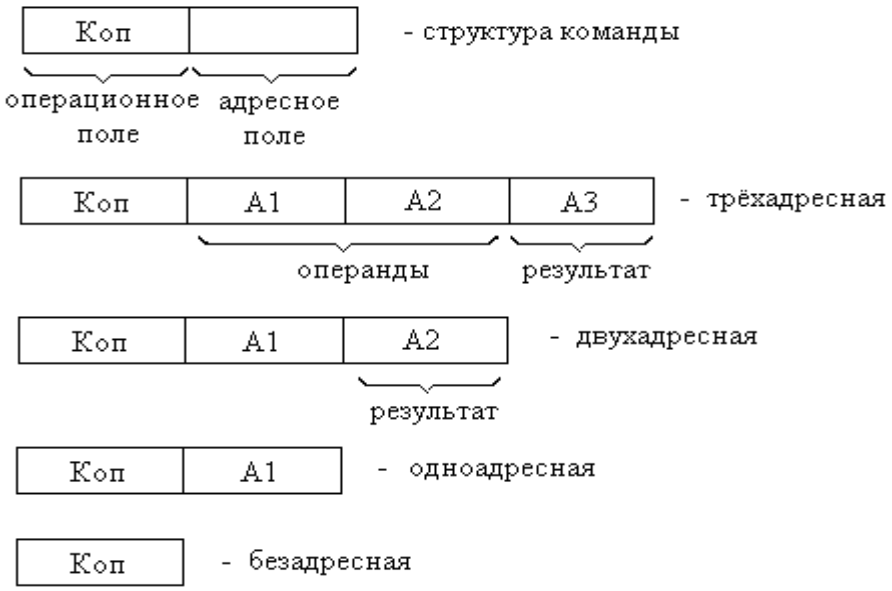


Рисунок 7.2 – Структура команды и её разновидности

В трёхадресных командах указаны адреса первого (А1) и второго (А2) операндов и адрес, куда следует поместить результат (А3). После выполнения данной команды выполняется следующая по порядку команда. Такой порядок выборки называют естественным. Операция, выполняемая трехадресной командой, символически записывается так:

$$ОП[A3] := ОП[A1] \nabla ОП[A2] ,$$

где ∇ – символ операции (+,-, x, / и др.).

Можно условиться, что результат помещают на место одного из операндов (например, А2), тогда получаем двухадресную команду:

$$ОП[A2] := ОП[A1] \nabla ОП[A2]$$

То есть используется подразумеваемый адрес.

В одноадресной команде подразумеваемые адреса имеют один из операндов и результат операции. В качестве второго операнда используются содержимое внутреннего регистра, называемого в этом случае аккумулятором. Туда же помещается и результат:

$$Акк := Акк \nabla ОП[A]$$

Это аккумуляторные машины. В некоторых случаях возможно использование безадресных команд, когда подразумеваются адреса операндов и результата операции – двухаккумуляторные машины.

Наиболее логичны и естественны трёхадресные команды, но они недопустимо длинные. Кроме того, в качестве операндов используются результаты предыдущих операций, хранимых в регистрах машины. Поэтому выполняемая операция приобретает характер одно- или двухадресной. В современных ЭВМ применяют одно-, двухадресные команды и их модификации.

7.2 Типовая структура обрабатывающей части микропроцессора

Основой любого МП являются арифметико-логическое устройство и набор регистров общего назначения (РОН), часто называемый сверхоперативным запоминающим устройством. В РОН хранятся слова, подлежащие обработке, и результаты обработки. Эта структура приведена на рис. 7.3.

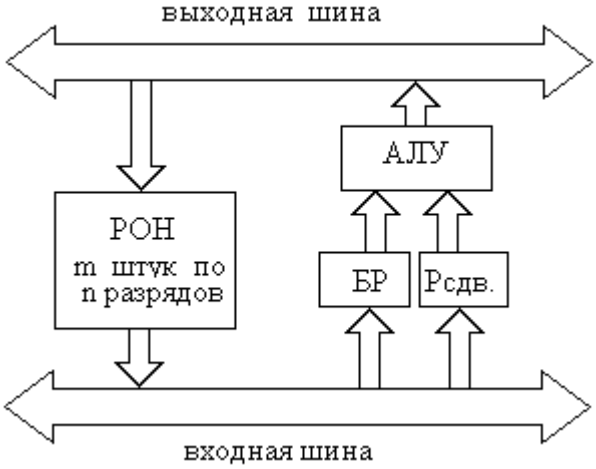


Рисунок 7.3 – Обрабатывающая часть МП без управляющих цепей

В микропроцессоре содержимое любого РОН может быть передано на буферный регистр (БР) и регистр сдвига (Рсдв.). АЛУ выполняет арифметические и логические операции над содержимым этих регистров, а результат этих операций записывается в любой из РОН.

