Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет радиотехники и электроники

Кафедра информационных радиотехнологий

А. А. Будько, Т. Н. Дворникова

ФУНКЦИОНАЛЬНЫЕ УСТРОЙСТВА РАДИОСИСТЕМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальностей 1-39 01 02 «Радиоэлектронные системы», 1-39 01 04 «Радиоэлектронная защита информации»

Минск БГУИР 2021

УДК 621.396.2(076.5)
ББК 32.844.1я73
Б90

Р е ц е н з е н т ы :

кафедра электротехники и электроники
Белорусского национального технического университета
(протокол №7 от 30.01.2020);

директор Института современных технологий связи
учреждения образования «Белорусская государственная академия связи»
кандидат технических наук, доцент Е. В. Новиков

Будько, А. А.

Б90 Функциональные устройства радиосистем : учеб.-метод. пособие / А. А. Будько, Т. Н. Дворникова. – Минск : БГУИР, 2021. – 218 с. : ил.
ISBN 978-985-543-566-3.

Содержит описание девяти практических и девяти лабораторных работ. Предназначено для изучения функциональных устройств радиосистем и приобретения практических навыков измерения их основных параметров и характеристик, а также навыков виртуального схемотехнического моделирования и программирования.

Включает теоретические сведения по проектированию и применению наиболее распространенных узлов функциональных устройств радиосистем.

УДК 621.396.2(076.5)
ББК 32.844.1я73

ISBN 978-985-543-566-3

Содержание

1 Практическая работа №1

ПРЕДСТАВЛЕНИЕ И ПРЕОБРАЗОВАНИЕ ЧИСЕЛ В СИСТЕМАХ СЧИСЛЕНИЯ, ПРИМЕНЯЕМЫХ В ЦИФРОВОЙ ТЕХНИКЕ

1.1 Цель работы

Приобретение практических навыков для представления и преобразования чисел в системах счисления, применяемых в цифровой технике.

1.2 Теоретические сведения

В цифровой технике используются десятичная, двоичная, восьмеричная и шестнадцатеричная системы счисления.

В компьютерах и других цифровых системах используются двоичные сигналы и при этом, если требуется обработка числовой информации, буквенной и другой, то эта информация должна быть представлена в двоичной форме. Для этого используется процесс кодирования числовой, буквенной и другой информации в комбинации 0 и 1, т. е. информация представляется в виде двоичного кода.

Существует большое количество двоичных кодов, которые используются для различных целей, например: для выполнения арифметических операций; для ввода данных; для обнаружения ошибок, которые могут случиться при передаче информации от одной системы к другой; для коррекции ошибок, если необходимо не только обнаружить эти ошибки, но и исправить их и т. п. Выбор того или иного кода зависит от того, для каких целей он будет использоваться. В цифровых системах могут использоваться для представления одной и той же информации различные коды, поэтому может возникнуть необходимость в преобразовании информации, представленной в одном коде, в другой код. Системы счисления (СС) могут быть позиционными и непозиционными.

Непозиционные СС используются редко, в основном – для целей нумерации. Примером такой системы является римская СС с символами, таблица 1.1.

Таблица 1.1 – Соотношение римских символов и десятичных чисел

Десятичные числа	1	5	10	50	100	500	1000	…
Римские символы	*I*	*V*	*X*	*L*	*C*	*D*	*M*	…

В общем случае любая позиционная система счисления определяется группой цифр и правилами, по которым осуществляются арифметические операции, такие как сложение, умножение и т. д.

Основной недостаток непозиционных СС – это большое число различных символов, громоздкость чисел и сложность выполнения арифметических операций.

Число в позиционной системе счисления представляется группой цифр.

В общем случае это число имеет две части – целую и дробную, разделенные точкой:

$$N_r = \underbrace{d_{n-1}d_{n-2}\dots d_i\dots d_1d_0}_{\text{целая часть}}\ .\ \underbrace{d_{-1}d_{-2}\dots d_{-f}\dots d_{-m}}_{\text{дробная}} = \sum_{i=-m}^{n-1}(d_i r_i), \qquad (1.1)$$

точка

где N – число;

r – основание или база системы счисления;

n – число цифр в целой части числа;

m – число цифр в дробной части;

d_{n-1} – наиболее значимая цифра;

d_{-m} – наименее значимая цифра;

$0 \le (d_i \text{ или } d_{-f}) \le r-1$.

Цифры в числе располагаются друг возле друга, и каждая позиция в числе имеет свой вес, или индекс значимости, который заранее определен по какому-то правилу. Любое число X, записанное в позиционной СС, можно представить в виде полинома по формуле (1.2).

Любое число, записанное в позиционной системе, может быть представлено в виде полинома с основанием системы счисления r по формуле

$$N_r = d_{n-1}\cdot r^{n-1} + d_{n-2}\cdot r^{n-2} + \dots + d_{-m}\cdot r^{-m} = \sum_{i=-m}^{n-1}(d_i r_i). \qquad (1.2)$$

Рассмотрим системы счисления, которые используются в цифровых устройствах и системах.

Например, десятичное число $X_1 = 43.25$ по формуле (1.2) имеет вид

$$X_1 = 43.25_{(10)} = 4\cdot 10^1 + 3\cdot 10^0 + 2\cdot 10^{-1} + 5\cdot 10^{-2}. \qquad (1.3)$$

На величину основания позиционной СС нет ограничения. Поэтому возникает вопрос, какая величина основания является оптимальной для применения в цифровых устройствах.

Наилучшая система счисления должна обеспечивать простоту технической реализации, простоту выполнения арифметических операций, малые затраты оборудования при построении устройств, высокую помехоустойчивость кодирования цифр и простоту применения математического аппарата для анализа и синтеза цифровых устройств. Краткий анализ систем счисления показывает, что для построения аппаратных средств в цифровых устройствах следует применять двоичную СС.

В двоичной СС ($P = 2$) используются два символа – это цифры 0 и 1. Любое число в двоичной СС записывается в виде последовательности нулей и единиц, расставленных согласно формуле (1.2). Например, уже рассмотренное нами десятичное число $X_1 = 43.25$ в двоичной СС запишется следующим образом:

$$X_1 = 101011.01_{(2)} = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + \\ + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 32 + 0 + 8 + 0 + 2 + 1 + 0 + 0.25 = 43.25_{(10)}. \quad (1.4)$$

Двоичная СС имеет один недостаток: в ней трудно оценить величину чисел, следовательно, ее использование приводит к необходимости преобразования исходных десятичных чисел в двоичную СС, а результатов – из двоичной в десятичную СС. Для таких преобразований чисел требуются достаточно сложные аппаратные средства и дополнительное время. Для преодоления этого недостатка применяется двоично-десятичная СС, представляющая собой СС с основанием $P = 10$, цифры которой закодированы в виде четырехразрядных двоичных чисел (тетрад). Четырехразрядное двоичное число позволяет получить $2^4 = 16$ наборов, из которых при двоично-десятичном кодировании используются только 10. Поэтому возможны различные варианты кодирования десятичных цифр в двоичной СС. Наибольшее распространение получил код 8421 – *BCD*-код. Цифры 8421 указывают вес соответствующего двоичного разряда двоично-десятичного числа. Например, десятичное число $X_1 = 43.25$ в коде 8421 запишется следующим образом:

$$X_1 = 43.25_{(10)} = \underbrace{0100}_{4}\ \underbrace{0011}_{3}.\underbrace{0010}_{2}\ \underbrace{0101}_{5}{}_{(2-10)}. \quad (1.5)$$

Для преобразования десятичных чисел в код 8421 и обратно используются простые типовые узлы цифровой техники, такие как шифраторы и дешифраторы.

Двоично-десятичные СС используются как для ввода/вывода, так и для обработки числовых данных.

В технике программирования в качестве вспомогательной широкое применение получила шестнадцатеричная СС, у которой основание $P = 16$, т. е. используется 16 символов – это цифры от 0 до 9 и шесть заглавных букв латинского алфавита: *A, B, C, D, E, F*. У этой СС основание является целой степенью числа 2, в результате по аналогии с двоично-десятичной СС ее можно определить как систему с основанием $P = 16$, у которой символы закодированы в двоичной СС, что значительно облегчает перевод шестнадцатеричных чисел в двоичную СС и обратно.

Для преобразования шестнадцатеричного числа в двоичную систему счисления достаточно перевести в двоичную СС каждый символ исходного числа и записать эти символы в виде тетрад.

Например, преобразуем шестнадцатеричное число $X = 7B.6F$ в двоичную СС:

$$X_1 = 7B.6F_{(16)} = \underbrace{0111}_{7}\ \underbrace{1011}_{B}\ .\ \underbrace{0110}_{6}\ \underbrace{1111}_{F}{}_{(2)}. \qquad (1.6)$$

Для обратного преобразования двоичное число разбивают на тетрады влево и вправо от точки, а затем содержимое каждой тетрады заменяют соответствующим символом шестнадцатеричной СС. Если левая и правая тетрады получаются неполными, то их следует дополнить нулями.

Правило перевода целых чисел из одной позиционной СС в другую. Исходное целое число необходимо последовательно делить на основание новой СС до тех пор, пока не получится частное, у которого целая часть равна нулю. Деление необходимо производить в исходной СС. Результат перевода записывается из остатков от последовательного деления, причем последний остаток будет старшим разрядом (СР) числа в новой СС.

Процесс деления сначала самого числа, а затем целых частей получаемых частных на один и тот же делитель называется **последовательным делением.**

Пример 1. Преобразуем десятичное число $X = 29$ в двоичную и шестнадцатеричную СС в соответствии с рисунком 1.1.

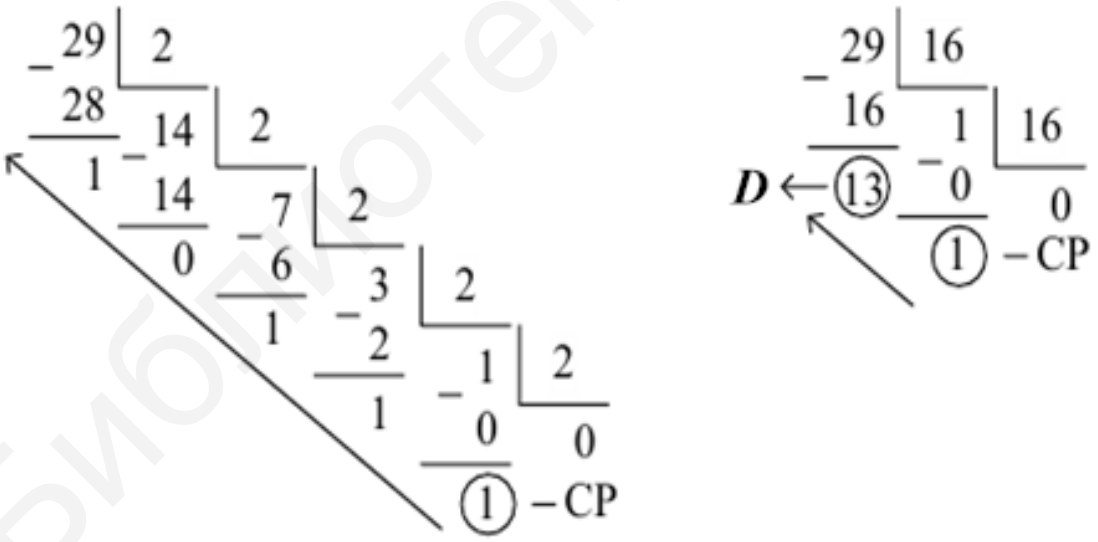


Рисунок 1.1 – Преобразования десятичного числа

Стрелкой показан порядок записи числа в новой СС.
Ответ: $X = 29_{(10)} = 11101_{(2)} = 1D_{(16)}$.

Правило перевода правильных дробей из одной позиционной СС в другую. Исходную правильную дробь необходимо последовательно умножать на основание новой СС до тех пор, пока в новой дроби не будет нужного количества цифр, которое определяется требуемой точностью перевода. Результат перевода записывается из целых частей произведений, получающихся при последовательном умножении, причем первая целая часть будет старшим разрядом результата. Умножение выполняется в исходной СС.

Процесс умножения сначала самой исходной дроби, а затем дробных частей получаемых произведений на один и тот же множитель называется **последовательным умножением**.

Пример 2. Переведем в двоичную и шестнадцатеричную СС правильную дробь $X = 0.325_{(10)}$ с точностью четыре знака после точки в соответствии с рисунком 1.2.

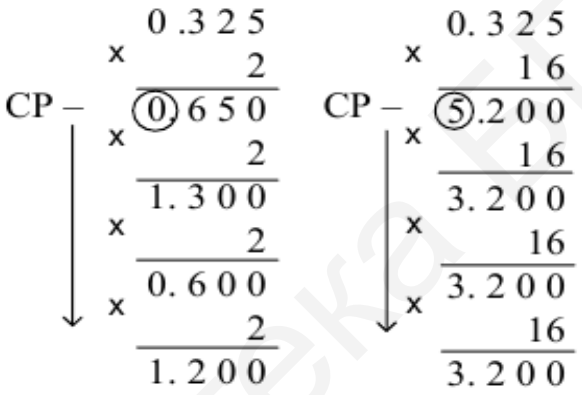


Рисунок 1.2 – Перевод в двоичную и шестнадцатеричную СС

Стрелкой показан порядок записи правильной дроби в новой СС.

Ответ: $X = 0.325_{(10)} = 0.0101_{(2)} = 0.5333_{(16)}$.

Заметим, что в случае если дробная часть на некотором шаге становится равной нулю, то процесс преобразования на этом заканчивается.

При переводе неправильных дробей отдельно преобразуют целую и дробную части по соответствующим правилам, приведенным выше, а затем записывают результаты перевода через точку в новой СС.

Пример 3. Преобразуем неправильную дробь $X = 29.325_{(10)}$ в двоичную и шестнадцатеричную СС с точностью четыре знака после точки:

$$X = 29.325_{(10)} = 11101.0101_{(2)} = 1D.5333_{(16)}. \quad (1.7)$$

Рассмотренный расчетный метод удобен в том случае, если исходной является десятичная СС. Если же перевод осуществляется из недесятичной СС, то вычисления затруднительны. В этом случае для преобразования чисел можно использовать формулу (1.1), причем расчеты ведутся в новой СС.

Пример 4. Преобразуем в десятичную СС двоичное число $X =$ $=11101.0101$:

$$X = 11101.0101_{(2)} = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + \\ +0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = 16 + 9 + 4 + 0 + 1 + 0 + \\ +0.25 + 0 + 0.0625 = 29.3125_{(10)} \approx 29.325_{(10)}. \quad (1.8)$$

Из выражения (1.8) следует, что целая часть числа переводится точно, а дробная – приближенно.

Таким образом, для перевода десятичных чисел в другую позиционную СС используется метод последовательного деления/умножения, а при обратном переводе исходное число записывается в виде полинома по формуле (1.2) и выполняются необходимые расчеты.

1.3 Порядок выполнения практической работы

Задание 1. Выполнить следующие преобразования из одной системы счисления в другую согласно варианту по таблице 1.2.

Таблица 1.2 – Исходные данные для задания 1 подраздела 1.3

Номер варианта	Задание
1	$1101011_2 = X_{16}$
2	$174003_8 = X_2$
3	$1011111_2 = X_{16}$
4	$67.24_8 = X_2$
5	$10100.1101_2 = X_{16}$
6	$F3F5_{16} = X_2$
7	$11011001_2 = X_8$
8	$AB3D_{16} = X_2$
9	$101111.0111_2 = X_8$
10	$15C.38_{16} = X_2$

Задание 2. Преобразовать следующие восьмеричные числа в двоичные и шестнадцатеричные согласно варианту по таблице 1.3.

Таблица 1.3 – Исходные данные для задания 2 подраздела 1.3

Номер варианта	Задание
1	$1023_8 = X_2 = X_{16}$
2	$761302_8 = X_2 = X_{16}$
3	$163417_8 = X_2 = X_{16}$
4	$552273_8 = X_2 = X_{16}$
5	$5436.15_8 = X_2 = X_{16}$

Номер варианта	Задание
6	$13705.207_8 = X_2 = X_{16}$
7	$511_8 = X_2 = X_{16}$
8	$10.27_8 = X_2 = X_{16}$
9	$0.112_8 = X_2 = X_{16}$
10	$1.245_8 = X_2 = X_{16}$

Задание 3. Преобразовать следующие шестнадцатеричные числа в двоичные и восьмеричные согласно варианту по таблице 1.4.

Таблица 1.4 – Исходные данные для задания 3 подраздела 1.3

Номер варианта	Задание
1	$1023_{16} = X_2 = X_8$
2	$7E6A_{16} = X_2 = X_8$
3	$ABCD_{16} = X_2 = X_8$
4	$C350_{16} = X_2 = X_8$
5	$9E36.7A_{16} = X_2 = X_8$
6	$DEA.B1F_{16} = X_2 = X_8$
7	$F1150_{16} = X_2 = X_8$
8	$C1.EE0_{16} = X_2 = X_8$
9	$01.AF_{16} = X_2 = X_8$
10	$F66F_{16} = X_2 = X_8$

Задание 4. Преобразовать следующие числа в десятичные согласно варианту по таблице 1.5.

Таблица 1.5 – Исходные данные для задания 4 подраздела 1.3

Номер варианта	Задание
1	$1101011_2 = X_{10}$
2	$174003_8 = X_{10}$
3	$10110111_2 = X_{10}$
4	$67.24_8 = X_{10}$
5	$10100.1101_2 = X_{10}$
6	$F3A5_{16} = X_{10}$
7	$12010_3 = X_{10}$
8	$AB3D_{16} = X_{10}$
9	$7156_8 = X_{10}$
10	$15C.38_{16} = X_{10}$

Задание 5. Выполнить следующие преобразования из одной системы счисления в другую согласно варианту по таблице 1.6.

Таблица 1.6 – Исходные данные для задания 5 подраздела 1.3

Номер варианта	Задание
1	$125_{10} = X_2$
2	$349_{10} = X_8$
3	$209_{10} = X_2$
4	$9714_{10} = X_2$
5	$132_{10} = X_2$
6	$23851_{10} = X_{16}$
7	$727_{10} = X_8$
8	$57190_{10} = X_{16}$
9	$1435_{10} = X_8$
10	$65113_{10} = X_{16}$

1.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1–5 согласно варианту.
3. Вывод.
4. Ответы на контрольные вопросы.

1.5 Контрольные вопросы

1. Каковы отличия позиционной системы счисления от непозиционной системы счисления?
2. Какое основание имеет двоично-десятичная система счисления?
3. Каким образом осуществляется перевод неправильной дроби из десятичной системы счисления в любую другую систему счисления?
4. Приведите полином, используемый для перевода числа в десятичную систему счисления.
5. В чем заключаются особенности перевода чисел из десятичной системы счисления в двоично-десятичную?

2 Практическая работа №2

ПРЕДСТАВЛЕНИЕ И КОДИРОВАНИЕ ДВОИЧНЫХ ЧИСЕЛ В ЦИФРОВЫХ УСТРОЙСТВАХ. АРИФМЕТИЧЕСКАЯ ОБРАБОТКА ЧИСЕЛ В ЦИФРОВЫХ УСТРОЙСТВАХ

2.1 Цель работы

Изучение и приобретение практических навыков кодирования двоичных чисел, а также методов обработки чисел.

2.2 Теоретические сведения

Способы кодирования двоичных чисел со знаком. Кодирование двоичных чисел в цифровых устройствах необходимо для представления чисел со знаком, т. е. упрощения операции вычитания. Двоичные числа, представленные в естественной и нормальной формах, кодируются в цифровых устройствах с помощью специальных машинных кодов: прямого, обратного и дополнительного.

Прямой код является простейшим машинным кодом и получается при кодировании в числе только знака. Прямой код положительного числа совпадает с его изображением в естественной форме. Прямой код отрицательного числа совпадает с его изображением в естественной форме, за исключением знакового разряда, в который ставится единица.

Пример 1. Представим в прямом коде двоичные числа X_1 и X_2 (рисунок 2.1).

$$X_1 = +0.11010, \quad [X_1]_{\text{пр}} = 0.11010,$$

$$X_2 = -0.11010, \quad [X_2]_{\text{пр}} = 1.11010.$$

Рисунок 2.1 – Представление двоичных чисел в прямом коде

Знаковый разряд числа веса не имеет и отделяется от цифр числа запятой. Прямой код используется для хранения чисел в памяти, при вводе-выводе информации, а также при выполнении операций умножения и деления. Недостатком прямого кода являются разные правила счета для положительных и отрицательных чисел. Поэтому в цифровых устройствах вычитание заменяют операцией сложения чисел в обратном или дополнительном кодах.

Дополнительный код. Положительное двоичное число совпадает с его прямым кодом. Дополнительным кодом отрицательного двоичного числа называется его дополнение до граничного числа $X_{\text{гр}}$. Таким образом, в знаковом

разряде записывается единица, а во всех других разрядах цифры заменяются на взаимно обратные, после чего к младшему разряду числа добавляется единица.

Аналитически дополнительный код для двоичных чисел, меньших единицы, вычисляется по выражению

$$[X]_{\text{доп}} = \begin{cases} X, \text{если } X \geq 0, \\ X + X_{\text{гр}}, \text{если } X < 0, \end{cases} \tag{2.1}$$

где $X_{\text{гр}}$ – граничное число, $X_{\text{гр}} = |X|_{\max} + 10_{(2)}^{-m} = 1{,}0_{(2)}$.

Пример 2. Представим в дополнительном коде двоичные числа X_1 и X_2 (рисунок 2.2).

$X_1 = +0.11010;$ $\quad [X_1]_{\text{доп}} = 0.11010$

$X_2 = -0.11010;$ $\quad [X_2]_{\text{доп}} = 1.00101$

$+\ 1$

1.00110

Рисунок 2.2 – Представление двоичных чисел в дополнительном коде

Обратное преобразование, т. е. получение двоичного числа в естественной форме из дополнительного кода выполняется по тому же правилу, что и получение дополнительного кода. Для этого следует вначале заменить все цифры, на взаимно обратные, после чего добавить единицу к младшему разряду. В результате получается прямой код, затем выполняется обратное преобразование из прямого кода к естественной форме числа.

Пример 3. Выполним обратное преобразование из дополнительного кода в естественную форму двоичного числа (рисунок 2.3).

$[X]_{\text{доп}} = 1.00110$ $\quad [X]_{\text{пр}} = 1.11001$

$+\ 1$

$1.11010;$

$X = -0.11010_{(2)}.$

Рисунок 2.3 – Обратное преобразование числа из дополнительного кода в естественную форму

Полученный результат преобразования совпадает с исходным значением числа X_2 (см. рисунок 1.2).

Обратный код. Положительное двоичное число совпадает с его прямым кодом. Обратным кодом отрицательного двоичного числа называется его дополнение до максимального числа $X_{\max}$, представляемого в данной разрядной сетке. Таким образом, в знаковом разряде записывается единица, а во всех остальных разрядах цифры заменяются на взаимно обратные.

Аналитически обратный код для двоичных чисел, меньших единицы, определяется из выражения

$$[X]_{\text{обр}} = \begin{cases} X, \text{если } X \geq 0, \\ X + X_{\max}, \text{если } X < 0. \end{cases} \quad (2.2)$$

Пример 4. Представим в обратном коде числа X_1 и X_2 (рисунок 2.4).

$$X_1 = +0.11010, \quad [X_1]_{\text{обр}} = 0.11010,$$

$$X_2 = -0.11010, \quad [X_2]_{\text{обр}} = 1.00101.$$

Рисунок 2.4 – Представление двоичных чисел в обратном коде

Основное достоинство обратного кода по сравнению с дополнительным состоит в простоте процесса его формирования. Однако скорость вычитания чисел в этом коде несколько ниже, чем в дополнительном.

Обратное преобразование, т. е. получение двоичного числа в естественной форме из обратного кода выполняется по тому же правилу, что и получение обратного кода. Для этого следует вначале получить прямой код, меняя все цифры, за исключением знаковой, на взаимно обратные, а затем из прямого кода перейти к естественной форме числа.

Пример 5. Выполняем обратное преобразование из обратного кода в естественную форму двоичного числа (рисунок 2.5).

$$[X]_{\text{обр}} = 1.00101,$$

$$[X]_{\text{пр}} = 1.11010,$$

$$X = -0.11010_{(2)}.$$

Рисунок 2.5 – Представление двоичного числа в естественной форме

Основной операцией, которая используется в цифровых устройствах при различных вычислениях, является операция алгебраического сложения (сложение положительных и отрицательных чисел). Вычитание, умножение и деление также выполняются с помощью операции сложения и некоторых других действий.

Сложение двоичных чисел. В цифровых устройствах сложение чисел производится так же, как и при ручном счете, т. е. путем поразрядного суммирования чисел, начиная с младших разрядов (таблица 2.1).

Таблица 2.1 – Сложение двоичных чисел

Слагаемые		Сумма	Перенос
X_1	X_2	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Из таблицы 2.1 следует, что при сложении двух единиц образуется единица переноса в соседний старший разряд, т. е. $1+1 = 10_{(2)}$.

Пример 6. Выполним сложение двух положительных двоичных чисел, представленных в естественной форме (рисунок 2.6).

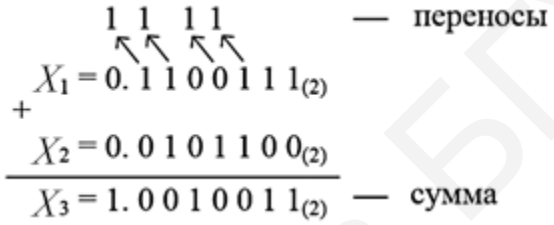


Рисунок 2.6 – Сложение двух положительных двоичных чисел в естественной форме

Из примера следует, что, начиная с четвертого разряда, сложение производится с учетом переноса единицы из соседнего младшего разряда.

Вычитание двоичных чисел выполняется в соответствии с таблицей 2.2.

Таблица 2.2 – Вычитание двоичных чисел

Заем	Уменьшаемое	Вычитаемое	Разность
0	0	0	0
1	0	1	1
0	1	0	1
0	1	1	0

Из таблицы 2.2 следует, что при вычитании единицы из нуля производится заем из соседнего старшего разряда, т. е. $10 – 1 = 1_{(2)}$.

Пример 7. Выполним вычитание двух двоичных чисел, представленных в естественной форме (рисунок 2.7).

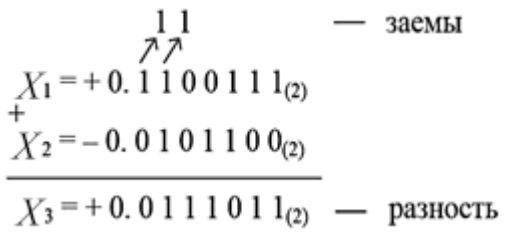


Рисунок 2.7 – Вычитание двух двоичных чисел в естественной форме

Так как $|X_1| > |X_2|$, то выполняется вычитание X_2 из X_1 и результату присваивается знак плюс.

Из примера следует, что при вычитании происходит заем из соседнего старшего разряда. Следовательно, при сложении и вычитании двоичных чисел в естественной форме используются различные правила, поэтому вычитание заменяется сложением в машинных кодах.

При алгебраическом сложении двоичных чисел в обратном и дополнительном кодах используются следующие правила:

1) знаковые разряды суммируются так же, как и значащие разряды числа;
2) единица переноса из знакового разряда при сложении в дополнительном коде отбрасывается, а в обратном коде для получения правильного результата эта единица прибавляется к младшему разряду суммы. Эта операция называется циклическим переносом;
3) если в знаковом разряде суммы получился нуль, то сумма положительна и для перехода к естественной форме числа для оценки результата обратное преобразование не требуется. Если же в знаковом разряде суммы получилась единица, то сумма отрицательна и требуется обратное преобразование.

Пример 8. Выполним суммирование двоичных чисел $X_1 = +0.1100111_{(2)}$ и $X_2 = -0.0101100_{(2)}$ в дополнительном коде (рисунок 2.8).

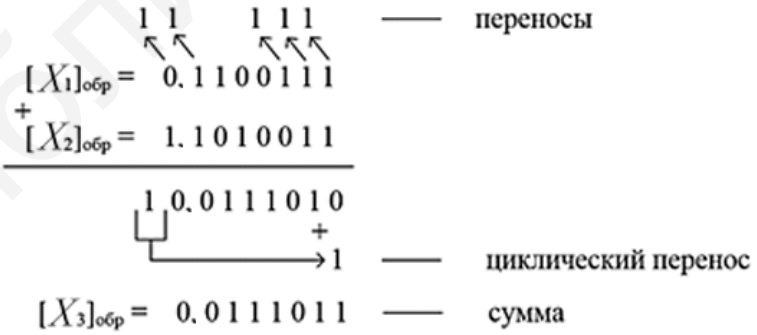


Рисунок 2.8 – Суммирование двоичных чисел в дополнительном коде

Результат получился положительным, т. е. $X_3 = +0.0111011_{(2)}$.

Пример 9. Выполним суммирование тех же двоичных чисел в дополнительном коде (рисунок 2.9): $X_1 = +0.1100111_{(2)}$ и $X_2 = -0.0101100_{(2)}$.

Рисунок 2.9 – Суммирование двоичных чисел в дополнительном коде

Результат получился положительным и совпал с результатами суммирования в обратном коде и вычитания в естественной форме.

При сложении чисел одного знака может произойти переполнение разрядной сетки, что приведет к искажению результата. Для выявления переполнения разрядной сетки применяют модифицированные машинные коды, в которых два знаковых разряда. Для положительных чисел в знаковых разрядах записывается 00, для отрицательных – 11. Появление комбинаций 01 или 10 в знаковых разрядах свидетельствует о переполнении разрядной сетки, т. е. является сигналом ошибки. Например, двоичное число $X = -0.110101$ в модифицированном дополнительном коде будет записано следующим образом:

$$[X]^{M}_{\text{доп}} = 11.001011. \qquad (2.3)$$

Пример 10. Выполним суммирование в дополнительном модифицированном коде двух двоичных чисел $X_1 = +0.1100111$ и $X_2 = -0.0101100$ (рисунок 2.10). Результат получился отрицательным, и потребовалось обратное преобразование.

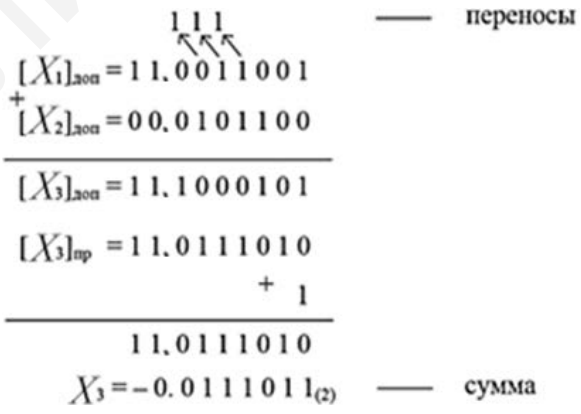


Рисунок 2.10 – Суммирование в дополнительном модифицированном коде двоичных чисел

Умножение. В цифровых устройствах умножение двоичных чисел производится по тем же правилам, что умножение десятичных чисел, и делится на два этапа.

Первый этап. Определяется знак произведения путем суммирования по модулю 2 знаковых разрядов сомножителей. Если знаковые разряды сомножителей одинаковы и равны 0 или 1, то сумма по модулю 2 равна 0 (произведение положительно); если же знаковые разряды сомножителей не совпадают, то эта сумма будет равна 1 (произведение отрицательно).

Второй этап. Определяется абсолютное значение произведения путем перемножения этих чисел без учета знака (кодовое умножение). Для этого необходимо вычислить частичные произведения, затем сдвинуть их и сложить между собой.

Суммирование частичных произведений осуществляется в несколько этапов, поскольку арифметическое устройство может выполнять операцию сложения сразу только с двумя числами.

Если умножение выполняется начиная со старшего разряда множителя, то частичные произведения следует сдвигать вправо.

Сложение двоично-десятичных чисел со знаком. Способ суммирования двоично-десятичных чисел зависит от того, какой двоичный код выбран для представления десятичных цифр. Ниже рассматривается операция суммирования при использовании кода 8421.

При сложении двоично-десятичных чисел выполняются дополнительные действия:

- анализ межтетрадных переносов при сложении;
- анализ диапазона двоичных чисел в каждой тетраде начальной суммы;
- введение поправок в некоторые тетрады по результатам анализа.

При алгебраическом сложении двоично-десятичных чисел необходимо рассмотреть два случая:

- операнды имеют одинаковые знаки;
- операнды имеют разные знаки.

Случай 1. Сложение в каждой тетраде должно выполняться по модулю 10. Если в результате сложения в тетраде возникает начальное число $T_{i\,\text{нач}} \geq 10$, то должен быть перенос в соседнюю старшую тетраду, а в данной тетраде должна формироваться разность $T_i = T_{i\,\text{нач}} - 10$. На самом деле в каждой тетраде выполняется сложение по модулю 16, т. е. перенос в соседнюю старшую тетраду формируется в том случае, если $T_{i\,\text{нач}} \geq 16$, а в данной тетраде получается разность $T_i = T_{i\,\text{нач}} - 10$.

При этом в зависимости от величины $T_{i\,\text{нач}}$ возможны три подслучая.

Подслучай 1. $0 \leq T_{i\,\text{нач}} \leq 9$, когда нет ни десятичного, ни шестнадцатеричного переноса, в результате чего в данной тетраде формируется правильная десятичная сумма, вычисляется по формуле

$$T_i = T_{i\,\text{нач}}. \qquad (2.4)$$

Пример 11. Выполним суммирование двух десятичных цифр 6 и 2 и единицы переноса, поступающей из предыдущего десятичного разряда, в десятичной системе счисления и коде 8421 (рисунок 2.11).

	Десятичная СС	Код 8421
Переносы	1⟵	1 1 1⟵
Первая цифра	6	0110
+		+
Вторая цифра	2	0010
Сумма	9	1001
Коррекция		—
Результат		1001

Рисунок 2.11 – Суммирование двух десятичных цифр и единицы переноса

Полученное в результате суммирования число $1001_{(2)}$ меньше десяти, и коррекция суммы не требуется.

Подслучай 2. $10 \le T_{i\,\text{нач}} \le 15$, когда имеет место нереализованный десятичный перенос, но нет еще шестнадцатеричного переноса, в результате чего в данной тетраде формируется десятичная сумма с избытком 10:

$$T_{i\,\text{нач}} = T_i + 10. \tag{2.5}$$

Следовательно, во втором подслучае требуется вычитание 10 единиц. Поскольку в цифровых устройствах вычитание заменяется сложением в дополнительном коде, то вычитание 10 единиц заменяется прибавлением шести единиц, так как число $6_{(10)} = 0110_{(2)}$ является дополнительным кодом отрицательного числа $10_{(10)} = 1010_{(2)}$, следовательно, получим выражение

$$[-1010]_{\text{доп}} = 0101 + 0001 = 0110_{(2)}. \tag{2.6}$$

Пример 12. Выполним суммирование двух десятичных цифр 8 и 5 в десятичной системе счисления и коде 8421 (рисунок 2.12).

Полученное в результате суммирования число $1101_{(2)}$ больше десяти, и требуется коррекция суммы, т. е. вычитание 10 единиц. В результате коррекции возникает перенос в соседнюю старшую тетраду, что уменьшает сумму не на 10, а на 16 единиц. Уход из суммы шести лишних единиц компенсируется прибавлением шести единиц в процессе коррекции.

Подслучай 3. $T_{i\,\text{нач}} \geq 15$, когда имеет место и нереализованный десятичный перенос, и реализованный шестнадцатеричный перенос, в результате которых в данной тетраде формируется десятичная сумма с избытком минус 6:

$$T_{i\,\text{нач}} = T_i + 10 - 16 = T_i - 6. \quad (2.7)$$

Следовательно, для коррекции суммы требуется прибавление шести единиц.

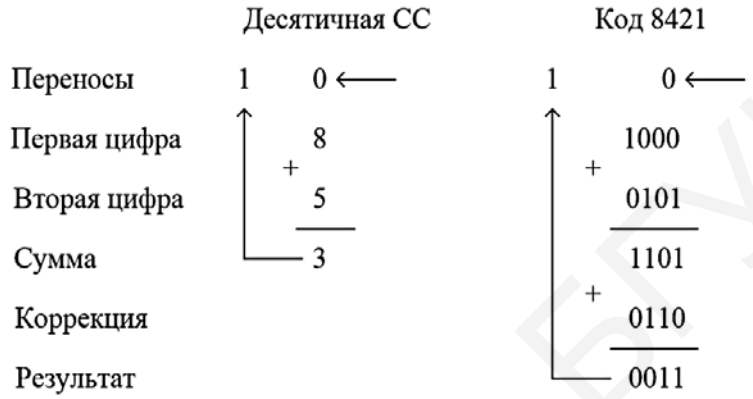


Рисунок 2.12 – Суммирование двух десятичных цифр в десятичной системе счисления и коде

Правило сложения десятичных чисел с одинаковыми знаками в коде 8421: к каждой тетраде любого из операндов, в которой возникло переполнение, необходимо прибавить поправку

$$\Delta_1 = 6_{(10)} = 0110_{(2)}. \quad (2.8)$$

Выполнить сложение операндов по правилам двоичной арифметики.

Случай 2. Если один из операндов является отрицательным, то перед сложением он преобразуется из прямого кода в обратный (или дополнительный). Наиболее просто получается обратный код путем инвертирования всех двоичных цифр в числе.

В результате в каждой тетраде будет получена взаимно обратная десятичная цифра (дополнение исходной цифры до 9) с избытком 6:

$$T_i = 15 - T_{i\,\text{нач}} = (9 - T_{i\,\text{нач}}) + 6. \quad (2.9)$$

После выполнения операции сложения и второй коррекции получится сумма, представленная в обратном коде 8421. Если сумма положительная, то операция на этом заканчивается. Если же сумма отрицательная, то необходимо преобразовать ее из обратного кода в прямой. Обычно инвертирование двоичных

цифр отрицательной суммы выполняется до второй коррекции и поправка $\Delta_2 = 10_{(10)} = 1010_{(2)}$ прибавляется к тем тетрадам, в которых во время сложения сформировался шестнадцатеричный перенос.

2.3 Порядок выполнения практической работы

Задание 1. Воспользовавшись обратными и дополнительными кодами, вычислить $S = A + B$. Значения A и B выбрать согласно варианту по таблице 2.3.

Таблица 2.3 – Исходные данные к заданию 1 подраздела 2.3

Номер варианта	Значение	
	A	***B***
1	-1010	1000
2	0,11010	$-0,10001$
3	$-0,11010$	0,10001
4	0,1010	$-0,1101$
5	1011010	1001001
6	101101	$-1,01010$
7	-1010111	1101011
8	10001	$-1,01110$
9	1,1100	11100
10	-01011	011110

Задание 2. Воспользовавшись обратными и дополнительными кодами, вычислить $R = A - B$. Значения A и B выбрать согласно варианту по таблице 2.4.

Таблица 2.4 – Исходные данные к заданию 2 подраздела 2.3

Номер варианта	Значение	
	A	***B***
1	0,11101	0,01011
2	$-0,0101$	0,1010
3	$-0,1001$	0,0111
4	1101010	1010101
5	1010101	1101010
6	1010111	1111110
7	$-0,01011$	0,111101
8	0,00111	101110
9	001100	0,010101
10	01011110	0011010

Проверить результаты вычисления в десятичной системе счисления.

Задание 3. Представить заданные числа в форме с плавающей запятой и вычислить сумму $S = A + B$, воспользовавшись дополнительными кодами. Значения A и B выбрать согласно варианту по таблице 2.5.

Таблица 2.5 – Исходные данные к заданию 3 подраздела 2.3

Номер варианта	Значение	
	A	***B***
1	0,11101	0,0101
2	−0,0101	0,101
3	−0,1001	−0,0111
4	1101010	1010101
5	0,11101	0,01011
6	−0,0101	0,1010
7	−0,1001	0,0111
8	1101010	1010101
9	0,11101	0,01011
10	−0,0101	0,1010

Проверить результаты вычисления в десятичной системе счисления.

Задание 4. Перевести нижеуказанные числа A и B в двоично-десятичный код 8421, сложить в обратном и дополнительном кодах, проверить результаты вычисления в десятичной системе счисления. Значения A и B выбрать согласно варианту по таблице 2.6.

Таблица 2.6 – Исходные данные к заданию 4 подраздела 2.3

Номер варианта	Значение	
	A	***B***
1	356	241
2	842	−456
3	123	−753
4	0,597	0,346
5	−0,2153	−0,1879
6	−67	−94
7	−0,1245	−1289
8	148	527
9	255	1023
10	0,1486	0,2044

Задание 5. Перевести нижеуказанные числа A и B в двоично-десятичный код 8421, вычислить разность $R = A - B$, воспользовавшись дополнительными кодами, проверить результаты вычисления в десятичной системе счисления. Значения A и B выбрать согласно варианту по таблице 2.7.

Таблица 2.7 – Исходные данные к заданию 5 подраздела 2.3

Номер варианта	Значение	
	A	***B***
1	74	24
2	−82	−56
3	123	−753
4	0,597	0,346
5	−0,2153	−0,7879
6	67	94
7	55	44
8	0,1299	0,1548
9	5007	1024
10	112	103

2.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1–5 согласно варианту.
3. Вывод.
4. Ответы на контрольные вопросы.

2.5 Контрольные вопросы

1. С какой целью используется кодирование двоичных чисел в цифровых устройствах?
2. В чем заключаются особенности дополнительного кода?
3. Что означает переполнение разрядной сетки устройства?
4. В чем заключаются особенности модифицированных кодов?
5. Перечислите признаки, согласно которым выполняется коррекция при сложении двоично-десятичных чисел.

3 Практическая работа №3

ТОЖДЕСТВЕННЫЕ ПРЕОБРАЗОВАНИЯ И МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ

3.1 Цель работы

Изучение тождественных преобразований и способов минимизации логических функций.

3.2 Теоретические сведения

Значение функций двух переменных в общей теории логических функций состоит в том, что с их помощью может быть представлена любая сколько угодно сложная ФАЛ. Средством для такого представления является суперпозиция булевых функций или подстановка одних логических функций вместо аргументов в другие функции. Возможность такой подстановки обуславливается тем, что области значений функций и их аргументов совпадают.

Для выражения сложных логических функций достаточно использовать не все элементарные функции, а только их некоторую часть, называемую базисом или системой.

Система элементарных функций $F_1, F_2, \dots, F_k$ называется функционально полной, если любую функцию алгебры логики можно записать в виде формулы через функции $F_1, F_2, \dots, F_k$.

Минимальным базисом называется такая функционально полная система $F_1, F_2, \dots, F_m$, для которой удаление любой одной из входящих в нее функций превращает эту систему в функционально неполную.

ФАЛ можно изменять, упрощать. Для проведения таких манипуляций используются основные законы алгебры логики, правила, теоремы упрощения, которые представлены в таблице 3.1.

Таблица 3.1 – Законы алгебры логики

Для оператора (+)	**Для оператора $(\bullet)$**
1	**2**
Переместительный закон (коммутативный)	
$x_1 + x_2 = x_2 + x_1$	$x_1 \cdot x_2 = x_2 \cdot x_1$
Сочетательный закон (ассоциативный)	
$(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$	$(x_1 \cdot x_2)x_3 = x_1(x_2 \cdot x_3)$
Распределительный закон (дистрибутивный)	
$x_1(x_2 + x_3) = x_1x_2 + x_1x_3$	$x_1 + x_2 \cdot x_3 = (x_1 + x_2)(x_1 + x_3)$

1	2
Закон отрицания (закон де Моргана)	
$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$	$\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$
Операции с логической 1 и 0	
$x + 0 = x$	$x \cdot 0 = 0$
$x + 1 = 1$	$x \cdot 1 = x$
Правило повторения (идемпотентности)	
$x + x + \cdots + x = x$	$x \cdot x \cdot \ldots \cdot x = x$
Правило дополнительности	
$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$
Правило двойного отрицания	
$\bar{\bar{x}} = x$	
Теоремы упрощения	
$x_1x_2 + x_1\overline{x_2} = x_1$	$(x_1 + x_2)(x_1 + \overline{x_2}) = x_1$
$x_1(x_1 + x_2) = x_1$	$x_1 + x_1x_2 = x_1$
$x_1(\overline{x_1} + x_2) = x_1x_2$	$x_1 + \overline{x_1}x_2 = x_1 + x_2$

Как известно, основные параметры логической схемы, например количество необходимого оборудования (а значит, стоимость) и быстродействие, можно определить по виду логической функции до построения схемы. Это приводит к необходимости оптимизации функции, т. е. к необходимости получения ее оптимального вида по выбранному критерию. В общем случае речь должна идти об оптимизации функции по таким критериям, как быстродействие, надежность (достижение их максимума), количество необходимого оборудования, габариты, энергопотребление, стоимость (достижение их минимума) и т. д. Указанные критерии противоречивы.

Например, увеличение быстродействия, как правило, достигается за счет организации параллельной работы данного устройства, но это ведет к увеличению оборудования, а значит, к уменьшению надежности и увеличению стоимости. Поэтому на практике обычно решается частная задача оптимизации по одному из критериев. Чаще всего это делается по минимуму необходимого оборудования, так как при этом, как правило, автоматически решаются задачи получения минимальных габаритов, массы, энергопотребления, стоимости.

Такая частная задача оптимизации логической функции носит название минимизации, а форма функции, полученная в результате ее решения, называется минимальной (МДНФ, или МКНФ).

Существуют различные методы минимизации, но наиболее широко используются три:

1) расчетный (метод непосредственных преобразований);

2) расчетно-табличный (метод Квайна – Мак-Класки);

3) табличный (метод Вейча – Карно).

Исходной формой логической функции для любого из этих методов является СДНФ, или СКНФ, и вычисляется по формуле

$$f_{\text{СДНФ}} = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3. \quad (3.1)$$

Разобьем метод минимизации на этапы.

Первый этап. Выполняются все возможные склеивания членов заданной функции. Эта процедура осуществляется за несколько шагов и в результате каждого из них происходит понижение ранга склеиваемых членов на единицу.

Шаг 1. Склеивание первого и третьего минтермов:

$$\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \text{ и } \bar{x}_1 \cdot x_2 \cdot x_3. \quad (3.2)$$

Результат склеивания:

$$\bar{x}_1 \cdot x_3(\bar{x}_2 \vee x_2) = \bar{x}_1 \cdot x_3. \quad (3.3)$$

Шаг 2. Склеивание второго и третьего минтермов:

$$\bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \text{ и } \bar{x}_1 \cdot x_2 \cdot x_3. \quad (3.4)$$

Результат склеивания:

$$\bar{x}_1 \cdot x_2(\bar{x}_3 \vee x_3) = \bar{x}_1 \cdot x_2. \quad (3.5)$$

Шаг 3. Склеивание первого и четвертого минтермов:

$$\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \text{ и } x_1 \cdot \bar{x}_2 \cdot x_3. \quad (3.6)$$

Результат склеивания:

$$\bar{x}_2 \cdot x_3(\bar{x}_1 \vee x_1) = \bar{x}_2 \cdot x_3. \quad (3.7)$$

Результатом склеивания всех минтермов является сокращенная формула:

$$f_{\text{ДНФ}} = \bar{x}_1 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \vee \bar{x}_2 \cdot x_3. \quad (3.8)$$

Затем выполняются испытания на склеивание всех членов функции в промежуточной форме $f_{\text{ДНФ}}$. Из соотношения $f_{\text{ДНФ}}$ видно, что все его члены изолированы друг от друга. Следовательно, полученная промежуточная форма является сокращенной ДНФ исходной СДНФ.

Необходимо убедиться, что все конституенты исходной СДНФ участвовали хотя бы в одном склеивании и в сокращенной форме максимального ранга не будет, воспользовавшись формулой

$$f_{\text{СДНФ}} = \bar{x}_1 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \vee \bar{x}_2 \cdot x_3. \quad (3.9)$$

Второй этап. Выполняется проверка каждой простой импликанты в СДНФ в несколько шагов, чтобы выявить и удалить лишние члены.

На значение истинности функции влияет только импликанта, которая сама равна 1.

Любая импликанта становится равной 1 лишь на одном определенном наборе значений истинности своих аргументов.

Однако если на этом наборе сумма остальных членов тоже обращается в 1, то рассматриваемая импликанта не влияет на значение истинности функции даже в этом единственном случае, т. е. является лишней.

Шаг 1. Выражение:

$$\bar{x}_1 \cdot x_3 = 1, \quad (3.10)$$

где $x_1 = 0$;
$x_3 = 1$.

Сумма остальных членов на этом же наборе:

$$\bar{x}_2 \cdot 1 \vee 1 \cdot x_2 = 1. \quad (3.11)$$

Следовательно, проверяемый член лишний.

Шаг 2. Выражение:

$$\bar{x}_2 \cdot x_3 = 1, \quad (3.12)$$

где $x_2 = 0$;
$x_3 = 1$.

Сумма остальных членов на этом же наборе:

$$\bar{x}_1 \cdot 1 \vee \bar{x}_1 \cdot 0 = \bar{x}_1. \quad (3.13)$$

Следовательно, проверяемый член не лишний.

Шаг 3. Выражение:

$$\bar{x}_1 \cdot x_2 = 1, \quad (3.14)$$

где $x_1 = 0$;
$x_2 = 1$.

Из (3.14) следует выражение

$$1 \cdot x_3 \vee 0 \cdot x_3 = x_3, \quad (3.15)$$

значит, проверяемый член не является лишним.

Таким образом, отбросив лишний член, получим тупиковую ДНФ исходной функции по формуле

$$f_{\text{ТНДФ}} = \bar{x}_1 \cdot x_2 \vee \bar{x}_2 \cdot x_3. \qquad (3.16)$$

Если лишних членов оказывается больше, например два, то оба отбросить нельзя, так как каждый из них проверялся при вхождении другого в оставшуюся сумму. Следовательно, можно отбросить только один из них, а затем снова произвести проверку возможности отбрасывания второго члена.

Третий этап. На этом этапе происходит упрощение ТДНФ.

Каждому входу любого логического элемента в некотором приближении соответствует один составной КМОП-транзистор (в схемах конъюнктора и дизъюнктора присутствует дополнительный (входной КМОП-транзистор, осуществляющий инверсию, но во многих случаях он может и отсутствовать).

Следовательно, любую логическую схему можно оценить по количеству необходимого для ее реализации оборудования, подсчитав количество входов всех элементов, составляющих эту схему (суммарное число входов всех элементов). Это оценка по Квайну, которая является приближенной, так как логические элементы могут строиться на различных принципах и по различным схемам (на многоэмиттерных транзисторах, по схеме токовых переключателей и т. д.). Такая оценка является очень удобной, так как объективна и доступна на самых ранних стадиях логического проектирования, а главное – достаточно точна при сравнительных (а не абсолютных) оценках различных вариантов одного и того же устройства.

Применим к $f_{\text{ТНДФ}}$ распределительный закон второго рода:

$$f = \bar{x}_1 \cdot x_2 \vee \bar{x}_2 \cdot x_3 = (\bar{x}_1 \vee \bar{x}_2) \cdot (\bar{x}_1 \vee x_3) \cdot (x_2 \vee \bar{x}_2) \cdot (x_2 \vee x_3). \qquad (3.17)$$

Распределительный закон второго рода: сумма некоторой логической переменной и произведения нескольких аргументов равна произведению сумм каждого сомножителя и этой переменной:

$$x_2 \cdot x_3 \vee x_1 = (x_3 \vee x_1) \cdot (x_1 \vee x_3) \cdot x_2 \vee \bar{x}_2 = 1. \qquad (3.18)$$

Из выражения (3.18) следует

$$f = (\bar{x}_1 \vee \bar{x}_2) \cdot (\bar{x}_1 \vee x_3) \cdot (x_2 \vee x_3). \qquad (3.19)$$

Применим распределительный закон 1-го рода.

Произведение суммы нескольких аргументов на какую-либо (логическую) переменную равно сумме произведений каждого слагаемого на эту переменную:

$$(x_2 \vee x_3) \cdot x_1 = x_1 \cdot x_2 \vee x_3 \cdot x_1 \cdot x_2 \cdot x_3 \vee x_1 = (x_1 \vee x_2) \cdot (x_1 \vee x_3). \qquad (3.20)$$

Из выражения (3.20) следует

$$f = (\bar{x}_1 \vee \bar{x}_2) \cdot (\bar{x}_1 \vee x_3) \cdot (x_2 \vee x_3) = (\bar{x}_1 \vee x_2 \cdot x_3) \cdot (x_2 \vee x_3) =$$
$$= \bar{x}_1 \cdot x_2 \vee \bar{x}_1 \cdot x_3 \vee \bar{x}_2 \cdot x_3 \cdot x_2 \vee \bar{x}_2 \cdot x_3 \cdot x_3 = \bar{x}_1 \cdot x_2 \vee \bar{x}_1 \cdot x_3 \vee \bar{x}_2 \cdot x_3. \quad (3.21)$$

Введем в состав функции нулевой член $x_2 \cdot \bar{x}_2 = 0$ и выполним преобразования по формуле

$$f = \bar{x}_1 \cdot x_2 \vee \bar{x}_1 \cdot x_3 \vee \bar{x}_2 \cdot x_3 \vee x_2 \cdot \bar{x}_2 = \bar{x}_1(x_2 \vee x_3) \vee$$
$$\vee \bar{x}_2(x_3 \vee x_2) = (x_2 \vee x_3) \cdot (\bar{x}_1 \vee \bar{x}_2). \quad (3.22)$$

Применим ко второму конъюнктивному члену правило Де Моргана и получим минимальную форму исходной функции:

$$f = (x_2 \vee x_3) \cdot \left(\overline{\bar{x}_1 \vee \bar{x}_2}\right) = (x_2 \vee x_3) \cdot \left(\bar{\bar{x}}_1 \cdot \bar{\bar{x}}_2\right) = (x_2 \vee x_3) \cdot (\overline{x_1 \cdot x_2}). \quad (3.23)$$

Для аппаратурной реализации потребуется всего семь условных транзисторов.

Расчетно-табличный метод минимизации. Отличается от расчетного только методикой выявления лишних членов в сокращенной ДНФ (КНФ). Этот метод был предложен американским ученым У. Квайном.

Первый и третий этапы будут идентичны соответствующим этапам при расчетном методе:

$$f_{\text{СДНФ}} = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3; \quad (3.24)$$

$$f_{\text{СДНФ}} = \bar{x}_1 \cdot x_3 \vee \bar{x}_1 \cdot x_2 \vee \bar{x}_2 \cdot x_3. \quad (3.25)$$

Второй этап. Для выявления возможных лишних членов в СДНФ строится таблица, входными величинами в которой будут конституенты – члены СДНФ и простые импликанты – члена СДНФ. Поэтому такую таблицу называют конституентно-импликантной матрицей, либо таблицей Квайна, либо таблицей покрытий. Число строк равно числу простых импликант в СДНФ.

Строки делятся на столбцы, число столбцов равно числу конституент в СДНФ (таблица 3.2).

Таблица 3.2 – Конституентно-импликантная матрица

Простые импликанты	**Конституенты**			
	$\bar{x}_1 \cdot \bar{x}_2 \cdot x_3$	$\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$	$\bar{x}_1 \cdot x_2 \cdot x_3$	$x_1 \cdot \bar{x}_2 \cdot x_3$
$\bar{x}_1 \cdot x_3$ } лишняя	×	–	×	×
$\left.\begin{matrix}\bar{x}_1 \cdot x_2 \\ \bar{x}_2 \cdot x_3\end{matrix}\right\}$ ядро	×	×	×	

Процесс минимизации начинается с последовательного сопоставления каждой простой импликанты со всеми конституентами.

Если какая-либо импликанта является собственной частью некоторой конституенты, то в табличной клетке, соответствующей обоим членам, проставляется любой условный значок.

Значки в каждой строке заполненной таблицы показывают, какие члены совершенной формы функции появляются в семействе конституент при развертывании данной простой импликанты, т. е. значки показывают, какие конституенты покрываются рассматриваемой импликантой.

В идеальном случае каждая импликанта покрывает только свою конституенту и в каждом столбце находится только один значок. Но на практике этого не происходит. Часто одна и та же конституента покрывается в таблице несколькими импликантами.

Необходимо вычеркиванием некоторых (лишних) импликант попытаться оставить в каждой колонке только один значок или минимальное их число.

Вначале по таблице находится ядро функции, состоящее из импликант, каждая из которых осуществляет единственное покрытие некоторой конституенты и поэтому никоим образом не может оказаться лишней.

В ядро входят импликанты $\bar{x}_1 \cdot x_2$ и $\bar{x}_2 \cdot x_3$. Вычеркивание $\bar{x}_1 \cdot x_3$ не нарушает условия о наличии хотя бы одного покрытия каждой конституенты.

Минимизация табличным методом с помощью карт Карно. Рассмотрим пример синтеза КЦУ в базисах И, ИЛИ, НЕ, функционирование которого задано таблицей истинности (таблица 3.3).

Заполним карту Карно значениями логической функции $f_{11}(x_1, x_2, x_3, x_4)$ (рисунок 3.1) и проведем минимизацию, в результате которой получим функцию МДНФ.

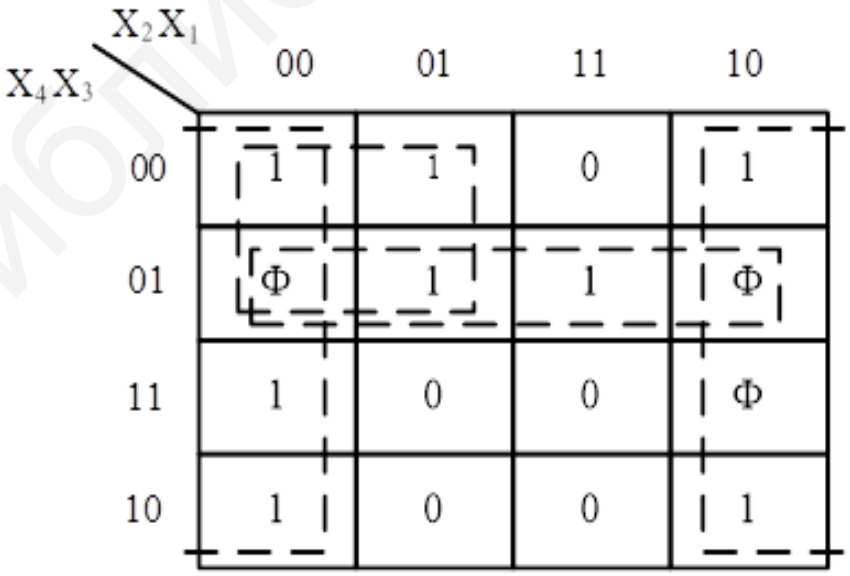


Рисунок 3.1 – Карта Карно для минимизации логической функции в МДНФ

Результат минимизации:

$$f_{11(\text{МДНФ})(x_1,x_2,x_3,x_4)} = \bar{x}_1 \vee \bar{x}_2 \cdot \bar{x}_4 \vee x_3 \cdot \bar{x}_4. \qquad (3.26)$$

Запишем результат минимизации в МКНФ, объединив в области соседние макстермы (рисунок 3.2).

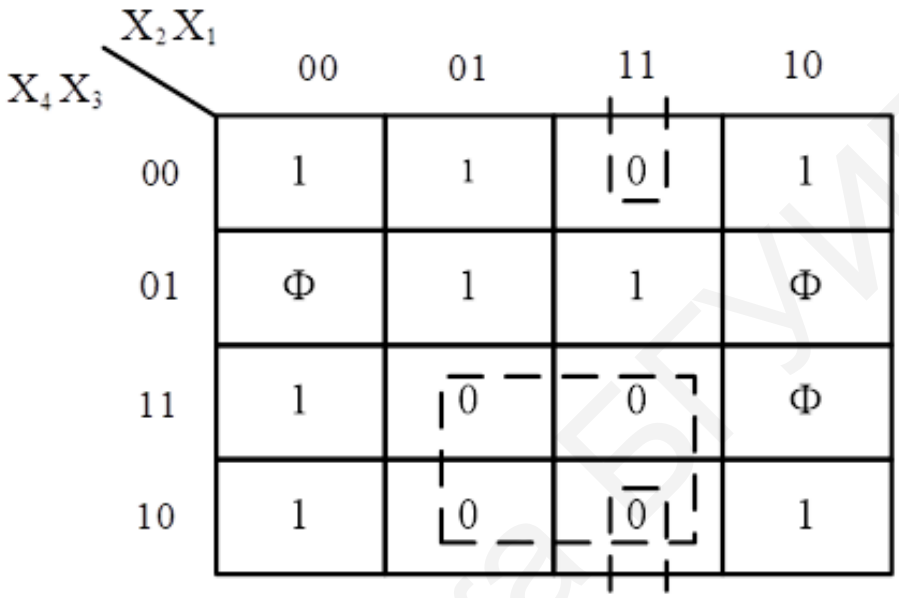


Рисунок 3.2 – Карта Карно для минимизации логической функции в МКНФ

Результат минимизации:

$$f_{11(\text{МКНФ})(x_1,x_2,x_3,x_4)} = (\bar{x}_1 \vee \bar{x}_4) \cdot (\bar{x}_1 \vee \bar{x}_2 \vee x_3). \qquad (3.27)$$

3.3 Порядок выполнения практической работы

Задание 1. Синтезировать КЦУ в базисе И, ИЛИ, НЕ согласно заданному варианту по таблице 3.3:

1) задать логическую функцию (ЛФ) с помощью карты Карно. Для МДНФ и МКНФ ЛФ использовать отдельные карты Карно;

2) минимизировать ЛФ табличным методом с помощью карты Карно. Результат минимизации записать в МДНФ и МКНФ;

3) проверить правильность функционирования логических схем устройства для пятого набора аргументов. Для этого проставить уровни сигналов на выводах всех логических элементов. Значения сигналов на выходах схем сравнить со значением ЛФ для пятого набора аргументов, они должны совпадать.

Таблица 3.3 – Исходные данные для синтеза КЦУ

Номер набора аргументов	Наборы аргументов				Номер варианта										
					1	2	3	4	5	6	7	8	9	10	11
	x_4	x_3	x_2	x_1	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
0	0	0	0	0	1	1	1	0	1	0	0	1	1	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1
2	0	0	1	0	1	1	Ф	0	Ф	Ф	Ф	1	1	0	1
3	0	0	1	1	Ф	0	1	Ф	0	1	0	Ф	Ф	1	0
4	0	1	0	0	0	1	0	0	1	0	0	1	1	1	Ф
5	0	1	0	1	0	1	1	1	Ф	0	1	1	0	1	1
6	0	1	1	0	1	0	0	0	1	Ф	1	Ф	Ф	0	Ф
7	0	1	1	1	0	0	Ф	1	Ф	0	Ф	1	1	1	1
8	1	0	0	0	1	Ф	0	0	Ф	1	0	0	1	Ф	1
9	1	0	0	1	Ф	1	1	0	0	1	1	1	1	0	0
10	1	0	1	0	1	1	Ф	Ф	0	Ф	Ф	0	Ф	0	1
11	1	0	1	1	1	Ф	1	1	1	1	0	1	1	1	0
12	1	1	0	0	1	1	0	Ф	Ф	1	Ф	0	0	Ф	1
13	1	1	0	1	1	1	1	1	0	1	1	1	0	0	0
14	1	1	1	0	1	0	0	1	0	0	0	Ф	Ф	Ф	Ф
15	1	1	1	1	0	1	1	Ф	Ф	1	1	0	0	1	0

Задание 2. Выполнить минимизацию логической функции согласно заданному варианту (см. таблицу 3.3):

1) расчетным методом;
2) методом Мак-Класки.

3.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1 и 2 согласно варианту.
3. Вывод.
4. Ответы на контрольные вопросы.

3.5 Контрольные вопросы

1. Перечислите этапы синтеза КЦУ.
2. Что такое минимизация логических функций?
3. Перечислите этапы минимизации.
4. Укажите достоинства и недостатки различных методов минимизации.
5. Что такое минтерм?
6. Что такое макстерм?

4 Практическая работа №4

СИНТЕЗ КОМБИНАЦИОННЫХ УСТРОЙСТВ

4.1 Цель работы

Приобретение практических навыков синтеза комбинационных цифровых устройств (КЦУ) в базисах И-НЕ, ИЛИ-НЕ, а также построения принципиальных электрических схем КЦУ на микросхемах стандартной логики схемотехники КМОП.

4.2 Теоретические сведения

Синтез КЦУ заключается в определении таких способов соединения простейших логических элементов, при которых построенное устройство реализует поставленную задачу по преобразованию двоичной информации.

Синтез КЦУ делится на пять этапов:

1) описание проектируемого узла в содержательных терминах (словесное описание);
2) формализация описания, т. е. составление таблицы истинности синтезируемого узла согласно его назначению и словесному описанию принципа работы;
3) переход от табличного описания синтезируемого узла к логико-математическому описанию в виде логической функции;
4) анализ полученной функции с целью ее упрощения (минимизация);
5) составление функциональной (логической) схемы узла в заданном базисе.

В некоторых случаях, если известна электрическая схема каждого элемента, составляется принципиальная схема узла. В данном случае анализ и синтез выступают как две взаимосвязанные части единого процесса – проектирования цифровых устройств. Анализ может быть самостоятельным процессом, если логическая схема устройства уже имеется и необходимо проанализировать ее работу с целью ремонта либо модернизации.

Построим логическую схему устройства по формуле

$$f_{11(\text{МДНФ})(x_1,x_2,x_3,x_4)} = \bar{x}_1 \vee \bar{x}_2 \cdot \bar{x}_4 \vee x_3 \cdot \bar{x}_4. \quad (4.1)$$

Для этого потребуется:

1) два логических элемента 2И;
2) один логический элемент 3ИЛИ;
3) три логических элемента НЕ.

Логическая схема устройства в базисе И, ИЛИ, НЕ, построенная по логической функции (4.1), представлена на рисунке 4.1.

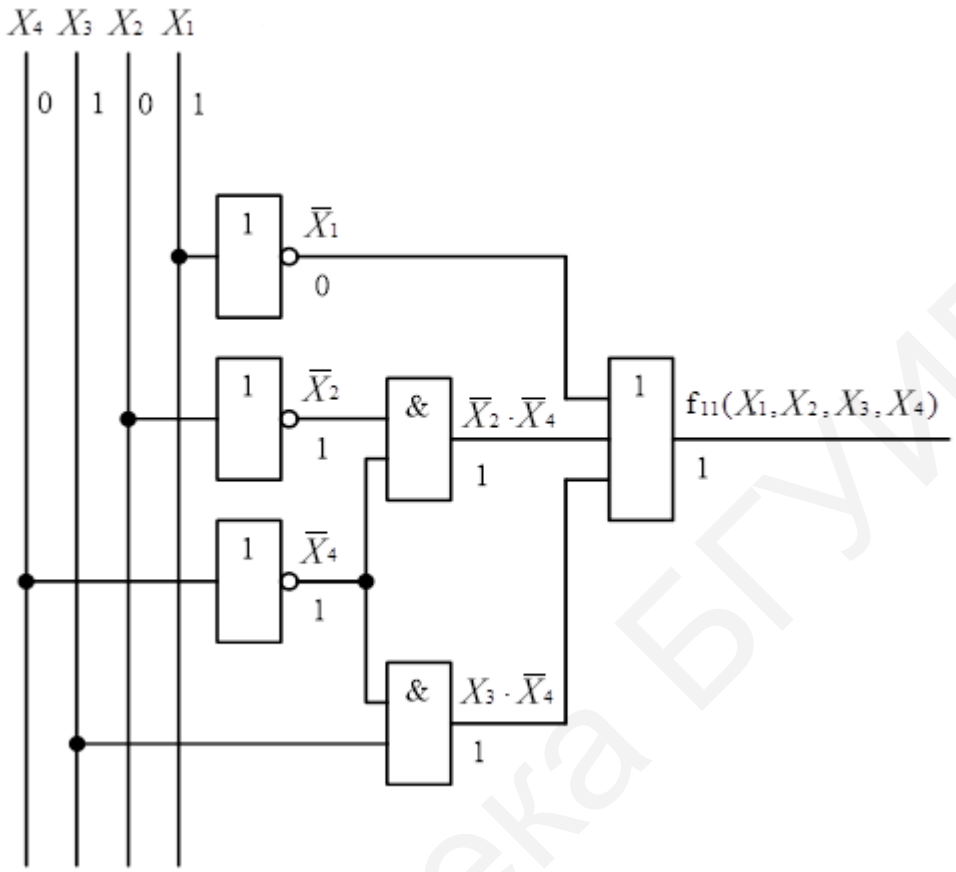


Рисунок 4.1 – Логическая схема устройства в базисе И, ИЛИ, НЕ в МДНФ

Построим логическую схему устройства по формуле

$$f_{11(\text{МКНФ})(x_1,x_2,x_3,x_4)} = (\bar{x}_1 \vee \bar{x}_4) \cdot (\bar{x}_1 \vee \bar{x}_2 \vee x_3). \qquad (4.2)$$

Для этого потребуется:

1) один логический элемент 2ИЛИ;
2) один логический элемент 3ИЛИ;
3) один логический элемент 2И;
4) три логических элемента НЕ.

