Е. В. Тимощенко

АРХИТЕКТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

ЧАСТЬ 1

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА

Могилев 2020

Деривативное электронное издание
на основе печатного издания:

Е. В. Тимощенко
Архитектура и программное обеспечение вычислительных систем
Ч. 1 : Арифметические и логические основы компьютера
Могилев : МГУ имени А. А. Кулешова, 2020. – 104 с. : ил.

ISBN 978-985-568-737-6

Лабораторный практикум предназначен для организации процесса обучения дисциплине «Архитектура и программное обеспечение вычислительных систем» у студентов специальностей 1-02 05 01 «Математика и информатика» и 1-02 05 02 «Физика и информатика». Он также может быть использован для дополнительного или самостоятельного образования по ряду дисциплин, которые содержат разделы, касающиеся основ информатики, а также арифметических, логических и цифровых основ вычислительной техники.

УДК 004.4(075.8)
ББК 32.97я73

Тимощенко, Е. В. Архитектура и программное обеспечение вычислительных систем [Электронный ресурс] : лабораторный практикум / Е. В. Тимощенко. – Электрон. данные. – Могилев : МГУ имени А. А. Кулешова, 2020. – Загл. с экрана.

212022, г. Могилев
ул. Космонавтов, 1
тел.: 8-0222-28-31-51
e-mail: alexpzn@mail.ru
http://www.msu.by

ВВЕДЕНИЕ

Актуальность учебной дисциплины «Архитектура и программное обеспечение вычислительных систем» обусловлена стремительным развитием и значительной ролью вычислительной техники в образовательном процессе, а также профессиональной направленностью на подготовку будущего преподавателя информатики.

Хорошо известна роль вычислительной техники также в других сферах человеческой деятельности. Большой популярностью пользуются персональные компьютеры, обладающие высокой производительностью и не требующие от пользователя глубокого знания процессов, происходящих в компьютере во время вычислений и выполнения операций. Без преувеличения можно сказать, что появление ПК в середине 70-х годов прошлого века и бурное их распространение в наше время открыло новую эру в массовом использовании вычислительной техники людьми всех рангов и профессий. Человечество стремительно перешло от экономики, основанной на тяжёлой промышленности, к цифровой экономике с применением современных информационно-коммуникационных технологий, цифровым средствами связи и услугам.

Целью учебной дисциплины «Архитектура и программное обеспечение вычислительных систем» является формирование у будущих преподавателей физики, математики и информатики профессиональных компетенций в области физических основ построения и функционирования компьютера, аппаратного и программного обеспечения вычислительных систем. При изучении данной дисциплины студенты приобретут знания о способах и формах представления информации в вычислительных системах, об архитектуре и принципах работы ЭВМ и её основных логических блоков, организации вычислительных систем. Поэтому умения и навыки, полученные при выполнении предложенных лабораторных работ, станут важным звеном в профессиональной подготовке, неотъемлемым компонентом их будущей профессиональной деятельности.

Лабораторный практикум предназначен для организации процесса обучения дисциплине «Архитектура и программное обеспечение вычислительных систем» у студентов специальностей 1-02 05 01 «Математика и информатика» и 1-02 05 02 «Физика и информатика». Он состоит из 10 лабораторных работ, в которых присутствуют обязательный раздел с

краткими теоретическими сведениями по теме работы, примеры решения типовых задач, порядок выполнения лабораторной работы, перечень заданий для индивидуального выполнения студентами и контрольные вопросы для проверки усвоения материала. Также студентам предложен список рекомендуемой литературы в качестве дополнительного источника информации по темам изучаемой дисциплины.

Предложенный лабораторный практикум предназначен для закрепления знаний и приобретения необходимых навыков по следующим темам: «Системы счисления», «Представление информации в компьютере», «Арифметические операции в компьютере», «Булева алгебра», «Минимизация логических выражений», «Синтез логических схем», «Исследование интегральных элементов и узлов компьютера», что составляет содержание учебного материала из разделов «Архитектура компьютера» и «Физические основы функционирования компьютера» учебной программы дисциплины «Архитектура и программное обеспечение вычислительных систем» у студентов специальностей 1-02 05 01 «Математика и информатика» и 1-02 05 02 «Физика и информатика».

Лабораторный практикум также может быть использован для дополнительного или самостоятельного образования по ряду дисциплин, которые содержат разделы, касающиеся основ информатики, а также арифметических, логических и цифровых основ вычислительной техники. Практикум позволяет обеспечить формирование у студентов академических, социально-личностных и профессиональных компетенций в области физических, логических и цифровых основ построения и функционирования современных вычислительных систем.

ЛАБОРАТОРНАЯ РАБОТА № 1
ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В РАЗНЫХ СИСТЕМАХ СЧИСЛЕНИЯ

Цель работы: Закрепить знания студентов о способах представления чисел в позиционных системах счисления, приобрести навык перевода чисел из одной системы счисления в любую другую и обратно.

Краткие теоретические сведения

1. *Понятие и виды систем счисления*

Под **системой счисления (СС)** понимается способ представления любого числа посредством алфавита символов, называемых цифрами. По количеству символов, используемых для записи числа, системы счисления подразделяются на **позиционные** и **непозиционные**.

Если для записи числа используется бесконечное множество символов и значение цифры не зависит от ее положения в ряду цифр, изображающих число, то система счисления называется **непозиционной**. Примером непозиционной системы счисления может служить римская система счисления. Цифры в римской системе обозначаются различными знаками: 1 – I; 3 – III; 5 – V; 10 – X; 50 – L; 100 – C; 500 – D; 1000 – M.

Позиционные системы счисления (ПСС) для записи чисел используют ограниченный набор символов, называемых цифрами, и величина числа зависит не только от набора цифр, но и от того, в какой последовательности записаны цифры, т.е. от позиции, занимаемой цифрой, например, 125 и 215.

Количество цифр, используемых для записи числа, называется **основанием системы счисления**, в дальнейшем его обозначим **q**.

В повседневной жизни мы пользуемся десятичной позиционной системой счисления, где $q = 10$, т.е. используется 10 цифр: 012345689.

В электронных вычислительных системах применяют позиционные системы счисления с недесятичным основанием: двоичную, восьмеричную, шестнадцатеричную и др. В двоичной системе счисления используются две цифры: 0 и 1; восьмеричная система счисления имеет восемь цифр: 01234567, шестнадцатеричная – шестнадцать, причем первые 10 цифр совпадают по написанию с цифрами десятичной системы счисле-

ния, а для обозначения оставшихся шести цифр применяются большие латинские буквы A= 10, B=11, C=12, D=13, E=14, F=15, т.е. для шестнадцатеричной системы счисления получим набор цифр: 0123456789ABCDEF.

Для обозначения используемой системы счисления числа заключают в скобки и индексом указывают её основание:

$(15)_{10}$; $(1011)_2$; $(735)_8$;

Иногда скобки опускают и оставляют только индекс:

15_{10}; 1011_2;735_8.

Двоично-десятичная система счисления получила большое распространение в современных компьютерах ввиду легкости перевода в десятичную систему и обратно. Она используется там, где основное внимание уделяется не простоте технического построения машины, а удобству работы пользователя. В этой системе счисления все десятичные цифры отдельно кодируются четырьмя двоичными цифрами и в таком виде записываются последовательно друг за другом.

Для примера, сравнение записей чисел в десятичной, двоичной, восьмеричной, шестнадцатеричной и двоично-десятичной системах счисления приведено в таблице 1.

Таблица 1

Системы счисления

Десятичная запись	Двоичная запись	Восьмеричная запись	Шестнадцатеричная запись	Двоично-десятичная запись
0	0000	0	0	0000-0000
1	0001	1	1	0000-0001
2	0010	2	2	0000-0010
3	0011	3	3	0000-0011
4	0100	4	4	0000-0100
5	0101	5	5	0000-0101
6	0110	6	6	0000-0110
7	0111	7	7	0000-0111
8	1000	10	8	0000-1000
9	1001	11	9	0000-1001
10	1010	12	A	0001-0000
11	1011	13	B	0001-0001
12	1100	14	C	0001-0010

Окончание таблицы

Десятичная запись	Двоичная запись	Восьмеричная запись	Шестнадцате-ричная запись	Двоично-десятичная запись
13	1101	15	D	0001-0011
14	1110	16	E	0001-0100
15	1111	17	F	0001-0101
16	1 0000	20	10	0001-0110
17	1 0001	21	11	0001-0111
18	1 0010	22	12	0001-1000

Двоичная ПСС получила самое широкое применение в компьютере благодаря следующим достоинствам:

1. Числовая информация в компьютере отождествляется с состоянием используемых двоичных физических элементов – триггеров, регистров, счетчиков и т.п. Реализация элементов, которые должны различать одно из двух состояний (0 или 1), оказывается проще и надежнее, чем реализация элементов, которые должны различать одно из 10 состояний.

2. Арифметические операции выполняются наиболее просто. В то же время громоздкость записи чисел в двоичной ПСС и трудность их восприятия человеком приводит к необходимости перевода исходных данных (чисел) из десятичной системы счисления в двоичную, а результатов – из двоичной в десятичную.

2. *Единая запись чисел в различных позиционных системах счисления*

Число в позиционной системе счисления с основанием q может быть представлено в виде полинома по степеням q.

Например, в десятичной системе мы имеем число

$123{,}45_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$.

В двоичной:

$0110{,}101_2 = 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$,

а в общем виде это правило запишется так:

$$X_{(q)} = x_{n-1} \times q^{n-1} + x_{n-2} \times q^{n-2} + \ldots + x_1 \times q^1 + x_0 \times q^0 + x_{-1} \times q^{-1} + \ldots + x_{-m} \times q^{-m} . \quad (1)$$

Здесь $X_{(q)}$ – запись числа в системе счисления с основанием q;

x_i – натуральные числа меньше q, т.е. цифры, где $i \in (-m, n)$;

n – число разрядов целой части;

m – число разрядов дробной части.

Записывая слева направо цифры числа, мы получим закодированную запись числа в *q*-ичной системе счисления:

$$X_{(q)} = x_{n-1}\, x_{n-2}\, ..\, x_1\, x_0\, ,\, x_{-1}\, ..\, x_{-m}\, .$$

Запятая отделяет целую часть числа от дробной части. В вычислительной технике (ВТ) чаще всего для отделения целой части числа от дробной части используют точку. Позиции цифр, отсчитываемые от точки, называют разрядами.

В позиционной системе счисления вес каждого разряда отличается от веса соседнего разряда в число раз, равное основанию системы счисления. Так, например, в десятичной системе счисления цифры 1-го разряда – единицы, 2-го – десятки, 3-го – сотни и т. д.

3. *Правила перевода чисел из одной системы счисления в другую*

Для перевода числа в любую систему счисления нужно придерживаться определенной последовательности действий.

Для перевода из любой системы счисления в десятеричную используют выражение (1).

Пример 1:

$1010{,}1_2 = 1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0+1\cdot 2^{-1} = 73_{10}$;

$1AD{,}B_{16} = 1\cdot 16^2+10\cdot 16^1+13\cdot 16^0+11\cdot 16^{-1} = 429{,}11_{10}$.

Обратный перевод из десятичной системы счисления в систему счисления с другим основанием непосредственно по (1) затруднителен, поскольку все арифметические действия, предусмотренные этой формулой, следует выполнять в той системе счисления, в которую число переводится. Целые числа и дроби переводят разными способами.

Перевод целых чисел

Исходное число последовательно делят на основание системы записывая при этом остатки от деления. Результат – это полученные остатки, записанные в обратном порядке.

Пример 2. Перевести число 47_{10} в двоичную и шестнадцатеричную систему счисления.

47:2 = 23 (остаток 1).

23:2= 11 (остаток 1).

11:2 = 5 (остаток 1).

5:2 = 2 (остаток 1).

2:2=1 (остаток 0).

1:2 = 0 (остаток 1).

В результате получаем $47_{10} = 101111_2$.

47:16 = 2 (остаток 15).

2:16= 0 (остаток 2).

Учитывая, что $15_{10} = F_{16}$, в результате получаем $47_{10} = 2F_{16}$.

Перевод правильных дробей

Правильная дробь имеет нулевую целую часть, т.е. у нее числитель меньше знаменателя. Результат перевода правильной дроби всегда правильная дробь. Дробную часть числа последовательно умножают на основание системы счисления, в которую переводится, до тех пор, пока она не превратится в 0 или пока не появится период.

Пример 3. Перевести число $0{,}847_{10}$ в двоичную и шестнадцатеричную систему счисления до четырех значащих цифр.

Для перевода в *двоичную* систему счисления дробную часть числа последовательно умножаем на основание 2:

0	847
1	694
1	388
0	776
1	552

и т.д.

В результате получаем $0{,}847_{10} = 0{,}1101_2$.

Для перевода в *шестнадцатеричную* систему счисления дробную часть числа последовательно умножаем на основание 16:

0	847
13	552
8	832
13	312
4	992

и т.д.

В итоге $0{,}847_{10}=0{,}D8D4_{16}$.

Перевод неправильных дробей

Неправильная дробь имеет ненулевую дробную часть, т.е. у нее числитель больше знаменателя. Результат перевода неправильной дроби всегда неправильная дробь. При переводе смешанного числа следует переводить его целую и дробную части отдельно, а затем сложить полученные результаты.

Пример 4. Перевести неправильную дробь из десятичной в двоичную систему счисления на примере числа 46,625.

1. Переводим целую часть числа:

46:2 = 23 (остаток 0).

23:2= 11 (остаток 1).

11:2 = 5 (остаток 1).

5:2 = 2 (остаток 1).

2:2=1 (остаток 0).

1:2 = 0 (остаток 1).

Записываем остатки последовательно справа налево – 101110, т.е. $46_{10}=101110_2$.

2. Переводим дробную часть числа:

0,625×2=1,250.

0,250×2=0,500.

0,500×2=1,000 (дробная часть равна 0 => стоп).

Записываем целые части получающихся произведений после запятой последовательно слева направо – 0,101, т. е. $0{,}625_{10} = 0{,}101_2$.

3. Суммируем результаты предыдущих действий. Окончательно: $46{,}625_{10}= 101110{,}101_2$.

4. *Арифметические операции в позиционных системах счисления*

Арифметические операции в рассматриваемых позиционных системах счисления выполняются по законам, известным из десятичной арифметики. Двоичная система счисления имеет основание 2, и для записи чисел используются всего две цифры 0 и 1 в отличие от десяти цифр десятичной системы счисления.

Рассмотрим сложение одноразрядных чисел: 0+0=0, 0+1=1, 1+0=0. Эти равенства справедливы как для двоичной системы, так и для десятичной системы. В двоичной системе старшей цифрой является 1. Следовательно, в двоичной системе 1+1=10, так как при сложении двух единиц происходит переполнение разряда и производится перенос в старший разряд. Переполнение разряда наступает тогда, когда значение числа в нем становится равным или большим основания. Для двоичной системы это число равно 2. Продолжая добавлять единицы, заметим: $10_2+1=11_2$, $11_2+1=100_2$ – произошла "цепная реакция", когда перенос единицы в один разряд вызывает перенос в следующий разряд.

Пример 5. Выполнить сложение в двоичной системе счисления.

$$\begin{array}{r} +110_2 \\ 11_2 \\ \hline 1001_2 \end{array}$$

Сложение многоразрядных чисел в системах счисления с любым основанием происходит по этим же правилам с учетом возможности переносов из младших разрядов в старшие.

Вычитание многоразрядных двоичных чисел производится с учетом возможных заèмов из старших разрядов. Действия *умножения* и *деления* чисел в двоичной арифметике можно выполнять по общепринятым для позиционных систем правилам.

Пример 6. Выполнить вычитание в двоичной системе счисления.

$$\begin{array}{r} -110_2 \\ 11_2 \\ \hline 11_2 \end{array}$$

Пример 7. Выполнить умножение и деление в двоичной системе счисления.

Выполним умножение в столбик.

$$\begin{array}{r} \times\ 110_2 \\ 11_2 \\ \hline +\ 110_2 \\ 110_2 \\ \hline 10010_2 \end{array}$$

Теперь выполним деление традиционным уголком.

$$\begin{array}{r|l} -\ 110_2 & 11_2 \\ \cline{2-2} 11_2 & 10_2 \\ \cline{1-1} 0 & \end{array}$$

Приведем ещё несколько примеров арифметических операций над числами в различных системах счисления.

Пример 8. Сложить числа:

а) $10000000100_2 + 111000010_2 = 10111000110_2$;

б) $3B3{,}6_{16} + 38B{,}4_{16} = 73E{,}A_{16}$.

$$\begin{array}{r} 10000000100 \\ +\ 111000010 \\ \hline 10111000110 \end{array} \qquad \begin{array}{r} 3B3{,}6 \\ +\ 38B{,}4 \\ \hline 73E{,}A \end{array}$$

Пример 9. Выполнить умножение:

а) $100111_2 \times 1000111_2 = 101011010001_2$;

б) $61{,}A_{16} \times 40{,}D_{16} = 18B7{,}52_{16}$.

$$\begin{array}{r} 100111 \\ \times\ 1000111 \\ \hline 100111 \\ +\ 100111 \\ 100111 \\ 100111 \\ \hline 101011010001 \end{array} \qquad \begin{array}{r} 61{,}A \\ \times\ 40{,}D \\ \hline 4F52 \\ +\ 1868 \\ \hline 18B7{,}52 \end{array}$$

Задания к лабораторной работе

В соответствии с выданным преподавателем вариантом (см. таблицу 2) выполните следующие задания.

1. Переведите числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы счисления. В п. 1 б) получите пять знаков после запятой в двоичном представлении.

2. Переведите числа в десятичную систему счисления.

3. Выполните сложение и вычитание чисел (переставлять местами числа при вычитании нельзя). В п. 3 в) для представления слагаемых в одинаковой системе счисления используйте метод определенного соотношения между основаниями систем счисления.

4. Выполните умножение и деление чисел в заданных системах счисления.

Примечание: В заданиях 3 - 4 проверьте правильность вычислений переводом исходных данных и результатов в десятичную систему счисления.

Таблица 2

Вариант 1		**Вариант 2**	
1	149,375; 228,79	1	213,125; 261,78.
2	$110101101{,}0011_2$; $274{,}32_8$; $1A9{,}7D_{16}$.	2	$10011100{,}1101_2$; $273{,}61_8$; $512{,}3F_{16}$
3	110100,01 ± 10101101,011; 3CA,8 ± 65,2C; 1011001,11 ± 2F,31.	3	1011011,101 ± 11001100,01; 22D,1 ± 123,8; 1110001,011 ± E2,5.
4	1011001,101 и 1011,01; 6A,4 и B,12.	4	11001,11 и 1010,011; 2B,A1 и 36,9.
Вариант 3		**Вариант 4**	
1	129,375; 238,79	1	193,125; 161,78.
2	$11001101{,}0111_2$; $174{,}35_8$; $149{,}7D_{16}$.	2	$1011100{,}1001_2$; $573{,}21_8$; $C52{,}3F_{16}$
3	110110,01 ± 1011101,011; 3C9,8 ± 1A5,2C; 101001,101 ± 2F,31.	3	1001011,101 ± 1001100,01; 29D,1 ± 12A,8; 110001,011 ± E2,5C.
4	1011011,101 и 1011,01; 6F,4 и B,1A.	4	110110,11 и 1110,011; 2C,A1 и 3D,9.

Окончание таблицы 2

Вариант 5		**Вариант 6**	
1	143,375; 218,79	1	133,125; 161,58.
2	$110001101{,}0011_2$; $274{,}33_8$; $1F9{,}7D_{16}$.	2	$10111100{,}1101_2$; $247{,}31_8$; $A12{,}3F_{16}$
3	10100,01 ± 100101,001; 3C2,8 ± 61,2C; 101101,11 ± 2D,37.	3	10011,101 ± 111100,01; 29D,1 ± 1F3,8; 11001,011 ± E2,54.
4	101001,111 и 1011,01; 6C,4 и B,72.	4	110101,11 и 1010,001; 2D,A1 и C6,9.
Вариант 7		**Вариант 8**	
1	243,375; 118,69	1	213,125; 168,68.
2	$10101101{,}0111_2$; $274{,}36_8$; $1A9{,}77_{16}$.	2	$10100100{,}1101_2$; $257{,}62_8$; $A18{,}3F_{16}$
3	10100,11 ± 101101,101; 3D2,8 ± 52,2C; 101001,11 ± 2C,37.	3	10111,111 ± 111101,01; 25D,1 ± 1F4,8; 11001,111 ± E7,54.
4	10101,111 и 1001,01; 6E,4 и D,72.	4	111101,11 и 1010,101; 2D,91 и C1,9.
Вариант 9		**Вариант 10**	
1	233,375; 190,26	1	183,125; 197,28.
2	$1100101{,}0011_2$; $274{,}37_8$; $1F5{,}2D_{16}$.	2	$1001100{,}1011_2$; $256{,}24_8$; $A22{,}8F_{16}$
3	10101,11 ± 101101,101; 8C2,4 ± 31,2C; 101101,101 ± 5A,17.	3	10001,101 ± 110100,01; 28D,1 ± 1F3,F; 101001,011 ± F2,54.
4	101101,011 и 1001,01; 6F,4 и B,92.	4	100101,11 и 1011,101; 8D,A1 и C6,D.

Контрольные вопросы:

1. Что называется системой счисления?

2. На какие два типа можно разделить все системы счисления? Приведите примеры.

3. Какие системы счисления применяются в вычислительной технике? Ответ поясните.

4. Охарактеризуйте двоичную, шестнадцатеричную, двоично-десятичную системы счисления: алфавит, основание, запись числа. Приведите примеры.

5. Сформулируйте правила перевода чисел из системы счисления с основанием p в десятичную систему счисления. Приведите примеры.

6. Сформулируйте правила обратного перевода: из десятичной системы счисления в систему счисления с основанием p. Приведите примеры.

7. Чему равны веса разрядов слева от точки, разделяющей целую и дробную части, в двоичной системе счисления (восьмеричной, шестнадцатеричной)? Приведите примеры.

8. Чему равны веса разрядов справа от точки, разделяющей целую и дробную части, в двоичной системе счисления (восьмеричной, шестнадцатеричной)? Приведите примеры.

9. Сформулируйте правила выполнения арифметических операций над числами в позиционных системах счисления. Приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА № 2
ПРЕДСТАВЛЕНИЕ ЧИСЛОВОЙ ИНФОРМАЦИИ В КОМПЬЮТЕРЕ

Цель работы: Закрепить знания о машинных кодах, приобрести навык представления чисел в прямом, обратном и дополнительном кодах. Изучить основы машинной арифметики.

Краткие теоретические сведения

1. *Представление целых чисел. Машинные коды*

Целые числа в компьютере хранятся в формате с **фиксированной запятой (точкой)**. Для записи числа с фиксированной точкой машинное слово (операнд) делится на два фиксированных поля (части) – поле знака и поле цифр.

Существует несколько соглашений о едином формате представления как положительных, так и отрицательных чисел.

Для алгебраического представления чисел, т. е. для представления чисел с учетом их знака, в компьютерах используются специальные коды, которые называются **машинными кодами.** Их использование позволяет обрабатывать знаковые разряды чисел так же, как и значащие разряды, а также заменять операцию вычитания операцией сложения.

Различают **прямой код (П)**, **обратный код (ОК)** и **дополнительный код (ДК)** двоичных чисел.

Прямой код двоичного числа образуется из абсолютного значения этого числа и кода знака (нуль или единица) перед его старшим числовым разрядом.

Пример 1.

A_{10}= +10 A_2=+1010 $[A_2]_П$=0:1010;

B_{10}= –15 B_2=-1111 $[B_2]_П$=1:1111.

Точечной вертикальной линией здесь отмечена условная граница, отделяющая знаковый разряд от значащих разрядов числа.

Обратный код двоичного числа образуется по следующему правилу. Обратный код положительных чисел совпадает с их прямым кодом. Обратный код отрицательного числа содержит единицу в знаковом разряде

числа, а значащие разряды числа заменяются на инверсные, то есть нули заменяются единицами, а единицы – нулями.

Пример 2.

A_{10}=+5 A_2=+101 $[A_2]_П$=$[A_2]_{ОК}$=0:101;

B_{10}= –13 B_2= –1101 $[B_2]_{ОК}$=1:0010.

Свое название обратный код чисел получил потому, что коды цифр отрицательного числа заменены на инверсные. Укажем наиболее важные *свойства обратного кода* чисел:

• сложение положительного числа *С* с его отрицательным значением в обратном коде дает так называемую машинную единицу МЕок= 1: 111... 11, состоящую из единиц в знаковом и значащих разрядах числа;

• нуль в обратном коде имеет двоякое значение. Он может быть как положительным – 0: 00...0, так и отрицательным числом – 1: 11... 11. Значение отрицательного нуля совпадает с МЕок.

Двойственное представление нуля явилось причиной того, что в современных ЭВМ все числа представляются не обратным, а **дополнительным кодом**. Дополнительный код положительных чисел совпадает с их прямым кодом. Дополнительный код отрицательного числа представляет собой результат суммирования обратного кода числа с единицей младшего разряда (2^0; - для целых чисел, 2^{-k} - для дробных).

Пример 3.

A_{10}=+19 =+10011 $[A_2]_П$=$[A_2]_{ОК}$=$[A_2]_{ДК}$=0:10011;

B_{10}= –13 = –1101 $[B_2]_{ДК}$=$[B_2]_{ОК}$ + 2^0=1:0010+1=1:0011.

Укажем основные *свойства дополнительного кода*:

• сложение дополнительных кодов положительного числа *С* с его отрицательным значением дает так называемую машинную единицу дополнительного кода:

$$МЕ_{ДК}=МЕ_{ОК} + 2^0=10: 00…00,$$

т.е. число 10_2 (два) в знаковых разрядах числа;

• дополнительный код получил такое свое название потому, что представление отрицательных чисел является дополнением прямого кода чисел до машинной единицы МЕдк.

Модифицированные обратные и *дополнительные коды* двоичных чисел отличаются соответственно от обратных и дополнительных кодов удвоением значений знаковых разрядов. Знак “+” в этих кодах кодируется двумя нулевыми знаковыми разрядами, а “–” – двумя единичными разрядами.

Пример 4.
A_{10}=9 A_2=+1001 $[A_2]_{П}=[A_2]_{ОК}=[A_2]_{ДК}$=0:1001
$[A_2]_{МОК}=[A_2]_{МДК}$=00:1001;
B_{10}= –9 B_2=–1001 $[B_2]_{ОК}$=1:0110 $[B_2]_{ДК}$=1:0111
$[B_2]_{МОК}$=11:0110 $[B_2]_{МДК}$=11:0111.

Целью введения модифицированных кодов являются фиксация и обнаружение случаев получения неправильного результата, когда значение результата превышает максимально возможный результат в отведенной разрядной сетке машины. В этом случае перенос из значащего разряда может исказить значение младшего знакового разряда. Значение знаковых разрядов “01” свидетельствует о положительном переполнении разрядной сетки, а “10” – об отрицательном переполнении. В настоящее время практически во всех моделях компьютеров роль удвоенных разрядов для фиксации переполнения разрядной сетки играют переносы, идущие в знаковый и из знакового разряда.