В этой системе возможны:

- передача данных из одного РОН в другой путем пересылки слова транзитом через БР и АЛУ;
- увеличение или уменьшение на единицу содержимого любого РОН и АЛУ и засылки результата в тот же или другой регистр;
- сдвиг содержимого любого РОН путем передачи через Рсдв и АЛУ в тот же РОН.

Очевидно, что для выполнения этих и других операций на АЛУ, РОН, БР и Рсдв должны подаваться определенные управляющие сигналы. Причем важное значение имеет их распределение во времени.

Например, для передачи слова из одного РОН в другой требуются два такта (две микрооперации):

такт 1 – выборка содержимого РОН и его прием в БР;

такт 2 – запись информации, переданной на вход РОН через АЛУ.

Таковые сигналы поступают от тактового генератора, причем максимально возможная частота, а значит и время выполнения одной операции будут определяться задержкой сигналов в различных элементах схемы.

В ряде случаев сигнал арифметического переноса из АЛУ и выходной бит регистра сдвига должны быть сохранены для последующих операций. Это выполняется с помощью двух D - триггеров. Тогда обрабатывающая часть МП принимает вид, приведенный на рис. 7.4.

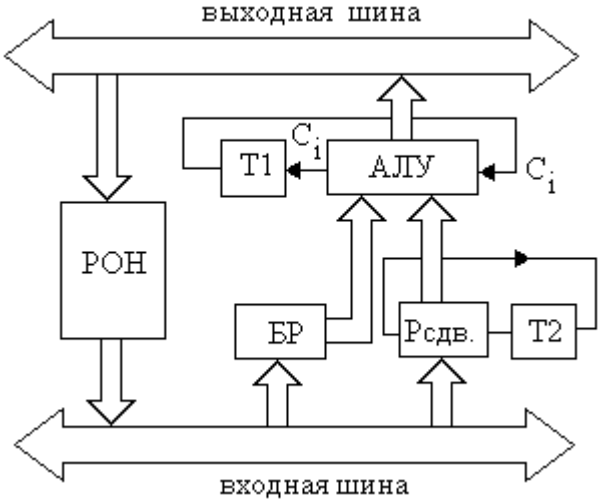


Рисунок 7.4 – Обрабатывающая часть МП с триггерами Т1 (хранения переноса) и Т2 (хранения сдвига)

В этом случае становятся возможными операции над словами с разрядностью большей, чем разрядность шин, АЛУ, РОН и других регистров.

Например, при 4-разрядной организации этих узлов можно выполнить обработку 12-разрядных слов, для хранения каждого из которых в блоке РОН отводятся три регистра. Но для их обработки уже требуются три цикла обработки четырехразрядного слова. Схематично это показано на рис. 7.5.

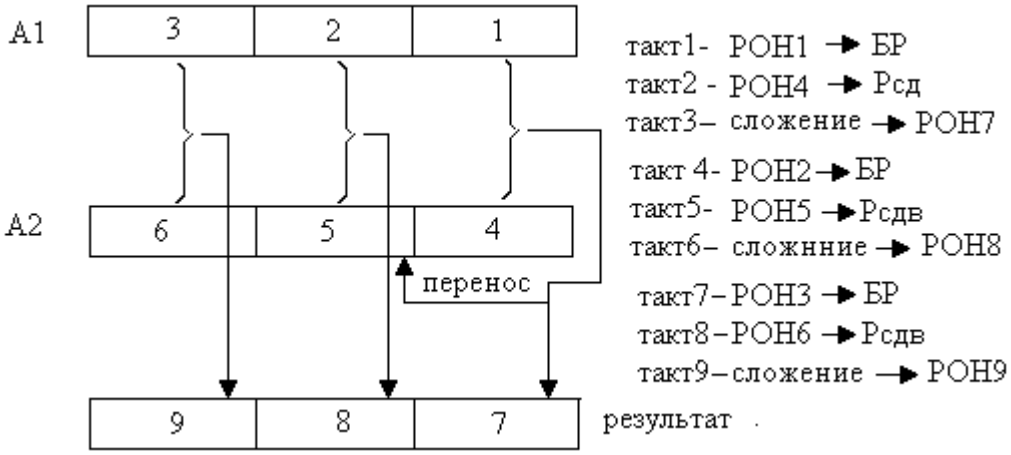


Рисунок 7.5 – Сложение 12-разрядных операндов при 4-разрядной организации МП

Рассмотренная структура с двумя шинами данных (входной и выходной) не является единственно возможной. В различных МП используются 1,2 или 3 внутренних шины. Их число существенно влияет на структуру и характеристики МП. На рис. 7.6 показана трёхшинная организация МП.