Логическая схема устройства в базисе И, ИЛИ, НЕ, построенная по ЛФ (4.2), представлена на рисунке 4.2.

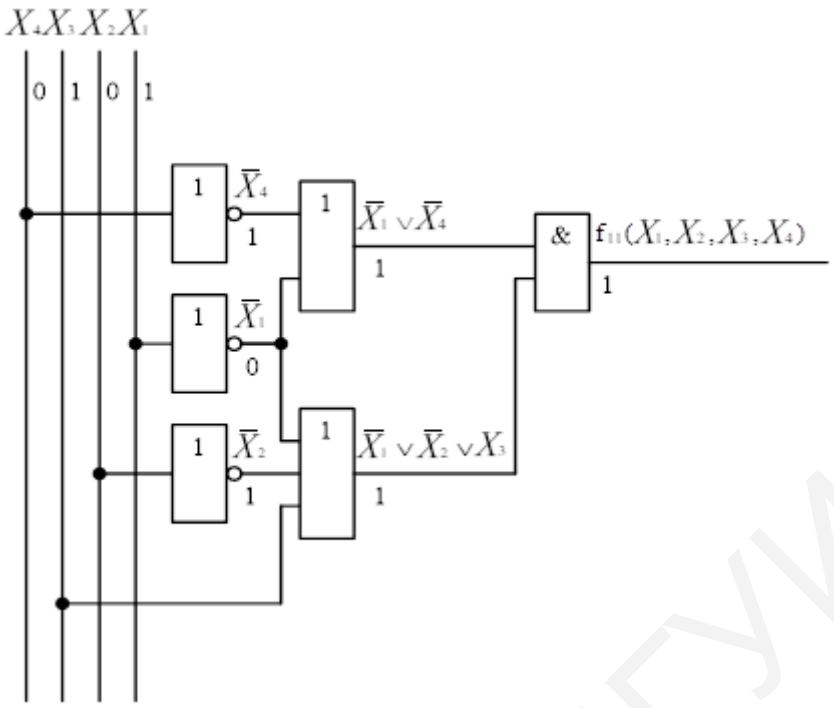


Рисунок 4.2 – Логическая схема устройства в базисе И, ИЛИ, НЕ в МКНФ

Уровни сигналов на выходах логических схем соответствуют значению ЛФ $f_{11}(x_1, x_2, x_3, x_4)$ для пятого набора аргументов (таблица 3.4), что позволяет судить о правильности их функционирования.

Сравним по сложности логические схемы КЦУ (см. рисунки 4.1 и 4.2), для этого подсчитаем общее число входов логических элементов. По этому критерию обе схемы одинаковы.

Рассмотрим пример синтеза КЦУ в базисе И-НЕ для той же ЛФ.

1. Преобразуем ЛФ (4.1) в базис И-НЕ, используя правило де Моргана, по формуле

$$f_{11(\text{МДНФ})(X_1X_1,X_2,X_3,X_4)} = \overline{X_1} \vee \overline{X_2} \cdot \overline{X_4} \vee X_3 \cdot \overline{X_4} =$$
$$= \overline{\overline{X_1} \vee \overline{X_2} \cdot \overline{X_4} \cdot X_3 \cdot \overline{X_4}} = \overline{\overline{\overline{X_1} \cdot \overline{X_2} \cdot \overline{X_4}} \cdot \overline{X_3 \cdot \overline{X_4}}} =$$
$$= X_1 | (\overline{X_2} | \overline{X_4}) | (X_3 | \overline{X_4}). \qquad (4.3)$$

2. Построим логическую схему устройства в базисе И-НЕ по формуле (4.3), при этом одиночные отрицания аргументов реализуем на основе ЛЭ 2И-НЕ с объединенными входами.

Для этого необходимо использовать:

- четыре логических элемента 2И-НЕ;
- один логический элемент 3И-НЕ.

Логическая схема устройства в базисе И-НЕ представлена на рисунке 4.3.

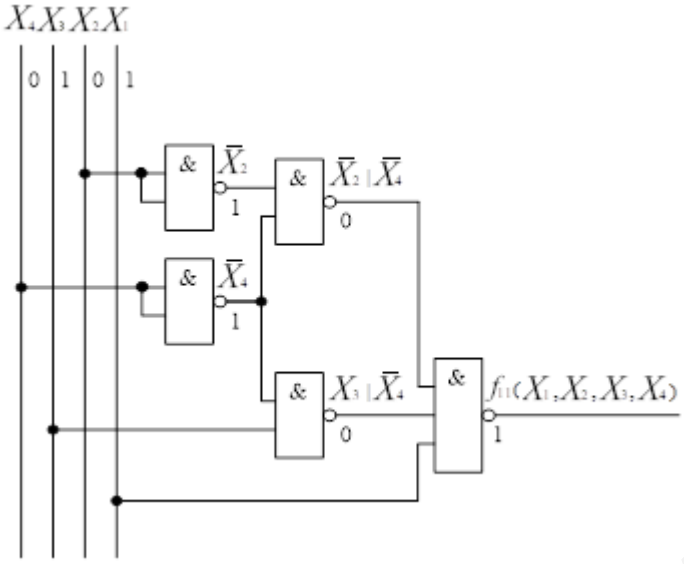


Рисунок 4.3 – Логическая схема устройства в базисе И-НЕ

3. Построим принципиальную электрическую схему устройства в базисе И-НЕ на микросхемах стандартной логики схемотехники КМОП серии 1554. Для этого из приложения А выбираем стандартные микросхемы интегральных ЛЭ серии 1554 и соединяем их между собой в соответствии с логической схемой (см. рисунок 4.3).

Выбираем микросхему ЭКР1554ЛА3 (*IN*74*AC*00*N*), в одном корпусе которой выполнено четыре ЛЭ 2 И-НЕ, а также ЭКР1554ЛА4 (*IN*74*AC*10*N*) – три ЛЭ 3И-НЕ (два останутся свободными). Принципиальная электрическая схема устройства в базисе И-НЕ представлена на рисунке 4.4.

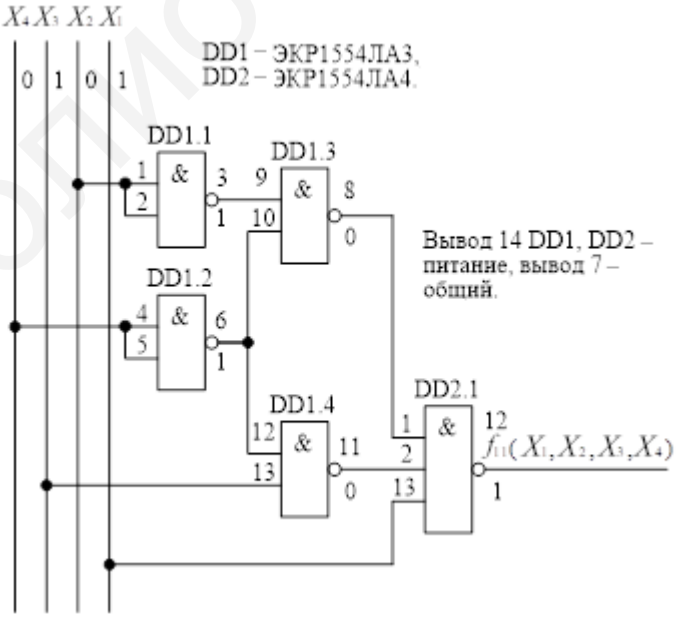


Рисунок 4.4 – Принципиальная электрическая схема устройства в базисе И-НЕ

4. Определим среднюю задержку распространения сигнала от входа к выходу устройства и среднюю потребляемую мощность.

В худшем случаем сигнал передается через три ЛЭ. Для расчета средней задержки возьмем данные из приложения А. В результате получаем суммарную среднюю задержку распространения сигнала от входа к выходу устройства:

$$t_{\text{зд.р.ср}} = 2 \cdot 7{,}75 + 7{,}25 = 22{,}75 \text{ нс}. \quad (4.4)$$

Для расчета средней потребляемой мощности всей схемы необходимо расчетным путем определить среднюю потребляемую мощность каждой микросхемы путем умножения среднего потребляемого тока (см. приложение А) на напряжение источника питания. В результате получаем суммарную среднюю потребляемую мощность:

$$P_{\text{пот.ср}} = (40 + 40) \cdot 5 = 400 \text{ мкВт}. \quad (4.5)$$

Рассмотрим пример синтеза КЦУ в базисе ИЛИ-НЕ для той же ЛФ.

1. Преобразуем ЛФ (4.2) в базис ИЛИ-НЕ, используя правило де Моргана, по формуле

$$f_{11(\text{МДНФ})(X_1,X_2,X_3,X_4)} = (\overline{X_1} \vee \overline{X_4}) \cdot (\overline{X_1} \vee \overline{X_2} \vee X_3) =$$
$$= \overline{\overline{(\overline{X_1} \vee \overline{X_4}) \cdot (\overline{X_1} \vee \overline{X_2} \vee X_3)}} = \overline{\overline{\overline{X_1} \vee \overline{X_4}} \vee \overline{\overline{X_1} \vee \overline{X_2} \vee X_3}} =$$
$$= (\overline{X_1} \downarrow \overline{X_4}) \downarrow (\overline{X_1} \downarrow \overline{X_2} \downarrow X_3). \quad (4.6)$$

2. Построим логическую схему устройства в базисе ИЛИ-НЕ по формуле (4.4), при этом одиночные отрицания аргументов реализуем на основе ЛЭ 2ИЛИ-НЕ с объединенными входами.

Для этого необходимо использовать:

- пять логических элементов 2ИЛИ-НЕ;
- один логический элемент 3ИЛИ-НЕ.

Логическая схема устройства в базисе ИЛИ-НЕ представлена на рисунке 4.5.

3. Построим принципиальную электрическую схему устройства в базисе ИЛИ-НЕ на микросхемах стандартной логики схемотехники КМОП серии 1564. Для этого из приложения А выбираем стандартные микросхемы интегральных ЛЭ серии 1564 и соединяем их между собой в соответствии с логической схемой (рисунок 4.5). Выбираем микросхему ЭКР1564ЛЕ1 (*IN*74*HC*02*AN*), в одном корпусе которой выполнено четыре ЛЭ 2ИЛИ-НЕ (необходимо использовать два корпуса этой микросхемы, причем три ЛЭ второго корпуса останутся свободными), а также ЭКР1564ЛЕ4 (*IN*74*HC*27*AN*) – три ЛЭ 3ИЛИ-НЕ (два ЛЭ останутся свободными).

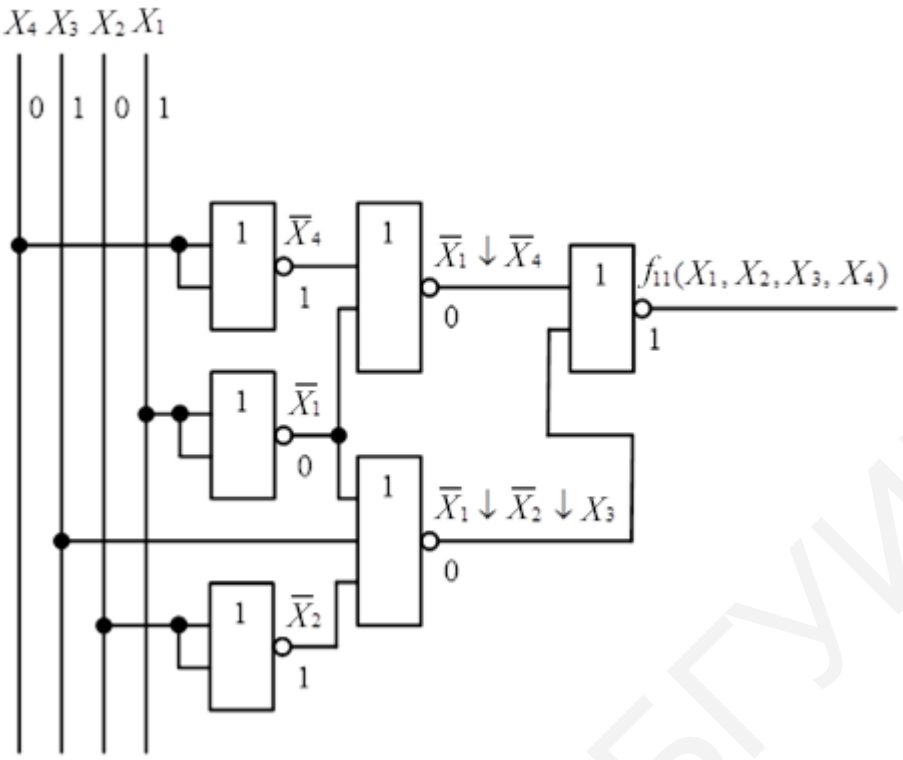


Рисунок 4.5 – Логическая схема устройства в базисе ИЛИ-НЕ

4. Определяем среднюю задержку распространения сигнала от входа к выходу устройства и среднюю потребляемую мощность.

В худшем случае сигнал передается через три ЛЭ. Расчет средней задержки выполним аналогично пункту 4 предыдущего примера. В результате получаем среднюю задержку

$$t_{\text{зд.р.ср}} = 2 \cdot 20 + 23 = 63 \text{ нс}. \quad (4.7)$$

Расчет средней потребляемой мощности выполним аналогично пункту 4 предыдущего примера. В результате получаем среднюю потребляемую мощность

$$P_{\text{пот.ср}} = (10 \cdot 2 + 20) \cdot 5 = 200 \text{ мкВт}. \quad (4.8)$$

Таким образом, устройство на микросхемах серии 1554 имеет более высокое быстродействие и, следовательно, большую среднюю потребляемую мощность.

Принципиальная электрическая схема устройства в базисе ИЛИ-НЕ представлена на рисунке 4.6.

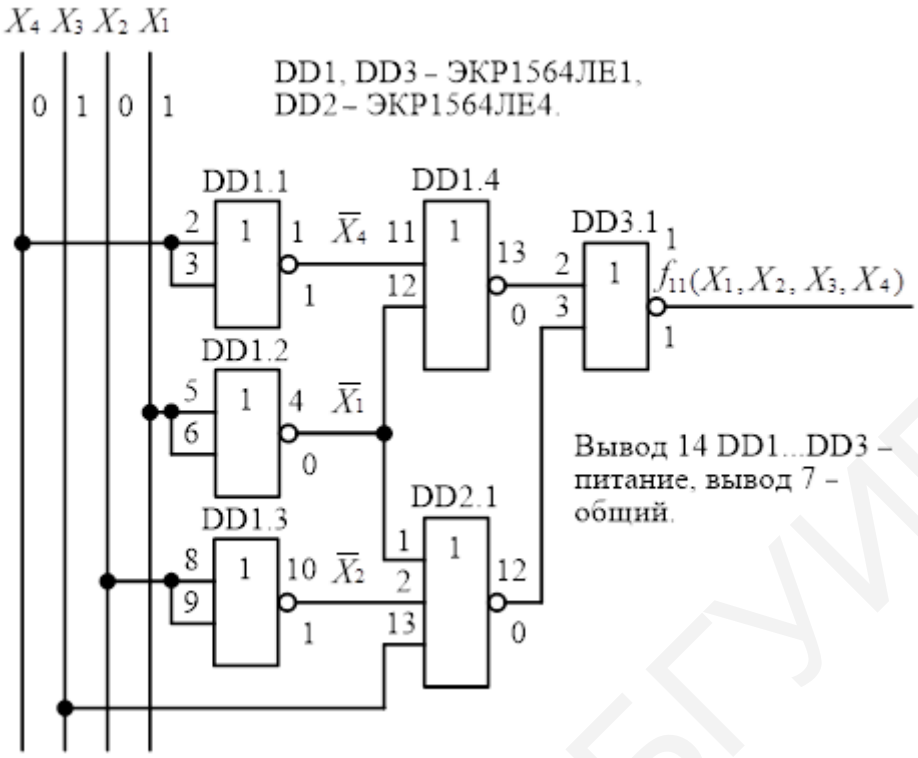


Рисунок 4.6 – Принципиальная электрическая схема устройства в базисе ИЛИ-НЕ

На схеме выполнены обозначения в соответствии с пунктом 3 предыдущего примера.

Проверим правильность функционирования логических и принципиальных схем КЦУ в базисах И-НЕ, ИЛИ-НЕ (см. рисунки 4.3–4.6). Для этого проставим на выходах всех логических элементов значения сигналов для пятого набора аргументов. Поскольку на выходах логических схем единичный уровень сигнала, то они функционируют в соответствии с таблицей истинности (см. значение логической функции f_{11} для пятого набора аргументов).

Сравним по сложности логические и принципиальные схемы КЦУ (см. рисунки 4.3–4.6), для этого подсчитаем общее число входов логических элементов. По этому критерию схемы в базисе И-НЕ построить проще.

4.3 Порядок выполнения практической работы

Задание 1. Синтезировать в базисе И-НЕ КЦУ согласно заданному варианту, функционирование которого задано таблицей истинности (см. таблицу 3.3). Для этого:

1) преобразовать МДНФ ЛФ в базис И-НЕ;
2) построить логическую схему КЦУ в базисе И-НЕ;
3) построить принципиальную электрическую схему КЦУ в базисе И-НЕ на микросхемах схемотехники КМОП серии 1554.

Задание 2. Синтезировать в базисе ИЛИ-НЕ КЦУ согласно заданному варианту, функционирование которого задано таблицей истинности (см. таблицу 3.3). Для этого:

1) преобразовать МКНФ ЛФ в базис ИЛИ-НЕ;

2) построить логическую схему КЦУ в базисе ИЛИ-НЕ на микросхемах схемотехники КМОП серии 1564;

3) построить принципиальную электрическую схему КЦУ в базисе ИЛИ-НЕ на микросхемах схемотехники КМОП;

4) проверить правильность функционирования логических и принципиальных схем КЦУ для пятого набора аргументов и сделать вывод о правильности их функционирования;

5) определить среднюю задержку распространения сигнала от входов к выходу в принципиальных электрических схемах устройств в базисах И-НЕ, ИЛИ-НЕ;

6) определить среднюю потребляемую мощность в принципиальных электрических схемах в базисах И-НЕ, ИЛИ-НЕ.

Задание 3. Построить две логические схемы устройства по МДНФ и МКНФ ЛФ. Сравнить логические схемы по сложности.

Задание 4. Построить схемы электрические принципиальные устройства для МДНФ и МКНФ. Для четного варианта использовать серию 1554, а для нечетного – серию 1564.

4.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1–4 согласно варианту.
3. Вывод.
4. Ответы на контрольные вопросы.

4.5 Контрольные вопросы

1. Что такое основной базис?
2. Что такое минимальный базис?
3. С какой целью осуществляется преобразование из одного базиса в другой?
4. Как рассчитать быстродействие синтезируемой КЦУ?
5. Приведите формулу расчета потребляемой мощности.

5 Практическая работа №5

ПРОГРАММИРОВАНИЕ И ОТЛАДКА МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ НА ЯЗЫКЕ АССЕМБЛЕРА

5.1 Цель работы

Исследование особенностей микропроцессора фирмы *Intel* модели 8086; приобретение практического навыка написания программы для работы микропроцессора.

5.2 Теоретические сведения

Микропроцессор (МП) – построенное на одном кристалле программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управление процессом обработки информации.

Intel 8086 (1810ВМ86) имеет 16-разрядную шину данных (рисунок 5.1).

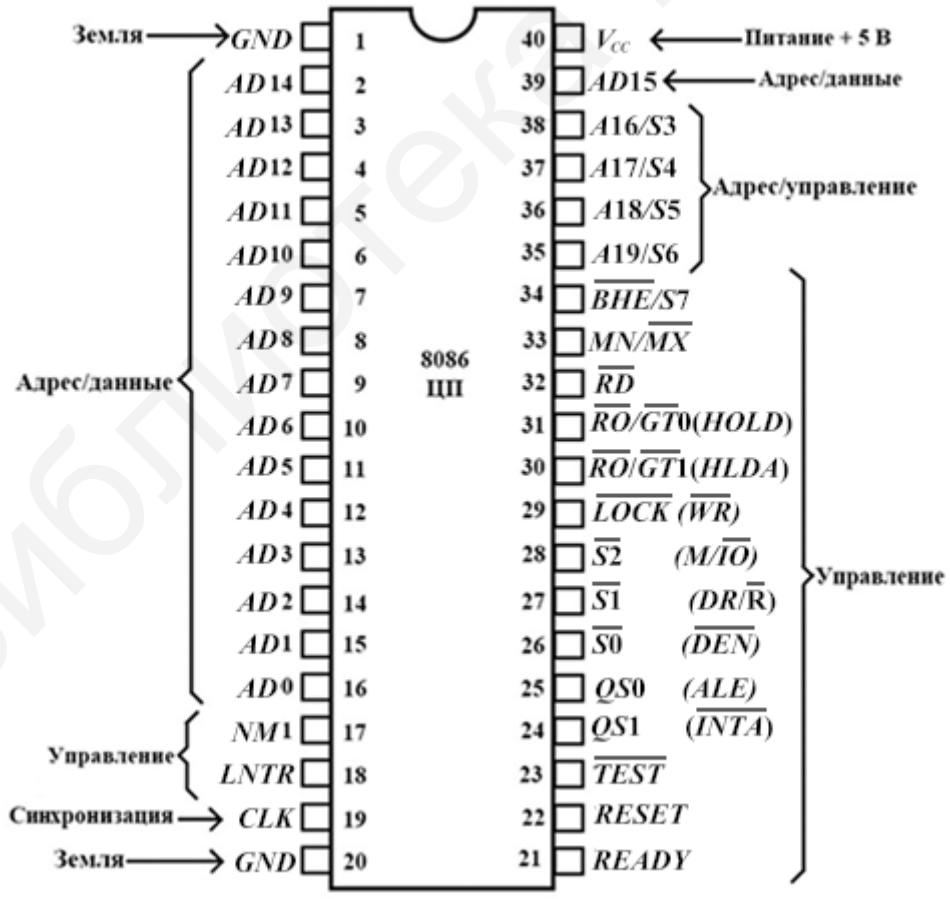


Рисунок 5.1 – Микропроцессор 8086 фирмы *Intel* (отечественный аналог 1810ВМ86)

В микропроцессоре *I*8086 использована примитивная форма конвейерной обработки. Блок интерфейса с шиной подает поток команд к исполнительному устройству через 6-байтовую очередь команд (таблица 5.1).

Таблица 5.1 – Назначение сигналов на выводах микропроцессора

Обозначение вывода	**Вход/выход**	**Назначение вывода**
*AD*15–*AD*0	Вход/выход	Мультиплексированная ША/Д
$A16/S_3$–$A19/S_6$	Выход	Линии адреса или состояния
$BHE\#/S_7$	Выход	Разрешение старшего байта шины
RD#	Выход	Управление чтением
WR#	Выход	Управление записью
M/IO#	Выход	Выбор памяти или ВУ
ALE	Выход	Разрешение фиксации адреса
DT/R#	Выход	Управление пересылкой данных
DEN#	Выход	Разрешение пересылки данных
MN/MX#	Вход	Установка режима
TEST#	Вход	Сигнал окончания режима ожидания
HOLD	Вход	Запрос захвата шины
HLDA	Выход	Подтверждение захвата
INTR	Вход	Запрос прерывания
NMI	Вход	Запрос немаскируемого прерывания
INTA#	Выход	Подтверждение прерывания
READY	Вход	Готовность памяти или ВУ
RESET	Вход	Сброс (начальная установка)
CLK	Вход	Такты ГТИ
GND, +5 В	Вход	Общий (земля), питание

Таким образом, выборка и выполнение новых команд могут происходить одновременно. Это значительно увеличивает пропускную способность процессора и при этом нет необходимости считывать команды из медленной памяти.

Рассмотрим строение процессора (рисунок 5.2).

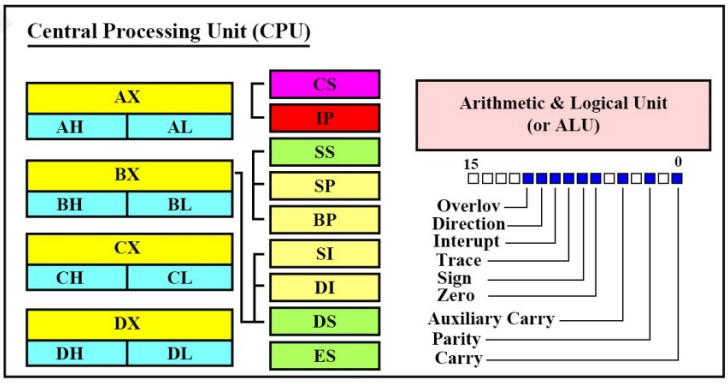


Рисунок 5.2 – Строение процессора

Процессор состоит из следующих частей:

- *AX* – регистр-аккумулятор (разделен на два регистра: *AH* и *AL*);
- *BX* – регистр базового адреса (разделяется на *BH*/*BL*);
- *CX* – регистр-счетчик (разделяется на *CH*/*CL*);
- *DX* – регистр данных (разделяется на *DH*/*DL*);
- *SI* – регистр-индекс источника;
- *BP* – указатель базы;
- *SP* – указатель стека.

Описание регистров:

- регистры *SI* (индекс источника) и *DI* (индекс приемника) используются в строковых операциях;
- регистры *DP* и *SP* задействуются при работе со стеком;
- регистр *CS* служит для хранения сегмента кода программы;
- регистр *DS* – для хранения сегмента данных;
- регистр *SS* – для хранения сегмента стека;
- регистр *ES* – дополнительный сегментный регистр, который может хранить адрес любого другого сегмента (например, видеобуфера).

Сегментные регистры необходимы для обращения к тому или иному сегменту памяти (например, видеобуферу).

Устройство управления микропроцессора осуществляет управление устройством с шиной, операционным устройством, периферийным оборудованием системы и обеспечивает обмен данными.

Устройство с шиной (УСШ). Организует опережающую выборку команд из памяти и формирует очередь выбранных байтов последовательности команд. В состав УСШ входят:

- 6-байтная очередь команд;
- счетчик команд;
- сегментные регистры;
- сумматор.

Операционное устройство (ОУ). Извлекает команды из очереди и реализует предписанные командами операции в 16-разрядном АЛУ. В состав ОУ входят:

- блок регистров общего назначения (в том числе два регистра-указателя и два индексных регистра);
- АЛУ;
- регистр признаков *F*.

Команды микропроцессора образуют шесть однобайтных регистров очереди команд. Из него операционное устройство последовательно извлекает очередную команду побайтно. Как только в очереди освободятся два регистра, так параллельно с работой операционного устройства и независимо от него устройство с шиной выбирает из памяти программ следующие 2 байта. При этом возможно совмещение, при котором в течение одного цикла обращения к памяти операционное устройство может выполнить две однобайтные команды.

В состав блока регистра общего назначения (РОН) входят четыре 16-разрядных регистра *AX, BX, CX, DX*, допускающих независимую адресацию старших (*H*) и младших (*L*) половин. Такая организация позволяет прямо обрабатывать как байты, так и двухбайтные слова.

Все регистры блока РОН на общих основаниях участвуют в выполнении арифметических и логических операций, представляя операнды и фиксируя результат выполнения операции. Однако в системе команд имеется множество команд, которые специализируют некоторые РОН.

С **регистром-аккумулятором *AX*** связаны операции умножения, деления, преобразования и десятичной коррекции. Участвует во всех операциях ввода/вывода в качестве источника или приемника информации. Он разделен на два регистра: *AL*, который соответствует аккумулятору 8085, и *AH*, который является его расширением.

Регистр базового адреса *BX* используется как источник базового адреса. В некоторой степени соответствует регистровой паре *HL* 8085.

Регистр-счетчик *CX* используется в качестве счетчика в командах сдвигов и зацикливания, а также при операциях с цепочками байтов.

Регистр данных *DX* неявным образом адресуется в командах умножения и деления и, кроме того, содержит адрес порта ввода/вывода при косвенно-регистровой адресации.

Четыре 16-разрядных указательных (таблица 5.2) и индексных (таблица 5.3) регистра (*SP, BP, SI, DI*) предназначены для хранения внутрисегментных смещений, обеспечивая косвенную адресацию и динамичные вычисления исполнительных адресов (рисунок 5.3).

Таблица 5.2 – Указательные регистры

SP (стека) *BP* (базы)	Предназначены для упрощения доступа к данным в текущем сегменте стека, но не в сегменте данных
Если сегмент специально не определен, то смещения *SP* и *BP* по умолчанию относятся к текущему сегменту стека	

SP (стека) и *BP* (базы) предназначены для упрощения доступа к данным в текущем сегменте стека, но не в сегменте данных.

Если сегмент специально не определен, то смещения *SP* и *BP* по умолчанию относятся к текущему сегменту стека.

Таблица 5.3 – Индексные регистры

SI (источника) *DI* (приемника)	Содержат смещения, которые по умолчанию относятся к текущему сегменту данных
Если сегмент специально не определен, то смещения *SP* и *BP* по умолчанию относятся к текущему сегменту стека	

SI (источника) и *DI* (приемника) содержат смещения, которые по умолчанию относятся к текущему сегменту данных. Если сегмент специально не определен, то смещения *SP* и *BP* по умолчанию относятся к текущему сегменту стека. Модуль байт *FL* соответствует регистру признаков *F* – 8085.

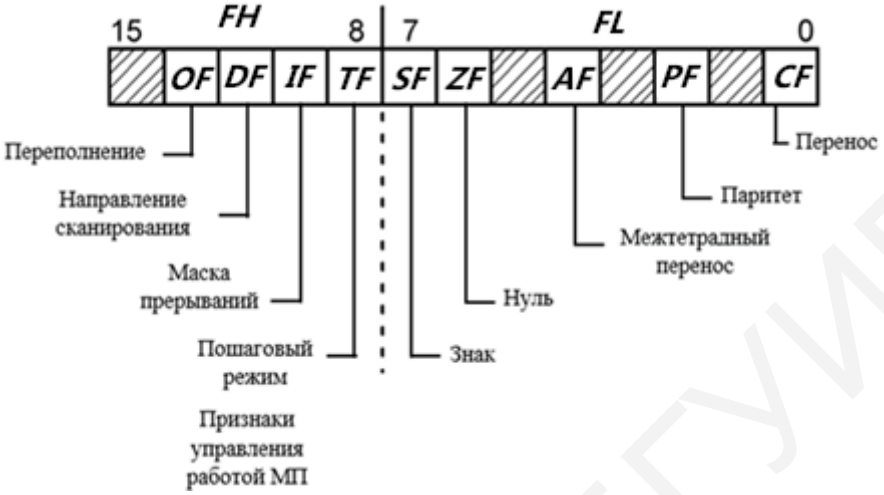


Рисунок 5.3 – 16-разрядный регистр признаков

Эти же регистры могут участвовать в выполнении арифметических и логических операций над двухбайтными словами.

Адресуемая область памяти составляет 1 Мбайт, следовательно, формат адреса равен 20 бит (1 Мбайт = 220). МП генерирует 20-разрядные адреса памяти, но сам он манипулирует логическими адресами, содержащими 16-разрядный сегментный (базовый) адрес и 16-разрядное внутрисегментное смещение. Логические адреса преобразуются МП в физические (исполнительные) адреса (рисунок 5.4).

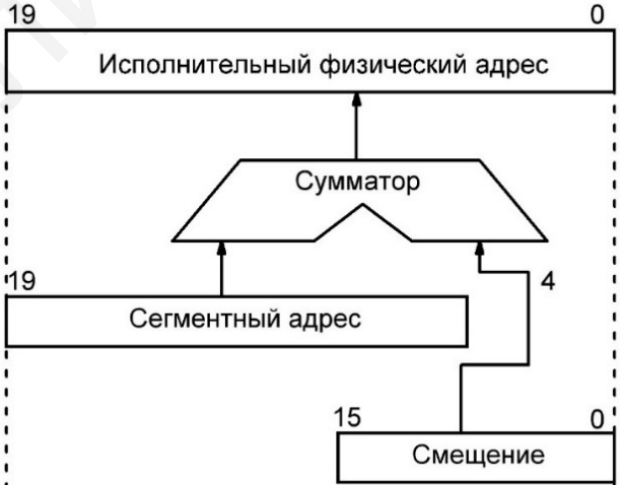


Рисунок 5.4 – Механизм адресации памяти

Язык ассемблера называется также языком символического кодирования и мнемокодом. Язык ассемблера представляет собой машинный язык в символической форме.

Порядок разработки программы:

1. Постановка задачи и составление проекта программы (составить блок-схемы, эскиз выполняемых действий).
2. С помощью редактора ввести команды программы в ЭВМ.
3. Оттранслировать программу с помощью ассемблера, если ассемблер обнаружит ошибки, то исправить их редактором и оттранслировать программу заново.
4. Преобразовать результат работы ассемблера в исполняемый модуль с помощью загрузчика.
5. Исполнить (вызвать) программу.
6. Проверить результаты, если они не соответствуют ожидаемым, найти ошибки и отладить программу.

Чтобы указать ассемблеру имена файлов, надо задать их прямо в командной строке через запятую при вызове соответствующей программы, например: *TASM, Test, Otest, Ltest, Ctest*. Ассемблер формирует правильную машинную команду, используя атрибуты операндов *dst* и (или) *src*.

Примеры записи команды *MOV*:

- *MOV CX, DX* – передача регистр – регистр (слово);
- *MOV AL, DH* – передача регистр – регистр (байт);
- *MOV ES, CX* – передача данных из регистра *CX* в сегментный регистр *EF*;
- *MOV CL*, 0*FFH* – загрузка константы 0*FF* в младший байт регистра *CX*.

Первым задается имя исходного файла, затем объектного, листингового и файла перекрестных ссылок, если какое-либо имя пропущено, то это служит указанием ассемблеру сгенерировать соответствующий файл по стандартному соглашению об именах.

Программу необходимо обработать командой «*TLINK*», которая может связать несколько различных объектных моделей в одну программу и на основе объектного модуля формирует исполняемый загрузочный модуль.

Входной информацией для программы «*TLINK*» являются имена объектных модулей (файлы указываются без расширения *OBJ*).

Если файлов больше одного, то их имена вводятся через разделитель «+». Модули связываются в том же порядке, в котором имена передаются программе *TLINK*. *TKINK* требует указание имени выходного исполняемого модуля. По умолчанию ему присваивается имя первого из объектных модулей, но с расширением «*.exe*». Далее можно указать имя файла, для хранения карты связей (по умолчанию формирование карты не производится). Последнее, что указывается программе «*TLINK*», – это библиотеки программ, которые могут быть включены в полученный при связывании модуль. По умолчанию такие библиотеки отсутствуют. Информацию обо всех этих файлах программа «*TLINK*» запрашивает у пользователя после ее вызова.

Рассмотрим пример реализации кода (таблица 5.4).

Таблица 5.4 – Пример реализации кода

Код	Комментарий
*MOVAX, VAR*1	; переменная 1 помещается в регистр аккумулятора
ADDAX 5	; увеличенное на 5 значение *VAR*_1
*MOV VAR*_2, *AX*	; присвоить *VAR*_2 содержимое регистра аккумулятора
POPAX	; восстановить *AX*
CODEENDS	; конец исходного модуля
ENDSTART	;

Объектный модуль – преобразуется в соответствии с реальными адресами основной памяти, где будет размещаться программа для выполнения. Объектный модуль получается в результате трансляции исходного текста модуля [*.obj*].

Исходный модуль – текстовый файл части программы на каком-либо языке программирования.

Рассмотрим общие особенности оформления программ для МП *I*8086 (рисунок 5.5).

Код	Комментарий
NAME – EXAMPLE	;
ASSUME CS:CODE, DS:DATA, SS:STACK	;
DATASEGMENT	;
VAR_1 DW0	;определить и инициализировать две переменные
VAR_2 DW0	;
DATA ENDS	;
STACK SEGMENT	;
DW 10 DUP(?)	; зарезервировать 10 слов
STK_TOP LABEL WORD	; вершина стека
STAK ENDS	;
CODE SEGMENT	;
START: MOVAX, DATA	; инициализировать регистр DS
MOV DS, AX	; инициализировать регистры SS, SP
MOV AX, STACK	;
MOV SS, AX	;
MOV SP, OFFSETSTK_TOP	;
PROG: PUSH AX	; запомнить (AX)
MOV AX, VAR_1	;
ADD AX, 5	; прибавить к регистру AX 5
MOV VAR_2, AX	; присвоить VAR_2 полученный результат
POP AX	; восстановить AX
CODE ENDS	; конец исходного модуля
END START	

Рисунок 5.5 – Особенности оформления кода

Описание работы программы. В первой строке программы находится директива *NAME* (наименовать), которая присваивает внутреннее имя объектному модулю, генерируемому ассемблером.

Программа состоит из трех различных логических сегментов *DATA*, *STACK* и *CODE*. Каждый сегмент начинается с директивы *SEGMENT* и заканчивается директивой *ENDS*. Причем обе директивы для одного и того же сегмента имеют одинаковые имена.

Логические сегменты соответствуют физическим сегментам в памяти, но привязки логических сегментов к физическим адресам памяти в этом исходном модуле нет.

Директива *ASSUME* (предположить, считать) во второй строке программы сообщает ассемблеру, что регистр *CS* будет содержать базовый адрес сегмента *CODE*, а регистр *DS* – базовый адрес сегмента *DATA*.

В данной программе при адресации переменных регистр *ES* не используется, поэтому указание о нем в директиве *ASSUME* отсутствуют, а по умолчанию принимается *NOTHING* (ничего).

Сегмент данных *DATA* содержит всего две переменные (*VAR*_1 и *VAR*_2), которые определены и инициализированы (установлены в нуль) с помощью директивы *DW*. Обе переменные имеют тип *WORD*, поэтому в основной области данных (данной программы) имеются два слова.

Первая строка в сегменте *STACK* (стек) содержит директиву *DW* с необычным операндом 10 *DUP*(?). Эта директива резервирует 10 слов в памяти, но не инициализирует их.

Конструкция 10 *DUP*(?) означает – «выдать 10 копий значения, находящегося в круглых скобках». Вопросительный знак указывает, что начальные значения резервируемых слов произвольны и никакие предположения о них недопустимы.

Таким образом, в программе для стека резервируется 10 слов (т. е. стек имеет глубину 10 слов). Следующая строка в сегменте *STACK* содержит директиву *LABEL* (отметить).

Директива *STK_TOP LABEL WORD* определяет имя *STK_TOP*, которое относится к слову, находящемуся после 10 слов, т. е. имя *STK_TOP* идентифицирует вершину стека (пока пустого).

Значение смещения *STK_TOP* от начала сегмента *STACK* должно находиться в указателе стека *SP*, когда стек пустой. Содержимое *SP* будет уменьшаться на два при выполнении каждой команды, включающей слово в стек. Когда стек полностью заполнен (в нем находится 10 слов), содержимое *SP* равно нулю. Включение еще одного слова в стек (заполненный) приведет к неуправляемому поведению системы.

Сегмент кода начинается с пяти команд пересылки данных *MOV*. Они осуществляют обязательную инициализацию сегментных регистров *DS, SS* и указателя стека *SP* (сегментные регистры и *SP* необходимо инициализировать до выполнения любой команды, осуществляющей обращение к памяти).

Имена *DATA* и *STACK* идентифицируют базовые адреса сегментов данных и стека.

Выражение *OFFSET STK_TOP* представляет собой значение смещения метки *STK_TOP* от начала содержащего ее сегмента *STACK*.

Программа начинается с метки *PROG*. Программа преследует только иллюстративные цели и выполняет очень простые действия.

Сначала содержимое аккумулятора *AX* включается в стек. Затем в *AX* загружается значение переменной *VAR*_1. Оно увеличивается на 5, и результат запоминается как значение переменной *VAR*_2. После этих действий из стека в аккумулятор извлекается его старое содержимое. Действия данной программы описываются простым оператором присваивания *VAR*_2: *VAR*_1+5.

Последняя строка программы с директивой *END*, имеющей операнд *START*, сообщает ассемблеру о достижении конца исходного модуля и необходимости начать выполнение программы с команды, отмеченной меткой *START*.

Любой исходный модуль состоит из логических сегментов, которые представляют собой блоки машинных команд или инициализаций данных, ограниченные директивами *SEGMENT* и *ENDS*.

Директива *ASSUME* сообщает ассемблеру, что он должен знать о содержимом сегментных регистров для того, чтобы генерировать команды с обращением к памяти.

5.3 Порядок выполнения практической работы

Задание 1. С помощью языка ассемблера и программного пакета «*Sim*8086 *Microprocessor Simulator*» вывести на экран сообщение «*Hello, World*!». Для этого:

1. Ввести тренировочную программу на языке ассемблера (рисунок 5.6).

```
Name «Li»
org 100 h
jmp start
msg: db «Hello World»,
      ODh, OAh, 24h
Start: mov dx, msg
mov ah, 09h
int 21h
mov ah,0
int 16h
ret
```

Рисунок 5.6 – Тренировочный код программы

2. Выполнить отладку программы (*Project* → *Debug Mode*). При наличии ошибок для выхода из режима отладчика использовать пункт меню *Project Terminate*.

3. Осуществить запуск программы, используя пункт меню *Project* → *Run*.

Задание 2. Составить программу на языке ассемблера для заполнения экрана символами, при этом используя основной и вложенный циклы. Основной цикл должен выполняться 254 раза, а вложенный – 2000 раз.

Используя программный пакет «*Sim*8086 *Microprocessor Simulator*», выполнить ввод, отладку и запуск составленной программы.

5.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1 и 2 согласно варианту.
3. Вывод.
4. Ответы на контрольные вопросы.

5.5 Контрольные вопросы

1. Что такое микропроцессор?
2. Какие языки высокого уровня программирования МП вы знаете?
3. Для чего используется язык ассемблера?
4. Перечислите директивы ассемблера.
5. Перечислите программы-ретрансляторы.

6 Практическая работа №6

ПРОГРАММИРОВАНИЕ И ОТЛАДКА ПРОЦЕДУР АРИФМЕТИЧЕСКИХ И ЛОГИЧЕСКИХ ПРЕОБРАЗОВАНИЙ ДАННЫХ В МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВАХ

6.1 Цель работы

Изучение структуры и особенностей работы портов микроконтроллеров (МК) семейства *dsPIC*33*F*, схемы подключения входных и выходных дискретных сигналов к МК *dsPIC*33*F*, особенностей программирования ввода/вывода дискретных сигналов на языке программирования *C*, а также создание проекта по следующей схеме: составить исходный код программы ввода/вывода дискретных сигналов по заданному алгоритму, откомпилировать программу в среде *MPLABIDE* 8, записать в память программ МК *dsPIC*33*FJ*32*MC*204.

6.2 Теоретические сведения

Описание архитектуры микроконтроллеров *dsPIC*33*F*. Микроконтроллер *dsPIC*33FJ32*MS*204 относится к семейству 16-разрядных *Flash* МК с поддержкой команд цифровой обработки сигналов. Ядро МК семейства *dsPIC*33*F* построено по модифицированной гарвардской архитектуре с расширенной системой команд. Микроконтроллерные инструкции *MCU* и команды цифровой обработки сигналов *DSP* равноправно интегрированы в архитектуру МК и обрабатываются одним ядром. Большинство инструкций (команд) имеют длину в одно слово разрядностью 24 бит и выполняются за один цикл, за исключением команд деления, переходов, команд пересылки данных из регистра в регистр и табличных команд.

Система команд содержит множество режимов адресации и оптимизирована для получения максимальной эффективности при программировании МК на языке высокого уровня *C*.

МК обладает обширным набором периферийных модулей:

1) порты ввода/вывода общего назначения;
2) таймеры;
3) модули захвата и сравнения;
4) модули генерации ШИМ сигналов *PWM*;
5) модуль интерфейса квадратурного энкодера *EQI*;
6) 10- и 12-битный аналого-цифровой преобразователь *ADC*;
7) коммуникационные интерфейсы (*UART*, *SPI*, I^2C и др.).

Общая структурная схема ядра и периферийных модулей МК семейства *dsPIC*33*F* показана на рисунке 6.1.

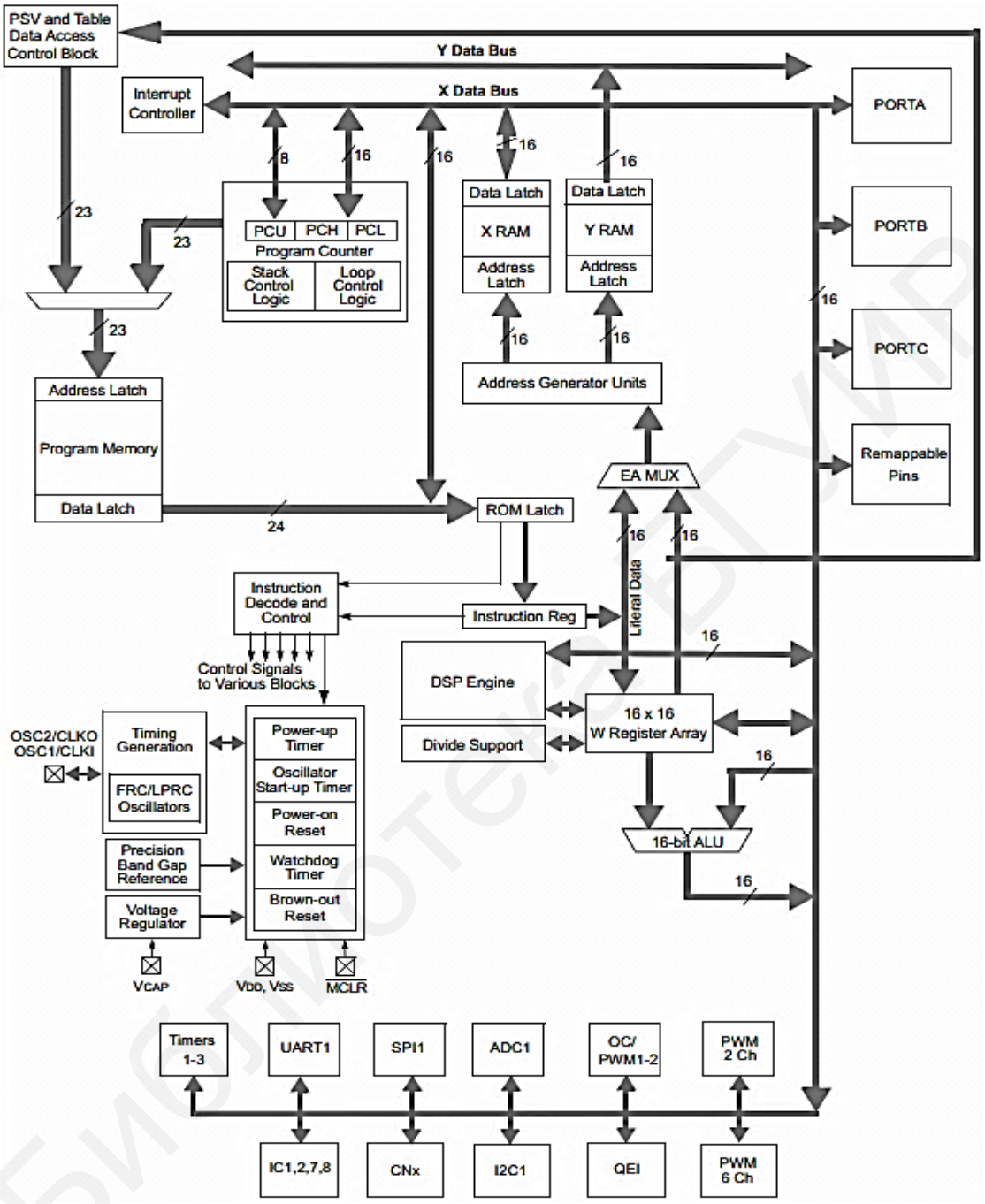


Рисунок 6.1 – Общая структурная схема ядра и периферийных модулей микроконтроллера семейства *dsPIC*33*F*

16-битное арифметико-логическое устройство (АЛУ) позволяет выполнять за один командный цикл следующие операции: сложение, вычитание, битовый сдвиг и поразрядные логические операции, включая инверсию. АЛУ выполняет операции с 16-битными словами или байтами в зависимости от синтаксиса инструкции. Результаты операции влияют на флаги состояния регистра *SR*.

Операндами и приемниками результатов могут быть использованы рабочие регистры W_0–W_{15} или память данных в зависимости от режима адресации инструкции.

В состав ядра МК входит умножитель 17×17 бит, который позволяет выполнять операции умножения 16×16, 16×8 и 8×8 бит, знаковые, беззнаковые и смешанные за один командный цикл. При знаковом и смешанном умножении используется расширение знака. Операция умножения не влияет на флаги состояний. В качестве операндов используются рабочие регистры, результат умножения возвращается в указанную регистровую пару.

Аппаратная поддержка деления позволяет значительно сократить время выполнения математических алгоритмов. Операции деления 32/16 и 16/16 бит (знаковые и беззнаковые) выполняются за 18 командных циклов. В качестве операндов используются рабочие регистры, после выполнения деления доступен результат и остаток.

Ядро МК *dsPIC*33*F* содержит специальный модуль (процессор) поддержки *DSP*-инструкций (*DSP Engine*), который разработан с учетом оптимальной обработки в режиме реального времени. Структурная схема *DSP*-процессора показана на рисунке 6.2.

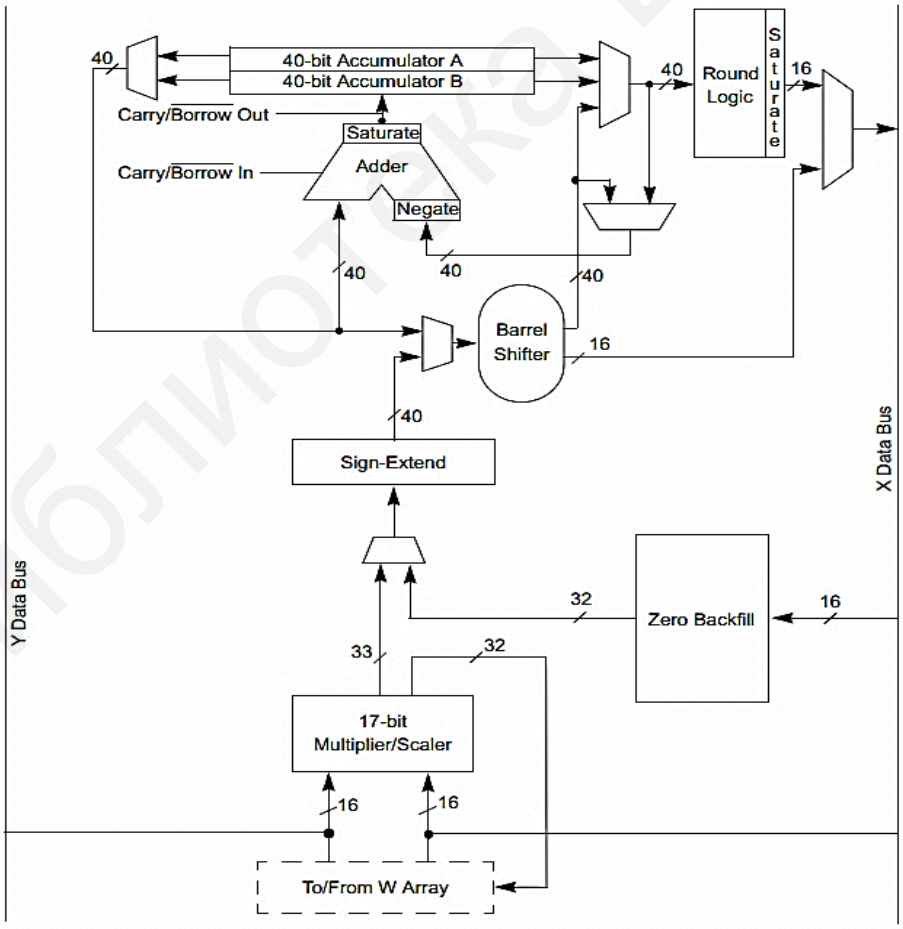


Рисунок 6.2 – Структурная схема *DSP*-процессора

Структура *DSP*-процессора включает в себя:

1) высокоскоростной 17-разрядный аппаратный умножитель;
2) 40-разрядный сдвиговый регистр (*Barrel Shifter*);
3) 40-разрядный сумматор/вычитатель (*Adder*) с двумя целевыми аккумуляторами *ACCA* и *ACCB*, логикой округления и насыщения результата.

40-разрядные аккумуляторы *ACCA* и *ACCB* могут использоваться сумматором/вычитателем в качестве источника и приемника данных. Для инструкций *ADD* и *LAC* данные могут быть произвольно масштабированы через сдвигающий регистр *Barrel Shifter*.

Сумматор/вычитатель (*Adder*) является 40-разрядным сумматором со входом переноса/заема. В случае вычитания данные на второй вход подаются в дополнительном коде. Сумматор/вычитатель генерирует биты насыщения *SA*/*SB* и переполнения *OA*/*OB* в аккумуляторе *ACCA* и *ACCB*, которые защелкиваются и отражаются в регистре статуса *SR*. Переполнение бита 39 – **«катастрофическое переполнение»**, в котором уничтожается знаковый разряд аккумулятора. Переполнение битов защиты аккумуляторов (биты с 32 по 39) – **«исправимое переполнение»**. Биты переполнения *OA*/*OB* устанавливаются, когда все биты защиты аккумуляторов *ACCA* и *ACCB* неидентичны друг другу.

Логика округления – комбинационный блок, который выполняет стандартное округление без смещения и со смещением 40-разрядных данных модуля *DSP* к 16-разрядному выходному формату.

Режим округления определяется состоянием бита *RND* в регистре конфигурации ядра *CORCON*. Например, для стандартного округления без смещения следует взять бит 15 аккумулятора, расширить его нулем и добавить в слово *ACCxH* (биты с 16 по 31 аккумулятора).

Блок логики насыщения записи области данных принимает 16-разрядное значение от блока логики округления вместе с состоянием переполнения от первоначального источника (аккумулятора). Они используются для округления выходного 16-разрядного значения, которое записывается в память данных.

Умножитель, делитель и сдвигатель задействованы в исполнении *MCU*- и *DSP*-инструкций. В качестве примеров *DSP*-инструкций умножителя можно привести следующие команды:

1) *MPY* – умножение вида $A = B * C$;
2) *MAC* – умножение с накоплением $A = A + B * C$;
3) *EDAC* – вычисление евклидовой метрики $A = A + (B - C)^2$.

Применение команды *EDAC* особенно эффективно в приложениях обработки изображений, где требуется вычисление расстояний по координатам объектов.

В МК возможно одновременное (за один машинный цикл) чтение инструкции, обращения к данным в памяти, к рабочему регистру и запись результата в память, что поддерживает трехоперандные команды, позволяя выполнить команды типа $A + B = C$ за один машинный цикл.

При выполнении инструкции типа *MAC* (умножение с накоплением) одновременно с умножением может осуществляться выборка двух операндов из секторов *X* и *Y* памяти данных. Это увеличивает эффективность циклически выполняемых операций.

40-разрядный сдвиговый регистр (*Barrel Shifter*) позволяет выполнять за один командный цикл операции сдвига и вращения (циклического сдвига) на произвольное число разрядов (до 15 – при сдвиге вправо и до 16 – при сдвиге влево). Сдвиг может быть логическим (без расширения знака) или арифметическим (с расширением знака). Для *DSP*-операций сдвига получается 40-разрядный результат, а для *MCU*-операций – 16-разрядный. Данные от шины *X* представлены сдвигателю от 16 до 31 двоичного разряда для правых сдвигов и от 0 до 15 разрядов для левых сдвигов.

МК семейства *dsPIC*33*F* имеет шестнадцать 16-битных рабочих регистров *W* (от англ. *Work*), обозначаемых W_0–W_{15}, которые могут использоваться как регистры данных, адреса или смещение адреса. Часть из рабочих регистров имеют два набора – **«основной»** и **«теневой»**. Такие же **«теневые»** наборы присутствуют у группы разрядов регистра статуса *SR*, а также у регистров поддержки инструкции *DO*. Это позволяет вести быстрое переключение между основной задачей и функциями обработки прерываний при помощи всего двух команд – *PUSH.S* и *POP.S*. Специальные назначения имеют несколько рабочих регистров:

1) W_4–W_{11} – указатели данных в *DSP*-инструкциях;
2) W_{14} – программный указатель фрейма стека;
3) W_{15} – программный указатель стека.

Совместно с регистром ограничения стека (*SPLIM*) регистры W_{14} и W_{15} используются для размещения стека в памяти, выделения в нем отдельных областей (фреймов) и контроля переполнения стека. Функция, выполняемая регистром, определяется режимом адресации, который используется в данной инструкции.

В регистре W_{15} содержится указатель стека *SP*, который автоматически модифицируется при обработке исключений, вызовов подпрограмм и обслуживании прерываний.

При инициализации МК в регистр W_{15} записывается значение 0*x*0800, что обеспечивает установку указателя стека *SP* на начало области памяти, которая доступна пользователю. Указатель стека *SP* всегда инкрементируется после помещения значения в стек и декрементируется перед чтением значения из стека. В случае переполнения или опустошения стека ядром МК генерируется аппаратное исключение. Размер стека определяется программно. При вызове подпрограммы в стек заносится адрес возврата и часть регистра статуса *SR*.

Счетчик циклов *RCOUNT* используется при выполнении инструкции REPEAT для загрузки числа повторений следующей за ней команды.

При выполнении инструкции *DO* МК заполняет следующие регистры:

1) *DCOUNT* – счетчик циклов, который хранит требуемое число повторов;
2) *DOSTART* – адрес первой инструкции цикла;
3) *DOEND* – адрес последней инструкции цикла.

«Теневые» регистры используются как регистры временного хранения. Не один из теневых регистров не доступен напрямую.

Регистр страницы таблицы данных *TBLPAG* используется для аппаратной поддержки инструкций табличного чтения-записи, позволяющих получить доступ как к слову (16 бит) так и к байту программной памяти.

Регистр страницы видимости пространства программы *PSVPAG* – 8-разрядный регистр, содержащий биты «22…15» адресов программной памяти, т. е. определяет страницу программной памяти, которая отображается в верхнюю половину области данных (остальные биты адреса в программном счетчике *PC*). Механизм *PSV* применяется, если алгоритм содержит большие массивы констант, например коэффициенты цифровых фильтров и т. п.

Регистр статуса *SR* содержит флаги арифметических операций инструкций *MCU* (переноса *C* – бит 0, нуля *Z* – бит 1, переполнения *OV* – бит 2, знака *N* – бит 3 и частичного переноса *DC* – бит 8). Флаг частичного переноса *DC* устанавливается, при наличии переноса из младшей тетрады для байтовых данных или из младшего байта для двухбайтовых данных. Биты 7–5 содержат биты $IPL_2 - IPL_0$ (*Interrupt Priority Level*), в которых устанавливается текущий уровень приоритета прерываний ядра МК.

Старший байт регистра статуса *SRH* содержит флаги, сигнализирующие о переполнении или насыщении аккумуляторов *ACCA* и *ACCB* при исполнении *DSP*-инструкций, что может вызывать генерацию прерывания. Кроме того, имеется бит активности цикла *DO* – это бит *DA* (бит 9), а также бит активности цикла *REPEAT* – бит *RA* (бит 4).

Карта памяти программ МК *dsPIC*33*F* линейная и несегментированная. Программный счетчик *PC* имеет разрядность 23 бит, младший бит всегда равен нулю, чтобы обеспечить выравнивание данных при выборке инструкций. Следовательно, для задания непосредственно адреса используется 22 бита и, таким образом, эффективное количество адресуемых инструкций равно 2^{22} (приблизительно равно 4 М слов).

Адрес памяти удобно рассматривать в виде младшего и старшего слова, при этом старший байт старшего слова не используется и равен нулю (фантомный байт). Младшее слово всегда располагается по четному адресу, а старшее – по нечетному. Адреса памяти программ всегда выравниваются по границе слова и при выполнении программ всегда инкрементируются или декрементируются на 2.

Пользовательским программам доступна область памяти в диапазоне адресов 0*x*000000…0*x*7*FFFFF*. Память программ организована в виде блоков, адресуемых пословно.

По адресам 0*x*000000 и 0*x*000002 размещается команда перехода к фактическому началу программы, при этом по первому адресу располагается код операции инструкции *GOTO*, а по второму – собственно адрес точки входа в программу. В памяти программы также размещаются две таблицы векторов прерываний. Одна из них располагается в диапазоне адресов

0*x*000004–0*x*0000*FE*, а вторая (альтернативная) – в диапазоне адресов 0*x*000104–0*x*0001*FE*.

Половину карты программной памяти занимает так называемое **«конфигурационное пространство»**. Во всех 16-разрядных МК физически реализована только небольшая часть этого сектора – это слово конфигурации, определяющее режим работы МК после сброса и идентификационный код кристалла, предназначенный для определения МК аппаратными средствами разработки программ.

Шина адреса данных МК *dsPIC*33*F* является 16-битной и позволяет адресовать до 64 Кбайт памяти ОЗУ, которая физически выполнена в виде статической памяти с возможностью байтового доступа. Почти все инструкции, работающие с ОЗУ, имеют спецификатор, указывающий будет ли осуществляться доступ к слову (16 бит) или к байту (8 бит).

В начале ОЗУ расположена область регистров специального назначения *SFR* объемом 2 Кбайта. Все регистры *SFR* расположены статически по одним адресам.

Данные в памяти выравниваются по границе слова, при этом младшие байты *LSB* имеют четные, а старшие *MSB* – нечетные адреса.

С адреса 0*x*0800 начинается сектор ОЗУ общего назначения, максимальный объем которого составляет 30 Кбайт (в разных МК только часть этого сектора может быть реализована физически). Верхнюю половину ОЗУ занимает область, в которую отображается часть программной памяти при использовании механизма *PSV*.

В семействе *dsPIC*33*F* реализовано два адресных генератора *AGU*, что позволяет инструкциям *DSP*-ядра делать две выборки из ОЗУ за один командный такт. Это объясняет разделение области ОЗУ общего назначения на два сектора *X* и *Y*. Однако это разделение не является сегментированием или ограничением – для всех инструкций *CPU*-ядра (*MCU* и *DSP*) сектор ОЗУ общего назначения линеен. Границы адресного пространства секторов *X* и *Y* зависят от конкретного устройства. Режим модульной адресации, используемой в *DSP*-инструкциях поддерживают *X* и *Y* сектора ОЗУ. Адресный генератор *X* поддерживает также бит-реверсивную адресацию для *DSP*-инструкций, что значительно упрощает реализацию алгоритмов быстрого преобразования Фурье *FFT*.

Первые 8 Кбайт ОЗУ (включая область *SFR*) называются пространством ближней памяти (*Near Data Space*). Прямая адресация возможна только к этому сегменту. Остальная часть ОЗУ может быть адресована косвенно.

Инструкции МК *dsPIC*33*F* могут быть регистровыми (модифицируют содержимое регистров) и адресными (используют содержимое рабочих регистров для доступа к данным, находящимся в памяти. В свою очередь, адресные инструкции могут быть как с прямой регистровой адресацией, так и с косвенной регистровой адресацией.

Прямая регистровая адресация используется для доступа к содержимому 16 рабочих регистров W_0–W_{15}, причем обращение возможно как к целому слову (16 бит), так и к байту.

Косвенная регистровая адресация применяется для доступа к любой ячейке памяти данных посредством формирования исполнительного адреса данных по содержимому одного или нескольких рабочих регистров, что является указателем на область памяти данных. Дополнительные возможности такого метода адресации обеспечиваются механизмами пред- и постинкремента содержимого регистра, что позволяет последовательно обрабатывать непрерывный диапазон адресов в памяти данных.

Рассмотрим примеры применения различных методов адресации.

Пример 1. Прямая адресация.

Адрес операнда указывается непосредственно в операционном слове инструкции. Операнд должен быть расположен в области *Near Data Space* (первые 8 Кбайт ОЗУ) (таблица 6.1).

Таблица 6.1 – Пример реализации прямой адресации

Код	Комментарий
ADD 0*x*0900, *W*0	; Сложение значения по адресу 0*x*0900 с W_0 результат сохраняется в W_0

Пример 2. Расширенный режим инструкции *MOV*.

Прямая адресация, но может адресоваться вся память ОЗУ (таблица 6.2).

Таблица 6.2 – Пример реализации в расширенном режиме

Код	Комментарий
MOV 0*x*2500, *W*7	; Сохранение значения по адресу 0*x*2500 в регистр W_7

Пример 3. Прямая регистровая адресация.

В качестве операндов используются значения рабочих регистров (таблица 6.3).

Таблица 6.3 – Пример реализации прямой регистровой адресации

Код	Комментарий
IOR *W*0, *W*2, *W*5	; Поразрядное логическое ИЛИ регистров W_0 и W_2, сохранение результата в W_5

Пример 4. Косвенная адресация.

Может использоваться в большинстве инструкций (таблица 6.4).

Таблица 6.4 – Пример реализации косвенной адресации

Код	Комментарий
*ADD W*4, [*W*5], [*W*6]	; Сложение W_4 со словом по указателю W_5, сохранение результата по указателю W_6

Пример 5. Косвенная адресация с пред-/постинкрементом и декрементом. При этом учитываются правила адресной арифметики в зависимости от типа операнда (байт или слово) (таблица 6.5).

Таблица 6.5 – Пример реализации косвенной адресации с пред-/постинкрементом и декрементом

Код	Комментарий
MOV [++*W*0], [*W*1- -]	; Увеличение указателя W_0 на 2, перемещение значения по указателю W_0 в ячейку, на которую указывает W_1, уменьшение указателя W_1 на 2

Пример 6. Косвенная адресация со смещением (таблица 6.6).

Таблица 6.6 – Пример реализации косвенной адресации со смещением

Код	Комментарий
MOV [*W*4 + *W*5], [*W*6++]	; Получение указателя на ячейку операнда, путем сложения W_4 и W_5, перемещение значения по полученному указателю по адресу W_6, увеличение значения W_6 на 2

Поддержка различных методов адресации позволяет получать очень компактный код при использовании компиляторов с языков высокого уровня.

6.3 Порядок выполнения практической работы

1. Выполнить тренировочное задание на примере программы вычисления значения выражения (25 + 13)(18 – 9). Результаты промежуточных вычислений хранить в рабочих регистрах. Выделить младшую тетраду результата и поместить ее в старшую. Проверить корректность выполнения алгоритма в симуляторе *MPLAB IDE*, а также проверить содержимое рабочих регистров МК.

2. Выполнить индивидуальное задание согласно заданному варианту (таблица 6.7).

Таблица 6.7 – Исходные данные к индивидуальному заданию

Номер варианта	Номер задания	Номер варианта	Номер задания
1, 2	1	9, 10	5
3, 4	2	11, 12	6
5, 6	3	13, 14	7
7, 8	4	–	–

3. Составить алгоритм и исходный код программы на языке ассемблера *ASM*30, скомпилировать ее и проверить корректность выполнения алгоритма в симуляторе *MPLAB SIM*, а также проверить содержимое использованных рабочих регистров МК.