2. *Представление вещественных чисел*

Вещественными числами (в отличие от целых) в компьютерной технике называются числа, имеющие дробную часть. При их изображении вместо запятой принято ставить точку. Так, например, число 5 – целое, а числа 5.1 и 5.0 – вещественные.

Для представления вещественных чисел в современных компьютерах принят способ представления с плавающей запятой.

В **форме представления с плавающей запятой (точкой)** число изображается в виде двух групп цифр:

- мантисса;
- порядок.

При этом абсолютная величина мантиссы должна быть меньше 1, а порядок – целым числом.

Число в форме с плавающей точкой может быть представлено в следующем виде:

$$N = \pm M \times P^{\pm r},$$

где *M* – мантисса числа; *p* – основание СС; *r* – порядок числа, который показывает, на сколько разрядов и в какую сторону сдвинута запятая при замене формы записи числа.

Число может иметь множество представлений в форме с плавающей запятой (например, число 12 может быть записано в видах: $0{,}12\times10^2$, $0{,}012\times10^3$, $0{,}0012\times10^4$ и т. д.).

Из приведенного выше примера видно, что благодаря изменению порядка точка перемещается по мантиссе. При этом, если порядок положительный, точка смещается по мантиссе ВЛЕВО, а если отрицательный – ВПРАВО.

Все числа с плавающей запятой хранятся в машине в нормализованном виде.

Нормализованное число – такое число, старший разряд мантиссы которого больше нуля, то есть первый разряд мантиссы после запятой не может равняться нулю.

Пример 5.

$-348 = -0{,}348\times10^5$ $\quad$ $+10427 = +0{,}10427\times10^5$

$+721{,}355 = +0{,}721355\times10^3$ $\quad$ $+0{,}00058 = +0{,}58\times10^{-3}$

3. *Арифметические операции над числами с фиксированной точкой*

Сложение (вычитание)

Операция вычитания приводится к операции сложения путем преобразования чисел в обратный или дополнительный код. Знаковые разряды участвуют в операции сложения наравне с цифровыми. Пусть числа $A \geq O$ и $B \geq O$, тогда операция алгебраического сложения выполняется в соответствии с таблицей 1.

Таблица 1

Таблица преобразования кодов при алгебраическом сложении

Требуемая операция	Необходимое преобразование
A+B	A+B
A-B	A+(-B)
-A+B	(-A)+B
-A-B	(-A)+(-B)

Скобки в представленных выражениях указывают на замену операции вычитания операцией сложения с обратным или дополнительным кодом соответствующего числа. Сложение двоичных чисел осуществляется последовательно и поразрядно. При выполнении сложения цифр необходимо соблюдать следующие правила.

1. Слагаемые должны иметь одинаковое число разрядов. Для выравнивания разрядной сетки слагаемых можно дописывать незначащие нули слева к целой части числа и незначащие нули справа к дробной части числа.

2. Знаковые разряды чисел участвуют в сложении так же, как и значащие.

3. Необходимые преобразования машинных кодов производятся с изменением знаков чисел. Приписанные незначащие нули изменяют свое значение при преобразованиях по общему правилу.

4. При образовании единицы переноса из старшего знакового разряда, в случае использования ОК, эта единица складывается с младшим числовым разрядом. При использовании ДК единица переноса теряется. Знак результата формируется автоматически, результат представляется в том коде, в котором представлены исходные слагаемые.

5. Если результат положителен – он представлен в прямом коде и не требует никаких преобразований. Если результат отрицателен, то он представлен в обратном или дополнительном коде в зависимости от того, в каком коде происходило сложение. Результат в таком случае преобразуется в прямой код.

Пример 6. Сложить в обратном коде числа –34 и +15. Разрядная сетка равна 8 бит.

Преобразуем слагаемые в прямые коды

$-34 = -100010_2$

$+15 = +1111_2$

Преобразуем отрицательное слагаемое -34 в обратный код

1 1 0 1 1 1 0 1

Разместим полученные представления в разрядных сетках и совершим операцию сложения слагаемых:

$$\begin{array}{r} +\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1 \\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \end{array}$$

Единица переноса не образована. Судя по знаку, результат отрицателен, значит, представлен в обратном коде (поскольку сложение выполнялось в этом коде) и требует перевода в прямой код:

1 0 0 1 0 0 1 1

Таким образом, получено число -10011_2.

Для проверки правильности результата представим его в десятичной системе счисления. Имеем: $-10011_2 = -19_{10}$, что соответствует правильному результату.

Пример 7. Сложить в обратном коде числа -34 и -15. Разрядная сетка равна 8 бит.

Преобразуем слагаемые в прямые коды и при сложении разместим их в разрядных сетках:

$-34 = -100010_2$

$-15 = -1111_2$

Преобразуем отрицательные слагаемые в обратный код:

для -34

1 1: 0 1 1 1 0 1

для -15

1 1: 1 1 0 0 0 0

Складываем слагаемые:

```
    +1 1 0 1 1 1 0 1
       1 1 1 1 0 0 0 0
  ---------------------
  1 ↵1 1 0 0 1 1 0 1
```

Образовалась единица переноса из знакового разряда. В соответствии с правилами сложения в обратном коде она прибавляется к младшему числовому разряду, получаем:

1 1: 0 0 1 1 1 0

Судя по знаку, результат отрицателен, значит, представлен в обратном коде (поскольку сложение выполнялось в этом коде) и требует перевода в прямой код:

1 0: 1 1 0 0 0 1

Таким образом, получено число -110001_2.

Для проверки правильности результата представим его в десятичной системе счисления.

Имеем: $-110001_2 = -49$, что соответствует правильному результату.

Пример 8. Сложить в дополнительном коде числа -13 и +8

Преобразуем слагаемые в прямые коды и разместим их в разрядных сетках: $-13 = -1101_2$

$+8 = +1000_2$

Преобразуем отрицательное слагаемое -13 в дополнительный код:

1: 0 0 1 1

Складываем слагаемые:

$$\begin{array}{r} {}_{+}1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 0\ 0 \\ \hline 1\ 1\ 0\ 1\ 1 \end{array}$$

В знаковом разряде стоит единица, значит, результат получен в дополнительном коде.

Для перехода к прямому коду необходимо выполнить следующее преобразование: От младшего разряда полученного дополнительного кода отнять единицу, получим

$[A_2]_{ОБ} = [A_2]_{ДП} - 1 = 1{:}1011 - 1 = 1{:}1010.$

$[A_2]_{П} \rightarrow [A_2]_{ОБ} = 1{:}1010 = -0101.$

Умножение

Умножение двоичных чисел наиболее просто реализуется в прямом коде. В итоге оно приводится к операциям сложения и сдвигам.

Пример 9. Умножить два числа +7 и +5.

Перемножим эти числа, представленные прямыми двоичными кодами, так же, как это делается в десятичной системе.

После преобразования чисел в прямые коды получаем:

$7_{10} = +111_2 = 0{:}111;$

$5_{10} = +101_2 = 0{:}101.$

Умножим столбиком:

```
   ×111
    101
 -------
   + 111
  + 000
   111
 -------
  100011
```

В результате получаем $0{:}100011_2 = +100011_2 = 35_{10}$.

Нетрудно видеть, что произведение получается путём сложения частных произведений, представляющих собой разряды множимого, сдвинутые влево в соответствии с позициями разрядов множителя. Частные произведения, полученные умножением на нуль, игнорируются. Важной особенностью операции умножения n-разрядных сомножителей является увеличение разрядности произведения до n+n=2n. Знак произведения формируется путём сложения знаковых разрядов сомножителей. Возможные переносы из знакового разряда игнорируются.

Операция, которую реализует компьютер для определения знака произведения, называется **суммой по модулю два** и обозначается ⊕. Для нее справедливы равенства: 0⊕0=0, 0⊕1=1, 1⊕0=1, 1⊕1=0.

Деление

Операция деления, как и в десятичной арифметике, является обратной операции умножения. Покажем, что и эта операция приводится к последовательности операций сложения и сдвига.

Пример 10. Разделить два числа +45 и +5.

После преобразования чисел в прямые коды получаем

$+45_{10} = +101101_2 = 0{:}101101$

$+5_{10} = +101_2 = 0{:}101$

Делим уголком:

```
 _101101 | 101
  101    | 1010
 -------
    _0101
     101
 -------
       0
```

Деление произведено так же, как это делается обычно в десятичной системе. Сначала проверяется, можно ли вычесть значение делителя из старших разрядов делимого. Если возможно, то в разряде частного записывается единица и определяется частная разница. В противном случае в частное записывается нуль и разряды делителя сдвигаются вправо на один разряд по отношению к разрядам делимого. К полученной предыдущей разнице сносится очередная цифра делимого, и данный процесс повторяется, пока не будет получена необходимая точность.

Если учесть, что все вычитания в компьютере заменяются сложением в ОК или в ДК, то действительно операция деления приводится к операциям сложения и сдвигам вправо разрядов делителя относительно разрядов делимого. Отметим, что делимое перед операцией деления должно быть приведено к 2*n*-разрядной сетке. Только в этом случае при делении на *n*-разрядный делитель получается *n*-разрядное частное.

4. *Арифметические операции над числами с плавающей точкой.*

В современных компьютерах числа с плавающей точкой хранятся в памяти машин, имея мантиссу и порядок (характеристику) в прямом коде и нормализованном виде. Все арифметические действия над этими числами выполняются так же, как это делается с ними, если они представлены в полулогарифмической форме (мантисса и десятичный порядок) в десятичной системе счисления. Порядки и мантиссы обрабатываются раздельно.

Сложение (вычитание)

Операция сложения (вычитания) производится в следующей последовательности.

1. Сравниваются порядки (характеристики) исходных чисел путем их вычитания $\Delta p = p_1 - p_2$. При выполнении этой операции определяется, одинаковый ли порядок имеют исходные слагаемые.

2. Если разность порядков равна нулю, то это значит, что одноименные разряды мантисс имеют одинаковые веса (двоичный порядок). В противном случае должно проводиться выравнивание порядков.

3. Для выравнивания порядков число с меньшим порядком сдвигается вправо на разницу порядков Δp. Младшие выталкиваемые разряды при этом теряются.

4. После выравнивания порядков мантиссы чисел можно складывать (вычитать) в зависимости от требуемой операции. Операция вычитания заменяется операцией сложения в соответствии с данными таблицы 1. Действия над слагаемыми производятся в ОК или ДК по общим правилам.

5. Порядок результата берется равным большему порядку.

6. Если мантисса результата не нормализована, то осуществляются нормализация и коррекция значений порядка.

Пример 11. Сложить два числа A_{10}=+1.375; B_{10}=-0.625.

A_2=+1.011=0: 1011×10^1; B_2=-0.101=-1:101×10^0.

В нормализованном виде эти числа будут иметь вид:

Число	Порядок	Мантисса
$[A_2]_П$	0:1	0:1011
$[B_2]_П$	0:0	1:101

1. Вычитаем порядки $\Delta p = p_1 - p_2 = 1-0 = 1$. $\Delta p \neq 0$.

2. Порядок первого числа больше порядка второго числа на единицу. Требуется выравнивание порядков.

3. Для выравнивания порядков необходимо второе число сдвинуть вправо на один разряд.

$[B_2]_{ИСХ}$=0: 01: 101

после сдвига

$[B_2]_П$=0: 11:0101

$[m_{B'}]_{ДК}$= 1: 1011

4. Складываем мантиссы.

$$
\begin{array}{rl}
+[m_A]_{ДК} = & 0{:}1011 \\
[m_{B'}]_{ДК} = & 1{:}1011 \\
\hline
[m_C]_{ДК} = & 0{:}0110
\end{array}
$$

Мантисса числа С положительная.

5. Порядок числа С равен порядку числа с большим порядком, т.е. p = +1.

$[C_2]_П$=0: 10: 0110.

Видно, что мантисса результата не нормализована, так как старшая цифра мантиссы равна нулю.

6. Нормализуем результат путем сдвига мантиссы на один разряд влево и, соответственно, вычитаем из значения порядка единицу:

$[C_2]_П$=0: 00: 110.

C_{10}= +0,75.

Умножение (деление)

Операция умножения (деления) чисел с плавающей точкой также требует разных действий над порядками и мантиссами. Алгоритмы этих операций выполняются в следующей последовательности.

1. При умножении (делении) порядки складываются (вычитаются) так, как это делается над числами с фиксированной точкой.

2. При умножении (делении) мантиссы перемножаются (делятся).

3. Знаки произведения (частного) формируются путем сложения знаковых разрядов сомножителей (делимого и делителя). Возможные переносы из знакового разряда игнорируются.

Пример 12. Умножить и разделить числа A_{10} =30,5 и B_{10} = 210,5

В десятеричной системе счисления операции умножения и деления будут выполняться следующим образом:

$A_{10}= 0.305 \times 10^2$;

$B_{10}= 0.2105 \times 10^3$;

$A \times B= (0.305 \times 0.2105) \times 10^{2+3} = 0.0642025 \times 10^5 = 6420,25;$

$A \div B = (0.305 \div 0.2105) \times 10^{2-3} = 1.4489 \times 10^{-1} = 0,14489.$

Также, следуя преобразованиям по примеру 11, эти операции будут выполнены в двоичном представлении чисел.

5. *Арифметические операции над двоично-десятичными кодами чисел*

Третья форма представления двоичных чисел – двоично-десятичная. Ее появление объясняется следующим. При обработке больших массивов десятичных чисел (например, больших экономических документов) приходится тратить существенное время на перевод этих чисел из десятичной системы счисления в двоичную для последующей обработки и обратно – для вывода результатов. Каждый такой перевод требует выполнения двух-

четырех десятков машинных команд. С включением в состав отдельных компьютеров специальных функциональных блоков или спецпроцессоров десятичной арифметики появляется возможность обрабатывать десятичные числа напрямую, без их преобразования, что сокращает время вычислений. При этом каждая цифра десятичного числа представляется двоичной тетрадой.

Например, A_{10}=3759, $A_{2\text{-}10}$= 0011 0111 0101 1001.

Положение десятичной точки (запятой), отделяющей целую часть от дробной, обычно заранее фиксируется. Значение знака числа отмечается кодом, отличным от кодов цифр. Например, “+” имеет значение тетрады 1100, а “-” – 1101.

Арифметические действия над двоично-десятичными числами также приводятся к операции алгебраического сложения отдельных цифр чисел, представленных дополнительными кодами в соответствии с таблицей 1.

Приведем один из *алгоритмов сложения*, который получил довольно широкое распространение.

1. Сложение чисел начинается с младших цифр (тетрад) и производится с учетом возникающих переносов из младших разрядов в старшие.

2. Знак суммы формируется специальной логической схемой по знаку большего слагаемого.

3. Для того чтобы при сложении двоично-десятичных цифр возникали переносы, аналогичные при сложении чисел в десятичном представлении, необходимо проводить так называемую десятичную коррекцию. Для этого к каждой тетраде первого числа прибавляется дополнительно по цифре $6_{10}=0110_2$, что позволяет исключить шесть неиспользуемых комбинаций $(1010 .. 1111)_2$, так как они кодируют шестнадцатеричные цифры A-F (числа 10 .. 15).

4. После операции суммирования осуществляется корректировка суммы. Из тех тетрад суммы, из которых не было переносов, изымаются ранее внесенные избытки $6_{10}=0110_2$. Для этого проводится вторая коррекция. При этой второй коррекции переносы из тетрад блокируются.

5. Операция вычитания реализуется достаточно своеобразно. По общему правилу сложения к тетрадам числа с большим модулем прибавляются дополнительные коды тетрад другого числа. В качестве знака результата берется знак числа с большим модулем.

Пример 13. Сложить числа $A_{10} = 177$ и $B_{10} = 418$.

Проведем последовательно перевод и корректировку исходных чисел.

$A_{2\text{-}10} \rightarrow$	0001	0111	0111	
	0110	0110	0110	←1-я коррекция
$A' \rightarrow$	0111	1101	1101	
+				Сложение A′+B
$B_{2\text{-}10} \rightarrow$	0100	0001	1000	
$C' \rightarrow$	1011	1111	0101	
+				
	1010	1010		←2-я коррекция
$C_{2\text{-}10}$	0101	1001	0101	результат

В результате получаем C_{10}=595.

Задания к лабораторной работе

В соответствии с выданным преподавателем вариантом (см. таблицу 2) выполните следующие задания.

1. Представьте числа в прямом, обратном, дополнительном кодах.
2. Выполните сложение чисел в обратом и дополнительном кодах.
3. Представьте числа в нормализованном виде и выполните четыре арифметических операции над числами в формате с плавающей точкой.
4. Представьте числа из задания 3 в формате одинарной точности, записав представление в двоичном и шестнадцатеричном виде.
5. Переведите данное число из десятичной системы счисления в двоично-десятичную и выполните над ними операцию сложения. Представьте результат в десятичной системе счисления.

Таблица 2

Варианты индивидуальных заданий

Вариант 1		**Вариант 2**	
1	16, -29;	1	26, -13;
2	-34 и 11; -11 и -29	2	12 и -25; -13 и -29
3	+0,002289 -1001,202	3	-261,78 +0,00056
5	625 145	5	356 258
Вариант 3		**Вариант 4**	
1	18, -25;	1	19, -17;
2	-24 и 18; -11 и -28	2	19 и -25; -17 и -29
3	+0,001189 -1000,502	3	-252,78 +0,000806
5	223 195	5	356 198
Вариант 5		**Вариант 6**	
1	15, -24;	1	21, -15;
2	-32 и 15; -12 и -29	2	15 и -25; -13 и -27
3	+0,001579 -1011,002	3	-181,38 +0,00156
5	625 145	5	356 258
Вариант 7		**Вариант 8**	
1	20, -19;	1	22, -14;
2	-24 и 11; -11 и -19	2	17 и -24; -14 и -25
3	+0,001129 -2010,202	3	-169,68 +0,00061
5	335 225	5	353 238

Окончание таблицы 2

Вариант 9		**Вариант 10**	
1	21, -19;	1	17, -22;
2	-24 и 11; -19 и -21	2	17 и -25; -14 и -22
3	+0,002519 -3001,212	3	-311,71 +0,00016
5	345 265	5	398 158

Контрольные вопросы:

1. Что такое кодирование информации в общем смысле? Каково место кодирования среди процессов обработки информации?

2. Что такое машинный код? Назовите виды и отличия машинных кодов. Приведите примеры.

3. Как получить прямой и дополнительный коды целого числа? Приведите примеры.

4. Назовите формы представления вещественных чисел в памяти компьютера? Приведите примеры.

5. Что такое нормализованный вид числа? Приведите примеры.

6. Какую форму представления чисел называют двоично-десятичной? Для чего она используется?

7. Сформулируйте алгоритм выполнения арифметической операции сложения двоично-десятичных чисел. Приведите пример.

8. Перечислите правила сложения (вычитания) чисел с фиксированной запятой. Приведите примеры.

9. Перечислите правила сложения (вычитания) чисел с плавающей запятой. Приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА № 3
ПРЕДСТАВЛЕНИЕ ДРУГИХ ВИДОВ ИНФОРМАЦИИ В КОМПЬЮТЕРЕ

Цель работы: Закрепить знания о кодировании символьной и графической информации в компьютере, приобрести навык представления текстовой и графической информации в компьютере.

Краткие теоретические сведения

1. *Представление символьной информации*

Для символьного кодирования в современных компьютерных системах используют универсальный код UNICODE (UNIversal CODE) – стандарт 16-разрядного кодирования символов. Он определяет коды, обеспечивающие идентификацию различных символов: букв, иероглифов, цифр и т. д. Код может использоваться вместо 7–8-битовых, в том числе и ASCII. Поскольку в 16-разрядном UNICODE можно закодировать 65 536 символов вместо 128 в ASCII, то отпадает необходимость в создании модификаций таблиц кодов. Это существенно упрощает обработку текстовых файлов, хотя и несколько увеличивает их размеры.

UNICODE охватывает 28 000 букв, знаков, слогов, иероглифов. 30 000 мест в UNICODE зарезервировано для ввода математических или технических символов, а также создания пользовательских символов.

Единая стандартизация языковых форматов наводит порядок в международном кодировании алфавитов различных языков. Здесь учтено также то, что в таких языках, как иврит и арабский, текст пишется справа налево.

При передаче данных часто используются избыточные коды, т. е. такие, которые за счет усложнения структуры позволяют повысить надежность передачи данных. К ним, в первую очередь, относятся коды с обнаружением ошибок. Чаще всего это циклические избыточные коды. Простая разновидность такого кода – код с контролем по четности. Широко используется для обнаружения ошибок в блоках данных также код контроля циклической избыточности CRC.

Важное значение имеют коды с исправлением ошибок. Использование этих кодов позволяет с большой вероятностью не только обнаруживать, но и исправлять возникшие при передаче ошибки.

2. *Представление графической информации*

Графическая информация на экране монитора представляется в виде изображения, которое формируется из точек (пикселей). В современных компьютерах разрешающая способность (количество точек на экране), а также количество цветов зависят от видеоадаптера и могут изменяться на программном уровне.

Цветные изображения могут иметь различные режимы: 16 цветов, 256 цветов, 65536 цветов (high color), 16777216 цветов (true color) и т.д.

Рассчитать количество бит на точку (пиксель), например, режима high color можно, используя формулу Хартли: **$I=\log_2 65536=16$ бит=2 байта**.

Объем видеопамяти, необходимый для хранения битовой карты изображений, рассчитывается с учетом такого показателя, как **разрешающая способность экрана**.

Например, для режима high color и разрешающей способности экрана, равной 800 на 600 точек, т. е. 480 000 точек, необходимый объем видеопамяти будет рассчитан следующим образом:

V= 2 байта × 480 000 = 960 000 байт = 937,5 Кбайт.

Аналогично рассчитывается объем видеопамяти, необходимый для хранения битовой карты изображений в других видеорежимах.

В таблице 1 представлены примеры объемов занимаемой памяти видеокарты в зависимости от количества кодируемых цветов и разрешающей способности экрана.

Таблица 1

Характеристики различных стандартов представления графики

Разрешение	16 цветов	256 цветов	65536 цветов	16777216 цветов
640x480	150 Кбайт	300 Кбайт	600 Кбайт	900 Кбайт
800x600	234,4 Кбайт	468,8 Кбайт	937,5 Кбайт	1,4 Мбайт
1024x768	384 Кбайт	768 Кбайт	1,5 Мбайт	2,25 Мбайт
1280x1024	640 Кбайт	1,25 Мбайт	2,5 Мбайт	3,75 Мбайт

В видеопамяти компьютера хранится битовый план (bit map), являющийся двоичным кодом изображения, отсюда она считывается (не реже 50 раз в секунду) и отображается на экране.

Для монохромных изображений общепринятым считается кодирование цвета одного пикселя в 1байте. Это позволяет передать 254 оттенка серого плюс черный и белый цвета (всего 256 вариантов).

Цветные изображения могут кодироваться разными способами в зависимости от того, для какой цели создается рисунок.

Метод (система) RGB (True color) – от слов Red, Green, Blue: удобен для изображений, рассматриваемых на экране. Оттенки цвета создаются смешением лучей трех базовых цветов разной интенсивности. Под значение интенсивности каждого луча отводится 1 байт, т. е. различают 256 уровней интенсивности. Для совокупности трех лучей получается 256^3=16 777 216, т.е. порядка 17 млн разных вариантов, каждый из которых создает свой оттенок цвета.

Метод (система) CMYK – от слов голубой (Cyan), пурпурный (Magenta), желтый (Yellow), черный (blacK): удобен для изображений, которые предполагается печатать на бумаге. Он учитывает особенности полиграфии, в которой цвет получается смешением четырèх красок. Для кодирования одного пикселя требуется 4 байта, можно передать $256^4 \approx 4$ млрд оттенков.

Для WEB-документов учитывать такое обилие оттенков неудобно, так как это приводит к файлам очень больших размеров, их неудобно пересылать по сети. Поэтому в них используются так называемые ***индексированные цвета***: из всего обилия возможных комбинаций выбрано 256 базовых оттенков. Это позволило для запоминания цвета каждого пикселя использовать только 1 байт. Каждому состоянию байта сопоставляется определенная комбинация интенсивностей базовых цветов. *.JPG, *.GIF, *.PNG- кодировки объединяют области рисунка, закрашенные близкими оттенками, и сохраняют для них усредненный цвет. За счет этого размеры графических файлов существенно уменьшаются.

В компьютерной документации коды RGB- и CMYK-цвета представляются так:

1. Интенсивности каждого базового цвета перечисляются в том порядке, который применен в аббревиатуре используемой системы. Эти интенсивности представляются в 16-ричной позиционной системе.

2. Перед полученным кодом оттенка размещают символ #.

Пример 1. По заданному коду определить базовые цвета и их интенсивность: #000000, #B5B5B5, #FFFFFF, #CF35D1.

#000000 – чѐрный цвет – нулевая интенсивность каждого луча, т. е. нет ни одного цвета;

#B5B5B5 – какой-то оттенок серого цвета, т. к. интенсивности всех лучей одинаковы;

#FFFFFF – белый цвет – все цвета в максимальной интенсивности;

#CF35D1 – номера интенсивностей базовых лучей в десятичной системе: красного CF = 12 × 16 + 15 = 207; зеленого – 35 = 3 × 16 + 5 = 53; синего – D1 = 13 × 16 + 1 = 209.

Задания к лабораторной работе

В соответствии с выданным преподавателем вариантом (см. таблицу 2) выполните следующие задания.

1. Определите, сколько бит содержит сообщение.

2. Отсортируйте по возрастанию последовательность текстовых величин.

3. Определите объем памяти, который потребуется для кодировки заданной фразы в UNICODE.

4. По заданному коду определите базовые цвета, их интенсивность и объем занимаемой памяти.

5. Нарисуйте в графическом редакторе цветное изображение на холсте размера 250×200 пикселей. Закодируйте изображение методом (в системе) RGB в двоичном и html представлении (используйте таблицы кодировки, предложенные преподавателем и хранящиеся на сервере компьютерного класса). Вычислите информационный объём нарисованного изображения.