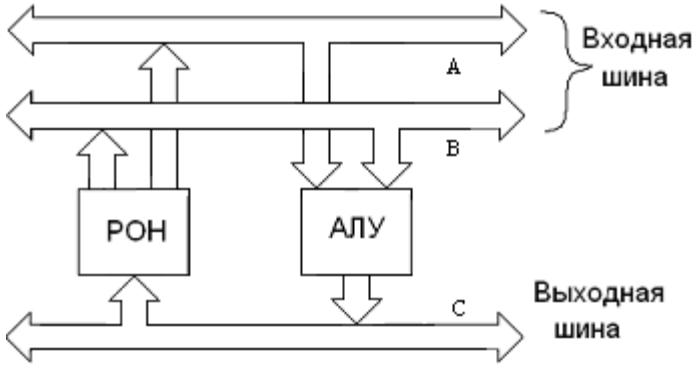


Рисунок 7.6 – Трёхшинная организация МП

В такой системе возможно выполнение арифметических и логических операций за один такт. Кроме высокого быстродействия здесь отсутствуют буферные регистры. Недостаток: большая площадь, занимаемая на кристалле.

При одношинной организации МП (рис. 7.7) обязательно наличие двух или трёх БР.

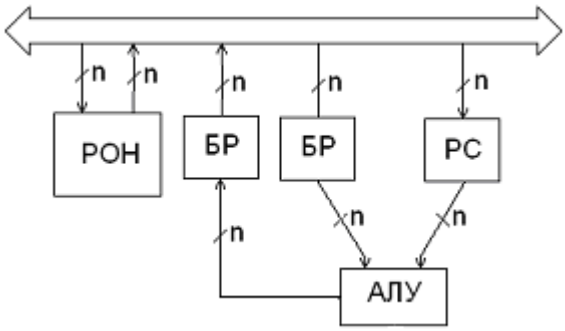


Рисунок 7.7 – Одношинная организация МП

Эта структура занимает на кристалле наименьшую площадь. Наличие БР увеличивает функциональные возможности этой системы, но у неё низкое быстродействие. Арифметические и логические операции выполняются не менее чем за 2 или 3 такта.

7.3 Устройство управления в МП

Устройство управления МП должно выполнять две основные функции: выборку команд программы в нужной последовательности и обработку полей команд.

Существуют два подхода к организации управления: первый – жесткое (схемное) управление, второй микропрограммное управление

Схемы, поясняющие эти способы организации управления, приведены на рис. 7.8 и 7.9.

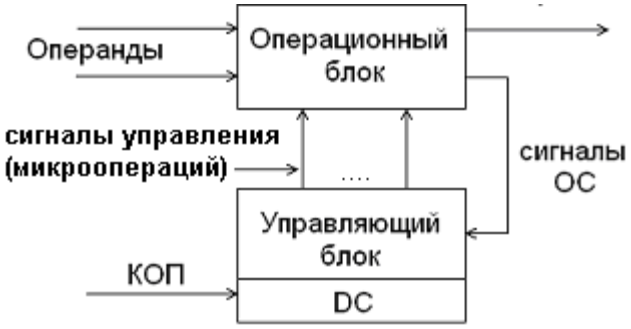


Рисунок 7.8 – Схема жесткого управления

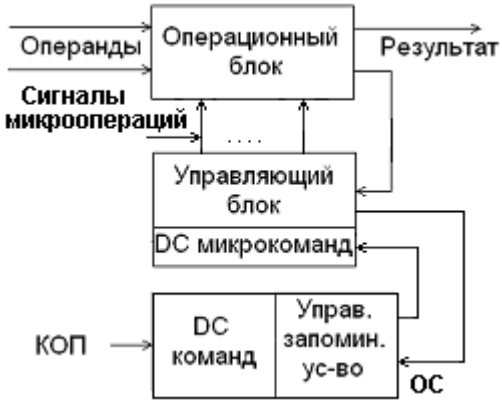


Рисунок 7.9 – Схема микропрограммного управления

Под действием кода операции (КОП) в управляющем блоке дешифратор команд (DC) запускает микропрограммный автомат, который в нужной последовательности вырабатывает сигналы микроопераций, которые, в свою очередь, порождают в операционном блоке требуемую обработку операндов. Грубо говоря, сколько команд – столько автоматов, при этом если требуется заменить код на другой, то выбирается другой микропрограммный автомат. Такая схема громоздкая и жесткая (изменить схему и список команд нельзя), но ее быстродействие достаточно высокое. Такой МП называется МП с фиксированным списком команд.