Задание 1. Загрузить в регистр число 15. Сложить его с 25 и результат поместить на вершину стека. Поместить по адресу 20*h* внутренней памяти данных младшую десятичную цифру результата, а по адресу 21*h* – старшую.

Задание 2. Найти разницу чисел 4836 и 2454. Младший байт результата поделить на 2. Поместить по адресу 30*h* внутренней памяти данных младшую десятичную цифру результата, а по адресу 32*h* – старшую.

Задание 3. Найти адрес ячейки памяти данных путем перемножения двух чисел 0*Ch* и 0*Eh*. В эту ячейку записать результат логической операции «Исключающее ИЛИ» между текущим содержимым регистра W_0 и числа 09*h*.

Задание 4. Найти частное чисел 236 и 59. Результат умножить на 8, используя операции сдвига. По вычисленному таким образом адресу ячейки внутренней памяти данных разместить результат двойного декремента полученного числа.

Задание 5. Загрузить регистр W_7 числом 023*h*. Найти сумму W_7 + 32. В ячейку внутренней памяти данных, расположенную по вычисленному таким образом адресу, загрузить число десятичных единиц результата сложения.

Задание 6. Вычислить значение выражения (81 + 64)(112 – 25) *OR* 10011010*b*, сохраняя промежуточные результаты в стеке.

Задание 7. Найти разницу чисел 4801 и 209. Число десятичных единиц старшего байта результата поместить в старшую тетраду порта *RA*. Младшую тетраду оставить без изменений.

6.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1–7 согласно варианту.
3. Вывод.
4. Ответы на контрольные вопросы.

6.5 Контрольные вопросы

1. Что такое микроконтроллер?
2. Перечислите особенности *dsPIC*33*F*.
3. Каково назначение *MPLab IDE*?
4. Сколько рабочих регистров имеет *dsPIC*33*F*?
5. Перечислите способы адресации *dsPIC*33*F*.

7 Практическая работа №7

ПРОГРАММИРОВАНИЕ И ОТЛАДКА ПРОЦЕДУР ФОРМИРОВАНИЯ ИМПУЛЬСНЫХ СИГНАЛОВ С ЗАДАННЫМИ ВРЕМЕННЫМИ ПАРАМЕТРАМИ

7.1 Цель работы

Изучение структуры и особенностей работы портов микроконтроллеров (МК) семейства *dsPIC*33*F*, схемы подключения входных и выходных дискретных сигналов к МК *dsPIC*33*F*, особенностей программирования ввода/вывода дискретных сигналов на языке программирования *C*, создание проекта по следующей схеме: составить исходный код программы ввода/вывода дискретных сигналов по заданному алгоритму, откомпилировать программу в среде *MPLAB IDE* 8, а также исследование работы дискретных входов и выходов.

7.2 Теоретические сведения

МК семейства *dsPIC*33*F* предоставляют разработчику широкие возможности по работе с дискретными сигналами благодаря наличию достаточного количества портов ввода/вывода и некоторым функциям, расширяющим функциональность самих портов.

Порты ввода/вывода общего назначения – это простейшие периферийные устройства, с помощью которых МК может получать цифровые данные и управлять другими устройствами (рисунок 7.1).

Все выводы МК, кроме *Vdd*, *Vss*, *AVdd*, *AVss*, *VCap* и *MCLR*, могут использоваться как периферийными модулями, так и параллельными портами ввода/вывода. Все вводные линии портов имеют триггер Шмидта по входу для исключения влияния электромагнитных помех и шумов.

Линии ввода/вывода МК разделены на три порта: *RA*, *RB*, *RC*. Подавляющее большинство линий ввода/вывода всех портов имеют дополнительные функции и могут использоваться различными периферийными модулями МК.

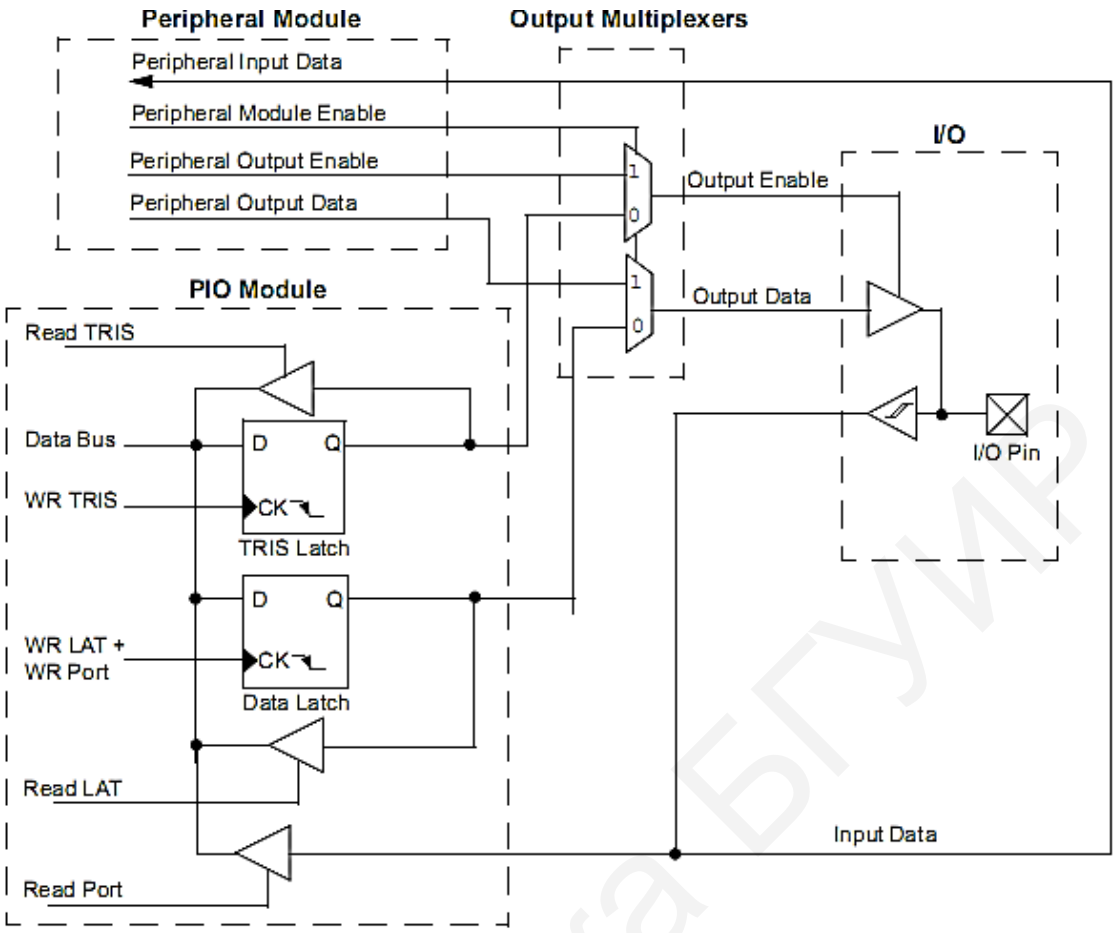


Рисунок 7.1 – Структура линии порта, объединенной с выводами периферийных функций

Мультиплексор (см. рисунок 7.1) выбирает, каким образом функционирует линия порта (в качестве части периферийного модуля либо в качестве линии параллельного порта ввода/вывода). Таким образом, перед использованием таких линии в качестве линий порта ввода/вывода необходимо предварительно отключить соответствующие периферийные устройства. Например, при использовании порта *RA2* как линии ввода, необходимо отключить дополнительную функцию порта *RA2* (выход тактирующего сигнала внутреннего генератора) директивой «*_FOSC* (*OSCIOFNC_ON & POSCMD_NONE*)».

Для работы с портами каждый из них имеет три специальных регистра:

1. *TRISx* – регистр направления данных – задает, каким образом используется линия порта (как вход либо как выход). При установке соответствующего бита в 1 линия порта будет сконфигурирована как вход.

2. *LATx* – регистр выводов порта – установка соответствующего бита данного регистра в 1 позволяет установить высокий уровень сигнала на выходе линии порта, при установке соответствующего бита в 0 – низкий уровень сигнала.

3. *PORTx* – регистр состояния порта – чтение соответствующего бита из данного регистра позволяет получить состояние сигнала (высокий либо низкий уровень) на входе линии порта.

Регистры порта *A* называются *TRISA*, *LATA*, *PORTA*, порта *B* – *TRISB*, *LATB*, *PORTB*, порта *C* – *TRISC*, *LATC*, *PORTC*.

Для того чтобы установка бита регистра *LATx* приводила к соответствующему изменению состояния линии вывода порта, необходимо, чтобы эта линия была предварительно сконфигурирована как выход установкой требуемого бита в регистре *TRISx*. Аналогично, чтобы чтение бита регистра *PORTx* отображало действительное состояние линии ввода порта, необходимо, чтобы эта линия была предварительно сконфигурирована как вход установкой требуемого бита в регистре *TRISx*.

Таким образом, настройка линии 2 порта *RA* на вход, а линии 15 порта *RB* на вывод осуществляется следующим образом:

- *TRISAbits*.*TRISA*2 = 1;
- *TRISBbits*.*TRISB*15 = 0.

7.3 Порядок выполнения практической работы

Задание 1. Создать проект для отображения на светодиоде *VD*1 состояния тумблера *SA*1, а на *VD*2 – состояния *SA*2. Блок-схема алгоритма работы программы приведена на рисунке 7.2.

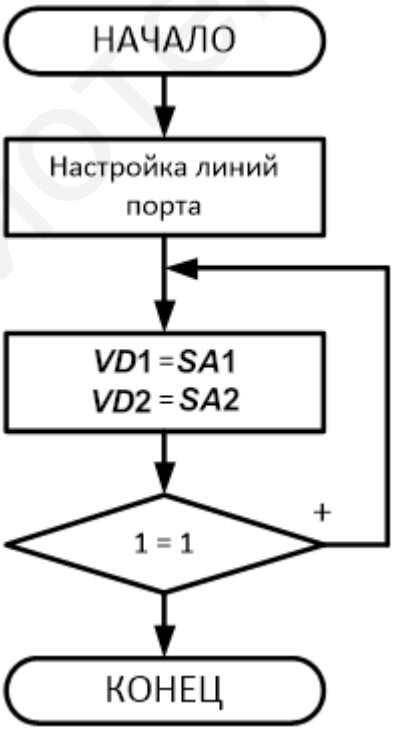


Рисунок 7.2 – Блок-схема алгоритма программы

Листинг (исходный код) программы к заданию для прошивки МК приведен на рисунке 7.3.

```
#include <P33FJ32MC204.h>
_FOSC(OSCIOFNC_ON & POSCMD_NONE)
 //отключение дополнительной функции порта RA2 – выход
//тактирующего сигнала внутреннего генератора
        int main()
        {
            TRISBbits.TRISB15 = 0; // настройка порта
                                                //RB15 на выход
            TRISBbits.TRISB13 = 0; // настройка порта
                                                //RB13 на выход
            TRISAbits.TRISA2 = 1; //настройка порта
                                                //RA2 на вход
            TRISAbits.TRISA3 = 1; //настройка порта
                                                //RA3 на вход
    while (1)
            {
            LATBbits.LATB15 = PORTAbits.RA2;
            // установка соответствующего
                    // уровня сигнала на RB15
            LATBbits.LATB13 = PORTAbits.RA3;
            // установка соответствующего
            // уровня сигнала на RB13
            }
}
```

Рисунок 7.3 – Исходный код

Задание 2. Создать проект, который реализует следующее условие:
- если *SA*1 = 1 и *SA*2 = 0, то *VD*1 = 0 и *VD*2 = 0;
- если *SA*1 = 0 и *SA*2 = 1, то *VD*1=1, *VD*2 = 1;
- если *SA*1 = *SA*2, то *VD*1= 0, *VD*2 = 1.

Блок-схема алгоритма работы программы приведена на рисунке 7.4.

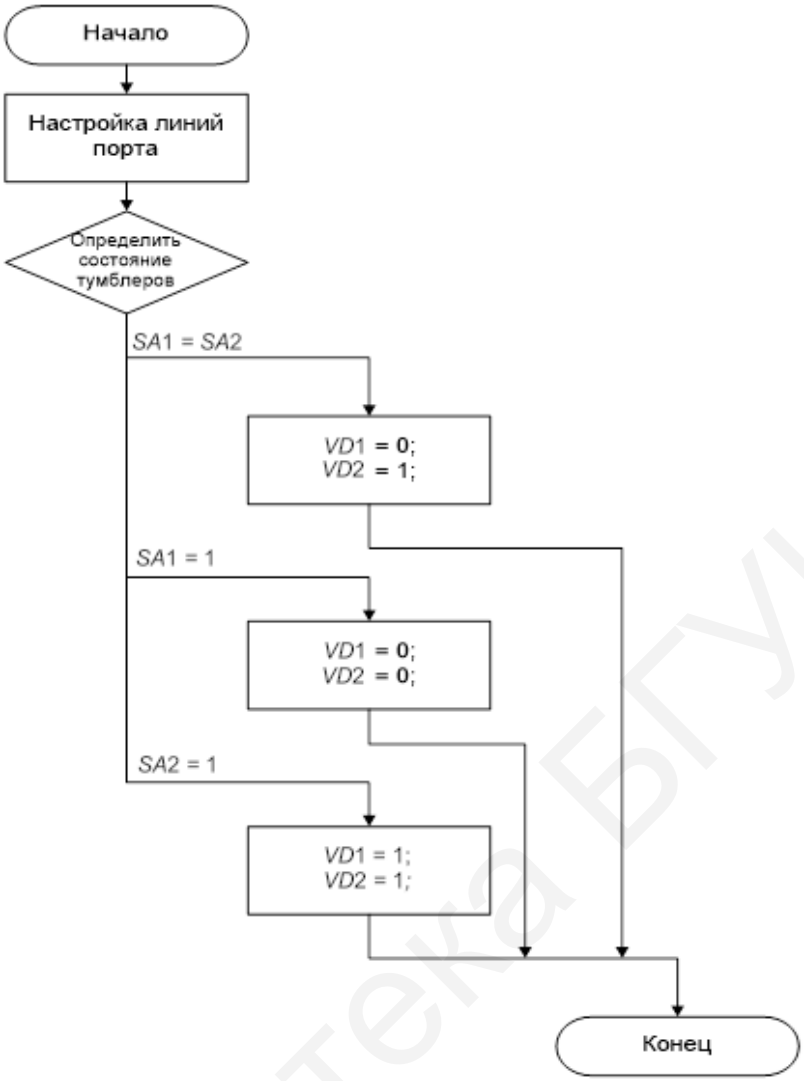


Рисунок 7.4 – Блок-схема алгоритма программы

7.4 Содержание отчета

1. Цель работы.
2. Выполненные задания 1 и 2 согласно варианту с результатами отладки и симуляции программы в интегрированной среде разработки *MPLab*.
3. Вывод.
4. Ответы на контрольные вопросы.

7.5 Контрольные вопросы

1. Перечислите порты микроконтроллера *dsPIC*33*F*.
2. Каково назначение портов ввода вывода микроконтроллера *dsPIC*33*F*?
3. Каково назначение регистра *TRISx*?
4. Каково назначение регистра *LATx*?
5. Каково назначение регистра *PORTx*?

8 Практическая работа №8

АНАЛОГО-ЦИФРОВОЕ ПРЕОБРАЗОВАНИЕ СИГНАЛОВ В МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВАХ

8.1 Цель работы

Изучение основных принципов работы АЦП, исследование структуры построения АЦП, ознакомление с основными техническими параметрами АЦП.

8.2 Теоретические сведения

Сигнал, величина которого зависит от времени, называется **динамическим**. Динамический сигнал можно описать как функцию времени. Если функция определена для всех точек на временной оси (на определенном конечном интервале времени), то мы говорим, что это непрерывный во времени сигнал. Если сигнал, а значит, и функция могут принимать любое значение в пределах некоторого интервала значений, то сигнал называют **сигнал с непрерывным множеством значений**. Непрерывные сигналы, изменяющиеся по мере того, как происходят изменения в непрерывных физических процессах, называются **аналоговыми**.

Существуют также **дискретные** во времени сигналы. Значение такого сигнала известно только в определенные моменты времени. Дискретный во времени сигнал можно рассматривать как результат взятия выборок непрерывного во времени сигнала. Операцию взятия выборок еще называют **оцифровыванием**.

Выборка берется через равные промежутки времени, называемые периодом взятия выборки или **периодом дискретизации** $T_д$. Период дискретизации обратно пропорционален частоте дискретизации $f_д$. Началу взятия выборки соответствует момент времени $t_0 = 0$, а окончание взятия выборки происходит в момент времени

$$t_N = N \cdot T_д, \tag{8.1}$$

где N – количество точек выборки.

Произвольный момент времени определяется по формуле

$$t_i = T_д i = \frac{1}{f_д} i, \tag{8.2}$$

где i – номер выборки.

С учетом формулы (8.2) оцифрованная гармоническая функция $x(t)$ будет записана в виде выражения

$$x(t) = x_a \cos(2\pi f T_д i) = x_a \cos\left(2\pi \frac{f}{f_д} i\right). \quad (8.3)$$

Важным параметром при цифровом анализе является безразмерная величина

$$m = \frac{1}{f T_д} = \frac{f_д}{f}, \quad (8.4)$$

которая имеет смысл количества точек выборки на период. На самом деле результат взятия выборки сигнала частотой f = 50 Гц с периодом дискретизации $T_д$ = 1/10 000 с будет абсолютно идентичен сигналу с частотой 500 Гц, оцифрованному с периодом дискретизации $T_д$ = 1/100 000 с. Для обоих случаев m = 200 точек на период, а сигнал будет описан выражением

$$\cos\left[\left(\frac{2\pi}{2\pi}\right) i\right]. \quad (8.5)$$

Таким образом, для восстановления сигнала необходимо отдельно запоминать величину $T_д$.

Точно так же, как и момент времени, величина сигнала тоже может принимать только некоторые дискретные значения между заданным верхним $X_{\max}$ и нижним $X_{\min}$ пределами. Тогда сигнал называют **дискретным по величине** сигналом. Количество значений выборочного сигнала равно 2^n, где n – разрядность преобразования. При этом любое значение сигнала будет записано как произведение целого числа $D = a_{n-1}a_{n-2} \ldots a_1 a_0$ на минимальный шаг по амплитуде:

$$X_0 = \frac{X_{\max} - X_{\min}}{2^n}. \quad (8.6)$$

Процесс преобразования сигнала с непрерывным множеством значений в сигнал с дискретными значениями называется **квантованием** и реализуется с помощью аналого-цифрового преобразователя (АЦП). Этот процесс происходит за конечный промежуток времени, поэтому непрерывность сигнала теряется, что приводит к дискретизации сигнала и по времени. Сигналы, дискретные по величине и по времени, называются **цифровыми**.

В цифровых вычислительных устройствах информация о сигнале хранится в виде столбца данных, где номер строки соответствует номеру выборки i, а сама строка содержит значение выборки в виде числа D. Величины $T_д$ и X_0 хранятся отдельно и нужны только для восстановления сигнала в аналоговую форму. Устройство, выполняющее обратное преобразование сигналов из цифровой формы в аналоговую, называется цифроаналоговым преобразователем (ЦАП).

Для точного преобразования и восстановления сигналов используются высокоточные источники частоты $f_д$, которые, как правило, выполняются на кварцевых резонаторах и в особо ответственных случаях помещаются в термостат.

Сигналы, поступающие на вход АЦП, являются источниками напряжения, и переменная X может быть заменена переменной U. Напряжения просто делятся, но сложно умножаются, поэтому вместо величины U_0 обычно сохраняют величину $U_{\max} - U_{\min}$, которая называется **опорным напряжением** $U_{оп}$. Минимальный квант напряжения в соответствии с формулой (8.6) описывается выражением

$$U_0 = \frac{U_{оп}}{2^n}. \qquad (8.7)$$

Источники опорного напряжения служат для сравнения с ними исходного сигнала при аналого-цифровом преобразовании, и при этом преобразовании происходит восстановление аналогового сигнала относительно $U_{оп}$.

Реальные сигналы, параметры которых приходится определять, в основном являются аналоговыми. Для обработки сигнала на цифровом вычислительном устройстве его необходимо перевести в цифровую форму. Для этого существует операция взятия выборки, или оцифровывания. Эта операция выполняется специальными устройствами, называемыми **аналого-цифровыми преобразователями (АЦП)**. Оцифровывание переводит аналоговый сигнал в сигнал, дискретный по времени и по амплитуде.

Фрагмент аналогового сигнала и соответствующий ему фрагмент цифрового сигнала представлены на рисунке 8.1.

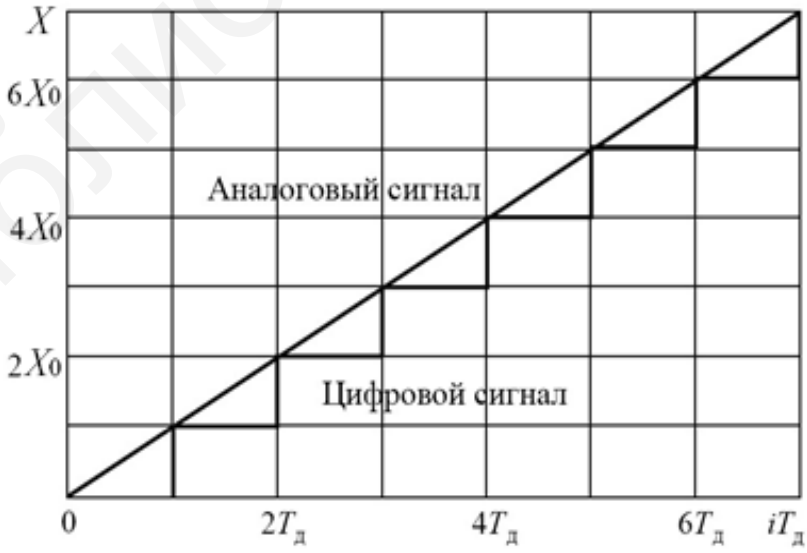


Рисунок 8.1 – Аналоговый и цифровой сигналы

Процесс взятия выборок сопровождается появлением ошибок дискретизации по уровню и по времени. Из аналогового сигнала (рисунок 8.1) видно, что существует разница между цифровым и аналоговым сигналом. Величина этой разницы лежит в пределах от 0 до X_0 и является ошибкой аналого-цифрового преобразования. Сигнал ошибки представлен на рисунке 8.2.

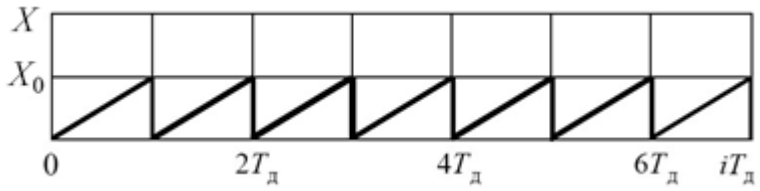


Рисунок 8.2 – Сигнал ошибки квантования по уровню аналого-цифрового преобразования

Таким образом, оцифровывание можно представить как добавление к входному сигналу некоторого пилообразного сигнала с амплитудой, равной кванту амплитуды X_0, и частотой, равной частоте дискретизации $f_д$.

Относительная ошибка аналого-цифрового преобразования находится как отношение кванта амплитуды к диапазону входных сигналов и обратно пропорциональна разрядности АЦП:

$$\varepsilon_{пр} = \frac{X_0}{X_{max} - X_{min}} = \frac{1}{2^n}. \quad (8.8)$$

Например, если необходимо оцифровывать напряжение изменяющееся в диапазоне от 0 до 10 В с абсолютной ошибкой 10 мВ, то относительная ошибка $\varepsilon_{пр} = 10^{-3}$, или 0,1 %, АЦП должно иметь разрядность n = 10, так как 2^{10} = 1024. При этом надо помнить, что если реальный входной сигнал будет меняться не в указанных пределах, а в интервале 2,5–7,5 В, то это приведет к увеличению относительной ошибки вдвое и уменьшению эффективной разрядности до 9.

Операция аналого-цифрового преобразования проходит за конечный интервал по времени, величина которого может меняться в пределах $T_д \pm \Delta t$, что приводит к появлению дополнительной ошибки дискретизации по уровню (рисунок 8.3).

Из ошибки дискретизации по уровню (см. рисунок 8.3) видно, что ошибка определения уровня, связанная с неопределенностью времени проведения замера Δt, зависит от скорости изменения сигнала. Полная ошибка дискретизации по уровню ΔX будет являться суммой перечисленных ошибок.

Если выборки на оси времени расположены неравномерно, то это случайные (или псевдослучайные) выборки. При случайном взятии выборок информация о форме теряется, и по ним можно определить только плотность распределения вероятностей, что означает возможность определения среднеквадратичного (действующего) или пикового (амплитудного) значения

сигнала, а также диапазона принимаемых сигналом значений и т. п. Определить форму сигнала или его спектр невозможно.

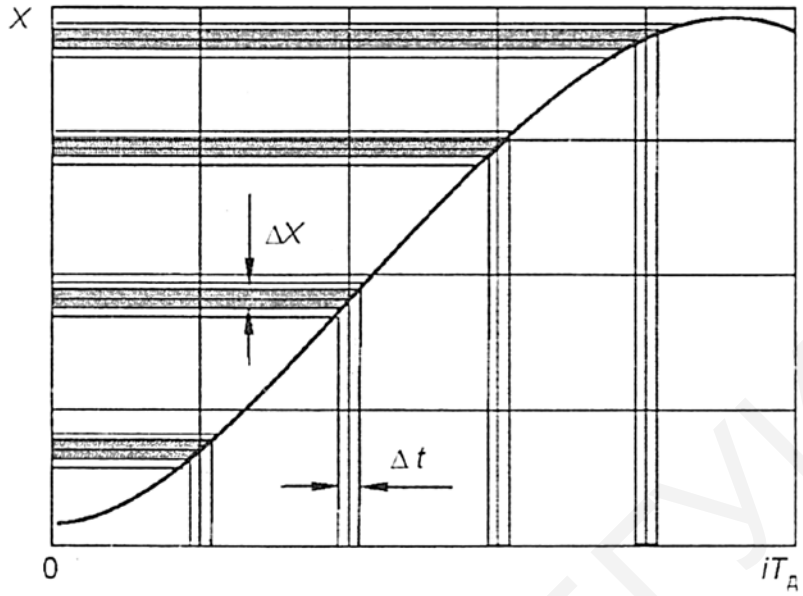


Рисунок 8.3 – Ошибки дискретизации по уровню

Если выборки распределены по оси времени равномерно, то говорят о взятии когерентных выборок. При этом возникает вопрос: сколько выборок необходимо взять для того, чтобы иметь возможность точно описать сигнал? В исследование этого вопроса большой вклад внесли ученые Котельников, Найквист и Шеннон, которые независимо друг от друга сделали вывод, что *для восстановления (без ошибок) исходного сигнала по его выборочным значениям, взятым через равные промежутки времени, частота взятия выборок (частота дискретизации)* $f_д$ *должна более чем вдвое превосходить частоту* $f_{\max}$ *самой высокочастотной составляющей, имеющейся в непрерывном входном сигнале:*

$$f_{\max} \leq f_д/2. \tag{8.9}$$

Здесь под максимальной частотой $f_{\max}$ понимается не только частота полезного сигнала, но и частота шума.

Для того чтобы представить себе, что произойдет, если условие Котельникова – Найквиста – Шеннона не выполнится, рассмотрим результат перемножения двух гармонических сигналов:

$$u_1 u_2 = (a_1 \cos 2\pi f_1 t)\,(a_2 \cos 2\pi f_2 t) =$$
$$= \frac{a_1 a_2}{2} \cos 2\pi (f_1 + f_2) t + \frac{a_1 a_2}{2} \cos 2\pi (f_1 - f_2) t. \tag{8.10}$$

Как следует из выражения (8.10), в спектре появляются суммарные и разностные частоты. Оцифрованный сигнал можно представить себе как произведение аналогового входного сигнала $U(t)$ и сигнала $S(t)$, представляющего собой последовательность равноотстоящих единичных δ-импульсов с частотой следования, равной $f_д$. В результате спектр исходного сигнала претерпевает серьезные изменения (рисунок 8 4).

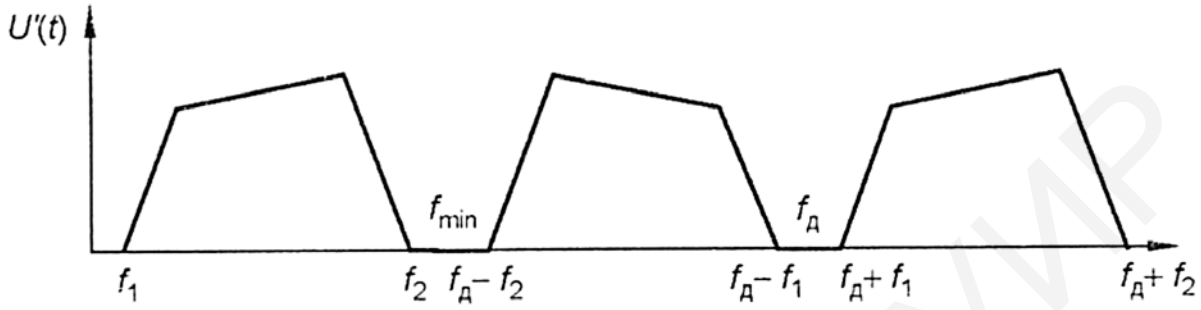


Рисунок 8.4 – Спектр оцифрованного сигнала

Спектр исходного сигнала ограничен частотами f_1 и f_2, но, как видно из выражения (8.3), в результате перемножения двух частот в результирующем спектре появляются суммарные и разностные частоты, что приводит к появлению прямой копии спектра сигнала, ограниченной частотами $f_д + f_1$ и $f_д + f_2$, и так называемой зеркальной копии спектра, ограниченной частотами $f_д - f_1$ и $f_д - f_2$.

Пока спектр исходного сигнала, ограниченный частотами f_1 и f_2, не превышает половины частоты дискретизации, его легко отфильтровать от спектров-копий, но, если частота f_2 превысит частоту f_{max}, т. е. не выполнится условие Котельникова – Найквиста – Шеннона, исходный и зеркальный спектр начнут **наползать** друг на друга и станут неразделимы. Это явление называется **наложением спектров**, или **элайсингом**.

Причиной появления элайсинга может быть как неправильно выбранная частота дискретизации, так и наличие помех.

Бороться с элайсингом можно увеличением частоты дискретизации или ограничением спектра сигнала при помощи фильтров.

Следящий АЦП. Следящий АЦП получается добавлением к ЦАП компаратора и счетчика (рисунок 8.5).

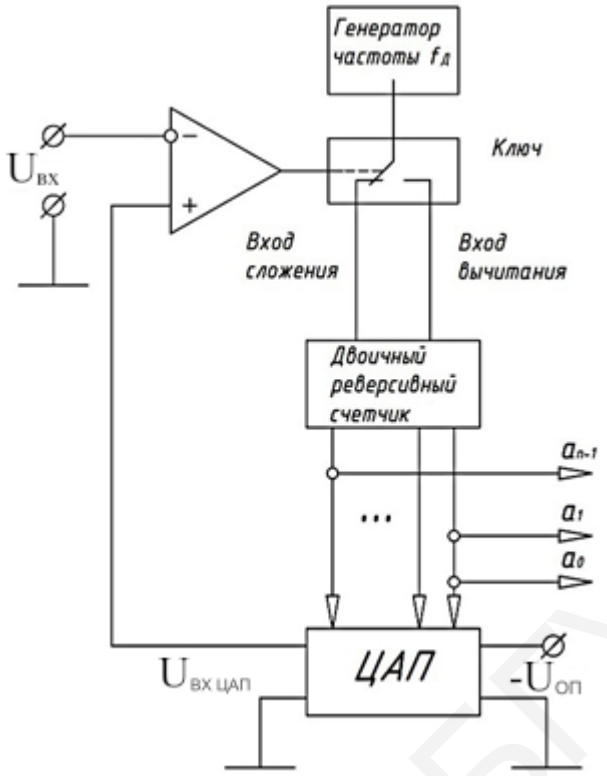


Рисунок 8.5 – Схема следящего АЦП

Компаратор сравнивает входное напряжение с напряжением на выходе ЦАП и управляет ключом, подающим тактовую частоту либо на вход сложения, либо на вход вычитания двоичного реверсивного счетчика, состояние которого является выходом АЦП. Этот же код подается на вход ЦАП. Следовательно, напряжение $U_{\text{вых ЦАП}}$ пропорционально выходному коду счетчика и напряжению $U_{\text{оп}}$, а приход каждого импульса с генератора частоты приводит к увеличению или уменьшению выходного кода счетчика.

Принцип действия следящего АЦП поясняет временная зависимость напряжений на выходе ЦАП от входного напряжения (рисунок 8.6).

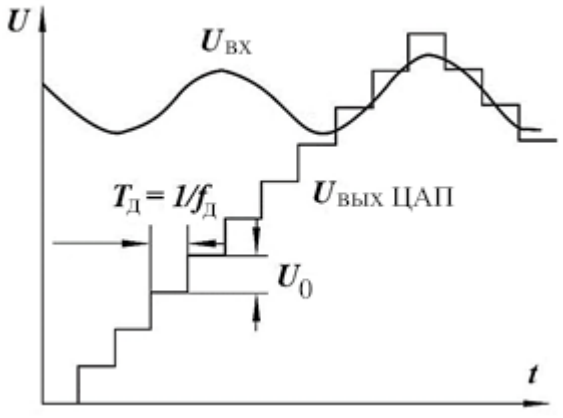


Рисунок 8.6 – Временная зависимость напряжений $U_{\text{вх}}$ и $U_{\text{вых ЦАП}}$

Предположим, что в момент включения АЦП двоичный счетчик устанавливается в нулевое состояние, поэтому напряжение $U_{\text{вых ЦАП}}$ на выходе ЦАП будет равно нулю. Выходной сигнал компаратора переключит счетчик в режим сложения, и выходной код счетчика увеличится с частотой импульсов генератора. Напряжение $U_{\text{вых ЦАП}}$ будет нарастать «ступеньками» (см. рисунок 8.6). Ширина этих «ступенек» по времени равна периоду следования импульсов $T_{\text{д}}$, поступающих от генератора. В некоторой точке напряжение $U_{\text{вых ЦАП}}$ превысит входное напряжение $U_{\text{вх}}$. Тогда выходной сигнал компаратора изменится и счетчик начнет считать в сторону убывания. Поэтому $U_{\text{вых ЦАП}}$ уменьшится, ключ переключится и т. д. В результате напряжение $U_{\text{вых ЦАП}}$ будет пошагово изменяться, приближаясь к значению $U_{\text{вх}}$.

Цифровой сигнал на выходе $D = a_{n-1}a_{n-2} \dots a_1a_0$ равен данным счетчика, которые определяют значение двоичных сигналов на входах ЦАП.

Следящий АЦП способен относительно быстро следить за малыми изменениями во входном сигнале. Если, однако, во входном сигнале имеются большие по величине изменения напряжения, то АЦП уже не сможет отследить его немедленно, так как он может приближаться к новому значению $U_{\text{вх}}$ лишь постепенно, проходя через большое число «ступенек» небольшой постоянной величины (по лестничной функции). Время, необходимое для достижения нового значения $U_{\text{вых ЦАП}}$, зависит не только от скачка $\Delta U_{\text{вх}}$ входного напряжения, но также от кванта амплитуды U_0 и частоты f_0 генератора импульсов. Время преобразования в этом случае определяется по формуле

$$f_{\text{пр}} = \frac{\Delta U_{\text{вх}}}{f_0 U_0} = \frac{2^n \Delta U_{\text{вх}}}{f_0 (U_{\max} - U_{\min})}. \quad (8.11)$$

АЦП последовательного приближения. В основе работы данного АЦП лежит последовательное поразрядное сравнение аналогового входного сигнала $U_{\text{вх}}$ с опорным напряжением $U_{\text{оп}}$.

Входной сигнал, величина которого зафиксирована устройством выборки-хранения, поступает на вход первого преобразователя (рисунок. 8.7), а на второй вход поступает половина опорного напряжения $U_{\text{оп}}/2$.

Если $U'_{\text{вх}} \geq U_{\text{оп}}/2$, то старший разряд АЦП $a_{n-1} = 1$, а на следующий преобразователь поступает напряжение, которое определяется по формуле

$$U'_{\text{вх}} - \left(\frac{U_{\text{оп}}}{2}\right). \quad (8.12)$$

Если $U'_{\text{вх}} < U_{\text{оп}}/2$, то $a_{n-1} = 0$, а для последующего сравнения используется непосредственно напряжение $U'_{\text{вх}}$. На втором преобразователе остаток напряжения $U'_{\text{вх}}$ сравнивается с напряжением $U_{\text{оп}}/4$. Так последовательным сравнением формируются все разряды искомого числа $D = a_{n-1}a_{n-2} \dots a_1a_0$.

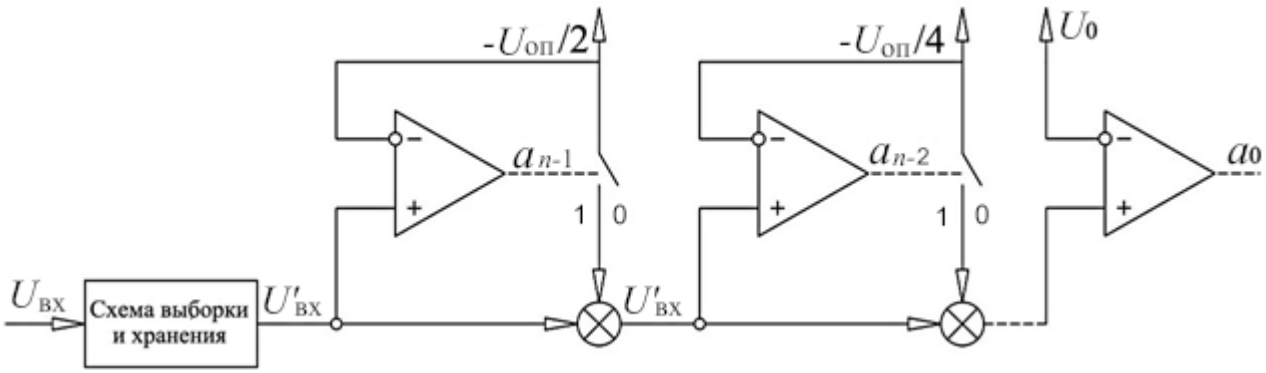


Рисунок 8.7 – АЦП последовательного приближения

Для АЦП последовательного приближения важным является сохранение значения входного сигнала в течение всего цикла преобразования, так как в противном случае будут возникать сбои в его работе. Время преобразования не зависит от скорости изменения входного напряжения и равно произведению времени преобразования одного разряда на их количество.

АЦП параллельного преобразования. В отличие от АЦП последовательного приближения АЦП параллельного преобразования является самым быстродействующим из всех видов АЦП. Его схема приведена на рисунке 8.8.

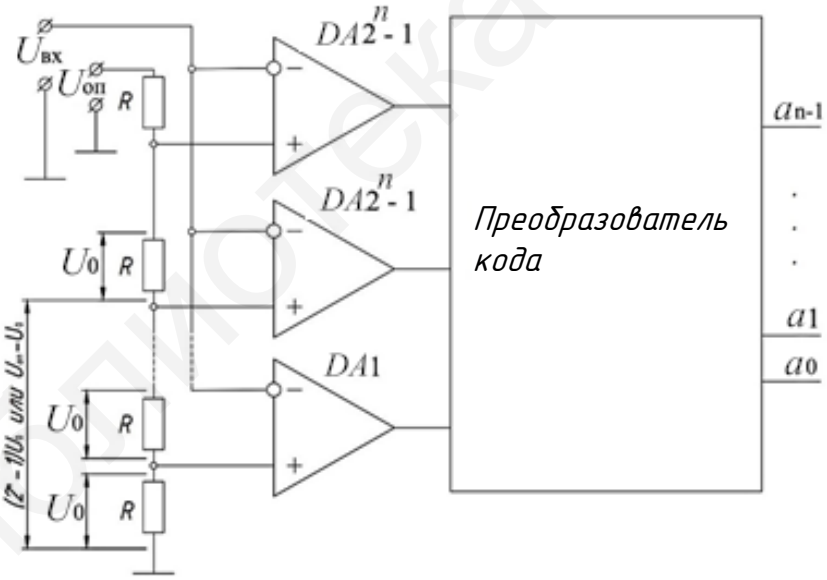


Рисунок 8.8 – Схема АЦП параллельного преобразования

Входное напряжение $U_{\text{вх}}$ сравнивается одновременно с большим числом различных опорных напряжений, полученных в результате деления напряжения $U_{\text{оп}}$ делителем, состоящим из 2^n резисторов одинакового номинала R. В результате на каждом резисторе падает напряжение U_0, а на неинвертирующий вход i-го компаратора поступает напряжение, равное произведению U_0 и номера преобразователя. Для n-разрядного преобразования в таком преобразователе требуется $2^n - 1$ компараторов и столько же получается выходных сигналов.

Сигма-дельта АЦП. Принцип сигма-дельта ($\sum\Delta$) модуляции известен с начала 1960-х годов, но до начала 1990-х годов почти не использовался (рисунок 8.9).

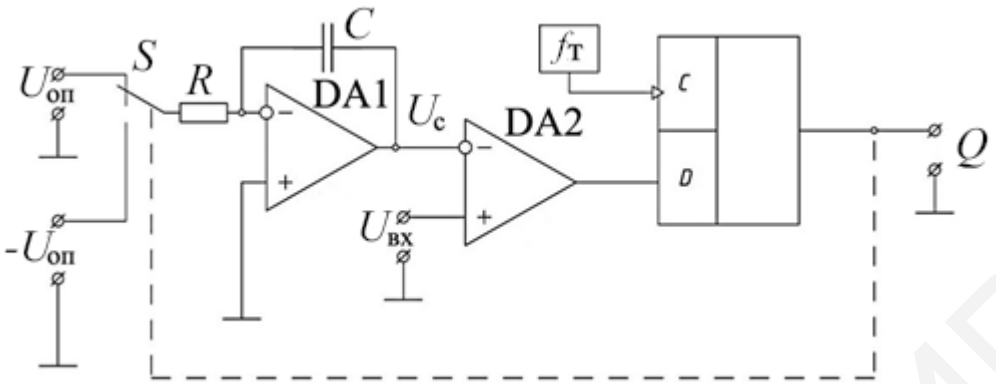


Рисунок 8.9 – Сигма-дельта АЦП

Входное напряжение $U_{вх}$ подается на компаратор DA2, который сравнивает его с напряжением U_c, поступающим с интегратора, выполненного на операционном усилителе DA1. На вход интегратора через ключ *S* подается либо $U_{оп}$, либо минус $U_{оп}$. Работой ключа управляет *D*-триггер, на счетный вход *C* которого подается тактовая частота f_T, а на вход *D* – напряжение с компаратора. Тактовая частота f_T должна быть выше частоты дискретизации $f_д$:

$$f_T = 2^n f_Д, \tag{8.13}$$

где *n* – разрядность преобразователя.

Диаграмма работы сигма-дельта АЦП приведена на рисунке 8.10.

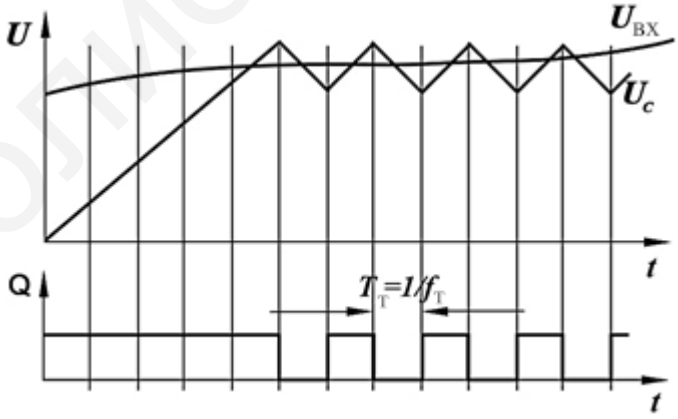


Рисунок 8.10 – Временная диаграмма работы сигма-дельта АЦП

Выходным сигналом сигма-дельта АЦП будет количество единиц на выходе *Q*, подсчитанных за период дискретизации $T_д$. Для уменьшения тактовой частоты модуляторы можно каскадировать.

Современные сигма-дельта АЦП обладают высокой точностью преобразования и достаточно высокой скоростью преобразования. Например, АЦП *AD*7762 фирмы «*Analog devises*» имеет разрядность n = 24 при тактовой частоте $f_{\text{Т}}$ = 40 МГц и частоте дискретизации $f_{\text{д}}$ = 650 кГц.

8.3 Порядок выполнения практической работы

1. Изучить схему АПЦ (рисунок 8.11).

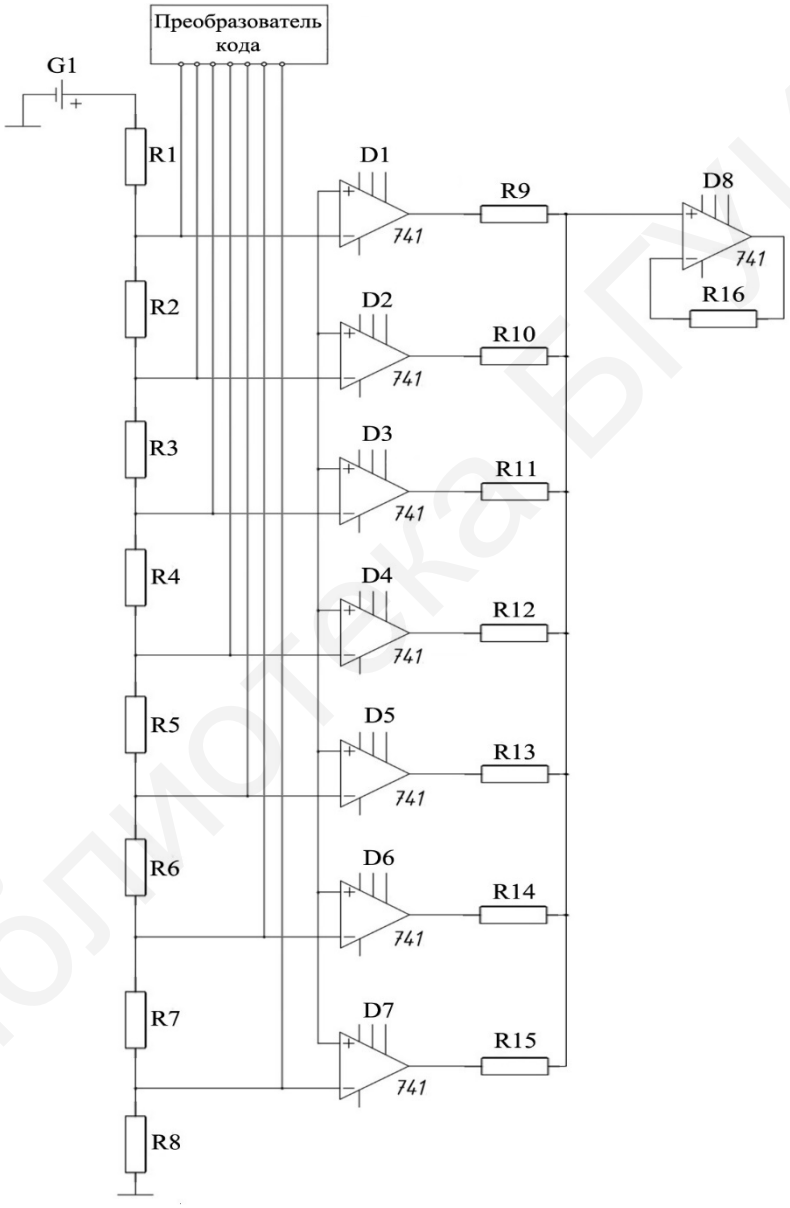


Рисунок 8.11 – Схема АЦП

2. Выполнить синтез изученной схемы АЦП в среде схемотехнического моделирования *MultiSim*.

8.4 Содержание отчета

1. Цель работы.
2. Схема и результаты исследования.
3. Вывод.
4. Ответы на контрольные вопросы.

8.5 Контрольные вопросы

1. Что такое АЦП?
2. Какие существуют схемы построения АЦП?
3. Какие существуют ошибки в процессе преобразования?
4. Что собой представляют дискретный и непрерывный сигналы?
5. Что такое элайсинг?
6. В чем состоит суть теоремы Котельникова?

9 Практическая работа №9

ЦИФРОАНАЛОГОВОЕ ПРЕОБРАЗОВАНИЕ СИГНАЛОВ В МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВАХ

9.1 Цель работы

Изучение основных принципов работы, исследование структуры построения и ознакомление с основными техническими параметрами ЦАП.

9.2 Теоретические сведения

Цифроаналоговый преобразователь (ЦАП) применяется в качестве узлов обратной связи аналого-цифровых преобразователей или для формирования выходных аналоговых сигналов цифровых измерительных или вычислительных устройств. Задача ЦАП – это преобразование двоичного числа $D = a_{n-1}a_{n-2} \dots a_1a_0$ в выходное напряжение или ток, пропорциональные весовым коэффициентам разрядов двоичной системы счисления (...8, 4, 2, 1).

В большинстве случаев входное число D – целое без знака, поэтому цифро-аналоговое преобразование определяется выражением

$$U_{\text{вых ЦАП}} = U_0 D = U_0 \sum_{i=0}^{n-1} a_i 2^i. \quad (9.1)$$

Исходя из выражения (9.1) можно сделать вывод, что для реализации ЦАП необходимо суммировать напряжения, пропорциональные весовым коэффициентам соответствующих разрядов. Для этого необходим многовходовый сумматор, выполненный на операционном усилителе, и переключатели. Такое устройство представлено на рисунке 9.1.

Цифровой вход D состоит из всех битов $a_i, i = 0,1, \dots n-1$. Если $a_i = 1$, то соответствующий переключатель a подключен к отрицательному опорному напряжению – $U_{\text{оп}}$, если же $a_i = 0$, то переключатель подключен к проводнику с нулевым потенциалом.

Коэффициент усиления современного операционного усилителя без обратной связи $k_u = 10^6 \dots 10^8$, разность потенциалов между входами определяется выражением

$$U_+ - U_- = \frac{U_{\text{вых}}}{k_u} \approx 0. \quad (9.2)$$

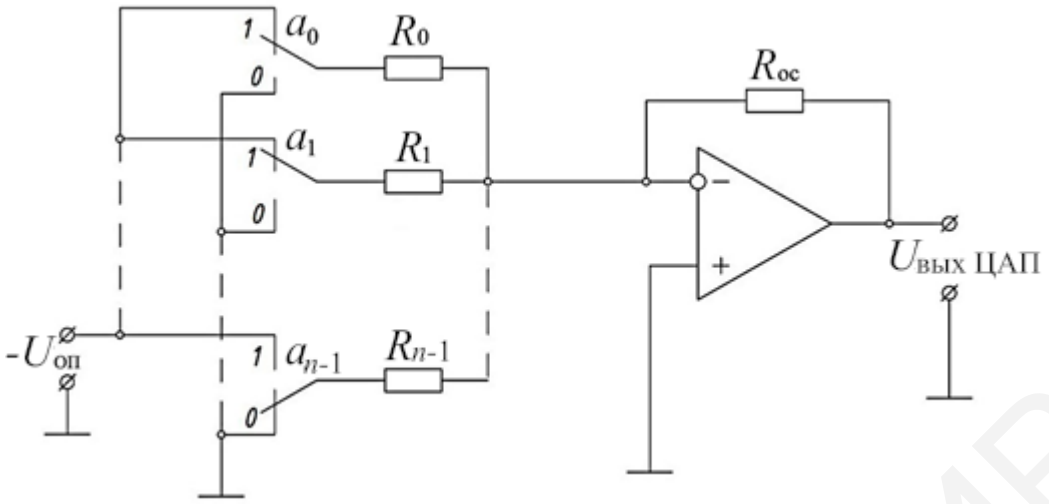


Рисунок 9.1 – Реализация ЦАП

Выводы резисторов R_i, обращенные к операционному усилителю, близки к нулевому потенциалу, и если a_i = 0, то ток через соответствующий этому разряду резистор не протекает. Следовательно, ток, в котором сопротивление каждого резистора R_i соответствует весу цифры или бита входной цепи усилителя, равен сумме токов, протекающих по всем резисторам R_i, для которых a_i = 1:

$$I_{вх} = \sum_{i=0}^{n-1} \frac{a_i U_{оп}}{R_i} = \frac{U_{оп}}{2^n} \sum_{i=0}^{n-1} a_i \frac{1}{R_i} = U_0 \sum_{i=0}^{n-1} a_i 2^i \frac{1}{R}. \qquad (9.3)$$

Напряжение на выходе ЦАП равно произведению сопротивления обратной связи и входного тока:

$$U_{вых\ ЦАП} = I_{вх} I_{о.с} = R_{о.с} U_0 \frac{1}{2^n R} \sum_{i=0}^{n-1} a_i 2^i. \qquad (9.4)$$

Если в выражении (9.4) сопротивление $R_{о.с} = 2^n R$, то его подстановка приводит выражение (9.4) к виду (9.1). Следовательно, сопротивление резисторов определяется формулой

$$R_i = \frac{R}{2^i}. \qquad (9.5)$$

Основной недостаток такого ЦАП состоит в том, что соотношение между наибольшим сопротивлением (R_0) и наименьшим (R_{n-1}) равно $\frac{R_0}{R_{n-1}} = 2^n$ и уже при n = 10 отношение указанных сопротивлений равно 1024, что соответствует

погрешности преобразования $\varepsilon_{ЦАП} = 0{,}1\ \%$. Если относительная ошибка определения величины резистора R_{n-1} равна

$$\varepsilon_{R_{n-1}} = \left(\frac{\Delta R_{n-1}}{R_{n-1}}\right) \cdot 100 = 10\ \%, \tag{9.6}$$

то относительная ошибка R_0 определяется выражением

$$\varepsilon_{R_0} = \frac{10}{1024} \approx 0{,}01\ \%. \tag{9.7}$$

Такой относительной ошибкой обладают только прецизионные резисторы, служащие эталоном сопротивления. По этой причине подобные ЦАП редко изготавливают с разрядностью $n > 10$.

На рисунке 9.2 приведена схема ЦАП, в которой используются резисторы только двух различных номиналов.

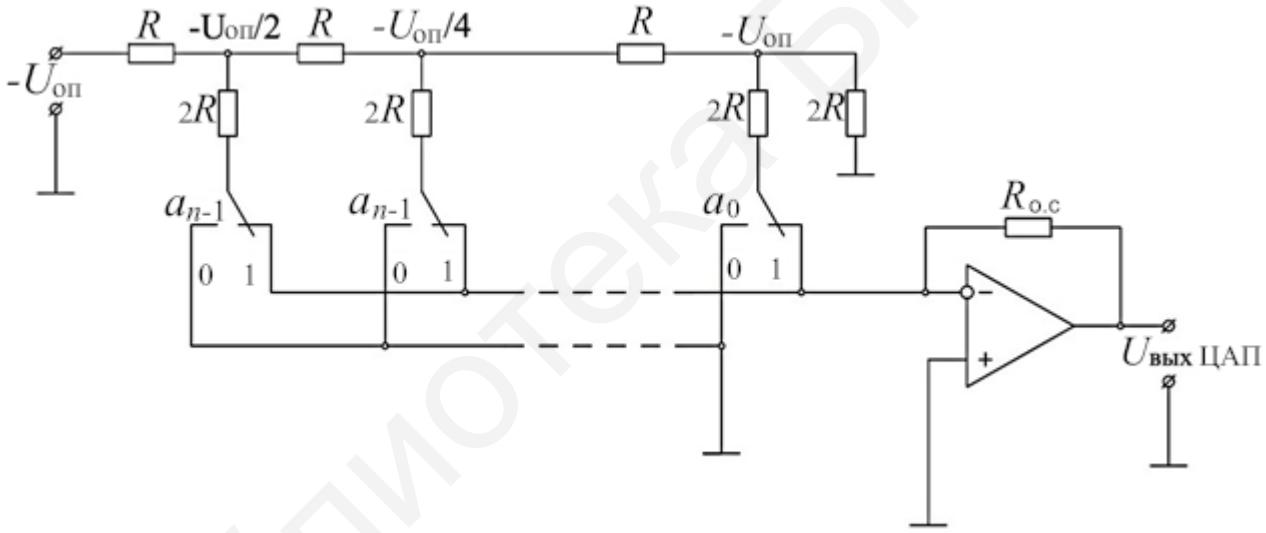


Рисунок 9.2 – Цифроаналоговое преобразование с использованием резисторной матрицы

Схема состоит из резисторной матрицы и усилителя, преобразующего ток в напряжение. Потенциал в точке переключения всегда равен потенциалу земли (независимо от того, установлено a_i = 0 или a_i = 1). Два правых резистора представляют собой соединенные параллельно два резистора номиналом 2R, их общее сопротивление равно R. Последовательно с ними включен резистор R, и их суммарное сопротивление равно 2R и т. д. до входа. Таким образом, опорное напряжение нагружено на сопротивление, номинал которого составляет 2R. Такая резисторная матрица представляет собой n делителей напряжения на 2. Следовательно, напряжения в узлах равны соответственно $-U_{оп}/2$, $-U_{оп}/4$, ..., $-U_{оп}/2^n = -U_0$. Токи, протекающие в каналах a_i, будут отличаться от соседних

ровно в два раза, а положение переключателя a_i = 1 будет добавлять соответствующий ток к сумме токов, протекающих во входных цепях операционного усилителя. Проведя несложные преобразования, можно прийти к выражению (9.1).

Так как здесь используются только два номинала резисторов, у такого ЦАП точность может быть гораздо большей, чем в предыдущем случае. Разрядность ЦАП на резисторной матрице может достигать n = 16.

9.3 Порядок выполнения практической работы

1. Изучить схему ЦАП (рисунок 9.3).

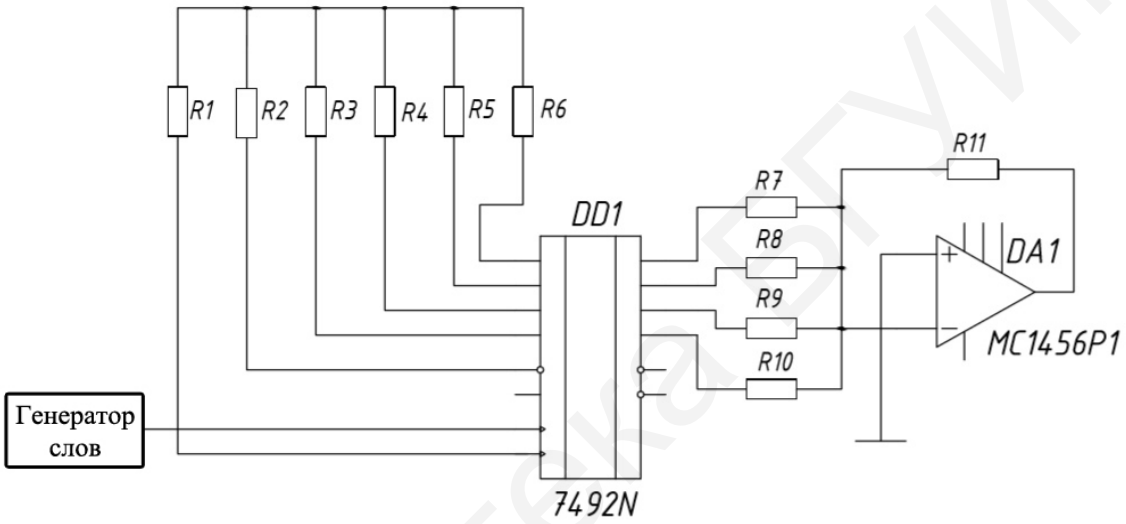


Рисунок 9.3 – Схема ЦАП

2. Выполнить синтез изученной схемы ЦАП в среде схемотехнического моделирования *MultiSim*.

9.4 Содержание отчета

1. Цель работы.
2. Схема и результаты исследования.
3. Вывод.
4. Ответы на контрольные вопросы.

9.5 Контрольные вопросы

1. Что такое ЦАП?
2. Нарисуйте схему ЦАП.
3. Каково основное назначение ЦАП?
4. Изобразите непрерывный и дискретный сигналы.

10 Лабораторная работа №1

ИССЛЕДОВАНИЕ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

10.1 Цель работы

Изучение базовых логических функций, исследование функционирования основных логических элементов.

10.2 Теоретические сведения

Все цифровые устройства построены на элементах, которые выполняют те или иные логические операции. Для анализа и синтеза цифровых устройств используется аппарат алгебры логики. В общем случае цифровые устройства разделяются на два типа:

1) комбинационные;
2) последовательные.

В комбинационных устройствах выходной сигнал в любой момент времени зависит только от значений входных сигналов в тот же момент времени.

В последовательных устройствах выходной сигнал в любой момент времени зависит как от значений входных сигналов в данный момент времени, так и от значений входных сигналов в предыдущие моменты времени. Для этого данные устройства имеют память. В последовательных устройствах комбинационные устройства являются составной частью, поэтому комбинационные устройства изучаются первыми.

Рассмотрим блок-схему комбинационного устройства (рисунок 10.1).

Рисунок 10.1 – Блок-схема комбинационного устройства

Входные и выходные сигналы комбинационного устройства могут принимать только два логических значения – 0 или 1, т. е. $x_i \in 0,1, y_i \in 0,1$. Любая совокупность входных сигналов может быть представлена вектором, которая вычисляется по формуле

$$X = \{x_1, x_2, \dots, x_n\} \quad (10.1)$$

и называется входным набором.

Очевидно, что существует 2^n различных входных наборов. Сопоставим каждому входному набору определенное значение выходного сигнала:

$$Y = (x_1, x_2, \dots, x_n). \quad (10.2)$$

Тогда работа комбинационной схемы (устройства) может быть описана с помощью функции, отображающей множество входных наборов в значение выходной переменной *Y*.

Определение. Функцией алгебры логики (ФАЛ) $f(x_1, x_2, \dots, x_n)$ называется функция, дающая однозначное отображение множества векторов *X* в переменную *Y*.

Так как число различных входных наборов равно 2^n, то любая ФАЛ может быть задана в виде таблицы с 2^n строками (таблица 10.1).

Таблица 10.1 – Функции алгебры логики

x_1	x_2	x_{n-1}	x_n	$f(x_1, x_2, \dots, x_n)$
0	0	0	0	α_1
0	0	0	1	α_2
0	0	1	0	α_3
…	…	…	…	…
1	1	0	0	α_{2^n-3}
1	1	0	1	α_{2^n-2}
1	1	1	0	α_{2^n-1}
1	1	1	1	α_{2^n}

В левой части таблицы перечислены все возможные входные наборы, а в правой записаны значения функции на этих наборах.

Каждая ФАЛ представляет собой двоичный набор длиной 2^n, а число возможных таких наборов равно 2^{2^n}, поэтому справедливо утверждение, что число различных функций алгебры логики, зависящих от *n* аргументов, конечно и равно 2^{2^n}.

Рассмотрим несколько примеров. Пусть $n = 1$, тогда число функций алгебры логики $2^{2^n} = 4$. Эти функции приведены в таблице 10.2.

Таблица 10.2 – Функции алгебры логики одного аргумента

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Функции f_0 и f_3 − логические константы (константа нуля и константа единицы). Функция f_1 называется функцией тождества или просто тождеством, так как $f_1 = x$, а функция f_2 называется функцией отрицания или просто отрицанием, $f_3 = \bar{x}$ (читается «не *x*»), функция НЕ.

Для $n = 2$ существует $2^{2^n} = 2^{2^2} = 16$ функций алгебры логики, которые представлены в таблице 10.3.

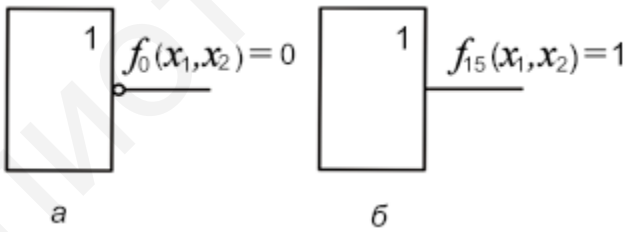
Таблица 10.3 – Функции алгебры логики двух аргументов

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Операц. символ		–	•	←	–	←	–	⊕	+	↓	⊙	–	→	–	→	↑	–

Рассмотрим эти функции.

Функции $f_0(x_1, x_2)$ и $f_{15}(x_1, x_2)$ – логические константы. Функция $f_0(x_1, x_2) = 0 = \overline{f_{15}}(x_1, x_2)$ – константа нуль (функция нуль). Технически f_0 реализуется генератором нуля в соответствии с рисунком 10.2, *а*.

Функция $f_{15}(x_1, x_2) = 0 = \overline{f_0}(x_1, x_2)$ – константа единицы (функция единицы). Технически реализуется генератором единицы в соответствии с рисунком 10.2, *б*.



а – генератор нуля; *б* – генератор единицы

Рисунок 10.2 – Условное обозначение генераторов логических констант

Будем продолжать рассматривать функции (см. таблицу 10.3) по парам, так как по отношению к любой функции вторая функция в паре является инверсной.

Функция $f_1(x_1, x_2)$ – конъюнкция, логическое умножение, функция И, вычисляется формуле

$$f_1(x_1, x_2) = x_1 \& x_2 = x_1 \cdot x_2 = x_1 x_2 = \overline{f_{14}}(x_1, x_2). \qquad (10.3)$$

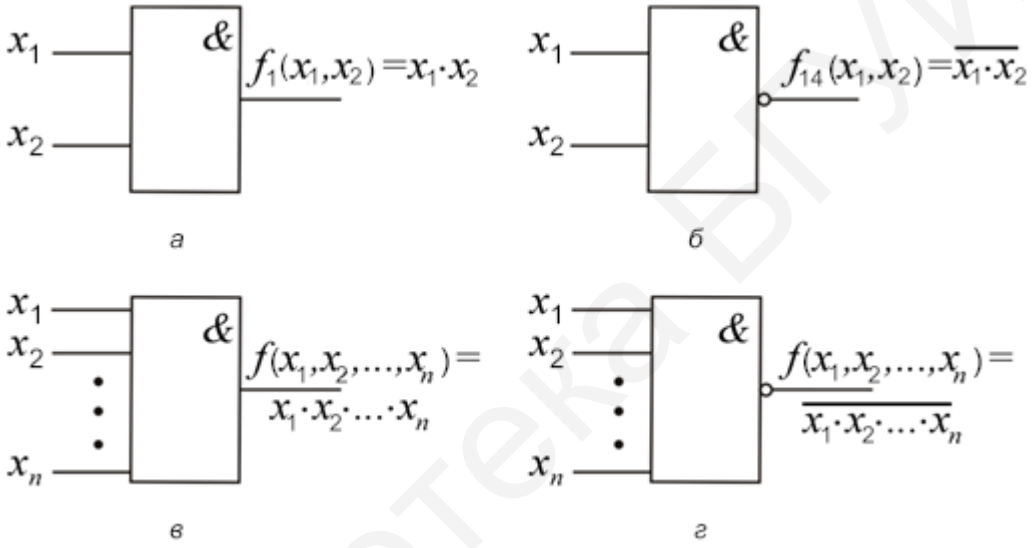
Технически эта функция реализуется логическим элементом И в соответствии с рисунком 10.3, *а*.

Функция $f_{14}(x_1, x_2)$ – функция Шеффера, функция И-НЕ, определяется выражением

$$f_{14}(x_1, x_2) = \overline{x_1 x_2} = \bar{f}_1(x_1, x_2). \quad (10.4)$$

Технически данная функция реализуется логическим элементом Шеффера, элементом И-НЕ в соответствии с рисунком 10.3, *б*.