Таблица 2

Варианты индивидуальных заданий

Вариант 1		**Вариант 2**	
1	На улице идет дождь	1	Я вымокла насквозь
2	8б; 8а; 10а; 10б; 11а	2	5а; 9а; 1в; 10б; 5в
3	Я помню чудное мгновенье	3	Передо мной явилась Ты
4	#1C2F55	4	#2B5C8F

Окончание таблицы 2

Вариант 3		**Вариант 4**	
1	Когда Вы зайдете?	1	Завтра будет Солнце?
2	6а; 5в; 1л; 10в; 3б	2	4д; 5а; 8в; 3а; 8б
3	Как мимолетное виденье	3	Как гений чистой красоты
4	#6C2B51	4	#BB4C8F
Вариант 5		**Вариант 6**	
1	На Марсе идет дождь?	1	Вы же вымокли насквозь!
2	5б; 4д; 10в; 10б; 11д	2	5д; 9а; 11в; 10а; 5в
3	У меня зазвонил телефон	3	Мне снились милые черты
4	#CC2BA1	4	#BA4B8F
Вариант 7		**Вариант 8**	
1	Дует северо-западный ветер	1	Самая высокая гора – Эверест
2	5в; 6д; 10а; 10б; 11д	2	5д; 9а; 11а; 10д; 9в
3	Да здравствует мыло душистое	3	Звучал мне долго голос нежный
4	#6C23A9	4	#B95B88
Вариант 9		**Вариант 10**	
1	Птицы осенью улетают на юг	1	Ель и сосна – хвойные деревья
2	7в; 6е 10а; 10б; 11а	2	10в; 6д; 7в; 10б; 8а
3	В томленьях грусти безнадежной	3	В тревогах шумной суеты
4	#8A29A9	4	#AA4B4F

Контрольные вопросы:

1. Что такое кодирование информации в общем смысле?
2. Каково место кодирования среди процессов обработки информации?
3. Как кодируется текстовая информация? Приведите примеры кодирования и декодирования.
4. Как кодируется графическая информация? Приведите примеры кодирования и декодирования.
5. Как кодируется звуковая информация? Приведите примеры кодирования и декодирования.
6. Как рассчитать объем памяти, необходимый для хранения 1 байта текстовой информации?
7. Как рассчитать объем памяти, необходимый для хранения черно-белого изображения размером 20×20 пикселей?

ЛАБОРАТОРНАЯ РАБОТА № 4
ПОСТРОЕНИЕ ЛОГИЧЕСКИХ СХЕМ

Цель работы: По функции переключения $f(A, B..Z)$ научиться строить функциональную схему, состоящую из логических элементов И, ИЛИ, НЕ. Освоить алгоритм построения таблиц истинности для логических функций.

Краткие теоретические сведения

Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры 1 и 0, а значений логических переменных тоже два: «1» и «0».

Из этого следует два вывода:

– одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных;

– на этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера и, следовательно, уменьшить число элементарных логических элементов, из десятков тысяч которых состоят основные узлы компьютера.

Схема, выражающая функцию переключения f(A,B..Z) через логические элементы И, ИЛИ, НЕ, называется ее **логической схемой.**

Любая функция переключения может быть выражена посредством логических элементов И, ИЛИ, НЕ.

Логический элемент компьютера – это часть электронной логической (функциональной) схемы, которая реализует элементарную логическую функцию.

Схемы строятся из логических блоков. **Логический блок** – это устройство, предназначенное для обработки информации в цифровой форме.

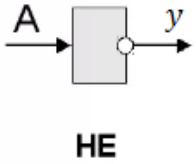
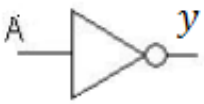
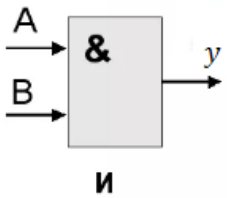
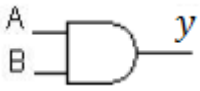
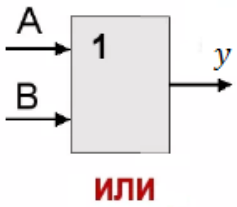
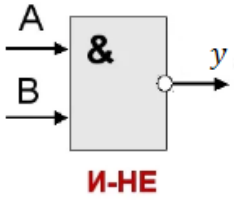
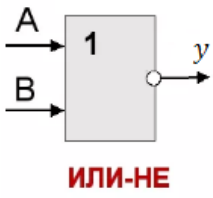
Таблица истинности является универсальным средством задания логической функции. Она включает все наборы для заданного количества переменных, определяющих значение логической функции, с указанием значений, которые принимает функция для каждого набора. В одной таб-

лице истинности может задаваться несколько логических функций, зависящих от одних и тех же переменных.

Каждой логической функции соответствует свой логический блок и таблица истинности (таблица 1).

Таблица 1

Основные логические элементы

№ п/п	Логическая операция	Название логического элемента	Условное обозначение логического элемента принятое в отечественной литературе	Международное обозначение логического элемента	Таблица истинности
1	Отрицание $y = \bar{A}$	НЕ (NOT)			A | Y 0 | 1 1 | 0
2	Конъюнкция $y = A \wedge B$ (также $A \cdot B, AB, A\&B$)	И (AND)			A | B | Y 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
3	Дизъюнкция $y = A \vee B$ (также $A + B$)	ИЛИ (OR)			A | B | Y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
4	Конъюнкция с отрицанием $y = \overline{A \wedge B}$ (также $\overline{A \cdot B}, \overline{AB}, \overline{A\&B}$)	И-НЕ			A | B | Y 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
5	Дизъюнкция с отрицанием $Y = \overline{A \vee B}$ (также $\overline{A + B}$)	ИЛИ-НЕ			A | B | Y 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0

При составлении таблицы истинности важно учитывать следующий порядок выполнения логических операций:

1. инверсия – НЕ;
2. конъюнкция – И;
3. дизъюнкция – ИЛИ.

Алгоритм построения таблицы истинности логической функции

1. Определяют количество строк: кол-во строк = 2^n + 1 (для строки заголовка), n – количество простых выражений.

Например, для функций двух переменных существует 2^2=4 комбинации наборов значений переменных, для функций трех переменных – 2^3=8 и т.д.

2. Определяют количество столбцов: кол-во столбцов = кол-во переменных + кол-во логических операций. При определении количества логических операций учитывают также порядок их выполнения.

3. Заполняют столбцы результатами выполнения логических операций в определенной последовательности, учитывая таблицы истинности основных логических операций.

Алгоритм построения функциональных схем

1. Определить число логических переменных (число входных сигналов).

2. Определить количество логических операций и их порядок: инверсия (НЕ); выражения в скобках; конъюнкция (И); дизъюнкция (ИЛИ).

3. Изобразить для каждой логической операции соответствующий ей логический элемент.

4. Соединить логические элементы в порядке выполнения логических операций.

Замечания

1) Логический элемент «НЕ» (инвертор) является простейшим логическим элементом, выполняющим функцию отрицания. У логического элемента «НЕ» один вход и один выход. Говорят также, что элемент «НЕ» инвертирует значение входной двоичной переменной. Если на вход поступает сигнал, соответствующий 1, то на выходе будет 0. И наоборот.

Логический элемент «И» (конъюнктор) выдает на выходе значение логического произведения входных сигналов. Он имеет один выход и не менее двух входов. Сигнал на выходе конъюнктора появляется тогда и только тогда, когда поданы сигналы на все входы.

Логический элемент «ИЛИ» (дизъюнктор) выдает на выходе значение логической суммы входных сигналов. Он имеет один выход и не менее двух входов. Сигнал на выходе дизъюнктора не появляется тогда и только тогда, когда на все входы не поданы сигналы.

2) При построении функциональной схемы указывать стрелки соединения необязательно.

Для каждой функциональной схемы можно сделать оценку ее сложности, которая выражается **ценой схемы С**. Цена С определяется суммарным числом входов логических элементов. Чем меньше величина С, тем проще функциональная схема.

Рассмотрим на примерах построение функциональных схем по заданной функции переключения.

Пример 1. По функции переключения построить функциональную схему и таблицу истинности к ней: $\boldsymbol{y = \bar{A} + B \cdot \bar{C}}$.

Будем строить функциональную схему по алгоритму, предложенному в теоретической части.

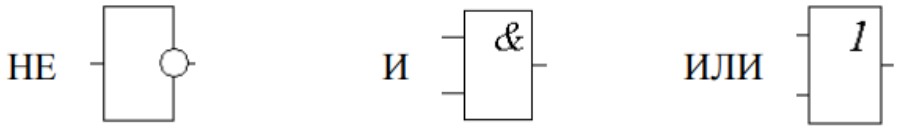
Порядок выполнения

1) Определяем число логических переменных. Число логических переменных три – A, B, C. Значит, число входных сигналов – 3.

2) Определяем количество логических операций и их порядок Число логических операций – четыре:

1. $\bar{A}$ – НЕ
2. $\bar{C}$ – НЕ
3. $B \cdot \bar{C}$ – И (логическое умножение)
4. $\bar{A} + B \cdot \bar{C}$ – ИЛИ (логическое сложение).

3) Изображаем для каждой логической операции соответствующий ей логический элемент.

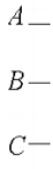
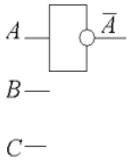
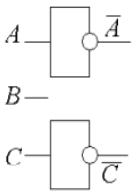
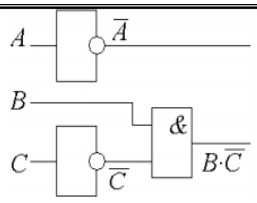
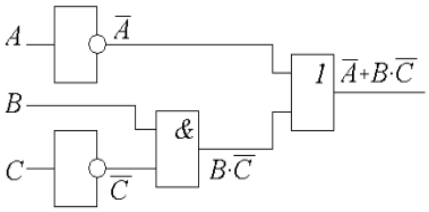


4) Чтобы соединить логические элементы в порядке выполнения логических операций, будем выполнять соединение по шагам.

Итак, схема составляется, как конструктор, и строится слева направо (таблица 2):

Таблица 2

Построение функциональной схемы

Шаг 1.Рисуем три входных высказывания:	**Шаг 2.**Строим блок «НЕ» $\bar{A}$
	
Шаг 3. Строим блок «НЕ» $\bar{C}$	**Шаг 4.** Строим блок «И» $B \cdot \bar{C}$
	
Шаг 5. Строим блок «ИЛИ» $\bar{A} + B \cdot \bar{C}$. Схема готова.	
	

Теперь построим таблицу истинности для функции $y = \bar{A} + B \cdot \bar{C}$.

Придерживаемся алгоритма, предложенного в теоретической части.

1) Определим количество строк: количество переменных – 3, значит, количество строк равно 2^3=8. И еще добавляем 1 строку для заголовка.

2) Определяем количество столбцов.

Количество переменных – 3. Количество логических операций – 4, и их последовательность определена выше. Значит, число столбцов 3+4=7.

Заполняем столбцы результатами выполнения логических операций в определенной последовательности, учитывая таблицы истинности основных логических операций. Исходная таблица примет вид (таблица 3).

Таблица 3

A	B	C	$\bar{A}$	$\bar{C}$	$B \cdot \bar{C}$	$y = \bar{A} + B \cdot \bar{C}$
0	0	0	1	1	0	1
0	0	1	1	0	0	1
0	1	0	1	1	1	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	1	0	0	0	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

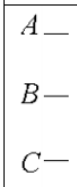
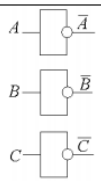
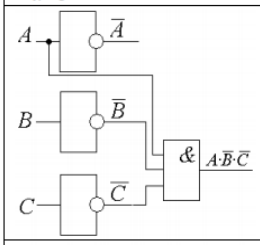
Пример 2. Построить функциональную схему функции переключения $\boldsymbol{y = A \cdot \bar{B} \cdot \bar{C} + \overline{\bar{A} \cdot B} \cdot C}$ и составить таблицу истинности к ней.

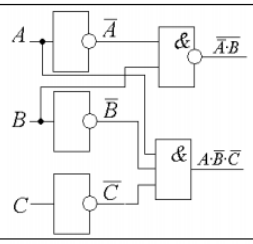
Как и в предыдущем примере воспользуемся алгоритмами построения из теоретической части.

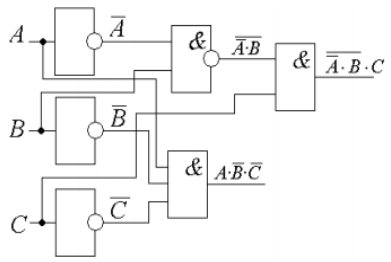
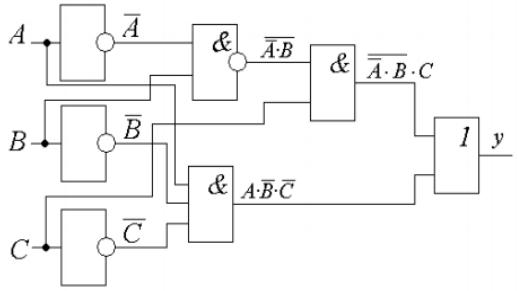
Определим количество и последовательность логических операций:

1. $\bar{B}$
2. $A \cdot \bar{B}$
3. $\bar{C}$
4. $A \cdot \bar{B} \cdot \bar{C}$
5. $\bar{A}$
6. $\bar{A} \cdot B$
7. $\overline{\bar{A} \cdot B}$
8. $\overline{\bar{A} \cdot B} \cdot C$
9. $A \cdot \bar{B} \cdot \bar{C} + \overline{\bar{A} \cdot B} \cdot C$

Перейдем к пошаговому построению функциональной схемы.

<table>
<tr><td>Шаг 1.Рисуем три входных высказывания:</td><td>Шаг 2.Строим блоки «НЕ» $\bar{A}, \bar{B}, \bar{C}$</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Шаг 3. Строим блок «И»: умножаем A на $\bar{B}$ и на $\bar{C}$</td><td>Шаг 4. Строим блок «И-НЕ» $\overline{\bar{A} \cdot B}$</td></tr>
<tr><td> 

Обратите внимание, что место соединения линий отмечено точкой.</td><td> 

Обратите внимание, что место, где линии не пересекаются, а скрещиваются, точкой не отмечается.</td></tr>
<tr><td>Шаг 4. Строим блок «И»: умножаем $\overline{\bar{A} \cdot B}$ на C</td><td>Шаг 5. Осталось последнее действие: Строим блок «ИЛИ»: складываем $A \cdot \bar{B} \cdot \bar{C}$ и $\overline{\bar{A} \cdot B} \cdot C$. Схема готова.</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

Теперь построим таблицу истинности для функции

$y = A \cdot \bar{B} \cdot \bar{C} + \overline{\bar{A} \cdot B} \cdot C.$

Так же, как и в предыдущем примере, будем руководствоваться алгоритмом, предложенным в теоретической части.

1) Определим количество строк: количество переменных – 3, значит, кол-во строк равно 2^3=8. И еще добавляем 1 строку для заголовка.

2) Определяем количество столбцов.

Количество переменных – 3. Количество логических операций – 9.

Значит, число столбцов 3+9=12.

Заполняем столбцы результатами выполнения логических операций в последовательности, определенной выше, учитывая таблицы истинности основных логических операций.

Исходная таблица истинности заданной функции примет следующий вид.

A	B	C	$\bar{B}$	$A \cdot \bar{B}$	$\bar{C}$	$A \cdot \bar{B} \cdot \bar{C}$	$\bar{A}$	$\bar{A} \cdot B$	$\overline{\bar{A} \cdot B}$	$\overline{\bar{A} \cdot B} \cdot C$	y
0	0	0	1	0	1	0	1	0	1	0	1
0	0	1	1	0	0	0	1	0	1	1	1
0	1	0	0	0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1	0	0	1
1	0	0	1	1	1	1	0	0	1	0	0
1	0	1	1	1	0	0	0	0	1	1	0
1	1	0	0	0	1	0	0	0	1	0	1
1	1	1	0	0	0	0	0	0	1	1	0

Задания к лабораторной работе

В соответствии с выданным преподавателем вариантом для заданной функции переключения y (A, B, C) построить функциональную схему и таблицу истинности к ней.

Варианты индивидуальных заданий

1. $y = \overline{(A \cdot B + \bar{A} \cdot C)} \cdot \bar{C}$
2. $y = (A + \bar{C}) \cdot (\bar{A} + B) \cdot \bar{B}$
3. $y = A \cdot \bar{B} + A \cdot \overline{B + C}$
4. $y = \overline{(A + B + C) \cdot (\bar{A} + \bar{B})}$
5. $y = \bar{A} \cdot B + \overline{A \cdot \bar{B} \cdot C}$
6. $y = \overline{\bar{A} \cdot B} + A \cdot \bar{B} + C$

7. $y = (A + \overline{C}) \cdot (\overline{A} + \overline{B}) \cdot C$
8. $y = \overline{(\overline{A} + B \cdot D)} + (\overline{D} + A)$
9. $y = \overline{C} \cdot \overline{B \cdot \overline{A}} + \overline{B} \cdot A$
10. $y = \overline{\overline{B \cdot A} + B \cdot C \cdot \overline{A}}$
11. $y = \overline{\overline{A} \cdot \overline{C}} \cdot B + \overline{B} \cdot C$
12. $y = \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B}$
13. $y = \overline{(\overline{A + B}) \cdot (\overline{A + C} + C)}$
14. $y = \overline{A} \cdot \overline{C} \cdot \overline{B} + A \cdot B$
15. $y = \overline{A \cdot \overline{B} + C \cdot \overline{A}} + B$

Контрольные вопросы:

1. Что такое алгебра логики? Для чего она используется?

2. Дайте определение понятиям: функция переключения, логический элемент компьютера, логическая схема, таблица истинности. Приведите примеры.

3. Дайте краткую характеристику базовым логическим элементам.

4. Назовите основные логические операции и сопоставьте с ними логические элементы функциональных схем компьютера. Ответ поясните.

5. Перечислите последовательно этапы построения логической схемы. Приведите пример.

6. Перечислите последовательно этапы построения таблицы истинности. Приведите пример.

7. Что такое цена схемы? Как ее определить? Приведите пример.

ЛАБОРАТОРНАЯ РАБОТА № 5
МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ

Цель работы: Научиться записывать логические функции в двух канонических формах. Научиться минимизировать логические функции несколькими способами: с помощью основных тождеств и теорем алгебры логики и карт Карно-Вейча.

Краткие теоретические сведения

1. *Запись функций алгебры логики в различных формах, их взаимосвязь*

Одну и ту же логическую функцию можно представить различными логическими выражениями. Среди множества выражений, которыми представляется логическая функция, особое место занимают две канонические формы: **совершенная конъюнктивная нормальная форма (СКНФ)** и **совершенная дизъюнктивная нормальная форма (СДНФ)**.

Совершенная дизъюнктивная нормальная форма представляет собой дизъюнкцию простых конъюнкций, где под термином *простая конъюнкция* имеется в виду конъюнкция переменных или их отрицаний. В СДНФ простые конъюнкции содержат все переменные в своей прямой или инверсной форме и представляют собой наборы, на которых описываемая функция имеет единичное значение. Такие конъюнкции называются *конституентами единицы* рассматриваемой функции. Поэтому СДНФ представляет собой дизъюнкцию (логическую сумму), слагаемыми которой являются конституенты единицы.

Общая запись СДНФ функции *y* имеет вид:

$$y = \vee\, x_1^{\delta 1}\, x_2^{\delta 2}\, x_3^{\delta 3} \ldots x_{(n-1)}^{\delta(n-1)}\, x_n^{\delta n},$$

$$x_i^{\delta_i} = \begin{cases} x_i, \text{если } \delta_i = 1, \\ \overline{x_i}, \text{если } \delta_i = 0. \end{cases}$$

СДНФ легко сформировать на основе таблицы истинности. Например, если функции задаются в виде таблицы истинности, то СДНФ для них будет иметь следующий вид (табл. 1):

Таблица 1

Таблица истинности СДНФ

x_1	x_2	x_3	y_1	y_2	y_3
0	0	0	1	1	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

В итоге логическое выражение СДНФ следующее:

$$y_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3;$$

$$y_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3;$$

$$y_3 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 + x_1x_2x_3.$$

Совершенная конъюнктивная нормальная форма – это конъюнкция простых дизъюнкций, где под термином *простая дизъюнкция* имеется в виду дизъюнкция переменных или их отрицаний. В СКНФ простые дизъюнкции содержат все переменные в своей прямой или инверсной форме и представляют собой отрицание конституент нуля.

Общая запись СКНФ функции *y* имеет вид:

$$y = \wedge (x_1^{\delta 1} + x_2^{\delta 2} + x_3^{\delta 3} + \ldots + x_{(n-1)}^{\delta(n-1)} + x_n^{\delta n}),$$

$$x_i^{\delta_i} = \begin{cases} x_i, \text{если } \delta_i = 1, \\ \bar{x}_i, \text{если } \delta_i = 0. \end{cases}$$

СКНФ легко сформировать на основе таблицы истинности. Например, для функций, из предыдущей таблицы 1 имеем

$$y_1 = (x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3);$$

$$y_2 = (x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3) \times$$
$$\times (\bar{x}_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3);$$

$$y_3 = (x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + x_2 + x_3).$$

СКНФ строится на основе конституенты нуля.

Конституента нуля представляет набор логических переменных, на котором логическая функция принимает значение «0».

Каждая скобка в приведенных выражениях представляет собой отрицание конституенты нуля соответствующей функции, а запись функции в виде конъюнкции таких скобок представляет собой условие, при котором отсутствуют все конституенты нуля определяемой функции, при выполнении которого функция имеет единичное значение.

Таким образом любую функцию можно представить или в СДНФ, или в СКНФ, а так как эти формы представлены в базисе Буля, то отсюда следует, что этот базис (базис И, ИЛИ, НЕ) является *функционально полным*.

Если функция задана в СДНФ и требуется найти ее СКНФ, то такой переход можно выполнить, составив по заданной СДНФ таблицу истинности для этой функции. На основе полученной таблицы составляется СКНФ заданной функции.

2. *Минимизация функций с помощью основных теорем и тождеств алгебры логики*

Учитывая то, что одну и ту же логическую функцию можно представить различными выражениями, перед реализацией функции в виде логической схемы весьма важным является выбор из всех возможных выражений, соответствующих данной функции, самого простого. Решить эту проблему можно за счет использования процедуры минимизации логического выражения. Чем проще логические выражения, описывающие функции, тем проще и дешевле реализующая их цифровая схема.

Один из способов *минимизации логических выражений* и последующего *построения эквивалентных схем* является использование законов, правил и операций булевой алгебры. Приведём часто используемые.

Переместительный (коммутативный) закон. Закон справедлив как для конъюнкции, так и для дизъюнкции.

$x_1 + x_2 = x_2 + x_1$ – *от перемены мест логических слагаемых сумма не меняется*.

$x_1x_2 = x_2x_1$ – *от перемены мест логических сомножителей их произведение не меняется*.

Этот закон справедлив для любого количества логических операндов.

Сочетательный (ассоциативный) закон справедлив как для конъюнкции, так и для дизъюнкции.

$x_1 + x_2 + x_3 + x_4 = (x_1 + x_4) + (x_2 + x_3)$ – *при логическом сложении отдельные слагаемые можно заменить их суммой.*

$x_1\, x_2\, x_3\, x_4 = (x_1\, x_4)\, (x_2\, x_3)$ – *при логическом умножении отдельные логические сомножители можно заменить их произведением*.

Распределительный (дистрибутивный) закон

$(x_1 + x_2)\, x_3 = x_1x_3 + x_2x_3$;

$(x_1 + x_2)\, (x_1 + x_3) = x_1 + x_2\, x_3$.

Правило де Моргана

1) $\overline{x_1 + x_2} = \overline{x_1}\ \overline{x_2}$ – отрицание суммы равно произведению отрицаний;

2) $\overline{x_1}\ \overline{x_2} = \overline{x_1} + \overline{x_2}$ – отрицание произведения равно сумме отрицаний.

Операция склеивания

1) $\overline{x_i}A + x_i = A$ – операция *склеивания* для конъюнкций, где A – переменная или любое логическое выражение;

2) $(\overline{x_i} + A)(x_i + A) = A$– операция *склеивания* для дизъюнкций.

Операции с отрицаниями

$\bar{\bar{x}} = x$ – двойное отрицание равносильно отсутствию отрицания.

Операции с одинаковыми операндами

$x_1 + x_1 + x_1 + \ldots + x_1 = x_1$;

$x_1\, x_1\, x_1\, \ldots .\, x_1 = x_1$ при любом числе повторений.

Пример 1. По таблице истинности построить логическую функцию и минимизировать ее, используя основные тождества алгебры логики.

A	**B**	**C**	**F**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Вначале приведем *алгоритм построения логической функции по ее таблице истинности*:

1. Выделить в таблице истинности те строки, в которых значение функции равно 1.

2. Выписать искомую формулу в виде дизъюнкции нескольких логических элементов. Число этих элементов равно числу выделенных строк.

3. Каждый логический элемент в этой дизъюнкции записать в виде конъюнкции аргументов функции.

4. Если значение какого-либо аргумента функции в соответствующей строке таблице равно 0, то этот аргумент взять с отрицанием.

5. Используя правило склеивания, упростить полученную функцию, заданную в СДНФ. Для этого в СДНФ сначала склеиваются между собой конъюнкции ранга n, затем полученные конъюнкции ранга $(n-1)$, $(n-2)$, и так до тех пор, пока в выражении для переключательной функции не останется ни одной пары склеиваемых между собой конъюнкций.

Операция склеивания позволяет понизить ранг конъюнкций, сократить их число и, следовательно, уменьшить количество логических элементов в функциональной схеме.

Итак, воспользуемся предложенным алгоритмом.

1. В 2, 4 и 6 строках таблицы истинности значение функции равно 1.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

2. Так как строки три, получаем дизъюнкцию трех элементов: ( ) + ( ) + ( ).

3. Каждый логический элемент в этой дизъюнкции запишем в виде конъюнкции аргументов функции A, B и C.

4. Берем аргумент с отрицанием, если его значение в соответствующей строке таблицы равно 0, и получаем искомую функцию:

$$F = (\overline{\mathrm{A}}\,\overline{\mathrm{B}}\,\mathrm{C}) + (\overline{\mathrm{A}}\,\mathrm{B}\,\mathrm{C}) + (\mathrm{A}\,\overline{\mathrm{B}}\,\mathrm{C})$$

5. Проведем минимизацию логической функции с целью построения упрощенной эквивалентной схемы. Для этого выполним склеивание конъюнкций ($\overline{\text{A}}\,\overline{\text{B}}\,\text{C}$) и ($\overline{\text{A}}\,\text{B}\,\text{C}$) по переменной B и конъюнкций ($\overline{\text{A}}\,\overline{\text{B}}\,\text{C}$) и ($\text{A}\,\overline{\text{B}}\,\text{C}$) по переменной A. В результате функция F преобразуется к виду $F = (\overline{\text{A}}\,\text{C}) + (\overline{\text{B}}\,\text{C})$.

3. *Минимизация функций с диаграммами Вейча (картами Карно)*

Если число логических переменных не превышает 5-6, минимизацию логических выражений удобно производить с помощью карт Карно или диаграмм Вейча. Минимизацию проводят путем объединения наборов (термов) на карте Карно.

Карта Карно для *n* логических переменных представляет собой множество квадратов (клеток), объединенных в близкую к квадрату прямоугольную форму. Каждая такая клетка соответствует одному набору логических переменных, причем наборы двух соседних клеток должны отличаться на значение одной переменной (их наборы образуют склеивающиеся конъюнкции).