В управляющем запоминающем устройстве (УЗУ) хранятся микропрограммы различных операций. Под действием кода операции дешифратор команды выбирает в УЗУ нужную микропрограмму, считывает из неё первую микрокоманду и передаёт её в управляющий блок, в котором дешифратор микрокоманды запускает требуемый микропрограммный автомат, вырабатывающий сигналы микроопераций для операционного блока и сигнал ОС на считывание следующей микрокоманды из УЗУ, и т.д. пока не будет выполнена вся микропрограмма данной операции. Такое управление имеет гибкую систему команд. Меняя УЗУ, можно изменить список команд. Это так называемые аморфные машины. У них низкое быстродействие, т.к. здесь происходит многократное обращение к ЗУ, т.е. быстродействие всей системы определяется быстродействием блока памяти. Создание нужных команд тоже непростая задача.

Эти МП выпускают обычно сразу с определенным (базовым) списком команд, который при необходимости видоизменяют.

Иерархию управления можно представить схематично в виде рис. 7.10.

Рисунок 7.10 – Иерархия управления

Рассмотрим структурную схему обрабатывающей части микропроцессора с элементами жесткого управления. Она приведена на рис.7.11.

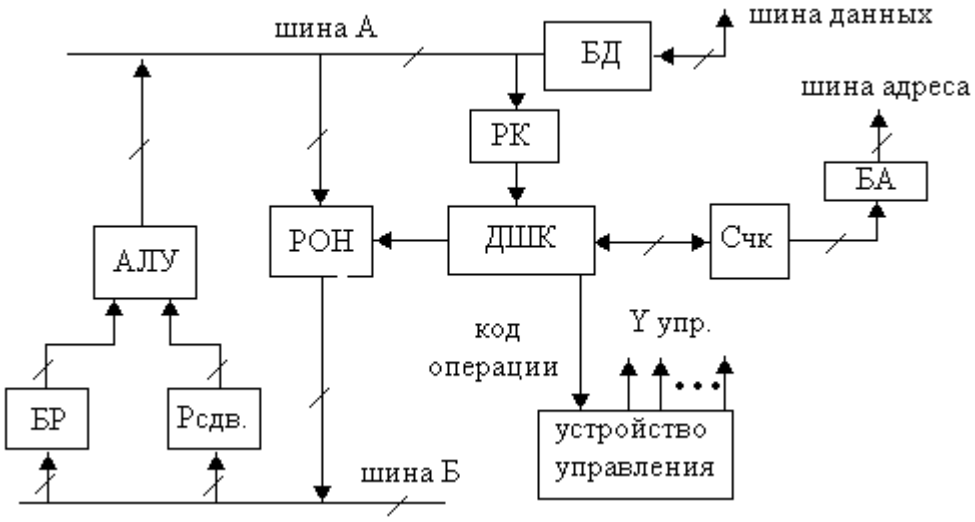


Рисунок 7.11 – Схема МП с элементами жесткого управления

Адрес команды, подлежащей выполнению, хранится в специальном регистре – счётчике команд или программном счётчике (Счк или PC –program counter). Этот адрес через буфер адреса (БА) по шине адреса поступает во внешнее запоминающее устройство (ЗУ), где хранится программа. Из программы считывается код команды и по шине данных через буфер данных (БД) он поступает на внутреннюю шину данных (шина А) и далее в регистр команд (РК), где он будет храниться на протяжении всего времени выполнения команды. Код команды расшифровывается дешифратором команды (ДШК), который анализирует отдельные поля команды и передает код операции в устройство управления, выходными сигналами которого охвачены все элементы МП. Результат операции может быть помещён в один из РОН или во внешнее ЗУ. В последнем случае ДШК передаёт в Счк адрес ячейки памяти, куда поместить результат, а сам результат передаётся через БД и шину данных в память.