Функции И и И-НЕ могут быть, как и соответствующие им логические элементы, с произвольным числом переменных входов в соответствии с рисунком 10.3, *в, г*.



а – логический элемент И; *б* – логический элемент И-НЕ;
в, *г* – логические элементы И и И-НЕ

Рисунок 10.3 – Условное обозначение логических элементов И и И-НЕ

Функция f_2 – запрет первого аргумента вычисляется с помощью выражения

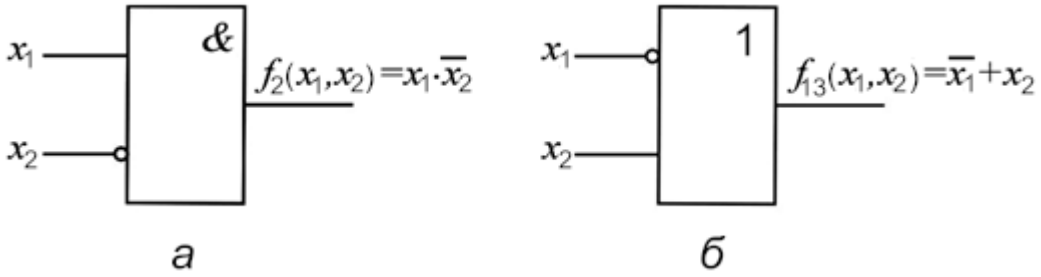
$$f_2(x_1, x_2) = x_1 \leftarrow x_2 = x_1 \cdot \overline{x_2} = \overline{f_{13}}(x_1, x_2). \quad (10.5)$$

Технически эта функция реализуется элементом запрета в соответствии с рисунком 10.4, *а*.

Функция f_{13} – импликация от первого аргумента ко второму, вычисляется с помощью выражения

$$f_{13}(x_1, x_2) = x_1 \rightarrow x_2 = \overline{x_1} \cdot x_2 = \bar{f}_2(x_1, x_2). \quad (10.6)$$

Технически данная функция реализуется импликатором (рисунок 10.4, *б*).



а – запрет; *б* – импликатор

Рисунок 10.4 – Условное обозначение запрета и импликатора

Функция f_3 – повторение первого аргумента (функция ДА), вычисляется с помощью выражения

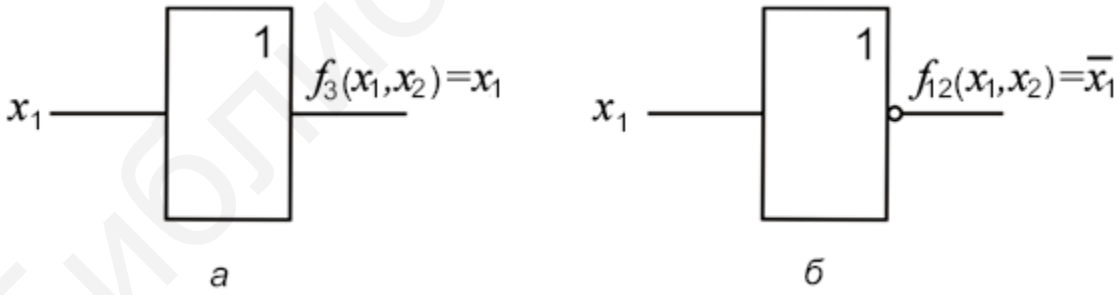
$$f_3(x_1, x_2) = x_1 = \overline{f_{12}}(x_1, x_2). \quad (10.7)$$

Технически данная функция реализуется повторителем (рисунок 10.5, *а*).

Функция f_{12} – отрицание первого аргумента (функция НЕ), вычисляется по формуле

$$f_{12}(x_1, x_2) = \overline{x_1} = \overline{f_3}(x_1, x_2). \quad (10.8)$$

Технически эта функция реализуется инвертором (рисунок 10.5, *б*).



а – повторитель; *б* – инвертор

Рисунок 10.5 – Условные обозначения повторителя и инвертора

Функция f_4 – запрет второго аргумента:

$$f_4(x_1, x_2) = x_2 \leftarrow x_1 = \overline{x_1} \cdot x_2 = \overline{f_{11}}(x_1, x_2). \quad (10.9)$$

Функция f_{11} – импликация от второго аргумента к первому:

$$f_{11}(x_1, x_2)x_2 \to x_1 = x_1 + \bar{x}_2 = f_4(x_1, x_2). \quad (10.10)$$

Функция f_5 – повторение второго аргумента:

$$f_5(x_1, x_2) = x_2 = \bar{f}_{10}(x_1, x_2). \quad (10.11)$$

Функция f_{10} – отрицание второго аргумента:

$$f_{10}(x_1, x_2) = \bar{x}_2 = \bar{f}_5(x_1, x_2). \quad (10.12)$$

Функция f_6 – неравнозначность, Исключающее ИЛИ:

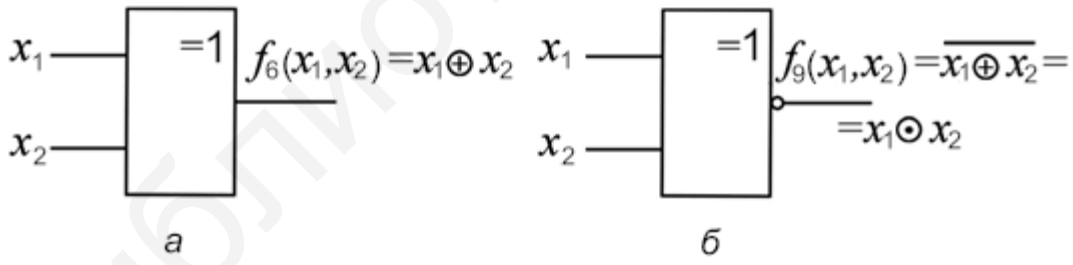
$$f_6(x_1, x_2) = x_1 \oplus x_2 = \overline{x_1}x_2 + x_1\overline{x_2} = \bar{f}_9(x_1, x_2). \quad (10.13)$$

Технически данная функция реализуется логическим элементом Исключающее ИЛИ (рисунок 10.6, *а*).

Функция f_9 – равнозначность, эквивалентность, Исключающее ИЛИ-НЕ:

$$f_9(x_1, x_2) = x_1 \otimes x_2 = x_1x_2 + \overline{x_1x_2} = \overline{x_1 \oplus x_2} = \bar{f}_6(x_1, x_2). \quad (10.14)$$

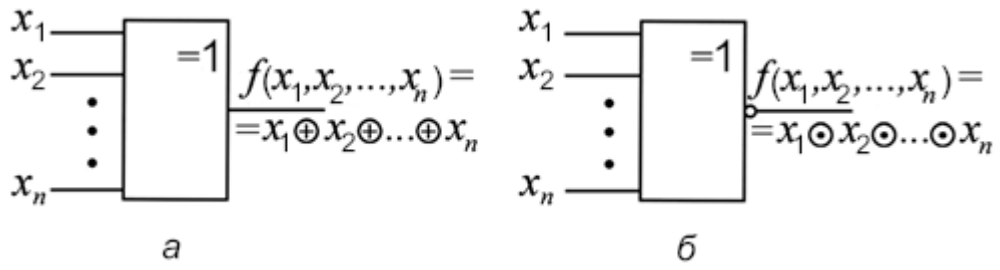
Технически эта функция реализуется элементом равнозначность, Исключающее ИЛИ-НЕ (рисунок 10.6, *б*).



а – Исключающее ИЛИ; *б* – Исключающее ИЛИ-НЕ

Рисунок 10.6 – Логические элементы, реализующие функции неравнозначность и равнозначность с двумя переменными

Функции неравнозначность и равнозначность могут быть, как и соответствующие им логические элементы, с произвольным числом переменных (входов) (рисунок 10.7). Функция неравнозначность равна 1, если число аргументов, равных 1, нечетно.



а – Исключающее ИЛИ; *б* – Исключающее ИЛИ-НЕ

Рисунок 10.7 – Логические элементы, реализующие функции неравнозначность и равнозначность с произвольным числом переменных

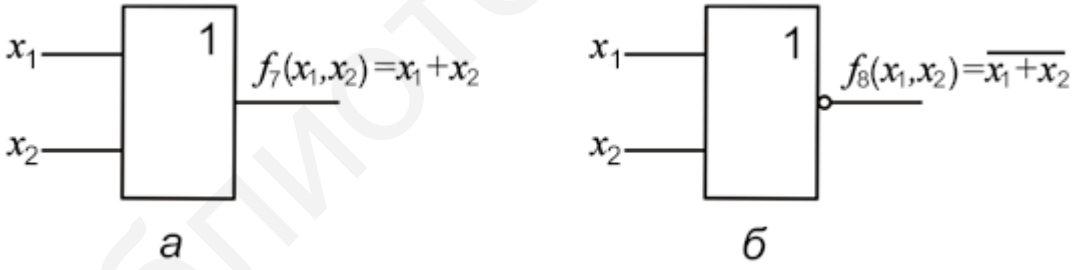
Функция f_7 – дизъюнкция, функция ИЛИ:

$$f_7(x_1, x_2) = x_1 + x_2 = \overline{f_8}(x_1, x_2). \qquad (10.15)$$

Технически функция f_7 реализуется элементом ИЛИ (рисунок 1.8, *а*).
Функция f_8 – функция Пирса или функция Вебба (функция ИЛИ-НЕ):

$$f_8(x_1, x_2) = \overline{x_1 + x_2} = \overline{f_7}(x_1, x_2). \qquad (10.16)$$

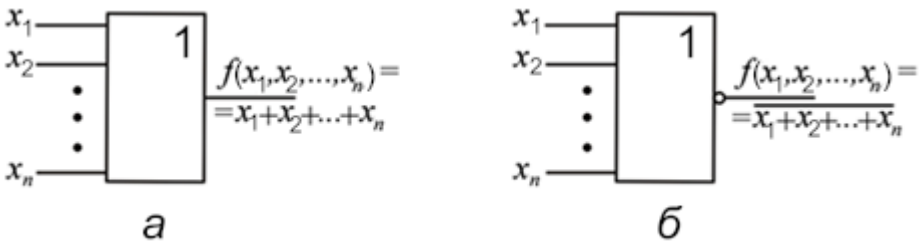
Технически функция f_8 реализуется элементом Пирса или Вебба (элемент ИЛИ-НЕ) (рисунок 10.8, *б*).



а – логический элемент ИЛИ; *б* – логический элемент ИЛИ-НЕ

Рисунок 10.8 – Логические элементы ИЛИ и ИЛИ-НЕ с двумя переменными

Функции ИЛИ и ИЛИ-НЕ могут быть, как и соответствующие им логические элементы, с произвольным числом переменных (входов) (рисунок 10.9).



а – логический элемент ИЛИ; *б* – логический элемент ИЛИ-НЕ

Рисунок 10.9 – Логические элементы ИЛИ и ИЛИ-НЕ
с произвольным числом переменных

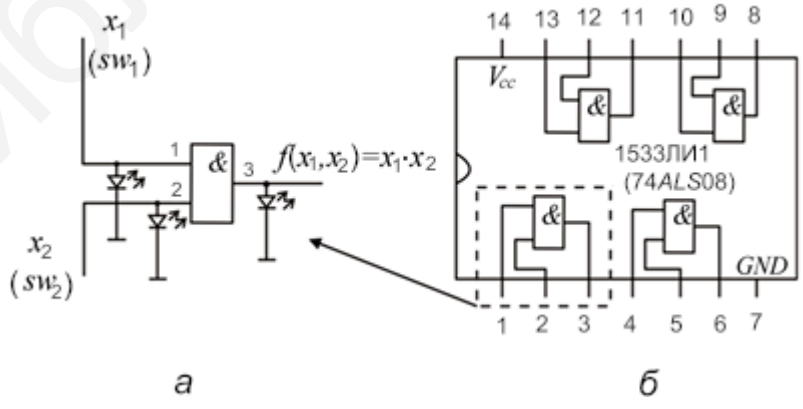
10.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудование и компоненты: универсальная лабораторная установка *IDL*-800, логические элементы: ИС 1533ЛА3 (74*ALS*00) − четыре логических элемента 2И-НЕ, ИС 1533ЛЕ1 (74*ALS*02) − четыре логических элемента 2ИЛИ-НЕ, ИС 1533ЛН1 (74*ALS*04) − шесть логических элементов НЕ, ИС 1533ЛИ1 (74*ALS*08) − четыре логических элемента 2И, ИС 1533ЛЛ1 (74*ALS*32) − четыре логических элемента 2ИЛИ, ИС 1533ЛП5 (74*ALS*86) − четыре двухвходовых логических элемента ИЛИ.

Задание 1. Исследовать логические элементы И:

1. Двухвходовый элемент И. Для этого разместить ИС 1533ЛИ1 на наборной панели *IDL*-800. Вывод 14 ИС соединить с источником питания +5*V*, а вывод 7 – с общей шиной установки. Собрать схему, как показано на рисунке 10.10, *а*.

Структурная схема логического элемента показана на рисунке 10.10, *б*.



а – принципиальная схема; *б* – структурная схема

Рисунок 10.10 – Двухвходовый элемент И на примере ИС 1533ЛИ1

Изменяя состояния входов x_1 и x_2 с помощью переключателей *SW*, заполнить таблицу истинности логического элемента 2И (таблица 10.4).

Таблица 10.4 – Таблица истинности логического элемента 2И

Входы		**Выход**
x_1	x_2	$f(x_1, x_2)$
0	0	
0	1	
1	0	
1	1	

2. Многовходовый элемент И. Для этого, используя двухвходовые элементы И, собрать трехвходовый элемент И, реализующий функцию $f(x_1, x_2, x_3) = (x_1 \cdot x_2) \cdot x_3 = x_1 \cdot x_2 \cdot x_3$, как показано на рисунке 10.11.

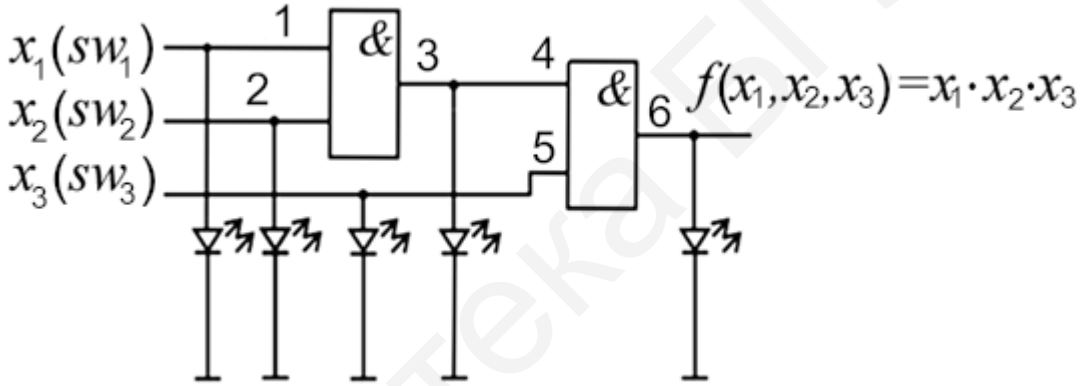


Рисунок 10.11 – Трехвходовый элемент И

Изменяя состояния входов x_1, x_2, x_3, исследовать работу схемы, заполнить таблицу истинности логического элемента 3И (таблица 10.5).

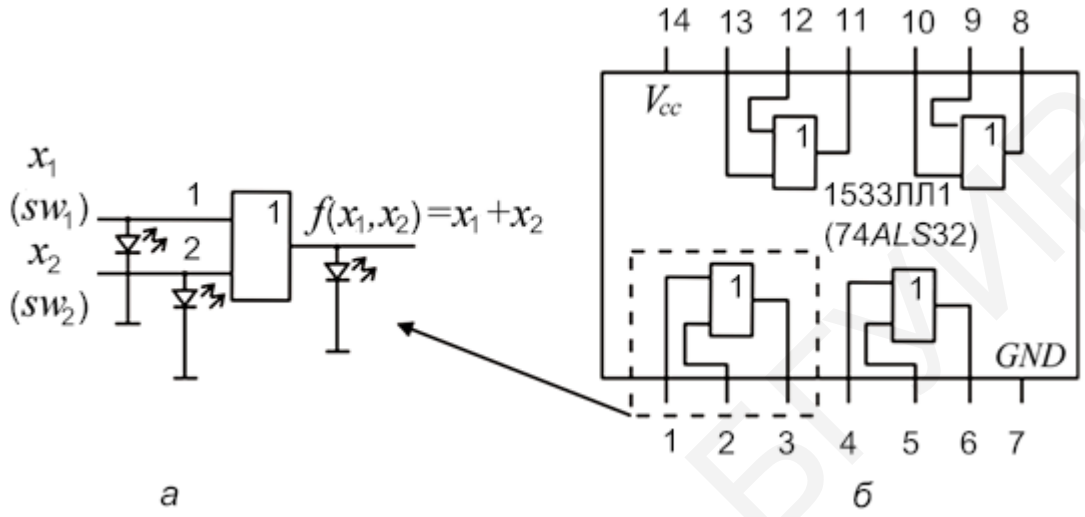
Таблица 10.5 – Таблица истинности логического элемента 3И

Входы			**Выходы**	
x_1	x_2	x_3	$x_1 \cdot x_2$	$x_1 \cdot x_2 \cdot x_3$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Задание 2. Исследовать логические элементы ИЛИ:

1. Двухвходовый элемент ИЛИ. Для этого разместить ИС 1533ЛЛ1 (74*ALS*32) на наборной панели *IDL*-800. Вывод 14 ИС соединить с источником питания +5*V*, а вывод 7 – с общей шиной установки. Собрать схему, как показано на рисунке 10.12, *а*.

Структурная схема логического элемента показана на рисунке 10.12, *б*.



а – принципиальная схема; *б* – структурная схема

Рисунок 10.12 – Двухвходовый элемент ИЛИ на примере ИС 1533ЛЛ1 (74*ALS*32)

Изменяя состояние входов x_1, x_2, исследовать работу двухвходового элемента ИЛИ, заполнить таблицу истинности логического элемента (таблица 10.6).

Таблица 10.6 – Таблица истинности двухвходового элемента ИЛИ

Входы		**Выход**
x_1	x_2	$f(x_1, x_2)$
0	0	
0	1	
1	0	
1	1	

2. Многовходовой элемент ИЛИ. Для этого, используя двухвходовые элементы ИЛИ, собрать трехвходовый элемент ИЛИ, реализующий функцию $f(x_1, x_2, x_3) = (x_1 + x_2) + x_3 = x_1 + x_2 + x_3$, как показано на рисунке 10.13.

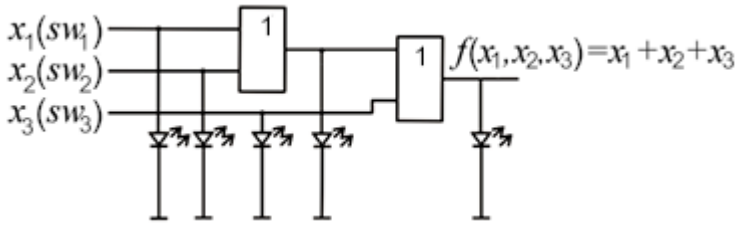


Рисунок 10.13 – Трехвходовый элемент ИЛИ

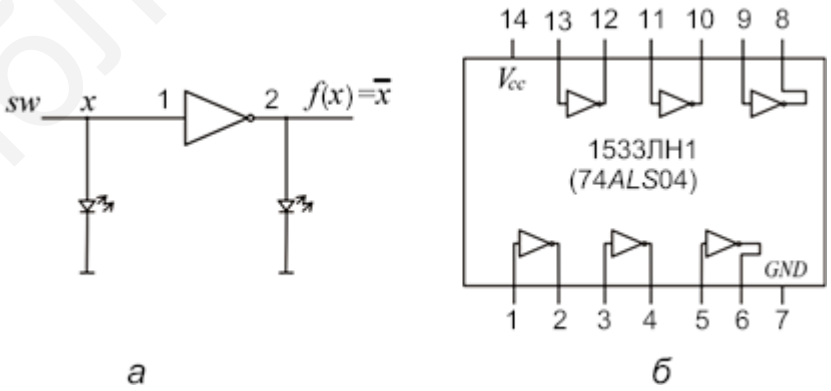
Изменяя состояния входов x_1, x_2, x_3, исследовать работу схемы, заполнить таблицу истинности логического элемента 3ИЛИ (таблица 10.7).

Таблица 10.7 – Таблица истинности логического элемента 3ИЛИ

Входы			**Выходы**	
x_1	x_2	x_3	x_1+x_2	$x_1+x_2+x_3$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Задание 3. Исследовать инвертор. Для этого разместить ИС 1533ЛН1 (74*ALS*04) на наборной панели *IDL*-800. Вывод 14 ИС соединить с источником питания +5*V*, а вывод 7 – с общей шиной установки. Собрать схему, как показано на рисунке 10.14, *а*.

Структурная схема логического элемента показана на рисунке 10.14, *б*.



а – принципиальная схема; *б* – структурная схема

Рисунок 10.14 – Инвертор на примере ИС 1533ЛН1 (74*ALS*04)

Изменяя состояние входа *x*, исследовать работу инвертора, заполнить таблицу истинности инвертора (таблица 10.8).

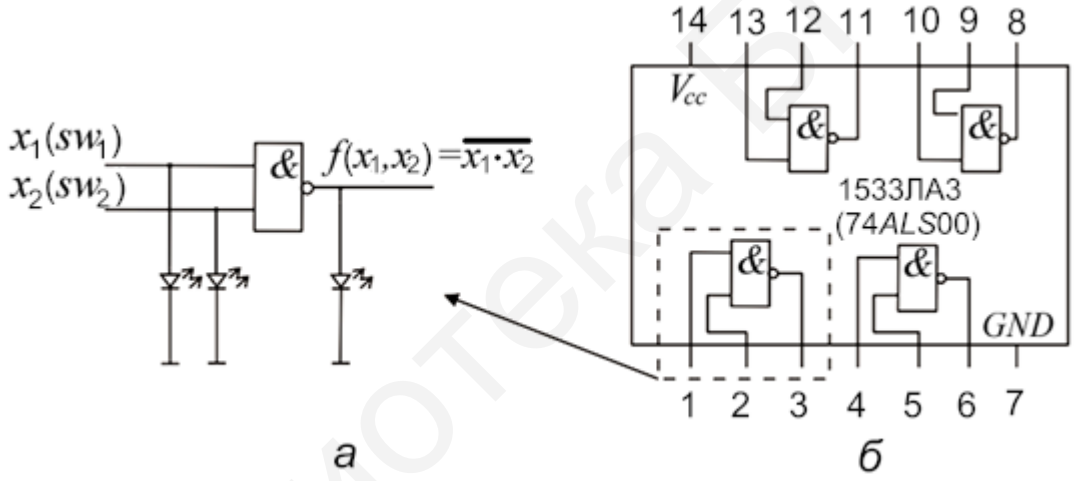
Таблица 10.8 – Таблица истинности инвертора

Вход	**Выход**
$\boldsymbol{x}$	$\boldsymbol{f(x) = \overline{x}}$
0	
1	

Задание 4. Исследовать логический элемент И-НЕ:

1. Двухвходовый элемент И-НЕ. Для этого разместить ИС 1533ЛА3 (74*ALS*00) на наборной панели *IDL*-800. Вывод 14 ИС соединить с источником питания +5*V*, а вывод 7 – с общей шиной установки. Собрать схему, как показано на рисунке 10.15, *а*.

Структурная схема логического элемента показана на рисунке 10.15, *б*.



а – принципиальная схема; *б* – структурная схема

Рисунок 10.15 – Двухвходовый элемент И-НЕ на примере ИС 1533ЛА3 (74*ALS*00)

Изменяя состояния входов x_1 и x_2, исследовать работу двухвходового элемента И-НЕ, заполнить таблицу истинности (таблица 10.9).

Таблица 10.9 – Таблица истинности двухвходового элемента И-НЕ

Входы		**Выход**
$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{f(x_1, x_2)}$
0	0	
0	1	
1	0	
1	1	

2. Двухвходовый элемент И-НЕ, используемый как инвертор.

Если соединить входы двухвходового элемента И-НЕ, то элемент будет работать как инвертор.

Собрать схему, как показано на рисунке 10.16.

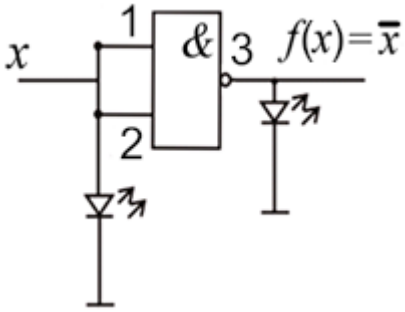


Рисунок 10.16 – Двухвходовый элемент И-НЕ, используемый как инвертор

Изменяя состояние входа *X*, исследовать работу схемы, заполнить таблицу истинности (таблица 10.10).

Таблица 10.10 – Таблица истинности двухвходового элемента И-НЕ, используемого как инвертор

Вход	**Выход**
$\boldsymbol{x}$	$\boldsymbol{f(x) = \overline{x}}$
0	
1	

3. Многовходовый элемент И-НЕ. Для этого, используя двухвходовые элементы И-НЕ, собрать трехвходовый элемент И-НЕ, реализующий функцию $f(x_1, x_2, x_3) = \overline{x_1 \cdot x_2 \cdot x_3}$ (рисунок 10.17).

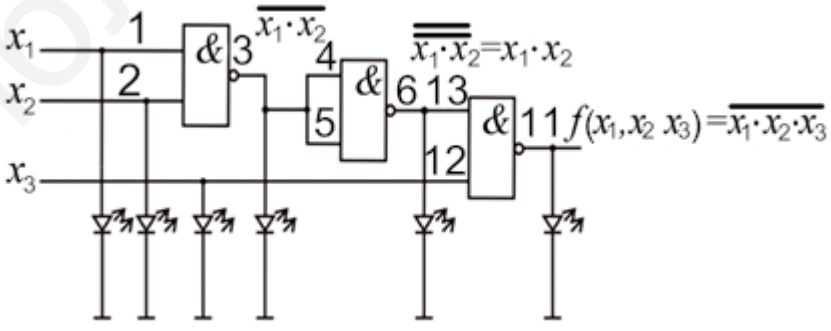


Рисунок 10.17 – Многовходовый элемент И-НЕ

Изменяя состояния входов x_1, x_2, x_3, исследовать работу схемы, заполнить таблицу истинности (таблица 10.11).

Таблица 10.11 – Таблица истинности многовходового элемента И-НЕ

Входы			**Выходы**		
x_1	x_2	x_3	$\overline{x_1 \cdot x_2}$	$x_1 \cdot x_2$	$\overline{x_1 \cdot x_2 \cdot x_3}$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

4. Реализацию функции ИЛИ с помощью логических элементов И-НЕ. Для этого, используя двухвходовые элементы И-НЕ, собрать схему, реализующую операцию ИЛИ, как показано на рисунке 10.18.

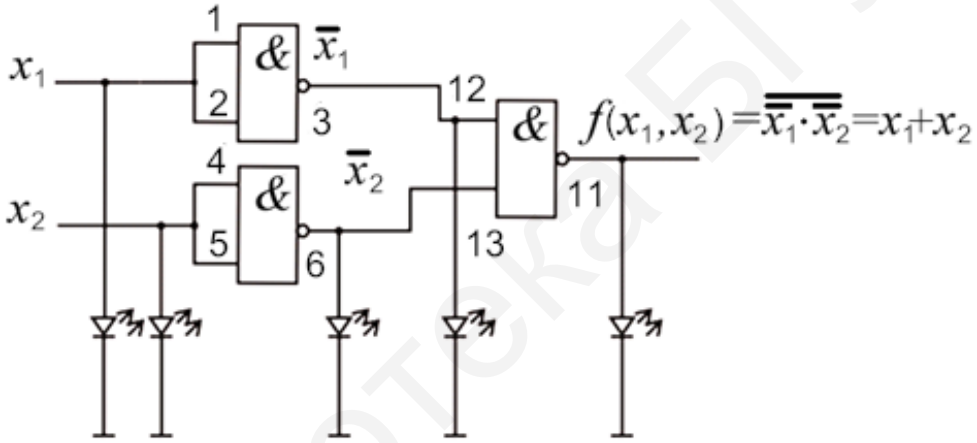


Рисунок 10.18 – Многовходовый элемент И-НЕ для реализации функции ИЛИ

Изменяя состояния входов x_1 и x_2, исследовать работу схемы, заполнить таблицу истинности (таблица 10.12).

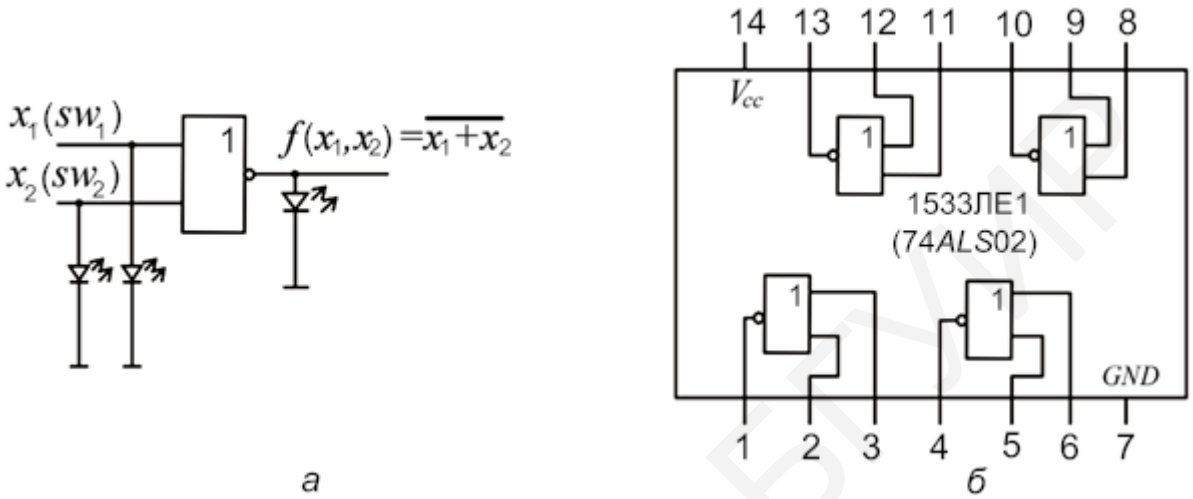
Таблица 10.12 – Таблица истинности многовходового элемента И-НЕ для реализации функции ИЛИ

Входы		**Выходы**		
x_1	x_2	$\overline{x_1}$	$\overline{x_2}$	$x_1 + x_2$
0	0			
0	1			
1	0			
1	1			

Задание 5. Исследовать логический элемент ИЛИ-НЕ:

1. Двухвходовый элемент ИЛИ-НЕ. Для этого разместить ИС 1533ЛЕ1 (74*ALS*02) на наборной панели *IDL*-800. Вывод 14 ИС соединить с источником питания +5*V*, а вывод 7 – с общей шиной установки. Собрать схему, как показано на рисунке 10.19, *а*.

Структурная схема логического элемента показана на рисунке 10.19, *б*.



а – принципиальная схема; *б* – структурная схема

Рисунок 10.19 – Двухвходовый элемент 2ИЛИ-НЕ ИС 1533ЛЕ1

Изменяя состояние входов x_1 и x_2, исследовать работу элемента 2ИЛИ-НЕ, заполнить таблицу истинности (таблица 10.13).

Таблица 10.13 – Таблица истинности двухвходового элемента 2ИЛИ-НЕ

Входы		**Выходы**
x_1	x_2	$f(x_1,x_2)=\overline{x_1+x_2}$
0	0	
0	1	
1	0	
1	1	

2. Двухвходовый элемент ИЛИ-НЕ, используемый как инвертор.

Если соединить входы элемента ИЛИ-НЕ, то элемент будет работать как инвертор.

Собрать схему, как показано на рисунке 10.20.

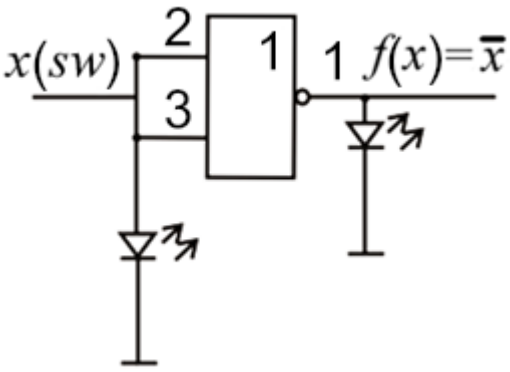


Рисунок 10.20 – Двухвходовый элемент ИЛИ-НЕ

Изменяя состояние входа x , исследовать работу схемы, заполнить таблицу истинности (таблица 10.14).

Таблица 10.14 – Таблица истинности двухвходового элемента ИЛИ-НЕ

Вход	**Выход**
$\boldsymbol{x}$	$\boldsymbol{f(x) = \bar{x}}$
0	
1	

3. Многовходовый элемент ИЛИ-НЕ. Для этого, используя двухвходовые элементы ИЛИ-НЕ, собрать трехвходовый элемент ИЛИ-НЕ, реализующий функцию $f(x_1, x_2, x_3) = \overline{x_1 + x_2 + x_3}$ (рисунок 10.21).

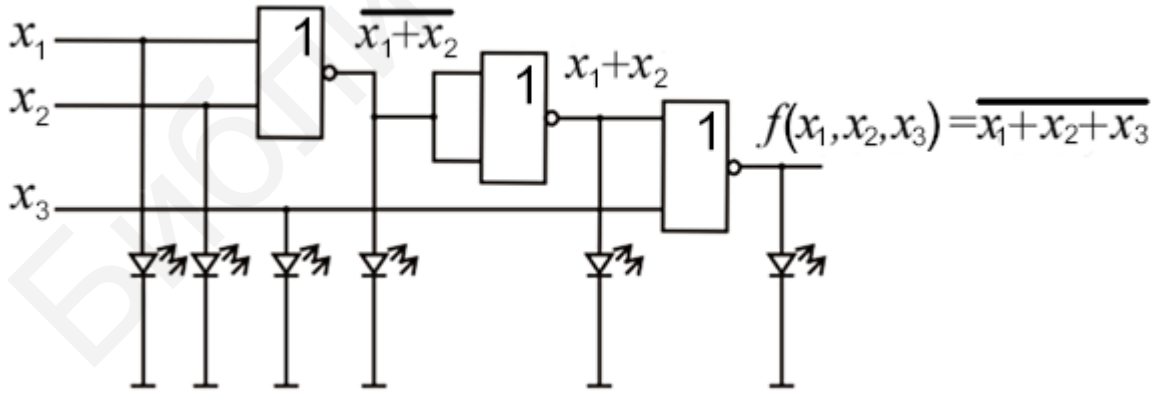


Рисунок 10.21 – Трехвходовый элемент ИЛИ-НЕ

Изменяя состояние входов x_1, x_2, x_3, исследовать работу схемы, заполнить таблицу истинности (таблица 10.15).

Таблица 10.15 – Таблица истинности трехвходового элемента ИЛИ-НЕ

Входы			**Выходы**		
x_1	x_2	x_3	$\overline{x_1 + x_2}$	x_1, x_2	$\overline{x_1 + x_2 + x_3}$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

4. Реализацию функции И с помощью логических элементов ИЛИ-НЕ. Для этого, используя двухвходовые элементы ИЛИ-НЕ, собрать схему, реализующую операцию И (рисунок 10.22).

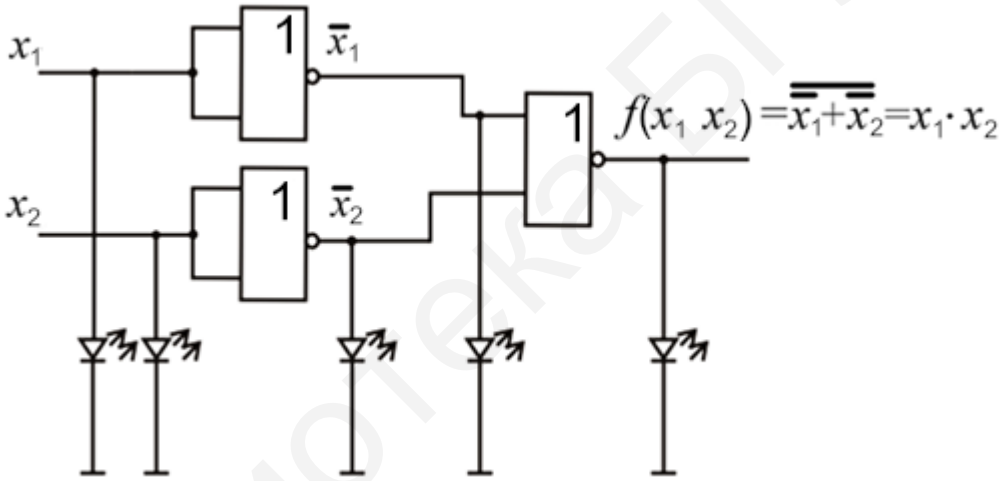


Рисунок 10.22 – Многовходовый элемент ИЛИ-НЕ

Изменяя состояния входов x_1, x_2, исследовать работу схемы, заполнить таблицу истинности (таблица 10.16).

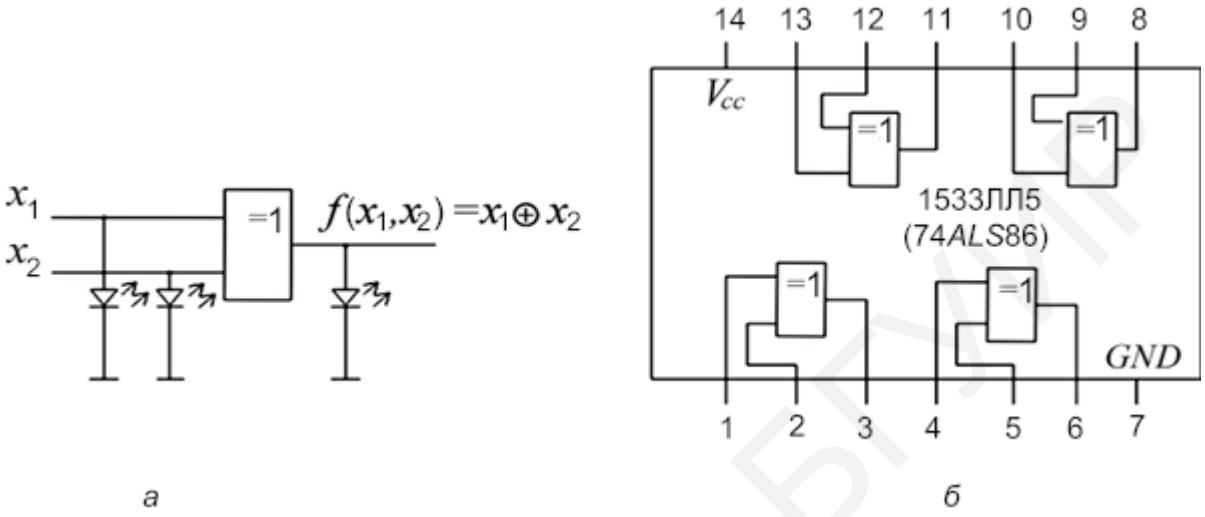
Таблица 10.16 – Таблица истинности многовходового элемента ИЛИ-НЕ

Входы		**Выходы**		
x_1	x_2	$\overline{x_1}$	$\overline{x_2}$	$x_1 \cdot x_2$
0	0			
0	1			
1	0			
1	1			

Задание 6. Исследовать логический элемент Исключающее ИЛИ:

1. Двухвходовый элемент Исключающее ИЛИ. Для этого разместить ИС 1533ЛП5 (74*ALS*86) на наборной панели *IDL*-800. Выход 14 ИС соединить с источником питания +5*V*, а вывод 7 – с общей шиной установки. Собрать схему, как показано на рисунке 10.23, *а*.

Структурная схема логического элемента показана на рисунке 10.23, *б*.



а – принципиальная схема; *б* – структурная схема

Рисунок 10.23 – Двухвходовый элемент Исключающее ИЛИ ИС 1533ЛП5

Изменяя состояние входов x_1 и x_2, исследовать работу двухвходового элемента исключающее ИЛИ, заполнить таблицу истинности (таблица 10.17).

Таблица 10.17 – Таблица истинности двухвходового элемента Исключающее ИЛИ

Входы		**Выход**
x_1	x_2	$f(x_1, x_2) = x_1 \oplus x_2$
0	0	
0	1	
1	0	
1	1	

2. Многовходовый элемент Исключающее ИЛИ. Для этого, используя двухвходовые элементы Исключающее ИЛИ, собрать схему трехвходового элемента Исключающее ИЛИ (рисунок 10.24).

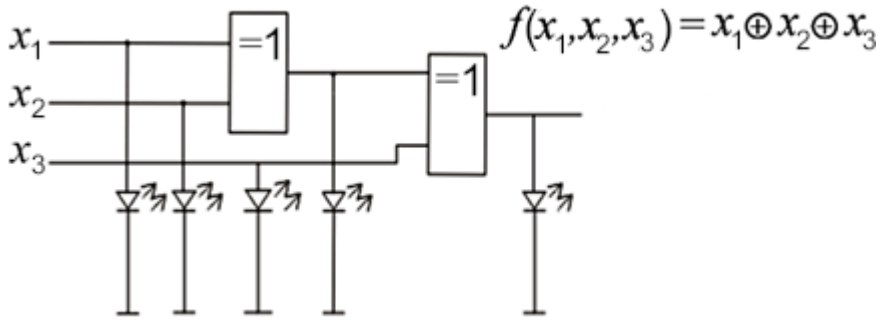


Рисунок 10.24 – Трехвходовый элемент Исключающее ИЛИ

Изменяя состояния входов x_1, x_2, x_3, исследовать работу схемы, заполнить таблицу истинности (таблица 10.18).

Таблица 10.18 – Таблица истинности трехвходового элемента Исключающее ИЛИ

Входы			**Выходы**	
x_1	x_2	x_3	$x_1\oplus x_2$	$x_1\oplus x_2\oplus x_3$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

3. Логический элемент Исключающее ИЛИ, используемый как повторитель. Для этого собрать схему, как показано на рисунке 10.25.

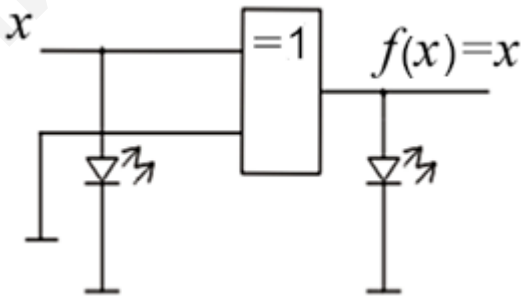


Рисунок 10.25 – Логический элемент Исключающее ИЛИ, используемый как повторитель

Изменяя состояние входа x, исследовать работу схемы, заполнить таблицу истинности (таблица 10.19).

Таблица 10.19 – Таблица истинности логического элемента исключающее ИЛИ, используемого как повторитель

Вход	Выход
x	$f(x) = x$
0	
1	

4. Логический элемент Исключающее ИЛИ, используемый как инвертор. Для этого собрать схему, как показано на рисунке 10.26.

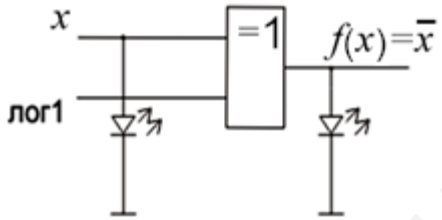


Рисунок 10.26 – Логический элемент Исключающее ИЛИ, используемый как инвертор

Изменяя состояния входа x, исследовать работу схемы, заполнить таблицу истинности (таблица 10.20).

Таблица 10.20 – Таблица истинности логического элемента Исключающее ИЛИ, используемого как инвертор

Вход	Выход
x	$f(x) = x$
0	
1	

10.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.
3. Таблицы, отражающие результаты исследований.
4. Выводы по результатам исследований.

10.5 Контрольные вопросы

1. Что называется функцией алгебры логики?
2. Каковы основные ФАЛ для двух переменных?
3. Что такое полная система ФАЛ?
4. Назовите основные логические элементы, их обозначения.
5. В чем суть многовходовых логических элементов?

11 Лабораторная работа №2

СИНТЕЗ КОМБИНАЦИОННЫХ УСТРОЙСТВ В ЗАДАННОМ БАЗИСЕ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

11.1 Цель работы

Изучение представления функций алгебры логики (ФАЛ) в дизъюнктивной нормальной форме и конъюнктивной нормальной форме. Реализация этих функций в базисах И-ИЛИ, И-НЕ; ИЛИ-И, ИЛИ-НЕ.

11.2 Теоретические сведения

Любая ФАЛ может быть представлена логическим выражением в одной из следующих форм:

- дизъюнктивной нормальной форме;
- конъюнктивной нормальной форме.

Это, конечно, не означает, что ФАЛ не может быть в другой форме. ФАЛ может быть в различных формах, но эти две формы удобны при упрощении ФАЛ стандартными методами.

Пример. Пусть дана следующая логическая функция:

$$f(x_1, x_2, x_3) = (x_1 + x_2\overline{x_3})(x_2 + x_1x_3). \quad (11.1)$$

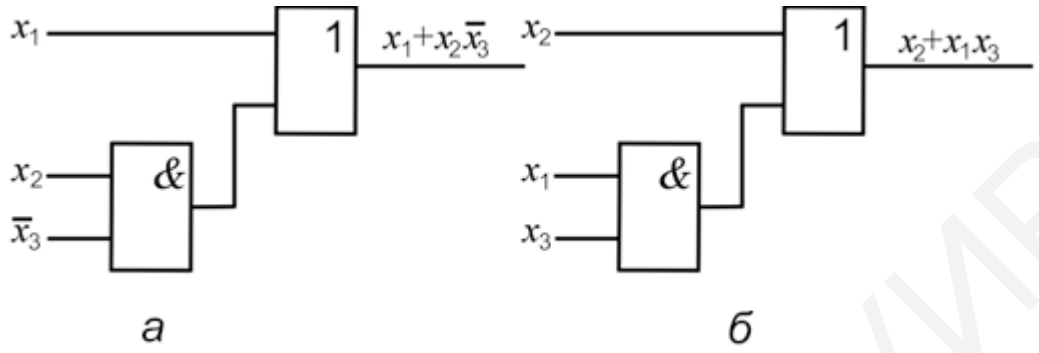
Необходимо:

1) реализовать эту функцию, используя логические элементы;
2) определить, возможно ли реализовать эту функцию с использованием только логических элементов И-НЕ и ИЛИ-НЕ, и, если возможно, построить такие схемы;
3) определить, возможно ли упростить эту функцию, если это возможно, то упростить;
4) построить схемы по упрощенным выражениям;
5) сравнить полученные схемы.

Решение. Из выражения для $f(x_1, x_2, x_3)$ видно, что имеются три переменные x_1, x_2, x_3. При этом переменная x_3 дана в нормальном (неинверсном) виде и в инверсном виде – $\overline{x_3}$.

Схема может быть построена путем рассмотрения выражения (11.1) и выбора логических элементов для реализации соответствующих термов выражения. Примем, что переменные возможны в неинвертированном и инвертированном виде.

Первый терм x_1 представляет собой только одну переменную, а второй терм имеет переменные x_2 и $\overline{x_3}$. Очевидно, что второй терм – не что иное, как элементарная конъюнкция и этот терм может быть реализован посредством использования двухвходового логического элемента И. Комбинация первых двух термов реализуется с помощью элемента ИЛИ (рисунок 11.1, *а*).



а – реализация комбинации первых двух термов; *б* – реализация комбинации третьего и четвертого термов

Рисунок 11.1 – Логические элементы, реализующие ФАЛ

Третий терм x_2 – снова просто одна переменная и четвертый терм представляет собой конъюнкцию двух переменных $x_1\, x_3$. Схема для реализации этих двух термов строится так же, как и для реализации двух предыдущих (см. рисунок 11.1, *б*).

Теперь с использованием дополнительного логического элемента И построим полную схему, которая реализует данную функцию (рисунок 11.2).

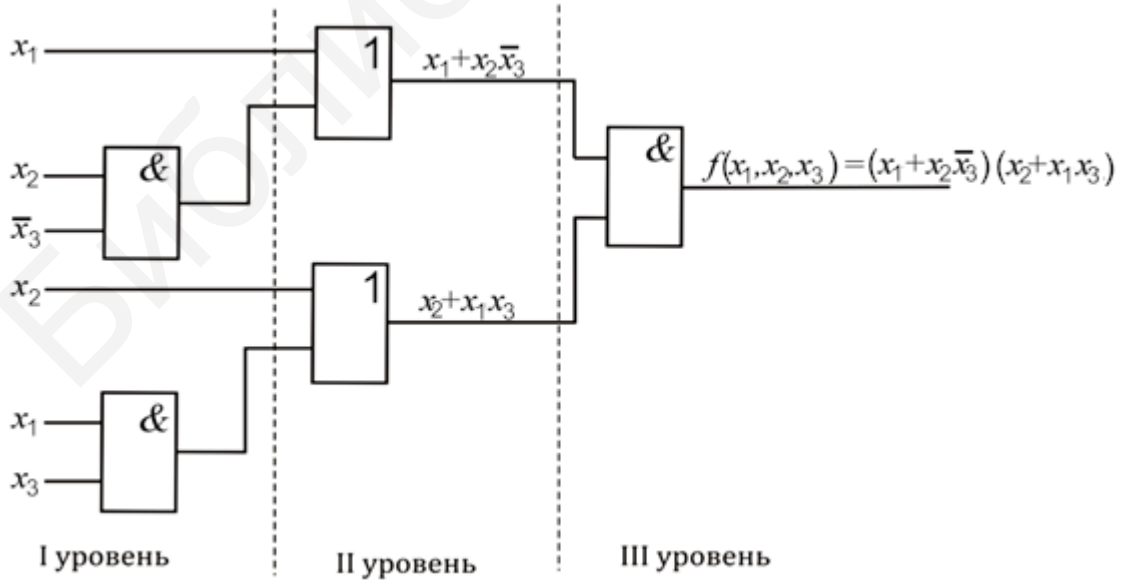


Рисунок 11.2 – Реализация ФАЛ на логических элементах ИЛИ

Для реализации схемы (см. рисунок 11.2) требуется три двухвходовых элемента И и два двухвходовых элемента ИЛИ. Такая реализация известна как трехуровневая реализация.

Дизъюнктивная нормальная форма. Преобразуем выражение (11.1) следующим образом:

$$f(x_1, x_2, x_3) = (x_1 + x_2\overline{x_3})(x_2 + x_1x_3) = x_1x_2 + x_1x_1x_3 + \\ +x_2\overline{x_3}x_2 + x_2\overline{x_3}x_1x_3 = x_2x_1 + x_3x_1 + x_2\overline{x_3}. \qquad (11.2)$$

Представление ФАЛ в таком виде (11.2) известно как представление ФАЛ в дизъюнктивной нормальной форме. Итак, формы ФАЛ, представляющие дизъюнкцию элементарных конъюнкций, называются дизъюнктивными нормальными формами (ДНФ). Под элементарной конъюнкцией понимается логическое произведение отдельных переменных в нормальном или инвертированном виде. Функция, представленная в ДНФ, может быть реализована посредством использования конфигурации И-ИЛИ (рисунок 11.3).

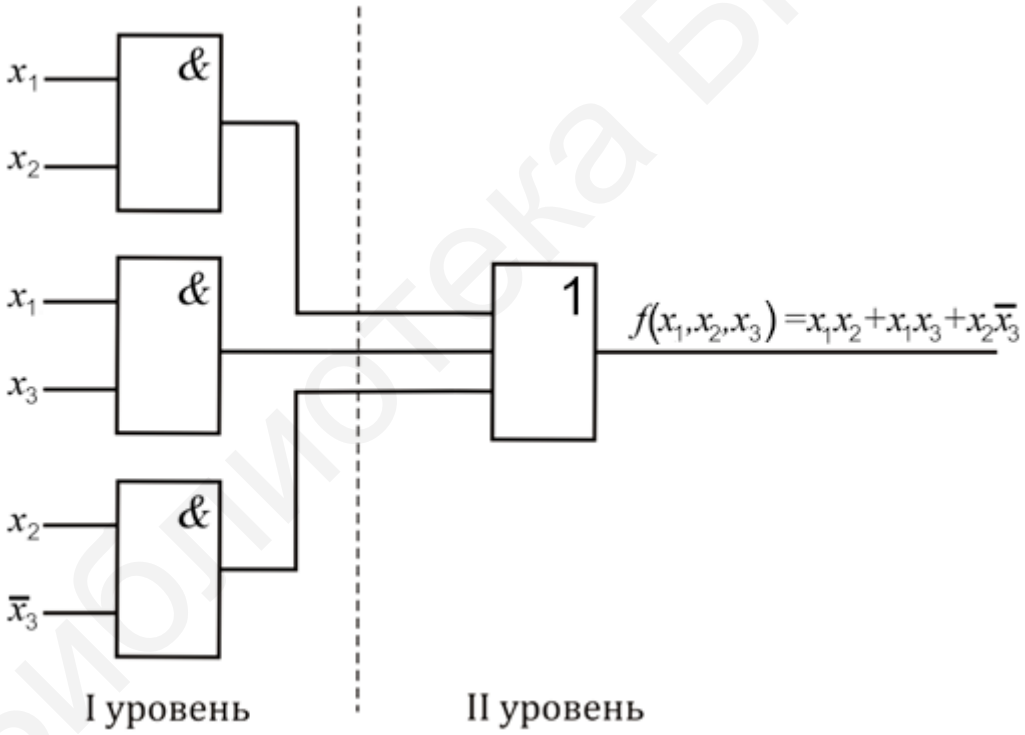


Рисунок 11.3 – Реализация ФАЛ, представленной в ДНФ (конфигурация И-ИЛИ)

Реализация ФАЛ (см. рисунок 11.3) известна как двухуровневая реализация. Первый уровень состоит из элементов И, второй уровень состоит из элемента ИЛИ.

Используя закон де Моргана, выражение (11.2) можно переписать следующим образом:

$$f(x_1, x_2, x_3) = \overline{\overline{x_2x_3 + x_3x_1 + x_2\overline{x_3}}} = \overline{\overline{x_2x_1} \cdot \overline{x_3x_1} \cdot \overline{x_2\overline{x_3}}}. \qquad (11.3)$$

Теперь выражение (11.3) может быть реализовано с использованием только элементов И-НЕ (рисунок 11.4).

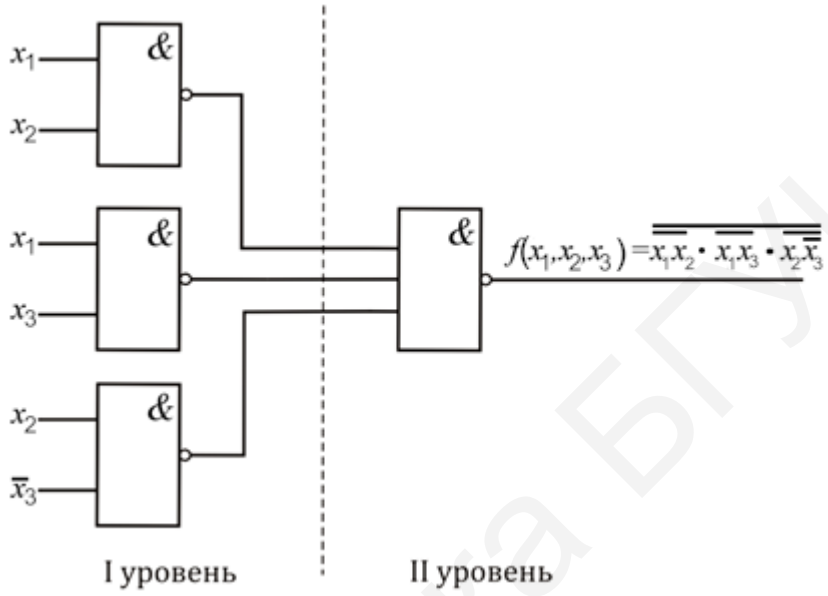


Рисунок 11.4 – Реализация ФАЛ с использованием только элементов И-НЕ

Вывод. Для того чтобы реализовать ФАЛ с помощью только элементов И-НЕ, необходимо представить ФАЛ в ДНФ, а затем использовать двойное инвертирование и закон де Моргана.

Опять преобразуем исходную ФАЛ, используя распределительный закон для оператора И:

$$f(x_1, x_2, x_3) = (x_1 + x_2\overline{x_3})(x_2 + x_1x_3) = (x_1 + x_3)(x_1 + \overline{x_3}) \times$$
$$\times (x_2 + x_1)(x_2 + +x_3) = (x_1 + x_2)(x_1 + \overline{x_3})(x_2 + x_3). \qquad (11.4)$$

Данное представление ФАЛ (выражение (11.4)) известно как представление ФАЛ в конъюнктивной нормальной форме. Итак, формы ФАЛ, представляющие конъюнкцию элементарных дизъюнкций, называются конъюнктивными нормальными формами (КНФ). Под элементарной дизъюнкцией понимается логическая сумма отдельных переменных в нормальном или инвертированном виде. Функция, представленная в КНФ, может быть реализована с использованием конфигурации ИЛИ-И (рисунок 11.5).

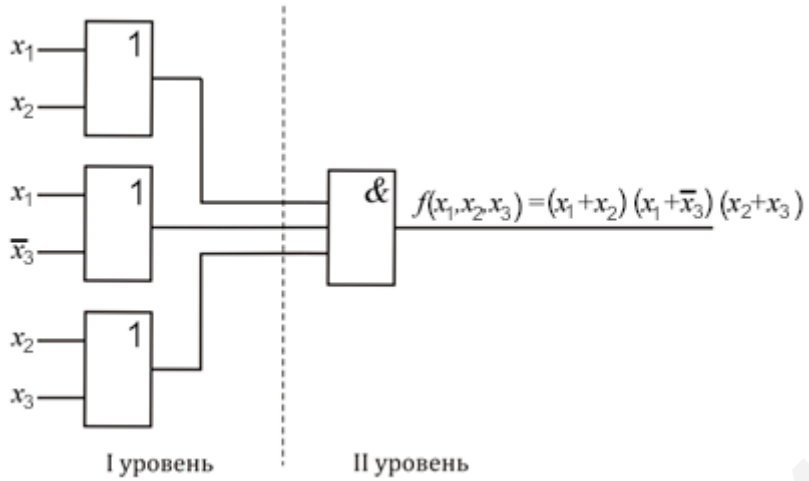


Рисунок 11.5 – Реализация ФАЛ, представленной в КНФ (конфигурация ИЛИ-И)

Используя двойную инверсию и закон де Моргана, выражение (11.4) можно представить в виде выражения

$$f(x_1, x_2, x_3) = \overline{\overline{(x_1 + x_2)(x_1 + \overline{x_3})(x_2 + x_3)}} = \\ = \overline{\overline{x_1 + x_3} + \overline{x_1 + \overline{x_3}} + \overline{x_2 + x_3}}. \qquad (11.5)$$

Теперь функция может быть реализована с использованием только элементов ИЛИ-НЕ (рисунок 11.6).

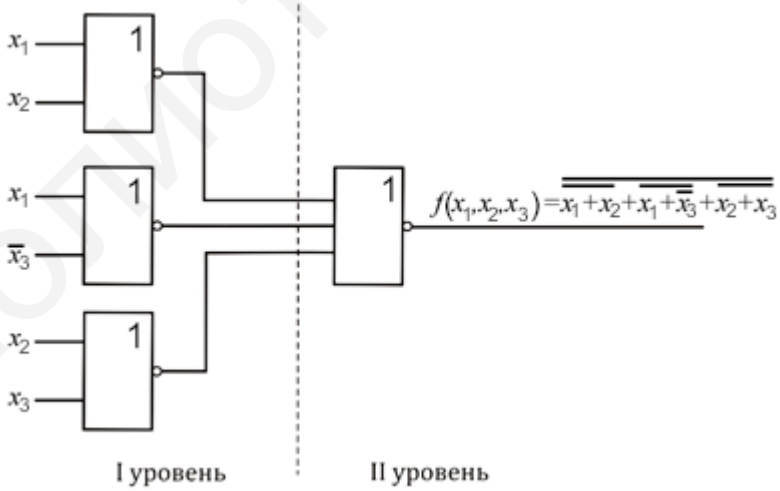


Рисунок 11.6 – Реализация ФАЛ с использованием только элементов ИЛИ-НЕ

Вывод. Для того чтобы реализовать ФАЛ с помощью только элементов ИЛИ-НЕ, необходимо представить ФАЛ в КНФ, а затем использовать двойное инвертирование и закон де Моргана.

Теперь рассмотрим, как выражения (11.2) и (11.4) могут быть упрощены:

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2\overline{x_3} = x_1x_2(x_3 + \overline{x_3}) + x_1x_3 + x_2\overline{x_3} = \\ = x_1x_2x_3 + x_1x_2\overline{x_3} + x_1x_2 + x_2\overline{x_3} = x_1x_3(x_2 + 1) + x_2\overline{x_3} \times \\ \times (x_1 + 1) = x_1x_3 + +x_2\overline{x_3}; \qquad (11.6)$$

$$f(x_1, x_2, x_3) = (x_1 + x_2)(x_1 + \overline{x_3})(x_2 + x_3) = (x_1 + x_2 + x_3\overline{x_3}) \times \\ \times (x_1 + +\overline{x_3})(x_2 + x_3) = (x_1 + x_2 + x_3)(x_1 + x_2 + \overline{x_3}) \times \\ \times (x_1 + +\overline{x_3})(x_2 + x_3) = (x_1 + \overline{x_3})(x_2 + x_3). \qquad (11.7)$$

Реализация выражений (11.6) и (11.7) с использованием только элементов И-НЕ и ИЛИ-НЕ представлена на рисунке 11.7.

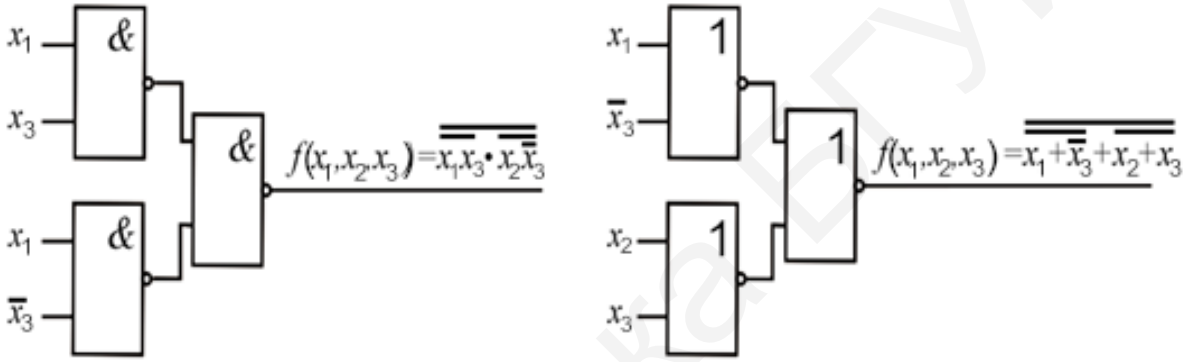


Рисунок 11.7 – Реализация функции после упрощения

Сравнение полученных схем:

1. Реализация функции в соответствии с выражением (11.1) потребовала максимальное количество элементов, и полученная схема оказалась трехуровневой, что снижает ее быстродействие.

2. Реализация функции в соответствии с выражениями (11.3) и (11.5) является очень полезной, поскольку используется только один тип логических элементов (И-НЕ/ИЛИ-НЕ), которые находятся в одном корпусе ИС. Эти схемы двухуровневые.

3. Для реализации функции в соответствии с выражениями (11.6) и (11.7) требуется минимальное количество элементов, поэтому упрощение логических выражений является очень полезными.

Рассмотрим снова выражение (11.2), в котором функция представлена в ДНФ, и выражение (11.4), в котором функция представлена в КНФ. В этих выражениях индивидуальные термы не содержат все переменные. Если все термы в ДНФ и КНФ содержат все переменные, то такие ДНФ и КНФ называют **совершенными**. Каждый индивидуальный терм в совершенной дизъюнктивной нормальной форме (СДНФ) называется **минтермом**, а каждый индивидуальный

терм в совершенной конъюнктивной нормальной форме (СКНФ) называется **макстермом**.

Функция в ДНФ может быть преобразована в СДНФ путем логического умножения термов (конъюнкций) ДНФ с термами, образованными путем логического сложения переменной и ее отрицания, которая отсутствует в исходных термах.

Пример 1. Преобразовать функцию $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2\overline{x_3}$ в СДНФ.

Решение. Преобразование функции в СДНФ:

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2\overline{x_3} = x_1x_2(x_3 + \overline{x_3}) + x_1x_3(x_2 + \overline{x_2}) + \\ + x_2\overline{x_3}(x_1 + \overline{x_1}) = x_1x_2x_3 + x_1x_2\overline{x_3} + x_1x_2x_3 + x_1\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} = \\ = x_1x_2x_3 + x_1x_2\overline{x_3} + x_1\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3}. \quad (11.8)$$

Функция в КНФ может быть преобразована в СКНФ путем логического сложения термов (дизъюнкций) КНФ с термами, образованными путем логического умножения переменной и ее отрицания, которая отсутствует в исходном терме.

Пример 2. Преобразовать функцию $f(x_1, x_2, x_3) = (x_1 + \overline{x_3})(x_2 + x_3)$ в СКНФ.

Решение. Преобразование функции в СКНФ:

$$f(x_1, x_2, x_3) = (x_1 + \overline{x_3})(x_2 + x_3) = (x_1 + \overline{x_3} + x_2\overline{x_2})(x_2 + x_3 + x_1\overline{x_1}) = \\ = (x_1 + x_2 + \overline{x_3})(x_1 + \overline{x_2} + \overline{x_3})(x_1 + x_2 + x_3)(\overline{x_1} + x_2 + x_3). \quad (11.9)$$

Если ФАЛ имеет n переменных, то число возможных минтермов (конституент единицы или составляющих единицы) равно 2^n. Также число возможных макстермов (конституент нуля или составляющих нуля) равно 2^n.

В таблице 11.1 представлены все возможные минтермы и макстермы для четырех переменных. В левой части таблицы записаны все наборы переменных. Минтерм для любого набора переменных записывается в виде конъюнкции. Причем если на данном наборе переменная равна единице, то переменная записывается в нормальном (неинвертированном) виде, и если переменная равна нулю, то переменная записывается в инвертированном виде. Макстерм для любого набора переменных записывается в виде дизъюнкции. Причем если на данном наборе переменная равна единице, то она записывается в инвертированном виде, и если переменная равно нулю, то она записывается в нормальном виде.

В правой части таблицы представлена ФАЛ, описывающая работу устройства сравнения, которое сравнивает два двоичных двухразрядных числа $A\ (x_1, x_2)$ и $B\ (x_3, x_4)$. Если эти два числа равны, то на выходе устройства сравнения должна быть единица, а если не равны, – нуль.