Для примера на рисунке 1 приведем диаграмму Вейча функции трех переменных. Она содержит восемь клеток.

	x_1	x_1	$\bar{x}_1$	$\bar{x}_1$
x_0	$f(\bar{x}_2, x_1, x_0)$	$f(x_2, x_1, x_0)$	$f(x_2, \bar{x}_1, x_0)$	$f(\bar{x}_2, \bar{x}_1, x_0)$
$\bar{x}_0$	$f(\bar{x}_2, x_1, \bar{x}_0)$	$f(x_2, x_1, \bar{x}_0)$	$f(x_2, \bar{x}_1, \bar{x}_0)$	$f(\bar{x}_2, \bar{x}_1, \bar{x}_0)$
	$\bar{x}_2$	x_2	x_2	$\bar{x}_2$

Рисунок 1 – Диаграмма Вейча функции трех переменных

Наборы входных переменных, соответствующие крайним левому и правому столбцам, являются соседними. Поэтому данную карту удобно представить как поверхность цилиндра, и она, в отличие от карты двух переменных, является объемной фигурой.

При минимизации логической функции используют либо ее нулевые, либо единичные значения. В обоих случаях получают равносильные выражения, которые, однако, могут отличаться по числу членов (т. е. цене) и выполняемым логическим операциям. Алгоритм минимизации логической функции (ЛФ) в общем виде сводится к следующим шагам.

1. На карте Вейча ЛФ *n*-переменных выделяют прямоугольные области, объединяющие выбранные значения функции (логический 0 или логическая 1). Каждая область должна содержать 2^k клеток, где k – целое число. Выделенные области могут пересекаться, т.е. одна или несколько клеток могут включаться в различные области.

2. Каждой из выделенных областей соответствует *k*-куб исходной ЛФ, который представляется самостоятельным логическим произведением переменных, значения которых в рамках выделенной области остаются постоянными. Каждое произведение содержит *n-k* переменных и носит название *импликанты*.

3. Из полученного множества выбирают минимальное число максимально больших областей, включающих все выбранные значения ЛФ.

4. Логически суммируют импликанты, соответствующие выбранным областям. Полученная сумма образует МДНФ – покрытие ЛФ минимальной стоимости (покрытие Квайна).

При объединении клеток с единичными значениями ЛФ получают МДНФ самой функции, а при объединении клеток с нулевыми значениями ЛФ – МДНФ функции, инверсной заданной.

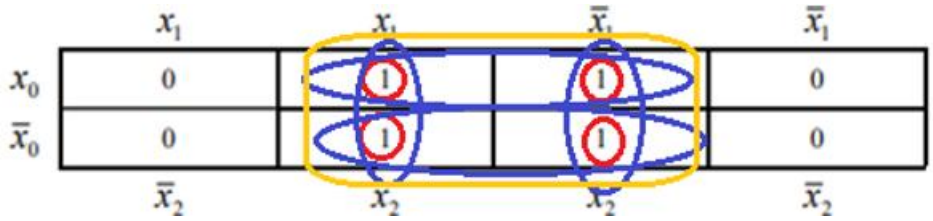
Последнее легко объясняется при помощи *закона исключённого третьего:* $\mathrm{A} + \overline{\mathrm{A}} = 1$ и *закона противоречия:* $\mathrm{A} \cdot \overline{\mathrm{A}} = 1$.

Пример 2. Минимизировать логическую функцию.

$f(x) = x_2\, x_1 x_0 + \; x_2\, \overline{x_1}\, x_0 + \; x_2\, x_1\, \overline{x_0} + \; x_2\, \overline{x_1}\, \overline{x_0}$.

Воспользуемся алгоритмом, предложенным в теоретической части.

1) По заданной ЛФ составим карту Вейча.



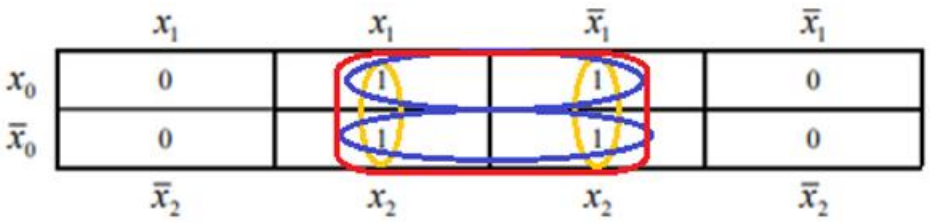
2) Запишем кубические комплексы:

$К_0$ = (111, 101, 110, 100);

$К_1$ = (1-1, 1-0, 11-, 10-);

$К_2$ = (1--)

3) Выделим три покрытия (контуры, охватывающие все единицы таблицы в количестве, равном 2^k , где k – целое число при минимальном количестве пересечений) и запишем соответствующие им ДНФ:



Каждое покрытие имеет свое расположение: $П_1$ – два горизонтальных овала, $П_2$ – два вертикальных овала, $П_3$ – центральный прямоугольник из четырех единиц.

$П_1$ = (1-1, 1-0), которое соответствует функции $f(x) = x_2\, x_0 + x_2\, \overline{x_0}$;

$П_2$ = (11-, 10-), которое соответствует функции $f(x) = x_2\, x_1 + x_2\, \overline{x_1}$;

$П_3$ = (1--), которое соответствует функции $f(x) = x_2$.

При определении покрытия каждая тройка символов характеризует значения набора $x_2\, x_1 x_0$: 1 или 0 фиксируют постоянное значение переменной, символ «-» – изменяемое значение переменной внутри конкретного контура. Например, покрытие $П_1$ = (1-1, 1-0) состоит из двух горизонтальных контуров, охватывающих все единицы. Верхний контур фиксирует неименное значение двух переменных – x_2 и x_0, причем неинвертированных, то есть равных 1. Переменная x_1 внутри рассматриваемого контура может принимать значения как 1, так и 0 (контур охватывает область, где имеются значения прямые x_1 и инвертированные $\overline{x_1}$), значит, этот контур можно описать набором 1-1. Второй контур покрытия (нижний горизонтальный овал) полностью аналогичен верхнему, за тем исключением, что фиксирует значение переменной x_0, равное 0, то есть $\overline{x_0}$. Таким образом, второй контур описывается набором 1-0. Каждый набор покрытия соответствует определенной элементарной конъюнкции. Так, набор 1-1 соответствует элементарной конъюнкции $x_2\, x_0$, а набор 1-0 – $x_2\, \overline{x_0}$. Все элементарные конъюнкции одного покрытия объединяются с помощью дизъюнкции. Таким образом, получаем ЛФ, соответствующую покрытию $П_1$: $f(x) = x_2\, x_0 + x_2\, \overline{x_0}$.

Преобразовывая полученные логические выражения для трех покрытий и используя основные тождества и теоремы алгебры логики (см. п. 2), получаем минимизированную функцию $f(x) = x_2$.

Рассуждая по-другому, обратим внимание на то, что последнее покрытие является покрытием минимальной стоимости, и ему соответствует МДНФ. Легко видеть, что этому покрытию соответствует область из четырех клеток, объединяющая единичные значения ЛФ и расположенная в центре карты Вейча.

Тогда при объединении нулевых значений функции максимальная область соответствует 2-кубу (1--), для которого $f(x) = x_2$, или $\bar{f}(x) = \overline{x_2}$.

В итоге в обоих случаях для минимальной ЛФ получено одно и то же выражение.

Следует еще раз отметить, что так как карты Вейча для ЛФ трех и четырех переменных являются объемными фигурами, то при формировании областей на таких картах могут объединяться крайние столбцы и строки, а на карте четырех переменных также четыре угловых клетки.

Приведем еще один способ минимизации ЛФ с помощью *карты Карно*, которая имеет вид таблицы, где наименования столбцов и строк представляют собой значения переменных, причем переменные располагаются в таком порядке, чтобы при переходе к соседнему столбцу или строке изменялось значение только одной переменной.

Таблицу заполняют значениями функции, соответствующими комбинациям значений переменных. На карте Карно отмечают группы, состоящие из 2^n ячеек (n – число переменных) и содержащие 1, т.к. они описываются простыми логическими выражениями. В конечном счете компактное выражение, описывающее функцию, будет представлять собой дизъюнкцию этих логических выражений.

Пример 3. Пусть ЛФ от четырех переменных задана таблицей истинности. Необходимо минимизировать СДНФ.

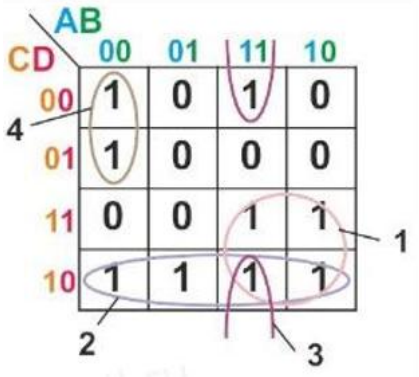
A	B	C	D	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0

A	B	C	D	f
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

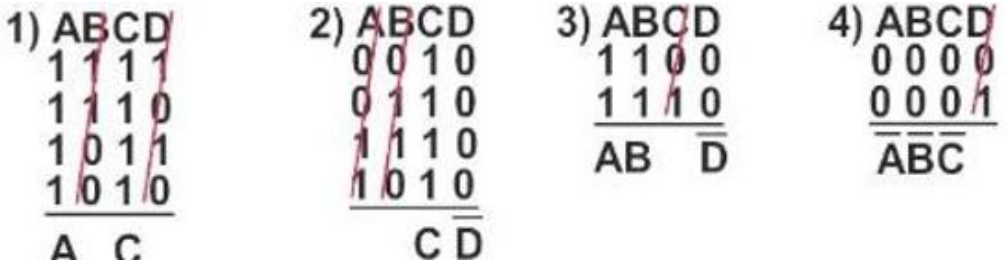
Карта Карно имеет несколько другой вид, отличающийся от диаграммы Вейча.

1) Заполним карту Карно, перенеся из таблицы истинности все минтерны.

2) Сгруппируем области с единицами, учитывая при этом то, что карта Карно – объемная фигура.



3) Для каждой группы исключаем переменные, меняющие свои значения. и получаем минимизированные конъюнкции (переменные в нулевых столбцах записываются с инверсией).



4) Результат записываем как логическую сумму полученных конъюнкций: $f_{\text{МДНФ}} = \text{A C} + \text{C}\,\overline{\text{D}} + \text{A B}\,\overline{\text{D}} + \overline{\text{A}}\,\overline{\text{B}}\,\overline{\text{C}}$. Это есть минимизированная логическая функция, заданная в СДНФ.

Задания к лабораторной работе

В соответствии с выданным преподавателем вариантом (таблица 2)

1) построить СДНФ и СКНФ логической функции;

2) минимизировать СДНФ двумя способами: используя основные тождества и теоремы алгебры логики (п. 2) и диаграмму Вейча (карту Карно) (п 3).

Таблица 2

Варианты индивидуальных заданий

x_2	x_1	x_0	Значения функции f по **вариантам**														
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
0	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	1
0	1	0	1	0	0	1	1	1	1	0	0	0	1	0	0	1	0
0	1	1	0	1	1	0	1	0	0	1	1	0	0	1	1	1	1
1	0	0	1	0	1	0	0	1	0	0	1	1	1	1	0	0	1
1	0	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1	0
1	1	0	1	0	1	0	1	0	1	1	0	1	0	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	0	0	0

Контрольные вопросы:

1. Как может быть представлена логическая функция? Приведите примеры.

2. Назовите две канонические формы представления логической функции.

3. Чем отличается СКНФ от СДНФ?

4. Сформулируйте алгоритм построения логической функции по ее таблице истинности. Приведите пример.

5. Что такое минимизация нормальной формы представления логической функции? Для чего и в каких областях она используется?

6. Как минимизировать СДНФ логической функции с помощью основных тождеств и теорем алгебры логики? Приведите пример.

7. Как минимизировать СДНФ логической функции с помощью диаграмм Вейча (карты Карно)? Приведите пример.

ЛАБОРАТОРНАЯ РАБОТА № 6
ПОСТРОЕНИЕ ЭКВИВАЛЕНТНЫХ СХЕМ

Цель работы: Используя МДНФ, научиться строить функциональные схемы переключательных функций $f(A, B, \ldots, Z)$, эквивалентные заданным.

Краткие теоретические сведения

При проектировании узлов и устройств цифровых вычислительных машин широко используются методы анализа и синтеза логических схем, которые получили название **методов логического проектирования**. Они основаны на использовании алгебры логики или булевой алгебры.

Основным понятием алгебры логики является **высказывание** – любое утверждение, в отношении которого имеет смысл говорить, что оно истинно или ложно. При этом считается, что каждое высказывание не может быть одновременно и истинно, и ложно. Каждое высказывание можно обозначить определенным символом. Запись $X_j = 1$ означает, что высказывание истинно, а $X_j = 0$ – что высказывание ложно.

В алгебре логики высказывания могут быть *простыми* и *сложными*. Высказывание, значение истинности которого не зависит от значений истинности других высказываний, называется **простым.**

При анализе и синтезе логических схем простое высказывание рассматривается как независимая переменная, принимающая два значения: 0 и 1.

Высказывание $Y(X_1, X_2, \ldots, X_n)$, значение истинности которого зависит от значения истинности других высказываний, составляющих его, называется **сложным** и также может принимать два значения: 0 и 1.

При технической реализации переключательных функций переменные $X_1, X_2, \ldots, X_n$ отождествляются с *входящими сигналами*, поступающими на физическую схему, реализующую переключательную функцию, а значение $Y(X_1, X_2, \ldots, X_n)$ представляет собой *выходной сигнал* схемы. Совокупность значений *n* переменных называется **набором.**

Одной из распространенных форм задания переключательных функций является таблица истинности, где переменные $X_1, X_2, \ldots, X_n$ обычно располагаются в порядке возрастания двоичных чисел, образованных набором. Для переключательной функции *n* переменных существует $m = 2^n$ различных наборов, на которые она может принимать значение 0 или 1.

Наиболее распространенной в алгебре логики является функционально полная система логических функций, которая в качестве базовых логических функций использует функцию одной переменной НЕ (функция отрицания) и две функции двух переменных – И (конъюнкция, или логическое умножение) и ИЛИ (дизъюнкция, или логическое сложение). Эта система получила название **система булевых функций**, или **булевый базис**. В алгебре логики имеется целый раздел Алгебра Буля, посвященный этому базису.

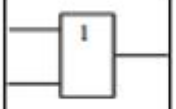
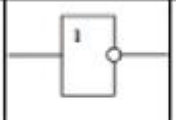
Функции, образованные логическими переменными, можно преобразовывать в соответствии с правилами или законами алгебры логики. При этом стремятся минимизировать логическое выражение, т.е. привести его к виду, удобному для практической реализации на логических элементах.

В электронных цифровых устройствах элементарные логические операции над двоичными переменными реализуются простыми логическими схемами, которые называются **логическими элементами**, или **вентилями**. Число входов логического элемента соответствует числу переменных реализуемой им переключательной функции.

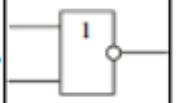
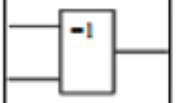
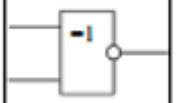
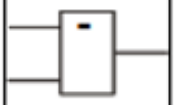
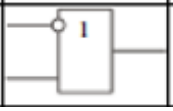
В таблице 1 приведены названия переключательной функции двух переменных и логического элемента, реализующего эту функцию, а также условное обозначение элемента (УГО) структурных и функциональных схем цифровых устройств.

Таблица 1

Переключательные функции двух переменных

Название переключательных функций	Название логического элемента	Другие названия логического элемента	УГО логического элемента	УГО в среде Electronics Workbench
Конъюнкция	Элемент И, схема совпадения	Конъюнктор, клапан И, вентиль И		
Дизъюнкция	Элемент ИЛИ, схема разделения	Дизъюнктор, клапан ИЛИ, вентиль ИЛИ		
Инверсия (отрицание)	Элемент НЕ, схема отрицания	Инвертор		

Окончание таблицы 1

Название переключательных функций	Название логического элемента	Другие названия логического элемента	УГО логического элемента	УГО в среде Electronics Workbench
Операция Пирса	Элемент ИЛИ–НЕ	Схема НЕ–ИЛИ, клапан НЕ–ИЛИ, вентиль НЕ–ИЛИ, схема НИ		
Операция Шеффера	Элемент И–НЕ	Схема НЕ–И, клапан НЕ–И, вентиль НЕ–И		
Сумма по mod 2	ИЛИ–ИЛИ, схема сложения по mod 2	Исключающая ИЛИ		
Сумма по mod 2 и инверсия	ИЛИ–ИЛИ-НЕ, схема сложения по mod 2	Исключающая ИЛИ-НЕ		
Операция запрета	Элемент НЕТ, схема запрета	«Запрет»		–
Логическая равнозначность (эквивалентность)	Схема логической равнозначности	Эквивалентность		–
Импликация	Схема импликации	Импликатор		–

Основное требование, предъявляемое к функционально полному набору логических элементов, состоит в том, чтобы с помощью этого набора можно было построить любую сложную логическую схему. Ввиду того, что законы функционирования элементов однозначно описываются переключательными функциями, применяя операцию суперпозиции, можно получить любую, сколь угодно сложную переключательную функцию.

Однако наборы логических элементов и соответствующие им наборы переключательных функций, как правило, обладают функциональной избыточностью. Таким, например, является широко используемый для построения логических схем цифровых устройств набор, состоящий из переключательных функций конъюнкции; дизъюнкции и отрицания, кото-

рый реализуется логическими элементами И, ИЛИ, НЕ соответственно. Этот набор элементов дает возможность достаточно гибко и экономично строить схемы, например, на полупроводниковых приборах. Кроме того, с помощью этого набора функций наиболее просто перейти от широко распространенной записи переключательной функции в канонической форме к структурной схеме на логических элементах И, ИЛИ, НЕ.

Задача логического проектирования на первом этапе полностью эквивалентна математической задаче представления заданной переключательной функции переключательными функциями выбранной функционально полной системы. При проектировании логических схем сначала необходимо записать переключательную функцию в определенной исходной форме: совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ). Однако эти формы, как правило, достаточно сложные. Поэтому их минимизируют или с помощью диаграмм Вейча (карт Карно), или путем преобразования выражений для переключательных функций с помощью формул и тождеств алгебры логики, приведенных в таблице 2.

Таблица 2

Основные теоремы и тождества алгебры логики

Название тождества	Тождество	
	для дизъюнкции	для конъюнкции
Элементарное высказывание	$X \vee X \vee \ldots \vee X = X$	$X \cdot X \cdot \ldots \cdot X = X$
-“-	$X \vee 0 = X$	$X \cdot 0 = 0$
-“-	$X \vee 1 = 1$	$X \cdot 1 = X$
-“-	$X \vee \overline{X} = 1$	$X \cdot \overline{X} = 0$
-“-	$\overline{\overline{X}} \vee 0 = \overline{\overline{X}} = X$	$\overline{\overline{X}} \cdot 1 = \overline{\overline{X}} = X$
Сочетательные (ассоциативные)	$(X_i \vee X_k) \vee X_j = X_i \vee (X_k \vee X_j)$	$(X_i \cdot X_k) \cdot X_j = X_i \cdot (X_k \cdot X_j)$
Переместительные (коммуника-	$X_i \vee X_k = X_k \vee X_i$	$X_i \cdot X_k = X_k \cdot X_i$
Распределительные (дистрибутивные)	$(X_i \vee X_k) \cdot (X_i \vee X_j) = X_i \vee X_k \cdot X_j$	$X_i \cdot (X_k \vee X_j) = X_i \cdot X_k \vee X_i \cdot X_j$

Окончание таблицы 2

Название тождества	Тождество для дизъюнкции	Тождество для конъюнкции
Поглощения	$X_i(X_i \vee X_k)=X_i$	$X_i \vee X_i \cdot X_k = X_i$
Склеивания	$(X_i \vee X_k)\cdot(X_i \vee \overline{X_k})=X_i$	$X_i \cdot X_k \vee X_i \cdot \overline{X_k} = X_i$
Соотношение двойственности (формула де Моргана)	$\overline{X_1 \vee X_2 \vee \ldots \vee X_n} = \overline{X_1} \cdot \overline{X_2} \cdot \ldots \cdot \overline{X_n}$	$\overline{X_1 \cdot X_2 \cdot \ldots \cdot X_n} = \overline{X_1} \vee \overline{X_2} \vee \ldots \vee \overline{X_n}$

Пример 1. По таблице истинности построить логическую функцию, минимизировать ее и построить эквивалентную схему.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Для выполнения задания будем придерживаться следующего алгоритма.

1. Выделить в таблице истинности те строки, в которых значение функции равно 1.

2. Выписать искомую формулу в виде дизъюнкции нескольких логических элементов. Число этих элементов равно числу выделенных строк.

3. Каждый логический элемент в этой дизъюнкции записать в виде конъюнкции аргументов функции.

4. Если значение какого-либо аргумента функции в соответствующей строке таблице равно 0, то этот аргумент взять с отрицанием.

5. Минимизировать полученное логическое выражение любым подходящим для этого способом.

6. Построить функциональную схему по МДНФ.

7. Убедиться в эквивалентности функциональных схем путем сравнения их таблиц истинности.

Применим предложенный алгоритм.

1. В 2, 4 и 6 строках таблицы истинности значение функции равно 1.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

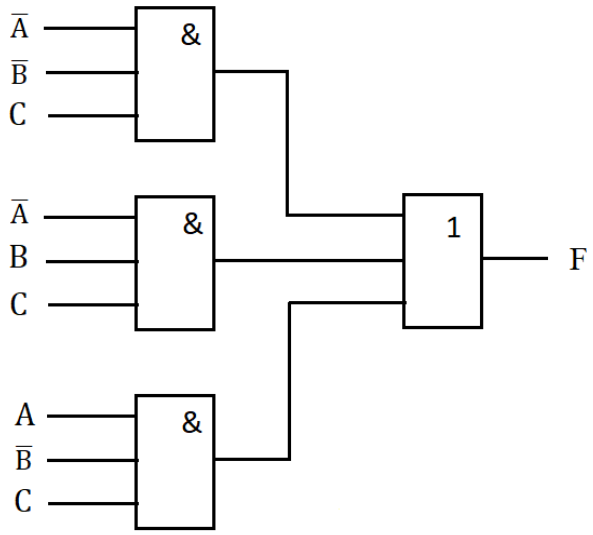
2. Так как строки три, получаем дизъюнкцию трех элементов: ( ) + ( ) + ( ).

3. Каждый логический элемент в этой дизъюнкции запишем в виде конъюнкции аргументов функции A, B и C.

4. Берем аргумент с отрицанием, если его значение в соответствующей строке таблицы равно 0, и получаем искомую функцию:

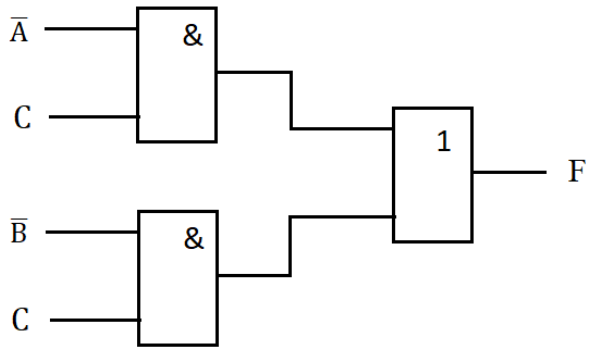
$F = (\overline{A}\,\overline{B}\,C) + (\overline{A}\,B\,C) + (A\,\overline{B}\,C)$

Ее функциональная схема после построения (см. Лабораторную работу № 4) будет выглядеть следующим образом.



5. Проведем минимизацию логической функции с целью построения упрощенной эквивалентной схемы. Для этого выполним склеивание конъюнкций $(\overline{A}\,\overline{B}\,C)$ и $(\overline{A}\,B\,C)$ по переменной B и конъюнкций $(\overline{A}\,\overline{B}\,C)$ и $(A\,\overline{B}\,C)$ по переменной A. В результате функция F преобразуется к виду $F = (\overline{A}\,C) + (\overline{B}\,C)$.

6. После построения ее функциональная схема будет выглядеть следующим образом:



Сравнивая две функциональные схемы можно сделать вывод о том, что вторая схема имеет меньшую цену *C*, которая определяется суммарным числом входов логических элементов, а также самим числом логических элементов.

7. Эквивалентность схем проверяется путем сравнения таблиц истинности. Построим таблицу истинности для минимизированной функции (см. Лабораторную работу №4).

A	**B**	**C**	$\overline{A}$	$\overline{B}$	$\overline{A}$ **C**	$\overline{B}$ **C**	**F**
0	0	0	1	1	0	0	0
0	0	1	1	1	1	1	1
0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	0	0	0
1	0	1	0	1	0	1	1
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

В данном случае таблицы истинности совпадают, следовательно, функциональные схемы эквивалентны.

Задания к лабораторной работе

1. В соответствии с вариантом задания записать логические выражения для переключательных функций в СДНФ.

2. Минимизировать полученные в предыдущем пункте логические выражения для переключательных функций с помощью диаграмм Вейча (карт Карно) и с помощью формул и тождеств алгебры логики.

3. На базе логических элементов И, ИЛИ, НЕ построить две функциональные схемы, реализующие заданные логические выражения до и после минимизации.

3. Доказать эквивалентность схем.

5. Провести анализ и оценку построенной эквивалентной функциональной схемы.

Варианты индивидуальных заданий

1. $F = \overline{(A \cdot B + \overline{A} \cdot C)} \cdot \overline{C}$
2. $F = (A + \overline{C}) \cdot (\overline{A} + B) \cdot \overline{B}$
3. $F = A \cdot \overline{B} + A \cdot \overline{B + C}$
4. $F = \overline{A} \cdot B + \overline{A \cdot \overline{B} \cdot C}$
5. $F = \overline{\overline{A} \cdot B} + A \cdot \overline{B} + C$
6. $F = (A + \overline{C}) \cdot (\overline{A} + \overline{B}) \cdot C$
7. $F = \left(\overline{\overline{A} + B \cdot D}\right) + (\overline{D} + A)$
8. $F = \overline{C} \cdot \overline{B \cdot \overline{A}} + \overline{B} \cdot A$
9. $F = \overline{\overline{B \cdot A} + B \cdot C \cdot \overline{A}}$
10. $F = \overline{\overline{A} \cdot \overline{C}} \cdot B + \overline{B} \cdot C$
11. $F = \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B}$
12. $F = \overline{(\overline{A + B}) \cdot (\overline{A + C} + C)}$
13. $F = \overline{A} \cdot \overline{C} \cdot \overline{B} + A \cdot B$
14. $F = \overline{A \cdot \overline{B} + C \cdot \overline{A}} + B$

Контрольные вопросы:

1. Дайте определение понятиям: функция переключения, логический элемент компьютера, логическая схема, таблица истинности. Приведите примеры.