7.4 Общие сведения о микропроцессорах

Микропроцессор (МП) – функционально законченное устройство обработки цифровой информации, управляемое хранимой в памяти программой и выполненное в виде одной или нескольких БИС.

Термин «микропроцессор» был впервые употреблен в 1972 г,, хотя годом рождения следует считать 1971 г., когда фирма Intel выпустила МП серии 4004 – «интегральное микропрограммируемое вычислительное устройство». Это был однокристальный процессор, имеющий 4-разрядный параллельный сумматор, 16 четырёхразрядных регистров, накапливающий сумматор и стек. Он был реализован на 2300 транзисторах, работал на частоте 750 кГц и мог выполнять 45 различных команд. С тех пор МП стал элементом, заменяющим сотни типов заказных БИС с фиксированной логикой и пригодным для реализации самых различных функций по обработке информации и управления.

По всем показателям темпы развития МП не имеют себе равных в технической области. Первоначально МП классифицировали по технологии изготовления. В связи с чем выделялось несколько поколений.

К первому поколению относят МП, выполненные по Р-МОП технологии, скорость выполнения команд которых достигала 60 мкс. Это четырёхразрядные МП с жестким управлением. Они требовали довольно много внешних согласующих цепей.

Ко второму поколению относят МП, выполненные по n-МОП технологии, здесь плотность компоновки выше, управление также жесткое, но список команд значительно шире и почти не требуют внешних согласующих устройств. Это восьмиразрядные МП, время выполнения команд до 8 мкс.

Третье поколение – МП, выполненные по биполярным технологиям. Плотность компоновки невысокая, поэтому выпускались двух- или четырёхразрядные секции, охваченные общим микропрограммных управлением. Это МП с наращиваемой разрядностью. Время выполнения команд до 0.1 мкс.

Четвертое поколение – это развитие второго поколения, используя Н-МОП, В-МОП, К-МОП технологии. Разрядность 8, 16, 32, время выполнения команд до 0,5 мкс. Управление схемное, жесткое. Список команд значительно расширен. Не нужны согласующие устройства.

В настоящее время развитие МП идет по пути совмещения различных технологий. Это уже многослойные БИС, и речь идет о толщине плёнки, в которой и создаются отдельные функциональные узлы МП. Так, существуют технологии: 0,5 мкм, 0,25 мкм, 0,18 мкм, 0,12 мкм. Повышение плотности компоновки позволило поднять тактовую частоту МП до единиц ГГц. Изменилась и архитектура (стиль постройки, представление МП с точки зрения программиста) микропроцессоров. Широкое внедрение RISC архитектуры (процессоры с сокращенным набором команд) позволило дополнительно увеличить быстродействие МП. Здесь команды короткие и быстро выполняются, нет обращения к ЗУ, а используется обращение только к регистрам.

Существующие фирмы - изготовители МП оказались неспособными в одиночку решить многие технические проблемы, и они пришли к выводу: надо

объединяться! Некто древний и великий (Теренций) сказал: «Когда двое делают одно и то же, это уже не одно и то же».

Именно бесперспективность дальнейшего одиночного развития побудила фирму Intel заключить соглашение с компанией Hewlett Packard (HP). Их совместный RISC процессор – PA-9000 разрядностью 64 бит выполнял три команды за один такт, изготовлен по 0,2 мкм технологии и состоял из 5,5 млн вентилей.

Другая «команда»: фирмы IBM, Motorola и Apple тоже разработала RISC процессор PPC-604, который не уступает по показателям PA-9000, но значительно проще и дешевле (0,5 мкм технология, 3,6 млн вентилей, выполняет 4 команды за один такт).

Безусловно, эти фирмы занимаются и самостоятельными разработками. Компания HP выпустила МП PA-8800. Это двухпроцессорная система. Она состоит из 300 млн транзисторов, размещена на одном кристалле площадью 366 $мм^2$, выполнена по 0,13 мкм, системная шина 128 бит, тактовая частота 400 МГц.

Крупнейший производитель МП в России коллектив «Эльбрус», который 40 лет разрабатывает суперкомпьютеры.

«Эльбрус-1» (1978 г.) это 10-процессорная система с разделяемой памятью. В системе использовались: перестановка последовательности выполнения операций, спекулятивное исполнение и переименование регистров, теговая защита и динамическая проверка типов данных. Все это было в серийной машине за 15 лет до появления первых зарубежных RISC процессоров! Тег – пометка, ярлык. Теговая архитектура заключалась в наличии в данных избыточных битов для указания типа данных. Это мощная форма аппаратной защиты памяти.