Таблица 11.1 – Минтермы и макстермы для четырех переменных

Переменные	**Минтермы**	**Макстермы**	**Функция *y***
$x_1x_2x_3x_4$	m_i	M_i	$A(x_1,x_2)=B(x_3,x_4)$
0 0 0 0	$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4=m_0$	$x_1+x_2+x_3+x_4=M_0$	1
0 0 0 1	$\bar{x}_1\bar{x}_2\bar{x}_3x_4=m_1$	$x_1+x_2+x_3+\overline{x_4}=M_1$	0
0 0 1 0	$\bar{x}_1\bar{x}_2x_3\bar{x}_4=m_2$	$x_1+x_2+\overline{x_3}+x_4=M_2$	0
0 0 1 1	$\bar{x}_1\bar{x}_2x_3x_4=m_3$	$x_1+x_2+\bar{x}_3+\bar{x}_4=M_3$	0
0 1 0 0	$\bar{x}_1x_2\bar{x}_3\bar{x}_4=m_4$	$x_1+\bar{x}_2+x_3+x_4=M_4$	0
0 1 0 1	$\bar{x}_1x_2\bar{x}_3x_4=m_5$	$x_1+\bar{x}_2+x_3+\bar{x}_4=M_5$	1
0 1 1 0	$\bar{x}_1x_2x_3\bar{x}_4=m_6$	$x_1+\bar{x}_2+\bar{x}_3+x_4=M_6$	0
0 1 1 1	$\bar{x}_1x_2x_3x_4=m_7$	$x_1+\bar{x}_2+\bar{x}_3+\bar{x}_4=M_7$	0
1 0 0 0	$x_1\bar{x}_2\bar{x}_3\bar{x}_4=m_8$	$\bar{x}_1+x_2+x_3+x_4=M_8$	0
1 0 0 1	$x_1\bar{x}_2\bar{x}_3x_4=m_9$	$\bar{x}_1+x_2+x_3+\bar{x}_4=M_9$	0
1 0 1 0	$x_1\bar{x}_2x_3\bar{x}_4=m_{10}$	$\bar{x}_1+x_2+\bar{x}_3+x_4=M_{10}$	1
1 0 1 1	$x_1\bar{x}_2x_3x_4=m_{11}$	$\bar{x}_1+x_2+\bar{x}_3+\bar{x}_4=M_{11}$	0
1 1 0 0	$x_1x_2\bar{x}_3\bar{x}_4=m_{12}$	$\bar{x}_1+\bar{x}_2+x_3+x_4=M_{12}$	0
1 1 0 1	$x_1x_2\bar{x}_3x_4=m_{13}$	$\bar{x}_1+\bar{x}_2+x_3+\bar{x}_4=M_{13}$	0
1 1 1 0	$x_1x_2x_3\bar{x}_4=m_{14}$	$\bar{x}_1+\bar{x}_2+\bar{x}_3+x_4=M_{14}$	0
1 1 1 1	$x_1x_2x_3x_4=m_{15}$	$\bar{x}_1+\bar{x}_2+\bar{x}_3+\bar{x}_4=M_{15}$	1

Наборы переменных в левой части таблицы и значения функции в правой части таблицы представляют собой таблицу истинности ФАЛ, описывающую работу устройства сравнения.

Из таблицы истинности ФАЛ может быть записана в СДНФ и в СКНФ.

В СДНФ ФАЛ записывается из таблицы истинности как дизъюнкция минтермов, на которых функция принимает значение 1:

$$Y=m_0+m_5+m_{10}+m_{15}=\sum m(0,5,10,15)=$$
$$=\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4+\bar{x}_1x_2\bar{x}_3x_4+x_1\bar{x}_2x_3\bar{x}_4+x_1x_2x_3x_4. \qquad (11.10)$$

В СКНФ ФАЛ из таблицы истинности записывается как конъюнкция макстермов, на которых функция принимает значение 0:

$$Y=M_1M_2M_3M_4M_5M_6M_7M_8M_9M_{10}M_{11}M_{12}M_{13}M_{14}=$$
$$=\text{ПМ}(1,2,3,4,5,6,7,8,9,10,11,12,13,14)=$$
$$=(x_1+x_2+x_3+\bar{x}_4)(x_1+x_2+\bar{x}_3+x_4)\cdot\ldots\cdot(\bar{x}_1+\bar{x}_2+\bar{x}_3+x_4). \quad (11.11)$$

Выражения (11.10) и (11.11) представляют одну и ту же функцию, выраженную в минтермах и макстермах.

Эти два представления обладают свойством дополнительности. И если имеется выражение для функции в минтермах (в СДНФ), то выражение в макстермах (в СКНФ) может быть получено с использованием этого свойства и наоборот.

11.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудование и компоненты: универсальная лабораторная установка *IDL*-800, ИС 1533ЛА3 (74*ALS*00) – четыре логических элемента 2И-НЕ, ИС 1533ЛА4 (74*ALS*10) – три логических элемента 3И-НЕ, ИС 1533ЛЕ1 (74*ALS*02) – четыре логических элемента 2ИЛИ-НЕ, ИС 1533ЛЕ4 (74*ALS*27) – три логических элемента 3ИЛИ-НЕ.

ФАЛ задана таблицей истинности (таблица 11.2).

Таблица 11.2 – Исходные данные таблицы истинности для выполнения индивидуального задания

Вход			**Выход**
x_1	x_2	x_3	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Задание 1. Исследовать представления ФАЛ в ДНФ. Для этого для заданной таблицы истинности записать логическое выражение для *Y* в СДНФ. Нарисовать схему и реализовать ее только с помощью элементов И-НЕ. Проверить таблицу истинности. Упростить логическое выражение для *Y*, нарисовать схему и реализовать ее только с помощью элементов И-НЕ. Проверить таблицу истинности.

Задание 2. Исследовать представления ФАЛ в КНФ. Для этого для заданной таблицы истинности записать логическое выражение для *Y* в СКНФ. Нарисовать схему и реализовать ее только с помощью элементов ИЛИ-НЕ. Проверить таблицу истинности. Упростить логическое выражение для *Y*, нарисовать схему и реализовать ее только с помощью элементов ИЛИ-НЕ. Проверить таблицу истинности.

Задание 3. Исследовать представления ФАЛ в МДНФ и МКНФ. Для этого реализовать схему $f_{\text{МДНФ}} = (x_1, x_2, x_3, x_4)$ и $f_{\text{МКНФ}} = (x_1, x_2, x_3, x_4)$ на логических элементах согласно варианту (таблица 11.3).

Таблица 11.3 – Варианты заданий

Номер набора аргументов	Наборы аргументов				Номер варианта										
					1	2	3	4	5	6	7	8	9	10	–
	x_4	x_3	x_2	x_1	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
0	0	0	0	0	1	1	1	0	1	0	0	1	1	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1
2	0	0	1	0	1	1	Ф	0	Ф	Ф	Ф	1	1	0	1
3	0	0	1	1	Ф	0	1	Ф	0	1	0	Ф	Ф	1	0
4	0	1	0	0	0	1	0	0	1	0	0	1	1	1	Ф
5	0	1	0	1	0	1	1	1	Ф	0	1	1	0	1	1
6	0	1	1	0	1	0	0	0	1	Ф	1	Ф	Ф	0	Ф
7	0	1	1	1	0	0	Ф	1	Ф	0	Ф	1	1	1	1
8	1	0	0	0	1	Ф	0	0	Ф	1	0	0	1	Ф	1
9	1	0	0	1	Ф	1	1	0	0	1	1	1	1	0	0
10	1	0	1	0	1	1	Ф	Ф	0	Ф	Ф	0	Ф	0	1
11	1	0	1	1	1	Ф	1	1	1	1	0	1	1	1	0
12	1	1	0	0	1	1	0	Ф	Ф	1	Ф	0	0	Ф	1
13	1	1	0	1	1	1	1	1	0	1	1	1	0	0	0
14	1	1	1	0	1	0	0	1	0	0	0	Ф	Ф	Ф	Ф
15	1	1	1	1	0	1	1	Ф	Ф	1	1	0	0	1	0

11.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.
3. Таблицы, отражающие результаты исследований.
4. Выводы по результатам исследований.

11.5 Контрольные вопросы

1. Как представляется ФАЛ в ДНФ?
2. Что такое СДНФ?
3. Что необходимо для реализации ФАЛ, используя только элементы И-НЕ?
4. Как представляется ФАЛ в КНФ?
5. Что такое СКНФ?
6. Что необходимо для реализации ФАЛ, используя только элементы ИЛИ-НЕ?

12 Лабораторная работа №3

ИССЛЕДОВАНИЕ СУММАТОРОВ И ВЫЧИТАТЕЛЕЙ

12.1 Цель работы

Исследование функционирования сумматоров и вычитателей, а также суммирования/вычитания в дополнительных кодах.

12.2 Теоретические сведения

Несомненно, что наиболее базовой арифметической операцией является суммирование двух двоичных цифр. Это простое сложение состоит из четырех возможных элементарных, а именно 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 и 1 + 1 = 10. Первые три операции дают сумму, чья длина есть одна цифра. Однако, когда оба слагаемых равны единице, двоичная сумма состоит из двух цифр. Старшая цифра называется переносом. Когда слагаемые числа состоят из нескольких цифр, то полученный перенос прибавляется к следующей более значащей паре бит. Комбинационная схема, осуществляющая сложение двух бит, называется **полусумматором**. Если схема осуществляет сложение трех бит (два значащих и перенос из предыдущего разряда), то такая схема называется **полным сумматором**.

Полусумматор. Полусумматор может быть использован для сложения двух наименее значащих бит A_0 и B_0 двух чисел, где отсутствует входной перенос. Условное обозначение полусумматора показано на рисунке 12.1, а его таблица истинности представлена в таблице 12.1.

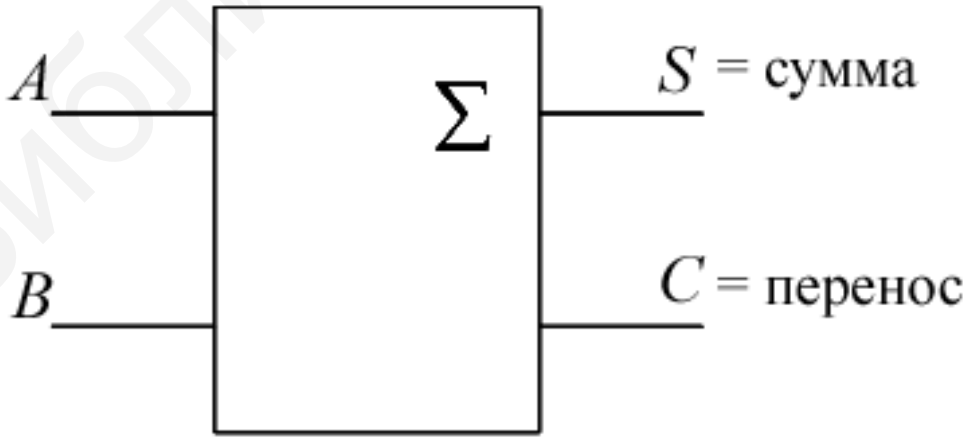


Рисунок 12.1 – Условное обозначение полусумматора

Таблица 12.1 – Таблица истинности полусумматора

Входы		**Выходы**	
A	***B***	***S***	***C***
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Из таблицы истинности (см. таблицу 12.1) мы можем получить логические выражения для *S* и *C*:

$$S = \bar{A}B + A\bar{B} = A \oplus B; \quad (12.1)$$

$$C = AB. \quad (12.2)$$

В соответствии с выражениями (12.1) и (12.2) полусумматор может быть легко реализован с помощью логических элементов Исключающее ИЛИ и И (рисунок 12.2).

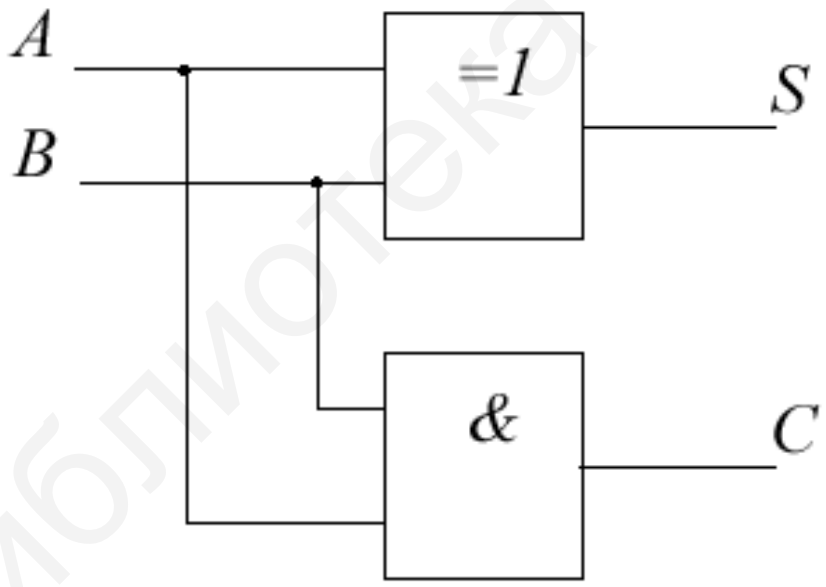


Рисунок 12.2 – Схема полусумматора

Полный сумматор. Полусумматор имеет только два входа и не имеет возможности суммировать перенос из младшего разряда при многобитовом сложении. Для этих целей используется третий вход, и схема осуществляет сложение A_n, B_n и C_{n-1}, где A_n, B_n – биты чисел A и B в разряде n, а C_{n-1} – перенос при сложении из $n-1$ разряда (таблица 12.2).

Таблица 12.2 – Таблица истинности полного сумматора

Входы			**Выходы**	
A_n	B_n	C_{n-1}	S_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Логические выражения для S_n и C_n могут быть упрощены с помощью карт Карно (рисунок 12.3).

Рисунок 12.3 – Упрощение выражений для S_n и C_n полного сумматора

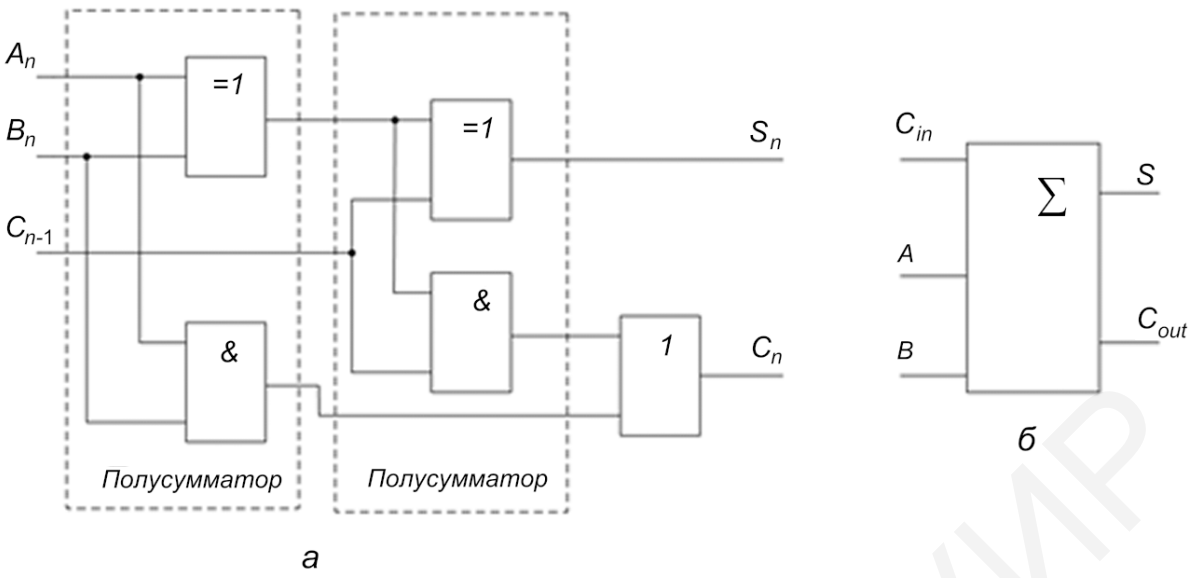
Логическое выражение для S_n может быть упрощено в базисе функций Исключающее ИЛИ:

$$S_n = (\bar{A}_n \bar{B}_n C_{n-1} + \bar{A}_n B_n \bar{C}_{n-1}) + (A_n \bar{B}_n \bar{C}_{n-1} + A_n B_n C_{n-1}) = \\ = \bar{A}_n (B_n \oplus C_{n-1}) + A_n \overline{(B_n \oplus C_{n-1})} = A_n \oplus B_n \oplus C_{n-1}. \quad (12.3)$$

ФАЛ для C_n может быть покрыта с помощью обычного логического соседства, а также диагонального соседства (см. рисунок 12.3), представим это в виде выражения

$$C_n = (A_n \bar{B}_n C_{n-1} + \bar{A}_n B_n C_{n-1}) + (A_n B_n C_{n-1} + A_n B_n \bar{C}_{n-1}) = \\ = C_{n-1}(A_n \oplus B_n) + A_n B_n. \quad (12.4)$$

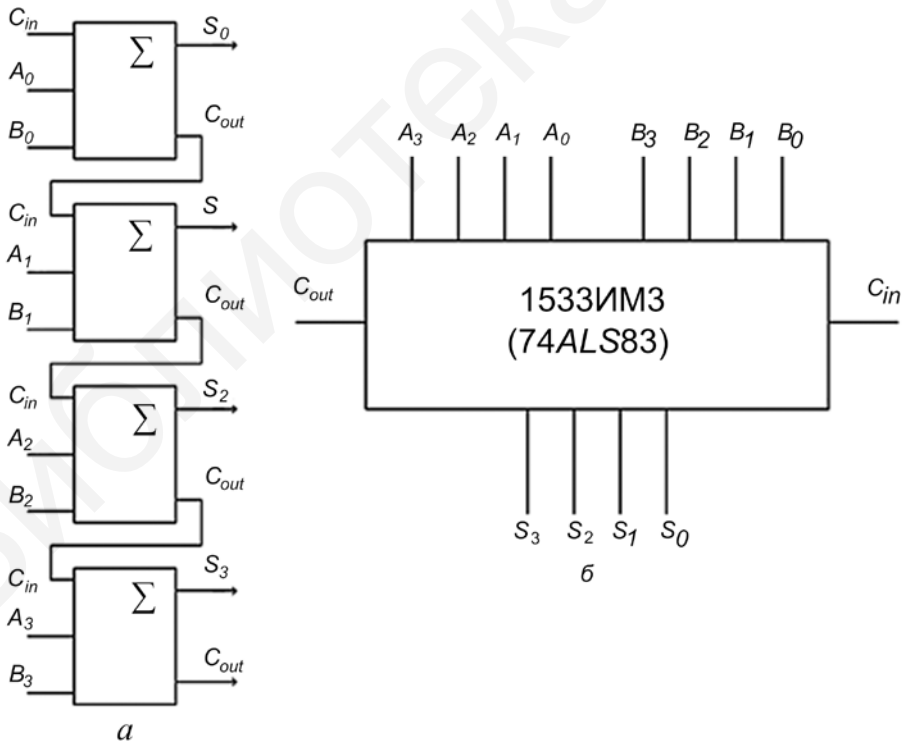
В соответствии с выражениями (12.3) и (12.4) на рисунке 12.4 показана схема полного сумматора.



а – логическая схема; *б* – условное обозначение

Рисунок 12.4 – Полный сумматор

Параллельный сумматор. Для сложения четырехразрядных двоичных чисел четыре полных сумматора могут быть соединены (рисунок 12.5).



а – псевдопараллельный сумматор; *б* – четырехразрядный параллельный сумматор

Рисунок 12.5 – Параллельный сумматор

Для получения четырехразрядного параллельного сумматора два четырехразрядных двоичных числа подаются на входы A и B четырех полных сумматоров. A_0, B_0 – это, конечно, наименее значимые биты. Выход переноса C_{out} каждого сумматора соединен непосредственно с входом переноса C_{in} следующего более значимого полного сумматора. Выход C_{out} наиболее значимого разряда является переносом всей схемы. Биты переноса передаются через схему от разряда к разряду и индивидуальная сумма будет правильной, когда перенос из предыдущего разряда появится в данном разряде. Это означает, что выходная, или полная, сумма появится, когда все биты переноса пройдут через схему. Поэтому такую схему (см. рисунок 12.5, *а*) более точно называют псевдопараллельным сумматором.

В интегральном исполнении выпускается четырехразрядный параллельный сумматор 1533ИМ3(74*ALS*83) (см. рисунок 12.5, *б*).

Полувычитатель. Логическая схема, которая осуществляет вычитание B (вычитаемое) из A (уменьшаемое), где A и B – однобитовые числа, называется полувычитателем. Процесс вычитания может быть представлен с помощью таблицы истинности (таблица 12.3).

Таблица 12.3 – Таблица истинности полувычитателя

Входы		**Выходы**	
A	***B***	***D***	***C***
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Здесь A и B – это входы, а выходы D – разность и C – заем.

Из таблицы истинности (см. таблицу 12.3) получим

$$D = \bar{A}B + A\bar{B} = A \oplus B; \tag{12.5}$$

$$C = \bar{A}B. \tag{12.6}$$

В соответствии с выражениями (12.5) и (12.6) логическая схема полувычитателя будет иметь вид, как показано на рисунке 12.6.

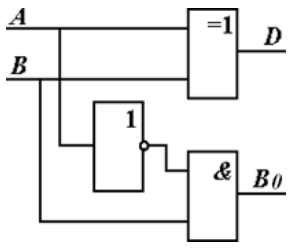


Рисунок 12.6 – Логическая схема полувычитателя

Полный вычитатель. Подобно полному сумматору необходим полный вычитатель для выполнения многобитового вычитания, где заем из предыдущего разряда присутствует. Таким образом, полный вычитатель имеет три входа: A_n (уменьшаемое); B_n (вычитаемое); C_{n-1} (заем от предыдущего разряда), а также два выхода: D_n (разность); C_n (заем).

Таблица истинности полного вычитателя представлена в таблице 12.4.

Таблица 12.4 – Таблица истинности полного вычитателя

Входы			**Выходы**	
A_n	B_n	C_{n-1}	D_n	C_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Таблица истинности и карта Карно для выхода D_n точно такие же, как и для S_n в полном сумматоре, следовательно, получим выражение

$$D_n = A_n \oplus B_n \oplus C_{n-1}. \tag{12.7}$$

Карта Карно для C_n показана на рисунке 12.7.

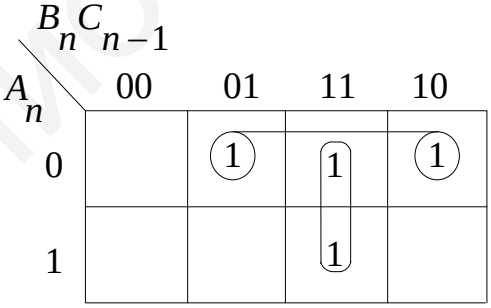


Рисунок 12.7 – Упрощение для C_n с помощью карты Карно

Существует три варианта упрощения выражения для C_n. Два из них дают схему полного сумматора, состоящего из двух полувычитателей и схемы ИЛИ. Однако наиболее интересным представляется третий вариант (см. рисунок 12.7):

$$C_n = (\bar{A}_n \bar{B}_n C_{n-1} + \bar{A}_n B_n \bar{C}_{n-1}) + (\bar{A}_n B_n C_{n-1} + A_n B_n C_{n-1}) = \\ = \bar{A}_n (B_n \oplus C_{n-1}) + B_n C_{n-1}. \tag{12.8}$$

Логическая схема полного вычитателя для этого варианта показана на рисунке 12.8.

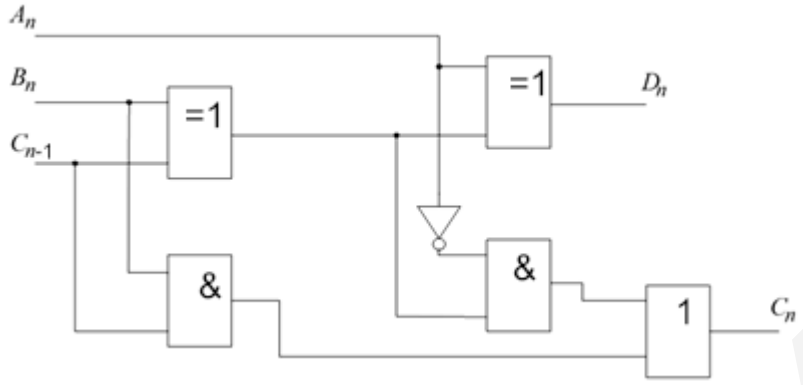


Рисунок 12.8 – Полный вычитатель

Полный вычитатель по сравнению с обычной схемой не состоит из двух полувычитателей, а содержит на один логический элемент (инвертор) меньше, и время распространения сигнала через схемы на один элемент уменьшается, т. е. схема полного вычитателя (см. рисунок 12.8) более быстродействующая по сравнению с обычной схемой полного вычитателя.

Суммирование/вычитание в системе счисления с дополнительным кодом. Вычитание двоичных чисел может осуществляться с использованием полных двоичных вычитателей, однако этот метод используется редко. Взамен этого вычитания обычно используется арифметика в системе счисления с дополнительным кодом и тогда только двоичные сумматоры используются для выполнения операций суммирования и вычитания. Даже с учетом того, что потребуется еще и схема для получения дополнения, этот метод является предпочтительным.

Обычно используется комбинированная суммирующая/вычитающая схема, в которой вид выполняемой операции (суммирование или вычитание) зависит от управляющего сигнала.

Рассмотрим схему (рисунок 12.9), в которой параллельный сумматор соединен таким образом, чтобы выполнять или суммирование, или вычитание.

Когда управляющий сигнал имеет низкий уровень (лог. 0) на входе C_{in}, то и на одном из входов схем Исключающее ИЛИ также лог. 0, поэтому биты числа *B* передаются на входы сумматора без изменения. Схема работает как сумматор.

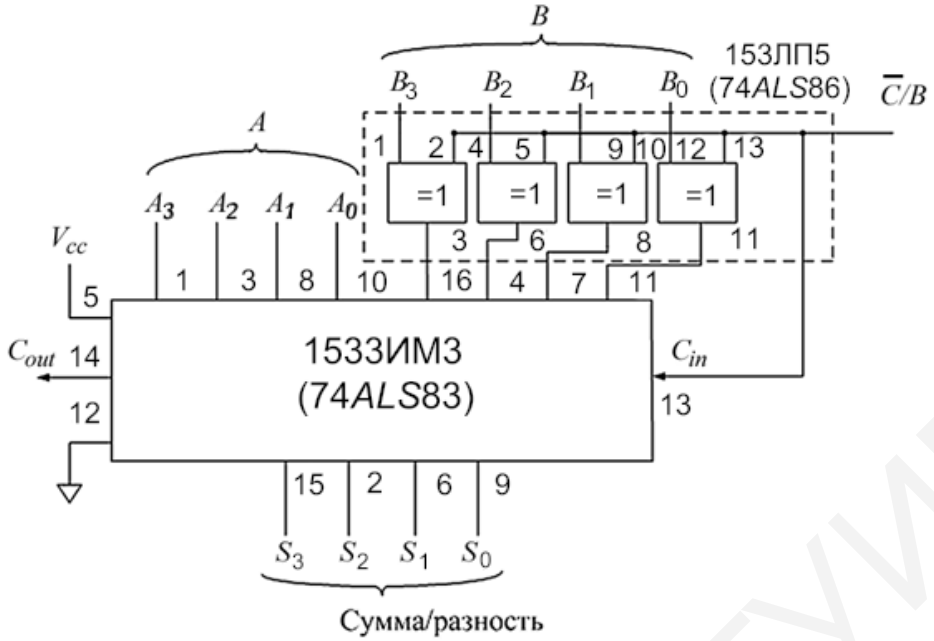


Рисунок 12.9 – Сумматор/вычитатель

Когда управляющий сигнал имеет высокий уровень, на вход C_{in} подается лог. 1 и каждая схема Исключающее ИЛИ инвертирует биты числа *B*. Таким образом осуществляется инвертирование каждого бита числа *B* и прибавление единицы (по входу C_{in}), и в результате осуществляется операция дополнения с числом *B*. При этом суммирование числа *A* с дополнением числа *B* равносильно вычитанию числа *B* из числа *A*.

12.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудование и компоненты: универсальная лабораторная установка *IDL*-800, ИС 1533ЛП5 (74*ALS*86) – четыре логических элемента Исключающее ИЛИ, ИС 1533ЛИ1 (74*ALS*08) – четыре логических элемента 2И, ИС 1533ЛЛ1 (74*ALS*32) – четыре логических элемента 2ИЛИ, ИС 1533ИМ3 (74*ALS*83) – четырехразрядный сумматор, ИС 1533ЛН1 (74*ALS*04) – шесть инверторов.

Задание 1. Исследовать сумматоры. Для этого:

1) используя логические элементы, собрать схему полного сумматора (см. рисунок 12.4). Изменяя состояния входов A_n, B_n, C_{n-1}, исследовать работу полного сумматора, получить таблицу истинности полного сумматора;

2) реализовать на логических элементах схему сумматора с параллельным переносом.

Задание 2. Исследовать вычитатели. Для этого, используя логические элементы, собрать схему полного вычитателя (см. рисунок. 12.8). Изменяя состояния входов A_n, B_n, C_{n-1}, исследовать работу полного вычитателя, получить таблицу истинности полного вычитателя.

Задание 3. Исследовать суммирование/вычитание в дополнительном коде. Для этого собрать схему сумматора/вычитателя (см. рисунок 12.9). Исследовать работу сумматора/вычитателя, используя несколько примеров. Числа *A* и *B* представляются в дополнительном коде. Объяснить результаты экспериментов.

12.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.
3. Таблицы, отражающие результаты исследований.
4. Выводы по результатам исследований.

12.5 Контрольные вопросы

1. Какие комбинационные схемы называются полу-/полным сумматором/ вычитателем?
2. Нарисуйте схему полу-/полного сумматора/вычитателя.
3. Напишите таблицу истинности полу-/полного сумматора/вычитателя.
4. Приведите примеры суммирования/вычитания в дополнительном коде.

13 Лабораторная работа №4

ИССЛЕДОВАНИЕ ПРЕОБРАЗОВАТЕЛЕЙ КОДОВ

13.1 Цель работы

Синтез преобразователей двоичных кодов и исследование их функционирования.

13.2 Теоретические сведения

Для представления информации в цифровых устройствах используются двоичные коды. И хотя любой код, используемый для представления информации, имеет двоичную форму, правильная интерпретация этих кодов возможна, только если этот код известен. Наиболее часто используемые двоичные коды представлены в таблице 13.1.

Таблица 13.1 – Двоичные коды

Десятичные числа	**Прямой двоичный код**				**Код *BCD***				**Код с избытком 3 (*Excess* 3)**				**Код Грея**				**2421**	**7421**
	B_3	B_2	B_1	B_0	*D*	*C*	*B*	*A*	E_3	E_2	E_1	E_0	G_3	G_2	G_1	G_0		
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0000	0000
1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0001	0001
2	0	0	1	0	0	0	1	0	0	1	0	1	0	0	1	1	0010	0010
3	0	0	1	1	0	0	1	1	0	1	1	0	0	0	1	0	0011	0011
4	0	1	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0100	0100
5	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	1	1011	0101
6	0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	1	1100	0110
7	0	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0	1101	1000
8	1	0	0	0	1	0	0	0	1	0	1	1	1	1	0	0	1110	1001
9	1	0	0	1	1	0	0	1	1	1	0	0	1	1	0	1	1111	1010
10	1	0	1	0	–	–	–	–	–	–	–	–	1	1	1	1	–	–
11	1	0	1	1	–	–	–	–	–	–	–	–	1	1	1	0	–	–
12	1	1	0	0	–	–	–	–	–	–	–	–	1	0	1	0	–	–
13	1	1	0	1	–	–	–	–	–	–	–	–	1	0	1	1	–	–
14	1	1	1	0	–	–	–	–	–	–	–	–	1	0	0	1	–	–
15	1	1	1	1	–	–	–	–	–	–	–	–	1	0	0	0	–	–

Прямой двоичный код используется для представления чисел в двоичной системе счисления.

Двоично-десятичный 8-4-2-1 код (*BCD*-код) используется для представления десятичных цифр. Числа 8, 4, 2 и 1 являются весами разрядов. Запись десятичной цифры в коде 8-4-2-1 совпадает с записью двоичных чисел от 0 до 9, а n – разрядное десятичное число и представляется с помощью тетрад, каждая из которых состоит из четырех двоичных разрядов (например, $395_{10} = = 0011\ 1001\ 0101$).

Двоично-десятичный код с избытком 3 (код *Excess*-3) (также используемый для представления десятичных цифр) образуется от соответствующих представлений цифр в *BCD*-коде путем прибавления двоичного числа 0011. Код с избытком 3 является самодополняющим кодом. Правила преобразования прямого кода с избытком 3 в дополнительный с избытком 3 и правила обратного преобразования такие же, как и для двоичного дополнительного кода. Поэтому код с избытком 3 часто удобнее использовать для выполнения арифметических операций. При этом для сложения четырехразрядных кодов можно использовать четырехразрядные двоичные сумматоры.

Код Грея. В коде Грея десятичные числа представлены в двоичном виде таким образом, что представление каждого числа отличается от предыдущего, как и от последующего только в одном бите (разряде).

Рассмотрим схему преобразования *BCD*-кода в двоичный код (рисунок 13.1).

Рисунок 13.1 – Схема преобразования двухразрядного *BCD*-кода в прямой двоичный код

Входами являются две тетрады:

1) D_0, C_0, B_0, A_0, представляющая единицы;
2) D_1, C_1, B_1, A_1, представляющая десятки.

Выходом является семиразрядный двоичный код $b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$. На схеме показаны также веса каждого *BCD*-входа и каждого двоичного выхода. Разряды в двоично-десятичном представлении имеют десятичный вес 8, 4, 2, 1 в

каждой тетраде, однако каждая тетрада отличается от предыдущего весового коэффициента 10 (одна десятичная цифра от предыдущей).

Десятичные веса каждого бита в двоично-десятичном представлении (см. рисунок 13.1) могут быть представлены двоичными эквивалентами (таблица 13.2).

Таблица 13.2 – Таблица *BCD*

***BCD*-биты**	**Десятичные веса**	**Двоичные эквиваленты**						
		B_6	B_5	B_4	B_3	B_2	B_1	B_0
A_0	1	0	0	0	0	0	0	1
B_0	2	0	0	0	0	0	1	0
C_0	4	0	0	0	0	1	0	0
D_0	8	0	0	0	1	0	0	0
A_1	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
C_1	40	0	1	0	1	0	0	0
D_1	80	1	0	1	0	0	0	0

Используя эти веса, можно осуществить преобразование двоично-десятичного кода в двоичный путем двоичного суммирования двоичных эквивалентов тех битов, которые в *BCD*-представлении равны единицам.

Например, преобразуем 01010101 (двоично-десятичное представление десятичного 55) в двоичный эквивалент. Для этого запишем двоичные эквиваленты для всех единиц в двоично-десятичном представлении, а затем просуммируем их (рисунок 13.2).

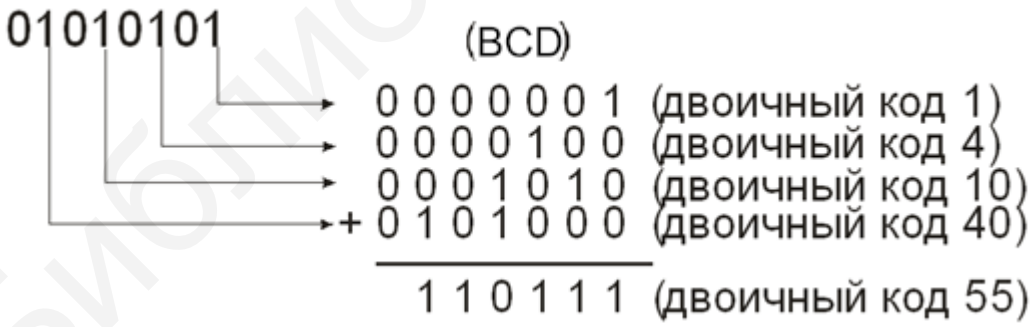


Рисунок 13.2 – Двоичные преобразования

Отсюда следует один из способов реализации преобразователя двоично-десятичного *BCD*-кода в прямой двоичный с помощью двоичных сумматоров. На рисунке 13.3 показана схемная реализация такого преобразователя с помощью двух четырехразрядных двоичных сумматоров 1533ИМ3 (74*ASL*83).

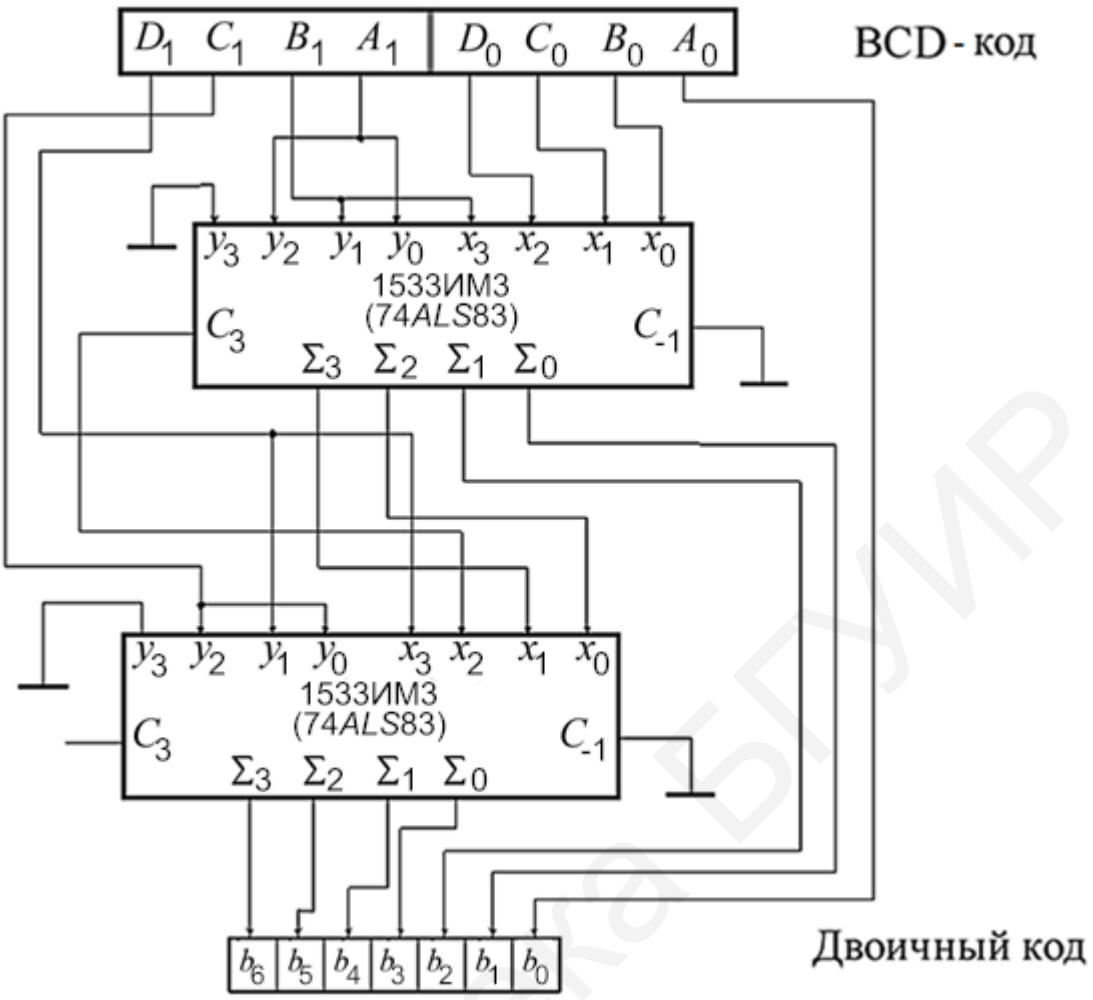


Рисунок 13.3 – Преобразователь двоично-десятичного кода в двоичный

13.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудование и компоненты: универсальная лабораторная установка *IDL*-800, ИС 1533ЛП5 (74*ALS*86) – четыре двухвходовых логических элемента Исключающее ИЛИ, ИС 1533ИМ3 (74*ALS*83) – четырехразрядный двоичный сумматор.

Задание 1. Исследовать преобразователь *BCD*-кода в двоично-десятичный код с избытком 3 (код *Excess*-3). Для этого:

1) разработать и реализовать схему преобразователя, используя четырехразрядный двоичный сумматор;

2) проверить работу преобразователя.

Задание 2. Исследовать работу преобразователя кодов (согласно заданию преподавателя) в среде схемотехнического моделирования *MultiSim*.

13.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.
3. Таблицы, отражающие результаты исследований.
4. Выводы по результатам исследований.

13.5 Контрольные вопросы

1. Что такое преобразователь кодов?
2. Перечислите способы реализации преобразователей кодов.
3. Объясните работу преобразователей кодов, исследуемых в работе.

14 Лабораторная работа №5

СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ С ИСПОЛЬЗОВАНИЕМ МУЛЬТИПЛЕКСОРОВ И ДЕМУЛЬТИПЛЕКСОРОВ

14.1 Цель работы

Изучение мультиплексоров и демультиплексоров, а также синтеза комбинационных схем на основе мультиплексоров и демультиплексоров.

14.2 Теоретические сведения

Традиционный синтез комбинационных схем (КС) включает в себя минимизацию ФАЛ и реализацию минимальной ФАЛ с помощью логических элементов. С помощью этого метода некоторые КС были синтезированы и реализованы как отдельные ИС. Среди них мультиплексоры и демультиплексоры, которые широко представлены как ИС средней степени интеграции. Мультиплексоры и демультиплексоры могут успешно использоваться для реализации различных комбинационных устройств. При этом уменьшается количество требуемых ИС, повышается надежность и снижается стоимость реализации КС.

Мультиплексоры. Мультиплексор (или селектор данных) – это комбинационная схема, которая коммутирует один из 2^m входных сигналов на один выход. Выбор информационного входа, который коммутируется на выход, осуществляется с помощью m адресных входов. Условные обозначения мультиплексора показаны на рисунке 14.1.

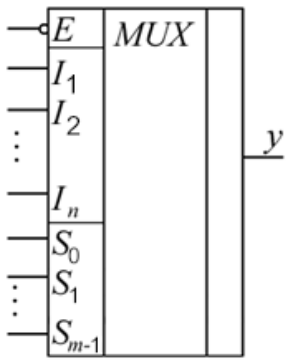


Рисунок 14.1 – Условное обозначение мультиплексора

Для выбора одного из n входов, коммутируемого на выход, требуется группа из m адресных входов, где $2^m = n$. В зависимости от цифрового кода на адресных входах, один из входов выбирается и соединяется с выходом. Обычно **стробируемый (*G*)** или **разрешающий (*EN*)** вход используется для каскадного

соединения мультиплексоров, и этот вход обычно активен при низком уровне, т. е. разрешает работу мультиплексора, когда сигнал на этом входе низкий, лог. 0.

Число информационных входов, коммутируемых на выход Y, составляет $n = 2, 4, 8, 16$. При $n = 4$ мультиплексор имеет размерность 4:1. Это 4-канальный одноразрядный мультиплексор, на выход которого передается один из четырех входных сигналов. Рассмотрим, как построить мультиплексор 4:1, таблица истинности которого приведена в таблице 14.1.

Таблица 14.1 – Таблица истинности мультиплексора

Адресные входы		**Выход**
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Выход Y может быть записан в виде

$$Y = (\bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3). \qquad (14.1)$$

Преобразуем выражение (14.1), используя двойную инверсию и закон Де Моргана:

$$Y = \overline{\overline{\bar{S}_1\bar{S}_0I_0} \cdot \overline{\bar{S}_1S_0I_1} \cdot \overline{S_1\bar{S}_0I_2} \cdot \overline{S_1S_0I_3}}. \qquad (14.2)$$

Выражение (14.2) реализуется, как показано на рисунке 14.2.

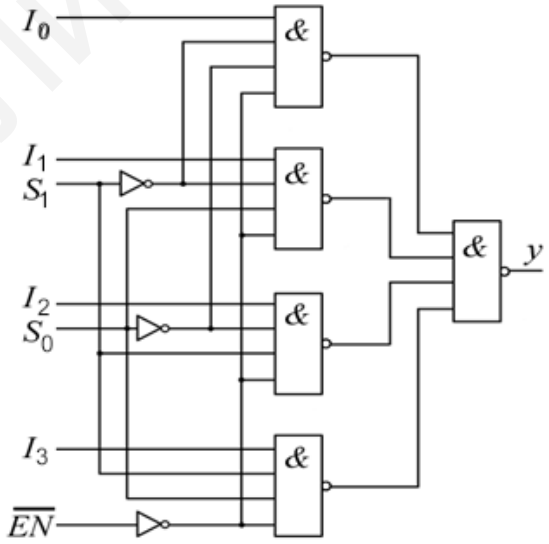


Рисунок 14.2 – Логическая схема мультиплексора 4:1
с разрешающим входом $\overline{EN}$

Промышленностью выпускаются мультиплексоры размерностью 8:1 и 16:1 со стробирующим входом и без него. Также выпускаются двухразрядные мультиплексоры 4:1 и четырехразрядные мультиплексоры 2:1.

Мультиплексоры могут быть использованы как логические элементы для синтеза комбинационных схем. Использование мультиплексоров дает следующие преимущества:

1) не требуется упрощение ФАЛ;
2) минимизируется число требуемых интегральных схем;
3) синтез КС упрощается.

Для реализации КС с использованием мультиплексора требуется представление ФАЛ таблицей истинности или в СДНФ, или в СКНФ. Синтез КС сводится к следующему:

1) определяются десятичные номера каждого минтерма ФАЛ и входы мультиплексора, соответствующие этим номерам, соединяются с лог. 1;
2) все остальные входы соединяются с лог. 0;
3) входные переменные ФАЛ подаются на адресные входы.

Пример 1. Реализовать следующую ФАЛ, используя мультиплексор:

$$f(x_1, x_2, x_3, x_4) = \sum m(0, 1, 2, 5, 7, 8, 11, 14). \qquad (14.3)$$

Решение. Для реализации ФАЛ четырех переменных необходимо выбрать мультиплексор с четырьмя адресными входами (рисунок 14.3).

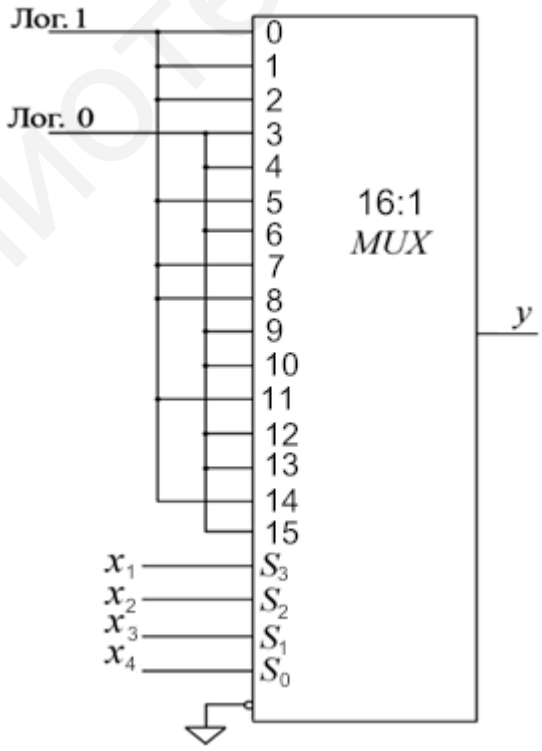


Рисунок 14.3 – Реализация ФАЛ $f(x_1, x_2, x_3, x_4)$

Реализация функции m + 1 переменных на мультиплексорах с *m* адресными входами

Пример 2. Реализовать ФАЛ, заданную таблицей истинности (таблица 14.2).

Таблица 14.2 – Таблица истинности заданного мультиплексора

Входы				**Выход**	–
x_1	x_2	x_3	x_4	**Y**	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	x_4
0	0	1	1	1	
0	1	0	0	0	x_4
0	1	0	1	1	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	0	0
1	0	0	1	0	
1	0	1	0	1	$\bar{x}_4$
1	0	1	1	0	
1	1	0	0	0	$\bar{x}_4$
1	1	0	1	1	
1	1	1	0	1	$\bar{x}_4$
1	1	1	1	0	

ФАЛ четырех переменных может быть реализована с использованием мультиплексора размерностью 8:1 (рисунок 14.4).

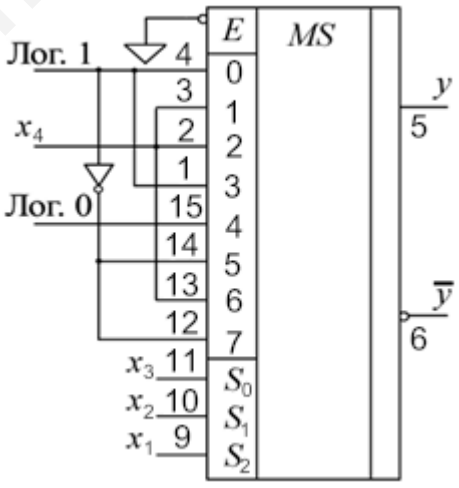


Рисунок 14.4 – Реализация ФАЛ с использованием мультиплексора КР1533КП7 (74*ALS*151)

Переменные ФАЛ x_1, x_2, x_3 подсоединим к адресным входам S_2, S_1, S_0 соответственно. Переменную, не подсоединенную к адресным входам, называют «выделенной». Без выделенной переменной наборы переменных x_1, x_2, x_3 образуют пары. В таблице истинности эти пары выделены пунктирными линиями.

Теперь рассмотрим соотношения между выделенной переменной x_4 и выходом для каждой пары. При этом возможны четыре варианта, когда выход Y не зависит от переменной x_4 и равен 0 или 1 и когда выход Y зависит от переменной x_4 и равен x_4 или $\bar{x}_4$, как отмечено справа от таблицы истинности, (см. таблицу 14.2). Исходя из этого, на информационные входы мультиплексора и подается лог. 0, лог. 1, x_4 или $\bar{x}_4$, (см. рисунок 14.4).

Для синтеза ФАЛ можно использовать карты Карно (рисунок 14.5).

$x_1 x_2$ \ $x_3 x_4$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

а

$x_1 x_2$ \ $x_3 x_4$	00	01	11	10
00	0			1
01	2			3
11	6			7
10	4			5

б

$x_1 x_2$ \ $x_3 x_4$	00	01	11	10
00	1	1	1	0
01	0	1	1	1
11	0	1	0	1
10	0	0	0	1

в

а – карта Карно для синтеза ФАЛ; *б* – карта Карно после объединения парных наборов ФАЛ; *в* – карта Карно заданной ФАЛ

Рисунок 14.5 – Синтез ФАЛ с помощью карт Карно

Парные наборы на ней расположены рядом, т. е. являются соседними. Объединим эти наборы и пометим числом, соответствующим наборам переменных x_1, x_2, x_3. Можно заметить, что парные наборы легко выделяются проведением разделительных линий по переменным x_1, x_2, x_3 (см. рисунок 14.5, *б*). После объединения парных наборов получается карта Карно для трех переменных.

Зададим теперь ФАЛ с помощью карты Карно (см. рисунок 14.5, *в*). Сопоставляя рисунки (см. рисунок 14.5, *а* и *б*), можно заметить, что в клетке с номером 0 $Y = 1$ и, следовательно, $I_0 = 1$. В клетке с номером 1 $Y = x_4$, что дает $I_1 = x_4$. Рассматривая дальше, получим итоговый результат $I_0 = I_3 = 1$, $I_1 = I_2 = I_6 = x_4, I_4 = 0, I_5 = I_7 = \bar{x}_4$

До сих пор мы различали парные наборы по переменной x_4, которая затем подавалась на информационные входы. Однако в качестве «выделенной» может быть взята любая переменная. Более того, для технической реализации, не безразлично какую переменную следует выделять. Это связано с тем, что выбор выделенной переменной определяет количество информационных входов, на которые подаются константы 0, 1. Такие информационные входы не нагружают

предыдущие цепи, поэтому их желательно иметь как можно больше. Очевидно, что для достижения этого в качестве выделенной следует использовать переменную, от которой ФАЛ зависит меньше всего. Последнее можно установить по минимальной дизъюнктивной форме, подсчитав количество вхождений переменной в эту форму как с инверсией, так и без нее.

Увеличение размерности мультиплексора. Максимальный размер мультиплексора, выпускаемого промышленностью 16:1. Мультиплексоры с большим числом входов можно построить из мультиплексоров с меньшим числом входов. Используются два метода, которые можно проиллюстрировать следующими примерами.

Пример 3. Построить мультиплексор размерностью 32:1. Такой мультиплексор может быть построен, как показано на рисунке 14.6, с использованием двух мультиплексоров 16:1.

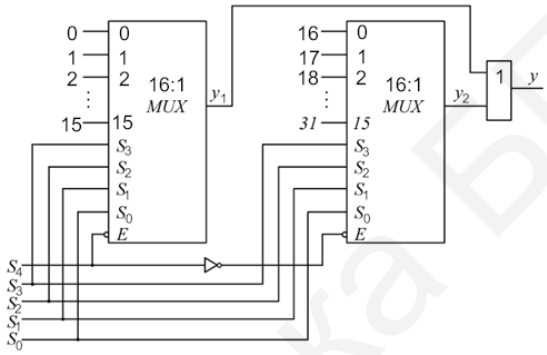


Рисунок 14.6 – Мультиплексор 32:1 с использованием двух мультиплексоров 16:1

Пример 4. Построить мультиплексор размерностью 256:1. Такой мультиплексор может быть построен по древовидной схеме с использованием 17 мультиплексоров 16:1, рисунок 14.7.

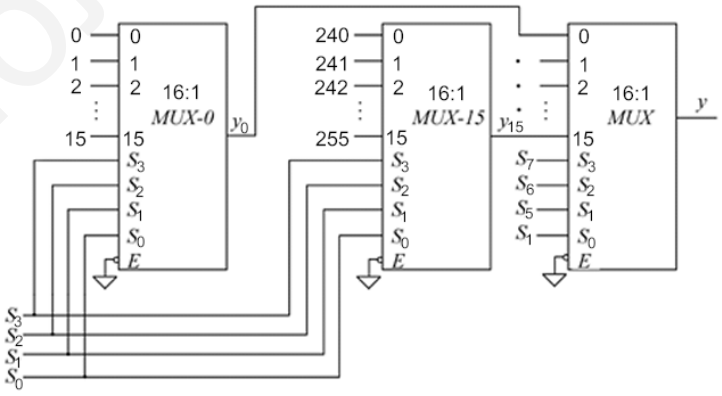


Рисунок 14.7 – Мультиплексор 256:1 с использованием 17 мультиплексоров 16:1

Декодеры/демультиплексоры и их использование в синтезе комбинационных схем. В интегральном исполнении декодеры (дешифраторы) реализуются с использованием элементов И-НЕ, и поэтому на выходах таких декодеров генерируются минтермы в инверсном виде. Большинство интегральных декодеров имеют один или несколько стробирующих или разрешающих входов. Схема декодера 2:4 с разрешающим входом, построенная на элементах И-НЕ, показана на рисунке 14.8.

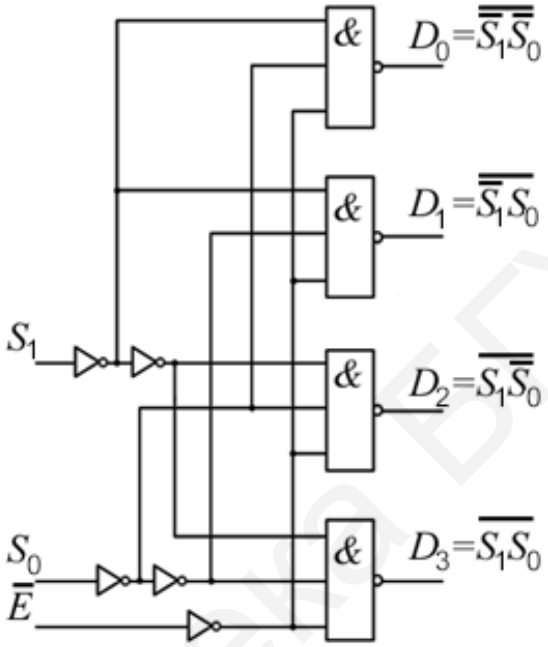


Рисунок 14.8 – Логическая схема декодера 2:4 с разрешающим входом $\bar{E}$

Таблица истинности декодера 2:4 представлена таблицей 14.3.

Таблица 14.3 – Таблица истинности декодера

$\bar{E}$	S_1	S_0	D_0	D_1	D_2	D_3
1	*X*	*X*	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Если разрешающий вход $\bar{E} = 1$, то все выходы декодера равны 1 независимо от значения входов S_1 и S_0. Когда разрешающий вход $\bar{E} = 0$, схема работает, как декодер с инверсными выходами. Условное обозначение схемы показано на рисунке 14.9.

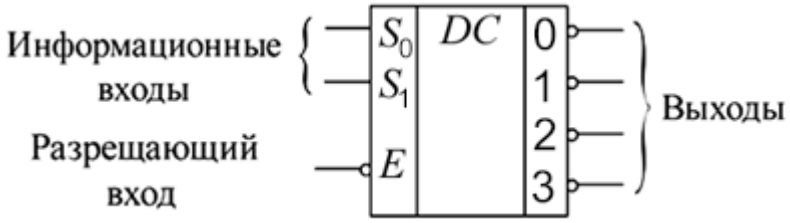


Рисунок 14.9 – Условное обозначение декодера 2:4

Декодер с разрешающим входом может работать как демультиплексор. Демультиплексор – это комбинационная схема, которая принимает информацию на единственный вход и передает эту информацию на один из $n = 2^m$ возможных выходов. Выбор определенного выхода осуществляется с помощью адресных входов. Декодер, который мы рассмотрели, может работать как демультиплексор, если вход $\bar{E}$ использовать как информационный вход, а информационные входы декодера S_1 и S_0 использовать как адресные входы. Условное обозначение демультиплексора показано на рисунке 14.10.

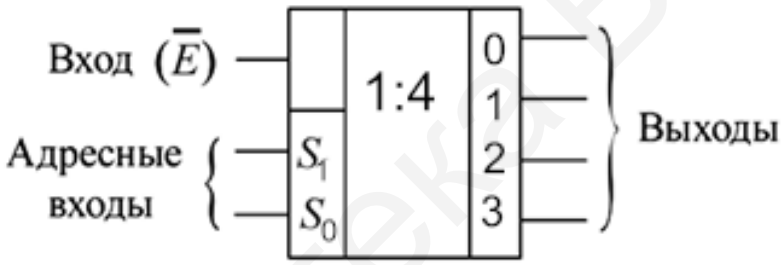


Рисунок 14.10 – Условное обозначение схемы, используемой как демультиплексор

Информация с единственного входа демультиплексора поступает на один из выходов в зависимости от двоичных значений на адресных входах. Это можно проверить, используя таблицу истинности (см. таблицу 14.3).

Поскольку для декодера и демультиплексора используется одна и та же логическая схема, декодер с разрешающим входом называется **декодером/демультиплексором**. Демультиплексор осуществляет операцию, обратную мультиплексору.

Декодеры/демультиплексоры используются в синтезе комбинационных схем. Особенно эти устройства полезны при синтезе КС с несколькими выходами. Декодеры/демультиплексоры в интегральном исполнении возможны как 2:4, 3:8 и 4:16 линий. Выходы таких устройств, как правило, имеют низкий активный уровень.

Рассмотрим использование декодеров/демультиплексоров в синтезе комбинационных схем.

Пример 5. Реализовать схему полного сумматора, используя декодер/демультиплексор.

Решение. Из таблицы истинности для полного сумматора составим выражения:

$$S_n = \bar{A}_n\bar{B}_nC_{n-1} + \bar{A}_nB_n\bar{C}_{n-1} + A_n\bar{B}_n\bar{C}_{n-1} + A_nB_nC_{n-1}; \quad (14.4)$$

$$C_n = \bar{A}_nB_nC_{n-1} + A_n\bar{B}_nC_{n-1} + A_nB_n\bar{C}_{n-1} + A_nB_nC_{n-1}. \quad (14.5)$$

Используя двойную инверсию и закон Де Моргана, преобразуем выражения (14.4) и (14.5) в (14.6) и (14.7) соответственно:

$$S_n = \overline{\overline{\bar{A}_n\bar{B}_nC_{n-1}} \cdot \overline{\bar{A}_nB_n\bar{C}_{n-1}} \cdot \overline{A_n\bar{B}_n\bar{C}_{n-1}} \cdot \overline{A_nB_nC_{n-1}}}; \quad (14.6)$$

$$C_n = \overline{\overline{\bar{A}_nB_nC_{n-1}} \cdot \overline{A_n\bar{B}_nC_{n-1}} \cdot \overline{A_nB_n\bar{C}_{n-1}} \cdot \overline{A_nB_nC_{n-1}}}. \quad (14.7)$$

Или функция S_n может быть записана как $S_n = \overline{D_1D_2D_4D_7}$, а функция C_n – как $C_n = \overline{D_3D_5D_6D_7}$.

Для реализации полного сумматора необходимо использовать декодер/демультиплексор 3:8 линий, например КР1533ИД7 (*SN*74*ALS*138). Схема полного сумматора показана на рисунке 14.11.

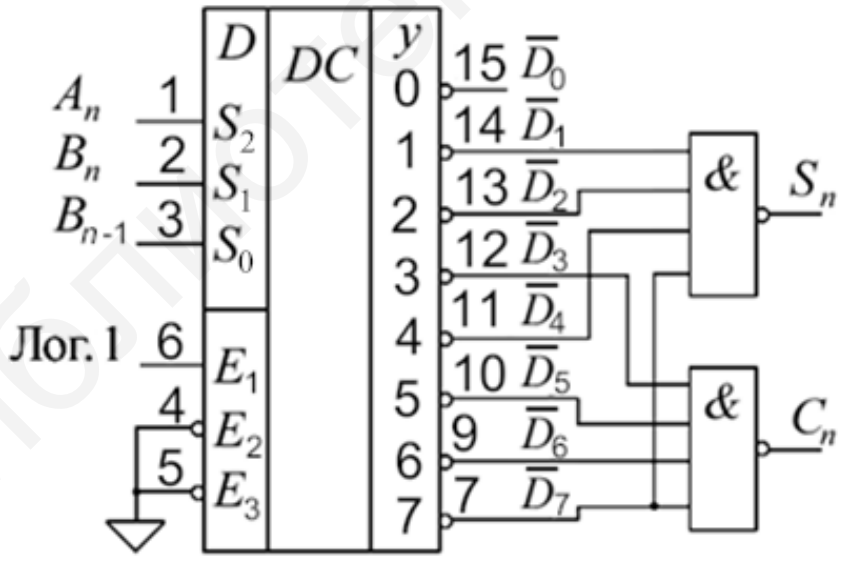
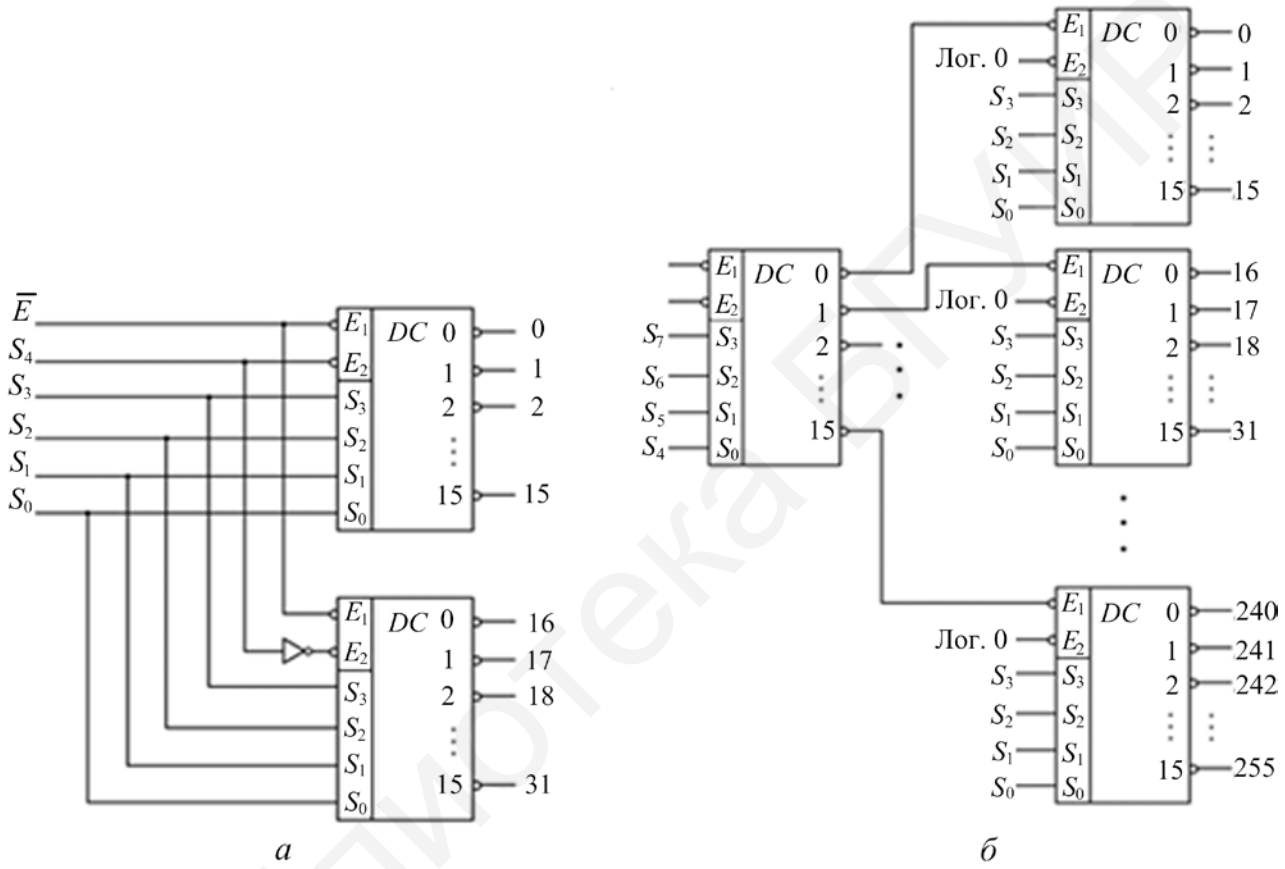


Рисунок 14.11 – Реализация полного сумматора с использованием декодера/демультиплексора

При синтезе КС с использованием мультиплексоров дополнительные элементы не используются, в то время как их необходимо использовать при синтезе КС на основе декодеров/демультиплексоров. Тем не менее, несмотря на

это декодеры/демультиплексоры являются более экономичными в случаях синтеза многовыходовых КС. В таких случаях один мультиплексор требуется для реализации каждого выхода, а при использовании декодера/ демультиплексора – только дополнительные схемы И-НЕ.

Увеличение размерности декодеров/демультиплексоров. Поскольку ИС КР1533ИД3 является самым большим (4:16 линий) декодером/ демультиплексором, то для увеличения размерности декодеров/ демультиплексоров используются методы, представленные на рисунке 14.12.



а – построение 5:32 линии декодера/демультиплексора с использованием двух 4:16 линий декодера/демультиплексора; *б* – построение 8:256 линий декодера/демультиплексора с использованием 17 декодеров/демультиплексоров 4:16 линий

Рисунок 14.12 – Декодеры/демультиплексоры

14.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудование и компоненты: универсальная лабораторная установка *IDL*-800, ИС 1533КП7 (74*ALS*151), ИС 1533ИД7 (74*ALS*138), ИС 1533ЛА1 (74*ALS*20), ИС 1533ЛН1 (74*ALS*04).

Задание 1. Исследовать работу мультиплексора. Для этого:

1) установить ИС 1533КП7 на наборной панели *IDL*-800;

2) вывод 16 ИС соединить с источником питания +5*V*, а вывод 8 – с общей шиной установки;

3) изменяя значения адресных входов от 000 до 111 и входные данные (0,1), проследить за состоянием выходов мультиплексора и проверить работу мультиплексора. Результаты отразить в таблице истинности мультиплексора.

Задание 2. Синтезировать и реализовать, используя мультиплексор ИС 1533КП7, следующие ФАЛ:

$$f(x_1, x_2, x_3) = \sum m(\dots); \quad (14.8)$$

$$f(x_1, x_2, x_3, x_4) = \sum m(\dots). \quad (14.9)$$

Задание 3. Исследовать работу декодера/демультиплексора. Для этого:

1) установить ИС 1533ИД7 на наборной панели *IDL*-800;

2) вывод 16 ИС соединить с источником питания +5*V*, а вывод 8 – с общей шиной установки;

3) изменяя значения адресных входов от 000 до 111, проследить за состояниями выходов. Результаты наблюдений отразить в таблице истинности и получить выражения для выходов D_0–D_7.