2. Какие требование предъявляются к функционально полному набору логических элементов?

3. Дайте краткую характеристику логическому базису И, ИЛИ, НЕ.

4. Назовите основные логические операции и сопоставьте им логические элементы функциональных схем компьютера. Ответ поясните.

5. Как осуществляется проектирование цифровых логических схем?

6. С какой целью проводится минимизация логических выражений переключательных функций?

7. Перечислите последовательно этапы построения цифровой логической схемы, эквивалентной заданной. Приведите пример.

8. Что такое цена цифровой логической схемы? Как ее определить? Приведите пример.

ЛАБОРАТОРНАЯ РАБОТА № 7
МОДЕЛИРОВАНИЕ РАБОТЫ ОСНОВНЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Цель работы: Исследовать логические схемы, реализующие работу основных логических элементов «И», «И-НЕ», «ИЛИ», «ИЛИ-НЕ», «Исключающее ИЛИ».

Краткие теоретические сведения

В настоящее время широкое распространение получило компьютерное проектирование и анализ цифровых устройств в среде схемотехнического моделирования *Multisim.*

Особенностью программной среды *Multisim* является наличие в ее библиотеке более 16 000 электронных компонентов, а также виртуальных контрольно-измерительных приборов, которые по характеристикам приближены к их промышленным аналогам.

Среда моделирования *Multisim* имитирует реальное рабочее место в исследовательской лаборатории, которое оборудовано измерительными приборами: генераторами, мультиметрами, осциллографами, анализатором спектра, измерителем АЧХ и ФЧХ, измерителем нелинейных искажений, преобразователем и анализатором логических сигналов и др.

Multisim – программа с многооконным графическим интерфейсом, позволяющим строить и редактировать схемы, модели и изображения компонентов, а также представлять результаты расчетов в удобном графическом виде.

Пользовательский интерфейс программы проиллюстрирован на рисунке 1. Он состоит из следующих элементов: строка меню, панель инструментов, панель разработки, окно редактирования, приборная панель.

Пользовательский интерфейс можно настроить с учетом любых предпочтений. Панели инструментов можно закрепить в любом месте и изменить их форму. Инструменты всех панелей также можно изменять и создавать новые панели. Система меню полностью настраивается, вплоть до контекстных меню разных объектов.

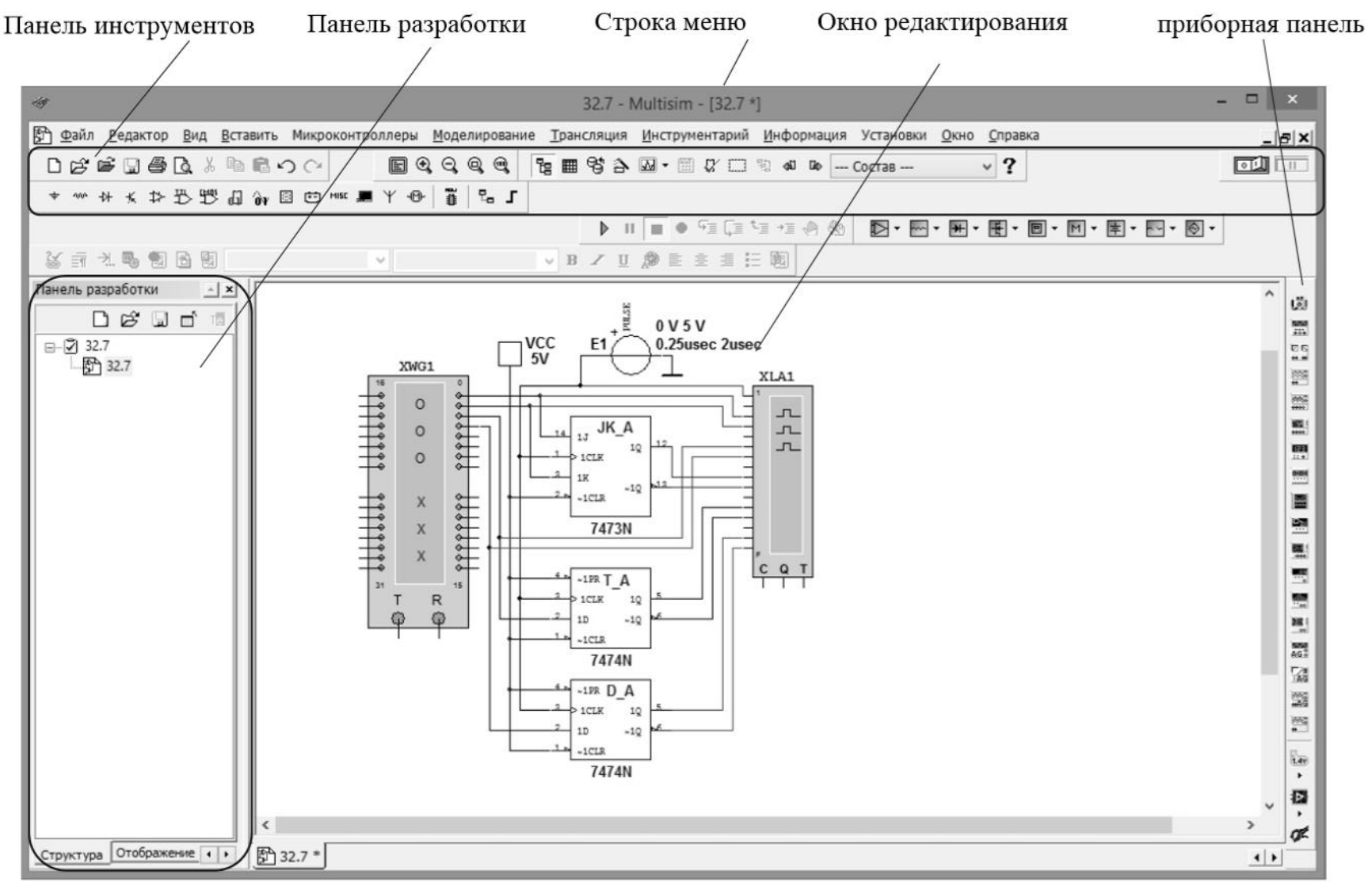


Рисунок 1 – Интерфейс программы Multisim

Рассмотрим подробнее некоторые модели контрольно-измерительных приборов, которые будем использовать в процессе моделирования цифровых логических схем.

1. ***Контрольно-измерительные приборы***

Генератор слов (**Word Generator**) предназначен для генерации до 8192 32-разрядных двоичных слов. Внешний вид и лицевая панель его показаны на рисунке 2. Ввод генерируемых слов производится в буфере ввода. Формат отображения кодовых слов выбирается с помощью группы кнопок *Display* (<*Hex*> – шестнадцатеричный; <*Dec*> – десятичный; <*Binary*> – двоичный; <*ASCII*> – символьный код). Частота генерации кодовых слов задается в окне *Frequency* (*Частота*) и лежит в диапазоне от 1 Гц до 1000 МГц. В процессе работы на каждом выводе генератора появляется логический уровень согласно разряду двоичного кодового слова, при этом генератор работает в трех режимах:

• *Step* (*Пошаговый*) – каждый раз при подаче очередного слова на выход моделирование останавливается;

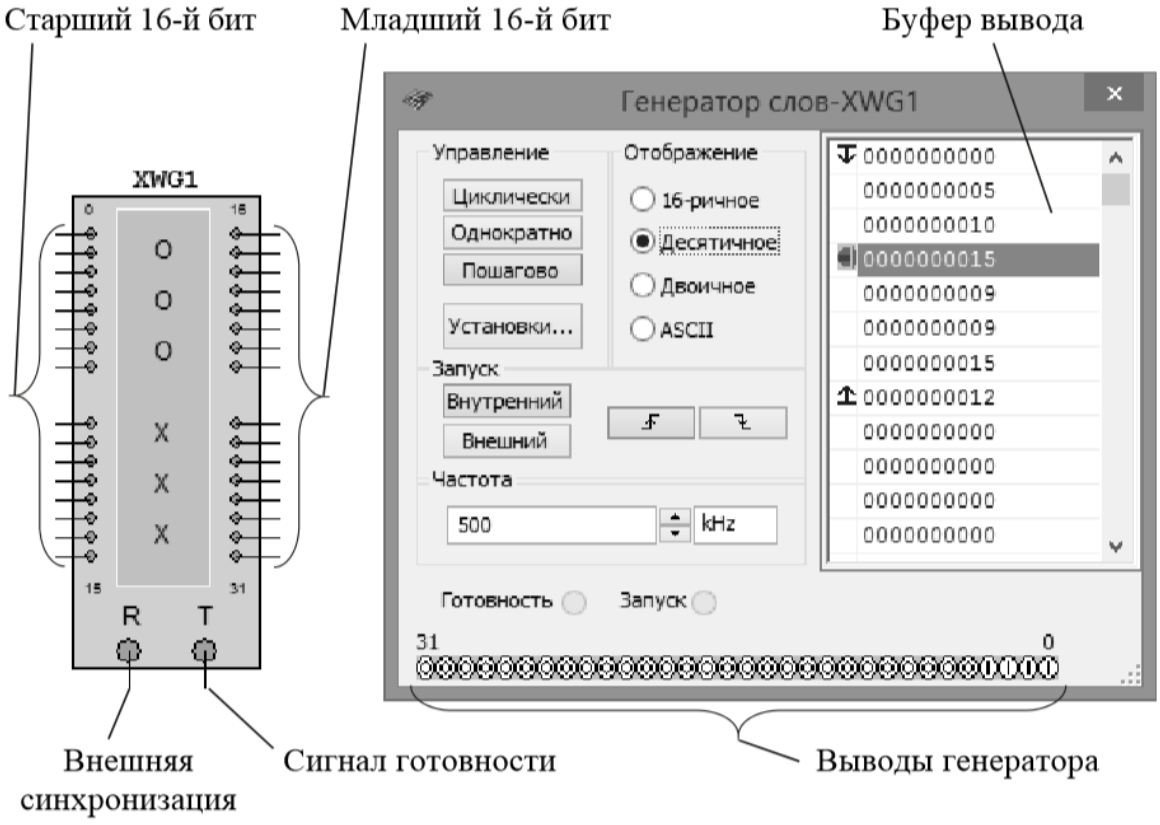


Рисунок 2 – Генератор слов

• *Burst* (*Пакетный*) – генерируется последовательность кодовых слов начиная с начальной позиции и заканчивая конечной позицией, моделирование останавливается при достижении конечной позиции;

• *Cycle* (*Циклический*) – на выводах генератора последовательно появляются логические уровни согласно комбинации слов, генерирование осуществляется до тех пор, пока не будет остановлено моделирование или достигнута точка прерывания (Breakpoint).

Во время моделирования курсор ► в окне «Буфер вывода» указывает на текущее генерируемое слово. Остановив моделирование, можно изменить положение курсора, начальную позицию, конечную позицию, а также точку прерывания. При нажатии кнопки «Установки» открывается диалоговое окно свойств буфера (рис. 3):

• Без изменений – оставить без изменений;

• Загрузить – загрузить кодовые слова из файла;

• Сохранить – сохранить кодовые слова в файл;

• Очистить буфер – обнулить содержимое буфера;

• Вверх – заполнить буфер кодовыми словами начиная с кода, указанного в поле Инициализировать конфигурацию (по умолчанию 0×0000), с последующим увеличением на 1 в каждой следующей строке;

• Вниз – заполнить буфер кодовыми словами начиная с кода, указанного в поле Инициализировать конфигурацию (по умолчанию 0×0400), с последующим уменьшением на 1 в каждой следующей строке;

• Вправо – заполнить буфер кодовыми словами начиная с кода, указанного в поле Инициализировать конфигурацию (по умолчанию 0×80000000), с последующим двоичным сдвигом вправо на 1 разряд в каждой следующей строке;

• Влево – заполнить буфер кодовыми словами начиная с кода, указанного в поле Инициализировать конфигурацию (по умолчанию 0×0001), с последующим двоичным сдвигом влево на 1 разряд в каждой следующей строке.

Запуск генератора может синхронизироваться как внутренним (Internal), так и внешним (External) сигналом синхронизации. На вывод Ready подается сигнал готовности.

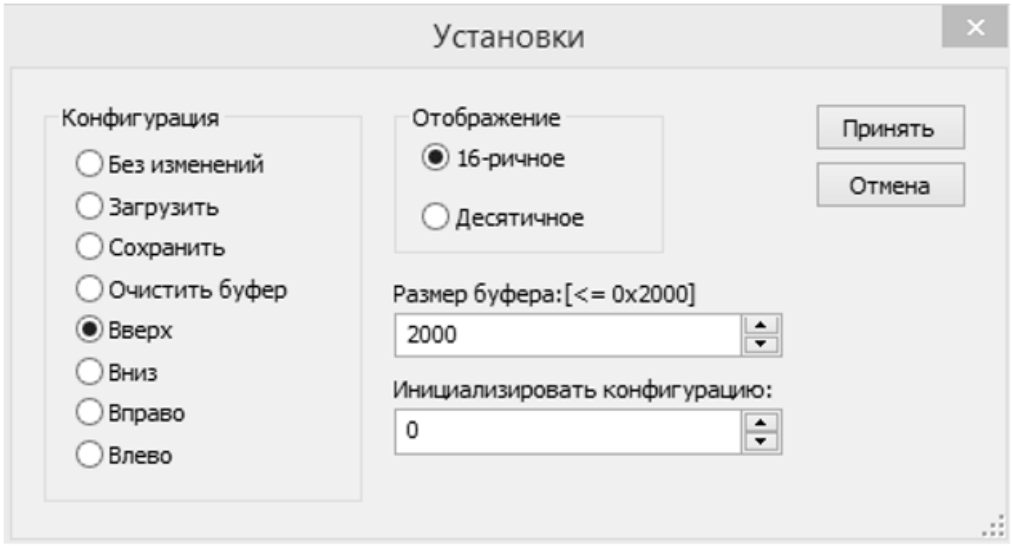


Рисунок 3 – Окно свойств буфера

Логический анализатор (Logic Analyzer). Логический анализатор (ЛА) – устройство, предназначенное для диагностики цифровых схем. ЛА позволяет отслеживать и записывать состояния логических элементов цифровых электронных устройств, анализировать и визуализировать их. Внешний вид и лицевая панель устройства показаны на рисунке 4.

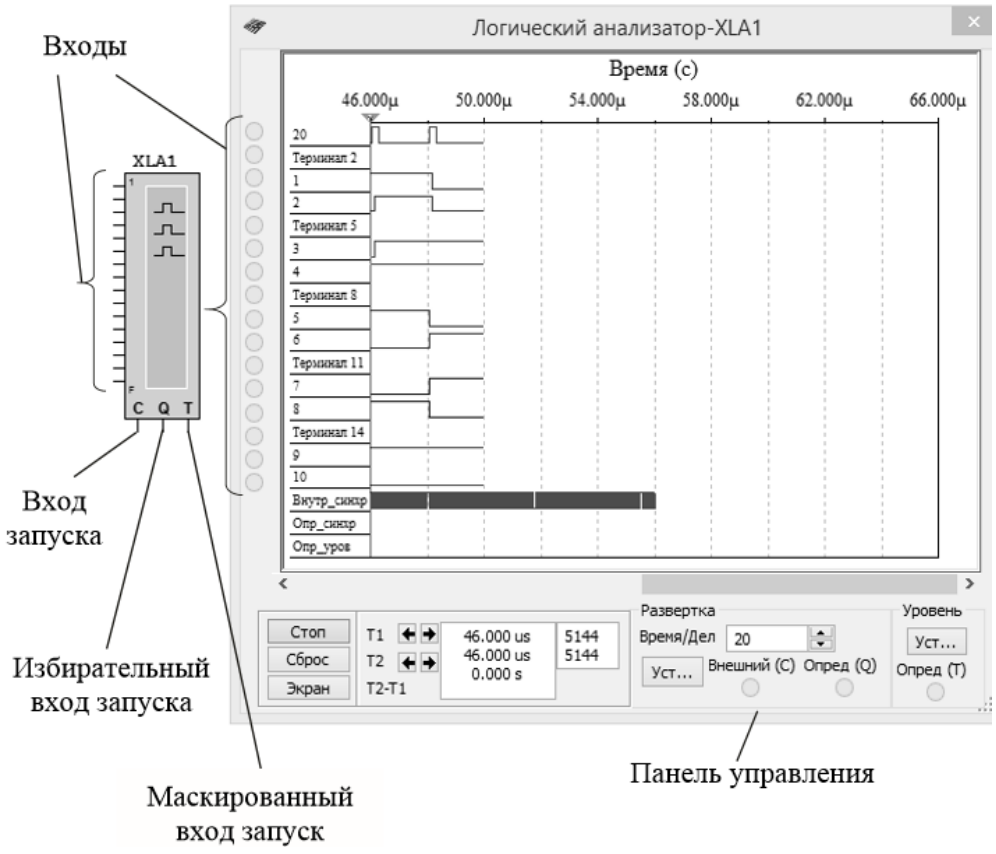


Рисунок 4 – Логический анализатор

Логический анализатор имеет 16 каналов для съема сигналов, а также несколько входов запуска. Кроме этого, прибор снабжен двумя курсорами, позволяющими проводить измерения во временной области. Если вход 1 считать младшим разрядом, а вход 16 – старшим, то состояние всех входов может быть представлено 16-разрядным двоичным кодом. Код, соответствующий позиции курсора, отображается в поле *Входной код* (рис. 5).

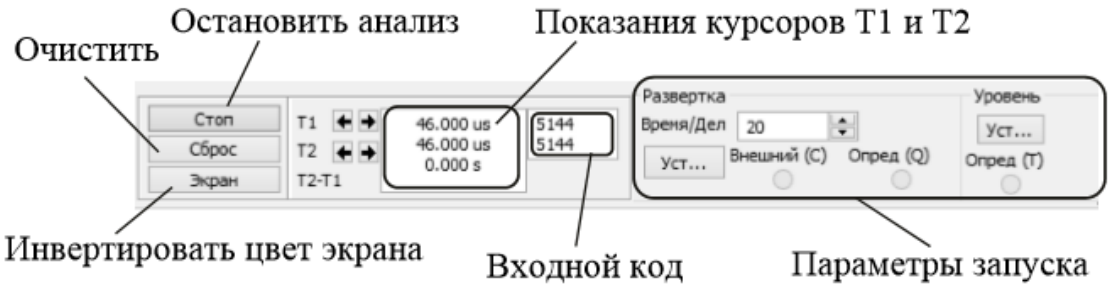


Рисунок 5 – Панель управления логического анализатора

При нажатии кнопки *Установки* в группе *Развертка* (тактовый генератор) открывается диалоговое окно настройки параметров тактирования входных сигналов (рис. 6).

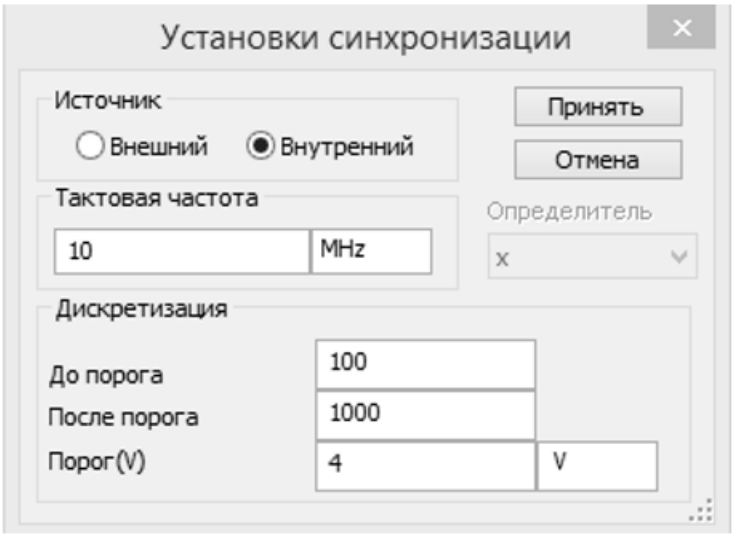


Рисунок 6 – Настройка параметров синхронизации

Тактирование сигналов осуществляется с использованием внешнего (External) или внутреннего (Internal) источников. В поле *Clock Qualifier (Определитель)* устанавливается активный уровень сигнала синхронизации. В поле *Clock Rate (Тактовая чистота)* устанавливается частота вы-

борки анализатора. В группе *Sampling Setting (Дискретизация)* задаются параметры выборки сигналов:

• *Pre-trigger Samples (До порога)* – сбор данных производится до поступления импульса запуска;

• *Post-trigger Samples (После порога)* – сбор данных начинается после поступления импульса запуска и продолжается до тех пор, пока не будет набрано заданное количество отсчетов;

• *Threshold Volt (Порог)* – пороговая величина.

Дополнительные условия запуска анализатора осуществляются с помощью диалогового окна *Trigger Settings*.

В данном окне настраивается маска, по которой осуществляются фильтрация логических уровней и синхронизация входных каналов.

2. ***Компоненты Multisim***

Компоненты – это основа любой схемы, т. е. элементная база, из которой состоит схема.

В *Multisim* работа осуществляется с двумя категориями компонентов: *виртуальными (virtual)* и *реальными (real)*.

Реальные компоненты являются полными аналогами компонентов, выпускаемых или выпущенных радиоэлектронной промышленностью. Виртуальные компоненты – математические модели семейств (Family) компонентов (резисторы, конденсаторы и т. д.) с любыми произвольными параметрами, присущими данной категории.

Добавление компонентов в схему происходит из меню ***Place*** либо панели инструментов ***Components***.

Рассмотрим основные группы компонентов базы данных *Multisim*.

Группа **Sources (Источники)**.

В данной группе содержатся модели источников питания (однофазный источник питания постоянного (DC_POWER) и переменного напряжения (AC_POWER), трехфазные источники питания, источники питания постоянного тока (VCC1, VDD2, VEE3, VSS4), а также заземление (GROUND)), источников напряжения (источник прямоугольного сигнала, кусочнолинейного сигнала (PWL Voltage) и др.), источников тока и др.

Группа **Basic (Базовые компоненты)**.

В базовую группу входят модели резисторов, конденсаторов, индуктивностей, трансформаторов, виртуальных механических ключей и др.

Группа **Diodes (Диоды)**.

В этой группе содержатся модели таких компонентов, как диод, стабилитрон (zener), светодиод, диодный мост (FWB), диод Шоттки, тиристор и др.

Группа **Transistors (Транзисторы)**.

В данной группе находятся модели биполярных транзисторов (BJT), полевых транзисторов (JFET), МОП-транзисторов и др.

Группа **Analog (Аналоговые компоненты)** содержит модели операционных усилителей (OPAMP), компараторов (COMPARATOR) и др.

Группа **TTL (цифровые микросхемы по технологии ТТЛ)**. В данной группе содержатся модели микросхем серий 74Sxx, 74LSxx, 74ALSxx и др.

Группа **CMOS (цифровые микросхемы по технологии КМОП)** содержит модели микросхем серии 74HCxx, NC7Sx (Tiny Logic) и др.

Группа **Misc Digital (Цифровые устройства)** включает виртуальные модели цифровых устройств (**TIL**) (логические элементы, триггеры, регистры, счетчики, мультиплексоры, декодеры, элементы арифметико-логических устройств и др.), микросхемы цифровой обработки сигналов (**DSP**), программируемые логические интегральные схемы, микросхемы памяти, микроконтроллеры и др.

Группа **Indicators (Индикаторные устройства)** включает следующие модели компонентов: индикаторы напряжения и тока, логические пробники (лампочки), семисегментные индикаторы, звуковые индикаторы и др.

Порядок выполнения работы

Запустите среду разработки *Multisim* и соберите на рабочем поле среды схему, приведенную на рисунке 7, для испытания основных и базовых логических элементов, установите в диалоговых окнах компонентов их параметры или режимы работы.

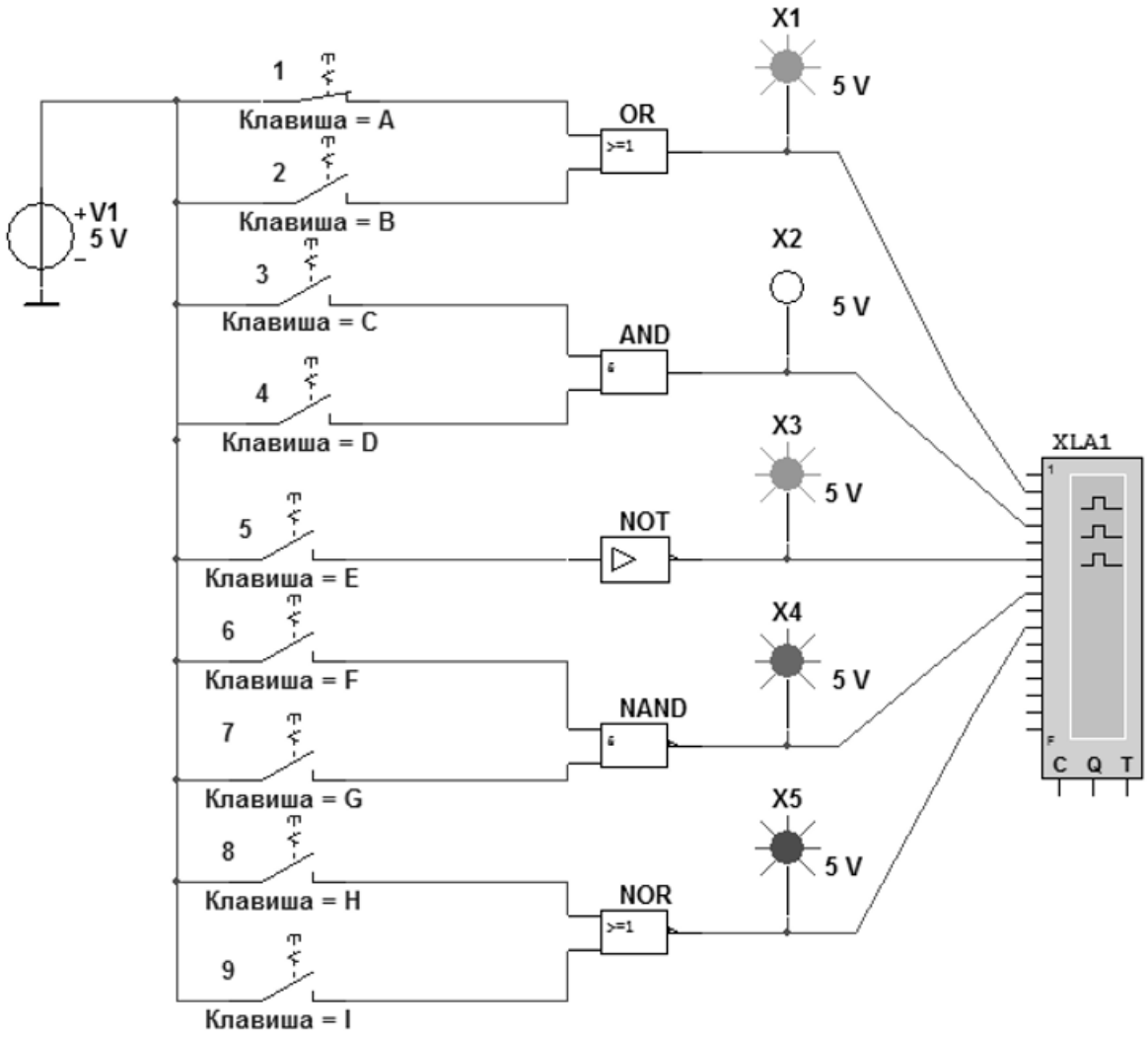


Рисунок 7 – Схема исследования логических элементов

Схема исследования собрана на двоичных основных [**OR (ИЛИ)**, **AND (И)** и **NOT (НЕ)**] и универсальных (базовых) [**NAND (И-НЕ)** и **NOR (ИЛИ-НЕ)**] логических элементах, расположенных в библиотеке **Misc Digital/TIL (Цифровые микросхемы/TIL)**, с уровнем высокого постоянного напряжения 5В. В схему включены ключи **1, 2, ..., 9**, пробники (лампочки) **X1, X2, …, X5** с пороговыми напряжениями 5В, генератор постоянных токов **V1** с напряжением V = 5 В и логический анализатор **XLA1**.