«Эльбрус-3» (1991г.) – это 16-разрядный процессорный комплекс. Несмотря на несовершенную технологию изготовления кристаллов, его быстродействие было в два раза выше, чем у суперкомпьютера США (CRAY-MP).

Ведущим разработчиком процессора Pentium III (фирма Intel) был В.Пентковский – выходец из команды «Эльбрус».

В 90-е годы XX в. реорганизованная компания «Эльбрус-интернэшнл» занималась совершенствованием низковольтной КМОП технологии. Результат этой работы – процессор E2k. В нём впервые реализованы параллельное выполнение команд и технология двоичной компиляции. Процессор E2k в 3-5 раз превосходит по быстродействию аналогичный совместный процессор фирм Intel и HP – «Merced». E2k имеет меньший размер кристалла и себестоимость, защищен 70 патентами в США.

Другой творческий коллектив НТЦ «Модуль» создаёт МП по «беспроизводственной» схеме: архитектура и инструментальное программное обеспечение разработаны в России, а производство – за рубежом по технологии 0,5 мкм.

Очередной шаг – технология 0,2 мкм, за рубежом 0,09 мкм. Процессор NM 6403 по техническим параметрам находится на уровне лучших мировых достижений. Аналогичный процессор TMS320TC40 (сигнальный процессор с плавающей запятой компании Texas Instrument) при такой же тактовой частоте имеет в 5-10 раз меньшую производительность. Лицензию на использование

схемотехнических решений приобрело Германское отделение Fujitsu. Ядро этого процессора использовано в составе собственных микросхем Fujitsu для применения в высокоскоростных системах передачи данных и сотовой телефонии.

7.5 Классификация микропроцессоров

Введем некоторые понятия.

МПС – микропроцессорная система – любая вычислительная, контрольно-измерительная или управляющая система, обрабатывающим элементом которой является микропроцессор.

МПК – микропроцессорный комплекс (набор) – семейство совместимых по конструктивным и электрическим характеристикам БИС, специально разработанных для построения различных микропроцессорных систем. В него входят: МП, ПЗУ, ОЗУ, БИС управления и др. Обычно МПК включают от 5 до 20 различных интегральных схем.

Микроконтроллер – устройство логического управления, выполненное на основе одной или нескольких БИС.

МикроЭВМ – вычислительная МПС с устройствами ввода / вывода, собственным программным обеспечением и оформленная в виде автономного прибора.

Однокристальная микроЭВМ это микро ЭВМ, выполненная в виде одной БИС, в которой размещены: МП, устройство управления, ПЗУ, ОЗУ и каналы ввода / вывода.

МП используют в двух направлениях:

а) как вычислители, для решения традиционных математических задач;

б) вместо устройств с жесткой логикой работы. В этом случае МП заменяет 100-200 и более корпусов интегральных схем и называется микроконтроллером. У микроконтроллера настройка и переделка проще. Выше надежность. Меньшее время разработки.

В настоящее время развитие микропроцессоров идет в двух направлениях:

- CISC процессоры (процессоры с полным набором команд);
- RISC процессоры (процессоры с сокращённым набором команд).

В процессорах с полным набором команд производители стараются увеличить количество команд, которые может выполнять микропроцессор, тем самым увеличивая длину и время выполнения каждой команды. Как правило, это процессоры с микропрограммным управлением.

В процессорах с сокращённым набором команд (жёсткое управление) декодирование и исполнение команды производится аппаратно, поэтому количество команд ограничено. В этих процессорах команда переводится сразу в сигналы микроопераций. Преимуществом этого типа процессоров является то, что команда может быть в принципе выполнена за один такт (не требуется выполнение микропрограммы).

В большинстве случаев быстродействие RISC процессоров выше, чем CISC процессоров, однако может оказаться, что общий объём исполняемой

программы для RISC процессора больше объёма подобной программы для CISC процессора.

По **архитектуре** микропроцессоры отличаются огромным разнообразием. Тем не менее, можно определить два направления построения микропроцессоров:

- микропроцессоры с регистрами общего назначения;
- аккумуляторные микропроцессоры.

В микропроцессорах с регистрами общего назначения математические операции могут выполняться над любой ячейкой памяти. Принципиальным отличием аккумуляторных процессоров является то, что математические операции могут производиться только над одним особым регистром – аккумулятором. Для того чтобы произвести операцию над произвольной ячейкой памяти, её содержимое необходимо скопировать в аккумулятор, произвести требуемую операцию, а затем скопировать полученный результат в произвольную ячейку памяти.