Задание 4. Исследовать работу полного сумматора на основе декодера/демультиплексора. Для этого:

1) собрать схему полного сумматора (см. рисунок 14.12);

2) исследовать работу полного сумматора, результаты отразить в таблице истинности.

Задание 5. Синтезировать и исследовать работу полного вычитателя на основе декодера/демультиплексора.

Задание 6. Синтезировать и исследовать работу компаратора для сравнения двухразрядных двоичных чисел.

14.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.
3. Таблицы, отражающие результаты исследований.
4. Выводы по результатам исследований.

14.5 Контрольные вопросы

1. Что такое мультиплексор? Объясните его работу.
2. Каким образом мультиплексоры используются для синтеза КС?

3. В чем заключаются основные преимущества использования мультиплексоров при реализации ФАЛ?

4. Что такое декодер/демультиплексор? Объясните его работу.

5. Каким образом декодеры/демультиплексоры используются для реализации ФАЛ?

6. Почему декодеры/демультиплексоры наиболее удобны для реализации многовыходовых ФАЛ?

15 Лабораторная работа №6

ТРИГГЕРЫ

15.1 Цель работы

Изучение основ теории, методов логического синтеза и функционирования основных типов триггеров.

15.2 Теоретические сведения

Триггером называется устройство, имеющее два устойчивых состояния и способное под действием управляющих сигналов скачкообразно переходить из одного состояния в другое. Одно состояние называется единичным, а второе – нулевым. В общем случае триггер имеет два выхода – прямой Q и инверсный $\overline{Q}$, поскольку логическое состояние одного выхода всегда инверсно логическому состоянию другого. Состояние триггера определяется логическим уровнем на прямом выходе. Если на прямом выходе имеется потенциал, соответствующий лог. 1, то триггер находится в единичном состоянии, или говорят, что триггер установлен (при этом потенциал на инверсном выходе соответствует лог. 0). И если на прямом выходе имеется потенциал, соответствующий лог. 0, то триггер находится в нулевом состоянии, или говорят, что триггер сброшен (и при этом потенциал на инверсном выходе соответствует лог. 1).

В качестве основных классификационных признаков используются функциональный признак и способ записи информации.

По функциональному признаку, т. е. по виду характеристического уравнения, связывающего логические переменные на входах и выходах триггера в момент срабатывания t_n и после срабатывания t_{n+1}, различают триггеры *RS*-, *D*-, *JK*-, *T*- и других типов.

По способу записи информации триггеры делятся на две категории:

1) асинхронные (нетактируемые);
2) синхронные (тактируемые).

В асинхронных триггерах запись информации происходит под действием изменений входных сигналов с момента подачи их на информационные входы. В синхронных триггерах запись информации происходит только при подаче сигнала синхронизации.

По способу синхронизации триггеры подразделяются на три категории, в зависимости от того какие параметры синхросигнала используются для записи информации:

1) со статическим управлением;
2) двухступенчатые, управляемые синхроимпульсом;
3) с динамическим управлением.

Синхронный триггер со статическим управлением воспринимает информационные сигналы, когда синхросигнал достигает своего активного уровня. Характерной особенностью этого типа является то, что смена управляющего сигнала в течение времени действия импульса синхронизации вызывает новые срабатывания триггеров, т. е. синхронные триггеры со статическим управлением при активном уровне синхросигнала ведут себя подобно асинхронным.

Асинхронные триггеры и синхронные триггеры со статическим управлением имеют ограниченное применение. Например, эти триггеры не могут использоваться в счетчиках или регистрах сдвига. В зарубежной литературе триггерные устройства подразделяются на два типа – *latch*(*s*) и *flip-flop*(*s*) (*FF*). Асинхронные триггеры и синхронные со статическим управлением относятся к типу *latch*, а двухступенчатые триггеры, управляемые синхроимпульсом, и триггеры с динамическим управлением относятся к типу *FF*. Двухступенчатые триггеры, управляемые импульсом, и триггеры с динамическим управлением являются более универсальными.

Двухступенчатые триггеры, управляемые импульсом, воспринимают информационные сигналы, когда синхросигнал изменяет свое состояние с низкого на высокое, а затем снова на низкое, т. е. управляются импульсом.

Синхронный триггер с динамическим управлением воспринимает информационные сигналы только в момент действия положительного перехода (переход 0→1=↑) или в момент действия отрицательного перехода (переход 1→0=↓) синхроимпульса. Вход триггера *C* (или *CK*) называется прямым динамическим, если переключение триггера осуществляется положительным перепадом импульса синхронизации. Вход называется инверсным динамическим, если переключение триггера осуществляется отрицательным перепадом синхросигнала. Характерной особенностью триггеров с динамическим управлением является то, что в остальное время импульса синхронизации триггер не реагирует на информационные сигналы и остается в прежнем состоянии независимо от уровня синхросигнала.

В двухступенчатых триггерах переход в новое состояние происходит после окончания действия синхроимпульса.

Асинхронный *RS*-триггер (*SR-latch*) – это устройство с двумя устойчивыми состояниями, имеющее два входа *S* (*Set*-установка) и *R* (*Reset*-сброс) и два выхода Q и $\bar{Q}$. Асинхронный триггер функционирует в соответствии с таблицей истинности (таблица 15.1).

Таблица 15.1 – Таблица истинности для асинхронного триггера

t_n		t_{n+1}		**Режим работы**
Q_n	S_n	R_n	Q_{n+1}	
1	**2**	**3**	**4**	**5**
0	0	0	0	Хранение информации
1	0	0	1	

1	2	3	4	5
0	1	0	1	Установка лог. 1
1	1	0	1	
0	0	1	0	Установка лог. 0
1	0	1	0	
0	1	1	X	Неопределенность
1	1	1	X	

Если входы S_n и R_n находятся в состоянии лог. 0, то триггер не изменяет свое состояние, т. е. триггер хранит один бит информации.

Если $S_n = 1$ и $R_n = 0$, то независимо от того, в каком состоянии триггер находился, следующее состояние триггера $Q_{n+1} = 1$.

Если $S_n = 0$ и $R_n = 1$, то независимо от того, в каком состоянии находился триггер, следующее его состояние $R_n = 0$.

Комбинация входных сигналов $R_n = S_n = 1$ является неопределенной, поскольку триггер после воздействия на входах активных уровней может равновероятно перейти как в нулевое, так и в единичное состояние. Поэтому одновременная подача активных уровней на входы *S* и *R* не допускается.

Работа *RS*-триггера также может быть представлена функцией на карте Карно и логическим уравнением. Входные и выходные переменные триггера в момент срабатывания t_n и после срабатывания t_{n+1} можно представить функциональной зависимостью:

$$Q_{n+1} = f(S_n, R_n, Q_n), \qquad (15.1)$$

где R_n, S_n– состояние информационных входов;

Q_n– значение выходного сигнала триггера в момент времени t_n;

Q_{n+1} – значение выходного сигнала триггера в момент времени t_{n+1}.

Функция алгебры логики, (см. таблицу 15.1) может быть представлена картой Карно. Значения ФАЛ, показанные значком «х», указывают на то, что данная ФАЛ является недоопределенной. При минимизации ФАЛ мы можем ее доопределить, так как нам это удобно, чтобы получить минимальную форму (рисунок 15.1).

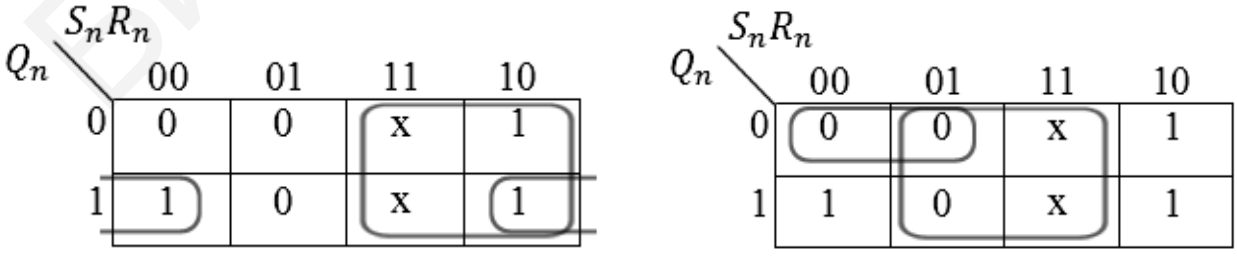


Рисунок 15.1 – Карты Карно для минимизации выходных функций *RS*-триггера, которые используются для построения *RS*-триггера

В результате минимизации для логических функций прямое и инверсное выражения будут иметь вид

$$Q_{n+1} = S_n + Q_n\bar{R}_n; \tag{15.2}$$

$$\bar{Q}_{n+1} = R_n + \bar{Q}_n\bar{S}_n. \tag{15.3}$$

Применяя инверсию и закон де Моргана, выражения (15.2) и (15.3) можно преобразовать следующим образом:

$$\bar{Q}_{n+1} = \overline{\overline{S_n}\overline{\bar{Q}_n + R_n}}; \tag{15.4}$$

$$\bar{Q}_{n+1} = \overline{\overline{S_n}\overline{Q_n + R_n}}. \tag{15.5}$$

Выражения (15.4) и (15.5) используются для реализации *RS*-триггера на элементах ИЛИ-НЕ.

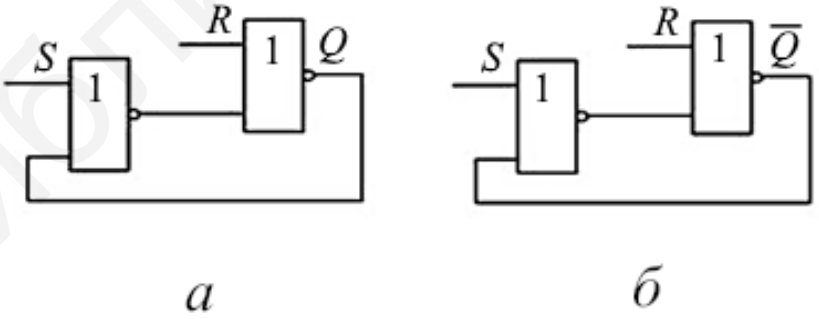
Применяя двойную инверсию и закон де Моргана, выражения (15.2) и (15.3) можно преобразовать следующим образом:

$$Q_{n+1} = \overline{\bar{S}_n\overline{Q_n\bar{R}_n}}; \tag{15.6}$$

$$\bar{Q}_{n+1} = \overline{\bar{R}_n\overline{\bar{Q}_n\bar{S}_n}}. \tag{15.7}$$

Выражения (15.6) и (15.7) используются для реализации *RS*-триггера на элементах И-НЕ.

Реализация *RS*-триггера на элементах ИЛИ-НЕ показана на рисунке 15.2.



а – результат неинверсный; *б* – результат инверсный

Рисунок 15.2 – Реализация *RS*-триггера на элементах ИЛИ-НЕ

Обычно *RS*-триггер изображается с расположением элементов ИЛИ-НЕ, как показано на рисунке 15.3.

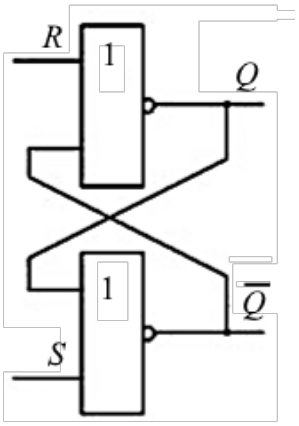


Рисунок 15.3 – *RS*-триггеры на элементах ИЛИ-НЕ

На рисунке 15.4 показано условное изображение *RS*-триггера.

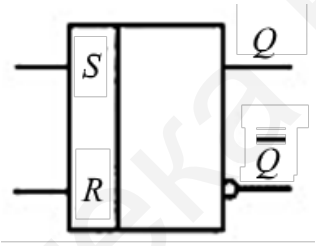


Рисунок 15.4 – Условное изображение *RS*-триггера на элементах ИЛИ-НЕ

При анализе работы *RS*-триггера и при синтезе других триггеров используется таблица переходов *RS*-триггера (таблица 15.2), которая определяет комбинации входных сигналов, необходимых для того или иного перехода триггера, т. е. триггер в момент времени t_n находится в каком-то состоянии (Q_n= 0 или Q_{n+1}=1), и определяются состояния входов, чтобы триггер перешел в определенное следующее состояние Q_{n+1}.

Таблица 15.2 – Таблица переходов

Настоящее состояние	**Следующее состояние**	**Требуемые входы**	
$\boldsymbol{Q_n}$	$\boldsymbol{Q_{n+1}}$	$\boldsymbol{S_n}$	$\boldsymbol{R_n}$
0	0	0	*X*
0	1	1	0
1	0	0	1
1	1	*X*	0

На рисунке 15.5 приведены временные диаграммы, поясняющие принцип работы асинхронного *RS*-триггера (см. рисунок 15.2).

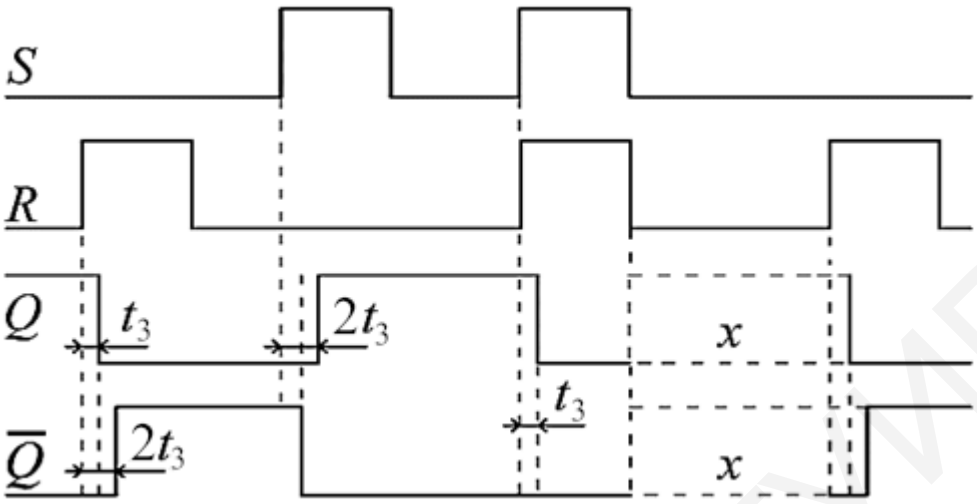


Рисунок 15.5 – Временные диаграммы работы асинхронного *RS*-триггера

Для устойчивого функционирования триггера длительность сигнала на входах *R* и *S* должна быть не меньше времени переключения триггера, $t_n \geq t_з$.

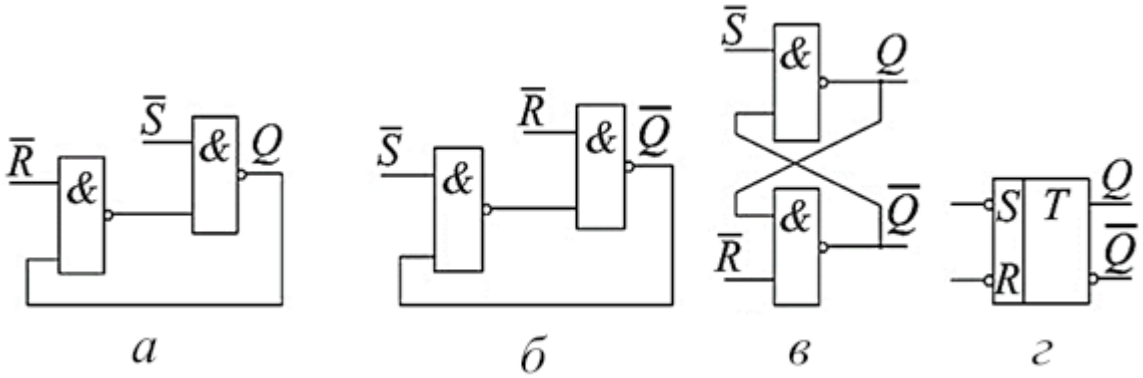
Информационные сигналы должны поступать на входы поочередно и только после окончания переходных процессов в триггере, тогда максимальная частота переключения триггера определяется формулой

$$f_{\max} = \frac{1}{2t_з}. \qquad (15.8)$$

Однако при f_{max} длительность выходных сигналов не будет превышать $t_{з.ср}$. Такие сигналы не являются достаточными для надежной передачи информации в логическую цепь, поэтому максимальная рабочая частота триггера, $f_р$ определяется формулой

$$f_р = \frac{1}{2t_{з.ср}}. \qquad (15.9)$$

Асинхронный *RS*-триггер на элементах И-НЕ. Воспользуемся выражениями (15.6) и (15.7), реализуем *RS*-триггер на элементах И-НЕ. Эта реализация показана на рисунке 15.6.



а, *б* – построение *RS*-триггера; *в* – изображение триггера на элементах И-НЕ; *г* – условное изображение $\overline{RS}$-триггера

Рисунок 15.6 – Асинхронный *RS*-триггер

RS-триггер на элементах И-НЕ (см. рисунок 15.6) называется асинхронным триггером с инверсными входами. Характеристическая таблица или таблица истинности такого триггера представлена таблицей 15.3, а таблица переходов – таблицей 15.4.

Таблица 15.3 – Таблица истинности триггера на элементах И-НЕ

$\frac{t_n}{Q_n \overline{S}_n \overline{R}_n}$			$\frac{t_{n+1}}{Q_{n+1}}$
Q_n	$\overline{S}_n$	$\overline{R}_n$	
0	1	1	0
1	1	1	1
0	0	1	1
1	0	1	1
0	1	0	0
1	1	0	0
0	0	0	*X*
1	0	0	*X*

Таблица 15.4 – Таблица переходов *RS*-триггера

Настоящее состояние	**Следующее состояние**	**Требуемые выходы**	
Q_n	Q_{n+1}	$\overline{S}_n$	$\overline{R}_n$
0	0	1	*X*
0	1	0	1
1	0	1	0
1	1	*X*	1

На рисунке 15.7 показаны временные диаграммы, поясняющие принцип работы асинхронного *RS*-триггера на элементах И-НЕ.

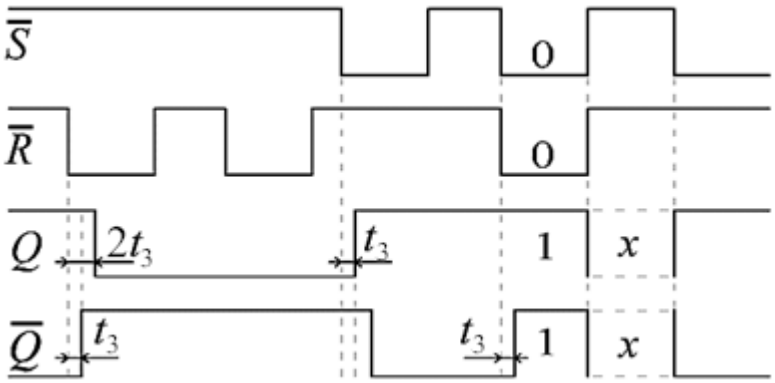
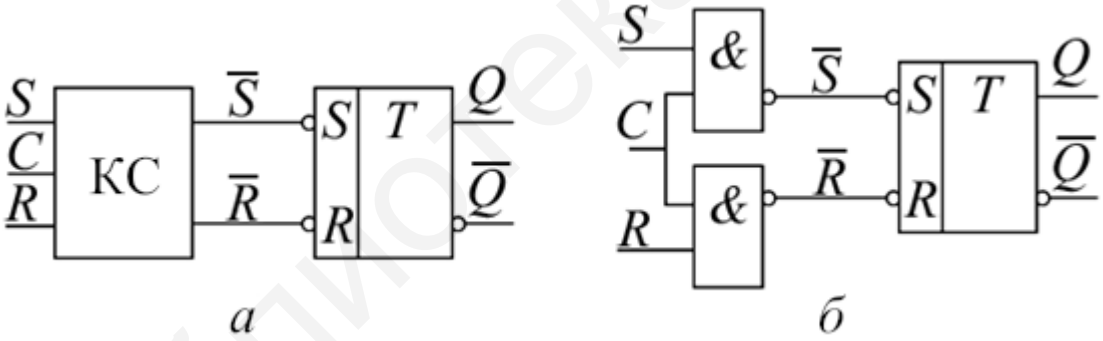


Рисунок 15.7 – Временные диаграммы работы $\overline{RS}$-триггера

Синхронный *RS*-триггер (*SR-latch*) со статическим управлением. Асинхронный *RS*-триггер является элементарной запоминающей ячейкой и используется при синтезе других более сложных триггеров. В том числе синхронный *RS*-триггер со статическим управлением может быть синтезирован на основе базового асинхронного *RS*-триггера с инверсными входами. В этом случае блок-схема синхронного *RS*-триггера со статическим управлением состоит из асинхронного *RS*-триггера и комбинационной схемы (КС), как показано на рисунке 15.8, *а*.



а – синхронный *RS*-триггер на основе асинхронного *RS*-триггера;
б – синхронный *RS*-триггер на основе элементов И-НЕ

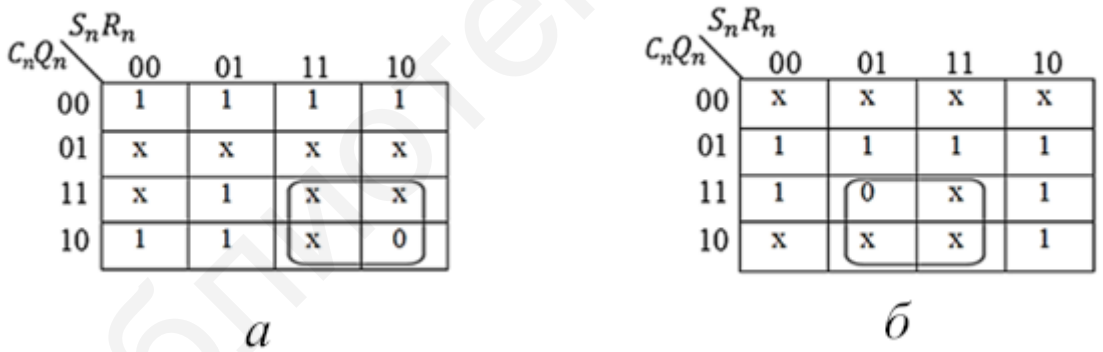
Рисунок 15.8 – Синхронный *RS*-триггер со статическим управлением

Задачей синтеза является определение функции $\bar{S}_n$ и $\overline{\mathrm{R}}_\mathrm{n}$ и построение КС, которая соответствующим образом управляет работой асинхронного $\overline{RS}$-триггера. Для этого построим таблицу истинности функций $\bar{S}_n$и $\overline{\mathrm{R}}_\mathrm{n}$ (таблица 15.5) и минимизируем эти функции с помощью карт Карно. Таблицу истинности для функций $\bar{S}_n$ и $\overline{\mathrm{R}}_\mathrm{n}$ строим на основании таблицы истинности синхронного *RS*-триггера и таблицы переходов асинхронного *RS*-триггера.

Таблица 15.5 – Таблица истинности синхронного *RS*-триггера

C_n	Q_n	S_n	R_n	Q_{n+1}	$\bar{S}_n$	$\bar{Q}_n$
0	0	0	0	0	1	*X*
0	1	0	0	1	*X*	1
0	0	1	0	0	1	*X*
0	1	1	0	1	*X*	1
0	0	0	1	0	1	*X*
0	1	0	1	1	*X*	1
0	0	1	1	0	1	*X*
0	1	1	1	1	*X*	1
1	0	0	0	0	1	*X*
1	1	0	0	1	*X*	1
1	0	1	0	1	0	1
1	1	1	0	1	*X*	1
1	0	0	1	0	1	*X*
1	1	0	1	0	1	0
1	0	1	1	*X*	*X*	*X*
1	1	1	1	*X*	*X*	*X*

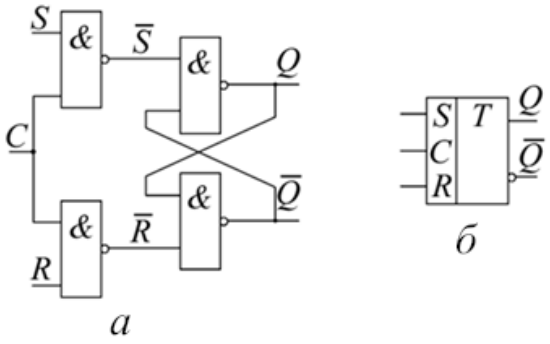
На основании выражений для $\bar{S}$ и $\bar{R}$ блок-схема (рисунок 15.9, *а*) преобразуется в логическую схему синхронного триггера со статическим управлением (рисунок 15.9, *б*).



а – карта Карно на основании выражений для $\bar{S}$ и $\bar{R}$; *б* – карта Карно для синхронного триггера со статическим управлением

Рисунок 15.9 – Карты Карно

Полная логическая схема синхронного триггера на элементах И-НЕ и условное обозначение этого триггера показаны на рисунке 15.10.



а – синхронный *RS*-триггер; *б* – условное обозначение синхронного *RS*-триггера

Рисунок 15.10 – Синхронный *RS*-триггер

***D*-триггер** имеет информационный вход *D* (*data*, *delay*) и вход синхронизации *C*. Триггер принимает информационные сигналы по разрешению синхросигнала и повторяет их на выходе с некоторой задержкой. Синтез синхронного *D*-триггера (*D-latch*) осуществим на основе базового асинхронного триггера. Блок-схема такого триггера показана на рисунке 15.11.

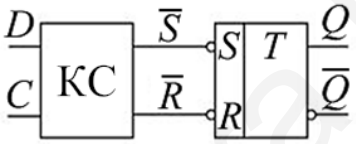


Рисунок 15.11 – Синхронный *D*-триггер со статическим управлением

Из блок-схемы (см. рисунок 15.11) очевидно, что для получения логической схемы синхронного *D*-триггера со статическим управлением, необходимо определить комбинационную схему (КС) генерирующую функции $\bar{S}$ и $\bar{R}$ такими, чтобы базовый *RS*-триггер функционировал как *D*-триггер. Для получения КС составим таблицу истинности для функций $\overline{\mathrm{S}}_{\mathrm{n}}$ и $\bar{R}_n$, на основе таблицы истинности синхронного *D*-триггера со статическим управлением и таблицы переходов асинхронного *RS*-триггера (таблица 15.6).

Таблица 15.6 – Таблица переходов *D*-триггера

C_n	D_n	Q_n	Q_{n+1}	$\overline{S_n}$	$\overline{R_n}$
1	0	0	0	1	*X*
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	1	*X*	1
0	0	0	0	1	*X*
0	0	1	1	*X*	1
0	1	0	0	1	*X*
0	1	1	1	*X*	1

С помощью карт Карно минимизируем выражения для $\bar{S}_n$ и $\bar{R}_n$ (рисунок 15.12).

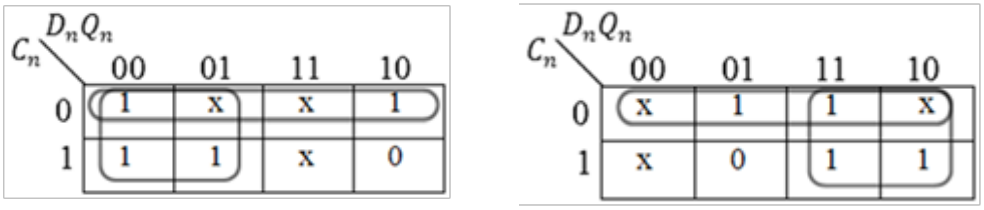


Рисунок 15.12 – Карты Карно для $\bar{S}_n$ и $\bar{R}_n$

Используя выражения для $\bar{S}$ и $\bar{R}$, построим логическую схему синхронного *D*-триггера со статическим управлением (рисунок 15.13).

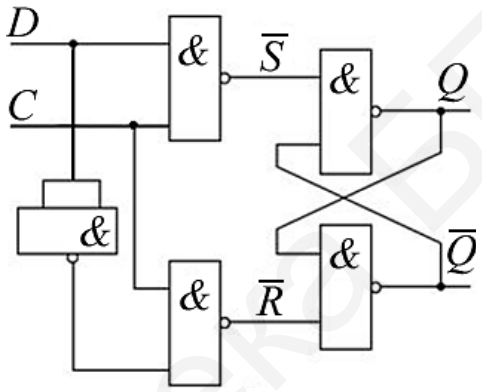


Рисунок 15.13 – *D*-триггер со статическим управлением

Логическая схема синхронного *D*-триггера со статическим управлением может быть упрощена, если при построении КС использовать не минимальное выражение для $\bar{R}_n$, а как показано на рисунке 15.14.

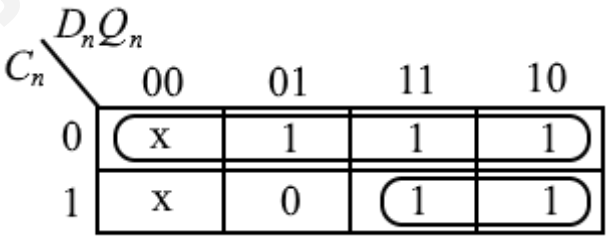


Рисунок 15.14 – Карта Карно для $\bar{R}_n$

Используем для построения триггера выражения

$$\bar{S}_n = \overline{D_n C_n}; \qquad (15.10)$$

$$\bar{R}_n = \overline{C_n \overline{D_n C_n}}. \qquad (15.11)$$

Получается логическая схема синхронного *D*-триггера со статическим управлением, показанная на рисунке 15.15.

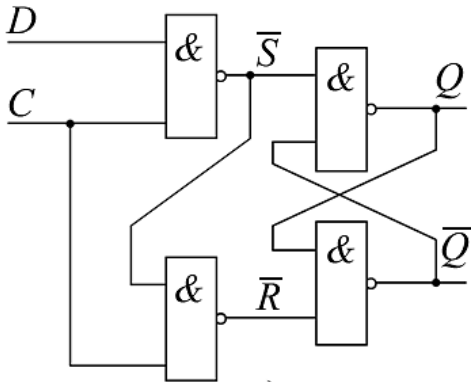
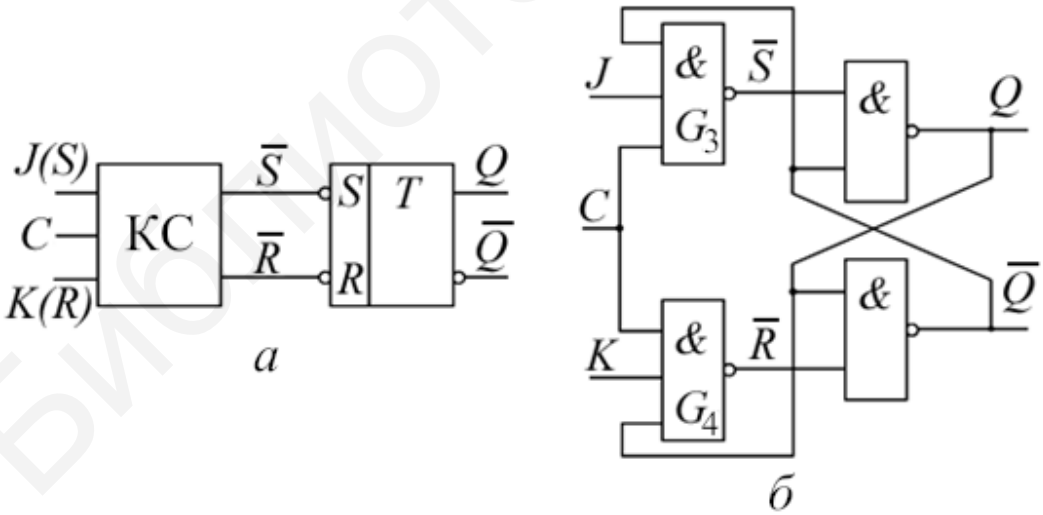


Рисунок 15.15 – Оптимальная схема *D*-триггера со статическим управлением

***JK*-триггеры.** *JK*-триггер – это модернизированный *RS*-триггер, в котором неопределенное состояние *RS*-триггера доопределено таким образом, что при $J = K = 1$ триггер переключается в противоположное состояние. При этом вход *J* соответствует входу *S*, а вход *K* – входу *R*.

Рассмотрим синтез *JK*-триггера (рисунок 15.16, *б*) со статическим управлением, используя базовый асинхронный *RS*-триггер. Блок-схема триггера показана на рисунке 15.16, *а*.



а – блок-схема; *б* – структура *JK*-триггера

Рисунок 15.16 – Синхронный *JK*-триггер со статическим управлением

Для синтеза *JK*-триггера со статическим управлением составим таблицу истинности для функций $\bar{S}$ и $\bar{R}$ (таблица 15.7).

Таблица 15.7 – Таблица истинности *JK*-триггера

C_n	Q_n	J_n	K_n	Q_{n+1}	$\overline{S_n}$	$\overline{R_n}$
1	0	0	0	0	1	X
1	1	0	0	1	X	1
1	0	1	0	1	0	1
1	1	1	0	1	X	1
1	0	0	1	0	1	X
1	1	0	1	0	1	0
1	0	1	1	1	0	1
1	1	1	1	0	1	0
0	0	0	0	0	1	X
0	1	0	0	1	X	1
0	0	1	0	0	1	X
0	1	1	0	1	X	1
0	0	0	1	0	1	X
0	1	0	1	1	X	1
0	0	1	1	0	1	X
0	1	1	1	1	X	1

Минимизируем функции $\bar{S}_n$ и $\bar{R}_n$ с помощью карт Карно и преобразуем их к форме, удобной для реализации, с помощью элементов И-НЕ (рисунок 15.17).

а – карта Карно для Q_n; *б* – карта Карно для $\bar{Q}_n$

Рисунок 15.17 – Карты Карно, поясняющие синтез *JK*-триггера

Рассмотрим работу синхронного *JK*-триггера со статическим управлением, когда $J = K = 1$ и на вход синхронизации поступают синхроимпульсы. Когда на входе C уровень лог. 0, на выходах логических элементов G_3 и G_4 уровни лог. 1 и асинхронный триггер на элементах G_1 и G_2 сохраняет свое состояние. Когда на вход C поступает импульс синхронизации, т. е. $C = 1$, то импульс будет передаваться через один из логических элементов G_3 или G_4, вход которого, соединенный с выходом триггера, будет в данный момент равен лог. 1. Если $Q = 1$, выход логического элемента G_4 становится равным нулю, когда поступает импульс синхронизации и триггер обнуляется. Если $\bar{Q} = 1$, выход G_3 становится равным нулю, при подаче импульса синхронизации и триггер устанавливается. В любом случае состояние триггера изменяется.

Однако, когда $J = K = 1$, $Q = 0$ и импульс синхронизации действует на входе, то после временного интервала Δt, равного времени задержки распространения сигнала через два элемента И-НЕ (G_3 и G_4), выход триггера изменится на $Q = 1$. Теперь мы имеем $J = K = 1$ и $Q = 1$, и после другого временного интервала Δt выход будет изменяться на $Q = 0$. Следовательно, мы можем сделать вывод, что в течение длительности импульса синхронизации $t_и$ схема (см. рисунок 15.10, *б*) имеет неустойчивое состояние, т. е. находится в автоколебательном режиме, а после окончания импульса синхронизации состояние триггера будет неизвестно. Это говорит о том, что синхронный *JK*-триггер со статическим управлением не может быть реализован на одной элементарной запоминающей ячейке.

Казалось, этой ситуации можно было бы избежать, если бы $t_и < \Delta t < T$. Однако выполнить это невозможно в виду очень малого времени задержки распространения сигнала в ИС.

Поэтому, практически используемыми *JK*-триггерами являются двухступенчатые триггеры, управляемые синхроимпульсом, и триггеры с динамическим управлением.

Управляемые синхроимпульсом триггеры строятся по двухступенчатой конфигурации *M-S* (*Master-Slave*). *JK*-триггер типа *M-S* состоит из каскада двух *RS*-триггеров с обратной связью с выхода второго на вход первого (рисунок 15.18).

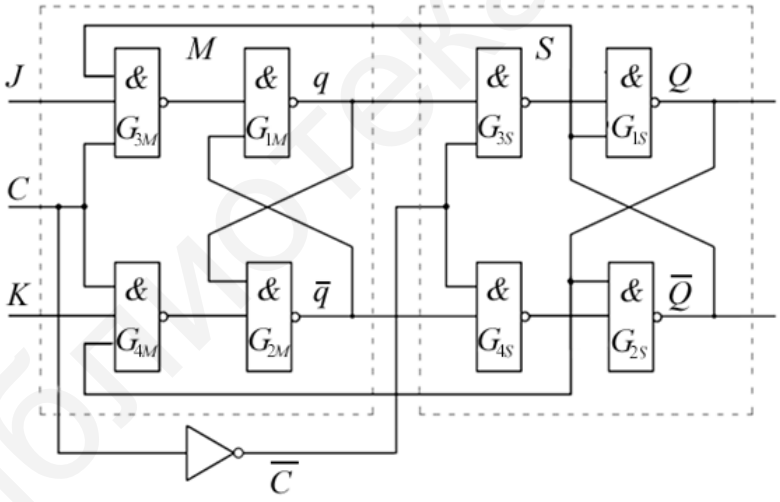


Рисунок 15.18 – Двухступенчатый *JK*-триггер с инвертором

Когда $C = 1$, первый триггер имеет разрешение и выходы q и $\bar{q}$ зависят от состояния входов J и K. В то же время на второй триггер подается запрет, поскольку $\bar{C}$. Когда C изменяется на низкий уровень, $C = 0$ ($\bar{C} = 1$), на первый триггер подается запрет, а на второй триггер разрешение, поскольку теперь его синхровход $\bar{C} = 1$. Поэтому выходы Q и $\bar{Q}$ повторяют состояния на выходах q и $\bar{q}$, соответственно. Поскольку второй триггер всегда повторяет состояние первого, его назвали *S* (*Slave*), а первый *M* (*Master*). В этой схеме входы $G_{3m} = \bar{Q}$ и $G_{4m} = Q$ не меняются в течение импульса синхронизации, поэтому *JK*-триггер

не может находиться в автоколебательном режиме. Состояние *M*-*S*-триггера изменяется при отрицательном перепаде импульса синхронизации.

Имеется другая версия *JK*-триггера в конфигурации *M*-*S* (рисунок 15.19).

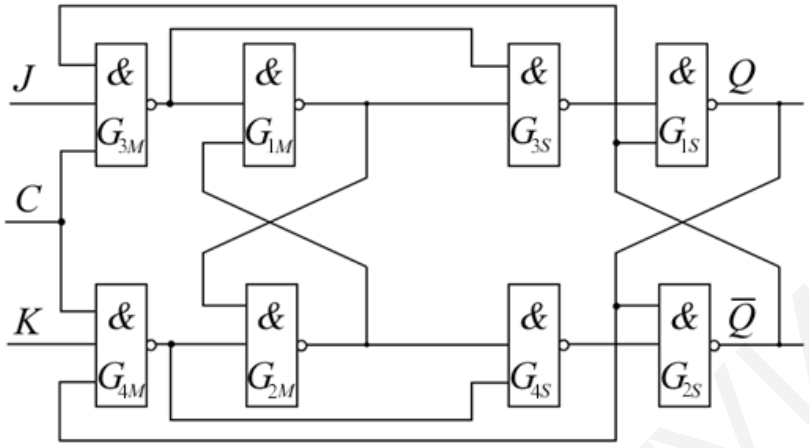


Рисунок 15.19 – Двухступенчатый *JK*-триггер с запрещающими связями

Условное обозначение *JK*-триггера показано на рисунке 15.20.

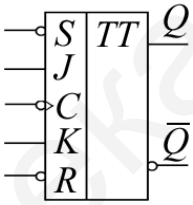
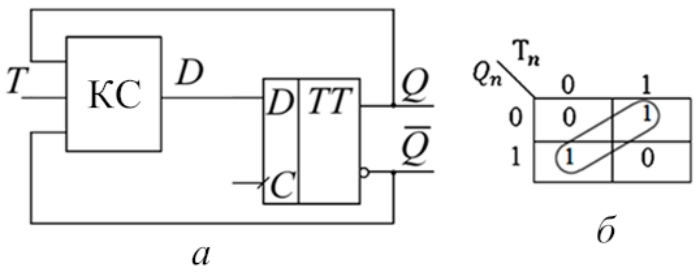


Рисунок 15.20 – Условное обозначение *JK*-триггера

***T*-триггер** имеет один вход *T*. Он изменяет свое состояние на противоположное с каждым импульсом синхронизации, если $T = 1$ и сохраняет свое состояние, если $T = 0$. *T*-триггеры не производятся в интегральном исполнении, но могут быть легко построены из *JK*- и *D*-триггеров. Исходя из определения *T*- и *JK*-триггеров очевидно, что *T*-триггер получается из *JK*-триггера, если входы *J* и *K* объединить (рисунок 15.21, *а*).



а – блок-схема *T*-триггера; *б* – карта Карно

Рисунок 15.21 – Блок-схема преобразования *D*-триггера в *T*-триггер

Поэтому *T*-триггер иногда рассматривают как одновходовый вариант *JK*-триггера.

Для того чтобы преобразовать *D*-триггер в *T*-триггер, воспользуемся таблицей переходов для обоих триггеров (таблица 15.8).

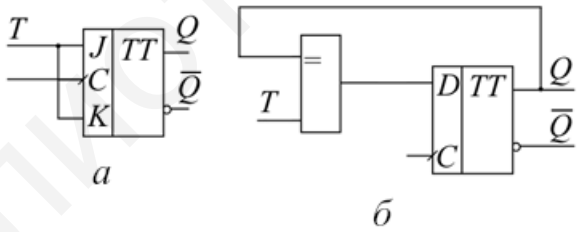
Таблица 15.8 – Таблица переходов для преобразования *D*-триггера в *T*-триггер

Настоящее состояние	**Следующее состояние**	***D*-триггер**	***T*-триггер**
Q_n	Q_{n+1}	D_n	T_n
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

Из блок-схемы (см. рисунок 15.15, *а*) очевидно, что необходимо синтезировать такую КС, чтобы *D*-триггер функционировал как *T*-триггер. ФАЛ, описывающую эту КС, имеет входные переменные T_n и Q_n, а выходная переменная – D_n (см. таблицу 15.8). Функция D_n (Q_n,T_n) может быть представлена и упрощена с помощью карты Карно (см. рисунок 15.21, *б*), результатом минимизации будет выражение

$$D_n = Q_n \otimes T_n. \tag{15.12}$$

Схема *T*-триггера будет выглядеть в соответствии с рисунком 15.22, *б*.



а *б*

а – условное обозначение; б – логическая схема

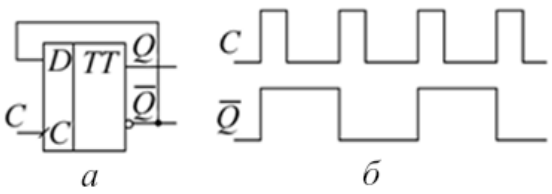
Рисунок 15.22 – *T*-триггер на основе *JK*- и *D*-триггера

Если вход T = 1, то *T*-триггер меняет свое состояние всякий раз, когда поступает импульс синхронизации.

Если переменная T_n в выражении (15.12) равна единице, то данное выражение может быть переписано в соответствии с выражением

$$D_n = Q_n \otimes 1 = \bar{Q}_n. \tag{15.13}$$

Выражение (15.13) показывает, что схему делителя на 2 можно построить простым соединением $\bar{Q}_n$ с *D* входом (рисунок 15.23).



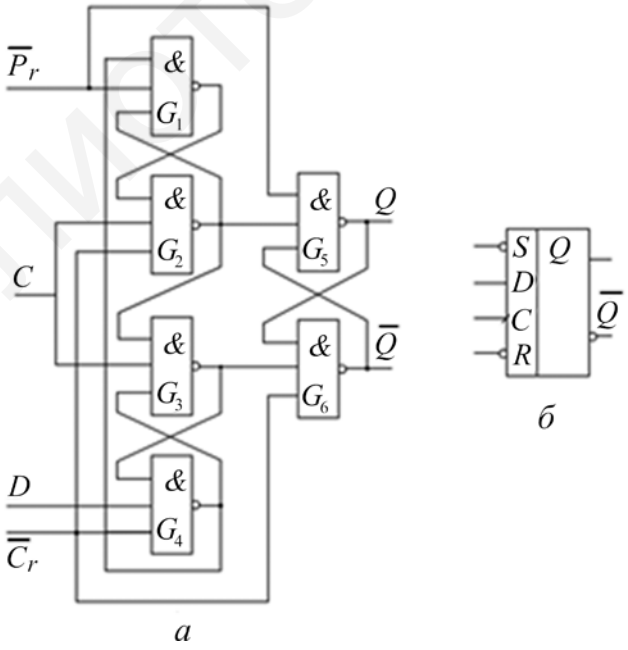
а – условное обозначение; *б* – эпюры делителя

Рисунок 15.23 – Схема делителя на 2 и его временные диаграммы

T-триггер не может быть построен на основе триггера со статическим управлением (*latch*).

Синхронный *D*-триггер с динамическим управлением. В триггерах с динамическим управлением изменения выходного состояния происходят в момент перехода синхросигнала с нулевого уровня на единичный (0 ⟶ 1), либо наоборот (1 ⟶ 0), при достижении порогового уровня. При этом информационные входы триггера в этот момент запираются, и триггер становится нечувствительным к изменениям входных сигналов до тех пор, пока синхросигнал не вернется в исходное нулевое состояние, и другой синхроимпульс не поступит на синхровход. Если триггер переключается положительным перепадом синхросигнала, то вход *C* называется прямым динамическим, если отрицательным – то инверсным динамическим.

Схема синхронного *D*-триггера с динамическим управлением приведена на рисунке 15.24.

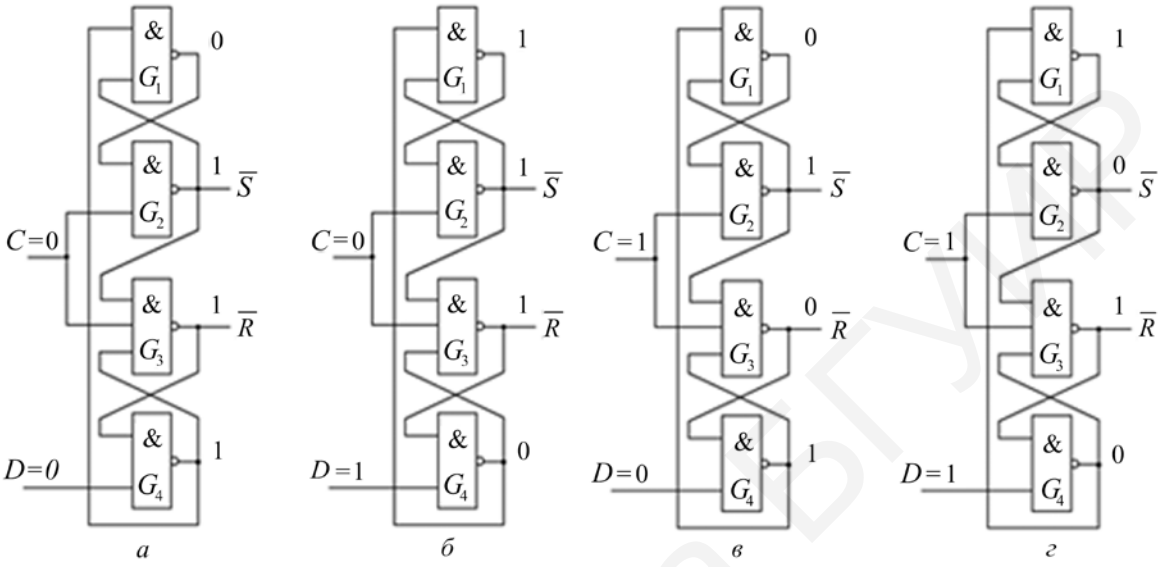


а – логическая схема; *б* – условное обозначение

Рисунок 15.24 – Логическая схема и условное обозначение синхронного *D*-триггера с динамическим управлением (*КР*1533*ТМ*2)

Схема состоит из трех базовых $\overline{\text{RS}}$-триггеров. И-НЕ элементы G_1 и G_2 образуют один базовый $\overline{RS}$-триггер, элементы G_4 и G_4 образуют другой $\overline{RS}$-триггер. Третий $\overline{RS}$-триггер, состоящий из G_5 и G_6, является выходом всей схемы.

Работу синхронного *D*-триггера с динамическим управлением рассмотрим с помощью рисунка 15.25.



а – вариант с *C* = 0, *D* = 0; *б* – вариант с *C* = 0, *D* = 1;
в – вариант с *C* = 1, *D* = 0; *г* – вариант с *C* = 1, *D* = 1

Рисунок 15.25 – Работа синхронного *D*-триггера с динамическим управлением

На рисунке 15.25 логические элементы G_1 и G_4 используются, чтобы показать все возможные переходы. На установочные входы $\bar{P}_r$ и $\bar{C}_r$ поданы логические единицы, поэтому для простоты они опущены (см. рисунок 15.18). Выходы логических элементов G_2 и G_3 являются управляющими $\bar{R}$ и $\bar{C}$ входами асинхронного $\overline{RS}$-триггера. На рисунке 15.25, *а* и *б* показаны значения на выходах G_1–G_4, когда *CK* = 0. На вход *D* может подаваться логический нуль или логическая единица. В любом случае *CK* = 0 и на выходах G_2 и G_3 лог. 1, т. е. $\bar{R} = \bar{S} = 1$ и выходной $\overline{RS}$-триггер находится в режиме хранения информации. Если *D* = 0, то на выходе G_4 лог. 1 и на выходе G_1 лог. 0. Если *D* = 1, то на выходе G_4 лог. 0 и на выходе G_1 лог. 1. При этих двух состояниях, когда на входе *CK* = 0, триггер не меняет свое состояние независимо от того, меняется ли состояние информационного входа *D*.

Теперь рассмотрим поведение синхронного *D*-триггера с динамическим управлением, когда на синхровход подается лог. 1. Если *D* = 0, когда на вход *C* поступает лог. 1, то $\bar{S}$ остается в состоянии лог. 1, а вход $\bar{R}$ переходит в состояние лог. 0. Этот лог. 0 устанавливает триггер в состояние *Q* = 0 ($\bar{Q} = 1$), а также поступает на один из входов G_4 и запирает вход *D*, блокируя любые изменения

на входе D. Выход G_4 может изменяться, лишь когда синхровход возвращается в состояние лог. 0, однако теперь оба входа $\bar{S}$ и $\bar{R}$ устанавливаются в состояние $\bar{R} = \bar{S} = 1$, запрещая тем самым любые изменения выхода D-триггера (и исключая неустойчивое состояние).

При анализе работы триггера с динамическим управлением необходимо принимать во внимание, что существует определенный промежуток времени, состоящий из времени установления и времени удержания, в течение которого состояние входа D не должно изменяться. Время установления, $t_{\text{уст}}$ равно времени задержки распространения через элементы G_4 и G_1, поскольку изменения на входе D приводят к изменению выходов этих элементов. Время удержания, $t_{\text{уд}}$ равно времени задержки распространения через элемент G_3, чтобы гарантировать, что $\bar{R} = 0$ и удерживает выход элемента G_4 в состоянии лог. 1 независимо от состояния входа D. Эти временные интервалы можно также пояснить с помощью рисунка 15.26.

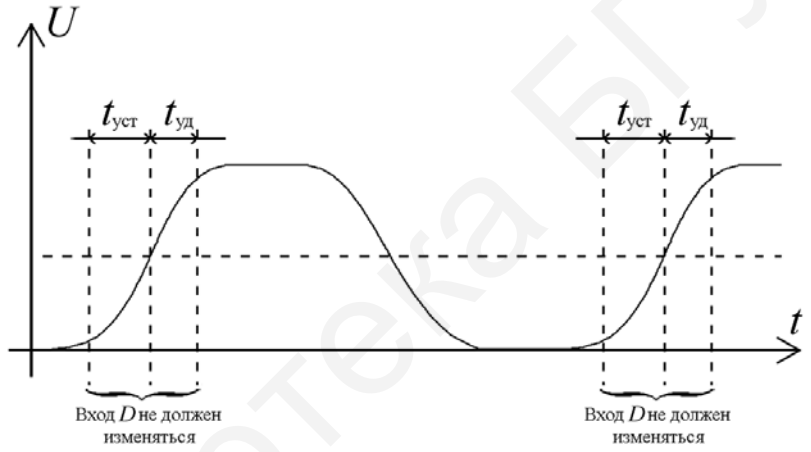


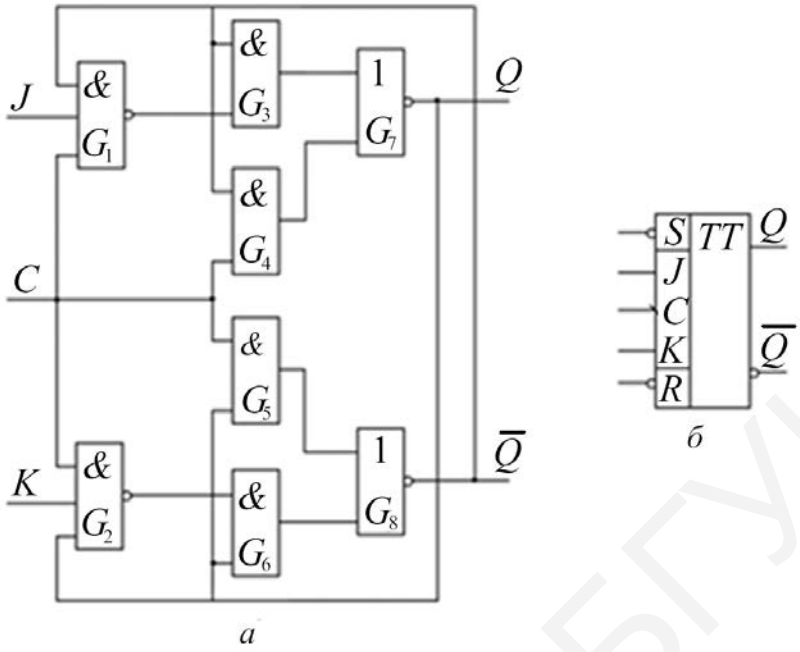
Рисунок 15.26 – Типичная форма синхроимпульсов для иллюстрации $t_{\text{уст}}$ и $t_{\text{уд}}$

Если $D = 1$, когда $CK = 1$, то $\bar{S}$ становится равным лог. 0, а $\bar{R}$ остается равным лог. 1, что устанавливает на выходе $Q = 1$. Смена сигналов на входе D, пока $CK = 1$, не изменяет логические уровни $\bar{S}$ и $\bar{R}$ потому, что на выходе G_1 сохраняется лог. 1. Когда синхроимпульс заканчивается и на входе CK устанавливается лог. 0, $\bar{R} = \bar{S} = 1$, что сохраняет состояние D-триггера.

Таким образом, когда на входе C происходит положительный перепад, значения входа D передается на выход Q. Изменения на входе D, когда $CK = 1$ или, когда происходит отрицательный перепад синхросигнала, или, когда $CK = 0$, не приводят к изменению состояния D-триггера.

***JK*-триггер с динамическим управлением** может быть построен с использованием логической схемы (см. рисунок 15.18), а также и другой логической схемы, которая является основой для построения таких триггеров как ТВ6, ТВ9, ТВ10, ТВ11.

На рисунке 15.27 представлена логическая схема и условное обозначение *JK*-триггера, подобного ТВ9.



а – логическая схема; *б* – условное обозначение

Рисунок 15.27 – Логическая схема и условное обозначение *JK*-триггера с инверсным динамическим управлением

Основным достоинством схемы (см. рисунок 15.21, *а*) является то, что время удержания $t_{уд}$ для нее равняется нулю, что повышает ее быстродействие.

Рассмотрим особенности схемной реализации и работу этой схемы. Для нормального функционирования задержка распространения сигнала через элементы G_1 и G_2 превышает суммарную задержку остальных элементов И, ИЛИ-НЕ. Изменения состояния триггера происходят по отрицательному перепаду синхросигнала, т. е. тактовый вход *C* триггера является инверсным динамическим. Проанализируем работу схемы для $J = K = 1$. Пусть триггер находится в состоянии $Q_n = 0$, а на синхровходе действует лог. 1 ($C = 1$), тогда на выходе элемента G_1 действует лог. 0, на выходе G_3 лог. 0 на выходе G_4 лог. 1, которая гарантирует на выходе триггера $Q = 0$. На инверсном выходе триггера установится $\bar{Q} = 1$, поскольку на выходах элементов G_2, G_5, G_6 уровень лог. 0.

Теперь при переходе *C* от лог.1 к лог. 0 (переход $1 \to 0 = \downarrow$) на выходах элементов G_3 и G_4 устанавливается лог. 0, а на выходе G_7 – лог. 1, т. е. выход триггера $G_{n+1} = 1$. Лог. 1 с выхода G_7 поступает на вход G_6, на второй вход которого поступает лог. 1 с выхода G_2, поэтому на выходе G_6 появляется лог. 1, а на выходе G_8 лог. 0, т. е. $\bar{Q} = 0$. Лог. 0 с выхода элемента G_8 поступает на вход элемента G_3 прежде, чем на втором входе появится лог. 1 с выхода элемента G_1, так как время распространения сигнала через элемент G_1 больше, чем суммарное

время распространения сигнала через элементы G_3–G_8. При возвращении синхросигнала в состояние лог. 1 ($C = 1$) на выходах элементов G_3 и G_4 сохраняется лог. 0, так как на один из входов G_3 и G_4 поступает лог. 0 с инверсного выхода триггера, что сохраняет на выходе $Q = 1$. На выходе элемента G_5 появляется лог. 1, обеспечивая сохранение на инверсном выходе $\bar{Q} = 0$. При поступлении следующего отрицательного перепада синхросигнала плечи триггера работают аналогично. Таким образом, при $J = K = 1$ триггер изменяет свое состояние на противоположное с каждым отрицательным перепадом синхроимпульса.

15.3 Порядок выполнения лабораторной работы

Оборудование и компоненты: универсальная лабораторная установка *IDL*-800, интегральные схемы: 1533ЛА3 (74*ALS*00) – четыре логических элемента 2И-НЕ, 1533ЛЕ1 (74*ALS*02) – четыре логических элемента 2ИЛИ-НЕ, 1533ЛА3 (74*ALS*10) – три логических элемента 3И-НЕ, 1533ТМ2 (74*ALS*74) – два синхронных динамических *D*-триггера, 1533ТМ7 (74*ALS*75) – четыре *D*-триггера со статическим управлением, 1533ТВ6 (74*ALS*107) – два синхронных динамических *JK*-триггера, 1533ТР2 (74*ALS*279) – четыре синхронных $\overline{\text{RS}}$-триггера.

Задание 1. Исследовать работу двухступенчатого *JK*-триггера с инвертором на логических элементах И-НЕ. Собрать схему (см. рисунок 15.11). По результатам исследования заполнить таблицу 15.9.

Таблица 15.9 – Результаты исследования работы двухступенчатого *JK*-триггера

t_n				t_{n+1}	Режим работы
C	Q_n	J	K	Q_{n+1}	
0 0	0 1	0 0	0 0		
0 0	0 1	0 0	1 1		
0 0	0 1	1 1	0 0		
0 0	0 1	1 1	1 1		
1 1	0 1	0 0	0 0		
1 1	0 1	0 0	1 1		
1 1	0 1	1 1	0 0		
1 1	0 1	1 1	1 1		

Задание 2. Исследовать асинхронный *RS*-триггер с инверсными входами на элементах И-НЕ. Собрать схему (см. рисунок 15.4, *в*). По результатам исследования заполнить таблицу 15.10.

Таблица 15.10 – Результаты исследования работы асинхронного *RS*-триггера

t_n			t_{n+1}	**Режим работы**
Q_n	S_n	R_n	Q_{n+1}	
0 1	0 0	0 0		
0 1	0 0	1 1		
0 1	1 1	0 0		
0 1	1 1	1 1		

Задание 3. Исследовать работу синхронного *RS*-триггера на элементах И-НЕ. Собрать схему (см. рисунок 15.7). По результатам исследования заполнить таблицу 15.11.

Таблица 15.11 – Результаты исследования работы синхронного *RS*-триггера

t_n				t_{n+1}	**Режим работы**
C	Q_n	S_n	R_n	Q_{n+1}	
0 0	0 1	0 0	0 0		
0 0	0 1	0 0	1 1		
0 0	0 1	1 1	0 0		
0 0	0 1	1 1	1 1		
1 1	0 1	0 0	0 0		
1 1	0 1	0 0	1 1		
1 1	0 1	1 1	0 0		
1 1	0 1	1 1	1 1		

Задание 4. Исследовать работу *D*-триггера с инверсными входами на элементах И-НЕ. Собрать схему (см. рисунок 15.8, *в*). По результатам исследования заполнить таблицу 15.12.

Таблица 15.12 – Результаты исследования работы *D*-триггера

t_n			t_{n+1}	Режим работы
C	Q_n	D	Q_{n+1}	
0 0	0 1	0 0		
0 0	0 1	1 1		
1 1	0 1	0 0		
1 1	0 1	1 1		

Задание 5. Исследовать асинхронный *RS*-триггер на элементах ИЛИ-НЕ. Собрать схему (см. рисунок 15.2, *в*). По результатам исследования заполнить таблицу 15.13, где t_n – текущий момент времени, а t_{n+1} – следующий.

Таблица 15.13 – Результаты исследования работы асинхронного *RS*-триггера с инверсными входами

t_n			t_{n+1}	Режим работы
Q_n	S_n	R_n	Q_{n+1}	
0 1	0 0	0 0		
0 1	0 0	1 1		
0 1	1 1	0 0		
0 1	1 1	1 1		

Задание 6. Используя интегральную схему 1533ТВ6, исследовать *T*-триггер. Собрать схему (см. рисунок 15.15, *а*). По результатам исследования заполнить таблицу 15.14.

Таблица 15.14 – Результаты исследования работы *T*-триггера

t_n				t_{n+1}	Режим работы
C	Q_n	T	R	Q_{n+1}	
X	X	X	0		
↑ ↑	0 1	0 0	1 1		
↑ ↑	0 1	1 1	1 1		
↓ ↓	0 1	0 0	1 1		
↓ ↓	0 1	1 1	1 1		

Задание 7. Исследовать работу *D*-триггера на микросхеме 1533ТМ2. По результатам исследования заполнить таблицу 15.15.

Таблица 15.15 – Результаты исследования работы *D*-триггера на микросхеме 1533ТМ2

t_n					t_{n+1}	Режим работы
Q_n	S	R	C	D		
0	0	0	X	X		
1	0	0	X	X		
0	0	1	X	X		
1	0	1	X	X		
0	1	0	X	X		
1	1	0	X	X		
0	1	1	0	X		
1	1	1	0	X		
0	1	1	1	0		
1	1	1	1	0		
0	1	1	1	1		
1	1	1	1	1		

Задание 8. Исследовать работу синхронного *JK*-триггера на интегральной микросхеме 1533ТВ6. По результатам исследования заполнить таблицу 15.16.

Таблица 15.16 – Результаты исследования работы синхронного *JK*-триггера

t_n					t_{n+1}	Режим работы
C	Q_n	J	K	R	Q_{n+1}	
X	X	X	X	0		
↓	0	0	0	1		
↓	1	0	0	1		
↓	0	0	1	1		
↓	1	0	1	1		
↓	0	1	0	1		
↓	1	1	0	1		
↓	0	1	1	1		
↓	1	1	1	1		
↑	0	0	0	1		
↑	1	0	0	1		
↑	0	0	1	1		
↑	1	0	1	1		
↑	0	1	0	1		
↑	1	1	0	1		
↑	0	1	1	1		
↑	1	1	1	1		

Задание 9. Используя интегральные схемы 1533ТМ2 и 1533ЛП5, собрать схему (см. рисунок 15.15, *б*) *Т*-триггера и исследовать его работу. По результатам исследования заполнить таблицу 15.17.

Таблица 15.17 – Результаты исследования работы *Т*-триггера на ИС 1533ТМ

t_n				t_{n+1}	Режим работы
C	Q_n	T	R	Q_{n+1}	
↓ ↓	0 1	0 0	1 1		
↓ ↓	0 1	1 1	1 1		
↑ ↑	0 1	0 0	1 1		
↑ ↑	0 1	1 1	1 1		
X	0 1	X	0 0		

15.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.
3. Таблицы и временные диаграммы работы исследуемых триггеров.
4. Выводы по результатам исследования.

15.5 Контрольные вопросы

1. Что такое триггер?
2. Какие признаки используются при классификации триггеров?
3. Что такое таблица истинности или характеристическая таблица триггера?
4. Что такое таблица переходов триггера?
5. В чем состоит отличие синхронных триггеров от асинхронных?
6. Чем различаются синхронные триггеры со статическим управлением и синхронные триггеры с динамическим управлением?
7. Почему не могут быть построены *Т*- и *JK*-триггеры со статическим управлением?
8. Объясните принцип действия двухступенчатого триггера (*M-S*-триггера).
9. Преобразуйте *RS*-, *D*-, *JK*-триггер в *Т*-триггер.
10. Поясните работу *D*-триггера с динамическим управлением.
11. Поясните работу *JK*-триггера с динамическим управлением.
12. В чем заключается различие и сходство *RS*-триггера и *JK*-триггера?

16 Лабораторная работа №7

РЕГИСТРЫ И ИХ ПРИМЕНЕНИЕ

16.1 Цель работы

Изучение принципов построения регистров, исследование режимов работы и применения регистров.

16.2 Теоретические сведения

Триггер может хранить (запоминать) один бит цифровой информации (1 или 0). Его также можно назвать одноразрядным регистром. Группа триггеров, предназначенная для хранения двоичной информации (один триггер на каждый бит информации), называется регистром. Регистры находят применение в различных цифровых устройствах, включая микропроцессоры.

Данные могут вводиться в регистр (записываться) в последовательной форме (бит за битом) или в параллельной форме (все биты одновременно) и могут выводиться из регистра в последовательной или параллельной форме.

Регистры классифицируются в зависимости от того, в какой форме информация вводится в регистр и в какой форме выводится.

Существует четыре возможности:

1) последовательно-последовательный регистр;
2) последовательно-параллельный регистр;
3) параллельно-последовательный регистр;
4) параллельно-параллельный регистр.

Регистры строятся, используя триггеры (*RS*, *JK*, *D*), и широко представлены как ИС средней степени интеграции.

Регистры, в которые данные вводятся или выводятся в последовательной форме, называются **сдвигающими**. Биты информации, находящиеся в триггерах регистра, сдвигаются то ли вправо, то ли влево при подаче синхроимпульсов. В некоторых регистрах информация может сдвигаться или вправо, или влево, в зависимости от специального управляющего сигнала. Такие регистры называются **реверсивными**. Если регистр может работать во всех четырех режимах и так же как реверсивный, то такой регистр называется **универсальным**.

Параллельные регистры предназначены для запоминания и хранения двоичной информации, поэтому параллельные регистры называют **регистрами хранения или регистрами памяти**. Такие регистры осуществляют операции записи и считывания информации параллельным кодом. Параллельные регистры могут использоваться в качестве буферных регистров, а также для преобразования прямого двоичного кода в обратный код и наоборот.

При построении параллельных регистров могут использоваться синхронные триггеры со статическим управлением (*latch*), а также синхронные триггеры с динамическим управлением и двухступенчатые (*RS*, *JK*, *D*) триггеры.

На рисунке 16.1 приведены структуры регистров ИР22 и ИР23. Микросхемы ИР22 и ИР23 – это восьмиразрядные параллельные регистры на *D*-триггерах. Причем регистр ИР22 построен на *D*-триггерах со статическим управлением, а ИР23 – с динамическим управлением.