Для удобства сравнения сигналов выходы логических элементов подключены к входам **2, 4, 6, 8** и **10** анализатора **XLA1** (рис. 8).

При моделировании происходит медленная развертка временных диаграмм в окне анализатора. По достижении интервала времени, равного 70–80% ширины окна, следует посредством кнопки <*Run/Stop*> выключать процесс моделирования.

Оперируя ключами **1, 2, …, 9**, сформируйте все возможные комбинации аргументов x_1 и x_2 (00, 10, 01 и 11) на входе дизъюнктора (**OR**), конъюнктора (**AND**), штриха Шеффера (**NAND**) и стрелки Пирса (**NOR**) и запишите значения выходных логических функций y_k (0 или 1) в подготовленную таблицу.

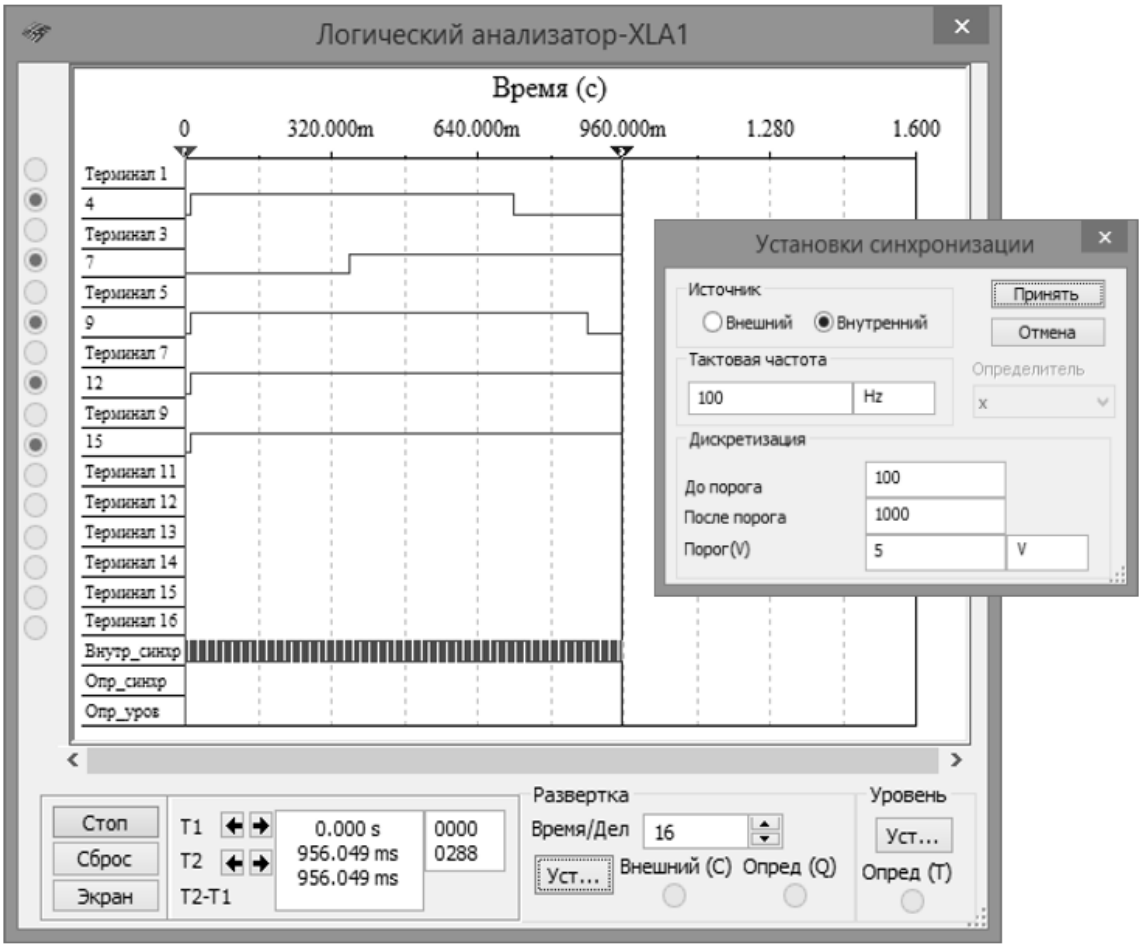


Рисунок 8 – Временная развертка сигналов исследуемой схемы

Заметим, что если ключ замкнут, то на этот вход элемента будет подана логическая единица (положительный потенциал 5 В), а при разомкнутом ключе – логический ноль.

Поскольку инвертор (**NOT**) имеет один вход, то для формирования двух значений входного сигнала (логической единицы или логического нуля) достаточно одного ключа **5**.

Значения функций исследуемых элементов можно контролировать с помощью пробников (индикаторов) **X1, X2, …, X5**: если выходной сигнал элемента равен логической единице, то включенный на выходе этого элемента пробник светится.

Если положение ключей схемы будет такое же, как на рисунке 7, тогда функции элементов **OR**, **AND** и **NOR** равны логической единице.

Составьте отчет о проделанной работе, в котором будут присутствовать изображения исследуемых логических схем, временная развертка выходных и входных сигналов, полученная посредством логического анализатора, таблицы со значениями входных комбинаций и сигнала на выходе для каждого логического элемента.

Сравните диаграммы входных и выходных сигналов с таблицей истинности для каждой из моделируемых логических функций.

Сделайте выводы.

Контрольные вопросы:

1. Дайте краткую характеристику среде моделирования *Multisim.*

2. Перечислите основные компоненты среды моделирования *Multisim.*

3. Какие функциональные схемы называют комбинационными?

4. Для чего используется генератор слов *Word Generation* в электронной среде моделирования *Multisim*?

5. Для чего используется логический анализатор *Logic Analizer* в электронной среде моделирования *Multisim*?

6. Как обозначаются логические элементы на схемах электрических функциональных и принципиальных?

7. Дайте характеристику основным и базовым логическим элементам.

ЛАБОРАТОРНАЯ РАБОТА № 8
ИССЛЕДОВАНИЕ ЗАПОМИНАЮЩИХ ЭЛЕМЕНТОВ И УЗЛОВ

Цель работы: Научиться моделировать работу триггерных регистров, строить диаграмму их выходных сигналов и по ней определять состояние регистра сдвига в определенный момент времени. Научиться моделировать работу суммирующего и вычитающего счетчиков.

Краткие теоретические сведения

1. *Триггеры*

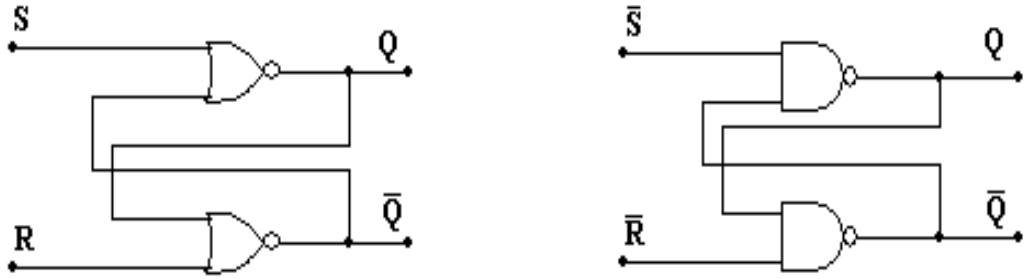
Триггерами называются устройства, обладающие двумя устойчивыми состояниями (Q = 1 и Q = 0) и способные находиться в одном из них сколь угодно долго и переходить из одного состояния в другое под воздействием внешних сигналов.

В каком из этих состояний окажется триггер, зависит от сигналов на входах триггера и от его предыдущего состояния, т.е. он имеет память. Таким образом, **триггер** – элементарная ячейка памяти.

Тип триггера определяется алгоритмом его работы, в зависимости от которого триггер может иметь установочные, информационные и управляющие входы. Установочные входы обусловливают состояние триггера независимо от состояния других входов. Входы управления разрешают запись данных, подающихся на информационные входы. Наиболее распространенными являются триггеры RS-, JK-, D- и T-типов.

RS-триггер – простейший автомат с памятью, который может находиться в двух состояниях. Триггер имеет два установочных входа: установки S (set – установка) и сброса R (reset – сброс), на которые подаются входные сигналы от внешних источников. При подаче на установки активного логического уровня триггер устанавливается в единицу (Q = 1, Q′ = 0, здесь штрих означает инвертирование), при подаче активного уровня на вход сброса триггер устанавливается в ноль (Q = 0, Q′ = 1). Если на оба установочных входа подать пассивный логический уровень, то триггер сохраняет предыдущее состояние выходов: Q = 1 или Q = 0. Каждое состояние устойчиво и поддерживается за счет действия обратных связей. Подача активного уровня одновременно на оба установочных входа запрещена, так как триггер не может быть установлен в ноль и единицу.

RS-триггер может быть выполнен на элементах «ИЛИ-НЕ» (рис. 1, *а*) или «И-НЕ» (рис. 1, *б*).



а *б*

Рисунок 1 – RS-триггер: а – на элементах «ИЛИ-НЕ», б – на элементах «И-НЕ»

Для *RS*-триггеров, выполненных на элементах «ИЛИ-НЕ», активным уровнем на управляющих входах является уровень логической единицы, а на элементах «И-НЕ» – уровень логического нуля.

RS-триггер – основной узел построения последовательных схем. Условия переходов триггеров из одного состояния в другое можно описать табличным, аналитическим или графическим способами.

Табличное описание работы *RS*-триггера на элементах «ИЛИ-НЕ» и «И-НЕ» представлено в таблице 1, где Q_t – предшествующее состояние выхода; Q_{t+1} – новое состояние, устанавливающееся после перехода; «-» – неопределенное состояние.

Таблица 1

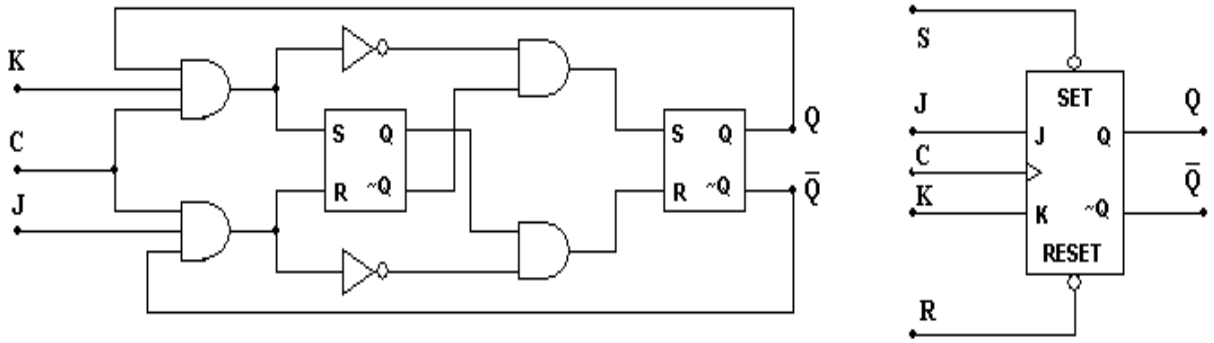
Состояния *RS*-триггера на элементах «ИЛИ-НЕ» и «И-НЕ»

ИЛИ-НЕ			И-НЕ		
R	S	Q_{t+1}	R	S	Q_{t+1}
0	0	Q_t	0	0	–
1	0	0	1	0	1
0	1	1	0	1	0
1	1	–	1	1	Q_t

Триггер *JK*-типа имеет более сложную структуру и более широкие возможности по сравнению в *RS*-триггером. Кроме информационных входов *J* и *K* и прямого и инверсного выходов *Q* и *Q'*, *JK*-триггер имеет вход управ-

ления *C* (тактирующий или счетный) и два асинхронных установочных входа *R* и *S*. Обычно активными уровнями установочных сигналов являются нули. Установочные входы имеют приоритет над остальными входами. Активный уровень сигнала на входе *S* устанавливает триггер в состояние единица ($Q = 1$), а на входе *R* – в состояние ноль ($Q = 0$) независимо от сигналов на остальных входах. Если на входы установки подать пассивный уровень сигнала, то состояние триггера будет изменяться по фронту импульса на счетном входе в зависимости от состояния входов *J* и *K*.

Один из вариантов функциональной схемы *JK*-триггера и его условное графическое изображение приведены на рисунке 2, временные диаграммы его работы при R=S=1 – на рисунке 3.



а *б*

Рисунок 2 – *JK*-триггер: *а* – функциональная схема; *б* – условное графическое обозначение

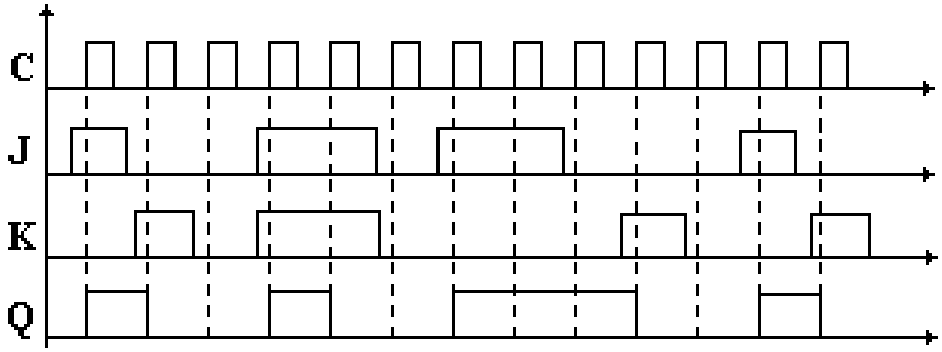
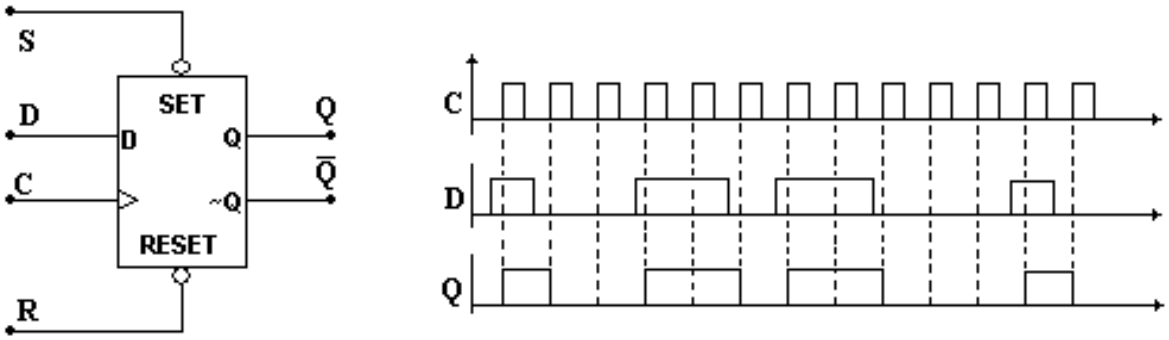


Рисунок 3 – Временная диаграмма работы *JK*-триггера

D-триггер имеет один информационный вход *D* (*data* – данные) и один счетный вход *C*. Информация с входа *D* записывается в триггер по положительному перепаду импульса на счетном входе и сохраняется до следующего положительного перепада. Кроме счетного *C* и информаци-

онного *D* входов, у триггера есть два асинхронных установочных входа *R* и *S*. Установочные входы приоритетные. Активный уровень сигнала на входе *S* устанавливает триггер в состояние единица (*Q*=1), а на входе *R* – в состояние ноль (*Q*=0) независимо от сигналов на остальных входах.

Условное обозначение *D*-триггера с диаграммами входных и выходных сигналов приведено на рисунке 4.

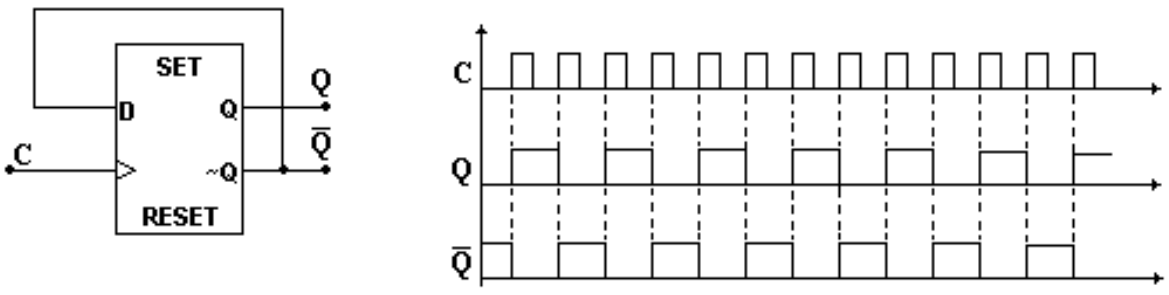


а *б*

Рисунок 4 – *D*-триггер:
а – условное обозначение; *б* – временные диаграммы

Т-триггер (счетный триггер) – устройство, осуществляющее счетный режим. Такие схемы можно построить на основе *JK*- или *D*-триггеров.

В *D*-триггере счетный режим (рис. 5 *а*) реализуется при помощи обратной связи, когда на вход D подается сигнал с инверсного выхода триггера, т.е. всегда осуществляется неравенство сигналов на входе *D* и на выходе *Q* (если *Q*=1, *D*=0 и наоборот). Следовательно, при каждом положительном перепаде сигнала на счетном входе *C* состояние выхода будет изменяться на противоположное.



а *б*

Рисунок 5 – *Т*-триггер: *а* – условное обозначение; *б* – временная диаграмма

Таким образом, на каждые два входных тактовых импульса *T*-триггер формирует один период выходного сигнала *Q*, т.е. период выходного сигнала в два раза больше периода входного сигнала. Следовательно, триггер осуществляет деление частоты $f_т$ на его входе на две: $f_Q = f_т/2$, где f_Q – частота следования импульсов на выходе триггера.

2. *Регистры*

Триггерным регистром называется совокупность триггеров с определенными связями между ними, при которых они действуют как единое устройство.

Регистры выполняются на синхронных триггерах *JK*- или *D*-типа. В зависимости от выполняемых функций регистры делятся на *накопительные (параллельные)* и *сдвигающие*.

В последовательном регистре (рис. 6) выход предыдущего триггера подается на вход следующего триггера, а тактовые импульсы подаются на входы *C* всех триггеров, составляющих регистр, одновременно. При этом содержимое каждого триггера записывается в последующий триггер. Такие регистры называются *сдвиговыми регистрами*, или *регистрами сдвига*.

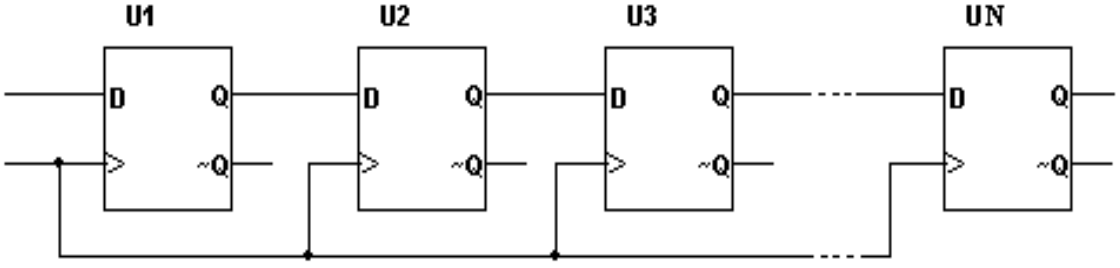


Рисунок 6 – Последовательный регистр (регистр сдвига)

Если на вход *D* регистра сдвига подать единицу, а на вход *C* тактовую частоту, то единица начнет продвигаться по регистру сдвига, т.е. под воздействием первого тактового импульса единица запишется в первый триггер регистра. Под воздействием второго тактового импульса эта единица перепишется во второй триггер и т.д., когда под воздействием *N*-го тактового импульса единица не выйдет из регистра сдвига. Временная диаграмма работы четырехразрядного регистра сдвига приведена на рисунке 7.

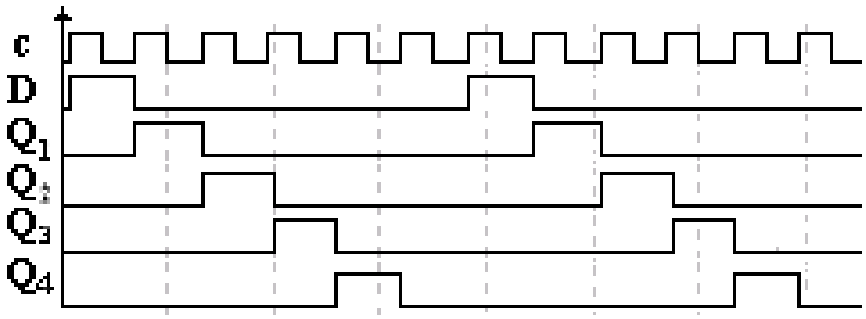


Рисунок 7 – Временная диаграмма четырехразрядного регистра сдвига

Различают следующие виды *регистров сдвига*:

- с последовательным вводом и выводом;
- с последовательным вводом и параллельным выводом;
- с параллельным вводом и последовательным выводом;
- с переменным направлением сдвига (реверсивные регистры сдвига).

Кроме последовательных регистров сдвига существуют параллельные регистры, в которых информация подается одновременно на все N триггеров и считывается одновременно с выходов всех триггеров регистра (рис. 8). Тактовая частота подается одновременно на все триггеры.

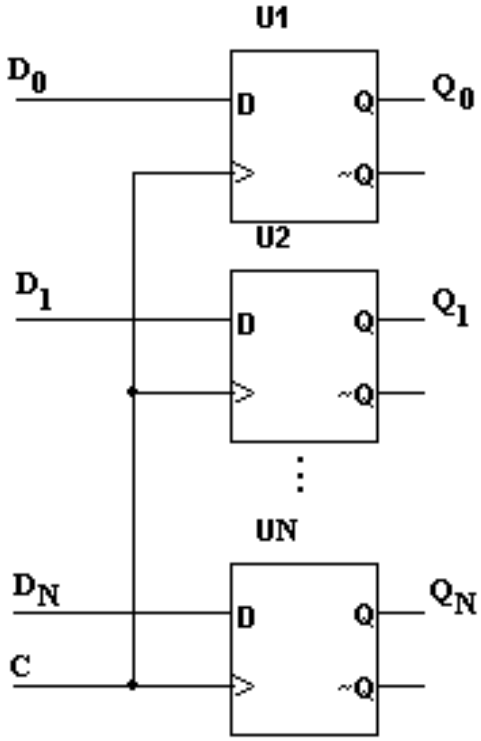


Рисунок 8 – Параллельный регистр

Параллельные регистры используются для хранения двоичной информации небольшого объема в течение короткого промежутка времени.

3. *Счетчики*

Счетчиком называется устройство, подсчитывающее число входных импульсов. Число, представляемое состоянием его выходов по фронту каждого входного импульса, изменяется на единицу. Счетчик состоит из *n* последовательно соединенных счетных триггеров, причем выход одного счетного триггера соединен с тактовым входом следующего триггера.

Счетчики бывают *суммирующими* (прямой счет) и *вычитающими* (обратный счет).

В суммирующих счетчиках каждый входной импульс увеличивает число на его выходах на единицу, в вычитающих счетчиках уменьшает это число на единицу. Для того чтобы построить суммирующий счетчик, необходимо счетный вход очередного триггера подключить к инверсному выходу предыдущего, как показано на рисунке 9.

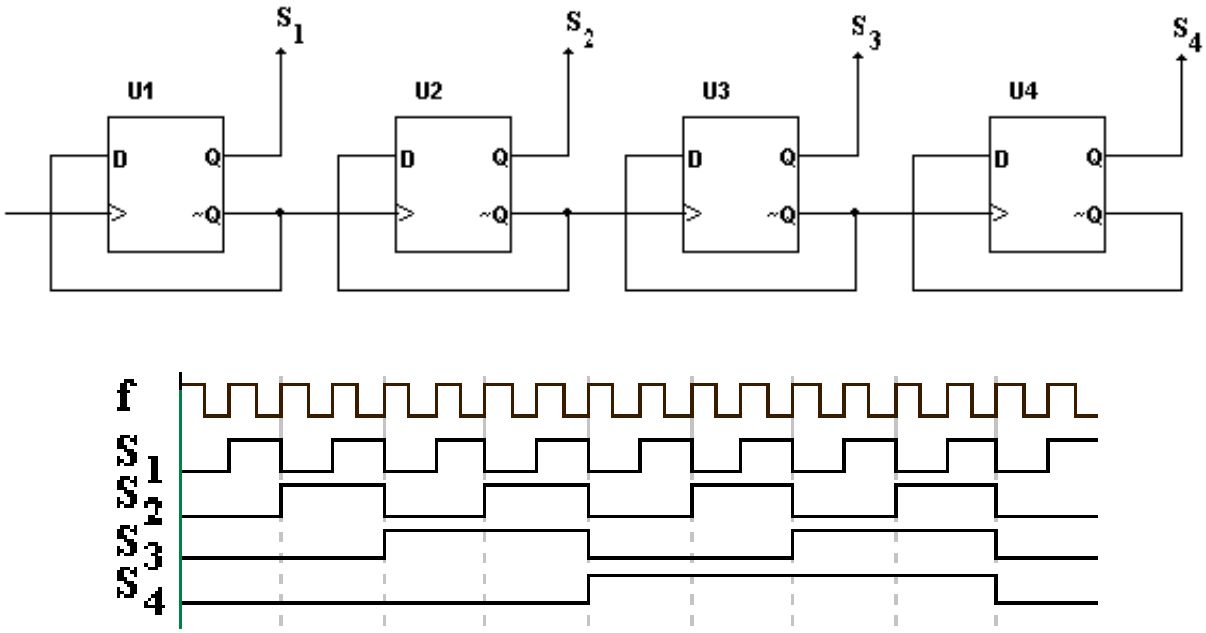


Рисунок 9 – Суммирующий счетчик и диаграмма его работы

Для того чтобы изменить направление счета (реализовать вычитающий счетчик), необходимо счетный вход очередного триггера подключить к прямому выходу предыдущего, как показано на рисунке 10, при этом изменяется последовательность переключения триггеров.

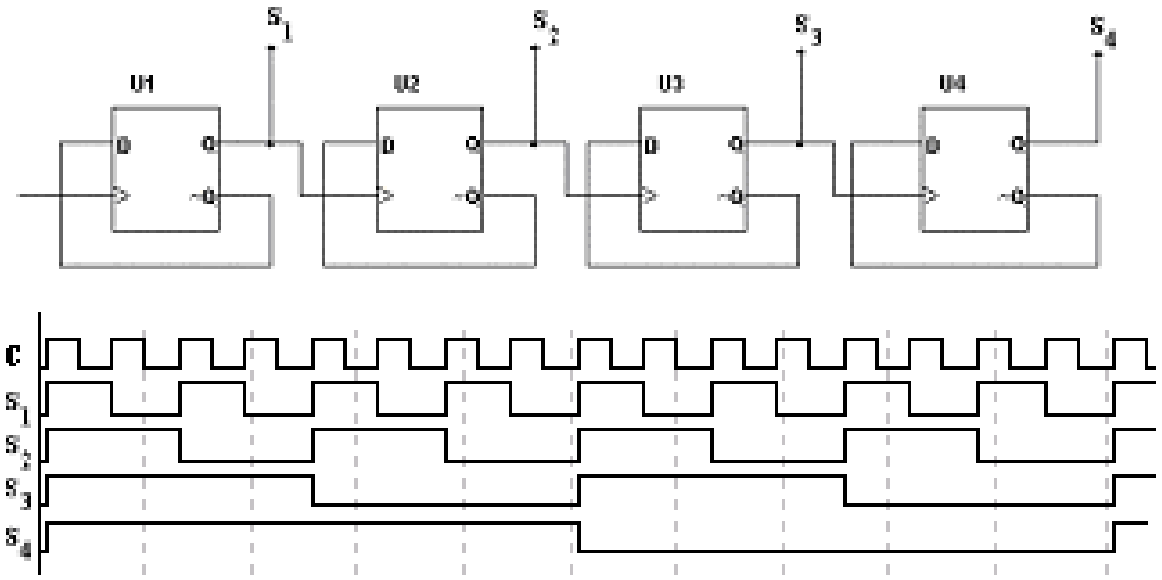


Рисунок 10 – Вычитающий счетчик и диаграмма его работы

Счетчик характеризуется числом состояний в течение одного периода (***цикла***). Для двоичных счетчиков полный цикл счета $N=2^n$ от состояния 0..000 до состояния 1…11. Число состояний называется коэффициентом пересчета $K_{сч}$, равным отношению числа импульсов N_c на входе к числу импульсов $N_{Qст}$ на выходе старшего разряда за период

$$K_{сч} = \frac{N_c}{N_{Qст}}.$$

Если на вход счетчика подавать периодическую последовательность импульсов с частотой f_c, то частота f_Q на выходе старшего разряда счетчика будет меньше в $K_{сч}$ раз: $K_{сч} = f_c / f_Q$.

Поэтому счетчики также называют *делителями частоты*, а $K_{сч}$ – *коэффициентом деления*.

Для увеличения величины $K_{сч}$ нужно увеличить число триггеров в цепочке. Каждый дополнительный триггер удваивает число состояний счетчика и число $K_{сч}$.

Порядок выполнения работы

Исследование регистра

1. Запустить среду разработки *Multisim*. На рабочем поле среды моделирования собрать схему для испытания четырехразрядного регистра

сдвига с автоматической записью единицы в первый разряд регистра, как показано на рисунке 11.

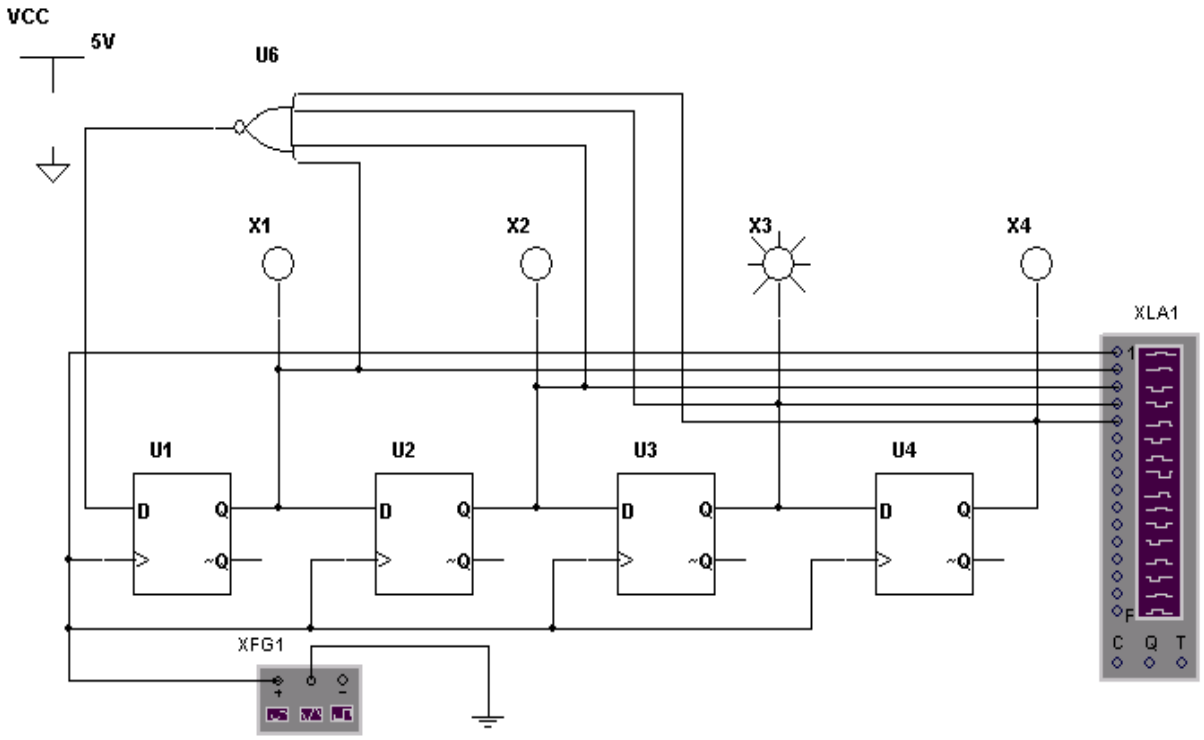


Рисунок 11 – Четырехразрядный регистр сдвига
с автоматической записью единицы

Схема содержит четыре *D*-триггера, четыре светодиода, одну логическую схему «4ИЛИ-НЕ», функциональный генератор и логический анализатор. Логическая схема «4ИЛИ-НЕ» служит для автоматической записи единицы в регистр. На выходе этой схемы единица будет только тогда, когда все разряды регистра будут находиться в нулевом состоянии.

2. Открыть окно функционального генератора и установить вид генерируемых сигналов (прямоугольные импульсы), генерируемую частоту 1000 Гц, амплитуду генерируемых импульсов 5 В.

Справочно

Диалоговое окно ***функционального генератора (Function Generator)*** *представлено на рисунке 12.*

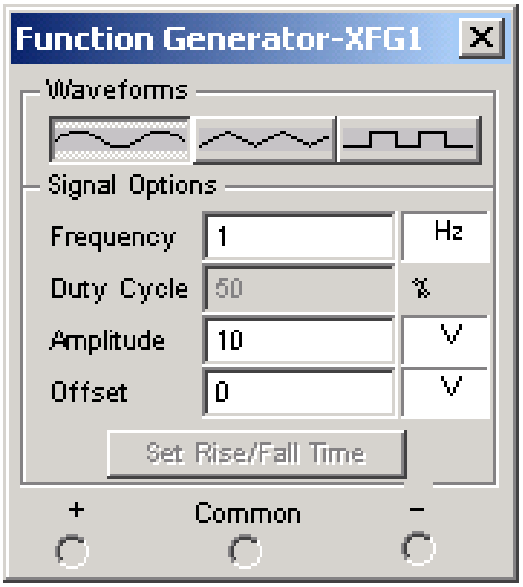


Рисунок 11 – Лицевая панель функционального генератора

На лицевой панели функционального генератора расположены три кнопки  *, с помощью которых выбирается форма генерируемого сигнала: синусоидальной (выбирается по умолчанию), треугольной и прямоугольной.*

В четырех белых окнах устанавливают параметры выходного сигнала:

- ***частоты*** –  ,
- ***коэффициента заполнения*** *в процентах* – 

  Для ***импульсных*** *сигналов это отношение длительности импульса к периоду повторения (величина обратная скважности), для* ***треугольных*** *сигналов – соотношение между длительностями переднего и заднего фронтов,*
- ***амплитуды выходного сигнала*** –  ,
- ***смещения*** *(постоянной составляющей)* –  .

Внизу лицевой панели расположены выходные зажимы –  *; при заземлении клеммы COM (Common – общий) на клеммах «+» и «-» генератор выдает парафазный сигнал.*

3. Открыть окно логического анализатора, дважды щелкнув по иконке логического анализатора.

4. Запустить процесс моделирования, нажав кнопку  на панели инструментов, и в появившемся меню выбрать команду *Run*.

5. Скопировать изображения диаграммы сигналов с экрана логического анализатора в отчет.

6. Зафиксировать и скопировать в отчет изображения всех состояний светодиодов и сравнить их с полученными временными диаграммами.

Исследование суммирующего счетчика

1. На рабочем поле среды моделирования собрать схему для испытания четырехразрядного счетчика, считающего в прямом направлении, как показано на рисунке 13.

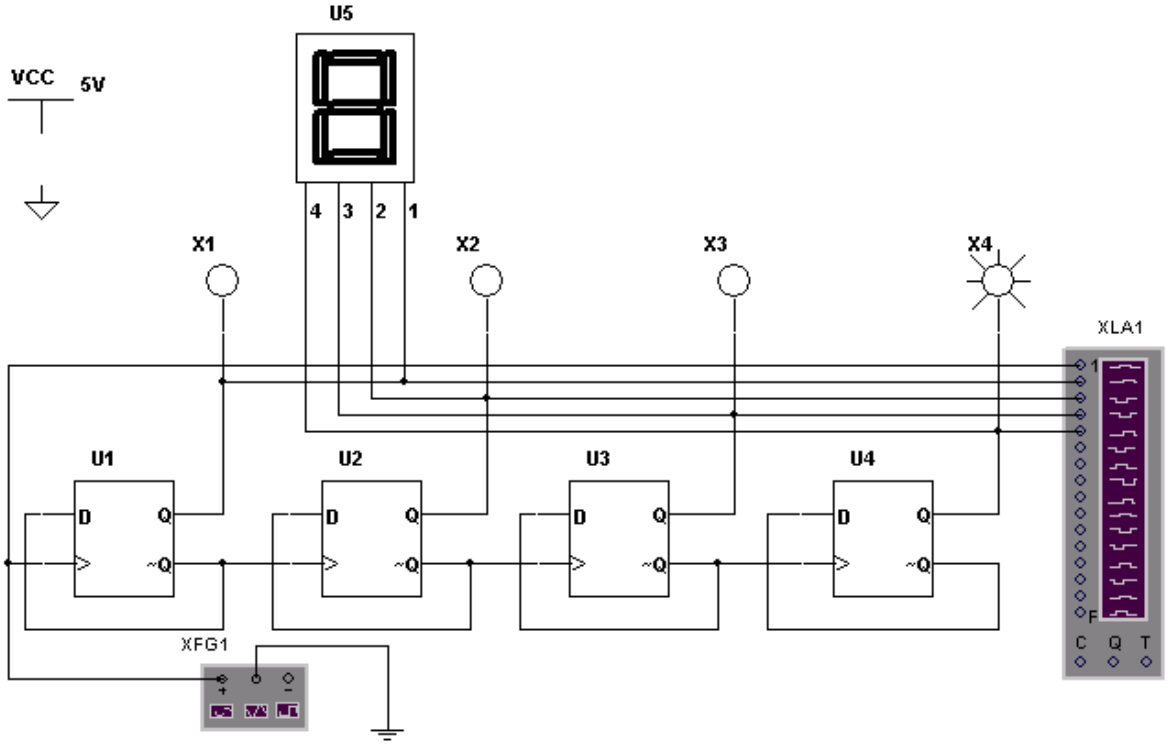


Рисунок 13 – Четырехразрядный суммирующий счетчик

Поместить на схему четыре *D*-триггера, четыре светодиода, функциональный генератор, логический анализатор и 16-ричный индикатор.

2. Открыть окно логического анализатора, щелкнув по иконке логического анализатора.

3. Запустить процесс моделирования, нажав кнопку  на панели инструментов, и в появившемся меню выбрать команду *Run*.

4. Скопировать изображения диаграммы сигналов с экрана логического анализатора в отчет.

5. Зафиксировать и скопировать в отчет изображения всех состояний светодиодов и сравнить их с полученными временными диаграммами.

6. Наблюдать за показаниями 16-ричного индикатора и сравнить его показания с соответствующими состояниями светодиодов. Зафиксировать и скопировать в отчет несколько изображений 16-ричного индикатора.

Исследование вычитающего счетчика

1. На рабочем поле среды моделирования собрать схему для испытания четырехразрядного счетчика, считающего в обратном направлении, как показано на рисунке 14.

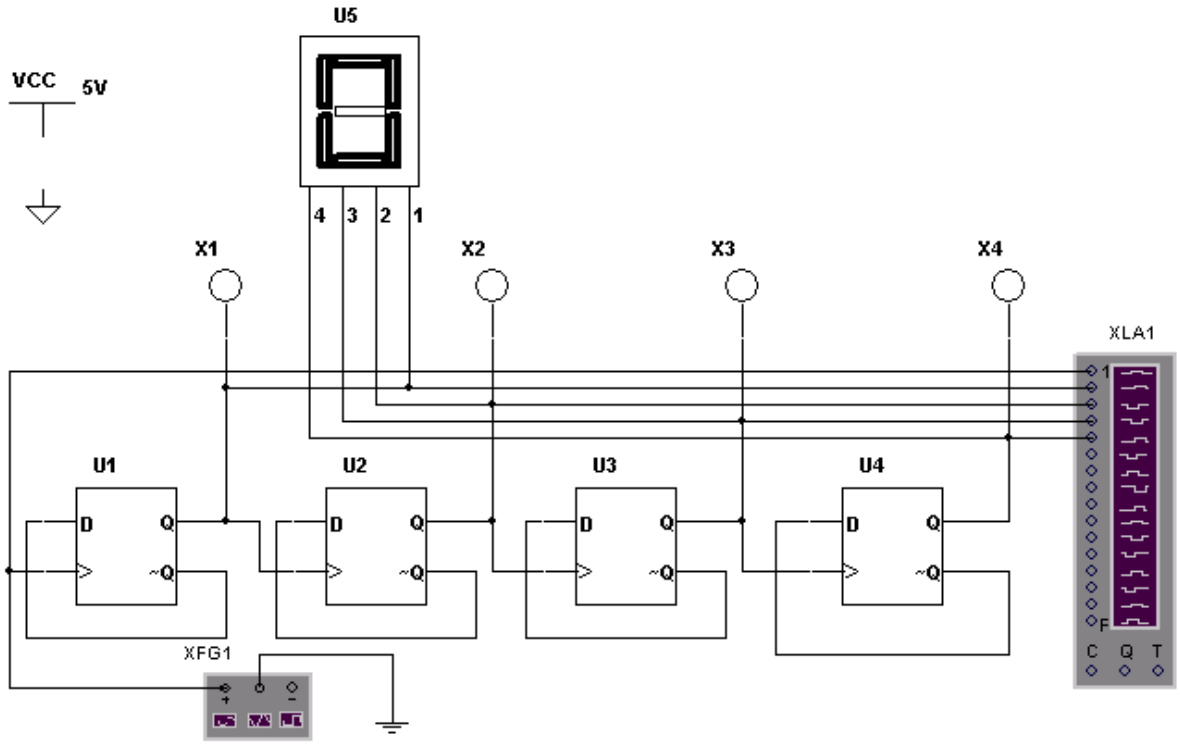


Рисунок 14 – Четырехразрядный вычитающий счетчик

2. Открыть окно логического анализатора, щелкнув по иконке логического анализатора.

3. Запустить процесс моделирования, нажав кнопку  на панели инструментов, и в появившемся меню выбрать команду *Run*.

4. Скопировать изображения диаграммы сигналов с экрана логического анализатора в отчет.

5. Зафиксировать и скопировать в отчет изображения всех состояний светодиодов и сравнить их с полученными временными диаграммами.

6. Наблюдать за показаниями 16-ричного индикатора и сравнить его показания с соответствующими состояниями светодиодов. Зафиксировать и скопировать в отчет несколько изображений 16-ричного индикатора.

Контрольные вопросы:

1. Перечислите запоминающие элементы и узлы. Дайте им краткую характеристику.

2. Что такое триггер? Перечислите виды и дайте им краткую характеристику.

3. Что такое регистр? Перечислите виды и дайте им краткую характеристику.

4. Что такое счетчик? Перечислите виды и дайте им краткую характеристику.

5. Почему счетчики также называют делителями частоты? Дайте определение коэффициенту деления.

6. Что нужно сделать для увеличения значения коэффициента деления. Покажите на примере.

7. Для чего на построенных в лабораторной работе схемах используются индикаторы нескольких видов?

8. Для чего предназначен функциональный генератор?

ЛАБОРАТОРНАЯ РАБОТА № 9
ИССЛЕДОВАНИЕ ИНТЕГРАЛЬНЫХ ПРЕОБРАЗОВАТЕЛЕЙ КОДОВ

Цель работы: Научиться моделировать работу интегральных преобразователей кодов на примере дешифратора.

Краткие теоретические сведения

Комбинационной называется логическая схема, реализующая однозначное соответствие между значениями входных и выходных сигналов.

Дешифратор – логическая комбинационная схема, имеющая n информационных входов и 2^n выходов. Каждой комбинации логических уровней на входах будет соответствовать активный уровень на одном из 2^n выходов.

Как любая логическая схема, дешифратор может быть задан таблицей истинности. Таблица истинности дешифратора 3×8 представлена таблицей 1 и состоит из трех столбцов, соответствующих входным сигналам X_0, X_1, X_2, и восьми столбцов, соответствующих выходным сигналам Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7. В первых слева трех столбцах расположены возможные комбинации входных сигналов, а в последних восьми – соответствующие им комбинации выходных сигналов.

Таблица 1

X_2	X_1	X_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Схема дешифратора имеет восемь выходов, на одном из которых потенциал высокий, на остальных – низкий. Номер единственного выхода, имеющего высокий потенциал, соответствует двоичному числу, формируемому состояниями входных сигналов.

Этот принцип формирования выходного сигнала можно описать следующим образом:

$$Y_i = 0, \text{ если } i = k;$$

$$Y_i = 1, \text{ если } i \neq k,$$

здесь i – номер разряда; $k = 2^2 X_2 + 2^1 X_1 + 2^0 X_0$.

Выражения для каждого выхода дешифратора:

$$\begin{aligned} \overline{Y}_0 &= \overline{X}_2 \overline{X}_1 \overline{X}_0, & Y_4 &= X_2 \overline{X}_1 \overline{X}_0, \\ \overline{Y}_1 &= \overline{X}_2 \overline{X}_1 X_0, & Y_5 &= X_2 \overline{X}_1 X_0, \\ \overline{Y}_2 &= \overline{X}_2 X_1 \overline{X}_0, & Y_6 &= \overline{X}_2 X_1 X_0, \\ Y_3 &= X_2 X_1 \overline{X}_0, & Y_7 &= X_2 X_1 X_0, \end{aligned}$$

где «$\overline{\ }$» – инвертирование значения сигнала.

Таким образом, схема дешифратора должна содержать три схемы «НЕ» и восемь схем «ЗИ» как показано на рисунке 1.

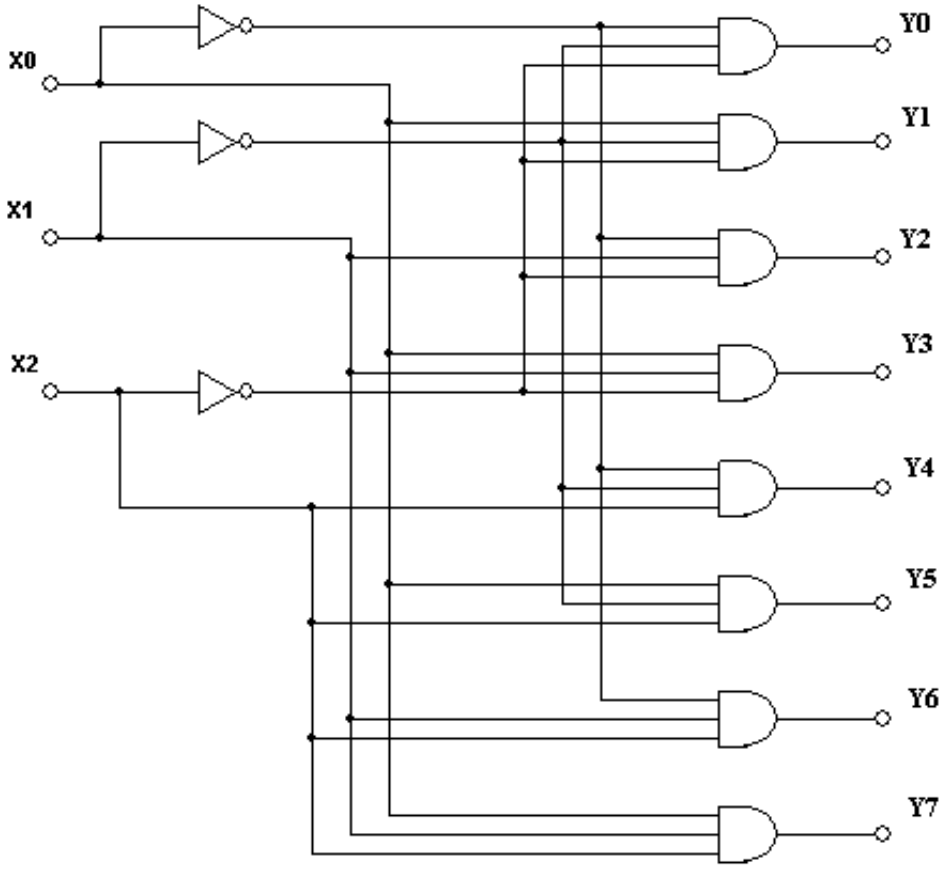


Рисунок 1 – Схема дешифратора 3×8.

Порядок выполнения работы

1. Запустить среду разработки Multisim. На рабочем поле среды моделирования собрать схему для испытания дешифратора 3×8 как показано на рисунке 2. Для этого поместить на схему восемь логических схем «3И», восемь светодиодов, пять логических схем «НЕ», три переключателя на два направления и три вольтметра.

2. Присвоить переключателям управляющие клавиши.

3. С помощью переключателей подать на вход дешифратора все возможные комбинации входных сигналов и записать для каждого входного сигнала выходной сигнал (восьмиразрядную комбинацию). Перенести таблицу в отчет.

4. Сравнить заполненную таблицу с таблицей истинности дешифратора. Сделать выводы.

5. Оформить отчет о проделанной работе, который должен содержать

1) название пункта работы,

2) иллюстрации исследуемой схемы,

3) результат моделирования, таблицы истинности,

4) выводы.

Контрольные вопросы:

1. Дайте определение интегральным преобразователям сигналов. Назовите основные из них.

2. Какие схемы называются комбинационными?

3. Дайте характеристику шифратору и дешифратору. Приведите примеры.

4. Приведите таблицу истинности шифратора и дешифратора. Сравните их. Приведите пример.

5. По какому принципу и на каких логических элементах строятся линейные дешифраторы?

6. Как можно проверить правильность собранной схемы?

ЛАБОРАТОРНАЯ РАБОТА № 10
ИССЛЕДОВАНИЕ АРИФМЕТИЧЕСКОГО СУММАТОРА

Цель работы: Научиться моделировать работу арифметического полусумматора и сумматора.

Краткие сведения из теории

Сумматор является простейшим цифровым устройством, предназначенным для сложения двух чисел, заданных в двоичном коде. Сложение производится поразрядно – от младшего разряда к старшему.

По числу входов различают *полусумматоры*, *одноразрядные сумматоры* и *многоразрядные сумматоры*.

Арифметические сумматоры – составная часть арифметико-логических устройств (АЛУ) микропроцессоров компьютера. Арифметический сумматор состоит из двух устройств: *полусумматора* и *n полных сумматоров*.

Полный сумматор имеет три входа: *A, B* – входы суммируемых операндов, C_i – вход переноса из предыдущего разряда сумматора и два выхода: *S* – выход полного сумматора и C_0 – выход переноса.

Полусумматор отличается от полного тем, что у него нет входа переноса из предыдущего разряда. Он используется в качестве первого разряда арифметического сумматора, а в качестве остальных разрядов – полные сумматоры (рис. 1). Полусумматор – одна из простейших комбинационных логических схем.

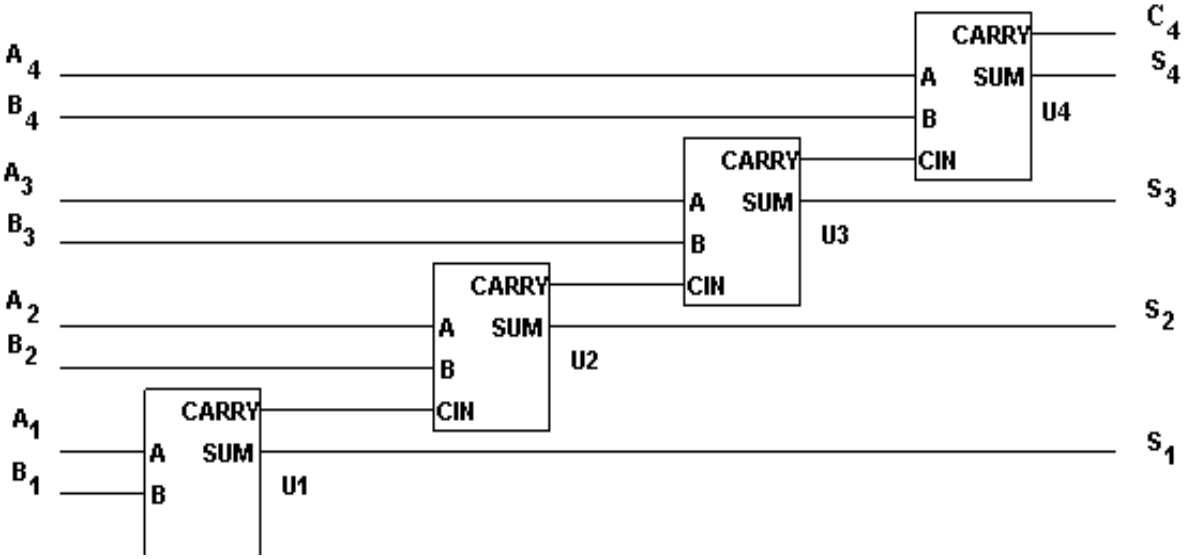


Рисунок 1 – Принципиальная схема четырехразрядного арифметического сумматора

Судя по таблице истинности полусумматора (таблица 1) можно заметить, что выход *S* полусумматора выполняет функции элемента «ИСКЛЮЧАЮЩЕЕ ИЛИ», а выход переноса *C* полусумматора – элемента «И».

Таблица 1

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

Таким образом, логические выражения для функций *S* и *C* равны

$$S=AB'+A'B,$$

$$C=AB.$$

Тогда, используя эти логические выражения, схема полусумматора будет выглядеть так, как представлена на рисунке 2.

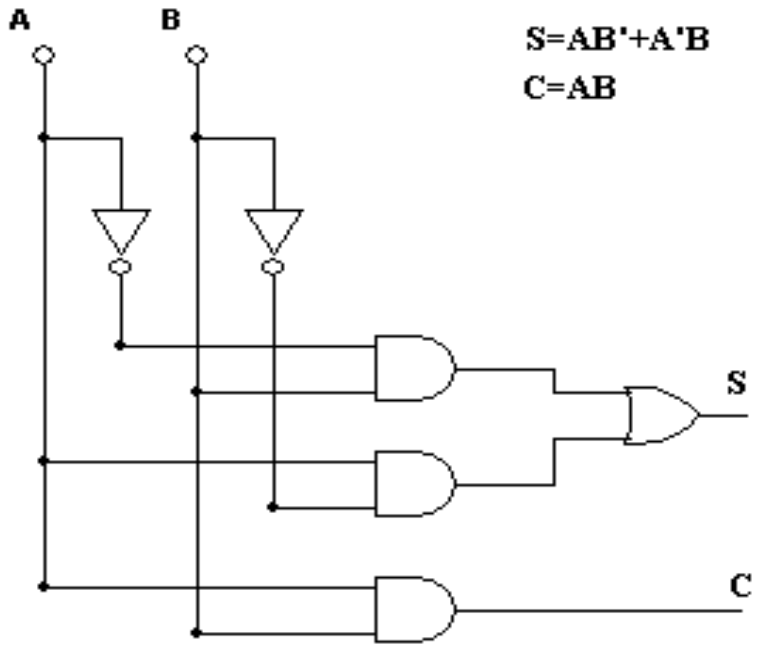


Рисунок 2 – Логическая схема полусумматора

Из таблицы истинности полного сумматора (таблица 2) можно получить логические выражения для *S* (суммы) и *C* (переноса в следующий разряд).

Таблица 2

Входы			**Выходы**	
A	**B**	$\mathbf{C_{i-1}}$	**S**	$\mathbf{C_i}$
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

Логическое выражение для *S* будет иметь четыре слагаемых, соответствующих строкам таблицы, в которых выход *S* равен единице (строки 4, 5, 7, 10):

$$S= A′B′C_{i-1}+A′BC_{i-1}′+AB′C_{i-1}′+ABC_{i-1}.$$

Логическое выражение для *C* также будет иметь четыре слагаемых (строки 6, 8, 9, 10):

$$C_i=A′BC_{i-1}+A′BC_{i-1}′+ABC_{i-1}′+ABC_{i-1}.$$

С помощью законов булевой алгебры это выражение можно минимизировать (упростить), тогда оно будет иметь следующий вид:

$$C_i=AC_{i-1}+BC_{i-1}+AB.$$

С учетом преобразований схема полного сумматора будет выглядеть так, как представлено на рисунке 3.

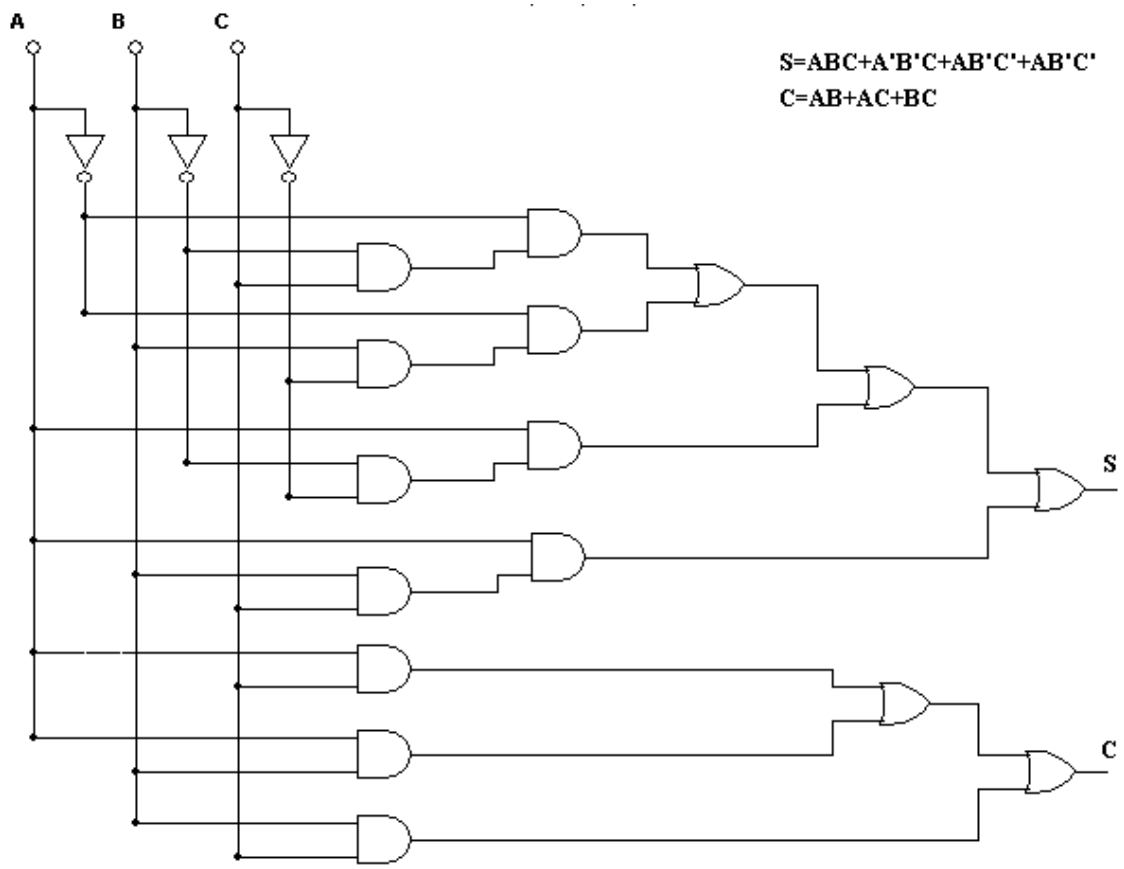


Рисунок 3 – Логическая схема полного сумматора

В зависимости от характера ввода-вывода кодов и организации переносов многоразрядные сумматоры бывают *последовательного* и *параллельного* принципа действия.

В **последовательном сумматоре** сложение кодов осуществляется поразрядно, начиная с младшего разряда, с помощью комбинационного сумматора на три входа. Образующийся в данном разряде перенос C_i задерживается на время $t_{зд}$ и поступает на вход C_{i-1} сумматора в момент поступления следующего разряда слагаемых. Таким образом, последовательно, разряд за разрядом, производится сложение кодов чисел.

Схематически последовательный сумматор можно представить рисунком 4.

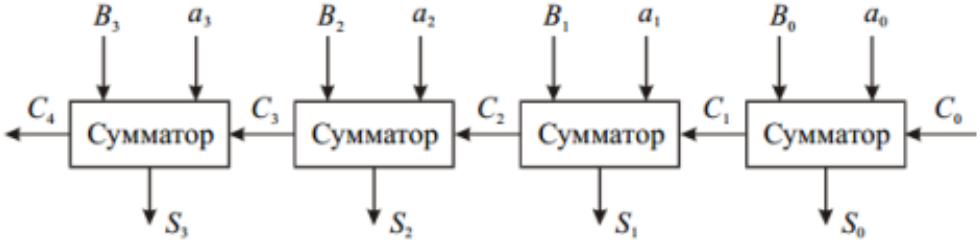


Рисунок 4 – Структура последовательного сумматора

Достоинством последовательного сумматора является простота аппаратурной реализации, а недостатком – достаточно большое время суммирования.

В **параллельном сумматоре** достигается более высокое быстродействие. Суммируемые коды поступают на входы сумматора одновременно по всем разрядам (рис. 5).

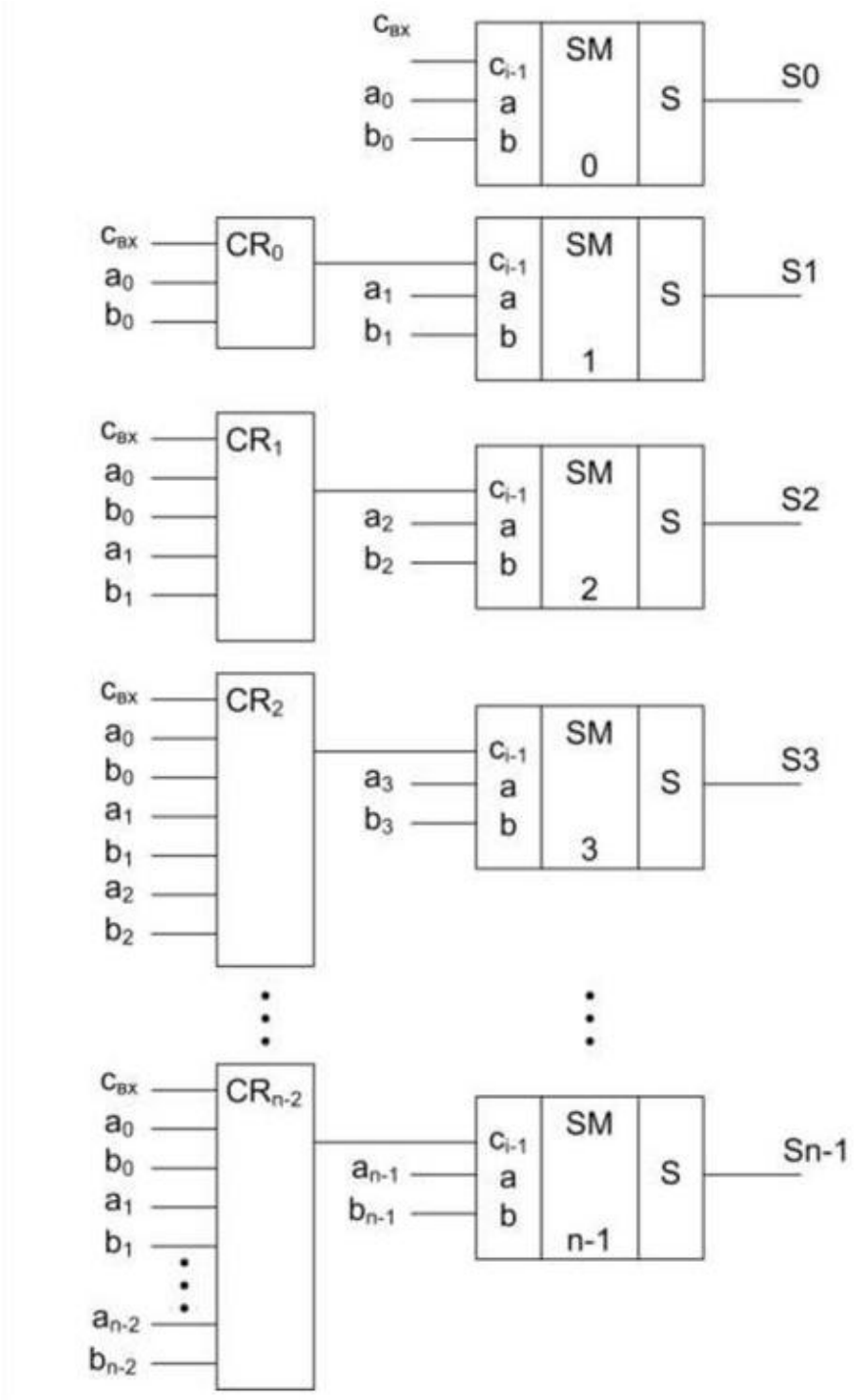


Рисунок 5 – Структура параллельного сумматора

Для этого в каждом разряде используется комбинационный сумматор на три входа, на выходах которого образуются значения суммы S_i данного разряда и переноса C_{i-1} в старший разряд. В процессе распространения сигнала переноса устанавливается окончательное значение суммы в каждом разряде. Очевидно, что в течение этого времени на входах сумматора присутствуют сигналы A_i, B_i, соответствующие суммируемым кодам.

Максимальное по времени суммирование получается в том случае, когда перенос, возникший в первом разряде, распространяется по всем разрядам (например, при сложении кодов 11..11 и 00..01).

В параллельном сумматоре обычно применяются различные способы ускорения переноса (параллельный перенос, групповой и др.).

Порядок выполнения работы

1. Запустить среду разработки *Multisim*. На рабочем поле среды моделирования собрать схему для испытания четырехразрядного арифметического сумматора, как показано на рисунке 6. Поместить на схему три 16-ричных индикатора и генератор слова.

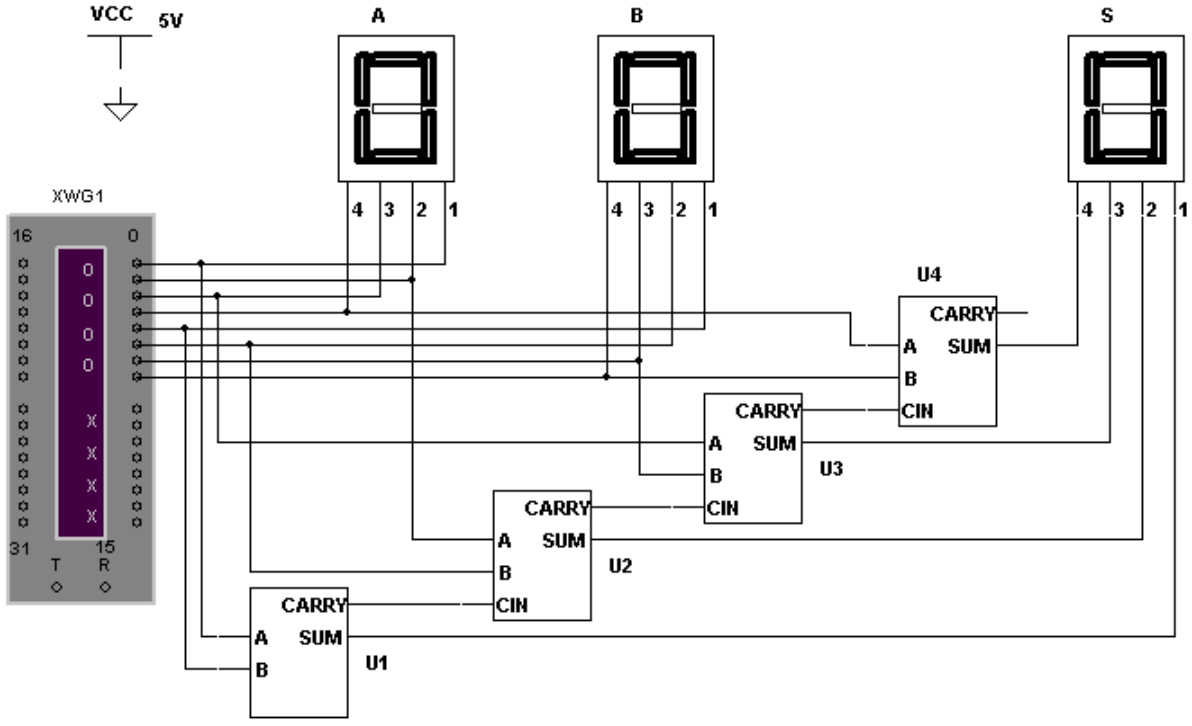
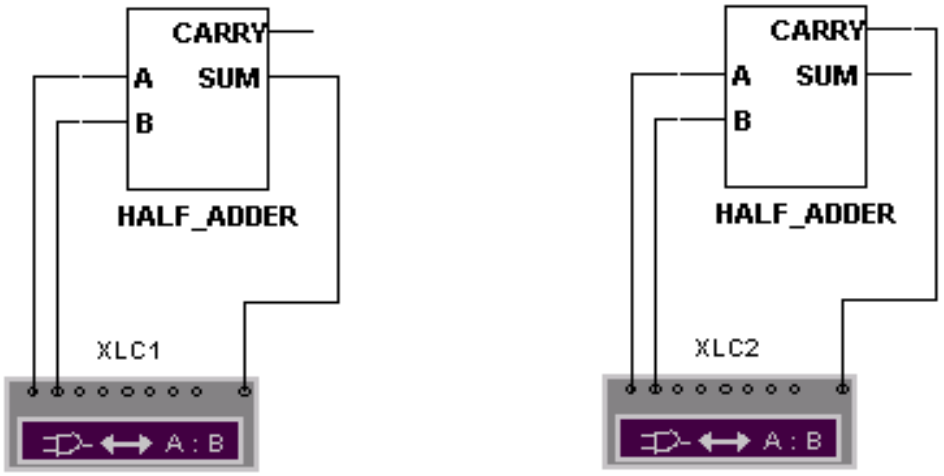


Рисунок 6 – Схема исследования четырехразрядного сумматора

2. Открыть генератор слова и в двоичном представлении задать суммируемые числа. Четыре младших разряда каждого генерируемого слова составляют первое слагаемое (операнд). Следующие четыре разряда составляют второе слагаемое (операнд). Записать суммируемые числа в двоичном представлении в отчет.

3. Запустить процесс моделирования и следить за показаниями индикаторов. Записать суммируемые числа и результат суммирования в отчет.

4. Собрать схему, изображенную на рисунке 7.

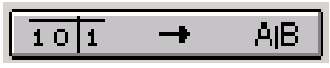


а *б*

Рисунок 7 – Схема исследования полусумматора: *а* – выход *S*, *б* – выход *C*

С помощью *логического преобразователя*, последовательно нажимая кнопки

Circuit to Truth Table (таблица истинности цепи)  ,

Truth Table to Boolean Expression (булево выражение по таблице истинности)  ,

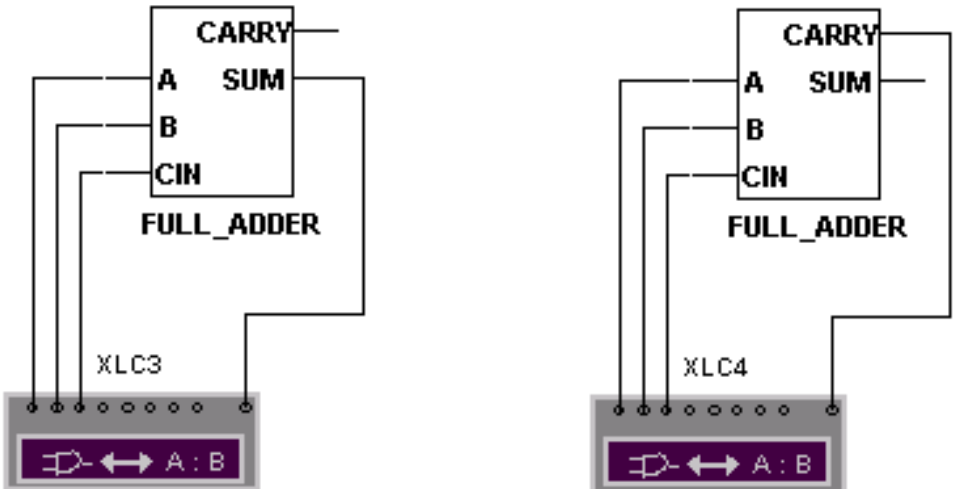
Boolean Expression to Circuit (создание схемы по булеву выражению)  ,

получить:

- таблицу истинности полусумматора,
- логические выражения для выходов *S* и *C*,
- схемную реализацию логических выражений для выходов *S* и *C*.

Сравнить полученные результаты с теоретическими сведениями. Сделать соответствующий вывод.

5. Собрать схему, изображенную на рисунке 8, и с помощью *логического преобразователя* получить таблицу истинности полного сумматора, логические выражения для выходов *S* и *C* и схемную реализацию логических выражений (см. п. 4).



а *б*

Рисунок 8 – Схема исследования полного сумматора:
а – выход *S*, *б* – выход *C*

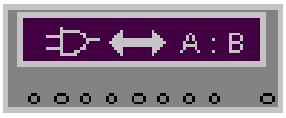
Сравнить полученные результаты с теоретическими сведениями. Сделать соответствующий вывод.

6. Оформить отчет о проделанной работе, который должен содержать:

1) название пункта работы;
2) иллюстрации исследуемой схемы;
3) результат моделирования;
4) выводы.

Справочно

Логический преобразователь (Logic Converter)



В диалоговом окне логического преобразователя, представленном на рисунке 9, расположены клеммы-индикаторы входов ***A, B, C, …, H*** *и клемма выхода* ***Out****, окно для отображения таблицы истинности исследуемой схемы, строка для отображения ее булева выражения и панель* ***Conversions****.*

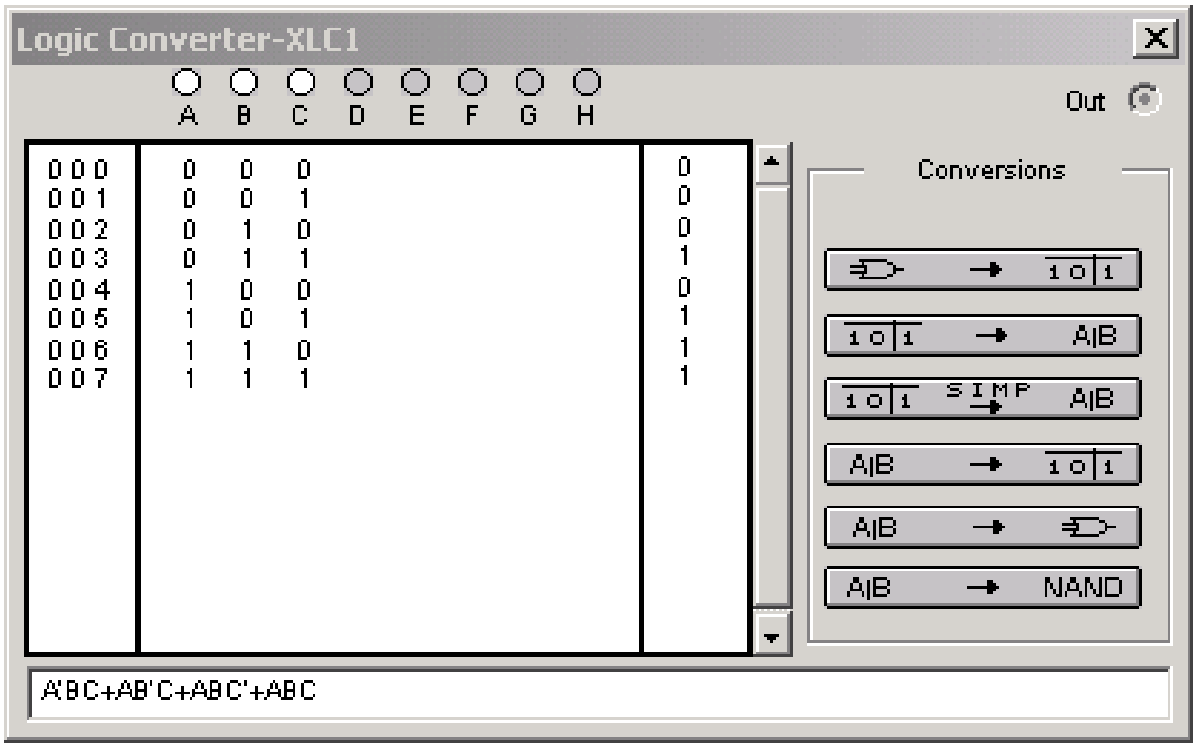


Рисунок 9 – Диалоговое окно логического преобразователя

На панели ***Conversions*** *расположены шесть кнопок, используемых для получения:*

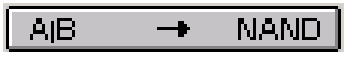
 *– таблицы истинности исследуемого устройства,*

 *– булева выражения, реализуемого исследуемым устройством,*

 *– минимизированного булева выражения,*

 *– таблицы истинности по булевому выражению;*

 *– схемы устройства по логическому выражению на логических элементах без ограничения их типа,*

 *– создания схемы устройства только на логических элементах «И-НЕ».*

Контрольные вопросы:

1. Дайте определение понятию «комбинационная схема». Приведите примеры.

2. Дайте характеристику комбинационному устройству сумматор.

3. Для чего используется арифметический сумматор? Приведите логическое выражение, описывающее его работу.

4. Что такое полусумматор? Для чего он используется? Приведите логическое выражение, описывающее его работу.

5. Как осуществляется сложение и вычитание многоразрядных чисел в сумматоре?

6. Для чего в лабораторной работе используется генератор слова? Дайте ему краткую характеристику.

7. Для чего в лабораторной работе используется логический преобразователь? Дайте ему краткую характеристику.

8. Для чего в лабораторной работе используется 16-ричные индикаторы? Как проверить правильность работы собранных схем?

9. В состав какого устройства компьютера входит арифметический сумматор? Дайте ему краткую характеристику.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Буза, М. К. Архитектура компьютеров : учебник / М.К. Буза. – Минск : Новое знание, 2015. – 559 с.

2. Гагарина, Л. Г. Современные проблемы информатики и вычислительной техники / Л. Г. Гагарина, А. А. Петров. – Москва : Форум : ИНФРА-М, 2017. – 367 с. : ил.

3. Бройдо, В. Л. Архитектура ЭВМ и систем : учебник / В.Л. Бройдо, О. П. Ильина. – Санкт-Петербург : Питер, 2006. – 718 с.

4. Избачков, Ю. И. Информационные системы : учебное пособие / Ю.И. Избачков, В.Н. Петров. – Санкт-Петербург : Питер, 2006. – 656 с.

5. Мощенский, А. В. Математические основы информатики : пособие для студентов / А. В. Мощенский, В. А. Мощенский. – Минск : БГУ, 2002. – 150 с.

6. Цилькер, Б. Я. Организация ЭВМ и систем: учебник / Б. Я. Цилькер, С. А. Орлов. – Санкт-Петербург : Питер, 2006. – 668 с.

СОДЕРЖАНИЕ