Сегодня в чистом виде не существуют ни та, ни другая архитектура. Все выпускаемые в настоящее время процессоры обладают системой команд с признаками как аккумуляторных процессоров, так и микропроцессоров с регистрами общего назначения.

По **способу работы с памятью** существуют два основных принципа построения микропроцессоров:

- гарвардская архитектура;
- архитектура Фон Неймана.

В **Гарвардской архитектуре** память принципиально разделена на две части:

- память программ;
- память данных.

В Гарвардской архитектуре невозможно производить операцию записи в память программ, что исключает возможность разрушения управляющей программы при неправильных действиях над данными. Кроме того, в ряде случаев для памяти программ и памяти данных выделяются отдельные шины обмена данными. Эти особенности определили области применения Гарвардской архитектуры микропроцессоров, а именно, в микроконтроллерах, где требуется обеспечить высокую надёжность работы аппаратуры, и в сигнальных процессорах, где необходима высокая скорость выполнения программы за счёт одновременного считывания управляющих команд и обрабатываемых данных.

В архитектуре Фон Неймана нет деления памяти. Здесь возможна работа над управляющими программами точно так же, как над данными. Это позволяет производить запись в произвольное место памяти процессора. Любой участок памяти может служить как памятью программ, так и памятью данных. Причём в разные моменты времени одна и та же область памяти может использоваться и как память программ, и как память данных. Эта особенность архитектуры позволяет наиболее гибко управлять работой микропроцессорной системы, но создаёт принципиальную возможность искажения управляющей программы, что понижает надёжность работы аппаратуры. Архитектура Фон Неймана используется в универсальных компьютерах и в некоторых видах микроконтроллеров.

7.6. Обобщенная схема микропроцессорной системы

Обобщённая схема микропроцессорной системы (МПС) обычно имеет три шины: шину адреса (ША), шину данных (ШД) и шину управления (ШУ). Эта схема приведена на рис.7.12.

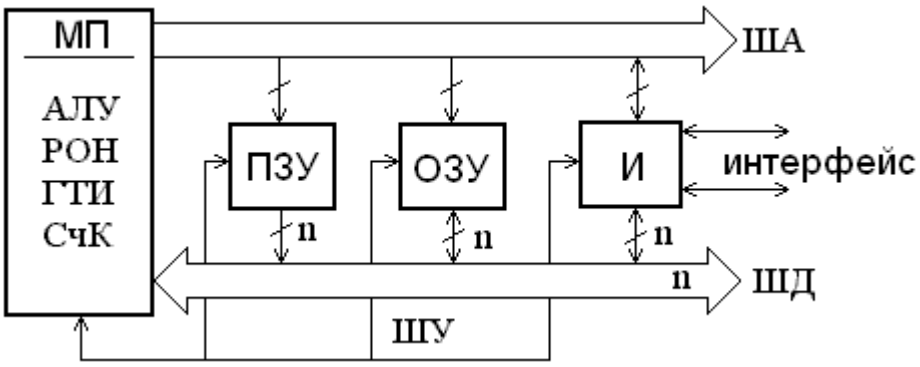


Рисунок 7.12 – Трёхшинная организация МПС

Схема содержит процессор (МП), память программ (ПЗУ), память данных (ОЗУ) и устройства ввода / вывода (И - интерфейс).

Для МПС характерна трехшинная организация ША, ШД, ШУ. Это внешние по отношению к МП шины. Совокупность ША, ШД и ШУ часто называют системной шиной. Возможны разные организации шин: одна двухнаправленная ШД или две однонаправленных (одна из которых является входной для МП, другая выходной). МП в заданной последовательности выбирает команды из ПЗУ и выполняет их. Для выбора команд МП последовательно устанавливает на ША адреса (номера) ячеек памяти. Код команды, выбранный из заданной ячейки, вводится в МП по ШД и вызывает выполнение соответствующей операции в МП.

Для хранения промежуточных результатов или для ввода на обработку совокупности исходных данных имеется память данных – ОЗУ, которая в отличие от ПЗУ должна работать как в режиме чтения, так и в режиме записи информации. Адреса ячеек для записи данных формируются МП и передаются в блок памяти по ША. Обмен информацией между МП и ОЗУ производится по ШД словами, разрядность которых равна разрядности ШД.

Ввод исходной информации на обработку и вывод результатов осуществляются через блок интерфейса (И), который служит для сопряжения сигналов МПС и внешних устройств.

В качестве внешних устройств (ВУ) могут использоваться любые ВУ из состава микроЭВМ: дисплеи, печатающие устройства, накопители на магнитных дисках, модемы и т.д. Однако при использовании МПС непосредственно в блоках РЭА роль ВУ ввода играют АЦП, преселекторы (устройства предварительной обработки информации), обнаружители сигналов и т.п. ВУ вывода – индикаторы, исполнительные устройства, блоки передачи и т.п.

Для синхронизации МПС в целом используют генератор тактовых импульсов (ГТИ). Период следования импульсов ГТИ является машинным тактом.

Для более подробного знакомства с микропроцессорной техникой можно воспользоваться доступной периодической и научно-технической литературой, а также Интернет-ресурсами.

Контрольные вопросы

1. Каковы общая структура и принцип действия ЭВМ?
2. Каковы основные функции устройства управления микропроцессоров?
3. Каковы классификационные признаки микропроцессоров?
4. В чем принципиальное отличие Гарвардской архитектуры от архитектуры Фон Неймана?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Смирнов Ю.А.* Основы микроэлектроники и микропроцессорной техники / Ю.А. Смирнов, С.В. Соколов, Е.В. Титов. – СПб.: Лань, 2013. 496 с.
2. *Микушин А.В.* Цифровые устройства и микропроцессоры: учеб. пособие / А.В. Микушин, А.М. Сажнев, В.И. Синин. – СПб.: БХВ-Петербург, 2010. 832с.
3. *Медведев М.Ю.* Программирование промышленных контроллеров / М.Ю. Медведев, В.Х. Пшихопов. – СПб.: Лань, 2011. 288 с.
4. *Лукинов А.П.* Проектирование мехатронных и робототехнических устройств. – СПб.: Лань, 2012. 608 с.
5. *Коледов Л.А.* Технология и конструкция микросхем, микропроцессоров и микросборок. – СПб.: Лань, 2009. 400 с.
6. *Смирнов Ю.А.* Основы нано- и функциональной электроники / Ю.А. Смирнов, С.В. Соколов, Е.В. Титов. – СПб.: Лань, 2013. 320 с.
7. *Угрюмов Е.П.* Цифровая схемотехника. – СПб.: БХВ – Петербург, 2001. 528с.
8. *Нарышкин А.К.* Цифровые устройства и микропроцессоры: учеб. пособие для студентов высш. учеб. заведений. – М., 2006. 321 с.
9. Цифровая и вычислительная техника: учеб. для вузов / Э.В. Евреинов [и др.]; под ред. Э.В. Евреинова. – М.: Р и С, 1991. 464 с.
10. *Каган Б.М.* ЭВМ и системы: учеб. пособие для вузов. – М.: Энергоатомиздат, 1991. 592 с.
11. *Гольденберг Л.М.* Цифровые устройства и микропроцессорные системы. Задачи и упражнения: учеб. пособие для вузов / Л.М. Гольденберг, В.А. Малеев, Г.Б. Малько. – М.: Р и С, 1992. 256 с.
12. Журнал «Современные компьютерные технологии» http://www.cta.ru
13. Журнал «Мир компьютерной автоматизации» http://mka.org.ru
14. Журнал «CHIP NEWS» http://www.chipnews.ru
15. http://www.chipinfo.ru/wb/index.html CHIPINFO — все об электронике и компонентах.
16. www.logicnet.ru/~electron RUSSIAN ELECTRONIC
17. http://www.promelec.ru Промэлектроника.
18. http://www.gaw.ru/index.cgi РЫНОК МИКРОЭЛЕКТРОНИКИ (справочник).
19. http://www.fulcrum.ru/index.htm ЭЛЕКТРОННЫЕ КОМПОНЕНТЫ СО ВСЕГО МИРА.
20. http://www.atmel.ru/ServerContents.htm Русскоязычная страница ATMEL.

Содержание

Сажнев Александр Михайлович
Тырышкин Игорь Сергеевич

Цифровые устройства и микропроцессоры

Учебное пособие

Редактор Н.К. Крупина
Компьютерная верстка

Подписано к печати 2015 г.
Формат 60х84 1/16. Тираж 50 экз.
Объем 8,8 уч.- изд. л., усл. печ. л.
Изд. № 76. Заказ №

Отпечатано в Издательском центре НГАУ «Золотой колос»
630039, Новосибирск, ул. Добролюбова, 160