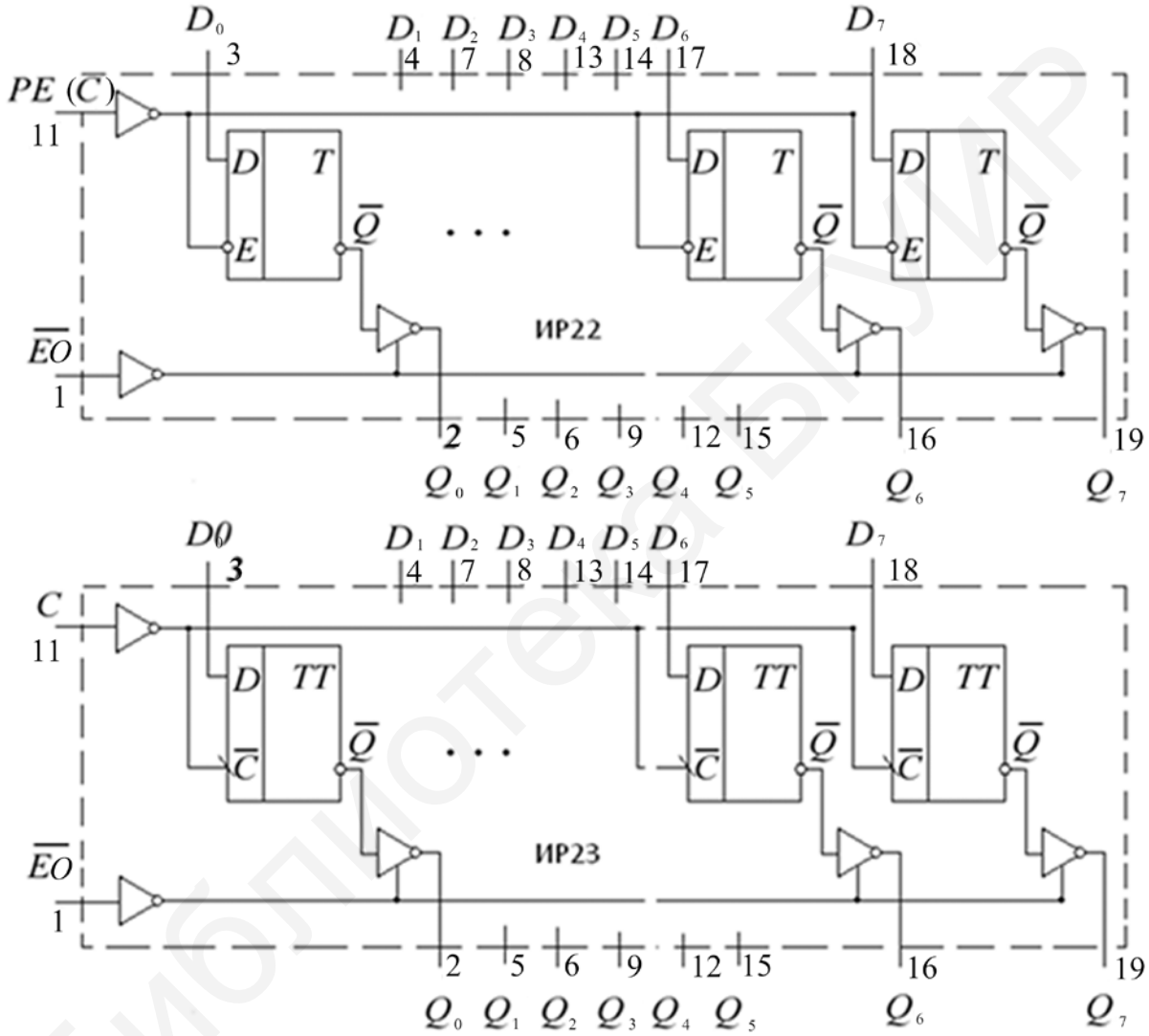


Рисунок 16.1 – Логическая структура регистров ИР22, ИР23

Регистры ИР22 и ИР23 имеют выходные буферные усилители с тремя состояниями. Третье высокоимпедансное состояние *Z* можно установить с помощью вывода разрешения $\overline{EO}$ (*Enable Output*), если подать на него напряжение высокого уровня. Выходные буферные усилители обладают высокой нагрузочной способностью.

Регистры состоят из восьми *D*-триггеров с входами разрешения параллельной записи *PE* (*Preset Enable*) (для ИР23 – вход *C*). Если на входе *PE*

действует высокий уровень напряжения, то данные от входов D_0–D_7 записываются в триггеры регистра.

Если на вход $\overline{EO}$ подано напряжение низкого уровня, то данные из D-триггеров регистра пройдут на выходы Q_0–Q_7.

Регистр ИР23 принимает информацию синхронно с положительным перепадом тактового импульса, подаваемого на вход C.

В настоящее время выпускается большое количество регистров разнообразного назначения. Однако наиболее универсальными являются регистры, которые могут работать во всех четырех режимах. Примером такого регистра может быть микросхема ИР16.

Микросхема ИР16 – это четырехразрядный регистр сдвига с третьим состоянием выхода. Логическая структура регистра показана на рисунке 16.2.

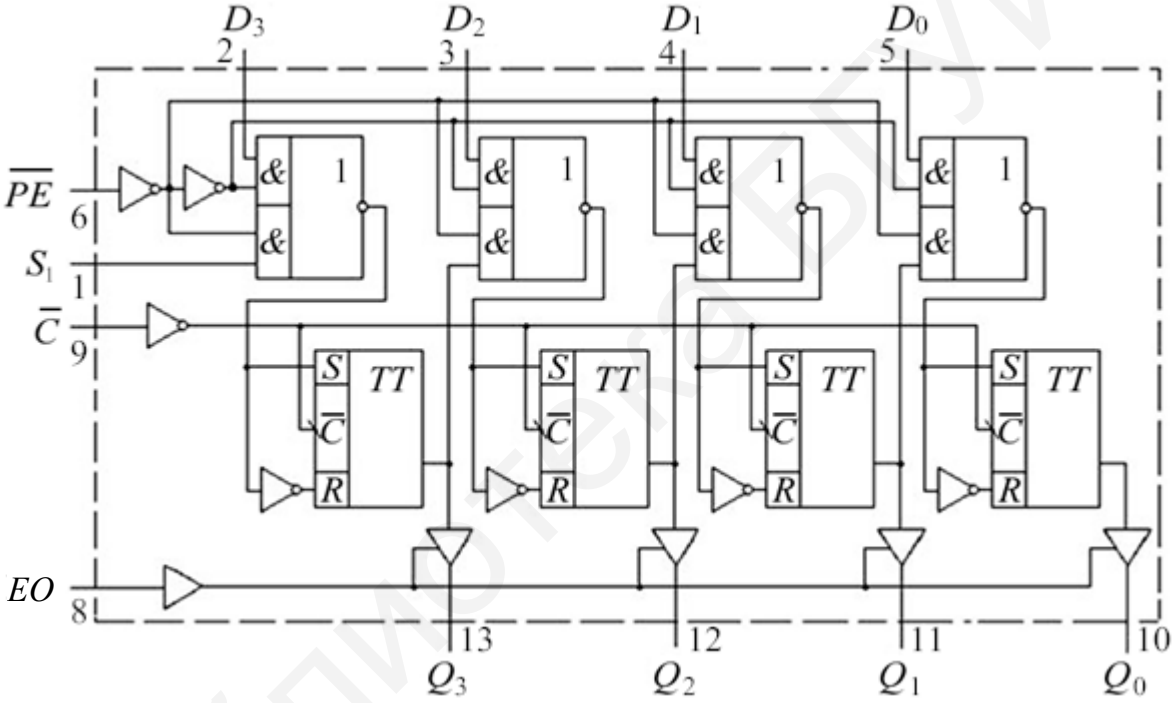


Рисунок 16.2 – Логическая структура регистра ИР16

Он построен на четырех синхронных RS-триггерах с инверсным динамическим управлением. RS-триггеры включены по схеме D-триггеров (вход S соединен через инвертор со входом R). На выходах регистра Q_0–Q_3 поставлены буферные усилители с повышенной нагрузочной способностью и тремя состояниями. Регистр имеет входы данных D_0–D_3, вход разрешения параллельной загрузки и сдвига $\overline{RE}$, тактовый вход C, вход последовательной загрузки данных S_1, вход разрешения выходам EO и выходы Q_0–Q_3.

Если на вход $\overline{RE}$ подать напряжения высокого уровня, то данные от входов D_0–D_3 параллельно загружаются в регистр синхронно с отрицательным перепадом импульса синхронизации на входе $\bar{C}$. Когда на входе $\overline{RE}$ действует напряжение низкого уровня, то загрузка данных в регистр происходит

последовательно от входа S_1, а сдвиг данных вправо от Q_3 к Q_0 синхронно с каждым отрицательным перепадом тактового импульса на входе $\bar{C}$.

Если на вывод разрешения выходам *EO* подать напряжение низкого уровня, то выходы Q_0–Q_3 перейдут в *Z*-состояние. На рисунке 16.3 показаны условные обозначения и цоколевка ИР22, ИР23 и ИР16.

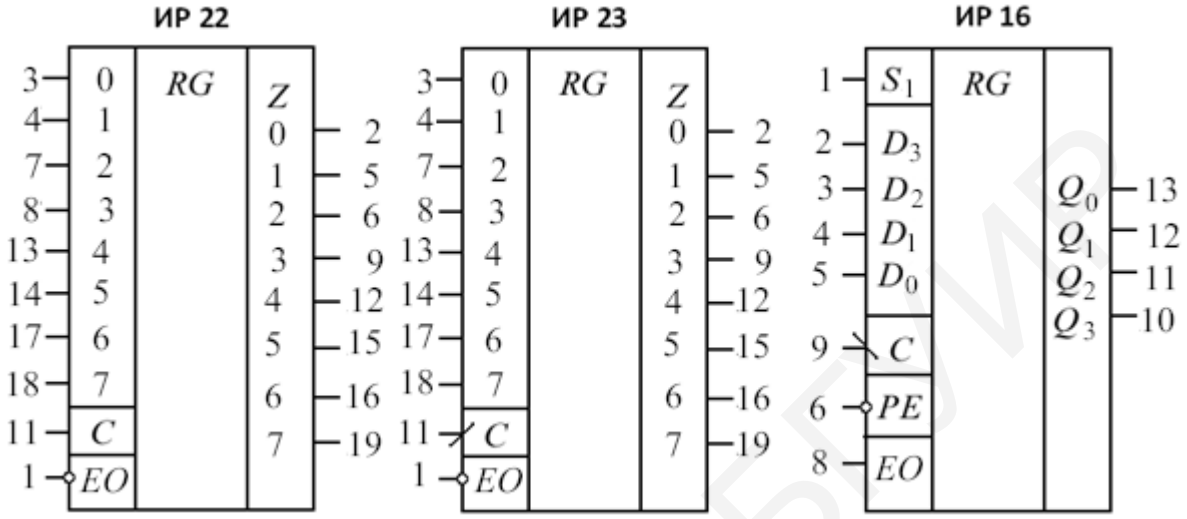


Рисунок 16.3 – Условные обозначения и цоколевка ИР22, ИР23 и ИР16

Реверсивные регистры сдвига могут осуществлять сдвиг информации как вправо (*SR – Shift Right*), так и влево (*SL – Shift Left*), в зависимости от сигнала на входе управления *M*. Пример построения реверсивного счетчика показан на рисунке 16.4.

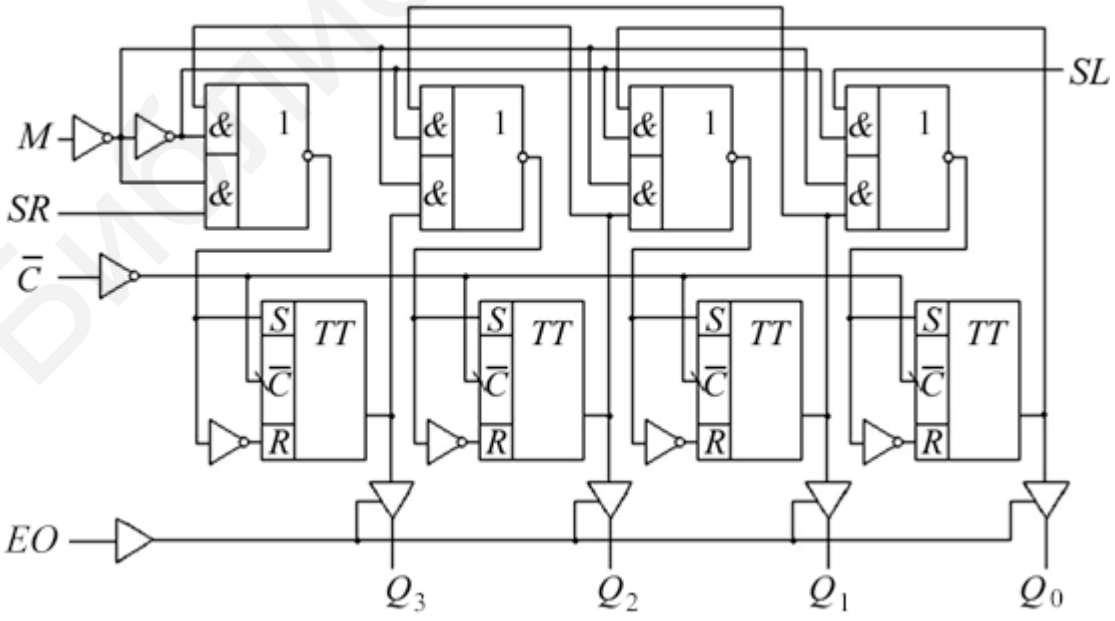


Рисунок 16.4 – Логическая структура реверсивного регистра

Если на вход *M* поступает низкий уровень напряжения, то при подаче тактовых импульсов информация с входа *SR* сдвигается вправо. И если на вход *M* поступает высокий уровень напряжения, то при подаче тактовых импульсов информация с входа *SL* сдвигается влево.

Применение регистров сдвига. Изначально регистры сдвига предназначались для временного хранения данных и некоторых манипуляций с этими данными. Рассмотрим некоторые наиболее общие применения регистров сдвига.

Линия задержки. Регистр сдвига с последовательным вводом и последовательным выводом данных можно использовать для задержки цифровых сигналов на время по формуле

$$\Delta t = \frac{N \cdot 1}{f_{\text{c}}}, \tag{16.1}$$

где N – число разрядов регистра сдвига;

f_{c} – частота следования импульсов синхронизации.

Таким образом, цифровой сигнал появляется на выходе регистра сдвига задержанным на время Δt. Время задержки можно варьировать с помощью изменения частоты следования импульсов синхронизации и числа триггеров регистра сдвига.

Преобразователь информации из *последовательного вида в параллельный вид*. Данные в последовательном виде легко преобразовать в параллельную форму с помощью последовательно-параллельного регистра.

Преобразователь информации из *параллельного вида в последовательный вид*. Данные в параллельном виде легко преобразовать в последовательную форму с помощью параллельно-последовательного регистра.

Кольцевой счетчик. Если последовательный выход регистра сдвига Q_0 соединить с последовательным входом, то единичный бит, записанный в один из триггеров, будет циркулировать по регистру при подаче синхроимпульсов. Такая схема называется **кольцевым счетчиком**. На выходах триггеров генерируется неперекрывающиеся последовательности импульсов, которые могут быть полезны для различных приложений.

Схема может быть использована для подсчета импульсов. Число сосчитанных импульсов определяется единичным уровнем на выходе соответствующего триггера. Модуль счета такого счетчика равен числу разрядов регистра, mod = *N*. Эта схема может рассматриваться и как делитель на *N* (*N*:1).

Счетчик Джонсона. Если выход $\bar{Q}_0$ соединить с последовательным входом, то такая схема называется **счетчиком Джонсона**. Если в такой схеме после обнуления регистра подать импульсы синхронизации, то на выходах триггеров будут генерироваться сигналы формы меандра.

Счетчик Джонсона – это делитель на $2N$ или модуль счета такого счетчика $\text{mod} = 2N$.

16.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудования и компоненты: универсальная лабораторная установка *IDL*-800, ИС КР1533ТМ2 (74*ALS*74) – два синхронных *D*-триггера с динамическим управлением, ИС КР1533ЛА3 (74*ALS*00) – четыре логических элемента 2И-НЕ, ИС КР1533ЛА4 (74*ALS*10) – три логических элемента 3И-НЕ, ИС КР555ИР16 (74*ALS*295) – четырехразрядный регистр сдвига.

Задание 1. Собрать схему четырехразрядного регистра сдвига, показанную на рисунке 16.5, используя две ИС КР1533ТМ2 и ИС КР1533ЛА3.

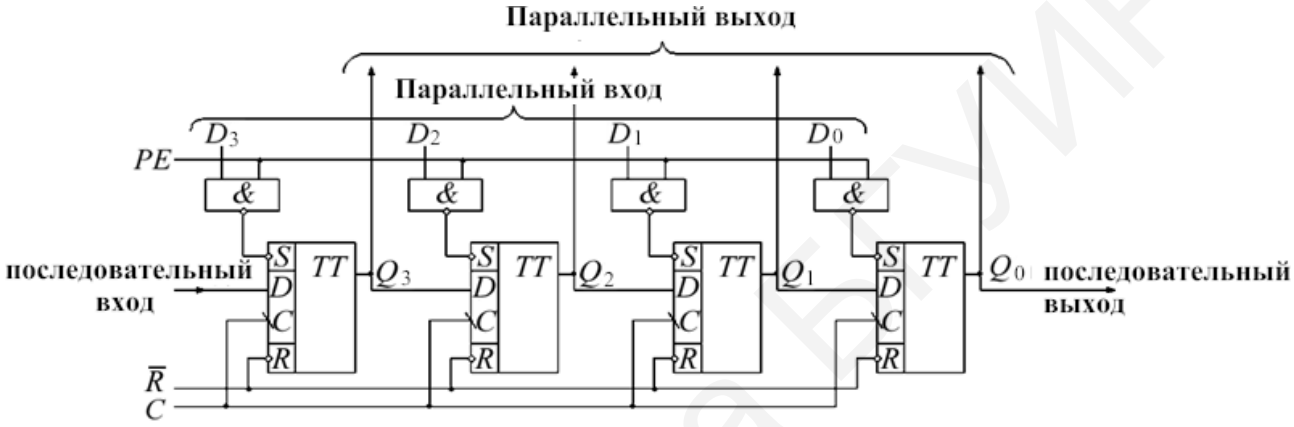


Рисунок 16.5 – Четырехразрядный регистр сдвига

Задание 2. Исследовать функционирование регистра сдвига (см. рисунок 16.5), который может работать во всех четырех режимах:

1. Последовательный вход, последовательный/параллельный выход. Исследовать работу регистра, подавая на его вход четырехразрядное слово «1011». Для любого другого слова работа регистра будет аналогична. На вход *PE* подать низкий уровень напряжения, затем обнулить регистр. Для этого на вход $\bar{R}$ подать низкий уровень напряжения, а затем установить $\bar{R} = 1$. Теперь, подавая импульсы синхронизации (использовать антидребезговую кнопку), проследить, как данные с последовательного входа будут сдвигаться в регистр. Наблюдая за состоянием выходов регистра, убедиться, что данные соответствуют временным диаграммам (рисунок 16.6).

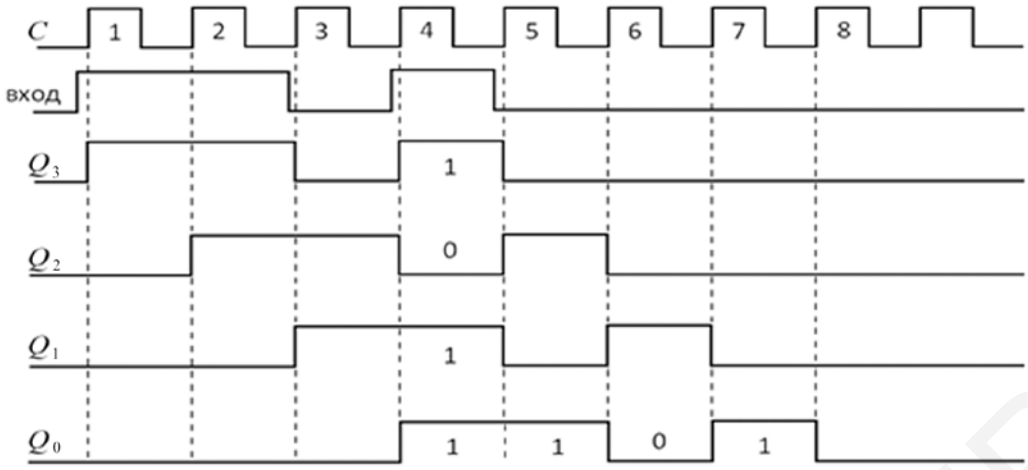


Рисунок 16.6 – Временные диаграммы работы регистра:
последовательный вход, последовательный/параллельный выход

2. Параллельный вход, последовательный/параллельный выход. Вначале обнулим регистр и установим $R = 1$. Установим входы $D_3 = 1$, $D_2 = 0$, $D_1 = 1$ и D_0, а затем на вход PE подадим высокий уровень напряжения и данные с входов D_3, D_2, D_1, D_0 запишутся в регистр. Теперь данные доступны в параллельном виде на выходах триггеров регистра. Подадим на синхровход регистра импульсы синхронизации и на выходе Q_0 получим выходные данные в последовательном виде (рисунок 16.7).

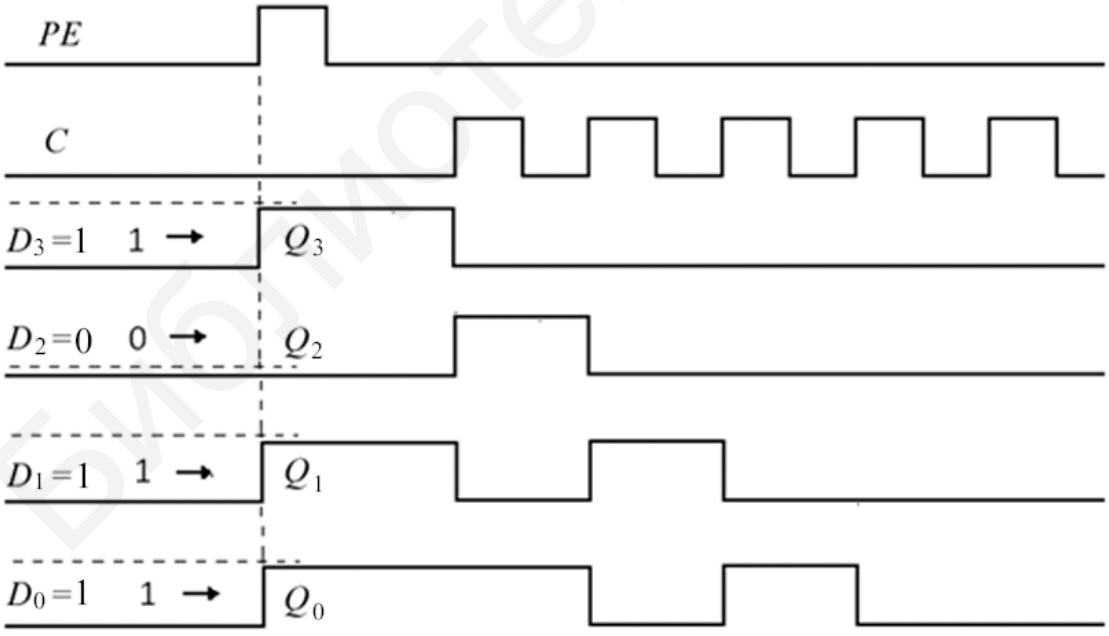


Рисунок 16.7 – Временные диаграммы, поясняющие работу регистра
в режимах последовательного входа, последовательного/параллельного выхода

3. Повторить пункты 1 и 2 для входных данных 1101. Построить временные диаграммы.

Задание 3. Исследовать кольцевой счетчик. Для этого:

1. На основе регистра сдвига (см. рисунок 16.5) построить кольцевой счетчик, для чего соединить последовательный выход регистра (выход Q_0) с последовательным входом.

2. Для исследования кольцевого счетчика вначале обнулить регистр, затем записать в него 0001 в параллельном виде и подать синхроимпульсы. Результаты наблюдения внести в таблицу 16.1.

Таблица 16.1 – Таблица исследований

C	Q_3	Q_2	Q_1	Q_0
0				
1				
2				
3				
4				
5				
6				
7				

3. Построить временные диаграммы работы кольцевого счетчика, сделать выводы.

Задание 4. Исследовать счетчик Джонсона. Для этого:

1. На основе регистра сдвига (см. рисунок 16.5) построить счетчик Джонсона, для чего необходимо соединить инверсный выход $\bar{Q}_0$ с последовательным входом регистра.

2. Для исследования работы счетчика Джонсона вначале обнулить регистр, а затем подать синхроимпульсы. Результаты наблюдений внести в таблицу 16.1.

3. Построить временные диаграммы работы счетчика Джонсона, сделать выводы.

16.4 Содержание отчета

1. Цель работы.
2. Схемы исследуемых в работе устройств.
3. Таблицы и временные диаграммы, отражающие результаты исследований.
4. Выводы по результатам исследований.

16.5 Контрольные вопросы

1. Объясните работу регистров во всех четырех режимах работы.
2. Как строится реверсивный регистр сдвига?
3. Как строится кольцевой счетчик?
4. Чему равен модуль счета кольцевого счетчика?
5. Как строится счетчик Джонсона?
6. Чему равен модуль счета счетчика Джонсона?

17 Лабораторная работа №8

ИССЛЕДОВАНИЕ АСИНХРОННЫХ И СИНХРОННЫХ СЧЕТЧИКОВ

17.1 Цель работы

Изучение методов построения и исследование функционирования основных типов асинхронных счетчиков, основ теории методов синтеза и исследование работы синхронных счетчиков.

17.2 Теоретические сведения

Асинхронные считчики. Цифровой счетчик – это группа триггеров, соединенных так, чтобы считать число импульсов, поданных на вход, и фиксировать число подсчитанных импульсов в том или ином коде.

Основными характеристиками счетчика являются коэффициент (модуль) счета и быстродействие.

Коэффициент или модуль счета характеризует число устойчивых состояний счетчика.

Быстродействие счетчика зависит от используемой элементной базы и схемы построения.

Счетчики классифицируются по ряду признаков:

1) по быстродействию и способу организации внутренних связей: асинхронные и синхронные;

2) по направлению счета: суммирующие, вычитающие, реверсивные;

3) по модулю счета: двоичные, двоично-десятичные или другим модулем счета.

Классификационные признаки независимы и могут встречаться в разных сочетаниях.

В наиболее общем случае рассматриваются асинхронные и синхронные счетчики. В случае асинхронных счетчиков триггеры перебрасываются не одновременно, а последовательно, и в случае синхронных счетчиков триггеры перебрасываются одновременно. Основным достоинством асинхронных счетчиков являются их схемная простота, а недостатком – низкое быстродействие. Основным достоинством синхронных счетчиков является их более высокое быстродействие, а недостатком – более сложная схемная реализация.

Рассмотрим построение асинхронных счетчиков. Для этого рассмотрим счетную последовательность (таблица 17.1).

Таблица 17.1 – Счетная последовательность

Счет	Состояние счета		
	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8(0)	0	0	0

Число состояний в такой последовательности равно 8, что требует использования трех триггеров ($2^3 = 8$).

Выход Q_0 наименее значащего триггера (разряда) изменяется каждым счетным импульсом. Это может быть реализовано путем использования триггера T-типа, при $T = 1$. Состояние на выходе Q_1 меняется всякий раз, когда выход Q_0 меняется из 1 в 0. Поэтому, если выход Q_0 соединить с входом синхронизации следующего T-триггера с инверсным динамическим входом, то выход Q_1 будет менять состояние всякий раз, когда выход Q_0 осуществляет переход из 1 в 0 (отрицательный перепад синхроимпульса). Таким же образом переход из 1 в 0 с выхода Q_1 изменяет состояние Q_2, что достигается путем соединения Q_1 с синхровходом следующего триггера. Анологичным образом может быть построен асинхронный счетчик с большим числом разрядов или модулем счета.

Асинхронный счетчик с использованием триггеров с инверсным динамическим входом (ТВ6). Входы J и K соединены вместе, образуя вход T-триггера, и на них подана лог. 1 (рисунок 17.1).

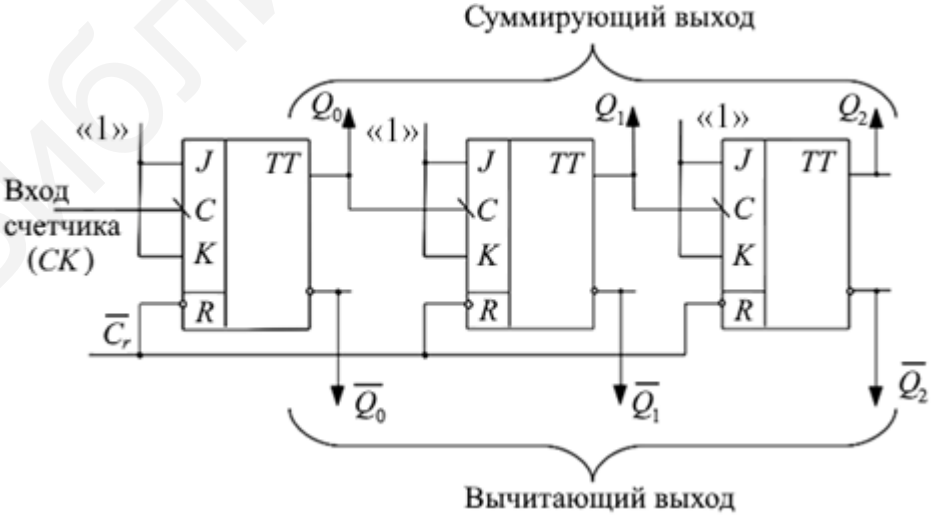


Рисунок 17.1 – Асинхронный счетчик на триггерах с инверсным динамическим синхровходом на ИС 1564ТВ6

На рисунке 17.2 показаны временные диаграммы, поясняющие работу счетчика. На прямых входах триггеров (Q_2, Q_1, Q_0) отражается состояние суммирующего счетчика, когда с каждым счетным импульсом состояние счетчика увеличивается. В то же самое время этот счетчик можно рассматривать как вычитающий, если информацию о состоянии счетчика снимать с инверсных выходов триггеров (Q_2, Q_1, Q_0).

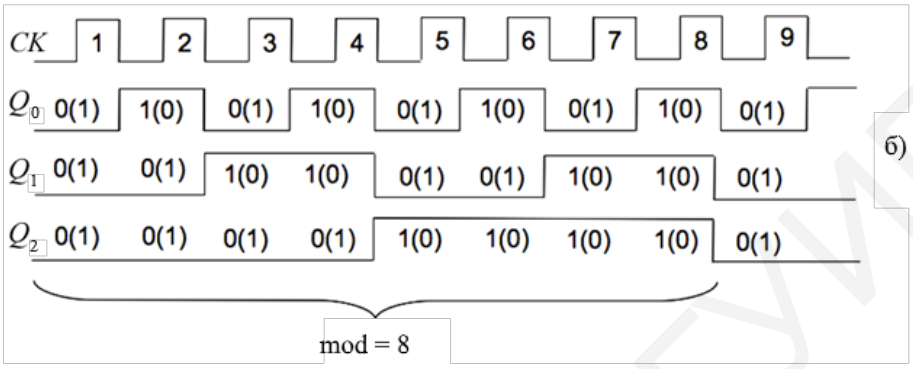


Рисунок 17.2 – Временные диаграммы асинхронного счетчика на триггерах с инверсным динамическим синхровходом на ИС 1564ТВ6

На рисунке 17.3 показан вариант схемы асинхронного счетчика на тех же триггерах с инверсным динамическим входом. На этот раз инверсные входы триггеров соединены с синхровходом последующих триггеров. В этой схеме суммирующий счетчик получается, если снимать информацию о состоянии счетчика с инверсных входов триггеров (Q_2, Q_1, Q_0), и вычитающий счетчик, если снимать информацию с прямых выходов триггеров (Q_2, Q_1, Q_0).

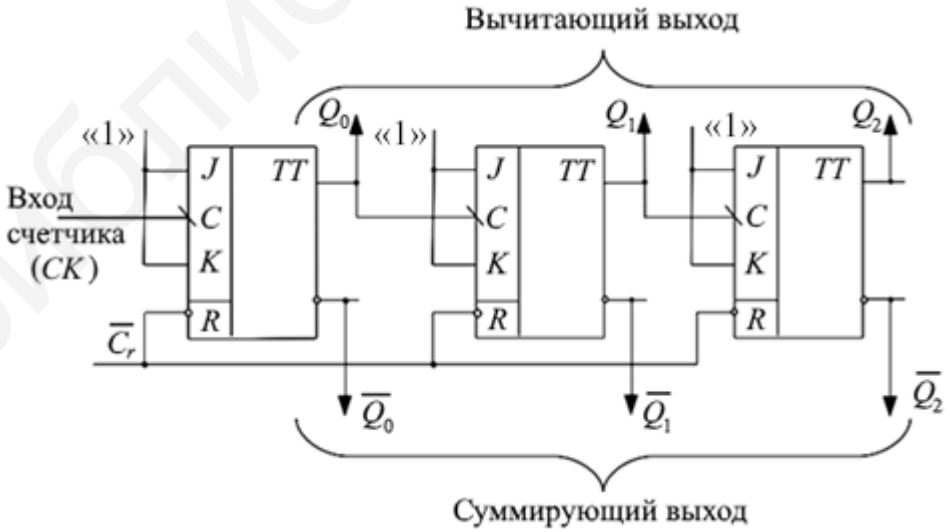


Рисунок 17.3 – Асинхронный счетчик на триггерах с инверсным динамическим синхровходом

На рисунке 17.4 показаны временные диаграммы, поясняющие работу счетчика.

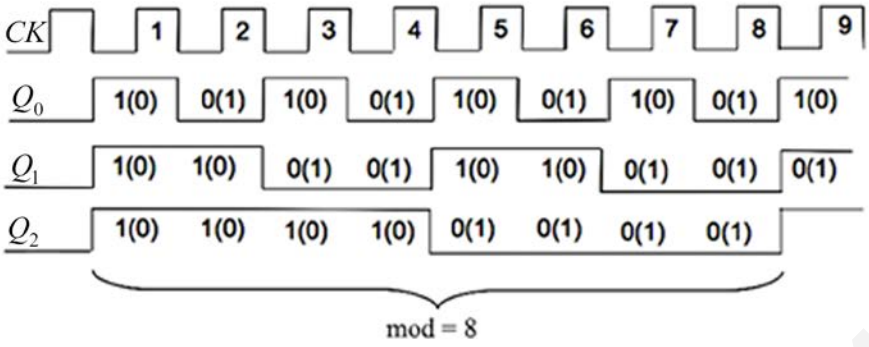
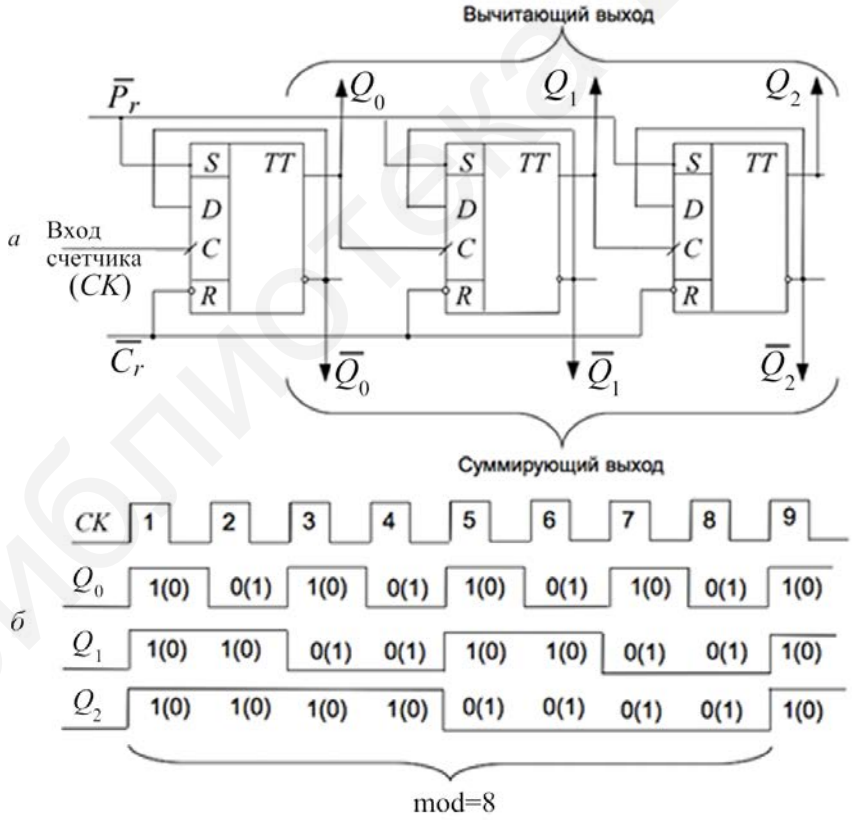


Рисунок 17.4 – Временные диаграммы асинхронного счетчика на триггерах с инверсным динамическим синхровходом

Рассмотрим теперь, как строятся асинхроные счетчики на основе триггеров с прямым динамическим входом (например, ТМ2) и временные диаграммы, поясняющие работу счетчика (рисунок 17.5).



а – асинхронный счетчик; *б* – временные диаграммы

Рисунок 17.5 – Асинхронный счетчик на триггерах с прямым динамическим синхровходом

Если в счетчике на триггерах с прямым динамическим синхровходом прямые выходы триггеров соединены с синхровходами последующих, то суммирующий и вычитающий выходы счетчика поменяются местами.

Асинхронные счетчики построенные на двухступенчатых триггерах *M-S*-типа, которые перебрасываются отрицательными перепадами синхроимпульса, работают аналогично счетчикам на триггерах с инверсным динамическим синхровходом .

Реверсивные счетчики. Счетчики способные работать как в прямом, так и обратном направлении, называются реверсивными. При построении реверсивного счетчика изменение направления счета достигается переключением межразрядных связей. Пример построения асинхронного реверсивного счетчика приведен на рисунке 17.6.

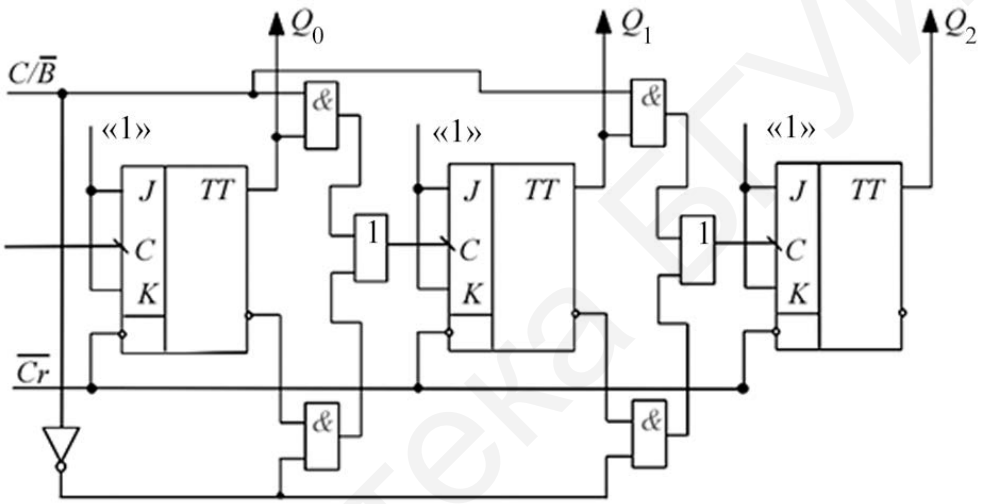


Рисунок 17.6 – Реверсивный асинхронный счетчик

В зависимости от сигнала на управляющим входе $C/\bar{B}$ счетчик (см. рисунок 17.6) может осуществлять счет в прямом или обратном направлении.

Счетчики с произвольным модулем счета. Счетчики, которые мы рассмотрели, имеют модуль счета $\mathrm{mod} = 2^N$ (2, 4, 6, 8, 16…). Однако при проектировании цифровых устройств часто возникает необходимость построить счетчик с модулем счета $\mathrm{mod} \neq 2^N$. Принцип построения таких счетчиков состоит в исключении избыточных состояний с помощью включения обратных связей внутри счетчика либо методом управляемого сброса в нуль, когда в нем устанавливается определенное состояние.

Итак, если необходимо получить счетчик с каким-то модулем счета mod, то число триггеров, необходимых при этом, определяется исходя из $\mathrm{mod} \leq 2^N$.

Например, при $N = 4$ любой mod в пределах от 9 до 16 может быть получен. Если необходимо получить счетчик с $\mathrm{mod} = 10$, то шесть состояний не используется. При этом если счет осуществляется в натуральном *BCD*-коде,

то счетчик будет проходить последовательно состояния, как показано на рисунке 17.7.

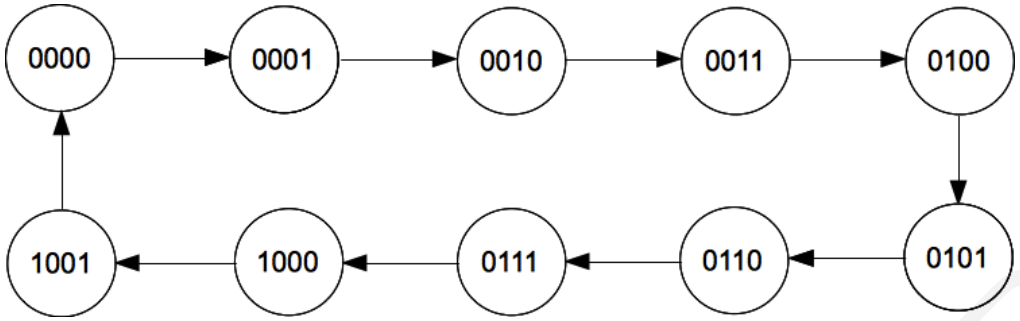


Рисунок 17.7 – Последовательное состояние счетчика

И это подобно двоичному счетчику, за исключением того, что после состояния 1001 (код десятичного числа 9) следующим состоянием будет 0000 (код десятичного числа 0). Синтез десятичного асинхронного счетчика или любого другого асинхронного счетчика с $\mathrm{mod} \neq 2^N$, или счетчика, работающего не в прямом двоичном коде, является сложной проблемой, поскольку не существует прямой процедуры синтеза.

Рассмотрим в качестве примера построение и работу двоично-десятичного асинхронного счетчика (рисунок 17.8).

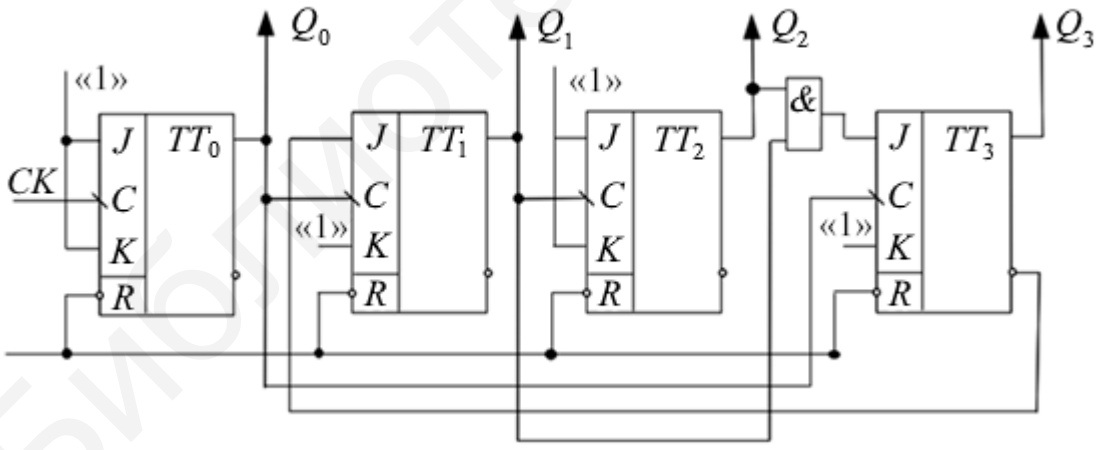


Рисунок 17.8 – Асинхронный двоично-десятичный счетчик

Триггеры в счетчике (см. рисунок 17.8) перебрасываются с отрицательным перепадом синхросигнала. В схеме счетчика выход Q_0 соединен с синхровходами двух триггеров TT_1 и TT_3, а вход Q_1 соединен с синхровходом TT_2. Входы J и K соединены либо постоянно с лог. 1, либо с выходами триггеров (см. рисунок 17.8). Работа счетчика может быть объяснена исходя из условий переброса триггеров. Вспомним, что если синхроимпульс имеет переход из 1 в 0, триггер

устанавливается в единичное состояние при $J = 1$ и $K = 0$, обнуляется при $J = 0$ и $K = 1$, меняет свое состояние на противоположное при $J = K = 1$ и не изменяет свое состояние при $J = K = 0$.

Из схемы (см. рисунок 17.8) очевидно, что:

- выход Q_0 меняет свое состояние с каждым отрицательным перепадом импульсов СК;

- выход Q_1 меняет свое состояние, если $Q_3 = 0$ и Q_0 имеет переход из 1 в 0;

- выход Q_2 меняет свое состояние всякий раз, когда Q_1 имеет переход из 1 в 0;

- выход Q_3 меняет свое состояние, когда Q_1, $Q_2 = 1$ и Q_0 имеет переход из 1 в 0. Выход Q_3 обнуляется, если Q_1 или Q_2 равны 0 и Q_0 имеет переход из 1 в 0.

Временные диаграммы, поясняющие работу двоично-десятичного асинхронного счетчика, приведены на рисунке 17.9.

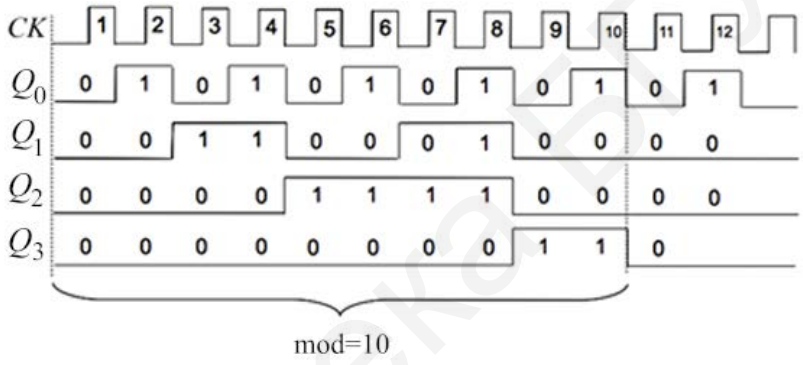


Рисунок 17.9 – Асинхронный двоично-десятичный счетчик

Интегральные схемы асинхронных счетчиков. Мы рассмотрели построение асинхронных счетчиков, используя отдельные триггеры. Ряд асинхронных счетчиков выпускаются промышленностью как интегральные схемы средней степени интеграции. Все ИС состоят из четырех *M*-*S*-триггеров. Загрузка, установка и сброс (обнуление) осуществляются асинхронно, т. е. независимо от импульсов синхронизации.

В зависимости от особенностей счетчиков, связанных с загрузкой, установкой и сбросом, эти счетчики условно разделяются на три группы.

К первой группе относится асихронный счетчик 1533ИЕ2 (74*ALS*90), который имеет входы установки и сброса. Блок-схема счетчика показана на рисунке 17.10.

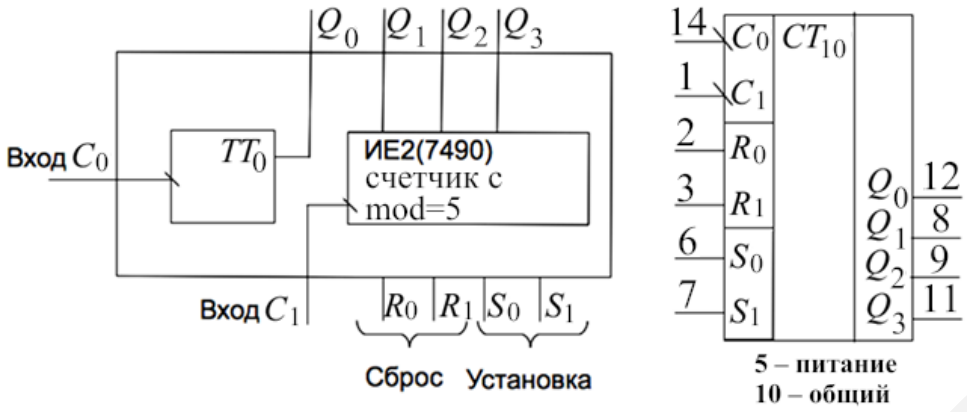


Рисунок 17.10 – Асинхронный счетчик ИЕ2

Счетчик состоит из четырех триггеров, объединенных внутри, как счетчик по $\mathrm{mod} = 2$ и счетчик по $\mathrm{mod} = 5$. Эти счетчики могут использоваться независимо или в комбинации. Если выход счетчика с $\mathrm{mod} = 2$ (Q_0) соединить со входом счетчика по $\mathrm{mod} = 5$, то образуется двоично-десятичный счетчик. Если соединить выход счетчика с $\mathrm{mod} = 5$ (Q_3) со входом счетчика $\mathrm{mod} = 2$, то образуется счетчик-делитель на 10. В счетчике имеются два входа «Сброс» R_0 и R_1, на которые необходимо подать лог. 1 для обнуления счетчика. Когда на оба входа «Установка» S_0 и S_1 подается лог. 1, двоично-десятичный счетчик устнавливается в состояние 1001.

Внутренняя схема и временные диаграммы работы счетчика ИЕ2 соответствуют двоично-десятичному счетчику, рассмотренному ранее (см. рисунок 17.8).

Ко второй группе асинхронных счетчиков можно условно отнести счетчики, которые имеют только входы сброса. Блок схема счетчиков 155ИЕ4, 1533ИЕ5 показана на рисунке 17.11.

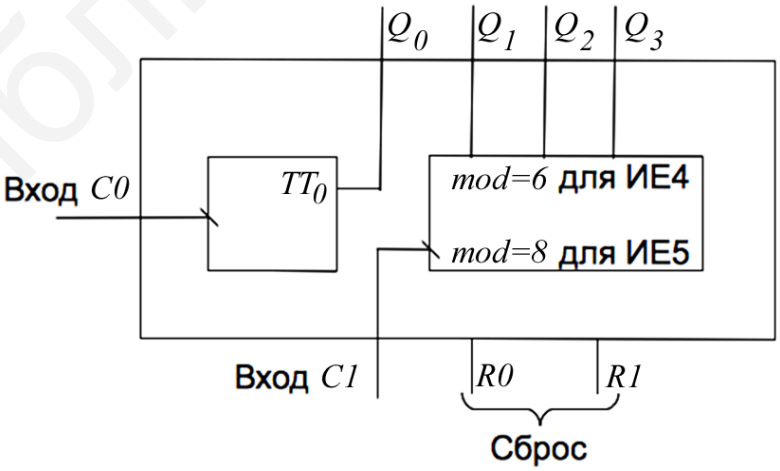


Рисунок 17.11 – Блок-схема асинхронных счетчиков ИЕ4, ИЕ5

Работа этого счетчика аналогична ИЕ2, за исключение того, что входы установки отсутствуют и счетчик по mod = 6 осуществляет счет не в натуральном двоичном коде. Эта последовательность дана в таблице 17.2.

Таблица 17.2 – Двоичные значения

Q_3	Q_2	Q_1
0	0	0
0	0	1
0	1	0
1	0	0
1	0	1
1	1	0

ИС555ИЕ20 (74*ALS*390) содержит два двоично-десятичных счетчика, подобных ИЕ2. Для каждого счетчика имеется один вход «Сброс» (*R*). ИС1533ИЕ19 (74*ALS*393) содержит два 4-разрядных двоичных счетчика с входом «Сброс» (*R*).

К третьей группе асинхронных счетчиков относятся счетчики 555ИЕ14 (74*ALS*176, 74*ALS*196) и 555ИЕ15 (74*ALS*177, 74*ALS*197). Эти счетчики являются версиями счетчиков ИЕ2 и ИЕ5 с предварительной установкой состояния. Блок-схема счетчиков показана на рисунке 17.12.

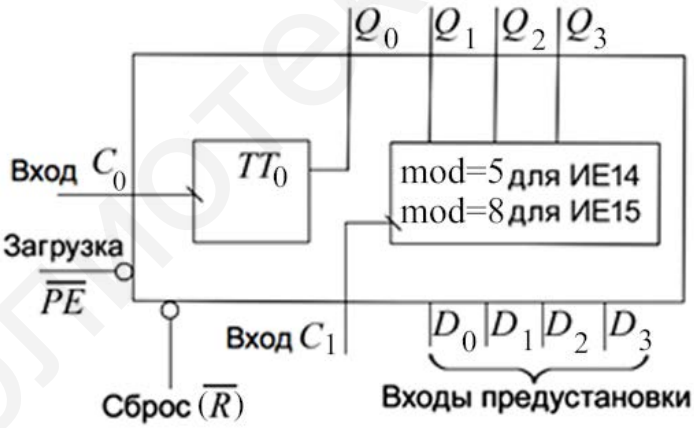


Рисунок 17.12 – Блок-схема асинхронных счетчиков ИЕ14, ИЕ15

Счетчики обнуляются при подаче лог. 0 на вход $\bar{R}$. Подача лог. 0 на вход $\overline{PE}$ (в это время на входе «Сброс» должно быть $\bar{R} = 1$) останавливает счет и в счетчик загружается число, установленное на входах предустановки. В режиме счета на оба входа «Сброс» и «Загрузка» должны быть поданы лог. 1.

Условное обозначение и цоколевка ИС ИЕ4, ИЕ5, ИЕ14, ИЕ15 показаны на рисунке 17.13.

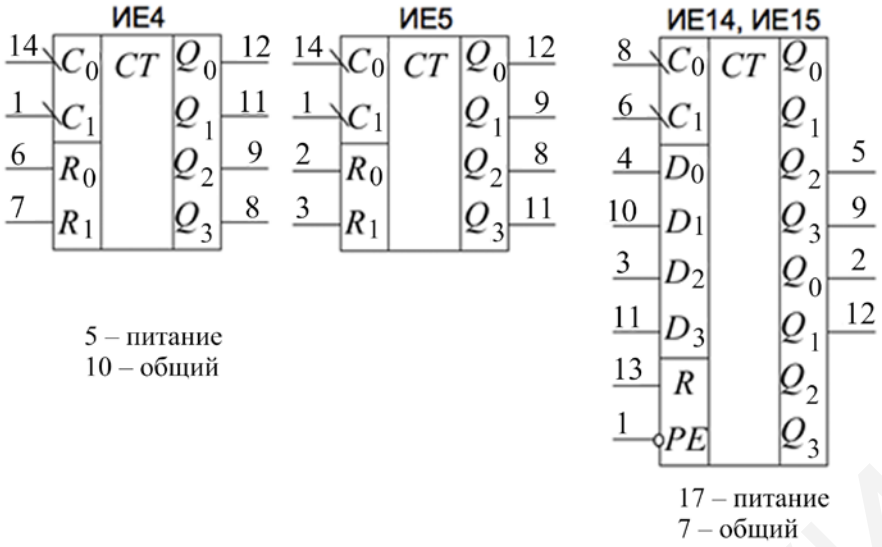
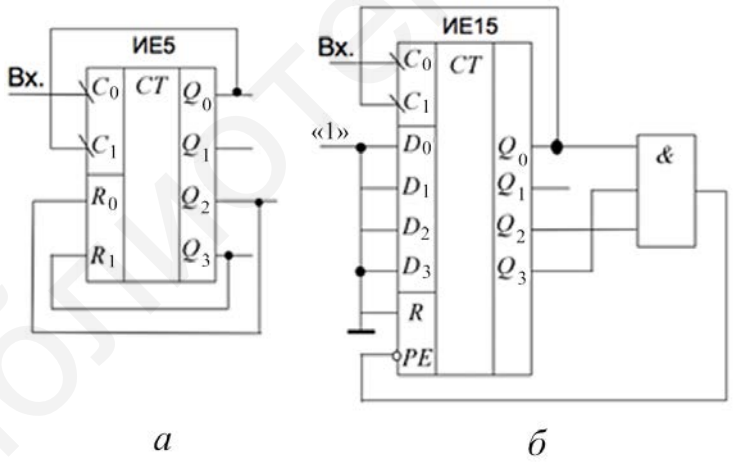


Рисунок 17.13 – Условные обозначения ИС ИЕ4, ИЕ5, ИЕ14, ИЕ15

Входы асинхронных счетчиков «Загрузка», «Установка», «Сброс» могут использоваться для изменения модуля и последовательности счета.

Например, на рисунке 17.14, *а* показана схема счетчика с $\mathrm{mod} = 12$, построенная на основе ИС ИЕ5. На рисунке 17.12, *б* показана схема десятичного счетчика, последовательность состояний которого соответствует коду с избытком 3.



а – счетчик с $\mathrm{mod} = 12$; *б* – десятичный счетчик

Рисунок 17.14 – Схемы счетчика

Исследование синхронных счетчиков. Максимальное время переключения асинхронного счетчика, когда выход счетчика последовательно изменяется 11...1 на 00...0, и это ограничивает его быстродействие. Быстродействие можно повысить, если все триггеры счетчика тактировать одновременно. В этом случае счетчик называется синхронным. Основу

синхронного счетчика составляют, как правило, синхронные триггеры с динамическим управлением. Все синхровходы триггеров объединены, образуя счетный вход счетчика. Количество триггеров зависит от модуля счета ($\mathrm{mod} < 2^N$). В задачу синтеза синхронного счетчика входит определение связей управляющих входов триггеров и их выходов, чтобы триггеры переключались в соответствии с заданной последовательностью состояний счетчика. Синхронные счетчики могут быть синтезированы с использованием системных методов. Однако, прежде чем рассматривать такой метод, рассмотрим интуитивный метод. Для этого воспользуемся таблицей 17.3, в которой дана счетная последовательность и соответствующие состояния счетчика для $\mathrm{mod} = 8$.

Таблица 17.3 – Состояние счетчика mod = 8

Счет	**Состояния счета**		
	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Из таблицы 17.3 видно, что выход Q_0 младшего триггера счетчика изменяется с каждым счетным импульсом. Это может быть достигнуто, если использовать T-триггер с T = 1. Выход Q_1 изменяет свое состояние всякий раз, когда Q_0 изменяется с 1 на 0. Поэтому, если соединить выход Q_0 с входом T_1, то Q_1 будет изменять свое состояние, когда $Q_0 = 1$, $Q_1 = 1$ ($T_1 = Q_0 = 1$), и будет оставаться без изменения, когда $Q_0 = T_1 = 0$. Из таблицы (см. таблицу 17.3) видно, что Q_2 изменяет свое состояние всякий раз, когда Q_1 и Q_0 оба равны 1. Это может быть реализовано, если вход наиболее значащего триггера $T_2 = Q_1 \cdot Q_2$. В результате получим схему синхронного счетчика (mod = 8), показанную на рисунке 17.15.

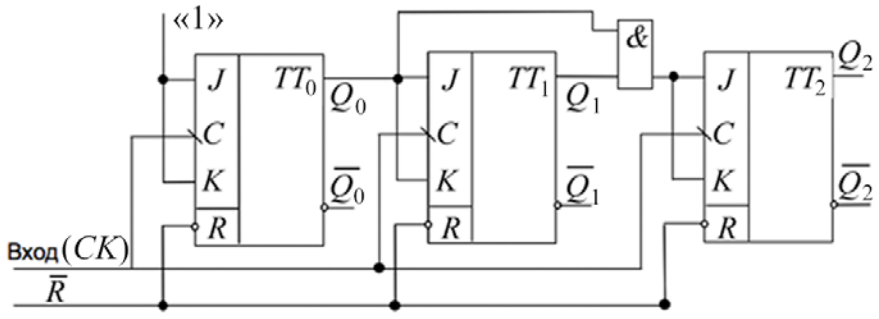


Рисунок 17.15 – Синхронный счетчик с mod = 8

Аналогичным образом могут быть построены синхронные счетчики с mod = 2^N.

Синтез синхронных счетчиков. Синхронный счетчик для любой заданной последовательности счета и модуля счета может быть синтезирован следующим образом:

1. Необходимо определить требуемое число триггеров по формуле

$$N \geq \log_2 m, \tag{17.1}$$

где m – модуль счета.

2. Записать счетную последовательность в табличном виде.

3. Определить состояния входов триггеров, которые должны быть для перехода в требуемые следующие состояния, исходя из настоящего состояния и таблицы переходов триггеров.

4. Приготовить карты Карно для каждого входа триггеров в терминах выходов триггеров как входных переменных. Используя метод Карно, получить минимизированные выражения для каждого входа триггеров.

5. Построить схему счетчика, используя триггеры и логические элементы, в соответствии с минимизированными выражениями.

Пример 1. Построить двоично-десятичный счетчик, который имеет десять состояний, поэтому для его построения необходимо ($N \geq \log_2 10$) $N = 4$ триггера. Построим таблицу состояний и таблицу истинности для входов триггеров (таблица 17.4).

Таблица 17.4 – Таблица состояний для синтеза счетчика с mod = 10

Состояния счетчика				**Входы триггеров**							
Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	*X*	0	*X*	0	*X*	1	*X*
0	0	0	1	0	*X*	0	*X*	1	*X*	*X*	1
0	0	1	0	0	*X*	0	*X*	*X*	0	1	*X*
0	0	1	1	0	*X*	1	*X*	*X*	1	*X*	1
0	1	0	0	0	*X*	*X*	0	0	*X*	1	*X*
0	1	0	1	0	*X*	*X*	0	1	*X*	*X*	1
0	1	1	0	0	*X*	*X*	0	*X*	0	1	*X*
0	1	1	1	1	*X*	*X*	1	*X*	1	*X*	1
1	0	0	0	*X*	0	0	*X*	0	*X*	1	*X*
1	0	0	1	*X*	1	0	*X*	0	*X*	*X*	1
0	0	0	0	–							

Используя метод Карно, получим минимизированные выражения для всех входов *J* и *K*, как показано на рисунке 17.16.

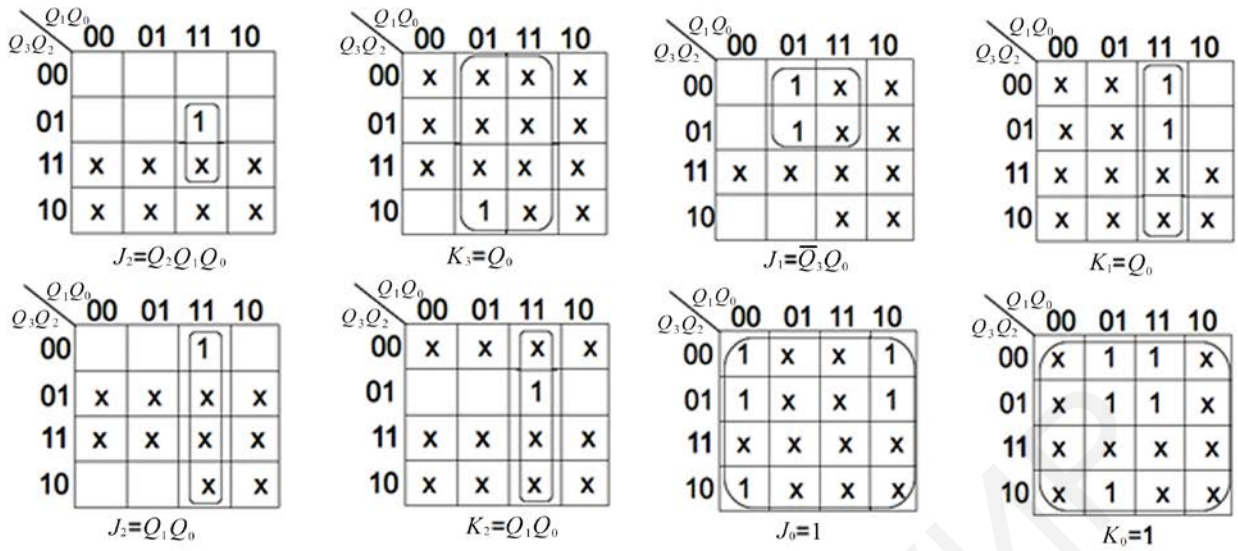


Рисунок 17.16 – Минимизация функций *J* и *K* с помощью карт Карно

По полученным выражениям (см. рисунок 17.16) построим синхронный двоично-десятичный счетчик (рисунок 17.17).

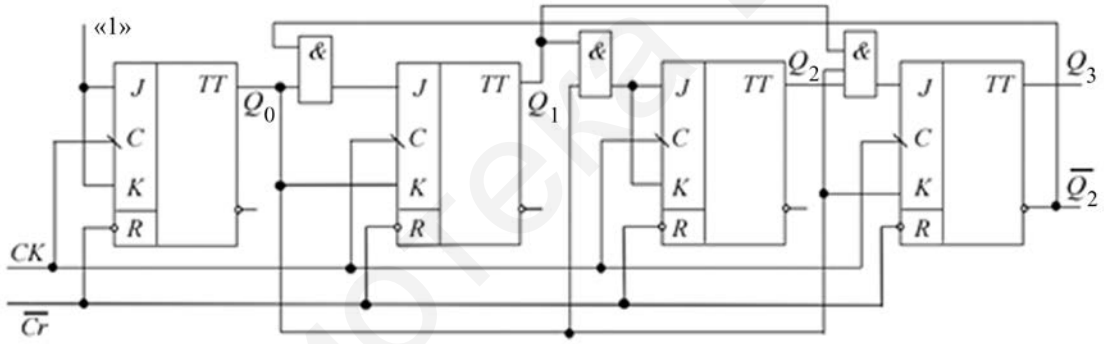
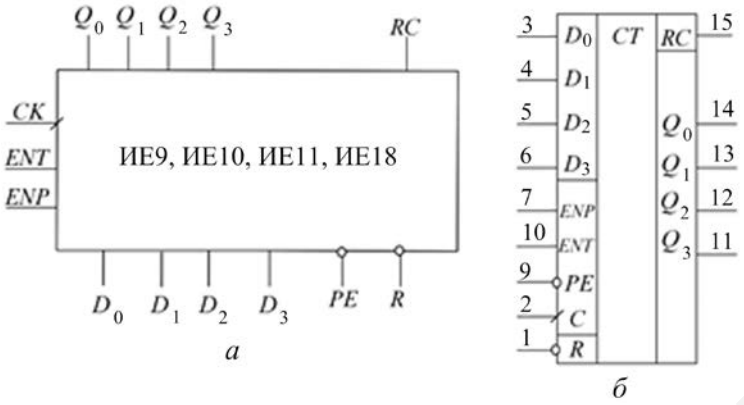


Рисунок 17.17 – Синхронный двоично-десятичный счетчик

Интегральные схемы синхронных счетчиков. Мы рассмотрели синтез синхронных счетчиков. Счетчики с любой счетной последовательностью и любым модулем счета могут быть синтезированы с использованием этого метода. Промышленностью выпускаются синхронные счетчики как интегральные схемы средней степени интеграции. Все счетчики содержат по четыре триггера с динамическим синхровходом. Изменения состояний триггеров, синхронная загрузка и синхронное обнуление происходит с положительным перепадом синхроимпульсов. В силу некоторых характерных особенностей синхронные счетчики могут быть подразделены на четыре группы.

Синхронные счетчики группы 1. К этой группе относятся счетчики ИЕ9(74160), ИЕ10(74161), ИЕ11(74162), ИЕ18(74163). Это двоично-десятичные и двоичные суммирующие счетчики с синхронной загрузкой и асинхронным

сбросом. Условное обозначение и цоколевка этих счетчиков показаны на рисунке 17.18.



а – условное обозначение; *б* – цоколевка

Рисунок 17.18 – Синхронные счетчики группы 1

Эти счетчики имеют два раздельных разрешающих входа *ENT* и *ENP*. Подавая на любой из этих входов лог. 0, счет останавливается асинхронно. Выход последовательного переноса *RC* находится в состоянии лог. 0 и становится лог. 1 всякий раз, когда счетчик достигает своего максимального значения (двоичное число 9 для двоично-десятичного счетчика и двоичное 15 для двоичного счетчика). Таблица 17.5 отражает функционирование счетчиков группы 1.

Таблица 17.5 – Таблица функционирования счетчиков

L	***ENP***	***ENT***	***Cr***	***CK***	**Операция**
0	*X*	*X*	1	↑	Установка (загрузка)
1	0	1	1	*X*	Остановка счета
1	*X*	0	1	*X*	Остановка счета, невозможность *RC*
X	*X*	*X*	0	*	Сброс в нуль
1	1	1	1	↑	Прямой счет

Обозначения в таблице:

1) «*» и «*X*» для ИЕ9, ИЕ10;

2) «↑» для ИЕ11, ИЕ18.

Пример 2. Построить делитель на 11, используя ИЕ18. Использовать *RC*-выход и установочные входы.

Решение. Для получения делителя на 11 счетчик устанавливается в двоичное состояние 0101 (десятичное 5). Когда счет достигнет 1111, на выходе *RC* появляется лог. 1. Эта лог. 1 используется для загрузки данных с установочных входов в счетчик (рисунок 17.19).

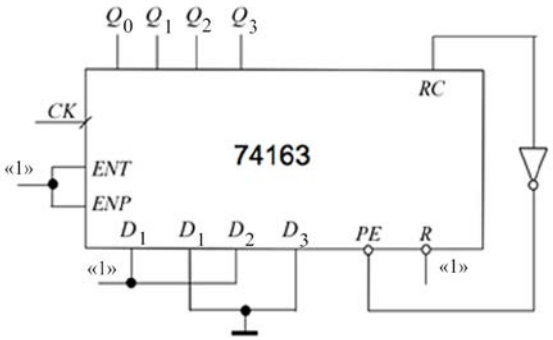


Рисунок 17.19 – Схема делителя на 11

Итак, для получения делителя на m на установочных входах должно быть: $D = 16 - m$ для двоичного счетчика и $D = 10 - m$ для десятичного счетчика.

Входы *ENT*, *ENP* и *RC* могут использоваться для организации каскадного соединения счетчиков (рисунок 17.20).

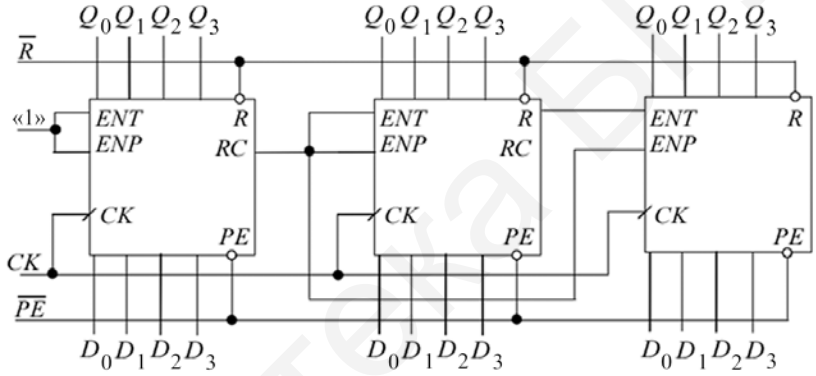
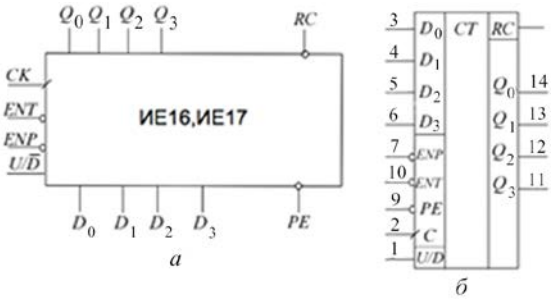


Рисунок 17.20 – Каскадное соединение счетчиков первой группы

Синхронные счетчики группы 2. К этой группе относятся счетчики ИЕ16(74168) и ИЕ17(74169). Это соответственно двоично-десятичный и двоичный реверсивные счетчики с синхронной установкой и без сброса (рисунок 17.21).



а – условное обозначение; *б* – цоколевка

Рисунок 17.21 – Условное обозначение и цоколевка ИЕ16, ИЕ17

Функция *ENT* и *ENP* та же, что и у счетчиков первой группы, но в этом случае эти входы активны при низком логическом входе. Выход последовательного переноса в нормальном состоянии равен лог. 1 и переходит в лог. 0, когда счет достигает максимального значения при прямом счете или когда счет достигает минимального значения при обратном счете.

Таблица 17.6 отражает функционирование счетчиков ИЕ16 и ИЕ17.

Таблица 17.6 – Таблица функционирования счетчиков ИЕ16 и ИЕ17

PE	***ENP***	***ENT***	***U/D***	***CK***	**Операция**
0	*X*	*X*	*X*	↑	Установка
1	1	0	*X*	*X*	Остановка счета
1	*X*	1	*X*	*X*	Остановка счета, запрет переноса
1	0	0	1	↑	Прямой счет
1	0	0	0	↑	Обратный счет

Сигнал на входе $U/\overline{D}$ определяет направление счета: $U/D = 1$ – для прямого счета и $U/\overline{D} = 0$ для обратного. В счетчиках этой группы отсутствует вход сброса – *R*. Поэтому, если необходимо остановить счет до достижения максимального значения, схема И-НЕ должна быть использована для детектирования состояния, соответствующего требуемому числу, и выход схемы И-НЕ соединяется с входом установки $\overline{PE}$. А входы установки дают необходимое начальное состояние счетчика.

Пример 3. Построить счетчик с начальным состоянием 0011 и конечным 1100, используя микросхему 1533ИЕ17.

Решение. Решение данной задачи представлено на рисунке 17.22.

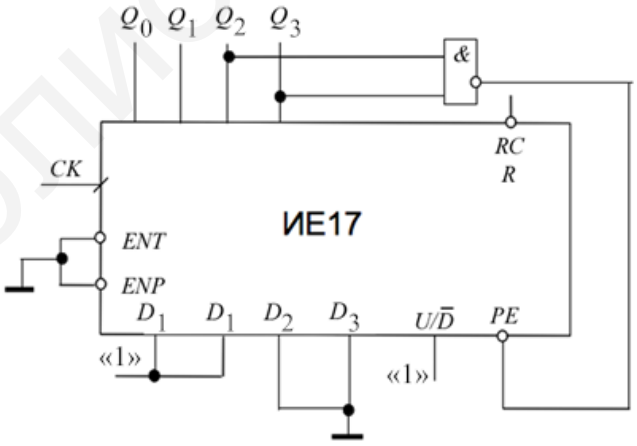


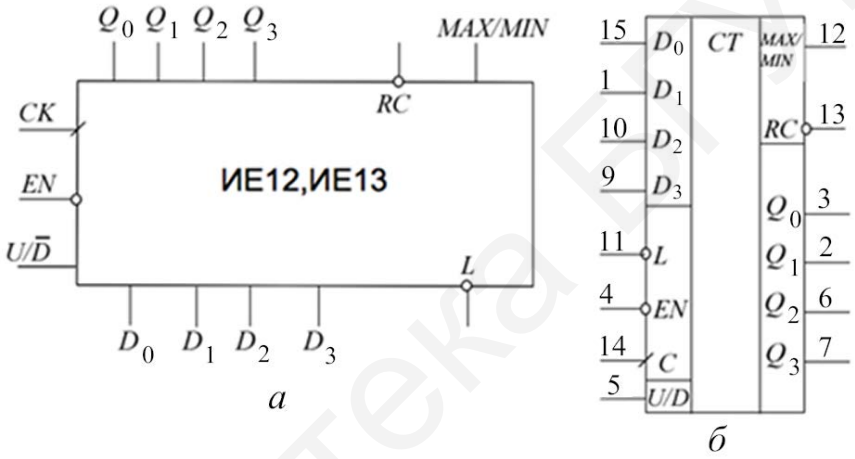
Рисунок 17.22 – Двоично-десятичный счетчик, осуществляющий счет в коде *Excess*-3

Когда счетчик достигнет состояния 1100, на выходе элемента И-НЕ и на входе *PE* появится лог. 0 и следующим синхроимпульсом (положительным

перепадом) счетчик установится в состояние 0011. Необходимо учитывать, что установка счетчика осуществляется синхронно.

Каскадное соединение счетчиков группы 2 осуществляется аналогично каскадному соединению счетчиков первой группы.

Синхронные счетчики группы 3. К этой группе относятся счетчики ИЕ12(74190) и ИЕ13(74191). Это соответственно двоично-десятичный и двоичный реверсивные счетчики с асинхронной установкой и без сброса. Эти счетчики имеют только один низкий активный разрешающий вход *EN*. Выход *MAX*/*MIN* (рисунок 17.23) используется для определения максимального или минимального состояния счетчика. На этом выходе в нормальном состоянии лог. 0, а лог. 1 появляется, когда максимальное состояние счетчика 1001 для ИЕ12 и 1111 для ИЕ13 при прямом счете или, когда минимальное состояние 0000 при обратном счете.



а – условное обозначение; *б* – цоколевка

Рисунок 17.23 – Условное обозначение и цоколевка ИЕ12, ИЕ13

Выход *RC* в нормальном состоянии лог. 1 переходит в лог. 0, когда счетчик достигает точки *MAX*/*MIN* и импульс синхронизации становится равным нулю. Таблица 17.7 отражает функционирование счетчиков ИЕ12, ИЕ13.

Таблица 17.7 – Таблица функционирования счетчиков ИЕ12, ИЕ13

L	***ENAB***	***U/D***	***CK***	**Операция**
X	1	*X*	*X*	Остановка счета
0	0	*X*	*X*	Установка счетчика
1	0	0	↑	Прямой счет
1	0	1	↑	Обратный счет

Синхронные счетчики третьей группы могут каскадироваться различными путями. Максимальное быстродействие будет, если использовать параллельный перенос (рисунок 17.24).

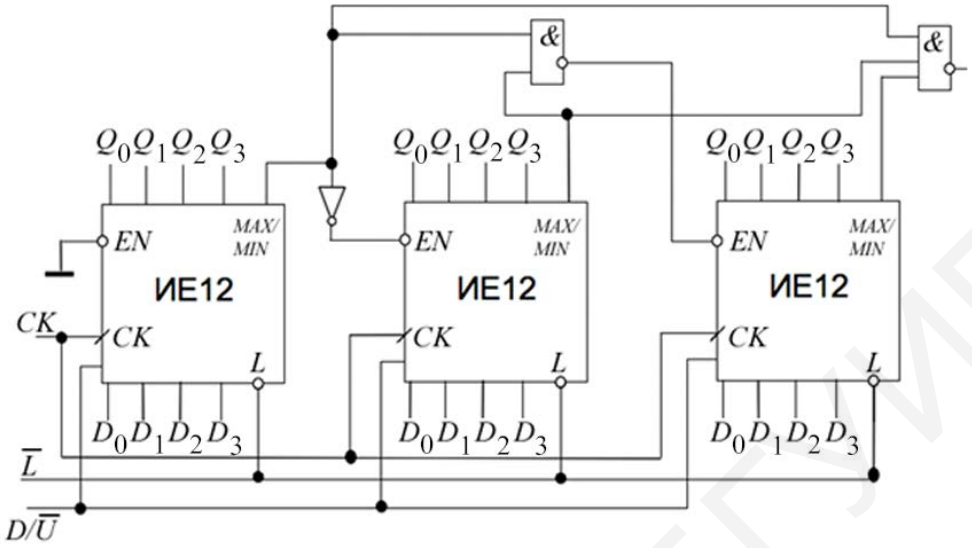
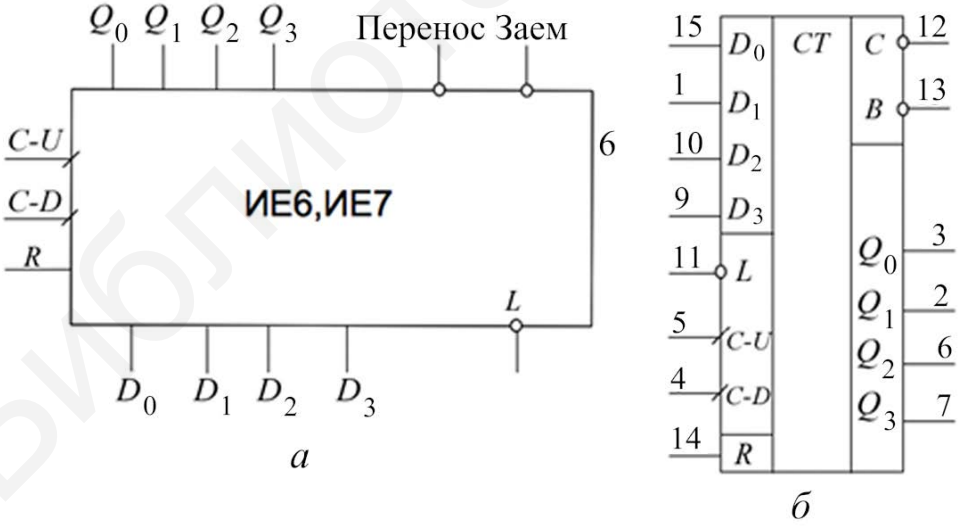


Рисунок 17.24 – Каскадные соединения счетчиков третьей группы

Синхронные счетчики группы 4. К этой группе относятся счетчики ИЕ6(74192) и ИЕ7(74193). Это соответственно двоично-десятичный и двоичный реверсивные счетчики с асинхронной загрузкой и сбросом (рисунок 17.25).



а – условное обозначение; *б* – цоколевка

Рисунок 17.25 – Условное обозначение и цоколевка ИЕ6 и ИЕ7

У этих счетчиков импульсы для прямого счета подаются на вход *C-U*, при этом вход *C-D* соединяется с лог. 1 и импульсы для обратного счета подаются на вход *C-D*, при этом вход *C-U* соединятся с лог. 1.

Выход *C* (перенос) и *B* (заем) обычно находятся в состоянии лог. 1. Выход переноса переходит в лог. 0, когда счетчик достигает максимального значения при прямом счете и вход *C-U* находится в лог. 0. Выход заем остается в лог. 1, когда счетчик работает от прямого входа *C-U*. Функция заем при обратном счете аналогична переносу при прямом счете.

Таблица 17.8 отражает функционирование счетчиков ИЕ6 и ИЕ7.

Таблица 17.8 – Таблица функционирования счетчиков ИЕ6 и ИЕ7

L	***R***	***C-U***	***C-D***	**Операция**
X	1	*X*	*X*	Сброс
1	0	↑	1	Прямой счет
1	0	↑	↑	Обратный счет
0	0	*X*	*X*	Установка
1	0	1	1	Остановка счета

При каскадировании этих счетчиков перенос и заем каждой ступени должны быть соединены с входами *C-U* и *C-D* последующей ступени соответственно. Для управления импульсами на прямой и обратный счетный вход может быть использована схема, показанная на рисунке 17.26.

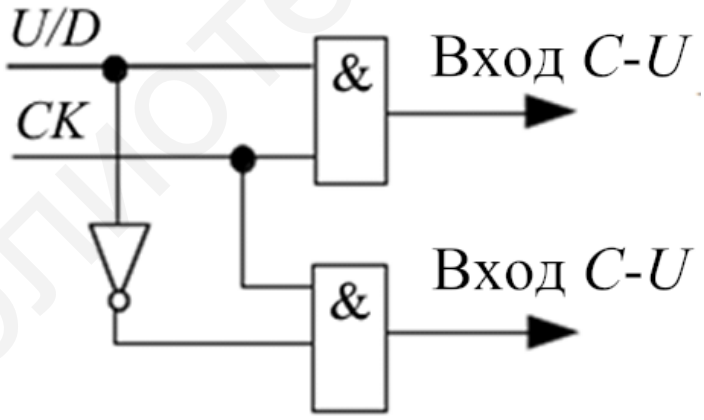


Рисунок 17.26 – Схема для управления счетными импульсами при прямом и обратном счете

17.3 Порядок выполнения лабораторной работы

Для выполнения лабораторной работы необходимо следующее оборудование и компоненты: универсальная лабораторная установка *IDL* – 800, 1533ТВ6 (74*ALS*107) – два *JK*-триггера с инверсным динамическим синхровходом, 1533ТМ2 (74*ALS*107) – два *D*-триггера с прямым динамическим

синхровходом, 1533ЛИ1 (74*ALS*08) – четыре логических элемента 2И, 1533ИЕ2 (74*ALS*90) – двоично-десятичный асинхронный счетчик, 1533ИЕ5 (74*ALS*93) – асинхронный двоичный счетчик, 155ИЕ4 (7492) – счетчик-делитель на 12, 555ИЕ15 (74*ALS*197) – асинхронный двоичный счетчик, 1533ИЕ19 (74*ALS*393) – два двоичных асинхронных счетчика, 1533ЛИ1 (74*ALS*08), 1533ТВ6 (74*ALS*107), 1533И9 (74*ALS*160), 1533ИЕ10 (74*ALS*161), 1533ИЕ11 (74*ALS*163), 1533ИЕ18 (74*ALS*163), 1533ИЕ16 (74*ALS*168), 1533ИЕ17 (74*ALS*169), 1533ИЕ12 (74*ALS*190), 1533ИЕ13 (74*ALS*191), 1533ИЕ6 (74*ALS*192), 1533ИЕ7 (74*ALS*193).

Задания для счетчиков:

Асинхронных:

Задание 1. Исследовать асинхронный двоичный счетчик на триггерах с инверсным динамическим синхровходом. Для этого:

1) построить асинхронный четырехразрядный счетчик на триггерах ТВ6;
2) исследовать работу счетчика;
3) результаты наблюдений представить в табличном виде и в виде временных диаграмм.

Задание 2. Исследовать асинхронный двоичный счетчик на триггерах с прямым динамическим синхровходом. Для этого:

1) построить асинхронный четырехразрядный счетчик на триггерах ТМ2;
2) исследовать работу счетчика. Результаты наблюдений представить в табличном виде и в виде временных диаграмм.

Задание 3. Исследовать двоично-десятичный счетчик. Для этого:

1) построить асинхронный двоично-десятичный счетчик;
2) исследовать работу счетчика. Результаты представить в табличном виде и в виде временных диаграмм.

Задание 4. Исследовать асинхронный счетчик ИЕ2. Для этого:

1) построить двоично-десятичный счетчик на основе ИЕ2;
2) исследовать работу счетчика. Результаты представить в табличном виде и виде временных диаграмм;
3) построить счетчик-делитель на 10 на основе ИЕ2;
4) исследовать работу счетчика-делителя. Результаты представить в табличном виде и виде временных диаграмм.

Задание 5. Исследовать асинхронный счетчик ИЕ5. Для этого:

1) построить счетчик с mod = 12 на основе ИЕ5;
2) исследовать работу счетчика. Результаты представить в табличном виде и в виде временных диаграмм.

Синхронных:

Задание 6. Выполнить синтез синхронного счетчика с произвольным модулем счета (согласно с заданием преподавателя).

17.4 Содержание отчета

1. Цель работы.
2. Схемы, исследуемые в работе.

3. Результаты исследований записать в таблицы и временные диаграммы.

4. Таблицы наблюдений и временные диаграммы работы исследуемых счетчиков.

5. Выводы.

17.5 Контрольные вопросы

1. Что такое цифровой счетчик?
2. По каким признакам классифицируются счетчики?
3. Нарисуйте схемы суммирующего и вычитающего счетчиков на триггерах с прямым и с инверсным динамическими входами.
4. Чем различаются между собой асинхронные и синхронные счетчики?
5. Каковы основные достоинства и недостатки асинхронных счетчиков?
6. Охарактеризуйте основные ИС асинхронных счетчиков.
7. Объясните использование входов «Загрузка», «Сброс» и «Установка».
8. Что такое электронной счетчик?
9. Чем отличается синхронный счетчик от асинхронного?
10. Каковы достоинства и недостатки синхронных счетчиков?
11. В чем заключается синтез синхронного счетчика?
12. Дайте характеристику особенностям модульных счетчиков.

18 Лабораторная работа №9

ГЕНЕРАТОРЫ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

18.1 Цель работы

Изучение синтеза генераторов последовательностей и исследование их работы.

18.2 Теоретические сведения

Схема, которая генерирует заданную последовательность битов синхронно с импульсами синхронизации, называется генератором последовательностей. Такие генераторы используются:

1) как счетчики;
2) генераторы псевдослучайных последовательностей;
3) генераторы заданной последовательности и заданного периода;
4) генераторы кодов.

Генераторы последовательностей являются одним из наиболее интересных применений регистров сдвига. Блок-схема генератора последовательностей показана на рисунке 18.1.

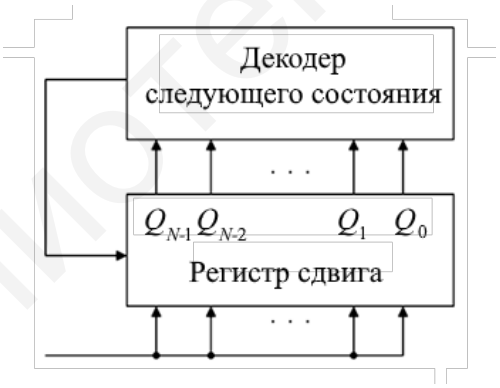


Рисунок 18.1 – Блок-схема генератора последовательностей

Выход декодера следующего состояния Y – это функция от $Q_{N-1}, Q_{N-2}, \dots, Q_0$, т. е. $Y = f(Q_{n-1}, Q_{n-2}, \dots, Q_0)$. Это схема подобна схеме кольцевого счетчика ($Y = Q_0$) или счетчика Джонсона ($Y = \bar{Q}_0$). Кольцевой счетчик и счетчик Джонсона являются частными случаями генераторов последовательностей.

Синтез генераторов последовательностей рассмотрим на примерах.

Пример 1. Синтезировать генератор последовательности … 1101011 … .

Решение. При синтезе генератора заданной последовательности необходимо определить число разрядов регистра сдвига и комбинационную схему декодера следующего состояния.

Минимально возможное число триггеров N в регистре сдвига для генерирования последовательности длинной S бит определяется по формуле

$$N \geq \log_2(S+1). \qquad (18.1)$$

В данном примере $S = 7$, поэтому минимально возможное значение $N = 3$. Однако это не значит, что это число триггеров является достаточным. Если данная последовательность ведет к семи различным состояниям регистра, то тогда три триггера будет достаточно, в противном случае число триггеров придется увеличить. Запишем состояние регистра в виде таблицы 18.1.

Таблица 18.1 – Таблица состояния регистра

Число синхроимпульсов	**Выходы триггеров**		
	Q_2	Q_1	Q_0
1	1	↑	↑
2	1	↑	↑
3	0	↑	↑
4	1	↓	↑
5	0	↑	↓
6	1	↓	↑
7	1	↑	↓

Допускаем, что данная последовательность генерируется на выходе Q_2. В таком случае на выходах Q_1 и Q_0 будет та же последовательность, только задержанная на один и два такта соответственно. Из таблицы (см. таблицу 18.1) видно, что не все состояния регистра отличаются от других (первая и вторая строки одинаковы, а также четвертая и шестая), что означает – число триггеров регистра $N = 3$ не является достаточным. Поэтому примем число $N = 4$ и построим аналогичную таблицу 18.2.

Таблица 18.2 – Таблица состояния регистра с $N = 4$

Число синхроимпульсов	**Выходы триггеров**				Y
	Q_3	Q_2	Q_1	Q_0	
1	**2**	**3**	**4**	**5**	**6**
1	1	1	1	0	1
2	1	1	1	1	0
3	0	1	1	1	1
4	1	0	1	1	0
5	0	1	0	1	1

1	2	3	4	5	6
6	1	0	1	0	1
7	1	1	0	1	1
8 (1)	1	1	1	0	1

Поскольку в данном случае все состояния регистра являются разными, добавим в эту таблицу колонку Y, в которой запишем требуемую последовательность на входе регистра.

Схему декодера получим, упрощая функцию $Y = f(Q_3, Q_2, Q_1, Q_0)$ с помощью карты Карно, рисунок 18.2.

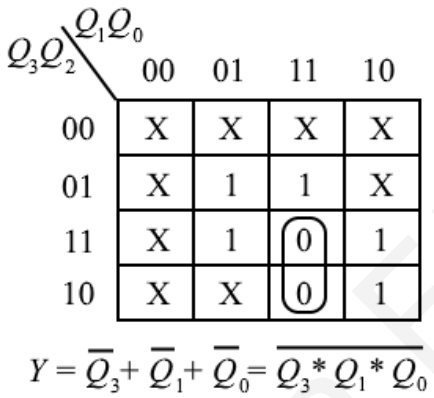


Рисунок 18.2 – Упрощение ФАЛ декодера следующего состояния

Упрощенная схема генератора последовательности … 1101011 … показана на рисунке 18.3.

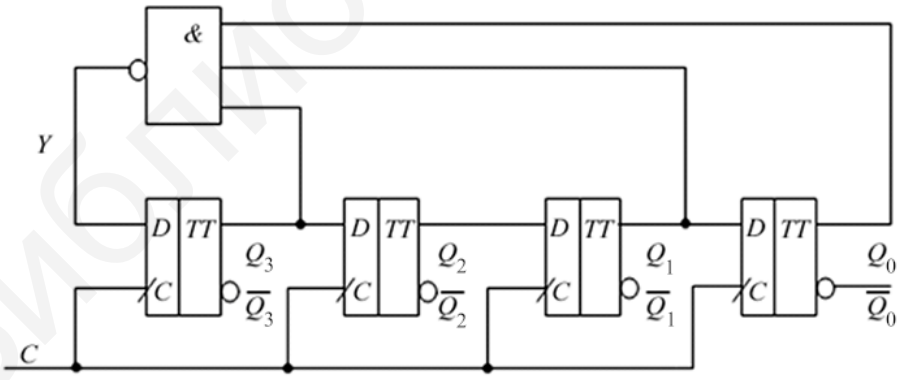


Рисунок 18.3 – Логическая структура генератора последовательности … 1101011 …

Пример 2. Синтезировать генератор последовательности … 1101001… .

Решение. Минимально возможное число триггеров регистра $N = 3$. Проверим, является ли это число достаточным. Для этого построим таблицу 18.3.

Таблица 18.3 – Таблица состояния регистра с $N = 3$

Импульсы синхронизации	**Выходы триггеров**			**Y**
	Q_2	Q_1	Q_0	
1	1	1	1	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	0
5	0	1	0	1
6	1	0	1	1
7	1	1	0	1

Из таблицы 18.3 видно, что все состояния регистра являются неодинаковыми, т. е. число триггеров $N = 3$ является достаточным для реализации генератора, и поэтому добавим в эту таблицу колонку с требуемой на входе регистра последовательностью. Схему декодера получим, упрощая функцию $Y = f(Q_2, Q_1, Q_0)$ с помощью карты Карно (рисунок 18.4).

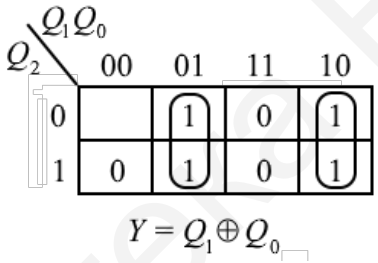


Рисунок 18.4 – Карта Карно для декодера

На рисунке 18.5 показана упрощенная схема генератора последовательности … 1101001… .

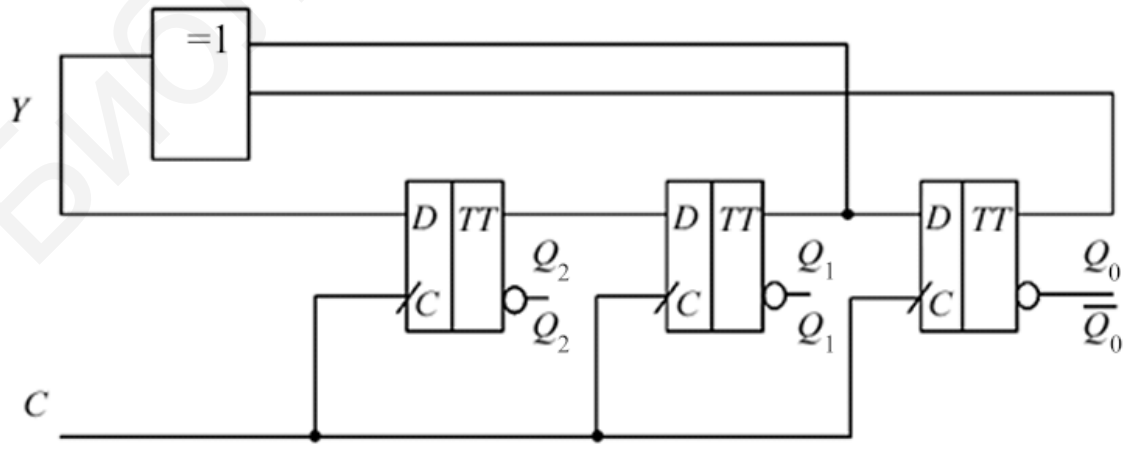


Рисунок 18.5 – Логическая структура генератора последовательности … 1101001…

Длина последовательности, генерируемой генератором (см. рисунок 18.5), равна $S = 2^N - 1 = 2^3 - 1 = 7$.

Генераторы, которые генерируют последовательности длины, называются генераторами последовательностей максимальной длины и вычисляются с помощью выражения

$$S = 2^N - 1. \tag{18.2}$$

Такие генераторы широко используются для генерирования помехоустойчивых кодов.

Пример реализация ПСП. Используя неприводимый и примитивный полином $x^4 + x^3 + 1$ порядка $n = 4$ и рекуррентное правило, составим рекуррентную формулу:

$$a_i = a_{i-4} + a_{i-3}. \tag{18.3}$$

При построении генератора необходимо учесть, что запрещенной начальной комбинацией в регистре сдвига является комбинация из всех нулей.

Исходя из нашего полинома, построим структурную схему ПСП генератора (рисунок 18.6).

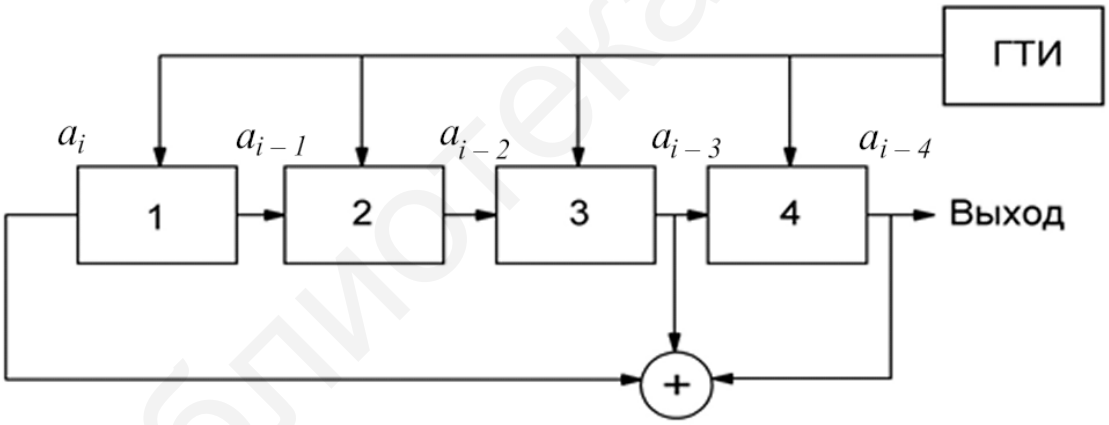


Рисунок 18.6 – Структурная схема генератора ПСП

На рисунке 18.6 блоки 1, 2, 3, 4 обозначают *D*-триггеры.

Результат тактирования ПСП представлен в таблице 18.4.

Таблица 18.4 – Результат тактирования ПСП

Номер тактирования	**Двоичная последовательность**	***Y***
1	**2**	**3**
0	1001	–
1	1100	1

1	2	3
2	0110	0
3	1011	0
4	0101	1
5	1010	1
6	1101	0
7	1110	1
8	1111	0
9	0111	1
10	0011	1
11	0001	1
12	1000	1
13	0100	0
14	0010	0
15	1001	0

18.3 Порядок выполнения лабораторной работы

Оборудование и компоненты: универсальная лабораторная установка *IDL*-800, ИС 1533ИР16 (74*ALS*295) – четырехразрядный регистр сдвига, ИС 1533ЛА4 (74*ALS*10) – три логических элемента 3И-НЕ, ИС 1533ЛП5 (74*ALS*86) – четыре двухвходовых логических элемента Исключающее ИЛИ.

Задание 1. Исследовать работу генератора последовательности (см. рисунок 8.2). Для этого:

1) используя ИС 1533ИР16 и 1533ЛА4, собрать схему генератора последовательности;

2) подавая одиночные синхроимпульсы (использовать антидребезговую кнопку), исследовать работу генератора последовательности;

3) результаты представить в табличной форме.

Задание 2. Исследовать работу генератора последовательности (см. рисунок 8.3). Для этого:

1) используя ИС К555ИР16 и 1533ЛП5, собрать схему генератора последовательности;

2) подавая одиночные синхроимпульсы, исследовать работу генератора;

3) результаты представить в табличной форме.

Задание 3. Синтезировать и исследовать генератор последовательности …1001011… . Для этого:

1) синтезировать и построить генератор последовательности …1001011…;

2) исследовать работу генератора, результаты предоставить в табличной форме.

Задание 4. Синтезировать и исследовать генератор последовательности. Для этого:

1) синтезировать и построить генератор последовательности по вариантам (таблица 8.5);

2) исследовать работу генератора, результаты предоставить в табличной форме.

Таблица 8.5 – Варианты заданий

Номер варианта	Начальная двоичная комбинация	Полином
1	1001	$x^4 + x^3 + 1$
2	1010	$x^4 + x + 1$
3	0110	$x^4 + x + 1$
4	0010	$x^4 + x^3 + 1$
5	1010	$x^3 + x^2 + 1$
6	1101	$x^4 + x^3 + 1$
7	0111	$x^2 + x + 1$
8	1011	$x^4 + x + 1$
9	0101	$x^4 + x^3 + 1$
10	1101	$x^3 + x + 1$

18.4 Содержание отчета

1. Цель работы.
2. Схемы генераторов исследуемых в работе.
3. Таблицы результатов наблюдений.
4. Выводы.

18.5 Контрольные вопросы

1. Чему равняется минимально возможное число триггеров для генератора последовательности длиной $S = 27$?
2. Какие генераторы называются генераторами последовательности максимальной длины?
3. Приведите примеры генераторов последовательности максимальной длины.
4. Обьясните работу генераторов, иследованных в работе.

ПРИЛОЖЕНИЕ А
(*справочное*)

Система условных обозначений и электрические параметры микросхем стандартной логики производства НПО «Интеграл»

Условные обозначения цифровых микросхем установлены в соответствии с ГОСТ 17021–88. Продукция НПО «Интеграл» ориентирована как на внутренний рынок, так и на экспорт. Поэтому введено международное условное обозначение ИМС.

Обозначение ИМС, выпускаемых НПО «Интеграл», состоит из двух элементов:

- первый элемент – две буквы *IN* (код НПО «Интеграл»);
- второй элемент – буквенно-цифровое обозначение в соответствии с обозначением зарубежного прототипа.

Например, прототип микросхемы зарубежного производства *MC*74*HC*08*AN*, выпускаемый на НПО «Интеграл», имеет обозначение *IN*74*HC*08*AN*.

Буква в конце международного обозначения ИМС указывает на тип корпуса:

- *N* – для пластмассовых *DIP*-корпусов с дюймовым шагом между выводами;
- *D* – для пластмассовых *SO*-корпусов (ширина корпуса 4 мм);
- *DW* – для пластмассовых 20-, 24- и 28-выводных *SO*-корпусов (ширина корпуса 7,6 мм).

Микросхема *IN*74*HC*08*AN* по ГОСТ 17021–88 имеет обозначение ЭКР1564ЛИ1, а *IN*74*HC*08*AD* – ЭКФ1564ЛИ1. Следовательно, буквы перед номером серии указывают на тип корпуса.

Основные электрические параметры микросхем стандартной логики схемотехники КМОП приведены в таблицах А.1–А.8.

Таблица А.1 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 1554 (*IN*74*AC*XXX*N*, *D*, *DW*)

Международное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
*IN*74*AC*00	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	8,5	7,0
*IN*74*AC*02	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	6,5	7,0
*IN*74*AC*04	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	7,5	7,0

1	2	3	4	5	6	7	8	9	10
*IN*74*AC*05	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	7,5	7,0
*IN*74*AC*08	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	8,5	7,5
*IN*74*AC*10	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	8,0	6,5
*IN*74*AC*11	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	8,5	7,5
*IN*74*AC*20	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	8,0	7,0
*IN*74*AC*21	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	6,5	7,0
*IN*74*AC*27	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	6,5	7,0
*IN*74*AC*32	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	8,5	7,5
*IN*74*AC*34	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	6,5	7,0
*IN*74*AC*86	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	9,0
*IN*74*AC*125	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	7,5
*IN*74*AC*138	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	10,5	10,5
*IN*74*AC*139	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN*74*AC*151	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11	11
*IN*74*AC*153	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	10,5	10,0
*IN*74*AC*157	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	7,0	7,0
*IN*74*AC*158	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	7,5	6,5
*IN*74*AC*240	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN*74*AC*241	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN*74*AC*244	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN*74*AC*245	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	7,0	7,0
*IN*74*AC*251	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	14	14
*IN*74*AC*253	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,0
*IN*74*AC*257	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,0
*IN*74*AC*258	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,0
*IN*74*AC*620	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	7,5	7,0
*IN*74*AC*623	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	7,5	7,0

Примечания:
1) напряжение питания: от 2 до 6 В;
2) диапазон рабочих температур: от минус 45 до плюс 85 °С;
3) параметры указаны для напряжения источника питания $U_{\text{И.П}} = 5$ В;
4) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $|I_{\text{вых}}| = 0$ мкА;
5) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $|I_{\text{вых}}| \leq 50$ мкА

Таблица А.2 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 1594 (*IN*74*ACT*XXX*N*, *D*, *DW*)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
*IN*74*ACT*00	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,5	8,0
*IN*74*ACT*02	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	10,0

1	2	3	4	5	6	7	8	9	10
*IN74ACT*04	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	8,5
*IN74ACT*05	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	8,5
*IN74ACT*08	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	10,0
*IN74ACT*10	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	9,5
*IN74ACT*11	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	9,5
*IN74ACT*20	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	7,0
*IN74ACT*21	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	10,0
*IN74ACT*27	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	10,0
*IN74ACT*32	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	10,0
*IN74ACT*34	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	9,0	10,0
*IN74ACT*86	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	10,5
*IN74ACT*125	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,0	10,0
*IN74ACT*138	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,5
*IN74ACT*139	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	10,5
*IN74ACT*151	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	12,5	13,5
*IN74ACT*153	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	11,0
*IN74ACT*157	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	8,5	8,5
*IN74ACT*158	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	8,5	8,5
*IN74ACT*240	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN74ACT*241	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN74ACT*244	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,5
*IN74ACT*245	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	8,0	9,0
*IN74ACT*251	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	13,0	14,0
*IN74ACT*253	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	12,5
*IN74ACT*257	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,0
*IN74ACT*258	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	8,0
*IN74ACT*620	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	8,5	9,0
*IN74ACT*623	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	8,5	9,0

Примечания:
1) напряжение питания: 5,0 В (±10 %);
2) диапазон рабочих температур: от минус 45 до плюс 85 °С;
3) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $|I_{\text{вых}}|$ = 0 мкА;
4) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $|I_{\text{вых}}| \leq 50$ мкА

Таблица А.3 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 1554 (*IN74ACXXXN*, *D*, *DW*)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
*IN74AC*74	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,5	10,5
*IN74AC*109	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,5	10,5

1	2	3	4	5	6	7	8	9	10
*IN*74*AC*112	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	13,5	13,5
*IN*74*AC*161	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	10,0
*IN*74*AC*163	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	10,0
*IN*74*AC*164	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	10,5	10,5
*IN*74*AC*174	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	9
*IN*74*AC*175	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	9,5	10,5
*IN*74*AC*192	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	11,0
*IN*74*AC*193	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,0
*IN*74*AC*273	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	10,0	11,0
*IN*74*AC*373	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,0
*IN*74*AC*374	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	9,5
*IN*74*AC*533	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,0
*IN*74*AC*534	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	12,0	11,0
*IN*74*AC*4015	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	15,0	15,0
*IN*74*AC*4520	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	15,0	15,0

Примечания:
1) напряжение питания: от 2 до 6 В;
2) диапазон рабочих температур: от минус 45 до плюс 85 °С;
3) параметры указаны для напряжения источника питания $U_{\text{И.П}}$ = 5 В;
4) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $|I_{\text{вых}}|$ = 0 мкА;
5) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $|I_{\text{вых}}| \leq 50$ мкА

Таблица А.4 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 1564 (*IN*74*HC*XXX*N*, *D*, *DW*)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
*IN*74*HC*00*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	19	19
*IN*74*HC*02*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	20	20
*IN*74*HC*03*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	30	30
*IN*74*HC*04*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	19	19
*IN*74*HC*05*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	30	30
*IN*74*HC*08*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	19	19
*IN*74*HC*10*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	24	24
*IN*74*HC*11*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	31	31
*IN*74*HC*20*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	23	23
*IN*74*HC*21*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	28	28
*IN*74*HC*22*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	30	30
*IN*74*HC*27*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	23	23
*IN*74*HC*30*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	44	44
*IN*74*HC*32*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	19	19

1	2	3	4	5	6	7	8	9	10
*IN*74HC86*AN*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	30	30
*IN*74*HC*125*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	23	23
*IN*74*HC*138*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	34	34
*IN*74*HC*139*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	29	29
*IN*74*HC*151*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	46	46
*IN*74*HC*153*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	35	35
*IN*74*HC*154*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	48	48
*IN*74*HC*155*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	29	29
*IN*74*HC*157*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	26	26
*IN*74*HC*158*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	26	26
*IN*74*HC*240*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	20	20
*IN*74*HC*241*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	23	23
*IN*74*HC*244*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	23	23
*IN*74*HC*245*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	20	20
*IN*74*HC*251*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	46	46
*IN*74*HC*253*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	35	35
*IN*74*HC*257*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	25	25
*IN*74*HC*258*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	25	25
*IN*74*HC*283*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	68	68
*IN*74*HC*620*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	35	35
*IN*74*HC*623*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	35	35

Примечания:
1) напряжение питания: от 2 до 6 В;
2) диапазон рабочих температур: от минус 45 до плюс 85 °С;
3) параметры указаны для напряжения источника питания $U_{\text{И.П}}$ = 5 В;
4) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $|I_{\text{вых}}|$= 0 мкА;
5) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $|I_{\text{вых}}| \leq 50$ мкА

Таблица А.5 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 5564 (*IN*74*HCT*XXX*AN*, *D*, *DW*)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
*IN*74*HCT*00*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	24	24
*IN*74*HCT*02*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	20	20
*IN*74*HCT*04*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	19	21
*IN*74*HCT*08*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	24	24
*IN*74*HCT*10*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	24	24
*IN*74*HCT*20*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	35	35
*IN*74*HCT*27*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	23	23
*IN*74*HCT*30*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	44	44

1	2	3	4	5	6	7	8	9	10
IN74HCT32A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	25	25
IN74HCT86A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	30	30
IN74HCT125A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	23	23
IN74HCT138A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	38	38
IN74HCT139A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	29	29
IN74HCT151A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	46	46
IN74HCT153A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	43	43
IN74HCT155A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	29	29
IN74HCT157A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	26	26
IN74HCT240A	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	25	25
IN74HCT241A	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	29	29
IN74HCT244A	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	25	25
IN74HCT245A	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	20	20
IN74HCT251A	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	46	46
IN74HCT283A	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	68	68
IN74HCT620A	–0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	45	45
IN74HCT623A	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	45	45

Примечания:
1) напряжение питания: 5,0 В (±10 %);
2) диапазон рабочих температур: от минус 45 до плюс 85 °С;
3) максимальный потребляемый $I_{\text{пот}}$ указан для выходного тока $|I_{\text{вых}}|$= 0 мкА;
4) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $|I_{\text{вых}}| \leq 50$ мкА

Таблица А.6 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 1594 (*IN74ACTXXXN, D, DW*) (дополнительно)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
IN74ACT74	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	13,0	11,5
IN74ACT109	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	13,0	11,5
IN74ACT112	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	15,0	14,5
IN74ACT161	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	12,0
IN74ACT163	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	12,0
IN74ACT164	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	40	13,0	11,5
IN74ACT174	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,5
IN74ACT175	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	12,0
IN74ACT192	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	13,5	13,5
IN74ACT193	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	13,5	13,5
IN74ACT273	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,0	12,0
IN74ACT373	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	12,0	12,0
IN74ACT374	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	12,0	11,0
IN74ACT533	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	11,5	11,0

1	2	3	4	5	6	7	8	9	10
*IN*74*ACT*534	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	12,5	12,0
*IN*74*ACT*4015	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	15,0	15,0
*IN*74*ACT*4520	0,1	4,9	– 1,0	+ 1,0	+ 24	– 24	80	15,0	15,0
Примечания: 1) напряжение питания: 5,0 В (±10%); 2) диапазон рабочих температур – от минус 45 до плюс 85 °С; 3) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $\lvert I_{\text{вых}}\rvert$ = 0 мкА; 4) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $\lvert I_{\text{вых}}\rvert \leq 50$ мкА									

Таблица А.7 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 1564 (*IN*74*HC*XXX*N*, *D*, *DW*) (дополнительно)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
*IN*74HC74*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	20	30	30
*IN*74HC75*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	31	31
*IN*74HC109*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	44	44
*IN*74HC112*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	31	31
*IN*74*HC*161*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	23	25
*IN*74*HC*163*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	23	25
*IN*74*HC*164*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	44	44
*IN*74*HC*165*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	38	38
*IN*74*HC*174*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	28	28
*IN*74*HC*175*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	38	38
*IN*74*HC*192*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	48	48
*IN*74*HC*193*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	48	48
*IN*74*HC*273*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	28	28
*IN*74*HC*279*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	25	30
*IN*74*HC*373*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	31	31
*IN*74*HC*374*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	31	31
*IN*74*HC*533*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	38	38
*IN*74*HC*534*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	45	45
*IN*74*HC*4015*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	80	44	44
Примечания: 1) напряжение питания: от 2 до 6 В; 2) диапазон рабочих температур: от минус 45 до плюс 85 °С; 3) параметры указаны для напряжения источника питания $U_{\text{И.П}}$ = 5 В; 4) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $\lvert I_{\text{вых}}\rvert$ = 0 мкА; 5) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $\lvert I_{\text{вых}}\rvert \leq 50$ мкА									

Таблица А.8 – Основные электрические параметры интегральных микросхем стандартной логики производства НПО «Интеграл» схемотехники КМОП серии 5564 (*IN*74*HCT*XXX*N*, *D*, *DW*)

Между-народное обозначение	$U^0_{\text{вых}}$ (V_{OL}), не более	$U^1_{\text{вых}}$ (V_{OH}), не менее	$I^0_{\text{вх}}$ (I_{IL}), не более	$I^1_{\text{вх}}$ (I_{IH}), не более	$I^0_{\text{вых}}$ (I_{OL}), не более	$I^1_{\text{вых}}$ (I_{OH}), не более	$I_{\text{пот}}$ (I_{CC}), не более	$t^{0,1}_{\text{зд.р}}$ (t_{PLH}), не более	$t^{1,0}_{\text{зд.р}}$ (t_{PHL}), не более
	В	В	мкА	мкА	мА	мА	мкА	нс	нс
*IN*74*HCT*74*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	10	30	30
*IN*74*HCT*163*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	43	51
*IN*74*HCT*164*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	44	44
*IN*74*HCT*165*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	30	30
*IN*74*HCT*174*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	30	30
*IN*74*HCT*273*A*	0,1	4,9	– 1,0	+ 1,0	+ 4,0	– 4,0	40	28	28
*IN*74*HCT*373*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	35	35
*IN*74*HCT*374*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	40	24	24
*IN*74*HCT*533*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	44	44
*IN*74*HCT*534*A*	0,1	4,9	– 1,0	+ 1,0	+ 6,0	– 6,0	80	44	44

Примечания:
1) напряжение питания: 5,0 В (±10 %);
2) диапазон рабочих температур: от минус 45 до плюс 85 °С;
3) максимальный потребляемый ток $I_{\text{пот}}$ указан для выходного тока $|I_{\text{вых}}| = 0$ мкА;
4) уровни выходных напряжений $U^0_{\text{вых}}$ и $U^1_{\text{вых}}$ указаны для выходного тока $|I_{\text{вых}}| \leq 50$ мкА

Условно-графическое обозначение, цоколевка и таблицы истинности интегральных микросхем стандартной логики схемотехники КМОП производства НПО «Интеграл» представлены в таблицах А.9–А.23.

Таблица А.9 – Четыре логических элемента 2И-НЕ

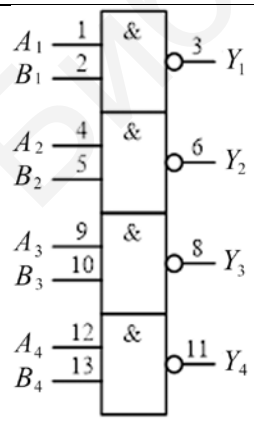
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	Входы | Выходы *A* | *B* | *Y* 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	$Y = \overline{A \wedge B} = A\|B$	Зарубежный аналог *IN*74*AC*00 *IN*74*ACT*00 *IN*74*HC*00*A* *IN*74*HCT*00*A* Отечественный аналог 1554ЛА3 1594ЛА3 1564ЛА3 5564ЛА3

Таблица А.10 – Четыре логических элемента 2ИЛИ-НЕ

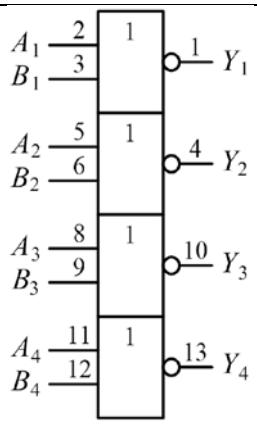
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	Входы: A, B; Выходы: Y 0 0 1 0 1 0 1 0 0 1 1 0	$Y = \overline{A \wedge B} =$ $= A \downarrow B$	Зарубежный аналог *IN*74*AC*02 *IN*74*ACT*02 *IN*74*HC*02*A* *IN*74*HCT*02*A* Отечественный аналог 1554ЛЕ1 1594ЛЕ1 1564ЛЕ1 5564ЛЕ1

Таблица А.11 – Четыре логических элемента 2И-НЕ с открытым стоком

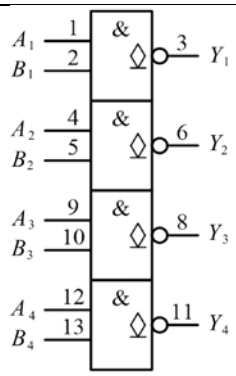
Условно-графическое обозначение	**Таблица истинности**	**Обозначения**
 14 – питание; 7 – общий	Входы: A, B; Выходы: Y 0 0 Z 0 1 Z 1 0 Z 1 1 0 *Примечание* – «Z» – третье состояние	Зарубежный аналог *IN*74*HC*03*A* Отечественный аналог 1564ЛА9

Таблица А.12 – Шесть логических элементов НЕ

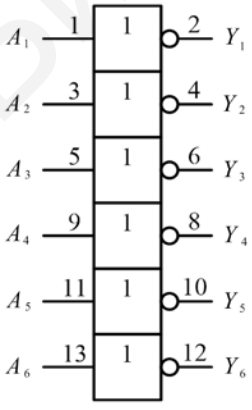
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	Вход: A; Выход: Y 0 1 1 0	$Y = \bar{A}$	Зарубежный аналог *IN*74*AC*04 *IN*74*ACT*04 *IN*74*HC*04*A* *IN*74*HCT*04*A* Отечественный аналог 1554ЛН1 1594ЛН1 1564ЛН1 5564ЛН1

Таблица А.13 – Шесть логических элементов НЕ с открытым стоком

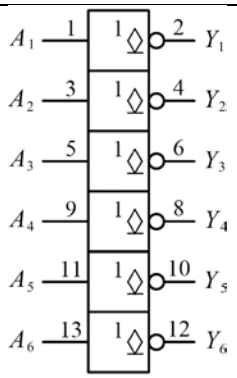
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	<table><tr><td>Вход</td><td>Выход</td></tr><tr><td>A</td><td>Y</td></tr><tr><td>0</td><td>Z</td></tr><tr><td>1</td><td>0</td></tr></table>	$Y = \bar{A}$	Зарубежный аналог *IN*74*AC*05 *IN*74*ACT*05 *IN*74*HC*05*A* Отечественный аналог 1554ЛН2 1594ЛН2 1564ЛН2

Таблица А.14 – Четыре логических элемента 2И

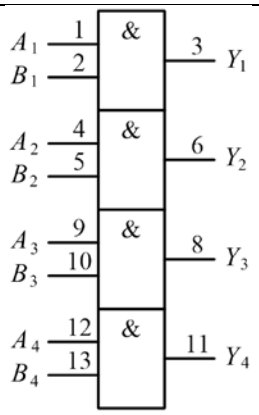
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	<table><tr><td colspan="2">Входы</td><td>Выходы</td></tr><tr><td>A</td><td>B</td><td>Y</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$Y = A \wedge B$	Зарубежный аналог *IN*74*AC*08 *IN*74*ACT*08 *IN*74*HC*08*A* *IN*74*HCT*08*A* Отечественный аналог 1554ЛИ1 1594ЛИ1 1564ЛИ1 5564ЛИ1

Таблица А.15 – Шесть повторителей

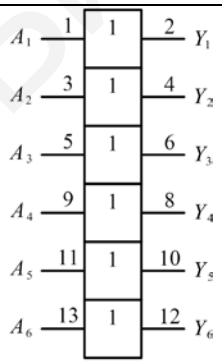
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	<table><tr><td>Вход</td><td>Выход</td></tr><tr><td>A</td><td>Y</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	$Y = A$	Зарубежный аналог *IN*74*AC*34 *IN*74*ACT*34 Отечественный аналог 1554ЛИ9 1594ЛИ9

Таблица А.16 – Три логических элемента «3 И-НЕ»

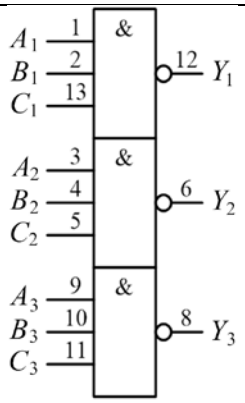
Условно-графическое обозначение	Таблица истинности	Выражение	Обозначения
 14 – питание; 7 – общий	Входы: A, B, C; Выходы: Y 0 X X 1 X 0 X 1 X X 0 1 1 1 1 0 *Примечание – «X» –* 0 или 1	$Y = \overline{A \wedge B \wedge C}$	Зарубежный аналог *IN*74*AC*10 *IN*74*ACT*10 *IN*74*HC*10*A* *IN*74*HCT*10*A* Отечественный аналог 1554ЛА4 1594ЛА4 1564ЛА4 5564ЛА4

Таблица А.17 – Три логических элемента 3И

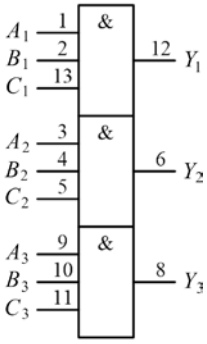
Условно-графическое обозначение	Таблица истинности	Выражение	Обозначения
 14 – питание; 7 – общий	Входы: A, B, C; Выходы: Y 0 X X 0 X 0 X 0 X X 0 0 1 1 1 1 *Примечание – «X» –* 0 или 1	$Y = A \wedge B \wedge C$	Зарубежный аналог *IN*74*AC*11 *IN*74*ACT*11 *IN*74*HC*11*A* Отечественный аналог 1554ЛИ3 1594ЛИ3 1564ЛИ3

Таблица А.18 – Два логических элемента 4И-НЕ

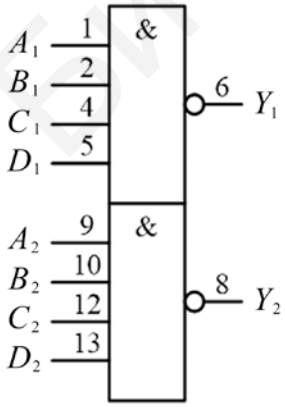
Условно-графическое обозначение	Таблица истинности	Выражение	Обозначения
 14 – питание; 7 – общий	Входы: A, B, C, D; Выходы: Y 0 X X X 1 X 0 X X 1 X X 0 X 1 X X X 0 1 1 1 1 1 0 *Примечание – «X» –* 0 или 1	$Y = \overline{A \wedge B \wedge C \wedge D} = \overline{A\|B\|C\|D}$	Зарубежный аналог *IN*74*AC*20 *IN*74*ACT*20 *IN*74*HC*20*A* *IN*74*HCT*20*A* Отечественный аналог 1554ЛА1 1594ЛА1 1564ЛА1 5564ЛА1

Таблица А.19 – Два логических элемента 4И

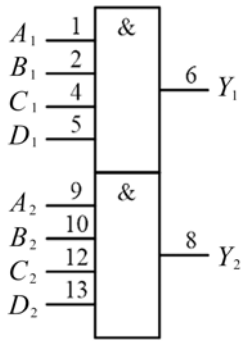
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	Входы: *A*, *B*, *C*, *D*; Выходы: *Y* 0 *X* *X* *X* → 0 *X* 0 *X* *X* → 0 *X* *X* 0 *X* → 0 *X* *X* *X* 0 → 0 1 1 1 1 → 1 *Примечание* – «*X*» – 0 или 1	$Y = A \wedge B \wedge C$	Зарубежный аналог *IN*74*AC*21 *IN*74*ACT*21 *IN*74*HC*21*A* Отечественный аналог 1554ЛИ6 1594ЛИ6 1564ЛИ6

Таблица А.20 – Три логических элемента 3ИЛИ-НЕ

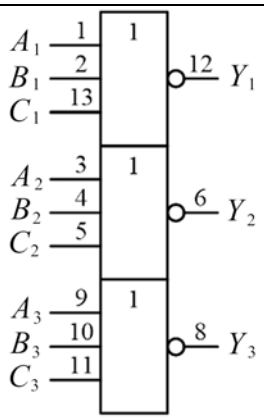
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	Входы: *A*, *B*, *C*; Выходы: *Y* 0 0 0 → 1 0 0 1 → 0 0 1 0 → 0 1 0 0 → 0	$Y = \overline{A \vee B \vee C} = A \downarrow B \downarrow C$	Зарубежный аналог *IN*74*AC*27 *IN*74*ACT*27 *IN*74*HC*27*A* *IN*74*HCT*27*A* Отечественный аналог 1554ЛЕ4 1594ЛЕ4 1564ЛЕ4 5564ЛЕ4

Таблица А.21 – Четыре логических элемента 2ИЛИ

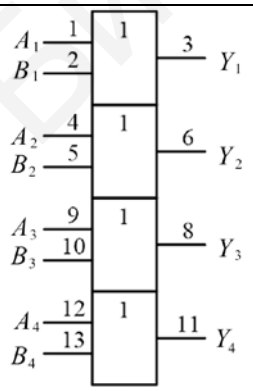
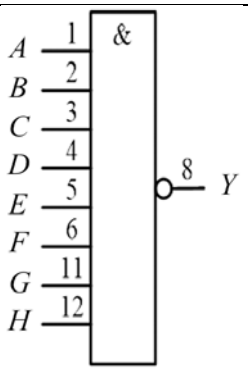
Условно-графическое обозначение	**Таблица истинности**	**Выражение**	**Обозначения**
 14 – питание; 7 – общий	Входы: *A*, *B*; Выходы: *Y* 0 0 → 0 0 1 → 1 1 0 → 1 1 1 → 1	$Y = A \vee B$	Зарубежный аналог *IN*74*AC*32 *IN*74*ACT*32 *IN*74*HC*32*A* *IN*74*HCT*32*A* Отечественный аналог 1554ЛЛ1 1594ЛЛ1 1564ЛЛ1 5564ЛЛ1

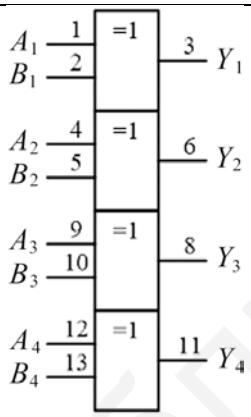
Таблица А.22 – Логический элемент 8И-НЕ

<table>
<tr><th>Условно-графическое обозначение</th><th>Таблица истинности</th><th>Выражение</th><th>Обозначения</th></tr>
<tr>
<td> 

14 – питание;
7 – общий</td>
<td>
<table>
<tr><td colspan="8">Входы</td><td>Вы-ходы</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td><td>Y</td></tr>
<tr><td>0</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>0</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>0</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>0</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>X</td><td>X</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>X</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>X</td><td>1</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>
Примечание – «X» – 0 или 1
</td>
<td>$Y = A|B|C|D|$
$|E|F|G|H|$</td>
<td>Зарубежный аналог
IN74HC30A
IN74HCT30A

Отечественный аналог
1564ЛА2
5564ЛА2</td>
</tr>
</table>

Таблица А.23 – Четыре логических элемента Исключающее ИЛИ

<table>
<tr><th>Условно-графическое обозначение</th><th>Таблица истинности</th><th>Выражение</th><th>Обозначения</th></tr>
<tr>
<td> 

14 – питание; 7 – общий</td>
<td>
<table>
<tr><td colspan="2">Входы</td><td>Выходы</td></tr>
<tr><td>A</td><td>B</td><td>Y</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
<td>$Y = A \oplus B$</td>
<td>Зарубежный аналог
IN74AC86
IN74ACT86
IN74HC86A
IN74HCT86A

Отечественный аналог
1554ЛП5
1594ЛП5
1564ЛП5
5564ЛП5</td>
</tr>
</table>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нефедов, А. В. Интегральные схемы и их зарубежные аналоги: справочник. В 12 т. Т. 10. – А. В. Нефедов. – М. : РадиоСофт, 2001. – 544 с.

2. Цифровые интегральные микросхемы: справочник / М. И. Богданович [и др.]. – Минск : Полымя, 1996. – 523 с.

3. Браммер, Ю. А. Цифровые устройства : учеб. пособие для вузов / Ю. А. Браммер, И. Н. Пашук. – М. : Высш. шк., 2004. – 229 с.

4. Угрюмов, Е. П. Цифровая схемотехника : учеб. пособие для вузов / Е. П. Угрюмов. – СПб. : БХВ-Петербург, 2004. – 528 с.

5. Jain, R. P. Modern Digital Electronics / R. P. Jain. – Tata McGran-Hill. – New Delhi, 1997. – 500 p.

6. Ronald, J. Tocci. Digital Systems, Principles and Applications / J. Ronald. – Prentice-Hall of India. – New Delhi, 1998. – 627 p.

7. Digital Logic Circuit Analysis and Design / P. Victor [et al.]. – Prentice Hall, Inc. – New Jersey, 1995. – 842 p.

8. Morris Mano, M. Digital Logic and Computer Design / M. Morris Mano. – Prentice-Hall of India. – New Delhi, 1998. – 612 p.

9. Charles, H. Fundamentals of Logic Design / H. Charles, Jr. Roth. – Jaico Publishing House. – Delhi, 1999. – 770 p.

10. Павлов, В. Н. Схемотехника аналоговых электронных устройств : учебник для вузов / В. Н. Павлов, В. Н. Ногин. – 3-е изд., испр. – М. : Горячая линия – Телеком, 2005. – 320 с.

11. Опадчий, Ю. Ф. Аналоговая и цифровая электроника : учебник для вузов / Ю. Ф. Опадчий, О. П. Глудкин, А. И. Гуров ; под ред. О. П. Глудкина. – М. : Горячая линия – Телеком, 2003, 2005. – 768 с.

12. Гусев, В. Г. Электроника и микропроцессорная техника : учебник для вузов / В. Г. Гусев. – М. : Высш. шк., 2004. – 790 с.

13. Попов, Э. Г. Основы аналоговой техники : учеб.-метод. пособие для студ. радиотехн. спец. / Э. Г. Попов. – Минск : БГУИР, 2006. – 276 с.

14. Безуглов, Д. А. Цифровые устройства и микропроцессоры : учеб. пособие для вузов / Д. А. Безуглов, И. В. Калиенко. – Ростов н/Д : Феникс, 2008. – 468 с.

15. Будько, А. А. Схемотехника аналоговых и цифровых устройств. Лабораторный практикум : учеб.-метод. пособие / А. А. Будько, Т. Н. Дворникова. – Минск : БГУИР, 2013. – 153 с.

16. Будько, А. А. Цифровые устройства. Технология IDL : учеб.-метод. пособие / А. А. Будько. – Минск : БГУИР, 2011. – 135 с.

17. Левкович, В. Н. Микропроцессорные устройства : учеб. пособие для студентов радиотехн. спец. всех форм обуч. / В. Н. Левкович, О. В. Шабров. – Минск : БГУИР, 2007 – 99 с.

18. Цифровые устройства. Лабораторный практикум : учеб.-метод. пособие / Р. Г. Ходасевич [и др.]. – Минск : БГУИР, 2010. – 112 с.

19. Новожилов, О. П. Основы цифровой техники / О. П. Новожилов. – М. : РадиоСофт, 2004. – 528 с.

20. Нефедов, В. И. Основы радиоэлектроники и связи: учебник для вузов / В. И. Нефедов. – 2-е изд. – М. : Высш. шк., 2002. – 510 с.

21. Уэйкерли, Дж. Проектирование цифровых устройств. В 2 т. / Дж. Уэйкерли ; пер. с англ. – М. : Постмаркет, 2002. – 1072 с.

22. Точчи, Р. Д. Цифровые системы. Теория и практика / Р. Д. Точчи, Н. С. Уидмер ; пер. с англ. – 8-е изд. – М. : Изд. дом «Вильямс», 2004. – 1024 с.

23. Загидуллин, Р. Ш. Multisim, LabVIEW. Практика автоматизированного проектирования электронных устройств / Р. Ш. Загидуллин. – М. : Горячая линия – Телеком, 2009. – 336 с.

24. Калабеков, Б. А. Цифровые устройства и микропроцессорные системы : учебник для ссузов / Б. А. Калабеков. – М. : Горячая линия – Телеком, 2002. – 336 с.

25. Браммер, Ю. А. Цифровые устройства : учеб. пособие для вузов / Ю. А. Браммер, И. Н. Пащук. – М. : Высш. шк., 2004. – 229 с.

26. Хернитер, М. Е. Multisim. Современная система компьютерного моделирования и анализа схем электронных устройств / М. Е. Хернитер ; пер. с англ. – М. : Изд. дом «ДМК-пресс», 2006. – 488 с.

27. Максфилд, К. Проектирование на ПЛИС. Курс молодого бойца / К. Максфилд. – М. : Изд. дом «Додэка-XXI», 2007. – 408 с.

28. Бибило, П. Н. Основы языка VHDL / П. Н. Бибило. – 3-е изд., доп. – М. : Изд. ЛКИ, 2007. – 328 с.

29. Сперанский, В. С. Сигнальные микропроцессоры и их применение в системах телекоммуникаций и электроники : учеб. пособие для вузов / В. С. Сперанский. – М. : Горячая линия – Телеком, 2008. – 168 с.

Св. план 2020, поз. 46

Учебное издание

Будько Анатолий Антонович
Дворникова Татьяна Николаевна

ФУНКЦИОНАЛЬНЫЕ УСТРОЙСТВА РАДИОСИСТЕМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *Е. И. Костина, Е. С. Юрец*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *М. В. Касабуцкий*

Подписано в печать 26.10.2020. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 12,9. Уч.-изд. л. 13,7. Тираж 100 экз. Заказ 46.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск