

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
МИРЭА - РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ

С.С. Смирнов, Д.А. Карпов

ИНФОРМАТИКА

Методические указания
по выполнению практических
работ для студентов,
обучающихся по направлениям:

- 09.03.01 - Информатика и вычислительная техника,
- 09.03.02 - Информационные системы и технологии,
- 9.03.03 - Прикладная информатика,
- 09.03.04 - Программная инженерия,
- 15.03.04 - Автоматизация технологических процессов
и производств.

МОСКВА – 2020

*Печатается по решению редакционно-издательского совета
Российского технологического университета (РТУ МИРЭА)*

*Рецензент:
Смольянинова Валерия Аполлоновна
к.т.н, доцент кафедры МОСИТ РТУ МИРЭА.*

Рекомендовано к изданию на заседании кафедры общей информатики
института кибернетики, протокол № ____ от _____ 2020 года.

Смирнов С.С., Карпов Д.А.

Информатика: Методические указания по выполнению практических работ / С.С. Смирнов, Д.А. Карпов — М., МИРЭА — Российский технологический университет, 2020. — 102 с.

Разработаны в помощь студентам, выполняющим практические работы по дисциплине Информатика. В состав методических указаний входят необходимые рекомендации, методический материал, примеры выполнения практических работ по дисциплине «Информатика».

Методические указания предназначены для студентов института Информационных технологий университета и может быть использован для самостоятельной работы.

© Смирнов С.С., Карпов Д.А.
© МИРЭА — Российский технологический университет, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
РАЗДЕЛ №1: АРИФМЕТИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ	7
1. Практическая работа № 1: выполнение арифметических операций в различных системах счисления	7
2. Практическая работа № 2: выполнение операций вычитания в различных кодах	14
3. Практическая работа № 3: перевод вещественных чисел и выполнения арифметических операций над ними в рамках стандарта IEEE754	18
РАЗДЕЛ №2: ЛОГИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ	28
4. Практическая работа № 4: анализ логических схем в игровой форме	28
5. Практическая работа №5: построение комбинационных схем, реализующих СДНФ и СКНФ заданной логической функции от 4-х переменных	30
6. Практическая работа №6: построение комбинационных схем, реализующих МДНФ и МКНФ заданной логической функции от 4-х переменных в базисах И-НЕ, ИЛИ-НЕ	35
7. Практическая работа № 7: реализация заданной логической функции от четырех переменных на дешифраторах 4-16, 3-8 и 2-4	44
8. Практическая работа № 8: реализация заданной логической функции от четырех переменных на мультиплексорах 16-1, 8-1, 4-1, 2-1	51
9. Практическая работа №9: преобразователи кодов	59
10. Практическая работа №10: изучение работы триггеров	62
11. Практическая работа №11: синтез четырехразрядного счетчика с параллельным переносом между разрядами двумя способами	69
РАЗДЕЛ №3: АЛГОРИТМИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ	78
12. Практическая работа №12: элементы алгоритмизации и процедурного программирования	82
ПРИЛОЖЕНИЕ 1. ОПИСАНИЕ ЛАБОРАТОРНОГО КОМПЛЕКСА	90
ПРИЛОЖЕНИЕ 2. ОБОБЩЕННАЯ МЕТОДИКА ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ ПРИ ПОМОЩИ МОДЕЛИ ЛАБОРАТОРНОГО СТЕНДА	93
ПРИЛОЖЕНИЕ 3. РЕКОМЕНДАЦИИ ПО ОТЛАДКЕ СХЕМ В СРЕДЕ LOGISIM	97

ВВЕДЕНИЕ

Учебный план курса «Информатика» для студентов института Информационных технологий предполагает выполнение в общей сложности 12 практических работ, целью проведения которых является закрепление теоретических знаний по данному курсу.

Выполнение практических работ стимулирует развитие следующих умений и способностей из перечня, закрепленного в Рабочей программе курса:

- умение использовать программные средства для решения практических задач;
- умение использовать различные ресурсы, в том числе электронные, и справочные материалы для поиска информации, необходимой для решения поставленных задач;
- быть способным к самоорганизации и самообразованию.

В процессе выполнения практических работ студент должен решить множество задач, относящихся к арифметическим, логическими алгоритмическим основам вычислительной техники.

Из области арифметических основ должны быть решены следующие задачи:

- Перевести числа из одной системы счисления в другую.
- Выполнить арифметические операции в различных системах счисления.
- Выполнить сложение и вычитание в смешанной К-Q-ичной системе счисления на примере двоично-десятичной системы счисления.
- Вычислить разность заданных чисел в обратном и дополнительном кодах на основе двоичной системы.
- Выполнить перевод вещественного числа из десятичной системы счисления в форматы половинной и одинарной точности стандарта IEEE754, а также выполнить обратное преобразование.
- Выполнить арифметические операции над числами в формате одинарной точности стандарта IEEE754.

Из области логических основ вычислительной техники должны быть решены следующие задачи:

- На примере системы Logisim познакомиться с инструментарием для проектирования и моделирования цифровых схем, изучить систему Logisim в объеме, достаточном для выполнения практических работ, для чего воспользоваться руководством пользователя от разработчика Logisim, а также данными

методическими указаниями.

- Записать формулы СДНФ и СКНФ логической функции, заданной в векторной форме, а также построить в общем базисе соответствующие комбинационные схемы и смоделировать их работу.
- Минимизировать методом карт Карно логическую функцию, заданную в векторной форме, получить МДНФ и МКНФ.
- Представить полученные ранее МДНФ и МКНФ в базисах «И-НЕ» и «ИЛИ-НЕ» (каждую минимальную форму в двух базисах).
- Для каждой минимальной формы, представленной в требуемом базисе, построить комбинационные схемы и смоделировать их работу (в схеме использовать только элементы из требуемого базиса).
- Для логической функции, заданной в векторной форме, построить её реализацию на дешифраторах различной адресности и смоделировать их работу.
- Для логической функции, заданной в векторной форме, построить её реализацию на мультиплексорах различной адресности и смоделировать их работу.
- При помощи дешифратора и шифратора построить преобразователь кодов и смоделировать его работу. В качестве исходных данных использовать таблицу переходов, образованную четырьмя логическими функциями, заданными в векторной форме.
- Спроектировать и промоделировать работу четырехразрядного счетчика с параллельным переносом между разрядами на основе заданных шага, направления счета, а также максимального допустимого значения.

В рамках изучения алгоритмических основ вычислительной техники требуется разработать блок-схему алгоритма и написать соответствующую программу обработки данных на любом известном студенту языке процедурного программирования (либо же на учебном алгоритмическом языке). Тематика заданий данного раздела связана с различными манипуляциями над двумерными и одномерными массивами чисел, строк и букв.

Результаты выполнения практических работ оформляются и предоставляются на проверку частично в рукописном, частично в электронном виде.

Задачи по арифметическим основам вычислительной техники являются расчётными, поэтому отчеты по ним оформляются в рукописном виде.

Задачи по логическим и алгоритмическим основам вычислительной техники требуют работы с компьютером, поэтому отчеты по ним оформляются в электронном виде.

Студент должен завести тетрадь в клетку объемом 12–18 листов и

подписать её. В данной тетради приводятся все чистовые расчеты, относящиеся к задачам из области арифметических основ.

Также в этой тетради ставятся отметки преподавателя о выполнении **всех** работ студента (как по арифметическим основам, так и по логическим и алгоритмическим).

Для этого на оборотной стороне обложки студентом рисуется таблица 1.

Таблица 1.

Учет сданных практических работ

№ работы	Дата представления	Дата защиты	Подпись преподавателя	Примечание
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Дата представления заполняется студентом в день первой сдачи работы на проверку. Остальные колонки заполняются преподавателем.

РАЗДЕЛ №1: АРИФМЕТИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

1. Практическая работа № 1: выполнение арифметических операций в различных системах счисления

1. Постановка задачи

Часть 1. Произвести вычисления в системах счисления с различным основанием (определяются индивидуальным вариантом) для выражения, задаваемого формулой 1:

$$(A_{SA}[+]_{plus} B_{SB}[-]_2 C_2)[/]_2 D_2 \Rightarrow X_{SX} \quad (1)$$

Часть 2. Произвести сложение двух чисел в двоично-десятичном коде.

Работа выполняется сначала на черновике. Все этапы вычислений должны быть подробно расписаны. Результаты расчетов вводятся в программу проверки. Программа проверки оценивает правильность полученных результатов в процентах. При 100% правильности студент аккуратно и читабельно оформляет решение на чистовике, демонстрирует проведенные расчеты и показания программы проверки преподавателю. Преподаватель проверяет ход решения, задает дополнительные вопросы (или задачи) для защиты работы. После успешной защиты и получает в тетрадь отметку о сдаче работы.

2. Последовательность выполнения работы

Часть 1.

Требуется вычислить выражение, задаваемое формулой 1 (см. выше).

Последовательность вычислений:

1. Перевести число А из системы счисления SA в десятичную (промежуточный шаг).
2. Перевести число В из системы счисления SB в десятичную (промежуточный шаг).
3. Перевести числа А и В из десятичной системы в систему с основанием plus.
4. Выполнить сложение в системе plus (т.е. по правилам системы plus).
5. Результат сложения перевести в десятичную систему, а затем в двоичную.
6. Вычесть из полученной в предыдущем пункте суммы число С в двоичной системе.

7. Разделить в двоичной системе полученную разность на число D. Деление выполнять до 5го разряда после запятой, 5ый разряд округлить с учетом значения шестого.

8. Перевести полученное округленное частное в десятичную систему.

9. Исходя из предположения, что полученное частное при переводе из десятичной системы в систему с основанием SX может оказаться бесконечной дробью, определить количество разрядов L после запятой в системе SX, необходимое для представления частного с точностью, равной половине веса младшего разряда. Для расчета L воспользоваться логарифмическим неравенством.

Если частное оказалось целое, то данный пункт не выполняется.

10. Осуществить перевод полученного частного из десятичной системы в систему SX с точностью до разряда с номером -L, учитывая округление от разряда -L-1. При округлении возможно распространение переноса. Напоминаем, что не стоит путать взаимосвязанные понятия «количество разрядов после запятой» и «номер разряда в дробной части»: количество может быть выражено только неотрицательным числом, а номер разряда в дробной части наоборот – число отрицательное.

Если в процессе перевода дробной части частного выяснилось, что в целевой системе эта дробная часть является конечной дробью и количество разрядов в ней оказалось меньше, чем рассчитанное L, то недостающие младшие разряды дополнить нулями.

Внимание! Отклонение от описанной последовательности выполнения работы **будет рассматриваться как ошибка.**

Часть 2.

Дано выражение, задаваемое формулой 2.

$$E_{10}[+]_{2-10} F_{10} \implies X_{2-10}, X_{10} \quad (2)$$

Необходимо перевести числа E и F из десятичной системы в двоично-десятичную и выполнить операцию сложения по правилам двоично-десятичной системы. Результат перевести в десятичную систему и выполнить проверку в десятичной системе.

При проведении вычислений следует помнить, что коррекция в младшей тетраде может вызывать цепочку коррекций в старших тетрадах.

При выполнении обеих частей работы **запрещается использовать** специальное программное обеспечение и сайты для перевода чисел в различные системы счисления и выполнения расчетов в этих системах счисления. Можно

использовать простой калькулятор для вычисления логарифмов. Студент, уличенный в применении запрещенных программных систем и сайтов, будет оштрафован снижением балла.

3. Содержание отчета (оформляется в тетради)

1. Решение части 1, выполненное по этапам, должно включать подробные вычисления, однозначно демонстрирующие ход решения.
2. Решение части 2 должно демонстрировать выполнение операции сложения в двоично-десятичной системе с проведением коррекции в тетрадах.

4. Пример

Часть 1.

Требуется вычислить значение выражения:

$$(563_9 [+]_3 1153_7 [-]_2 10000101_2) [/]_2 11001_2 => X_6$$

В первую очередь необходимо перевести числа 563_9 и 1153_7 в троичную систему счисления (далее — СС), делать это надо через десятичную систему, то есть сначала мы переведем эти числа в десятичную СС, а затем уже в троичную.

Для перевода числа из любой позиционной традиционной системы счисления в десятичную, необходимо представить его в виде полинома. Так $563_9 = 5 \cdot 9^2 + 6 \cdot 9^1 + 3 \cdot 9^0 = 5 \cdot 81 + 6 \cdot 9 + 3 = 462_{10}$.

Для перевода целого числа (или целой части числа) из десятичной системы счисления в произвольную позиционную традиционную систему счисления число надо делить на основание целевой системы счисления, записывая остатки в обратном порядке.

$$\begin{array}{r}
 462 \overline{) 3} \\
 \underline{462} \quad 154 \overline{) 3} \\
 \quad 0 \quad 153 \overline{) 3} \\
 \quad \quad 1 \quad 51 \overline{) 3} \\
 \quad \quad \quad 0 \quad 17 \overline{) 3} \\
 \quad \quad \quad \quad 15 \overline{) 3} \\
 \quad \quad \quad \quad \quad 2 \overline{) 3} \\
 \quad \quad \quad \quad \quad \quad 3 \overline{) 3} \\
 \quad \quad \quad \quad \quad \quad \quad 1
 \end{array}$$

Получим:

$$563_9 = 462_{10} = 122010_3$$

Аналогично:

$1153_7 = 430_{10} = 120221_3$ (в расчетах студента этот этап должен быть расписан **подробно**).

Далее, выполняем первое действие, то есть сложение в троичной СС, не забывая про перенос из младших разрядов в старшие.

$$\begin{array}{r}
 1111 \\
 + 122010 \\
 + 120221 \\
 \hline
 1020001
 \end{array}$$

То есть, $122010_3 [+]_3 120221_3 = 1020001_3$

Теперь переведем получившееся число в двоичную СС для выполнения дальнейших арифметических операций. Делаем это опять же через десятичную СС.

$1020001_3 = 892_{10} = 1101111100_2$ (в расчетах студента этот этап должен быть расписан **подробно**).

Выполняем второе действие, вычитание. Аналогично не забываем про заем из старшего разряда.

Получим:

$$\begin{array}{r}
 \begin{array}{c} \bullet \quad \bullet \quad \bullet \quad \bullet \\ 1101111100_2 \end{array} \\
 - \quad 10000101_2 \\
 \hline
 1011110111_2
 \end{array}$$

То есть $1101111100_2 [-]_2 10000101_2 = 1011110111_2$

Теперь, когда мы получили значение выражения в скобках, можем приступить к последнему арифметическому действию, то есть к делению получившегося числа на 11001_2 . Деление будем выполнять столбиком.

$$\begin{array}{r}
 \begin{array}{l}
 -1011110111 \\
 \hline
 11001 \\
 \hline
 -101100 \\
 \hline
 11001 \\
 \hline
 -100111 \\
 \hline
 11001 \\
 \hline
 -11101 \\
 \hline
 11001 \\
 \hline
 -100100 \\
 \hline
 11001 \\
 \hline
 -101100 \\
 \hline
 11001 \\
 \hline
 -100110 \\
 \hline
 11001 \\
 \hline
 -11010 \\
 \hline
 11001 \\
 \hline
 1
 \end{array}
 \quad \Bigg| \quad
 \begin{array}{l}
 11001 \\
 \hline
 11110,010111...
 \end{array}
 \end{array}$$

Результат деления — $11110,010111..._2$.

По заданию требуется произвести деление с точностью до 5-го разряда после запятой. Видим, что 6-ой разряд после запятой равен 1, поэтому выполняем округление 5-го разряда в большую сторону. При этом возникает распространение переноса.

Таким образом, округленный результат деления $11110,011_2$.

Если бы 6-ой разряд был равен 0, то округление производилось бы в меньшую сторону, т.е. все разряды после 5-го были бы отброшены.

Число, получившееся в результате округления двоичного частного, — это и есть окончательный результат вычисления заданного выражения, но в двоичной СС. А нам ответ нужен в шестеричной СС, следовательно, необходимо перевести результат в шестеричную систему счисления, для чего предварительно переведем его в десятичную СС, воспользовавшись уже известным полиномом.

$11110,011_2 = 30,375_{10}$ (в расчетах студента этот этап должен быть расписан подробно).

Исходя из предположения, что полученное частное при переводе в систему с основанием 6 может оказаться бесконечно большой дробью, нужно определить достаточное количество разрядов после запятой в системе с основанием 6, используя формулу (3).

$$K^{-L} > \frac{P^m}{2} > K^{-L-1}, \quad (3)$$

где K — основание целевой системы, P^m — вес младшего разряда частного в десятичной системе (m — отрицательная позиция младшего разряда дробной части числа), а L — искомое количество разрядов после запятой.

Эту формулу можно преобразовать к следующему виду (формула 4):

$$L < -\log_K \left(\frac{P^m}{2} \right) < L + 1 \quad (4)$$

Подставив все числа в формулу, получаем, что L у нас равно 4. Получается, в окончательном ответе у нас должно быть 4-ре знака после запятой.

Последнее, что нужно сделать — осуществить перевод полученного частного из десятичной системы в шестеричную с точностью до 4-го разряда. Чтобы это сделать, требуется перевести в нужную СС сначала целую, а затем и дробную часть.

Наше число — $30,375_{10}$.

Переведем целую часть в шестеричную СС.

$30_{10} = 50_6$ (в расчетах студента этот этап должен быть расписан подробно).

Для того, чтобы перевести дробную часть из десятичной СС в

шестеричную, нужно дробную часть умножать на 6 и выписывать целые части произведения. Затем каждую последующую дробную часть тоже умножаем на то же основание СС, и т.д., до тех пор, пока не будет достигнута требуемая точность.

$$0,375 \cdot 6 = 2,25$$

$$0,25 \cdot 6 = 1,5$$

$$0,5 \cdot 6 = 3,0$$

$$30,375_{10} = 50,213_6$$

Процесс перевода закончился раньше, чем рассчитанное количество знаков $L=4$. Такое может случиться, поскольку мы лишь предполагали, что в результате перевода может получиться бесконечная дробь.

Таким образом, мы получили окончательный ответ:

$$X_6 = 50,2130_6 \text{ — где явно обозначили младший ноль, поскольку } L=4.$$

Нам повезло в том, что переводимое число не оказалось бесконечной дробью в шестеричной системе. Однако, если бы реальное количество знаков после запятой у переводимого числа в шестеричной системе было бы больше рассчитанного нами L , то самый младший разряд (с номером $-L$) надо было бы округлить с учетом значения предыдущего разряда (с номером $-L-1$) по правилам округления шестеричной системы.

Часть 2

Нужно вычислить значение выражения

$$9507_{10} [+]_{2-10} 8003_{10} \Rightarrow X_{2-10}, X_{10}$$

Чтобы найти значение данного выражения, сначала нужно перевести числа 9507_{10} и 8003_{10} в двоично-десятичную систему счисления. Делается это путем перевода каждой цифры исходных чисел из десятичной СС в двоичную СС с записью получившихся результатов в виде двоичных тетрад.

$$9507_{10} = 1001\ 0101\ 0000\ 0111_{2-10}$$

$$8003_{10} = 1000\ 0000\ 0000\ 0011_{2-10}$$

Теперь получившиеся числа нужно сложить. Двоично-десятичные числа складывают по правилам двоичного сложения. Однако двоичное сложение может иногда давать неверный результат и тогда приходится выполнять коррекцию результата.

Коррекция заключается в прибавлении кода 0110 (число 6) к тем тетрадам результата в которых либо:

1. Получен код, входящий за диапазон двоично-десятичной системы счисления (например, 1100);

2. Был сформирован межтетрадный перенос в старшую тетраду.

$$\begin{array}{r}
 1001\ 0101\ 0000\ 0111_{2-10} \\
 + 1000\ 0000\ 0000\ 0011_{2-10} \\
 \hline
 + 0001\ 0001\ 0101\ 0000\ 1010_{2-10} \\
 0110 0110 \\
 \hline
 0001\ 0111\ 0101\ 0001\ 0000_{2-10}
 \end{array}$$

Как видим, пришлось сделать коррекцию два раза, чтобы получился правильный окончательный результат. Теперь нужно перевести ответ из двоично-десятичной СС в десятичную СС: каждая двоичная тетрада записывается как десятичная цифра.

$$X_{2-10} = 0001\ 0111\ 0101\ 0001\ 0000_{2-10}$$

$$X_{10} = 17510_{10}$$

5. Вопросы, возможные на защите

- Что такое СС вообще и позиционная СС в частности?
- Утверждается, что число задано в определенной СС. Подтвердить или опровергнуть это утверждение.
- Задано число в СС с основанием, равным степени двойки. Требуется перевести его в другую СС также с основанием, равным степени двойки, наиболее рациональным способом.
- Задано число в некоторой СС. Требуется наиболее рациональным образом разделить (умножить) это число на другое, равное степени основания.
- Где используется двоично-десятичная система?
- Почему величина коррекции в двоично-десятичной СС равна 6?
- Какова величина коррекции будет в другой К-Q-ичной системе, например, в двоично-девятеричной?
- Другие вопросы на тему работы.

2. Практическая работа № 2: выполнение операций вычитания в различных кодах

1. Постановка задачи

Часть 1. Произвести вычитание двух чисел в двоично-десятичном коде.

Часть 2. Произвести вычисления в обратном и дополнительном кодах для выражений вида: $A_2 - B_2$; $B_2 - A_2$; $-A_2 - B_2$.

Работа выполняется сначала на черновике. Все этапы вычислений должны быть подробно расписаны. Результаты расчетов вводятся в программу проверки. Программа проверки оценивает правильность полученных результатов в процентах. При 100% правильности студент оформляет решение на чистовике, демонстрирует проведенные расчеты и показания программы проверки преподавателю. Преподаватель проверяет ход решения, задает дополнительные вопросы (или задачи) для защиты работы. После успешной защиты студент получает в тетрадь отметку о сдаче работы.

2. Последовательность выполнения работы

Часть 1.

Необходимо перевести заданные числа из десятичной системы в двоично-десятичную и выполнить операцию вычитания в двоично-десятичной системе счисления. Результат перевести в десятичную систему.

При проведении вычислений следует помнить про заем из старших тетрадей и про их коррекцию.

Провести проверку сделанных вычислений в десятичной системе счисления.

Часть 2.

Требуется вычислить в обратном и дополнительном кодах выражения вида:

- а) $A_2 - B_2$;
- б) $B_2 - A_2$;
- в) $-A_2 - B_2$.

При вычислениях в соответствующих кодах операция вычитания заменяется операцией сложения с отрицательным числом. Для перевода отрицательного числа в обратный код в знаковом разряде ставится единица, остальные разряды инвертируются. Для перевода отрицательного числа в дополнительный код, оно сначала переводится в обратный, а затем к младшему разряду

арифметическим образом прибавляется 1. Положительные числа в обратном и дополнительном кодах выглядят также, как и в прямом. Перевод отрицательных чисел из обратного и дополнительного кода в прямой осуществляется по тем же правилам.

3. Содержание отчета (оформляется в тетради)

1. Решение части 1, включая операцию вычитания в двоично-десятичной системе с проведением коррекции в тетрадах.
2. Решение части 2, выполненное по этапам, включая подробное решение всех задач и для обратного, и для дополнительного кодов.

4. Пример

Часть 1.

Необходимо найти разность в двоично-десятичной системе:

$2515_{10} [-]_{2-10} 1597_{10}$ и сделать проверку в десятичной системе.

Первым шагом нужно перевести числа 2515_{10} и 1597_{10} в двоично-десятичную систему счисления. Делается это путем перевода каждой цифры исходных чисел из десятичной СС в двоичную СС с записью результатов перевода в виде двоичных тетрад.

$$2515_{10} = 0010\ 0101\ 0001\ 0101_{2-10}$$

$$1597_{10} = 0001\ 0101\ 1001\ 0111_{2-10}$$

Следующее действие — вычитание. Вычитание двоично-десятичных чисел производится по правилам двоичного вычитания с проводимой в необходимых случаях коррекцией результата. Такая коррекция заключается в вычитании кода 0110 из тех тетрад результата, которые использовали межтетрадный заем, или в которых получился код, выходящий за диапазон допустимых значений двоично-десятичной СС.

$$\begin{array}{r}
 \begin{array}{cccccccccccc}
 0 & 1 & 1 & 10 & 1 & 10 & 1 & 1 & 1 & 0 & 1 & 10 & 1
 \end{array} & \text{— заем} \\
 \begin{array}{r}
 \text{— } 0010\ 0101\ 0001\ 0101_{2-10} \\
 \underline{0001\ 0101\ 1001\ 0111_{2-10}} \\
 0000\ 1111\ 0111\ 1110_{2-10} \\
 \text{— } \underline{0110\ 0110\ 0110} \\
 1001\ 0001\ 1000_{2-10}
 \end{array}
 \end{array}$$

Получившийся окончательный ответ — $1001\ 0001\ 1000_{2-10}$

Теперь сделаем проверку в десятичной системе счисления:

$$2515_{10} - 1597_{10} = 918_{10}$$

$$1001\ 0001\ 1000_{2-10} = 918_{10}$$

Ответ верный.

Часть 2.

В этой части надо вычислить в обратном и дополнительном кодах каждое из выражений: $A_2 - B_2$; $B_2 - A_2$; $-A_2 - B_2$.

$$A_{10} = 134; B_{10} = 195$$

Чтобы выполнить необходимые арифметические операции, нужно перевести оба числа в двоичную СС, а затем в обратный и дополнительный коды.

$$A_2 = 0.10000110 \text{ — прям., обр.,} \\ \text{доп. код}$$

$$B_2 = 0.11000011 \text{ — прям., обр.,} \\ \text{доп. код}$$

$$-A_2 = 1.10000110 \text{ — прям. код}$$

$$-B_2 = 1.11000011 \text{ — прям. код}$$

$$-A_2 = 1.01111001 \text{ — обр. код}$$

$$-B_2 = 1.00111100 \text{ — обр. код}$$

$$-A_2 = 1.01111010 \text{ — доп. код}$$

$$-B_2 = 1.00111101 \text{ — доп. код}$$

В отчете студента этапы перевода должны быть расписаны подробно.

Теперь можно приступить к самому вычитанию.

$$A_2 - B_2:$$

Обратный код:

$$\begin{array}{r} 0.10000110 \\ + 1.00111100 \\ \hline 1.11000010 \end{array}$$

Дополнительный код:

$$\begin{array}{r} 0.10000110 \\ + 1.00111101 \\ \hline 1.11000011 \end{array}$$

Проверяем:

$$134_{10} - 195_{10} = -61_{10}$$

$$1.11000010_2 \text{ обр.} = 1.00111101_2 \text{ пр.} = -61_{10}$$

$$1.11000011_2 \text{ доп.} = 1.00111100_2 + 1 = 1.00111101_2 \text{ пр.} = -61_{10}$$

В отчете студента этапы переводов должны быть расписаны подробно.

$$B_2 - A_2:$$

Обратный код:

$$\begin{array}{r} 0.11000011 \\ + 1.01111001 \\ \hline \boxed{1} 0.00111100 \\ \quad \searrow \rightarrow 1 \\ \hline 0.00111101 \end{array}$$

Дополнительный код:

$$\begin{array}{r} 0.11000011 \\ + 1.01111010 \\ \hline \cancel{X} 0.00111101 \end{array}$$

Проверяем:

$$195_{10} - 134_{10} = 61_{10}$$

$$0.00111101_2 \text{ обр.} = 0.00111101_2 \text{ пр.} = 61_{10}$$

$$0.00111101_2 \text{ доп.} = 0.00111101_2 \text{ пр.} = 61_{10}$$

В отчете студента этапы переводов должны быть расписаны подробно.

$$-A_2 - B_2:$$

В этом примере, чтобы получился верный ответ, нужно увеличить разрядность, так как получается переполнение. В прямом коде мы добавляем к числам слева после знака незначащий ноль, который становится единицей в обратном и дополнительном кодах, если переводимое число отрицательное.

Обратный код:

$$\begin{array}{r} 1.101111001 \\ + 1.100111100 \\ \hline \boxed{1}1.010110101 \\ \quad \quad \quad \rightarrow 1 \\ \hline 1.010110110 \end{array}$$

Дополнительный код:

$$\begin{array}{r} 1.101111010 \\ + 1.100111101 \\ \hline \cancel{X}1.010110111 \end{array}$$

Проверяем:

$$-195_{10} - 134_{10} = -329_{10}$$

$$1.010110110_2 \text{ обр.} = 1.101001001_2 \text{ пр.} = -329_{10}$$

$$1.010110111_2 \text{ доп.} = 1.101001000 + 1 = 1.101001001_2 \text{ пр.} = -329_{10}$$

В отчете студента этапы переводов должны быть расписаны подробно.

5. Вопросы, возможные на защите

- Где используется двоично-десятичная система?
- Почему величина коррекции в двоично-десятичной СС равна 6?
- Какова величина коррекции будет в другой К-Q-ичной системе, например, в двоично-девятеричной?
- Что такое разрядная сетка?
- Что такое переполнение разрядной сетки?
- Зачем нужны обратный и дополнительный коды?
- Задана двоичная запись, перевести ее в десятичную СС.
- Выполнить вычитание двух отрицательных чисел в обратном или дополнительном коде.
- Существуют ли обратный и дополнительный коды в других позиционных традиционных системах счисления? Привести примеры.
- Что такое модифицированные обратный и дополнительный коды?
- Другие вопросы по теме работы.

3. Практическая работа № 3: перевод вещественных чисел и выполнения арифметических операций над ними в рамках стандарта IEEE754

1. Постановка задачи

Часть 1: перевести заданное число из десятичной системы в формат половинной точности стандарта IEEE754. Результат записать в свернутом виде при помощи 16-теричной системы.

Часть 2: перевести заданное число из формата половинной точности стандарта IEEE754 в десятичную систему с точностью до 10го знака после запятой.

Часть 3: представить заданные числа в формате половинной точности стандарта IEEE754 и выполнить их сложение. При необходимости результат нормализовать. Ответ записать в свернутом виде при помощи 16-теричной системы.

Часть 4: представить заданные числа в формате одинарной точности и выполнить их умножение. При необходимости результат нормализовать. Ответ записать в свернутом виде при помощи 16-теричной системы.

Работа по всем частям выполняется сначала на черновике. Все этапы вычислений должны быть подробно расписаны. Результаты расчетов вводятся в программу проверки. Программа проверки оценивает правильность полученных результатов в процентах. При 100% правильности студент оформляет решение на чистовике, демонстрирует проведенные расчеты и показания программы проверки преподавателю. Преподаватель проверяет ход решения, задает дополнительные вопросы (или задачи) для защиты работы. После успешной защиты студент получает в тетрадь отметку о сдаче работы.

Исходные данные для всех четырех частей практической работы ориентированы на работу с нормализованными (согласно требованиям стандарта IEEE754) числами. Однако в дополнительных заданиях (на защиту) могут попадаться числа, не нормализуемые в рамках конкретного формата.

2. Последовательность выполнения работы

Часть 1.

Решение начинается с того, что заданное десятичное число переводится в двоичную систему счисления. Далее это число нормализуется по правилам нормализации стандарта IEEE754, т.е. представляется в виде произведения мантиссы и порядка (экспоненты), где мантисса лежит в диапазоне $[1,2)$, а порядок является степенью двойки. Этот порядок называется истинным порядком числа. Но в формате половинной точности порядок записывается в смещенном коде, причем смещение равно 15. Поэтому к истинному порядку числа следует

прибавить величину смещения и получить, таким образом, смещенный порядок. Смещенный порядок следует перевести в двоичную систему и дополнитьзначащими старшими нулями до получения числа длиной 5 бит.

Теперь следует записать полученные результаты в разрядную сетку формата половинной точности, в которой имеющиеся 2 байта распределяются между полями числа следующим образом (рис. 1).

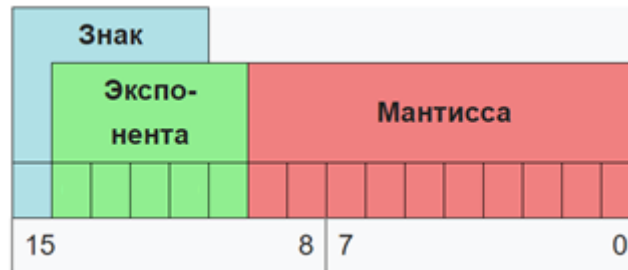


Рис.1 Разрядная сетка формата половинной точности

Знак числа — 0 или 1 — записывается в 15-ый разряд. Смещенный порядок — разряды с 14-го по 10-ый. Дробная часть мантииссы — в разряды с 9-го по нулевой.

При этом следует помнить, что целая часть нормализованного числа по стандарту IEEE754 всегда равна единице, поэтому ее в целях экономии памяти и повышения точности явным образом не хранят в разрядной сетке. Но эта единица всегда подразумевается строго между 9-ым и 10-ым разрядами и всегда участвует в выполнении арифметических операций.

Теперь осталось только перевести шестнадцатиразрядную двоичную запись в более компактное шестнадцатеричное число, для чего достаточно воспользоваться разбиением двоичной записи на тетрады.

Часть 2.

Задача, решаемая в этой части, по сути, обратна предыдущей задаче. Имеется 16-теричная запись числа в формате половинной точности стандарта IEEE754. Требуется перевести его в десятичную систему с точностью до 10-го знака после запятой.

Начинаем решение с того, что разворачиваем 16-теричную запись в двоичную, заменяя 16-теричные цифры на двоичные тетрады. На полученную двоичную запись накладываем шаблон формата половинной точности (рис. 1).

Выясняем значение отдельных полей: знака, смещенного порядка, дробной части мантииссы. Переводим смещенный порядок в десятичную систему, после чего вычитаем из него величину смещения (равна 15). В результате получаем значение истинного порядка. Далее берем дробную часть мантииссы и добавляем к ней подразумеваемую целую часть, получая, таким образом, всю мантииссу

числа.

Денормализуем мантиссу, сдвигая запятую вправо (или влево) в зависимости от знака и модуля истинного порядка. Далее приписываем нашему числу знак в зависимости от значения в старшем разряде числа в формате половинной точности.

В результате имеем обыкновенное вещественное число в двоичной системе, которое мы можем перевести в десятичную систему при помощи уже известного нам полинома.

Часть 3

Сложение двух чисел в формате половинной точности начинается с перевода слагаемых в этот формат способом, который уже был описан ранее (см. последовательность выполнения работы для части 1). Единственное отличие заключается в том, что для слагаемых не нужно выполнять последний шаг — свертку в 16-теричное число. Предположим, что сказанное уже было сделано, и перед нами два слагаемых, представленных в виде двухбайтных двоичных чисел формата половинной точности.

Далее необходимо сравнить смещенные порядки слагаемых. Если порядки равны, то можно переходить к сложению мантисс.

Если порядки не равны, то требуется провести выравнивание порядков. При этом меньший порядок выравнивается в сторону большего, т.е. порядок меньшего числа становится равным порядку большего числа, а мантисса меньшего числа сдвигается вправо (денормализуется) на количество разрядов, равное разности порядков. При денормализации следует помнить, что подразумеваемая целая часть при первом же сдвиге вправо превращается в старший разряд дробной части мантиссы. Также следует иметь в виду то, что в результате денормализации может теряться точность, т.к. единицы, стоящие в младших разрядах, при сдвиге вправо могут выпасть за пределы разрядной сетки.

Мантиссы хранятся в прямом коде, но складываются в дополнительном (перед сложением осуществляется перевод). В процессе сложения следует помнить про подразумеваемую целую часть и про знак слагаемых. Если слагаемое не было денормализовано, то целая часть равна 1, если было — то 0.

Результат сложения может быть как нормализованным, так и ненормализованным. Признаком ненормализованного результата является целая часть суммы, равная по абсолютной величине двойке. В этом случае результат нужно нормализовать, т.е. сдвинуть мантиссу вправо на один разряд, а порядок суммы увеличить на единицу.

Далее следует записать полученную сумму в разрядную сетку формата половинной точности (рис. 1), перенеся знак старший разряд, и сделав целую часть подразумеваемой.

Последний шаг – это свертка к 16-теричной записи.

Часть 4

Формат одинарной точности (рис. 2) отличается от формата половинной точности лишь количественно. Всего на запись числа отводится 4 байта. На порядок отводится 1 байт, поэтому величина смещения будет равна 127. Подразумеваемая целая часть находится между 22-ым и 23-им разрядами. На дробную часть мантиссы отводятся разряды с 22-го по нулевой.

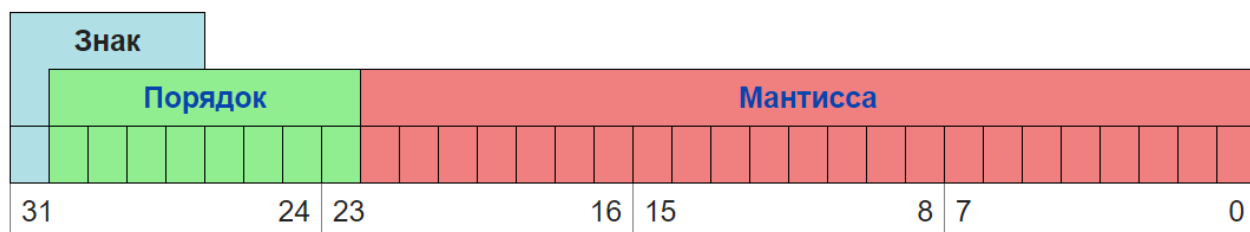


Рис. 2 Разрядная сетка формата одинарной точности

Умножение двух чисел в формате одинарной точности начинается с перевода множителей в этот формат. Последовательность действий при переводе аналогична процедуре, рассмотренной в части 1, с поправкой на другую величину смещения порядка. Предположим, что сказанное уже было сделано, и перед нами два множителя, представленных в виде четырехбайтных двоичных чисел формата одинарной точности.

Сам процесс умножения будет следующим. Порядки множителей необходимо сложить. Однако нужно помнить, что они представлены в коде со смещением, поэтому при сложении порядков у суммы получается двойное смещение. Следовательно, для получения правильного значения суммы порядков нужно после их сложения вычесть одно смещение $127_{10} = 1111111_2$.

Далее мантиссы перемножаются (с учетом их подразумеваемых целых частей). Очевидным образом выбирается знак произведения – если знаки множителей разные, то знак произведения равен 1, т.е. оно отрицательное.

Произведение может получиться как нормализованным, так и ненормализованным. Признаком ненормализованного произведения является наличие целой части большей единицы (2 или 3). В этом случае результат нужно нормализовать, т.е. сдвинуть мантиссу вправо на один разряд, а порядок произведения

увеличить на единицу.

Далее следует записать полученный результат в разрядную сетку формата одинарной точности (рис. 2), не забыв сделать целую часть подразумеваемой.

Последний шаг – это свертка к 16-теричной записи.

3. Содержание отчета (оформляется в тетради)

1. Решение части 1.
2. Решение части 2.
3. Решение части 3.
4. Решение части 4.

Каждое решение должно быть максимально подробно расписано, чтобы можно было проверить его ход.

4. Пример

Часть 1.

Предположим, что задано число $138,134_{10}$, которое необходимо представить в формате половинной точности.

Переводим целую часть в двоичную систему любым допустимым способом: $138_{10} = 10001010_2$.

После нормализации целая часть будет записана в виде:

$$1,0001010_2 * 2_{10}^7$$

Поскольку на запись мантиисы в формате половинной точности выделяется 10 разрядов, то получается, что для дробной части исходного числа в мантиисе остается $10 - 7 = 3$ разряда. Поэтому переведем дробную часть исходного числа в двоичную систему с точностью до третьего разряда после запятой.

$$0,134 * 2 = 0,268$$

$$0,268 * 2 = 0,536$$

$$0,563 * 2 = 1,072$$

$$0,134_{10} \approx 0,001_2$$

Таким образом, потеря точности составит $0,134 - 2^{-3} = 0,009$.

Добавляем вычисленные три разряда к нормализованной записи целой части, получаем:

$$1,0001010001_2 * 10_2^7$$

Зная величину смещения порядка для формата половинной точности (она всегда равна 15), рассчитываем значение смещенного порядка:

$$7 + 15 = 22 = 10110_2$$

Теперь запишем значения отдельных полей в разрядную сетку формата

половинной точности, не забыв сделать целую часть подразумеваемой (рис. 3).

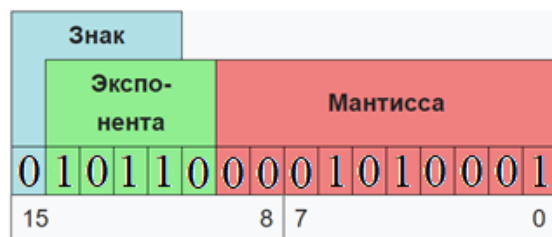


Рис. 3 Запись числа в формате половинной точности

Осталось только преобразовать эту запись к компактному шестнадцатеричному виду: $0101\ 1000\ 0101\ 0001_2 = 5851_{16}$

Часть 2.

Пусть дано число $BB5D_{16}$, известно, что это число в формате половинной точности, требуется перевести его в десятичную систему.

Развернем это число в двоичную запись и наложим на эту запись формат половинной точности (рис.4).

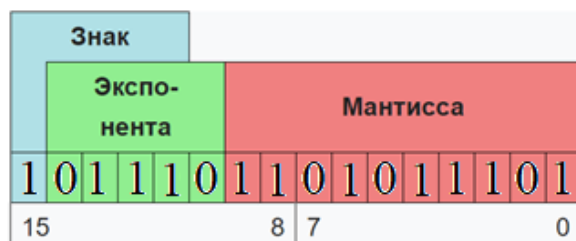


Рис. 4 Двоичное представление исходного числа с наложением формата половинной точности

Теперь нам видны значения отдельных полей. Видим, что число отрицательное. Смещенный порядок равен $01110_2 = 14_{10}$.

Мантисса с учетом подразумеваемой целой части равна $1,1101011101_2$.

Вычисляем истинный порядок, вычитая смещение: $14 - 15 = -1$.

Таким образом, перед нами число:

$$-1,1101011101_2 * 2_{10}^{-1} = -0,11101011101_2 = -0,92041015625_{10}$$

Часть 3.

Допустим, требуется вычислить в формате половинной точности следующее выражение: $324_{10} + 123_{10}$. Результат представить в сокращенной шестнадцатеричной форме.

Переводим 324 в двоичную систему: 101000100_2 .

Нормализуем: $1,01000100_2 * 2_{10}^8$ – истинный порядок равен 8.

Вычисляем смещенный порядок: $8 + 15 = 23 = 10111_2$.

Записываем вычисленные поля в разрядную сетку формата половинной точности (рис. 5).

З.М	Порядок							Мантисса						
0	1	0	1	1	1	0	1	0	0	0	1	0	0	0

1,

Рис. 5 Запись числа $324,0_{10}$ в формате половинной точности

Аналогичные преобразования проводим со вторым слагаемым, получаем запись, показанную на рис. 6.

З.М	Порядок							Мантисса						
0	1	0	1	0	1	1	1	1	0	1	1	0	0	0

1,

Рис. 6 Запись числа $123,0_{10}$ в формате половинной точности

На обоих рисунках подразумеваемая единица показана снизу разрядной сетки, а цветом выделены разряды, которые остались пустыми (т.е. равными нулю), поскольку дробная часть исходных слагаемых равна нулю.

Выполняем операцию сравнения порядков. Порядок первого слагаемого на 2 больше, чем порядок второго: $10111_2 - 10101_2 = 10_2$.

Выравниваем порядок меньшего слагаемого в сторону большего, при этом порядок меньшего числа становится равен порядку большего, а мантисса меньшего сдвигается на 2 разряда вправо. При этом подразумеваемая целая часть явным образом входит в разряды мантиссы, а справа пропадают 2 разряда, что может приводить к потере точности, но в данном случае не привело (рис. 7).

	З.М	Порядок							Мантисса						
	0	1	0	1	1	1	0	1	0	0	0	1	0	0	0
+															
	З.М	Порядок							Мантисса						
	0	1	0	1	1	1	0	1	1	1	1	0	1	1	0

0,

Рис. 7 Слагаемые в результате выравнивания порядков

Далее в процессоре мантиссы с учетом их знака складываются в дополнительном коде, а сумма нормализуется и снова переводится в прямой код для хранения в памяти. В нашем примере оба слагаемых положительные, поэтому для простоты сложим их в прямом коде (нормализация результата сложения не

потребовалась, рис. 8).

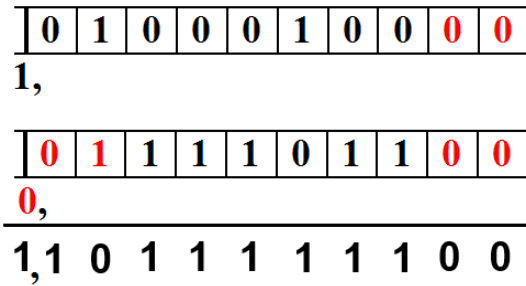


Рис. 8 Процесс сложения мантисс

Запишем результат сложения в разрядную сетку формата половинной точности и свернем в 16-теричную форму (рис. 9).

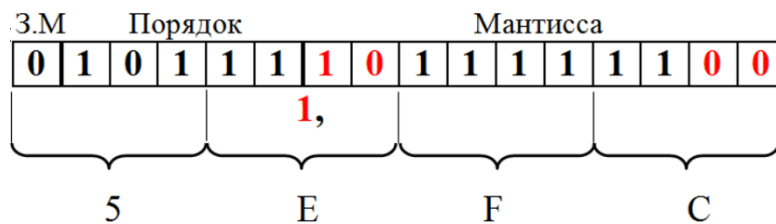


Рис. 9 Окончательный результат сложения

Часть 4

Предположим, требуется перемножить следующие числа в формате одинарной точности: $A = -12,125_{10}$ и $B = 9,5_{10}$.

Переведем их в двоичную систему.

$$-12,125_{10} = -1100,001_2;$$

$$9,5_{10} = 1001,1_2$$

Нормализуем и определим истинный порядок.

$$-1100,001_2 = -1,100001_2 * 2_{10}^3;$$

$$1001,1_2 = 1,0011 * 2_{10}^3$$

Порядок у обоих множителей получился одинаковый.

$$\text{Вычислим смещенный порядок: } 3_{10} + 127_{10} = 130_{10} = 10000010_2.$$

Запишем множители в формате половинной точности.

Число A показано на рис. 10, а число B на рис. 11.

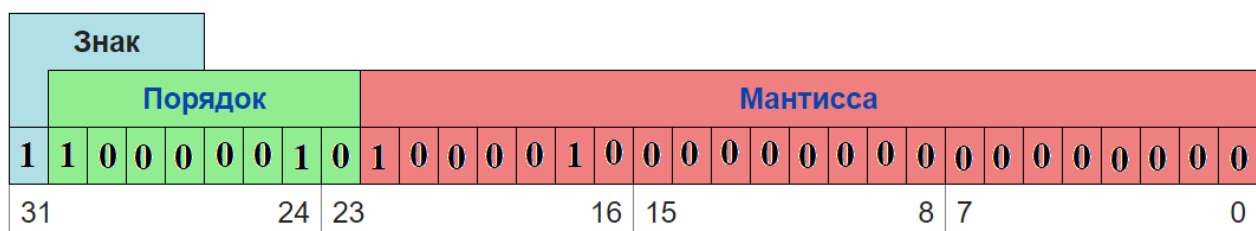


Рис. 10 Представление числа A в формате одинарной точности

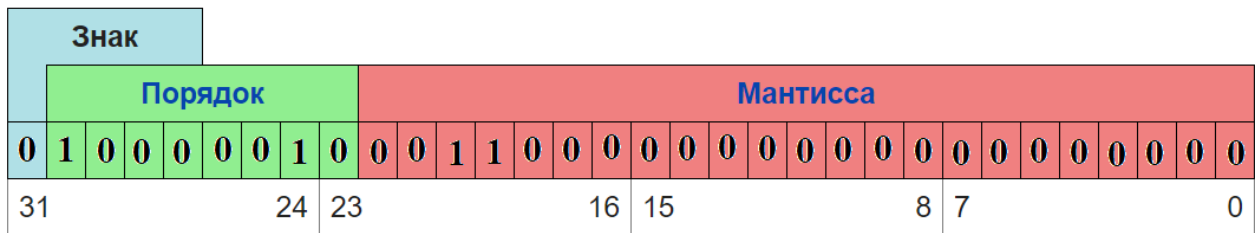


Рис. 11 Представление числа В в формате одинарной точности.

Известно, что при умножении порядки складываются. Для удобства произведем дальнейшие манипуляции с порядками в десятичной системе.

$$130 + 130 = 260$$

Однако, мы складывали порядки в коде со смещением. Поэтому, чтобы получился верный результат, необходимо вычесть одно лишнее смещение.

$$260 - 127 = 133$$

Таким образом, мы получили порядок произведения в смещенном коде.

Осталось перемножить мантиссы и нормализовать результат (рис. 12). Мантиссы перемножаются с учетом подразумеваемых целых частей. Знак произведения будет 1, если знаки множителей разные.

×	1, 1 0 0 0 0 1	еще 17 нулей
	1, 0 0 1 1 0 0	еще 17 нулей
<hr/>		
	1 1 0 0 0 0 1	
+	1 1 0 0 0 0 1	
	1 1 0 0 0 0 1	
<hr/>		
	1, 1 1 0 0 1 1 0 0 1 1 0 0	еще 34 нуля

Рис. 12 Процесс умножения мантиссы

Вычислены все составляющие результата, можно записать его в разрядную сетку формата одинарной точности, опуская подразумеваемую целую часть (рис. 13).

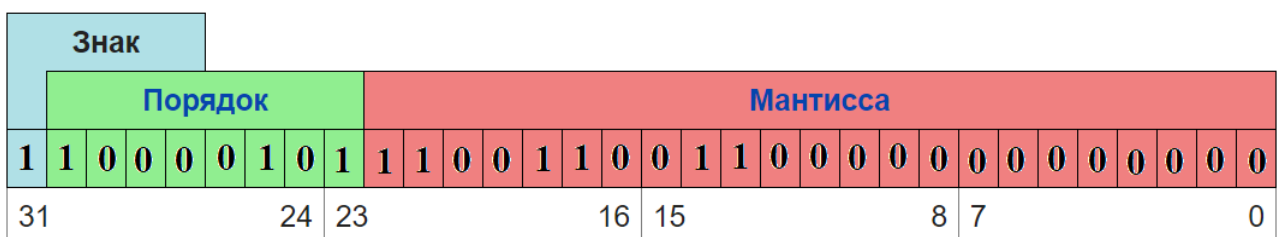


Рис. 13 Произведение, записанное в формате одинарной точности

Или то же самое в сокращенной шестнадцатеричной форме.
 $1100\ 0010\ 1110\ 0110\ 0110\ 0000\ 0000\ 0000_2 = C2E66000_{16}$

5. Вопросы, возможные на защите

- Что такое код со смещением, и почему он используется для представления порядка числа в стандарте IEEE754?
- Как величина смещения зависит от количества имеющихся разрядов?
- Что такое нормализованное число? Что такое нормализованное число в стандарте IEEE754 и для чего были сделаны эти изменения?
- Почему целая часть числа явным образом не хранится в разрядной сетке при использовании стандарта IEEE754?
- Как вычислить минимальное (максимальное) нормализованное (ненормализованное) число, которое можно представить в формате половинной (одинарной) точности?
- Задано небольшое число в десятичной системе, например 1_{10} . Требуется записать его в формате половинной точности и привести к 16-теричному виду. Также возможна и обратная задача.
- В виде 16-теричной записи задано ненормализованное число в формате половинной точности, перевести его в десятичную систему.
- Заданы два числа в 16-теричной записи формата половинной точности. Требуется по их виду определить, какое из них больше.
- Сформулировать правила выполнения арифметических операций над числами с плавающей точкой.
- Другие вопросы по теме работы.

РАЗДЕЛ №2: ЛОГИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

4. Практическая работа № 4: анализ логических схем в игровой форме

1. Постановка задачи

Используя игровую программу-тренажер для изучения работы логических элементов «Логика» [1] (рис. 14) решить все предлагаемые задачи, набрав положительное число очков.

Обучающий момент логической игры заключается в том, что необходимо понять, как работает комбинационная схема, управляющая движением платформ, чтобы, задавая правильные управляющие воздействия, постепенно транспортировать кристалл из верхней точки в нижнюю, не разбив его.

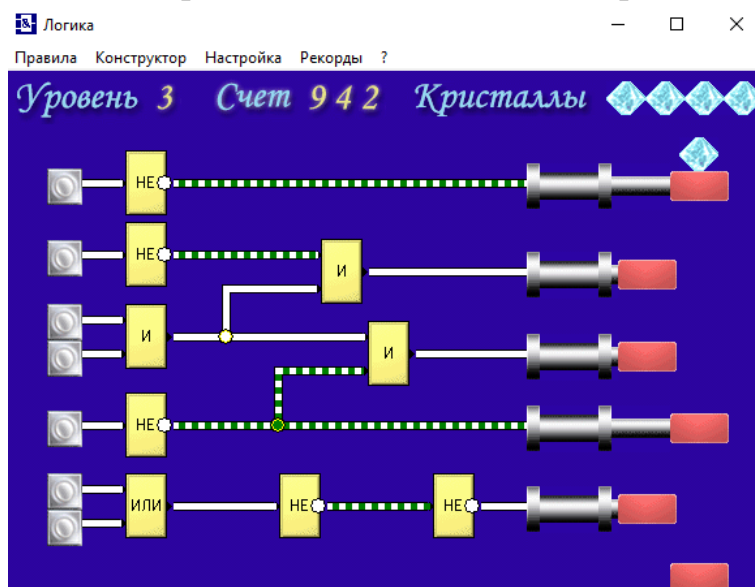


Рис. 14 Интерфейс программы «Логика»

От количества набранных очков зависит оценка работы.

Работа выполняется на компьютере. В случае выполнения работы в аудитории, результат демонстрируется преподавателю на экране компьютера и количество набранных баллов заносится в тетрадь. В случае выполнения работы вне аудитории, преподавателю демонстрируется фотография (в электронном виде) экрана с набранными очками и студенческого билета (на одной фотографии). Качество фотографии должно позволять распознать как текст в студенческом билете, так и набранные очки на экране компьютера. Затем набранные баллы заносятся в тетрадь.

2. Последовательность выполнения работы

1. Запустить программу.
2. Пройти все уровни, набрав максимально возможное число очков (больше 0).
3. Продемонстрировать результаты преподавателю.

3. Содержание отчета (в случае выполнения работы вне аудитории)

Фотография в электронном виде, содержащая студенческий билет и экран компьютера с набранными очками.

5. Практическая работа №5: построение комбинационных схем, реализующих СДНФ и СКНФ заданной логической функции от 4-х переменных

1. Постановка задачи

Логическая функция от четырех переменных задана в 16-теричной векторной форме. Восстановить таблицу истинности. Записать формулы СДНФ и СКНФ. Построить комбинационные схемы СДНФ и СКНФ в лабораторном комплексе, используя общий логический базис. Протестировать работу схем и убедиться в их правильности. Подготовить отчет о проделанной работе и защитить ее.

2. Последовательность выполнения работы

1. Перед выполнением работы ознакомиться с приложениями 1-3.
2. Запустить лабораторный комплекс и получить персональные исходные данные для этой работы. Для этого требуется в лабораторном комплексе установить правильное название группы и ФИО студента. Рекомендуется зафиксировать личный код. Логическая функция от четырех переменных, заданная в 16-теричной форме, будет показана на индикаторе F1.
3. На черновике восстановить таблицу истинности, записать формулы СДНФ и СКНФ в общем базисе.
4. Последовательно собрать в лабораторном комплексе комбинационные схемы, реализующие СДНФ и СКНФ в общем логическом базисе. Подключить входы и выходы схем (по очереди) к устройству проверки.
5. Запустить процесс тестирования схем, чтобы убедиться в правильности их работы. В случае обнаружения ошибки (загорятся индикаторы ошибки), найти ее и исправить.
6. Продемонстрировать правильность работы схем преподавателю, для чего необходимо при нем запустить процесс тестирования. Получить разрешение преподавателя на оформление отчета.
7. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).
8. Защитить практическую работу, отвечая на дополнительные вопросы, и получить роспись преподавателя в тетради учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.
3. Постановка задачи и персональный вариант.

4. Восстановленная таблица истинности.

5. Формулы СДНФ и СКНФ.

6. Схемы, реализующие СДНФ и СКНФ в общем логическом базисе (должны быть приведены фотографии экрана, на которых видны: группа, ФИО студента, индикаторы исходных данных, разработанные схемы с подключением к устройству проверки, а также положительный результат проверки).

7. Выводы.

8. Список информационных источников.

4. Пример

Предположим, что в соответствии с вариантом функция, заданная в 16-теричной форме имеет следующий вид:

$$F(a,b,c,d) = ACF8_{16}$$

Преобразуем ее в двоичную запись: $1010\ 1100\ 1111\ 1000_2$ – получили столбец значений логической функции, который необходим для восстановления полной таблицы истинности (см. табл.2).

Таблица 2

a	b	c	d	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Запишем формулу СДНФ, для чего рассмотрим наборы значений

переменных, на которых функция равна единице. Для каждого набора отвечаем на вопрос: каким образом при помощи конъюнкции переменных, принимающих значения из данного набора, можно получить единичное значения функции? Очевидно, что переменные, равные нулю, надо взять с отрицанием, а переменные, равные единице, без отрицания. В результате мы получим множество совершенных конъюнкций, объединив которые через дизъюнкцию образуем формулу СДНФ (формула 5).

$$F_{\text{сднф}} = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} \cdot d + \\ + a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + a \cdot \bar{b} \cdot \bar{c} \cdot d + a \cdot \bar{b} \cdot c \cdot \bar{d} + a \cdot \bar{b} \cdot c \cdot d + a \cdot b \cdot \bar{c} \cdot \bar{d} \quad (5)$$

Запишем формулу СКНФ, для чего рассмотрим наборы значений переменных, на которых функция равна нулю. Для каждого набора отвечаем на вопрос: каким образом при помощи дизъюнкции переменных, принимающих значения из данного набора, можно получить нулевое значения функции? Очевидно, что переменные, равные единице, надо взять с отрицанием, а переменные, равные нулю, без отрицания. В результате мы получим множество совершенных дизъюнкций, объединив которые через конъюнкцию образуем формулу СКНФ (формула 6).

$$F_{\text{скнф}} = (a + b + c + \bar{d}) \cdot (a + b + \bar{c} + \bar{d}) \cdot (a + \bar{b} + \bar{c} + d) \cdot \\ \cdot (a + \bar{b} + \bar{c} + \bar{d}) \cdot (\bar{a} + \bar{b} + c + \bar{d}) \cdot (\bar{a} + \bar{b} + \bar{c} + d) \cdot (\bar{a} + \bar{b} + \bar{c} + \bar{d}) \quad (6)$$

Построим в лабораторном комплексе комбинационные схемы, реализующие СДНФ и СКНФ рассматриваемой функции в общем логическом базисе, протестируем их работу и убедимся в их правильности (рис. 15,16).

На схеме СДНФ (рис.15) в целях размещения всей схемы в пределах одного экрана объединяющая дизъюнкция разбита на две части. Аналогично была разбита объединяющая конъюнкция на рис.16.

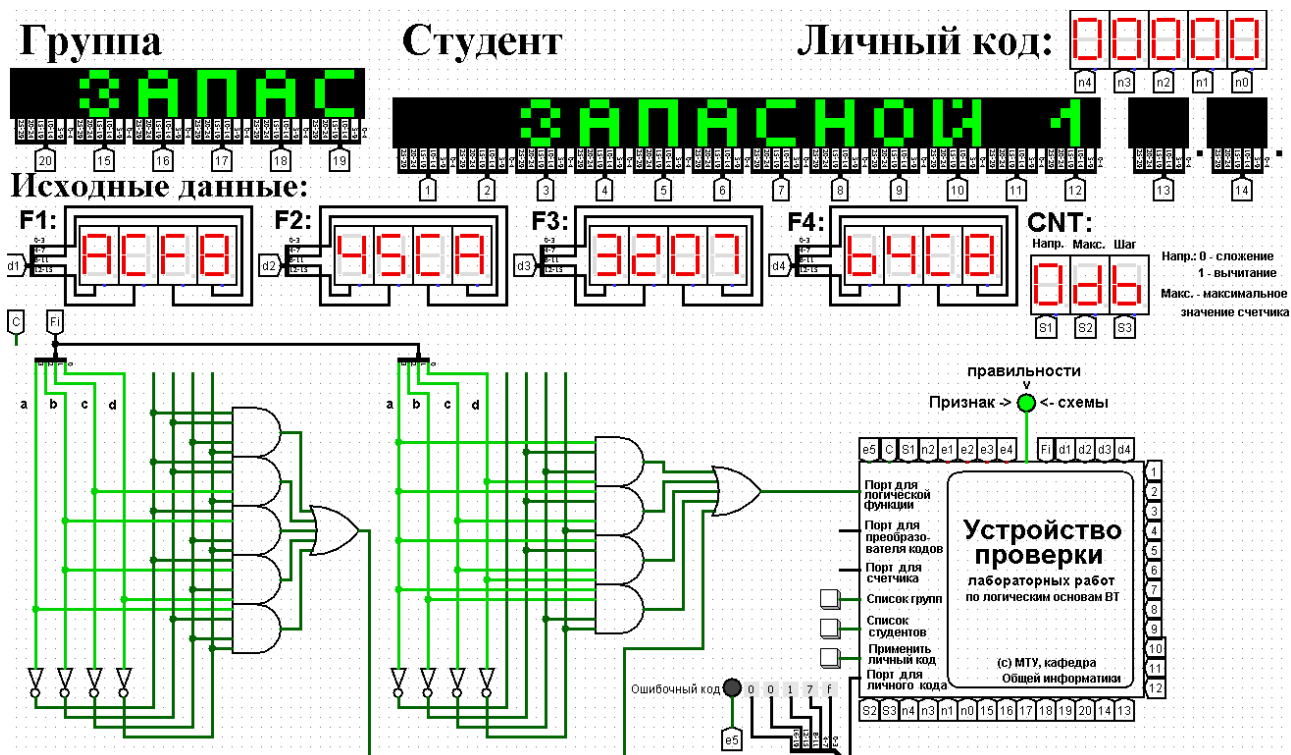


Рис.15 Тестирование схемы СДНФ

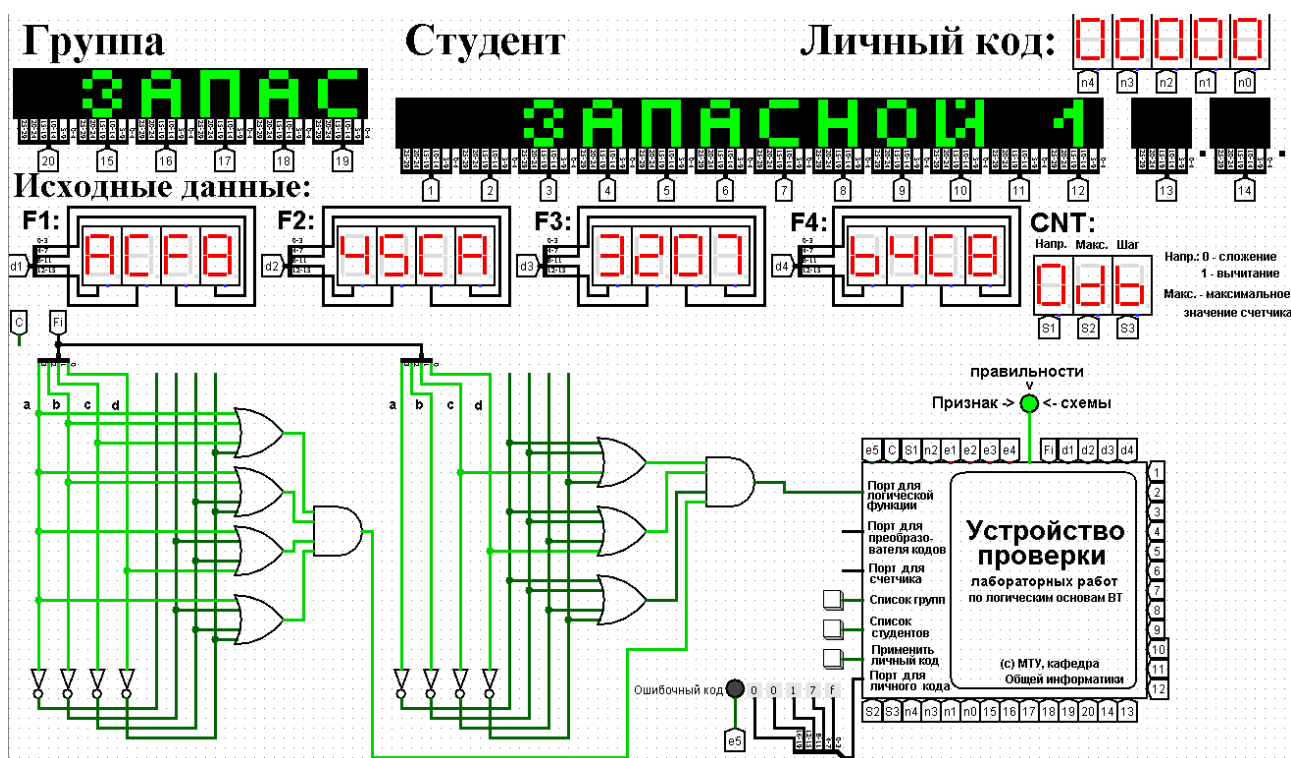


Рис.16 Тестирование схемы СКНФ

Тестирование показало, что все схемы работают правильно.

5. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое комбинационная схема?
- Что такое макстерм, минтерм, совершенный дизъюнкт, совершенный конъюнкт?
- Что такое СДНФ (СКНФ) и зачем они используются?
- Как построить СДНФ (СКНФ) по таблице истинности?
- Как построить СДНФ (СКНФ) путем эквивалентных логических преобразований?
- Как изменится текущая схема, если в таблице истинности сделать такие-то изменения?
- Как изменится таблица истинности, если в схеме сделать такие-то изменения?
- Чему равна некоторая логическая функция, если ее аргументами будут СДНФ и СКНФ?
- Другие вопросы по теме работы.

6. Практическая работа №6: построение комбинационных схем, реализующих МДНФ и МКНФ заданной логической функции от 4-х переменных в базисах И-НЕ, ИЛИ-НЕ

1. Постановка задачи

Логическая функция от четырех переменных задана в 16-теричной векторной форме. Восстановить таблицу истинности. Минимизировать логическую функцию при помощи карт Карно и получить формулы МДНФ и МКНФ в общем базисе. Перевести МДНФ и МКНФ в базисы «И-НЕ» и «ИЛИ-НЕ» (каждую минимальную форму в два базиса). Построить комбинационные схемы для приведенных к базисам формул МДНФ и МКНФ в лабораторном комплексе, используя только логические элементы, входящие в конкретный базис. Протестировать работу схем и убедиться в их правильности. Подготовить отчет о проделанной работе и защитить ее.

2. Последовательность выполнения работы

1. Перед выполнением работы ознакомиться с приложениями 1-3.

2. Запустить лабораторный комплекс и получить персональные исходные данные для этой работы. Для этого требуется установить правильное название группы и ФИО студента. Рекомендуется зафиксировать личный код. Логическая функция от четырех переменных, заданная в 16-теричной форме, будет показана на индикаторе F1.

3. На черновике восстановить таблицу истинности, провести минимизацию и получить формулы МДНФ и МКНФ в общем базисе, привести их к базисам «И-НЕ» и «ИЛИ-НЕ».

4. Последовательно собрать в лабораторном комплексе комбинационные схемы, реализующие МДНФ и МКНФ в требуемых логических базисах. Подключить входы и выходы схем к устройству проверки.

5. Протестировать работу схем и убедиться в их правильности. В случае обнаружения ошибки (загорятся индикаторы ошибки), найти ее и исправить.

6. Продемонстрировать правильность работы схем преподавателю, для чего необходимо при нем запустить процесс моделирования работы, и получить разрешение преподавателя на оформление отчета.

7. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).

8. Защитить практическую работу, отвечая на дополнительные вопросы, и получить роспись преподавателя в тетрадь учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.
3. Постановка задачи и персональный вариант.
4. Восстановленная таблица истинности.
5. Минимизация логической функции при помощи карт Карно.
6. Приведение МДНФ и МКНФ к базисам «И-НЕ» и «ИЛИ-НЕ».
7. Схемы, реализующие МДНФ и МКНФ в требуемых логических базах (должны быть приведены фотографии экрана, на которых видны: группа, ФИО студента, индикаторы исходных данных, разработанные схемы с подключением к устройству проверки, а также положительный результат проверки).
8. Выводы.
9. Список информационных источников.

4. Пример

Предположим, что имеется следующая функция, заданная в 16-теричной форме:

$$F(a,b,c,d) = ACF8_{16}$$

После восстановления таблицы истинности будет получена табл. 1 (см. практическую работу №4).

Далее построим МДНФ заданной функции. Для этого воспользуемся методом карт Карно. Разместим единичные значения функции на карте Карно, предназначенной для минимизации функции от четырех переменных (рис.17). Напоминаем, что местоположение значения функции на карте в каждом конкретном случае определяется координатами, которые представляют собой комбинацию значений переменных.

Пустые клетки карты на рис. 17 содержат нулевые значения функции, которые при построении МДНФ в целях повышения наглядности можно на карту не наносить.

$\begin{smallmatrix} cd \\ ab \end{smallmatrix}$	00	01	11	10
00	1			1
01	1	1		
11	1			
10	1	1	1	1

Рис. 17 Карта Карно, заполненная для построения МДНФ

Теперь необходимо выделить интервалы, на которых функция сохраняет свое единичное значение. Размер интервалов должен быть равен степени двойки. При выделении интервалов надо помнить, что карта Карно представляет собой развертку пространственной фигуры, поэтому некоторые интервалы могут разрываться краями карты. Интервалы выделяются так, чтобы выполнялись следующие правила:

- интервалы могут пересекаться, но каждый интервал должен иметь хотя бы одну клетку, принадлежащую только ему (не должно быть интервалов, полностью поглощенных другими интервалами);
- сами интервалы должны быть как можно больше (но без нарушения первого правила);
- при этом общее количество интервалов должно быть как можно меньше;

Подробное описание процесса минимизации можно найти в лекциях и рекомендуемой литературе.

Результат выделения интервалов для рассматриваемого примера показан на рис.18.

$\begin{smallmatrix} cd \\ ab \end{smallmatrix}$	00	01	11	10
00	1			1
01	1	1		
11	1			
10	1	1	1	1

Рис. 18 Результат выделения интервалов для МДНФ

Далее запишем формулу МДНФ, для чего последовательно рассмотрим каждый из интервалов. Для каждого интервала запишем минимальную конъюнкцию, куда будут входить только те переменные и их отрицания, которые сохраняют свое значение на этом интервале. Переменные, которые меняют свое значение на интервале, упростятся. Чтобы получить МДНФ остается только объединить при помощи дизъюнкции имеющееся множество минимальных конъюнкций.

Например, рассмотрим вертикальный интервал размера 4, расположенный вдоль левой границы карты (рис.18). Переменные «с» и «d», равные нулю, сохраняют свое значение на протяжении всего интервала, а переменные «a» и «b» меняют свое значение. Следовательно, логическая функция на рассматриваемом интервале никак не зависит от переменных «a» и «b», поэтому эти переменные упрощаются, а соответствующая интервалу формула будет записана как конъюнкция $\bar{c} \cdot \bar{d}$. Переменные, имеющие нулевое значение, входят в эту конъюнкцию с отрицаниями. Так происходит потому, что при построении формулы интервала мы отвечаем на вопрос: как единичное значение функции на интервале можно описать при помощи конъюнкции ее переменных, принимающих конкретные значения из рассматриваемого набора.

Рассуждая аналогично, получаем формулу для всей МДНФ (формула 7).

$$F_{\text{мднф}} = \bar{b} \cdot \bar{d} + a \cdot \bar{b} + \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} \quad (7)$$

Теперь приведем полученную МДНФ к базисам «И-НЕ» и «ИЛИ-НЕ».

Для этого воспользуемся законами де Моргана, в результате имеем формулы 8,9.

$$F_{\text{мднф}_{\text{и-не}}} = \overline{\overline{\overline{b \cdot d \cdot a \cdot b \cdot c \cdot d \cdot a \cdot b \cdot c}}} \quad (8)$$

$$F_{\text{мднф}_{\text{или-не}}} = \overline{\overline{\overline{b + d + a + b + c + d + a + b + c}}} \quad (9)$$

По заданию также требуется построить МКНФ рассматриваемой функции и тоже выразить ее в разных базисах.

МКНФ строится по нулевым значениям логической функции.

Обратимся еще раз к рис. 17 и изменим его: на пустых клетках поставим нулевые значения, а единичные значения удалим для повышения наглядности рисунка. Получится карта, показанная на рис.19.

$\begin{smallmatrix} cd \\ ab \end{smallmatrix}$	00	01	11	10
00		0	0	
01			0	0
11		0	0	0
10				

Рис.19 Карта Карно, заполненная для построения МКНФ

Выделим интервалы, на которых функция сохраняет свое нулевое значение (рис. 20). Выделение происходит по правилам, названным ранее.

$\begin{smallmatrix} cd \\ ab \end{smallmatrix}$	00	01	11	10
00		0	0	
01			0	0
11		0	0	0
10				

Рис. 20 Результат выделения интервалов для МКНФ

Запишем формулу МКНФ, для чего последовательно рассмотрим каждый из интервалов. Для каждого интервала запишем минимальную дизъюнкцию, куда будут входить только те переменные и их отрицания, которые сохраняют свое значение на этом интервале. Переменные, которые меняют свое значение на интервале, упростятся. Например, рассмотрим квадратный интервал у правой границы карты (рис. 20). На нем переменные «b» и «c» сохраняют свое значение, равное единице, а остальные переменные меняют свое значение, и, следовательно, упрощаются. Чтобы записать формулу интервала, отвечаем на вопрос: как нулевое значение функции на этом интервале можно описать при помощи дизъюнкции переменных, принимающих свое значение на соответствующем наборе. Очевидно, что переменные, равные единице, нужно взять с отрицанием, а переменные, равные нулю без отрицания. В результате формула интервала будет записана как дизъюнкция $\bar{b} + \bar{c}$.

Чтобы получить МКНФ, необходимо объединить при помощи конъюнкции множество минимальных дизъюнкций, построенных для всех имеющихся интервалов (формула 10).

$$F_{\text{МКНФ}} = (\bar{b} + \bar{c}) \cdot (\bar{a} + \bar{b} + \bar{d}) \cdot (a + b + \bar{d}) \quad (10)$$

Теперь приведем полученную МКНФ к базисам «И-НЕ» и «ИЛИ-НЕ».

Для этого воспользуемся законами де Моргана, в результате получим формулы 11, 12.

$$F_{\text{МКНФ}_{\text{или-не}}} = \overline{\overline{\bar{b} + \bar{c}} + \overline{\bar{a} + \bar{b} + \bar{d}} + \overline{a + b + \bar{d}}} \quad (11)$$

$$F_{\text{МКНФ}_{\text{и-не}}} = \overline{\overline{\overline{b \cdot c \cdot a \cdot b \cdot d \cdot \bar{a} \cdot \bar{b} \cdot d}}} \quad (12)$$

Построим в лабораторном комплексе комбинационные схемы, реализующие рассматриваемую функцию в базисах «И-НЕ» и «ИЛИ-НЕ» (всего 4 схемы), протестируем их работу и убедимся в их правильности (рис. 21–24).

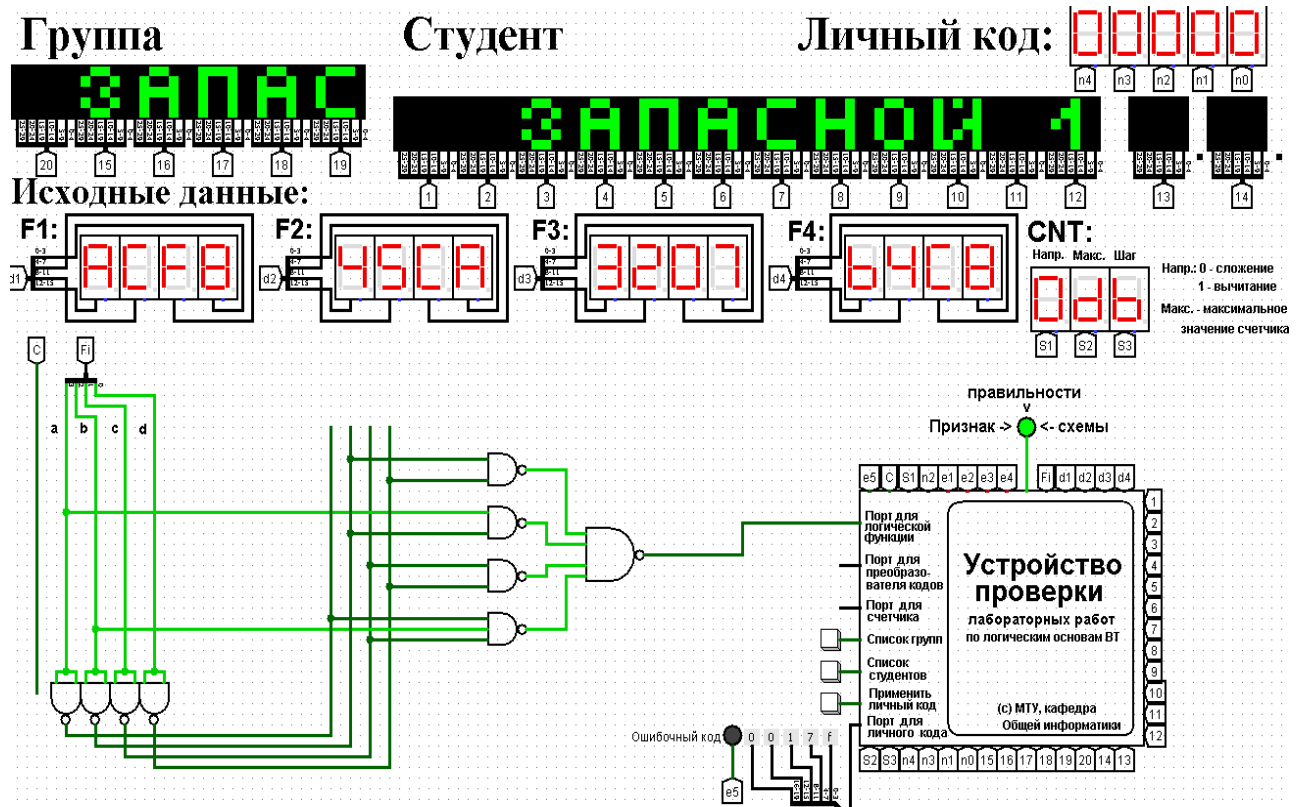


Рис.21 Тестирование схемы МДНФ, построенной в базе «И-НЕ»

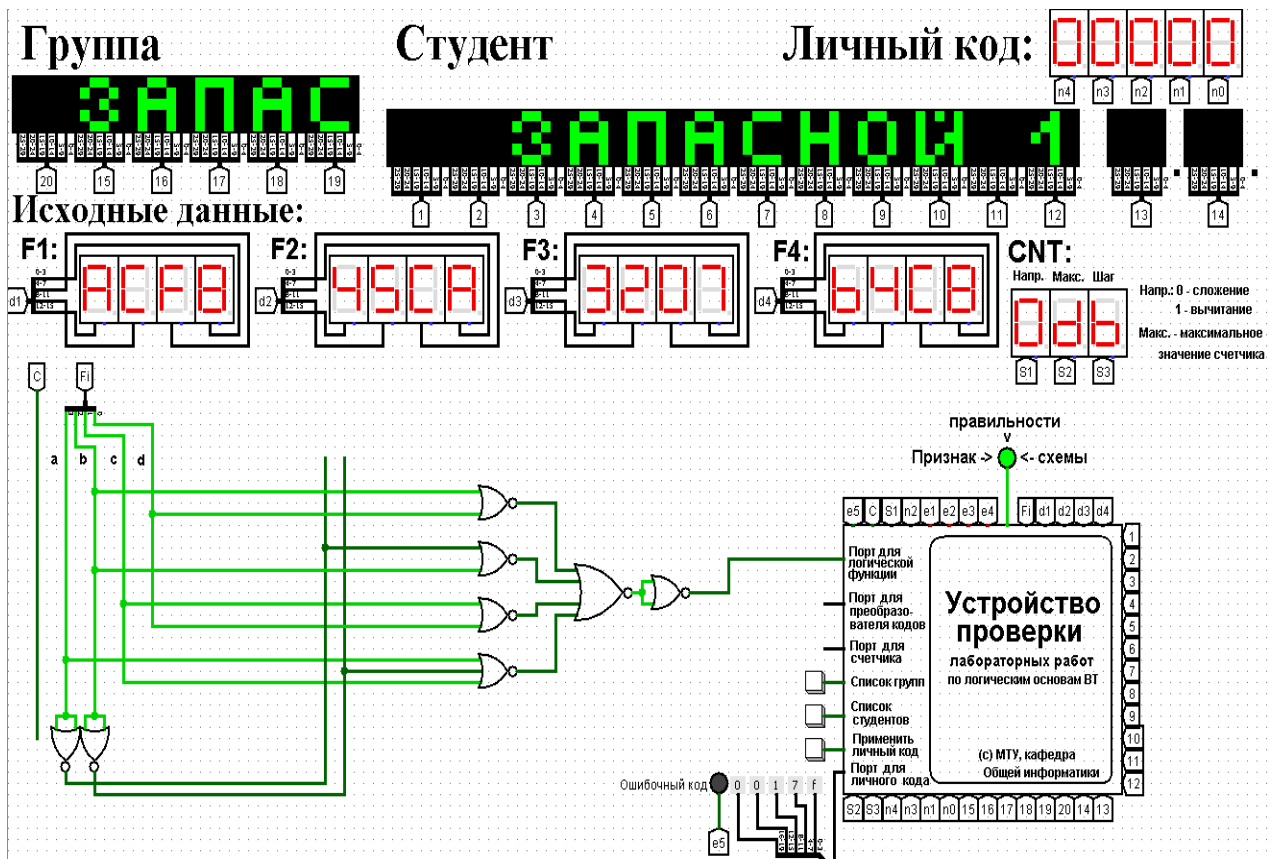


Рис.22 Тестирование схемы МДНФ, построенной в базе «ИЛИ-НЕ»

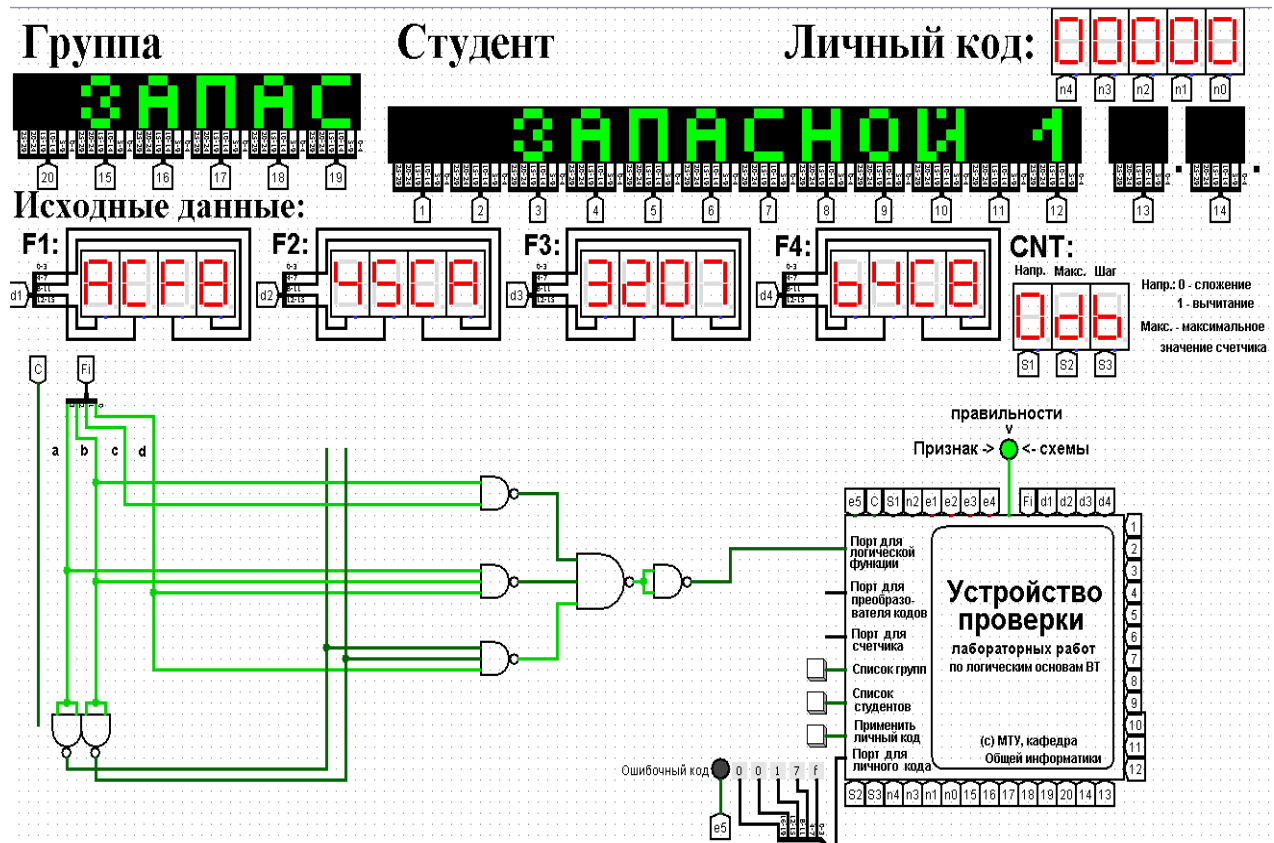
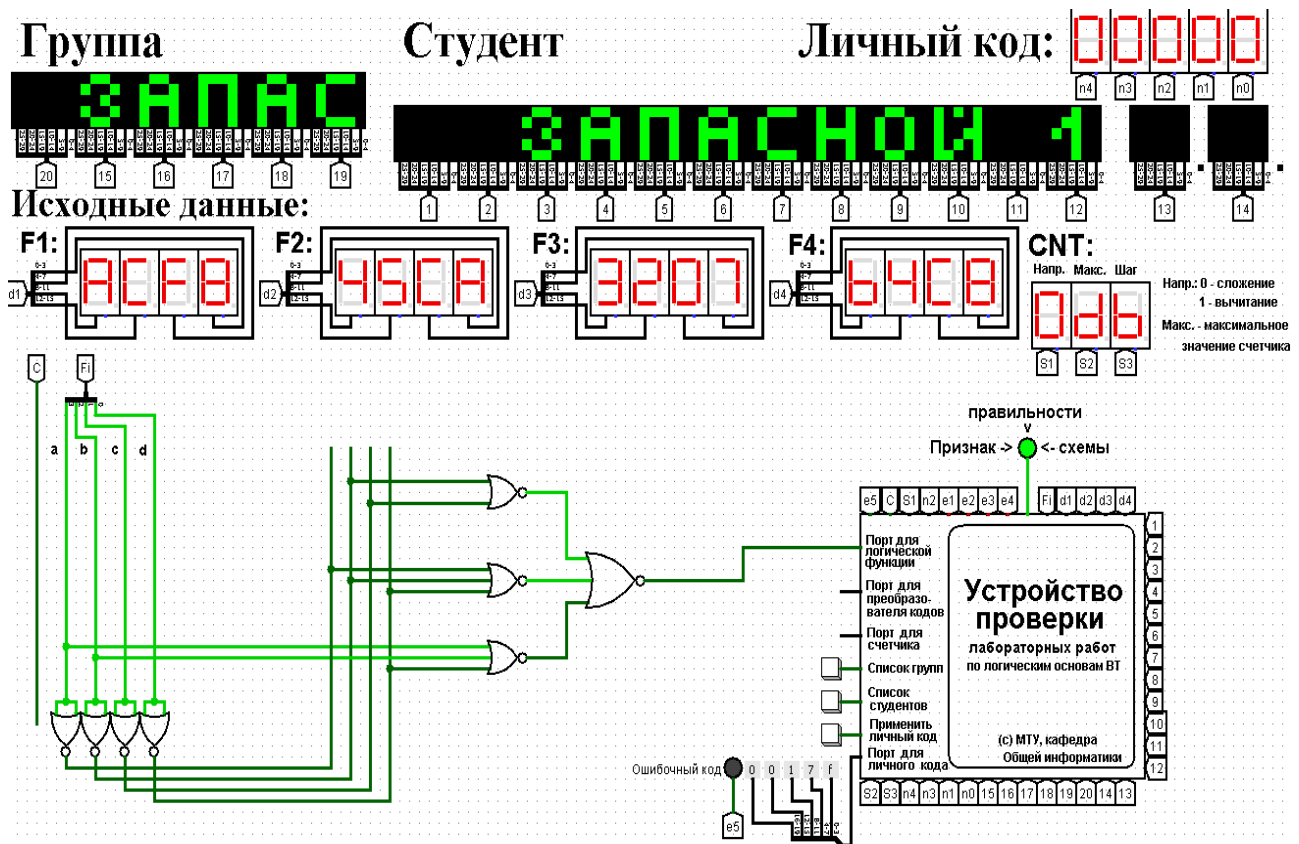


Рис.23 Тестирование схемы МКНФ, построенной в базисе «И-НЕ»

Рис. 24 Тестирование схемы МКНФ, построенной в базисе «ИЛИ-НЕ»
Тестирование показало, что все схемы работают правильно.

5. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое МДНФ (МКНФ)?
- Что такое логический базис?
- Что такое карта Карно, как ее построить и использовать для минимизации?
- Как изменится схема, если в таблице истинности сделать такие-то изменения?
- Как изменится таблица истинности, если в схеме сделать такие-то изменения?
- Чему равна некоторая логическая функция, если ее аргументами будут МДНФ и МКНФ?
- Дана карта Карно, заполненная значениями некоторой функции, как выделить интервалы и записать МДНФ (МКНФ)?
- Может ли формула МДНФ быть сложнее, чем формула МКНФ (и наоборот) и почему?
- Другие вопросы и задачи по теме работы.

7. Практическая работа № 7: реализация заданной логической функции от четырех переменных на дешифраторах 4-16, 3-8 и 2-4

1. Постановка задачи

Логическая функция от четырех переменных задана в 16-теричной векторной форме. Восстановить таблицу истинности. По таблице истинности реализовать в лабораторном комплексе логическую функцию на дешифраторах тремя способами:

- используя дешифратор 4-16 и одну дополнительную схему «или»;
- используя два дешифратора 3-8 и необходимую дополнительную логику;
- используя пять дешифраторов 2-4 и одну дополнительную схему «или».

Протестировать работу схем и убедиться в правильности их работы. Подготовить отчет о проделанной работе и защитить ее.

2. Последовательность выполнения работы

1. Перед выполнением работы ознакомиться с приложениями 1-3.
2. Запустить лабораторный комплекс и получить персональные исходные данные для этой работы. Для этого требуется установить правильное название группы и ФИО студента. Рекомендуется зафиксировать личный код. Логическая функция от четырех переменных, заданная в 16-теричной форме, будет показана на индикаторе F1.
3. Последовательно собрать в лабораторном комплексе комбинационные схемы, реализующие заданную логическую функцию на дешифраторах требуемыми способами. Подключить входы и выходы схем к устройству проверки.
4. Протестировать работу схем и убедиться в их правильности. В случае обнаружения ошибки (загорятся индикаторы ошибки), найти ее и исправить.
5. Продемонстрировать правильность работы схем преподавателю, для чего необходимо при нем запустить процесс моделирования работы, и получить разрешение преподавателя на оформление отчета.
6. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).
7. Защитить практическую работу, отвечая на дополнительные вопросы, и получить роспись преподавателя в тетрадь учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.
3. Постановка задачи и персональный вариант.
4. Восстановленная таблица истинности.
5. Схемы, реализующие логическую функцию на дешифраторах требуемыми способами (должны быть приведены фотографии экрана, на которых видны: группа, ФИО студента, индикаторы исходных данных, разработанные схемы с подключением к устройству проверки, а также положительный результат проверки).
6. Выводы.
7. Список информационных источников.

4. Пример

Предположим, что в соответствии с вариантом функция, заданная в 16-теричной форме имеет следующий вид: $F(a,b,c,d) = ACF8_{16}$.

После восстановления таблицы истинности будет получена табл. 1 (см. практическую работу №4).

Реализуем функцию, используя дешифратор 4-16 и одну дополнительную схему «или». Количество выходов дешифратора соответствует количеству значений логической функции, поэтому требуется только один такой дешифратор. Подадим значения переменных функции на адресные входы дешифратора: младшую переменную «d» - на младший адресный вход, старшую переменную «a» - на старший адресный вход, прочие переменные –аналогично (на схеме далее переменные подаются на адресные входы дешифратора при помощи шины). В процессе работы на выходах дешифратора (с нулевого по пятнадцатый) будут последовательно возникать единичные значения в соответствии с поступающей на адресные входы комбинацией значений переменных. Выберем лишь те выходы дешифратора, номера которых совпадают с номерами наборов значений переменных, на которых функция равна единице. Объединим эти выходы дешифратора через «или» и получим требуемую реализацию (рис.25).

Сразу после добавления дешифратора на рабочую область необходимо настроить ему некоторые свойства:

- «выбирающие биты» (адресные входы) установить равными 4;
- «три состояния» — нет;
- «на отключенном выходе» — установить равным 0;
- «разрешающий вход» — нет.

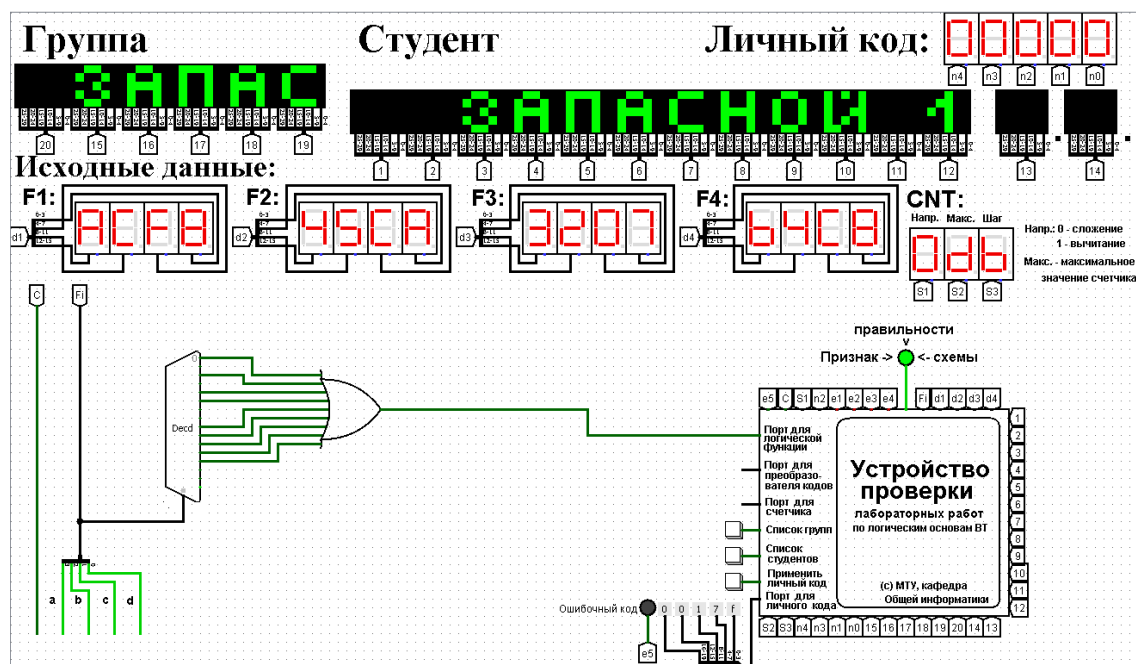


Рис.25 Тестирование схемы, реализующей логическую функцию на дешифраторе 4-16

Тестирование показало, что схема работает правильно.

Реализуем функцию, используя дешифраторы 3-8 и необходимую дополнительную логику. Количество выходов у дешифратора 3-8 в два раза меньше количества значений логической функции, поэтому нам потребуется разместить на рабочей области лабораторного комплекса два дешифратора 3-8. Также следует обратить внимание, что количество адресных входов дешифратора меньше, чем количество переменных функции.

Поэтому подадим значения трех младших переменных функции на адресные входы обоих дешифраторов: младшую переменную «d» — на младший адресный вход, старшую переменную «b» — на старший адресный вход, переменную «с» — аналогично (на схеме далее переменные подаются на адресные входы дешифраторов при помощи разветвителя и шины).

Переменная «a» используется для управления дешифраторами. Когда «a» равна нулю, то должен работать первый дешифратор - он отвечает за первую половину таблицы истинности. Когда «a» равна единице, то должен работать второй дешифратор — он отвечает за вторую половину таблицы истинности. Чтобы это реализовать, переменная «a» должна подаваться на разрешающий вход первого дешифратора через инверсию, а на вход второго — без инверсии.

Для большей наглядности проиллюстрируем сказанное выше рисунком 26.

	a	b	c	d	F	
Когда "a" равна нулю, работает первый дешифратор	0	0	0	0	1	Область ответственности первого дешифратора
	0	0	0	1	0	
	0	0	1	0	1	
	0	0	1	1	0	
	0	1	0	0	1	
	0	1	0	1	1	
	0	1	1	0	0	
	0	1	1	1	0	
Когда "a" равна единице, работает второй дешифратор	1	0	0	0	1	Область ответственности второго дешифратора
	1	0	0	1	1	
	1	0	1	0	1	
	1	0	1	1	1	
	1	1	0	0	1	
	1	1	0	1	0	
	1	1	1	0	0	
	1	1	1	1	0	

Рис. 26 Распределение областей таблицы истинности между дешифраторами 3-8

Для того чтобы у дешифраторов появился разрешающий вход, нам потребуется в их свойствах активировать соответствующую опцию.

Прочие настройки дешифраторов должны быть аналогичны предыдущей реализации.

В процессе работы на выходах всех дешифраторов будут последовательно возникать единичные значения в соответствии с поступающей на адресные входы комбинацией значений переменных. У первого дешифратора выберем лишь те выходы, чьи номера совпадают с номерами наборов значений переменных, на которых функция равна единице, из первой половины таблицы. У второго дешифратора выберем лишь те выходы, чьи номера совпадают с номерами наборов значений переменных за вычетом 8, на которых функция равна единице, из второй половины таблицы.

Объединим выбранные выходы обоих дешифраторов через «или» и получим требуемую реализацию (рис.27).

Тестирование подтвердило правильность работы схемы.

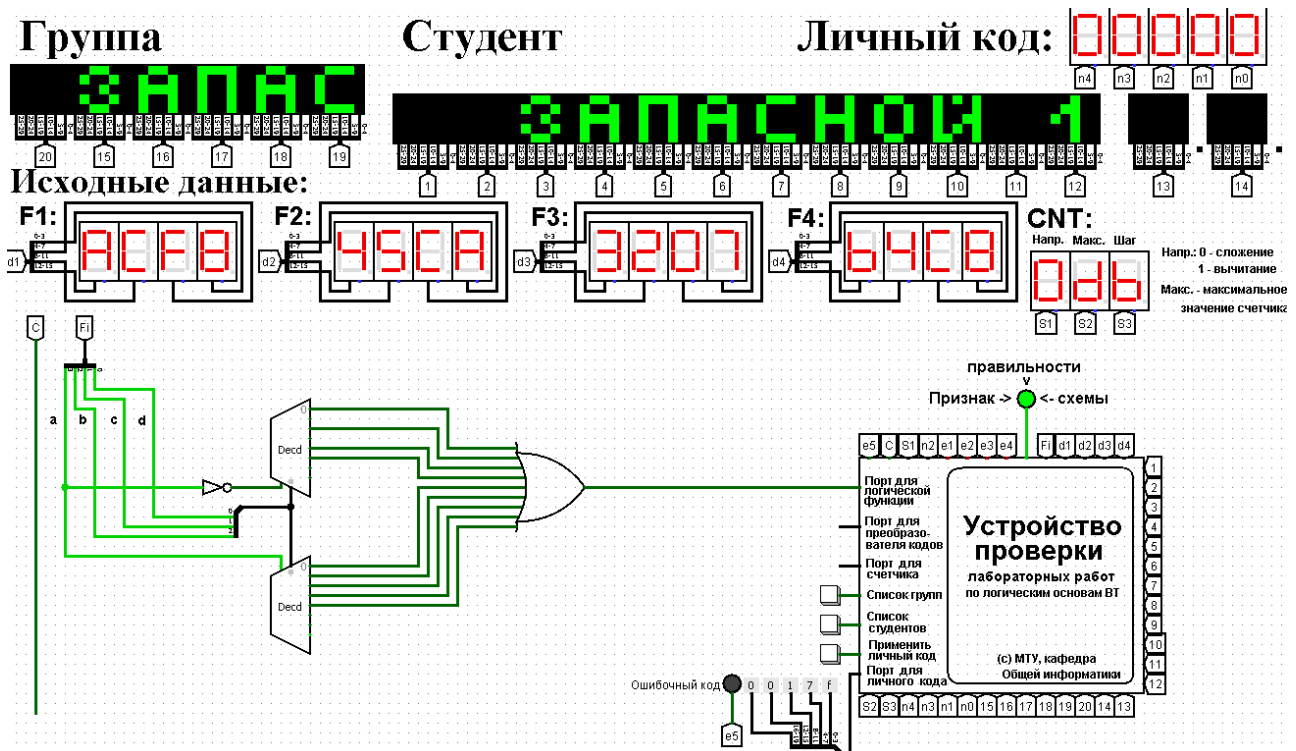


Рис. 27 Тестирование схемы, реализующей логическую функцию на дешифраторах 3-8 и дополнительной логике

Реализуем функцию, используя дешифраторы 2-4 и необходимую дополнительную логику. Количество выходов у дешифратора 2-4 в четыре раза меньше количества значений логической функции, поэтому нам потребуется разместить на рабочей области лабораторного комплекса четыре дешифратора 2-4, которые мы будем называть операционными, а также еще один дешифратор 2-4, который будет управлять первыми четырьмя – назовем его управляющим.

Итого всего потребуется пять дешифраторов 2-4 и дополнительная схема «или».

Следует обратить внимание, что количество адресных входов у каждого дешифратора в два раза меньше, чем количество переменных функции, поэтому каждый операционный дешифратор будет отвечать лишь за одну четверть исходной таблицы истинности. Для большей наглядности проиллюстрируем сказанное выше рисунком 28.

	a	b	c	d	F	
Первый операционный дешифратор включается, когда на адресных входах управляющего дешифратора комбинация 00	0	0	0	0	1	Область ответственности первого операционного дешифратора
	0	0	0	1	0	
	0	0	1	0	1	
	0	0	1	1	0	
Второй включается, когда на адресных входах управляющего 01	0	1	0	0	1	Область ответственности второго операционного дешифратора
	0	1	0	1	1	
	0	1	1	0	0	
	0	1	1	1	0	
Третий включается, когда на адресных входах управляющего 10	1	0	0	0	1	Область ответственности третьего операционного дешифратора
	1	0	0	1	1	
	1	0	1	0	1	
	1	0	1	1	1	
Четвертый включается, когда на адресных входах управляющего 11	1	1	0	0	1	Область ответственности четвертого операционного дешифратора
	1	1	0	1	0	
	1	1	1	0	0	
	1	1	1	1	0	

Рис. 28 Распределение областей таблицы истинности между дешифраторами 2-4

Значения двух младших переменных функции используются для адресации четырех операционных дешифраторов: младшая переменная «d» - подается на младший адресный вход, старшая переменная «c» - на старший адресный вход (на схеме далее переменные подаются на адресные входы дешифраторов при помощи разветвителя и шины).

Переменные «a» и «b» используются для управления операционными дешифраторами и аналогичным образом подаются на адресные входы управляющего дешифратора. Выходы управляющего дешифратора должны быть подключены к разрешающим входам операционных дешифраторов. Таким образом, когда «a» и «b» равны нулю, то на нулевом выходе управляющего дешифратора образуется единица, которая подается на разрешающий вход первого операционного дешифратора. И так далее, аналогично.

Теперь фактически каждый операционный дешифратор отвечает за свою двоичную тетраду в исходной векторной записи логической функции. Выберем у каждого операционного дешифратора лишь те выходы, где у двоичной тетрады стоят единицы. При этом необходимо считать, что нулевой выход соответствует старшему двоичному разряду тетрады.

Объединим выбранные выходы всех операционных дешифраторов через «или» и получим требуемую реализацию (рис.29).

Тестирование подтвердило правильность работы схемы.

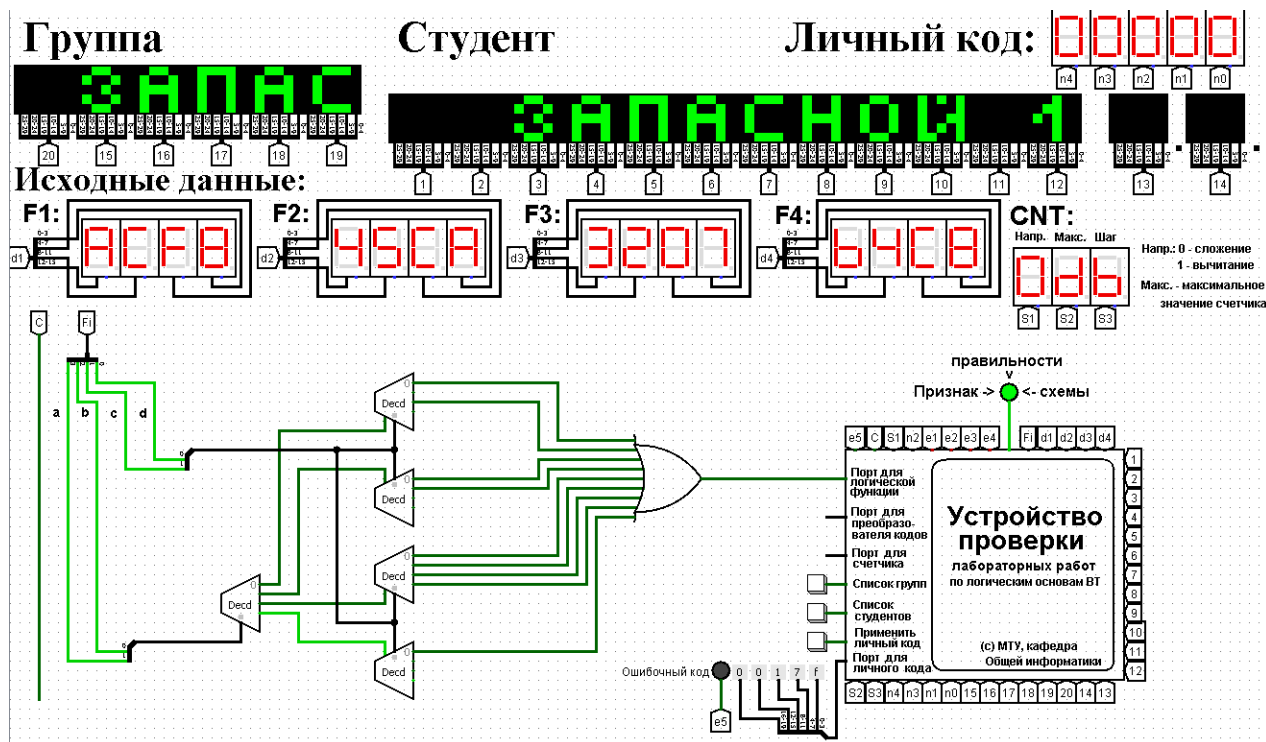


Рис. 29 Тестирование схемы, реализующей логическую функцию на дешифраторах 2-4 и дополнительной логике

5. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое дешифратор?
- Как дешифратор устроен внутри?
- Задана комбинация сигналов на входе дешифратора, что будет на его выходах?
- Задана комбинация сигналов на выходах дешифратора, какие сигналы нужно подать на его входы, чтобы возникла указанная комбинация на выходах?
- Как изменится логическая функция, если в текущую схему внести некоторые изменения?
- Какова область применения дешифраторов?
- Другие вопросы и задачи по теме работы.

8. Практическая работа № 8: реализация заданной логической функции от четырех переменных на мультиплексорах 16-1, 8-1, 4-1, 2-1

1. Постановка задачи

Логическая функция от четырех переменных задана в 16-теричной векторной форме. Восстановить таблицу истинности. По таблице истинности реализовать в лабораторном комплексе логическую функцию на мультиплексорах следующими способами:

- используя один мультиплексор 16-1;
- используя один мультиплексор 8-1;
- используя минимальное количество мультиплексоров 4-1;
- используя минимальную комбинацию мультиплексоров 4-1 и 2-1.

Протестировать работу схем и убедиться в их правильности. Подготовить отчет о проделанной работе и защитить ее.

2. Последовательность выполнения работы

1. Перед выполнением работы ознакомиться с приложениями 1-3.
2. Запустить лабораторный комплекс и получить персональные исходные данные для этой работы. Для этого требуется установить правильное название группы и ФИО студента. Рекомендуется зафиксировать личный код. Логическая функция от четырех переменных, заданная в 16-теричной форме, будет показана на индикаторе F1.
3. Последовательно собрать в лабораторном комплексе комбинационные схемы, реализующие заданную логическую функцию на мультиплексорах требуемыми способами. Подключить входы и выходы схем к устройству проверки.
4. Протестировать работу схем и убедиться в их правильности. В случае обнаружения ошибки (загорятся индикаторы ошибки), найти ее и исправить.
5. Продемонстрировать правильность работы схем преподавателю, для чего необходимо при нем запустить процесс тестирования, и получить разрешение преподавателя на оформление отчета.
6. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).
7. Защитить практическую работу, отвечая на дополнительные вопросы, и получить роспись преподавателя в тетрадь учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.
3. Постановка задачи и персональный вариант.
4. Восстановленная таблица истинности.
5. Схемы, реализующие логическую функцию на мультиплексорах требуемыми способами (должны быть приведены фотографии экрана, на которых видны: группа, ФИО студента, индикаторы исходных данных, разработанные схемы с подключением к устройству проверки, а также положительный результат проверки).
6. Выводы.
7. Список информационных источников.

4. Пример

Предположим, что в соответствии с вариантом функция, заданная в 16-теричной форме имеет следующий вид: $F(a,b,c,d) = ACF8_{16}$.

После восстановления таблицы истинности будет получена табл. 1 (см. практическую работу №4).

Для реализации заданной функции на мультиплексоре 16-1 выполним следующее.

Разместим мультиплексор на рабочей области лабораторного комплекса и сделаем ему следующие настройки:

- свойство «выбирающие биты» сделаем равным 4;
- «разрешающий вход» — нет;
- «положение выбирающего входа» — сверху (сделано в данном примере для удобства, можно оставить значение по умолчанию).

Количество информационных входов мультиплексора соответствует количеству значений логической функции. Поэтому просто подадим значения функции на соответствующие входы. Для этого удобно воспользоваться логическими константами из раздела «Провода» библиотеки элементов Logisim.

На адресные (выбирающие) входы мультиплексора подадим при помощи шины значения логических переменных. Несмотря на использование шины, следует помнить, что младшая переменная подается на младший адресный вход, а старшая – на старший.

Собранная и протестированная схема показана на рис. 30. Тестирование подтвердило правильность работы схемы.

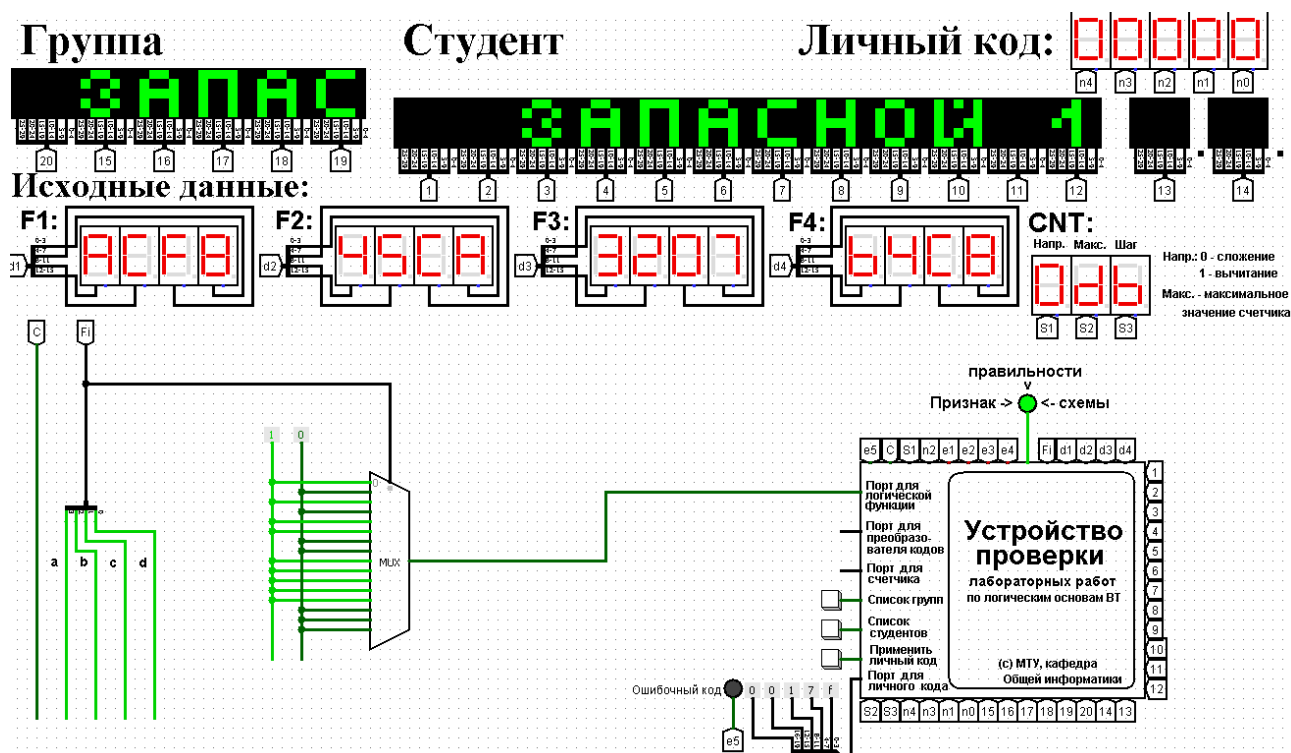


Рис.30 Тестирование схемы, реализующей логическую функцию на мультиплексоре 16-1

Выполним реализацию заданной логической функции при помощи мультиплексора 8-1.

Мультиплексор 8-1 имеет 3 адресных входа, что не позволяет подать на эти входы все 4 логические переменные, как это было сделано в предыдущем случае.

Однако мы можем в качестве адресных переменных выбрать любые три из имеющихся, а оставшуюся четвертую рассматривать наравне с логическими константами как элемент исходных данных для информационных входов.

Удобнее всего в качестве адресных переменных взять три старшие переменные нашей функции, т.е. a, b, c. Тогда пары наборов, на которых эти переменные будут иметь одинаковое значение, будут располагаться в соседних строчках таблицы истинности и поэтому можно будет легко увидеть, как значение логической функции для каждой пары наборов соотносится со значением переменной d (рис. 31).

Например, из рис. 31 видно, что для первых двух строчек $F = \bar{d}$. Всего же для разных пар наборов может быть четыре случая: $F = 1$, $F = 0$, $F = d$, $F = \bar{d}$.

Таким образом, мы перенесли одну переменную в область значений функции и получили таблицу, похожую на таблицу истинности функции от трех

переменных. Таблица 3 отображает «сжатую» таблицу истинности.

a	b	c	d	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Рис.31 Взаимосвязь значений функции и значений переменной «d»

Таблица 3

a	b	c	F
0	0	0	\bar{d}
0	0	1	\bar{d}
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	\bar{d}
1	1	1	0

Теперь, рассматривая переменную d наравне с константами 0 и 1 в качестве сигналов для информационных входов мультиплексора 8-1, можно по аналогии с предыдущим случаем выполнить реализацию требуемой функции.

Разместим на рабочей области новый мультиплексор, установим ему количество выбирающих (адресных) входов равным трем, и выполним необходимые соединения (рис. 32).

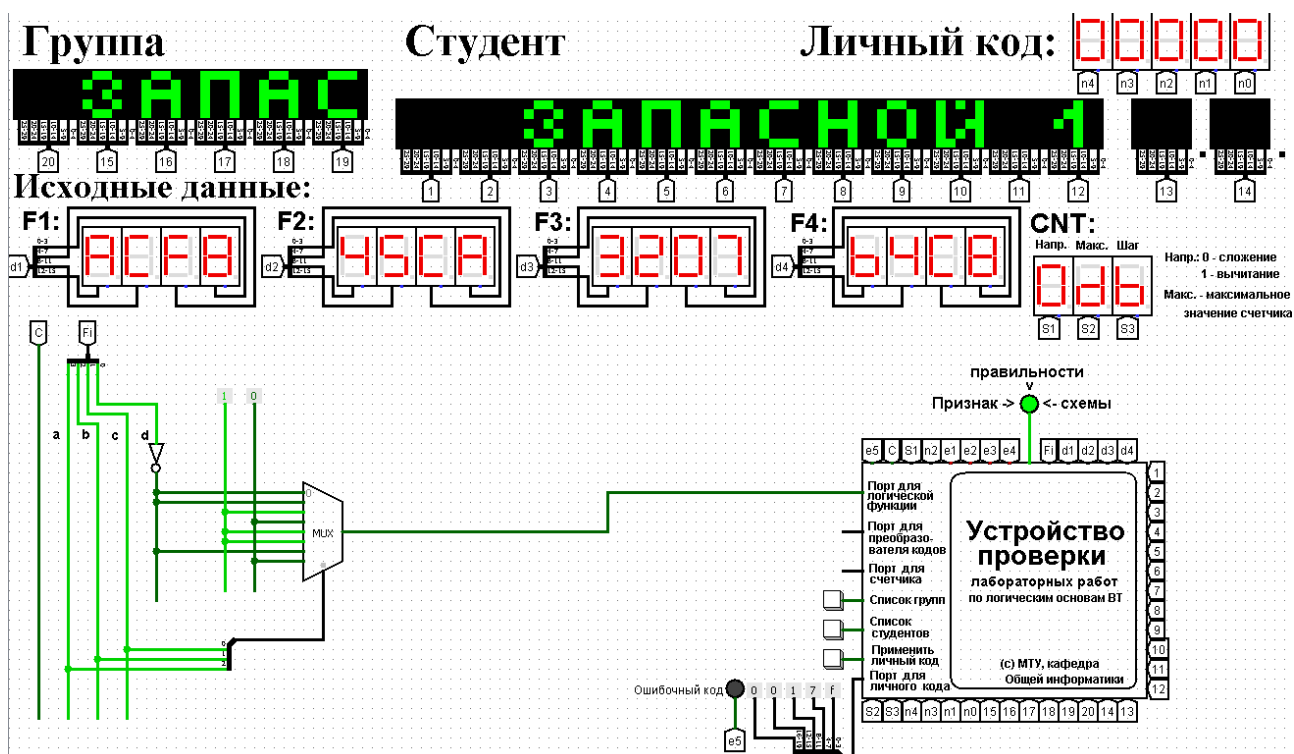


Рис. 32 Тестирование схемы, реализующей логическую функцию на мультиплексоре 8-1

Тестирование подтвердило правильность работы схемы.

Рассмотрим реализацию заданной функции на минимальном количестве мультиплексоров 4-1.

Мультиплексор 4-1 имеет 2 адресных входа и 4 информационных. Это означает, что мы должны разбить исходную таблицу истинности на 4 фрагмента, за реализацию каждого из которых в принципе должен отвечать отдельный мультиплексор (назовем его операционным). Однако, необходимо учесть требования минимальности по отношению к количеству используемых мультиплексоров и ставить их только там, где без них нельзя обойтись. Также нам нельзя в рамках данной работы использовать другие логические схемы, за исключением отрицания.

По аналогии с реализацией на дешифраторах 2-4 (см. предыдущую работу), нам обязательно потребуется управляющий мультиплексор, который будет выбирать один из вариантов, предлагаемых операционными мультиплексорами (либо один из очевидных вариантов, если без операционных мультиплексоров можно обойтись).

Разобьем исходную таблицу истинности на зоны ответственности между операционными мультиплексорами, а заодно посмотрим, нельзя ли в некоторых случаях обойтись вообще без операционного мультиплексора (рис. 33).

	a	b	c	d	F	
Первый операционный мультиплексор работает, когда "ab" равны 00	0	0	0	0	1	Но без первого операционного мультиплексора можно обойтись, поскольку на данном фрагменте $F = \overline{d}$
	0	0	0	1	0	
	0	0	1	0	1	
	0	0	1	1	0	
Второй, когда "ab" равны 01	0	1	0	0	1	Без второго тоже можно обойтись, поскольку на данном фрагменте $F = \overline{c}$
	0	1	0	1	1	
	0	1	1	0	0	
	0	1	1	1	0	
Третий, когда "ab" равны 10	1	0	0	0	1	И без третьего можно обойтись, поскольку на данном фрагменте $F = 1$
	1	0	0	1	1	
	1	0	1	0	1	
	1	0	1	1	1	
Четвертый, когда "ab" равны 11	1	1	0	0	1	А четвертый операционный мультиплексор действительно нужен
	1	1	0	1	0	
	1	1	1	0	0	
	1	1	1	1	0	

Рис. 33 Разбиение исходной таблицы истинности на зоны ответственности для потенциальных операционных мультиплексоров

Как видно из рис. 33, в трех случаях из четырех без операционного мультиплексора можно вполне обойтись, однако последний фрагмент таблицы требует реализации операционного мультиплексора. С учетом только что сказанного, схема логической функции на минимальном количестве мультиплексоров 4-1 будет такой, как показано на рис. 34.

Тестирование подтвердило правильность работы схемы.

Реализуем логическую функцию, используя минимальную комбинацию мультиплексоров 4-1 и 2-1. В качестве отправной точки рассмотрим результаты, полученные в предыдущей реализации. Управляющий мультиплексор нельзя заменить на мультиплексор 2-1, поскольку у него на входах уникальные сигналы, а вот единственный операционный заменить можно, поскольку он имеет дело с константами. Из рис. 33 выпишем отдельно фрагмент таблицы истинности, за который данный мультиплексор отвечает (табл. 4).

Таблица 4

c	d	F
0	0	1
0	1	0
1	0	0
1	1	0

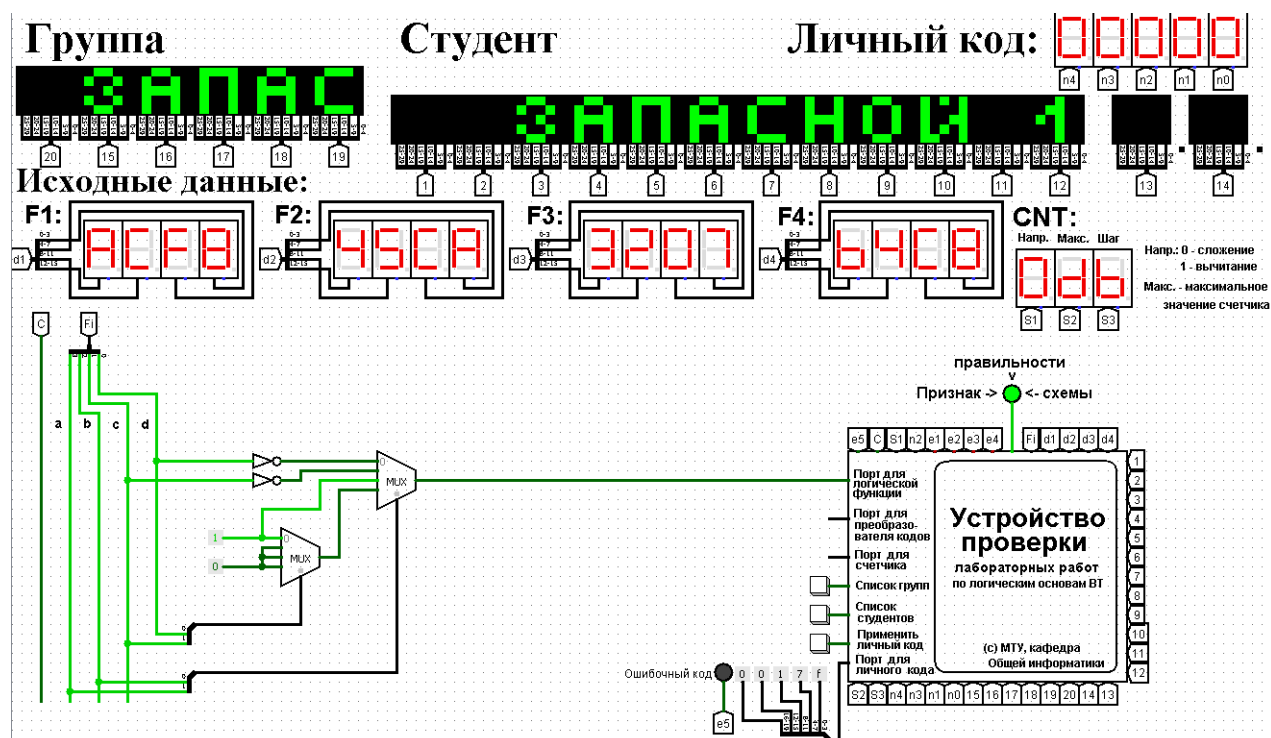


Рис. 34 Тестирование схемы, реализующей логическую функцию на минимальном количестве мультиплексоров 4-1

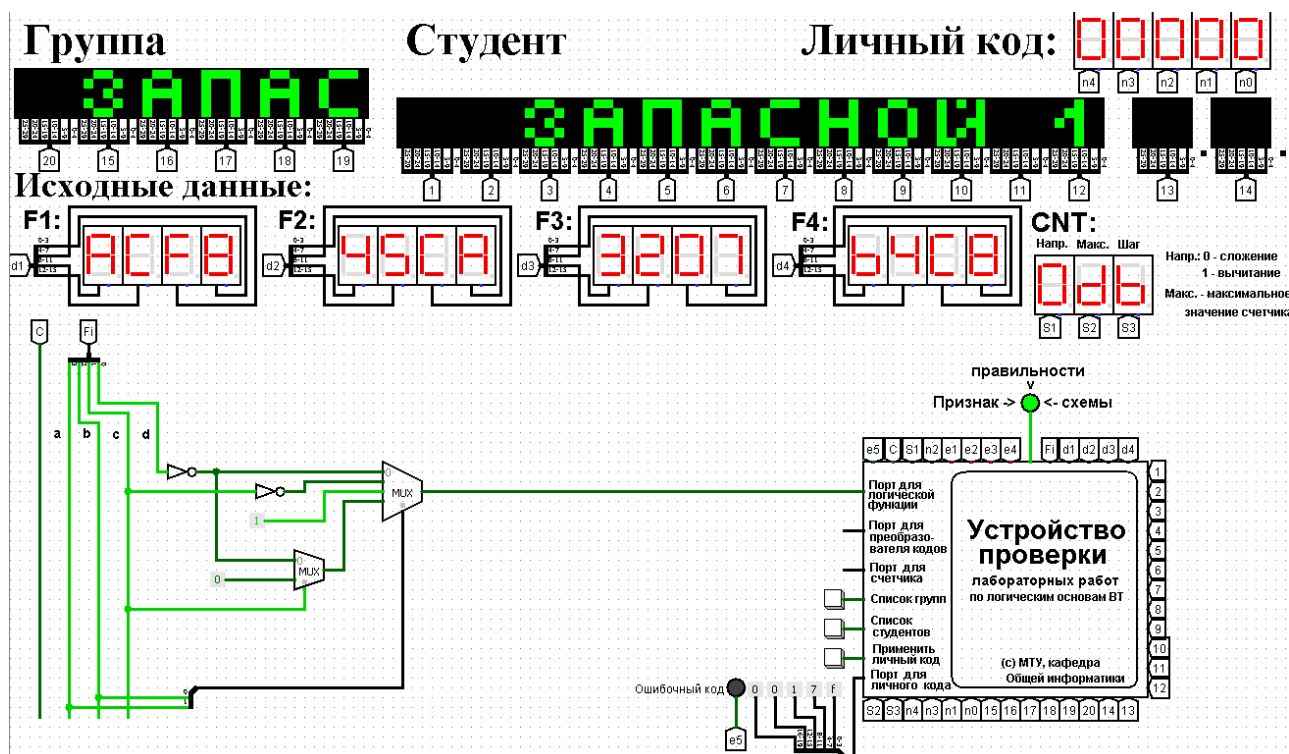


Рис.35 Тестирование схемы, реализующей логическую функцию на основе минимальной комбинации мультиплексоров 4-1 и 2-1

Из таблицы видно, что когда «с» равно 0, то функция равна не «d», а когда «с» равно 1, то функция равна 0. Значит, переменную «с» можно рассматривать как адресную для мультиплексора 2-1, а не «d» и 0 будут поданы на его

информационные входы. В результате получим схему, изображенную на рис. 35. Тестирование подтвердило правильность работы схемы.

5. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое мультиплексор?
- Как мультиплексор устроен внутри?
- Как изменится логическая функция, если в текущую схему внести некоторые изменения?
- Некоторые сигналы поданы на мультиплексор. Требуется восстановить таблицу истинности для функции, которую он реализует.
- Какова область применения мультиплексоров?
- Другие вопросы и задачи по теме работы.

9. Практическая работа №9: преобразователи кодов

1. Постановка задачи

Таблица переходов для преобразователя кодов задана как совокупность четырех логических функций от четырех переменных в 16-теричной векторной форме. Иначе говоря, код, формируемый для некоторого входного набора, образуется как совокупность значений четырех функций для этого набора. Первая задаваемая функция описывает множество старших битов (третий разряд) для всех формируемых кодов, вторая функция описывает второй разряд, третья функция – первый разряд, и четвертая – нулевой. Восстановить таблицу переходов. По таблице переходов реализовать в лабораторном комплексе преобразователь кодов на основе дешифратора, шифратора и дополнительной логики «или».

Протестировать работу схемы и убедиться в ее правильности. Подготовить отчет о проделанной работе и защитить ее.

2. Последовательность выполнения работы

1. Перед выполнением работы ознакомиться с приложениями 1-3.
2. Запустить лабораторный комплекс и получить персональные исходные данные для этой работы. Для этого требуется установить правильное название группы и ФИО студента. Рекомендуется зафиксировать личный код. Первая логическая функция будет отображена на индикаторе F1, вторая – на индикаторе F2, третья – на индикаторе F3, четвертая – на F4.
3. Собрать в лабораторном комплексе требуемый преобразователь кодов. Подключить его входы и выходы к устройству проверки.
4. Протестировать работу схемы и убедиться в ее правильности. В случае обнаружения ошибки (загорятся индикаторы ошибки), найти ее и исправить.
5. Продемонстрировать правильность работы схемы преподавателю, для чего необходимо при нем запустить процесс тестирования, и получить разрешение преподавателя на оформление отчета.
6. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).
7. Защитить практическую работу, отвечая на дополнительные вопросы, и получить роспись преподавателя в тетрадь учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.

3. Постановка задачи и персональный вариант.

4. Восстановленная таблица истинности.

5. Схемы преобразователя кодов (должны быть приведены фотографии экрана, на которых видны: группа, ФИО студента, индикаторы исходных данных, схема преобразователя с подключением к устройству проверки, а также положительный результат проверки).

6. Выводы.

7. Список информационных источников.

4. Пример

Предположим, что в соответствии с вариантом имеются следующие функции:

$$F1 = ACF8_{16}; F2 = 45CA_{16}; F3 = 3207_{16}; F4 = B4C8_{16}$$

Восстановим таблицу истинности (табл. 5).

Таблица 5

a	b	c	d	F1	F2	F3	F4
0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1
0	1	0	0	1	0	0	0
0	1	0	1	1	1	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	1	1	0	1
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	1	0	1
1	1	0	1	0	0	1	0
1	1	1	0	0	1	1	0
1	1	1	1	0	0	1	0

В данном случае в таблице присутствуют повторяющиеся коды, формируемые для разных исходных наборов (выделены одинаковыми цветами).

Схема устройства строится непосредственно по таблице. Значения

переменных «a», «b», «c», «d» указывают на номер выхода дешифратора, который необходимо подключить к некоторому входу шифратора. Номер входа шифратора определяется кодом из правой части таблицы истинности, который должен быть сформирован для данного входного набора значений переменных.

Если для нескольких разных наборов значений переменных должны быть получены одинаковые коды, то соответствующие выходы дешифратора объединяются через «или», а выход «или» уже подается на вход шифратора.

В результате получим схему, показанную на рис.36.

Тестирование доказало правильность работы схемы.

5. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое дешифратор (шифратор)?
- Как дешифратор (шифратор) устроен внутри?
- Чем приоритетный шифратор отличается от простого?
- Что такое преобразователь кодов?
- Какова область применения шифраторов (дешифраторов)?
- Другие вопросы по теме работы.

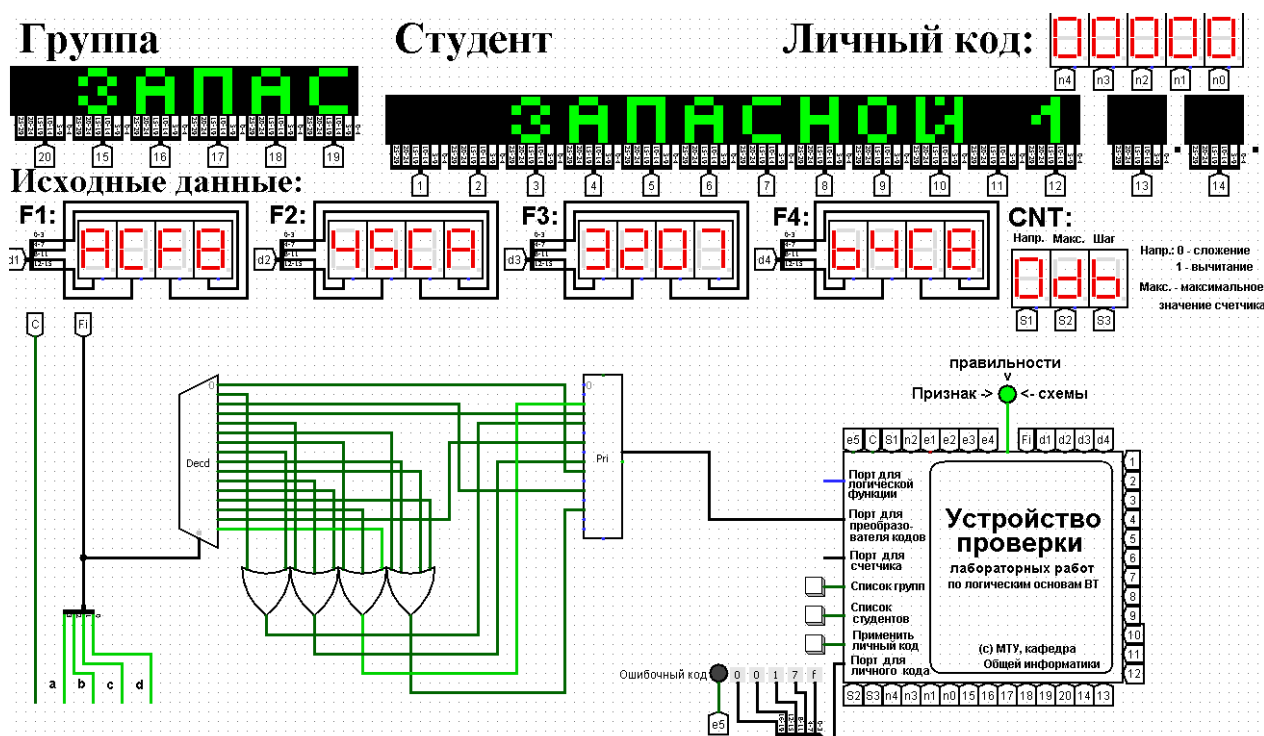


Рис. 36 Тестирование преобразователя кодов

10. Практическая работа №10: изучение работы триггеров

1. Постановка задачи

Изучить на практике работу триггеров, показанных на рисунках ниже (рис. 37-45).

1.1. Одноступенчатый асинхронный RS-триггер на элементах И-НЕ

Таблица переходов триггера (табл. 6) и его функциональная схема.

Таблица 6

\bar{S}	\bar{R}	$Q(t+1)$	$\overline{Q(t+1)}$	Режим
0	0	1	1	Запрещенная комбинация
0	1	1	0	Установка 1
1	0	0	1	Установка 0
1	1	$Q(t)$	$\overline{Q(t)}$	Хранение

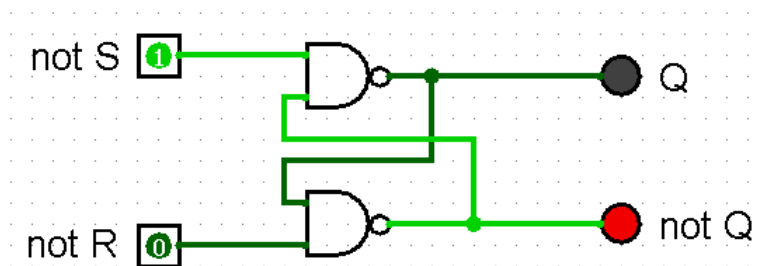


Рис.37 Одноступенчатый асинхронный RS-триггер на элементах И-НЕ

1.2. Одноступенчатый асинхронный RS-триггер на элементах ИЛИ-НЕ

Таблица переходов триггера (табл. 7) и его функциональная схема.

Таблица 7

S	R	$Q(t+1)$	$\overline{Q(t+1)}$	Режим
0	0	$Q(t)$	$\overline{Q(t)}$	Хранение
0	1	0	1	Установка 0
1	0	1	0	Установка 1
1	1	0	0	Запрещенная комбинация

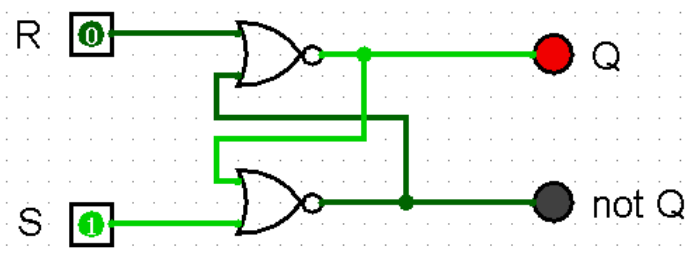


Рис.38 Одноступенчатый асинхронный RS-триггер на элементах ИЛИ-НЕ

1.3. Одноступенчатый синхронный RS-триггер на элементах И-НЕ

Таблица переходов триггера (табл. 8) и его функциональная схема.

Таблица 8

С	S	R	$Q(t+1)$	$\overline{Q(t+1)}$	Режим
0	*	*	$Q(t)$	$\overline{Q(t)}$	Хранение
1	0	0	$Q(t)$	$\overline{Q(t)}$	Хранение
1	0	1	0	1	Установка 0
1	1	0	1	0	Установка 1
1	1	1	1	1	Запрещенная комбинация

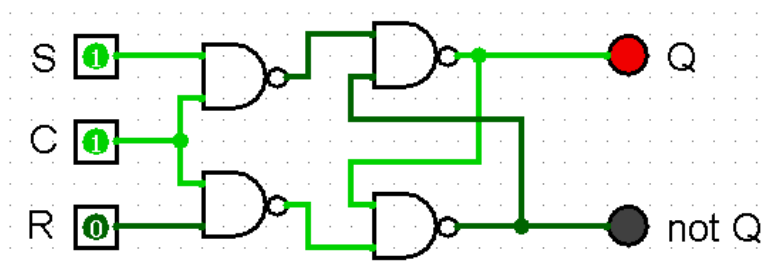


Рис. 39 Одноступенчатый синхронный RS-триггер на элементах И-НЕ

1.4. Двухступенчатый синхронный RS-триггер с асинхронными входами предустановки, выполненный на элементах И-НЕ

Таблица переходов триггера (табл. 9) и его функциональная схема.

Таблица 9

[illegible]

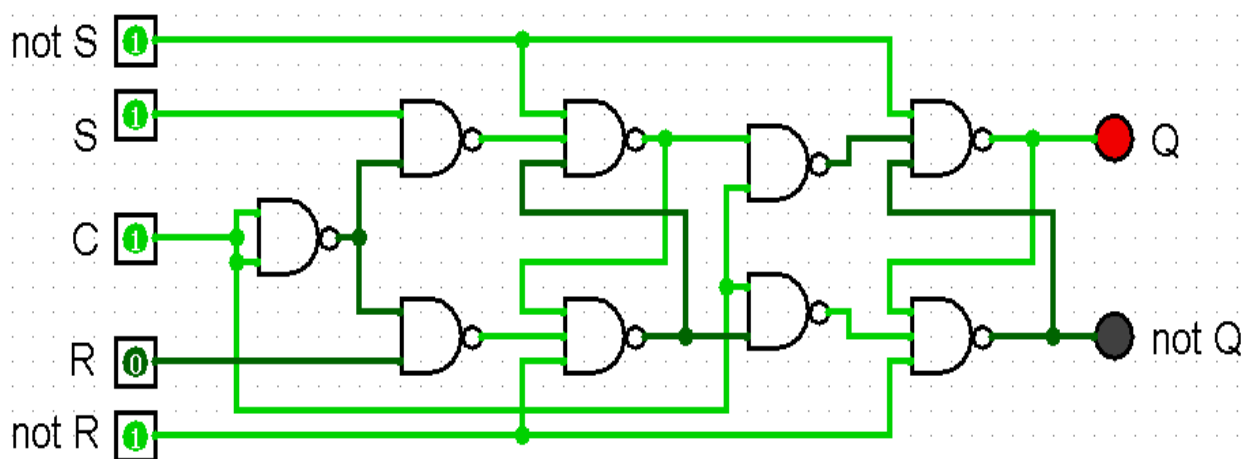


Рис. 40 Двухступенчатый синхронный RS-триггер с асинхронными входами предустановки, выполненный на элементах И-НЕ

1.5. Одноступенчатый D-триггер, выполненный на элементах И-НЕ

Таблица переходов триггера (табл. 10) и его функциональная схема.

Таблица 10

C	D	$Q(t+1)$	$\overline{Q(t+1)}$	Режим
0	*	$Q(t)$	$\overline{Q(t)}$	Хранение
1	0	0	1	Установка 0
1	1	1	0	Установка 1

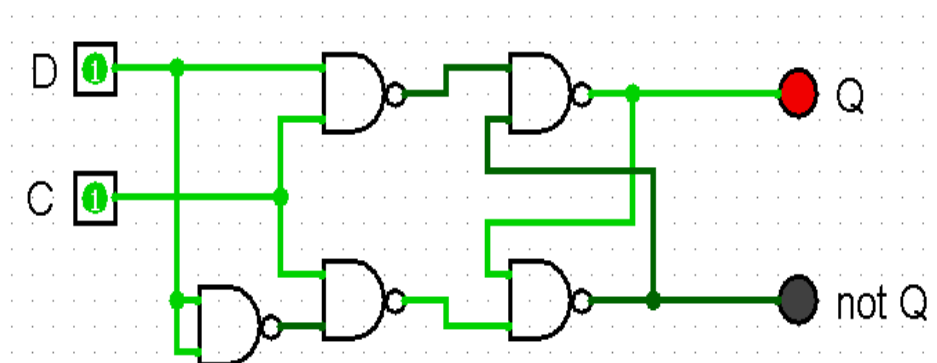


Рис.41 Одноступенчатый D-триггер, выполненный на элементах И-НЕ

1.6. Динамический RS-триггер, работающий по переднему фронту, выполненный на элементах И-НЕ

Таблица переходов триггера (табл. 11) и его функциональная схема.

Таблица 11

C	\bar{S}	\bar{R}	$Q(t+1)$	$\bar{Q}(t+1)$	Режим
0	*	*	$Q(t)$	$\bar{Q}(t)$	Хранение
1	*	*	$Q(t)$	$\bar{Q}(t)$	Хранение
\neg	0	0	0	0	Запрещенная комбинация
\neg	0	1	1	0	Синхронная установка 1
\neg	1	0	0	1	Синхронная установка 0
*	1	1	$Q(t)$	$\bar{Q}(t)$	Хранение

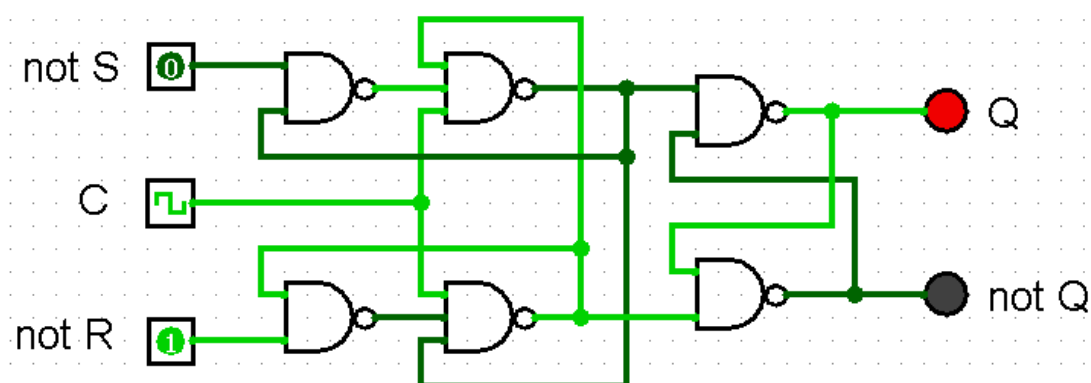


Рис.42 Динамический RS-триггер, работающий по переднему фронту, выполненный на элементах И-НЕ

1.7. Динамический RS-триггер, работающий по заднему фронту, выполненный на элементах ИЛИ-НЕ

Таблица переходов триггера (табл. 12) и его функциональная схема.

Таблица 12

C	\bar{S}	\bar{R}	$Q(t+1)$	$\bar{Q}(t+1)$	Режим
0	*	*	$Q(t)$	$\bar{Q}(t)$	Хранение
1	*	*	$Q(t)$	$\bar{Q}(t)$	Хранение
\neg	1	1	1	1	Запрещенная комбинация
\neg	0	1	1	0	Синхронная установка 1
\neg	1	0	0	1	Синхронная установка 0
*	0	0	$Q(t)$	$\bar{Q}(t)$	Хранение

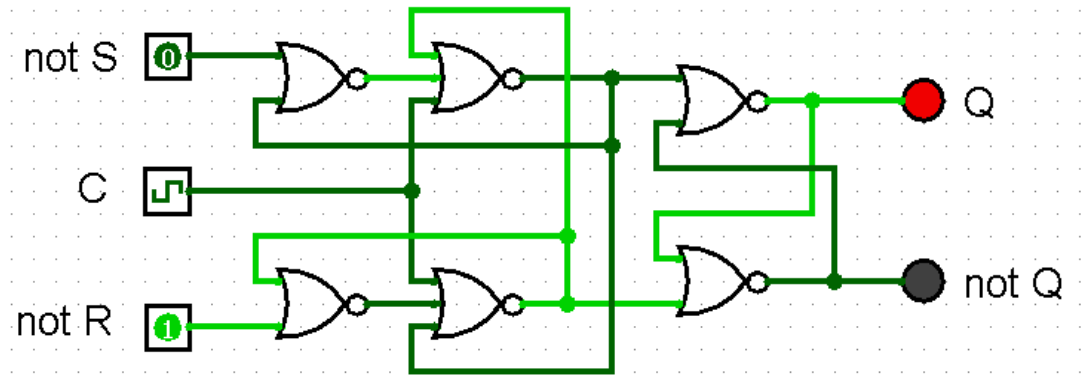


Рис.43 Динамический RS-триггер, работающий по заднему фронту, выполненный на элементах ИЛИ-НЕ

1.8. Т-триггер с асинхронными входами предустановки, выполненный на основе двухступенчатого RS-триггера

Таблица переходов триггера (табл. 13) и его функциональная схема.

Таблица 13

T	\bar{S}	\bar{R}	$Q(t+1)$	$\overline{Q(t+1)}$	Режим
*	0	0	1	1	Запрещенная комбинация
*	0	1	1	0	Асинхронная 1
*	1	0	0	1	Асинхронный 0
0	1	1	$Q(t)$	$\overline{Q(t)}$	Хранение
1	1	1	$Q(t)$	$\overline{Q(t)}$	Хранение
\neg	1	1	$\overline{Q(t)}$	$Q(t)$	Переключение в противоположное состояние

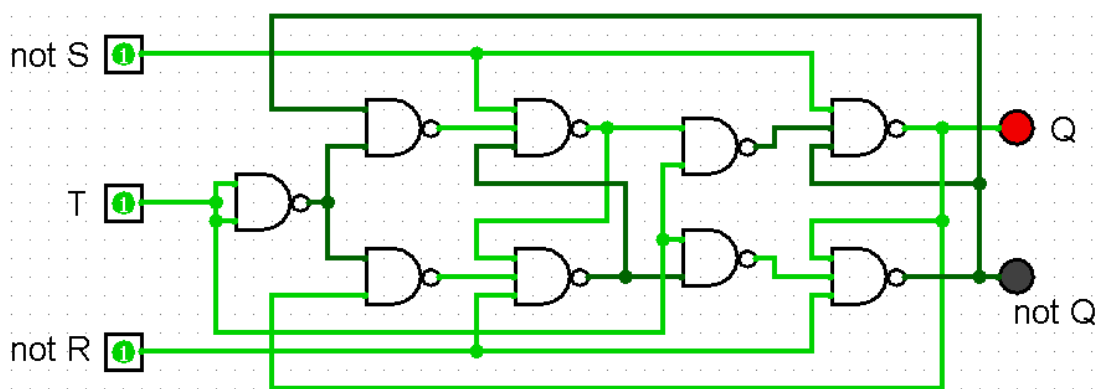


Рис.44 Т-триггер с асинхронными входами предустановки, выполненный на основе двухступенчатого RS-триггера

1.9. JK-триггер

Таблица переходов JK-триггера, собранного по схеме без инвертора, (табл. 14) и его функциональная схема.

Таблица 14

C	\bar{S}	\bar{R}	J	K	$Q(t+1)$	$\overline{Q(t+1)}$	Режим
*	0	0	*	*	1	1	Запрещенная комбинация
*	0	1	*	*	1	0	Асинхронная 1
*	1	0	*	*	0	1	Асинхронный 0
0	1	1	*	*	$Q(t)$	$\overline{Q(t)}$	Хранение
1	1	1	1	┐	0	1	Подмена входов C и K
1	1	1	┐	1	1	1	Подмена входов C и R
┐	1	1	0	1	0	1	Синхронная установка 0
┐	1	1	1	0	1	0	Синхронная установка 1
┐	1	1	1	1	1	1	Режим Т-триггера

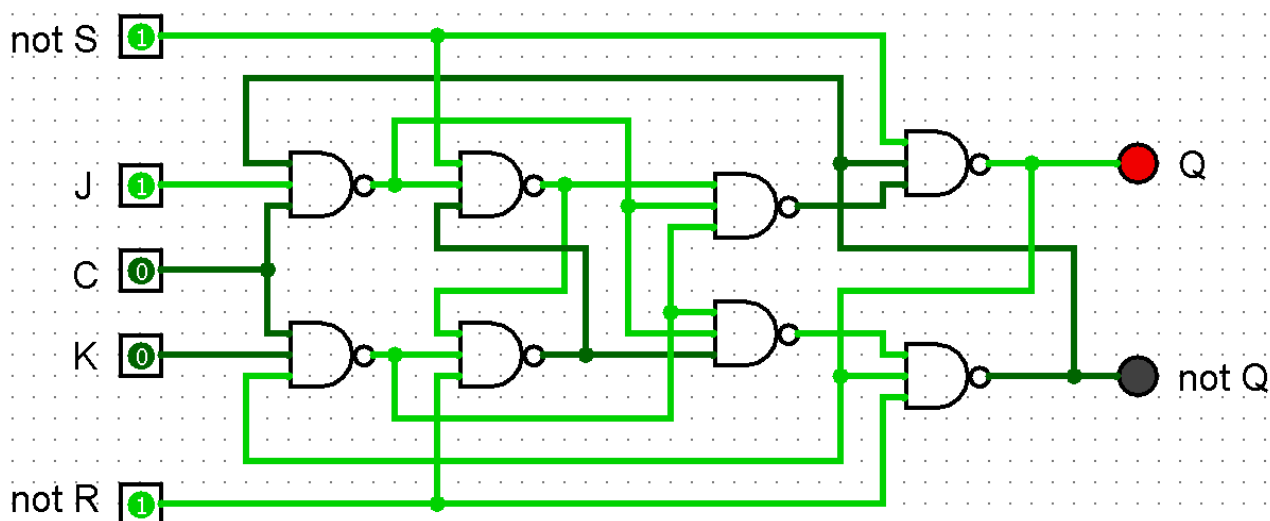


Рис. 45 JK-триггер, выполненный по схеме без инвертора

2. Последовательность выполнения работы

1. Запустить Logisim и создать в нем новый проект (создается по умолчанию).
2. Собрать в этом проекте перечисленные типы триггеров (рядом друг с другом).
3. Изучить режимы их работы, разобраться, как работает статическая и динамическая синхронизации, понять, как из одного триггера собрать другой.
4. Защитить практическую работу, отвечая на вопросы преподавателя.
5. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).

6. Получить роспись преподавателя в тетрадь учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.
3. Постановка задачи.
4. Схемы триггеров и их таблицы истинности.
5. Вопросы, заданные преподавателем.
6. Выводы.
7. Список информационных источников.

4. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое триггер?
- Классификация триггеров?
- За счет чего возникает эффект запоминания?
- Установите заданный триггер из некоторого начального состояния в противоположное требуемым способом (задание требуется выполнить при помощи минимального количества управляющих воздействий).
- Чем синхронный триггер отличается от асинхронного?
- Что такое статическая (динамическая) синхронизация? Виды динамической синхронизации? Достоинства и недостатки статической и динамической синхронизации?
- Как на основе одного триггера сделать другой?
- Что будет на выходах триггера, если на входы ему подать заданную последовательность сигналов?
- Из каких триггеров нельзя собрать Т-триггер и почему?
- Из каких триггеров нельзя собрать D-триггер и почему?
- Зачем необходимы двухступенчатые триггеры, почему нельзя обойтись только одноступенчатыми?
- Где используются триггеры?
- Другие вопросы по теме работы.

11. Практическая работа №11: синтез четырехразрядного счетчика с параллельным переносом между разрядами двумя способами

1. Постановка задачи

Разработать счетчик с параллельным переносом на D-триггерах двумя способами:

- с оптимальной схемой управления, выполненной на логических элементах общего базиса;
- со схемой управления, реализованной на преобразователе кодов (быстрая реализация, но не оптимальная схема).

В качестве исходных данных использовать индикатор CNT лабораторного комплекса, на котором слева направо отображены:

- направление счета (0 — сложение, 1 — вычитание);
- максимальное значение счетчика (не путать с модулем счета);
- шаг счета.

Протестировать работу схемы и убедиться в ее правильности. Подготовить отчет о проделанной работе и защитить ее.

2. Последовательность выполнения работы

1. Перед выполнением работы ознакомиться с приложениями 1-3.
2. Запустить лабораторный комплекс и получить персональные исходные данные для этой работы. Для этого требуется установить правильное название группы и ФИО студента. Рекомендуется зафиксировать личный код. В качестве исходных данных использовать показания индикатора CNT.
3. По исходным данным построить таблицу переходов состояний счетчика. При этом следует учесть, что:
 - начальным значением счетчика может быть любое двоичное число, меньшее или равное максимальному значению;
 - двоичные числа, большие максимального значения, не являются допустимыми состояниями.

Таким образом, таблица переходов счетчика (в зависимости от исходных данных) вполне может получиться частично определенной.

4. При помощи карт Карно спроектировать для каждого триггера в составе счетчика оптимальную схему управления.

5. Собрать в лабораторном комплексе счетчик с оптимальными схемами управления для триггеров. Рекомендации по сборке смотреть в примере далее. Через меню Logisim «Моделировать -> тактовая частота» задать частоту синхроимпульсов 16 Гц. Протестировать работу счетчика, для чего установить его в начальное состояние (CTRL-R) и запустить тактовый генератор, нажав «CTRL-K». Дождаться окончания распространения сигналов. В случае обнаружения ошибки индикатор правильности работы схемы не загорится (существующие в лабораторном комплексе индикаторы ошибок используются только для диагностики комбинационных схем). Обнаруженную ошибку необходимо устранить, для чего воспользоваться рекомендациями, приведенными в приложении 3.

6. Собрать другой вариант реализации счетчика - со схемой управления, построенной на преобразователе кодов, протестировать его работу аналогичным образом.

7. Продемонстрировать преподавателю процесс проектирования счетчика, а также правильность работы обеих его реализаций. Для этого необходимо показать восстановленную таблицу переходов, результаты минимизации управляющей логики, а также запустить в присутствии преподавателя процесс тестирования схемы. В случае успеха преподаватель даст разрешение на оформление отчета.

8. Оформить отчет по практической работе в соответствии с требуемым содержанием (см. далее).

9. Защитить практическую работу, отвечая на дополнительные вопросы, и получить роспись преподавателя в тетрадь учета.

3. Содержание отчета (отчет оформляется в электронном виде)

1. Титульный лист согласно требованиям Университета.
2. Содержание.
3. Постановка задачи и персональный вариант.
4. Таблица переходов счетчика.
5. Проектирование оптимальных схем управления триггерами (через минимизацию при помощи карт Карно).
6. Реализация счетчика с оптимальной схемой управления.
7. Реализация счетчика на преобразователей кодов.
8. Выводы.
9. Список информационных источников.

4. Пример

Предположим, что имеются следующие исходные данные:

- направление счета — сложение;
- максимальное значение — d (13 в десятичной системе);
- шаг счета — b (11 в десятичной системе).

По исходным данным восстановим таблицу переходов счетчика (табл. 15).

Таблица 15

$Q_3(t)$	$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	$Q_3(t+1)$	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	0	1	0	1	1
0	0	0	1	1	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	1	0	1	0
1	1	1	0	*	*	*	*
1	1	1	1	*	*	*	*

Таблица переходов является частично определенной: состояния 1110 и 1111 согласно исходным данным возникать никогда не должны, поэтому очередное состояние $Q(t+1)$ для этих случаев мы можем интерпретировать как нам удобно в целях минимизации управляющей логики.

Рассматриваем столбцы $Q_i(t+1)$ как самостоятельные функции от четырех переменных и проводим их минимизацию.

Также нам необходимо для каждой функции из двух возможных минимальных форм выбрать самую короткую.

Допустим, начнем с функции $Q_3(t+1)$. Оценим сложность минимальных форм, которые для нее получатся, по количеству переменных, входящих в них, и выберем оптимальную форму. Для этого построим необходимые карты Карно.

На рис. 46 показана карта для МДНФ функции $Q_3(t+1)$.

Q ₁ (t) Q ₀ (t)					
Q ₃ (t)	Q ₂ (t)	Q ₀ (t) Q ₁ (t)			
		00	01	11	10
00	00	1	1		1
	01				
	11	1	1	*	*
	10			1	

Рис.46 Карта Карно для МДНФ функции $Q_3(t+1)$

Пока не будем записывать формулу МДНФ, но оценим ее сложность. Это легко сделать, поскольку известно количество переменных, необходимых для описания каждого из интервалов. Напомним это количество для случая логической функции от четырех переменных:

- интервал размера 1 описывается четырьмя переменными;
- интервал размера 2 описывается тремя переменными;
- интервал размера 4 описывается двумя переменными;
- интервал размера 8 описывается одной переменной.

Из рисунка 46 видно, что в нашем случае МДНФ $Q_3(t+1)$ будет описана при помощи $2+3+3+3 = 11$ переменных либо их отрицаний.

Теперь сделаем аналогичную операцию для МКНФ этой же функции. Возьмем за основу уже построенную карту на рис. 46, ведь на всех пустых клетках там стоят нули. Кроме того, для повышения наглядности удалим единичные значения. Попробуем интерпретировать звездочки как нулевые значения функции. Выделим интервалы, получится следующий рис. 47.

Q ₁ (t) Q ₀ (t)					
Q ₃ (t)	Q ₂ (t)	Q ₀ (t) Q ₁ (t)			
		00	01	11	10
00	00			0	
	01	0	0	0	0
	11			*	*
	10	0	0		0

Рис.47 Карта Карно для МКНФ функции $Q_3(t+1)$

На рис.47 хорошо видно, что понимание звездочек как нулей не улучшает существующие интервалы, но приводит к ненужному усложнению: появляется лишний интервал размера четыре. Поэтому в данном случае звездочки следует интерпретировать как единичные значения. Итого имеем: один интервал размера четыре и три интервала размера три. Значит МКНФ будет иметь $2+3+3+3 = 11$ переменных либо их отрицаний, что эквивалентно сложности МДНФ. Следовательно, нам все равно, какую минимальную форму взять.

Запишем МДНФ для $Q_3(t+1)$ (формула 13).

$$Q_3(t+1)_{\text{МДНФ}} = Q_2(t) \cdot Q_3(t) + Q_0(t) \cdot Q_1(t) \cdot Q_3(t) + \overline{Q_0(t)} \cdot \overline{Q_2(t)} \cdot \overline{Q_3(t)} + \overline{Q_1(t)} \cdot \overline{Q_2(t)} \cdot \overline{Q_3(t)} \quad (13)$$

Далее по приведенной методике рассуждений рассмотрим функцию $Q_2(t+1)$. Сначала построим карту Карно для МДНФ (рис. 48).

$\begin{matrix} Q_1(t) \\ Q_0(t) \\ Q_3(t) \backslash Q_2(t) \end{matrix}$					
		00	01	11	10
00			1		1
01				1	
11				*	*
10	1	1			1

Рис.48 Карта Карно для МДНФ функции $Q_2(t+1)$

Оценим сложность МДНФ: $4 \cdot 3 = 12$ переменных или их отрицаний.

Теперь построим интервалы из нулевых значений и попытаемся интерпретировать звездочки как нули, чтобы построить МКНФ (рис.49).

$\begin{matrix} Q_1(t) \\ Q_0(t) \\ Q_3(t) \backslash Q_2(t) \end{matrix}$					
		00	01	11	10
00	0			0	
01	0	0			0
11	0	0		*	*
10				0	

Рис. 49 Карта Карно для МКНФ функции $Q_2(t+1)$

Оценим сложность МКНФ: $2 \cdot 2 + 2 \cdot 3 = 10$ переменных или их отрицаний.

Таким образом получается, что МКНФ для $Q_2(t+1)$ строить выгоднее, чем МДНФ.

Запишем формулу для МКНФ $Q_2(t+1)$ (формула 14):

$$Q_2(t+1)_{\text{МКНФ}} = (Q_1(t) + \overline{Q_2(t)}) \cdot (Q_0(t) + \overline{Q_2(t)}) \cdot (\overline{Q_0(t)} + \overline{Q_1(t)} + Q_2(t)) \cdot (Q_0(t) + Q_1(t) + Q_3(t)) \quad (14)$$

Переходим к рассмотрению $Q_1(t+1)$. Построим карту Карно для записи МДНФ этой функции (рис. 50).

<div>Q₁(t)</div> <div>Q₀(t)</div> <div>Q₃(t) \ Q₂(t)</div>	00	01	11	10
00	1			
01		1		1
11		1	*	*
10		1		1

Рис. 50 Карта Карно для МДНФ функции $Q_1(t+1)$

Оценим сложность МДНФ: $3 \cdot 4 + 4 = 16$ переменных или их отрицаний.

Построим карту Карно для МКНФ функции $Q_1(t+1)$ (рис. 51).

<div>Q₁(t)</div> <div>Q₀(t)</div> <div>Q₃(t) \ Q₂(t)</div>	00	01	11	10
00		0	0	0
01	0		0	
11	0		*	*
10	0		0	

Рис. 51 Карта Карно для МКНФ функции $Q_1(t+1)$

Оценим сложность МКНФ: $4 \cdot 3 + 2 = 14$ переменных или их отрицаний.

Таким образом получается, что МКНФ для $Q_1(t+1)$ строить выгоднее, чем МДНФ.

Запишем формулу для МКНФ $Q_1(t+1)$ (формула 15):

$$Q_1(t+1)_{\text{МКНФ}} = (\overline{Q_0(t)} + \overline{Q_1(t)}) \cdot (\overline{Q_0(t)} + Q_2(t) + Q_3(t)) \cdot (\overline{Q_1(t)} + Q_2(t) + Q_3(t)) \cdot (Q_0(t) + Q_1(t) + \overline{Q_2(t)}) \cdot (Q_0(t) + Q_1(t) + \overline{Q_3(t)}) \quad (15)$$

При помощи полученных формул выполним реализацию схем управления для триггеров счетчика (рис. 52).

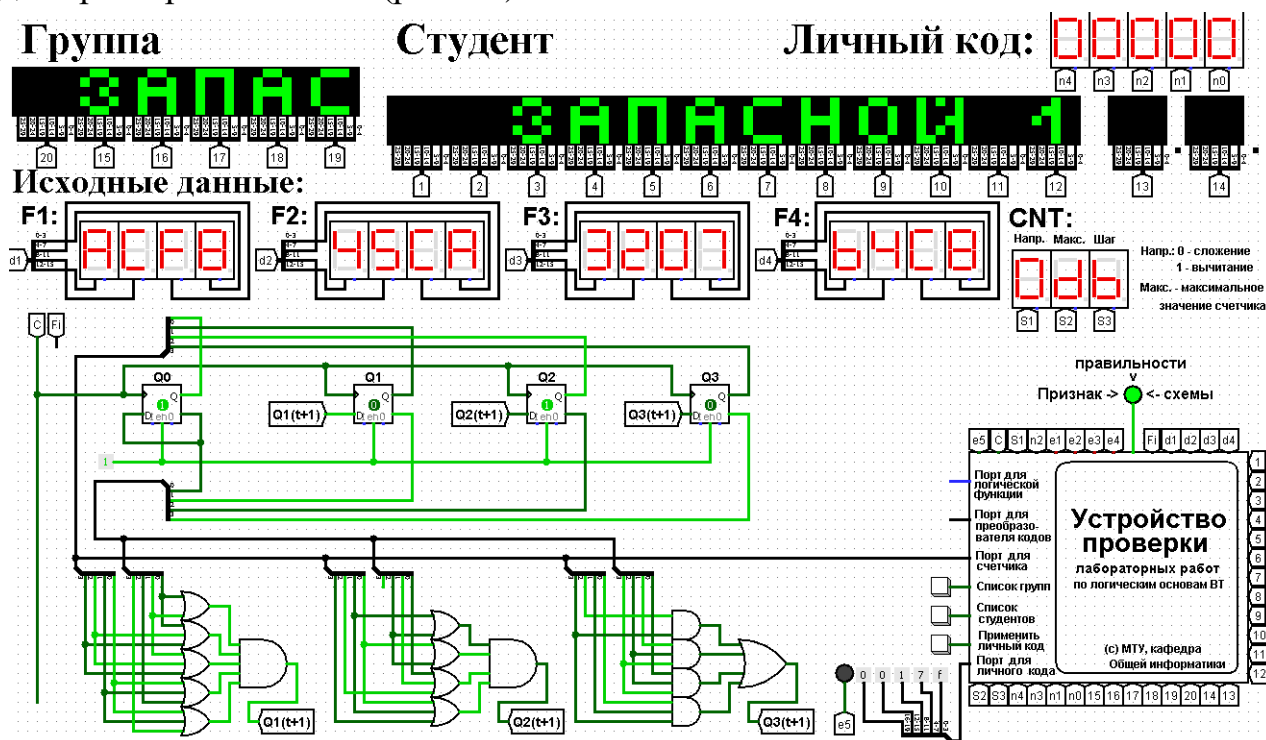


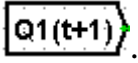
Рис. 52 Схема счетчика с подключением к устройству проверки

Как видно из рисунка, тестирование показало правильность работы схемы.

Рекомендуется следующий порядок сбора схемы счетчика. Сначала в верхней части рабочей области на определенном расстоянии друг от друга располагаются триггеры. Всем триггерам сразу подаем 1 на разрешающий вход «en», расположенный в их нижней части, а также подводим сигнал синхронизации «С» (крайний слева провод на схеме комплекса). Триггерам желательно задать текстовую метку с именем разряда счетчика, за который они отвечают. Прямые и инверсные состояния триггеров рекомендуется собрать в шины при помощи разветвителей (состояние триггера Q_0 на нулевой разряд разветвителя и т.д.). Далее сигналы с этих шин будут поданы на схемы управления триггерами, а шина, содержащее прямое состояние счетчика, также будет подключена к соответствующему порту устройства проверки. Схемы управления триггерами желательно (по

возможности) располагать в нижней части рабочей области, под соответствующим триггером. Схемы управления синтезируются по формулам, полученным на этапе минимизации.

В рассматриваемом примере $Q_0(t+1) = \overline{Q_0(t)}$, что видно сразу из таблицы переходов, поэтому никакой схемы управления не требуется – просто подключаем инверсный выход триггера Q_0 к его входу D. Для триггеров Q_1 , Q_2 и Q_3 уже требуются схемы управления, причем в двух случаях эта схема реализована на основе МКНФ, а в одном на основе МДНФ.

Результаты работы схем управления необходимо подать на входы D соответствующих триггеров. Чтобы избежать запутанного пересечения проводов, для этой цели в рассматриваемом примере был использован конструктивный элемент «туннель». Туннель изображается поименованным пятиугольником, например, так: .

Подразумевается, что между двумя одноименными туннелями пролегает провод, который просто не изображается из соображений экономии места на рабочей области, а также в целях повышения читаемости схемы. Как Вы можете заметить, в составе схемы лабораторного комплекса используется много туннелей. Однако злоупотреблять этой возможностью тоже не нужно, поскольку бездумное использование туннелей может привести к ухудшению понимания работы схемы.

При реализации счетчика на рабочей области лабораторного комплекса допустимы и любые другие варианты взаиморасположения и соединения элементов, которые не нарушают удобочитаемости схемы.

Выполним быструю реализацию счетчика при помощи преобразователя кодов в качестве схемы управления триггерами.

Здесь не требуется никакая минимизация, необходимо просто по таблице переходов правильно соединить выходы дешифратора со входами шифратора.

Таким образом, можно сразу построить схему счетчика (рис.53).

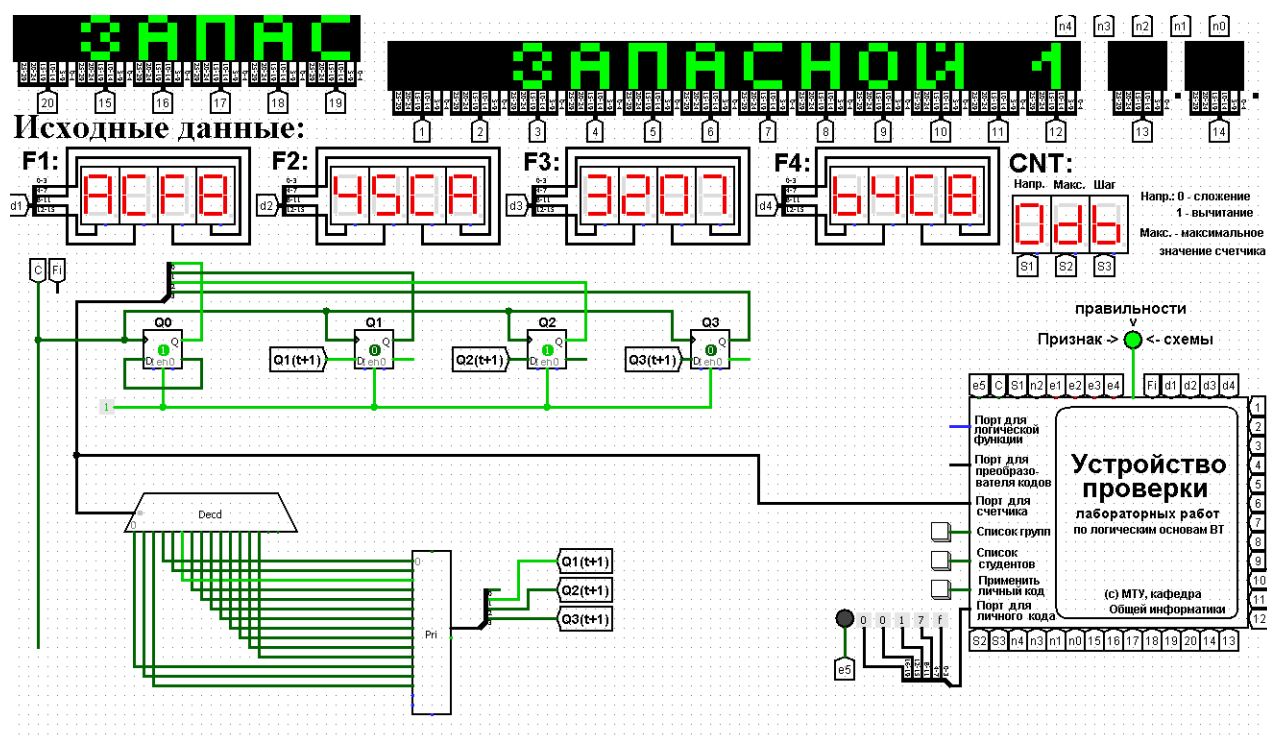


Рис. 53 Счетчик со схемой управления, выполненной на преобразователе кодов

Обратите внимание, для данной реализации счетчика уже не требуются инверсные выходы триггеров.

Тестирование показало, что схема работает правильно.

5. Примеры дополнительных вопросов, которые могут возникнуть на защите практической работы:

- Что такое цифровой автомат?
- Что такое счетчик?
- Приведите пример простейшего счетчика?
- Перечислить основные параметры счетчика и дать им определение.
- Классификация счетчиков.
- Что такое счетчик с последовательным переносом?
- Что такое счетчик с параллельным переносом?
- На каких триггерах можно (нельзя) сделать счетчик с последовательным переносом и почему?
- На каких триггерах можно (нельзя) сделать счетчик с параллельным переносом и почему?
- Заданы максимальное значение и шаг счетчика. Требуется определить его модуль счета.
- Другие вопросы и задачи по теме работы.

РАЗДЕЛ №3: АЛГОРИТМИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Введение к разделу

В рамках рассматриваемого раздела студент должен выполнить всего одну работу, посвященную разработке алгоритма решения учебной задачи по обработке данных и его реализации на любом известном языке программирования, поддерживающем структурно-модульный подход. Для реализации также допускается использование учебных алгоритмических языков: Кумир, Scratch, Исполнители.

На территории Российской Федерации действует единая система программной документации (ЕСПД), частью которой является Государственный стандарт — ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем». Этот ГОСТ практически полностью соответствует международному стандарту ISO 5807:1985.

Разрабатываемый в рамках данной работы алгоритм должен быть представлен в виде блок-схемы, изображенной по правилам указанного ГОСТа. Рассмотрим условные графические обозначения, взятые из ГОСТа, которые, скорее всего, потребуются для изображения алгоритма.

Терминальный блок (терминатор)

Символ, с которого начинается и заканчивается алгоритм любой программы или подпрограммы (рис. 54). Тип возвращаемого значения (для алгоритма функции) и список аргументов, передаваемых в подпрограмму, обычно указывается в комментариях к блоку терминатора. Блоки «начало» и «конец» у любого алгоритма могут быть только одним экземпляре.

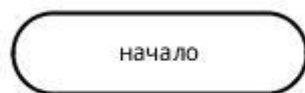


Рис. 54 Терминатор начала и конца работы программы (процедуры, функции)

Блок ввода/вывода

В ГОСТ определено множество символов ввода/вывода, например, вывод на магнитные ленты, дисплеи и т.п. Если источник данных не принципиален, то обычно используется один символ в виде параллелограмма (рис. 55). Подробности ввода/вывода могут быть указаны в комментариях.

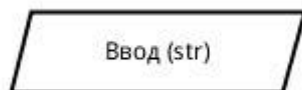


Рис. 55 Ввод и вывод данных

Блок операций

В блоке операций (рис. 56) обычно размещают одну или несколько (ГОСТ не запрещает) операций присваивания, не требующих вызова внешних функций. Справа от знака присваивания могут присутствовать не только константы, но и математические выражения. Присваивания результатов вызова пользовательских функций, имеющих собственный алгоритм, должны оформляться отдельным блоком (см. далее).

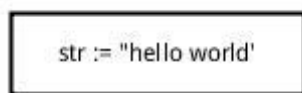


Рис. 56 Присваивание результатов выполнения операций над данными

Блок селектор

Используется для разветвления алгоритма на две или более ветки (рис. 57). Блок в виде ромба имеет один вход и несколько подписанных выходов. В случае, если блок имеет 2 выхода (соответствует разветвлению по результатам проверки условия), то на выходах подписывается результат сравнения — «да/нет». Если из блока выходит большее число линий (оператор множественного выбора), внутри него записывается имя переменной, а на выходящих дугах — значения этой переменной.

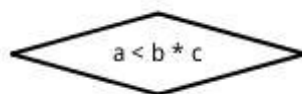


Рис. 57 Блок, иллюстрирующий ветвление алгоритма

Блок вызова подпрограмм

Вызов процедур и функций помещается в прямоугольник с дополнительными вертикальными линиями. Если вызывается функция, то результаты вызова должны быть присвоены переменной (рис. 58).

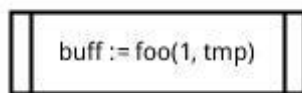


Рис. 58 Вызов внешней процедуры

Блоки ограничители условных циклов

Символы начала и конца цикла содержат имя цикла и условие (рис. 59). Имя цикла помогает более четко выделить границы данного цикла (что особенно важно в случае нескольких вложенных циклов), а также служит кратким комментарием, описывающим смысл цикла. Условие может присутствовать лишь в одном из символов пары. Условие представляет собой логическое выражение, в зависимости от результатов вычисления которого цикл либо продолжается, либо заканчивается. Если необходимо описать цикл с предусловием, то условие располагается в верхнем блоке. Если необходимо описать цикл с постусловием, то условие располагается в нижнем блоке. Для описания цикла со счетчиком используется блок подготовки данных (см. далее).

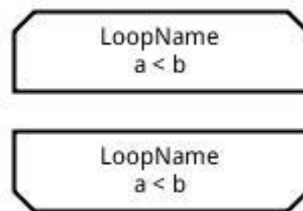


Рис. 59 Начало и конец цикла

Блок подготовки данных

Символ «подготовка данных» (рис. 60) используется либо для объявления типизированных переменных, либо как заголовок цикла со счетчиком. При объявлении цикла со счетчиком в блоке должны быть указаны имя переменной-счетчика и диапазон ее изменения, а также может быть указан шаг изменения. Если шаг не указан, то по умолчанию он считается равным единице.

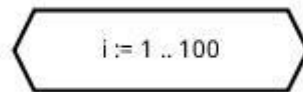


Рис. 60 Блок подготовки данных в качестве заголовка цикла со счетчиком.

Блок коннектор

В случае, если блок-схема не умещается на лист, используется специальный символ-соединитель (коннектор), отражающий переход потока управления между листами (рис. 61). Символ может использоваться и на одном листе, если по каким-либо причинам неудобно тянуть линию из одного места листа в другое. Порядок использования коннектора следующий. На месте, где требуется сделать разрыв алгоритма, линия, исходящая из предыдущего блока, упирается в поименованный коннектор. После коннектора идет разрыв алгоритма. В другом месте,

где требуется продолжить изображение данного алгоритма, опять ставится коннектор с этим же именем. Этот коннектор служит как бы блоком «начало» для следующей части алгоритма.

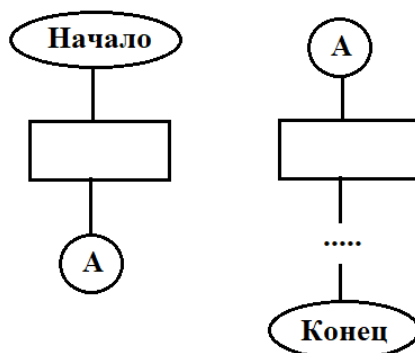


Рис. 61 Использование коннектора

Комментарий

Текст комментария как бы ограничивается квадратной скобкой, от которой тянется пунктирная линия, охватывающая один или несколько блоков алгоритма, подлежащих комментированию (рис. 62).

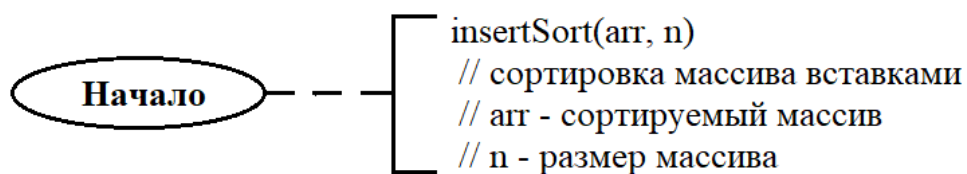


Рис. 62 Пример комментария в начале функции

Требования к изображению линий

Пунктирная линия используется для соединения блока алгоритма с комментарием. Сплошная линия отражает зависимости по управлению между отдельными блоками алгоритма и может снабжаться стрелкой. Стрелку можно не указывать при направлении дуги слева направо и сверху вниз. Линии должны подходить к блоку строго слева, либо сверху, а исходить – только снизу, либо справа.

12. Практическая работа №12: элементы алгоритмизации и процедурного программирования

1. Постановка задачи

Требуется разработать блок-схему алгоритма и написать программу обработки данных в соответствии с выбранным и согласованным с преподавателем вариантом (см. далее). При этом требуется контролировать типы и диапазоны вводимых данных, а также предусмотреть обработку других исключительных ситуаций (если они есть), например, ситуацию деления на ноль. Блок-схема должна быть полной, т.е. должна описывать и процесс диалога с пользователем, и контроль вводимых данных, и подпрограммы вычислений с обработкой возможных исключительных операций. Блок-схема должна изображаться по ГОСТу. При обнаружении ошибки ввода или ошибки вычислений программа должна информативно уведомлять пользователя о причине ошибки. Если ошибка произошла на этапе ввода данных, то программа должна просить пользователя повторить ввод.

2. Варианты заданий

2.1. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона $[2, 5]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[0, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать каждую строку матрицы по возрастанию, т.е. минимальный элемент должен находиться в начале строки, максимальный – в конце. После сортировки найти N -ный минимум в заданной строке, где номер строки и N задается пользователем. Результаты обработки матрицы вывести на экран.

2.2. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона $[2, 5]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[0, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать каждый столбец матрицы по убыванию, т.е. максимальный элемент должен находиться вверху столбца, минимальный – снизу. После сортировки найти N -ный максимум в заданном столбце, где номер столбца и N задается пользователем. Результаты обработки матрицы вывести на экран.

2.3. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона $[2, 5]$. Конкретный размер матрицы задается пользователем. Матрица исходно содержит только целые числа из диапазона $[1, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать по возрастанию элементы, лежащие на главной диагонали матрицы, остальные

элементы умножить на минус один. После сортировки найти N-ный максимум на главной диагонали, где N задается пользователем. Результаты обработки матрицы вывести на экран.

2.4. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона $[2, 5]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[1, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать по убыванию элементы, лежащие на побочной диагонали матрицы, остальные элементы умножить на минус один. После сортировки найти N-ный минимум среди элементов на побочной диагонали, где N задается пользователем. Результаты обработки матрицы вывести на экран.

2.5. Создать квадратную матрицу размера $M \times M$, где M является четным числом из диапазона $[2, 8]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[1, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать элементы матрицы по возрастанию и перераспределить их таким образом, чтобы в ее верхней половине располагались только старшие элементы, а в нижней – только младшие. Результаты обработки матрицы вывести на экран.

2.6. Создать квадратную матрицу размера $M \times M$, где M является четным числом из диапазона $[2, 8]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[1, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать элементы матрицы по убыванию и перераспределить их таким образом, чтобы в ее левой половине располагались только младшие элементы, а в правой – только старшие. Результаты обработки матрицы вывести на экран.

2.7. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона $[2, 5]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[1, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать по убыванию элементы, принадлежащие или лежащие ниже главной диагонали матрицы, остальные элементы умножить на минус один. Результаты обработки матрицы вывести на экран.

2.8. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона $[2, 5]$. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона $[1, 100]$, которые могут быть как случайными, так и вводиться пользователем. Отсортировать по возрастанию элементы, принадлежащие или лежащие выше побочной диагонали матрицы,

остальные элементы умножить на минус один. Результаты обработки матрицы вывести на экран.

2.9. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона [1, 100], которые могут быть как случайными, так и вводиться пользователем. На основе созданной матрицы сделать одномерный массив, в котором сначала идут все четные элементы матрицы, упорядоченные по возрастанию, а потом идут все нечетные элементы матрицы, упорядоченные по убыванию. Результаты обработки матрицы вывести на экран.

2.10. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица содержит только целые числа из диапазона [1, 100], которые могут быть как случайными, так и вводиться пользователем. На основе созданной матрицы сделать одномерный массив, в котором в первой половине расположены суммы элементов матрицы, стоящих в строках, а во второй половине – суммы элементов матрицы по столбцам. Первую половину массива упорядочить по убыванию, а вторую – по возрастанию. Результаты обработки матрицы вывести на экран.

2.11. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 4х букв английского алфавита, которые могут быть как случайными, так и вводиться пользователем. На основе матрицы сформировать одномерный массив, куда должны войти слова, содержащие определенную букву, которую задает пользователь. Полученный массив отсортировать по возрастанию количества гласных букв в словах. Результаты обработки матрицы вывести на экран.

2.12. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 4х букв английского алфавита, которые могут быть как случайными, так и вводиться пользователем. На основе матрицы сформировать одномерный массив, куда должны войти слова, **не** содержащие определенную букву, которую задает пользователь. Полученный массив отсортировать по убыванию количества согласных букв в словах. Результаты обработки матрицы вывести на экран.

2.13. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 4х букв английского алфавита, которые

могут быть как случайными, так и вводиться пользователем. На основе матрицы сформировать одномерный массив, состоящий из количеств гласных букв в словах. Полученный массив отсортировать по возрастанию. Результаты обработки матрицы вывести на экран.

2.14. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 4х букв английского алфавита, которые могут быть как случайными, так и вводиться пользователем. На основе матрицы сформировать одномерный массив, состоящий из количеств согласных букв в словах. Полученный массив отсортировать по убыванию. Результаты обработки матрицы вывести на экран.

2.15. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 4х букв английского алфавита, которые могут быть как случайными, так и вводиться пользователем. Полученную матрицу обработать следующим образом. Из каждого слова удалить все гласные буквы. Результат удаления вывести на экран. На основе измененной матрицы сформировать массив, состоящий из размеров строк матрицы, который необходимо упорядочить по убыванию. Результаты обработки вывести на экран.

2.16. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 4х букв английского алфавита, которые могут быть как случайными, так и вводиться пользователем. Полученную матрицу обработать следующим образом. Каждое слово матрицы заменить на число, образованное из кодов букв этого слова. Элементы полученной новой матрицы требуется упорядочить по возрастанию. Результаты обработки вывести на экран.

2.17. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 5ти букв английского алфавита, которые могут быть как случайными, так и вводиться пользователем. Полученную матрицу обработать следующим образом. Каждое слово матрицы заменить на другое слово, образованное из его букв, упорядоченных по алфавиту. Также требуется подсчитать общее количество гласных и согласных букв в исходной матрице. Результаты работы вывести на экран.

2.18. Создать квадратную матрицу размера $M \times M$, где M является целым числом из диапазона [2,5]. Конкретный размер матрицы задается пользователем. Матрица должна содержать слова из 5ти букв английского алфавита, которые

могут быть как случайными, так и вводиться пользователем. Полученную матрицу обработать следующим образом. Каждое слово матрицы заменить на число, у которого в разряде десятков стоит количество гласных букв данного слова, а в разряде единиц – количество согласных букв данного слова. Каждую строку новой матрицы упорядочить по возрастанию. Результаты работы вывести на экран.

2.19. Имеется массив из пяти пятибуквенных строк, составленных из букв английского алфавита. Строки могут как вводиться пользователем, так и генерироваться случайным образом. Необходимо из исходного массива сформировать матрицу размера 5 на 5, в которой каждый элемент — это код соответствующей буквы соответствующего слова. Например, по индексам [1,1] в требуемой матрице должен лежать код первой буквы первого слова. Далее требуется упорядочить строки полученной матрицы по убыванию. Результаты работы вывести на экран.

2.20. Создать квадратную матрицу размера 6 на 6. Матрица может заполняться как числами, вводимыми с клавиатуры, так и случайными числами (на выбор пользователя) из диапазона от 0 до 100. Над данной матрицей выполнить следующие преобразования. Поделить четные строки на нечетные, в результате чего получить матрицу размера 3 на 6. Далее поделить четные столбцы на нечетные, в результате чего получить матрицу 3 на 3. Столбцы полученной матрицы требуется упорядочить по возрастанию элементов. Предусмотреть корректную реакцию программы в случае возникновения деления на 0. Результаты преобразований вывести на экран.

2.21. Создать квадратную матрицу размера 4 на 4. Матрица может заполняться как числами, вводимыми с клавиатуры, так и случайными числами (на выбор пользователя) из диапазона от 0 до 100. Требуется найти в этой матрице все простые числа, сформировать из них отдельный массив и упорядочить его по убыванию. Алгоритм проверки чисел на принадлежность к простым должен быть универсальным, т.е. не должен зависеть от диапазона вводимых чисел. Результаты преобразований вывести на экран.

2.22. Создать квадратную матрицу размера 4 на 4. Матрица может заполняться как числами, вводимыми с клавиатуры, так и случайными числами (на выбор пользователя) из диапазона от 0 до 100. Требуется исключить из этой матрицы все числа, принадлежащие ряду Фибоначчи, из оставшихся сформировать отдельный массив и упорядочить его по убыванию. Алгоритм проверки на принадлежность к ряду Фибоначчи должен быть универсальным, т.е. не должен зависеть от диапазона вводимых чисел. Результаты преобразований вывести на экран.

2.23. Создать квадратную матрицу размера 8 на 8. Матрица заполняется случайными целыми числами в диапазоне от 1 до 100. На матрицу накладывается разметка, соответствующая шахматной доске. Пользователь выбирает на матрице диагональ, задавая координаты двух клеток (ее начала и конца). Например, A5-E1. Требуется упорядочить элементы матрицы, принадлежащие заданной диагонали, а остальные элементы обнулить. Результаты преобразований вывести на экран.

2.24. Создать квадратную матрицу размера 8 на 8. Матрица заполняется случайными целыми числами в диапазоне от 1 до 100. На матрицу накладывается разметка, соответствующая шахматной доске. Пользователь выбирает цвет клеток, с которыми будет происходить дальнейшая работа. Требуется из элементов матрицы, стоящих на клетках заданного цвета, сформировать одномерный массив и упорядочить его методом быстрой сортировки по возрастанию. Результаты преобразований вывести на экран.

2.25. Создать квадратную матрицу размера 4 на 4. Матрица заполняется двузначными шестнадцатеричными числами, которые могут как вводиться вручную, так и генерироваться случайно. Отсортировать строки с четными номерами по возрастанию, а с нечетными по убыванию при помощи алгоритма быстрой сортировки. Результаты преобразований вывести на экран.

2.26. Создать квадратную матрицу размера 4 на 4. Матрица заполняется двузначными шестнадцатеричными числами, которые могут как вводиться вручную, так и генерироваться случайно. Отсортировать столбцы с нечетными номерами по возрастанию, а с четными по убыванию при помощи алгоритма пузырьковой сортировки. Результаты преобразований вывести на экран.

2.27. Создать квадратную матрицу размера 4 на 4. Матрица заполняется двузначными шестнадцатеричными числами, которые могут как вводиться вручную, так и генерироваться случайно. Упорядочить матрицу таким образом, чтобы в ее левой половине содержались меньшие элементы, а в правой большие. Результаты преобразований вывести на экран.

2.28. Сформировать одномерный массив размера N , где N выбирается пользователем из диапазона $[1, 5]$. Массив содержит числа в формате половинной точности в шестнадцатеричной записи. Массив заполняется пользователем вручную. Преобразовать содержимое массива в десятичную систему и упорядочить по убыванию. Результаты преобразований вывести на экран.

2.29. Сформировать матрицу размера 4 на 4. Матрица заполняется вручную строками, каждая из которых может иметь не более 5 символов в длину. Допустимыми символами являются цифры и буквы английского алфавита.

Каждая строка рассматривается как число в некоторой системе счисления. Проанализировать содержимое данной матрицы и сформировать на ее основе другую, элементами которой будут числа, являющиеся минимально возможными основаниями системы счисления для соответствующих элементов исходной матрицы. Результаты преобразований вывести на экран.

2.30. Матрица размера 4 на 4 содержит цифры от 0 до 9. Матрица заполняется вручную за одну операцию ввода следующим образом. Вводится строка, содержащая группы стоящих рядом цифр, разделенных пробелами. Каждая группа цифр описывает строку матрицы, а пробел служит разделителем строк. Например, вводится строка “1234 6789 5432 0987 4455”. Это значит, что первая строка матрицы будет содержать цифры [1,2,3,4], вторая строка будет содержать [6,7,8,9] и так далее. Каждый четный столбец отсортировать по возрастанию, каждый нечетный столбец отсортировать по убыванию. Для сортировки использовать алгоритм Хоара. Результаты преобразований вывести на экран.

3. Содержание отчета

Отчет по практической работе должен включать:

- титульный лист в соответствии с требованиями Университета;
- содержание;
- постановку задачи;
- блок-схемы алгоритмов программы, выполненные по ГОСТу (рекомендуется не пользоваться программами, которые строят блок-схемы по коду, поскольку они делают это с нарушениями ГОСТа);
- структурированный код программы с комментариями;
- примеры тестирования, доказывающие работоспособность;
- выводы о проделанной работе;
- информационные источники.

4. Порядок выполнения

- Выбрать индивидуальное задание из списка предлагаемых вариантов (либо предложить свое аналогичное), согласовать его с преподавателем.
- Разработать алгоритм решения и зарисовать его в виде блок-схемы по ГОСТу.
- Запрограммировать алгоритм на любом языке, поддерживающем процедурно-модульный подход, либо на учебном алгоритмическом языке.
- Протестировать работу программы на предмет ее соответствия функциональному назначению. Подготовить соответствующие примеры для отчета.

- Протестировать работу программы на предмет корректной обработки некорректного ввода данных. Подготовить соответствующие примеры для отчета.
- Подготовить отчет о проделанной работе.
- Продемонстрировать работу программы преподавателю, защитить практическую работу, отвечая на дополнительные вопросы.
- Получить подпись преподавателя в тетрадь учета.

5. Примеры вопросов, возможных на защите

- Что такое алгоритм? Какие у него есть свойства?
- Виды алгоритмов.
- Что такое блок-схема и зачем она нужна?
- Задана блок-схема некоторого простого алгоритма. Что будет в результате исполнения этого алгоритма?
- Сформулируйте принципы структурного программирования Дейкстры.
- Чем процедура отличается от функции?
- Способы передачи аргументов в подпрограмму при ее вызове.
- Что такое рекурсия, какие существуют виды рекурсии?
- Как будет работать программа, если изменить в ней определенный фрагмент?
- Другие вопросы по теме работы.

ПРИЛОЖЕНИЕ 1.

ОПИСАНИЕ ЛАБОРАТОРНОГО КОМПЛЕКСА

Практические работы по логическим основам вычислительной техники проходят с использованием свободно распространяемой среды схемотехнического моделирования Logisim, в которой в целях автоматизирования учебного процесса реализована модель лабораторного стенда (МЛС).

Среда Logisim может быть скачена с сайта производителя по ссылке «<http://www.cburch.com/logisim/ru/>». Там же находится подробная русскоязычное руководство пользователя.

Модель лабораторного стенда поставляется как совокупность *.circ файлов для среды Logisim. Для запуска МЛС необходимо предварительно запустить среду Logisim, а уже из нее открыть файл start.circ.

Остальные файлы открывать ненужно, повреждение или отсутствие любого из файлов приведет к неработоспособности системы.

МЛС реализует следующие функции:

- Предоставление студенту индивидуального задания в зависимости от его группы и имени, которые выбираются путем пролистывания соответствующих списков.
- Предоставление студенту возможностей Logisim по разработке цифровой схемы, соответствующей полученному заданию.
- Автоматическая проверка правильности работы схемы, разработанной студентом, и индикация результатов проверки.

МЛС позволяет проводить практические работы по следующим темам:

- Построение всевозможных комбинационных схем, реализующих заданную функцию от четырех переменных разными способами.
- Построение четырехразрядных преобразователей кодов;
- Построение четырехразрядных счетчиков с заданным максимальным значением, шагом и направлением счета.

Всего система способна поддерживать работу максимум с 1500 студентами.

Информационное наполнение МЛС данными о студентах и вариантах заданий производится преподавателем при помощи дополнительного программного обеспечения.

Внешний вид МЛС в среде Logisim показан на рис. 63.

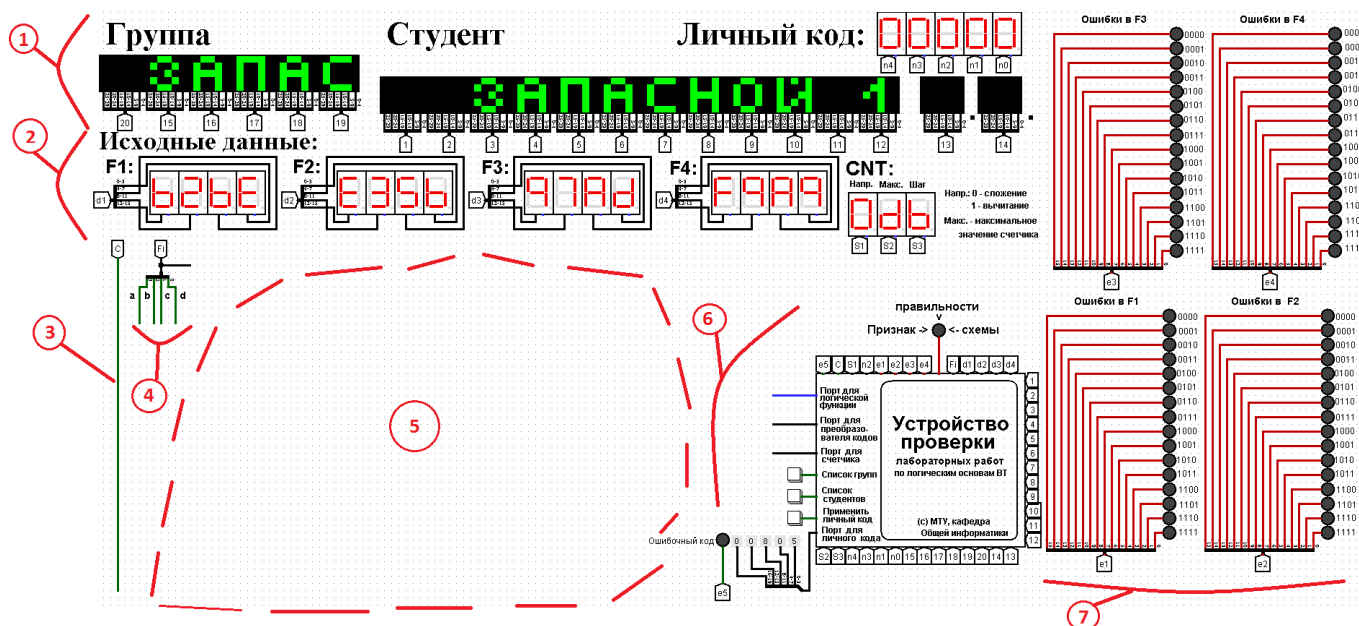


Рис. 63 Общий вид МЛС в среде Logisim с выделенными фрагментами

На рис.63 нумерованными областями показано:

- (1) — Совокупность индикаторов «Группа», «Студент», «Личный код». Индикатор «Группа» отображает название выбранной группы. Индикатор «Студент» отображает фамилию и инициалы студента, выбранного из группы. Индикатор «Личный код» отображает личный код выбранного студента. Личный код может быть использован для быстрого доступа к варианту без пролистывания списков групп и фамилий.
- (2) — Совокупность индикаторов «Исходные данные». Индикаторы F1-F4 используются для отображения значений логических функций от четырех переменных, заданных векторно (в виде 16-теричной свертки). Индикатор CNT отображает исходные данные для проектирования счетчика.
- (3) — Источник сигнала синхронизации, необходимый при выполнении заданий со счетчиками.
- (4) — источник значений аргументов для логических функций.
- (5) — рабочая область для реализации студентом своей схемы (может быть гораздо больше, всё определяется размером области проекта, которую предоставляет Logisim).
- (6) — устройство управления и автоматической проверки (УУАП - будет рассмотрено отдельно).
- (7) — группа индикаторов для диагностики ошибок в проектируемых схемах.

Рассмотрим более подробно УУАП (рис. 64).

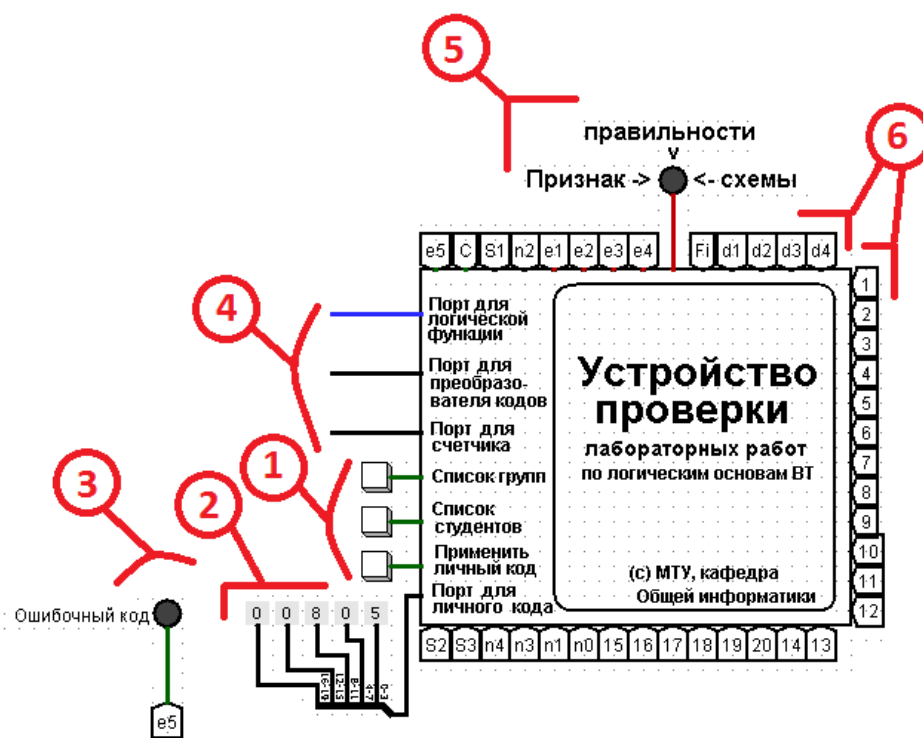


Рис. 64 УУАП крупным планом

На рис. 64 нумерованными областями показано:

- (1) — Кнопки, которые позволяют листать список групп и список студентов в группе. Пролистывание закольцовано и происходит только в одном направлении. Также имеется кнопка для применения личного кода студента. В результате применения личного кода система быстро переключается на вариант соответствующего студента, т.е. не придется листать списки.
- (2) — Группа шестнадцатеричных констант, доступных для редактирования, при помощи которых можно задать личный код студента.
- (3) — Индикатор ошибочного личного кода. Загорается при попытке ввести несуществующий код.
- (4) — Порты для подключения к УУАП схемы, разработанной студентом, с целью ее проверки. Для корректной работы УУАП порты должны задействоваться *по одному*, т.е. нельзя проверять разнотипные схемы одновременно (например, преобразователь кодов и счетчик). Порт для проверки логической функции представляет собой единичный провод, остальные же являются шиной на 4 разряда.
- (5) — Индикатор правильности схемы. Загорается, если проверка показала, что разработанная студентом схема работает в соответствии с исходными данными.
- (6) — По трем сторонам УУАП находятся контакты, которые подключены к разным другим частям МЛС, реализующим интерфейс системы с пользователем и проверяемой схемой.

ПРИЛОЖЕНИЕ 2.

ОБОБЩЕННАЯ МЕТОДИКА ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ ПРИ ПОМОЩИ МОДЕЛИ ЛАБОРАТОРНОГО СТЕНДА

В процессе выполнения практических работ по разделу «Логические основы вычислительной техники» при помощи модели лабораторного стенда необходимо придерживаться следующей методики.

1. Запустить МЛС путем загрузки файла start.circ из среды Logisim.
2. Если это Ваш первый сеанс работы с системой, то при помощи кнопок, показанных на рис. 64 (маркер 1), и, глядя на соответствующие индикаторы, требуется найти сначала свою группу, а потом свою фамилию. Листание списков закольцовано. Если не успели остановиться вовремя, то продолжайте листать список до очередного появления нужного имени.

Представление фамилии в системе ограничено 12тью знаками, поэтому более длинные фамилии будут урезаны до 11ти знаков, после чего будет поставлена точка.

Представление названий групп ограничено шестью символами: год и разделительные дефисы не отображаются.

Если Вы не нашли свою фамилию в вашей группе, то, возможно, Вы по ошибке были причислены к другой группе. Чтобы это проверить, попробуйте предпринять поиск по файлу индивидуальных заданий из раздела «Арифметические основы вычислительной техники». Если и в этом случае Вы не нашли свою фамилию, то обратитесь к преподавателю, чтобы он назначил Вам запасной вариант.

Когда Вы впервые нашли в системе свою фамилию или назначенный Вам запасной вариант, то **обязательно** обратите внимание на **личный код**, который будет показан в правом верхнем углу (см. рис. 63). Этот код упростит Вам работу с системой при многочисленных перезапусках моделирования.

Как только Вы узнали личный код, то лучше всего сразу зафиксировать его при помощи набора шестнадцатеричных констант (см. рис. 64 маркер 2).

Делается это так. Последовательно выбираем мышкой необходимые ряды (например, см. рис. 65).

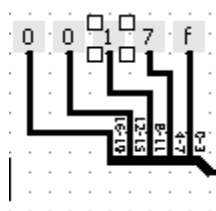


Рис. 65. Выбранный разряд обозначен контурными квадратами. В данном случае это шестнадцатеричная константа 1.

При этом активируется панель свойств в левой части экрана (рис. 66).

Выделение: Константа	
Направление	Юг
Биты данных	4
Значение	0x1

Рис. 66 Панель свойств элемента «константа»

На этой панели нас интересует лишь значение константы, записанное в шестнадцатеричной нотации в данном случае как 0x1. Аккуратно изменим это значение на нужное (в соответствии с личным кодом), например, просто обнулим (рис. 67).

Выделение: Константа	
Направление	Юг
Биты данных	4
Значение	0x0

Рис. 67. Состояние панели после редактирования.

После занесения в текстовое поле нужного значения необходимо нажать Enter, чтобы изменения вступили в силу.

Когда отредактированы все разряды личного кода, нужно его применить. Для этого просто нажимаем на соответствующую кнопку (рис. 64, маркер 1).

В результате система переключится на вариант, указанный личным кодом (в нашем случае получится вариант «Запасной 128»).

После изменения констант на личный код имеет смысл сохранить изменения, сделанные в файле start.circ.

Таким образом Вы произвели настройку системы на личный вариант и теперь можете все время работать именно с этой копией (для чего необходимо сохранить ее у себя в электронной почте).

Теперь всякий раз при сбросе процесса моделирования в исходное

состояние достаточно нажать кнопку «применить личный код», чтобы быстро вернуться к своему варианту.

3. Запишите на черновик исходные данные, соответствующие решаемому заданию:

– для заданий, связанных с реализацией логической функции от 4-х переменных разными способами, источником исходных данных является **индикатор F1**;

– при разработке преобразователя кода источником исходных данных являются все четыре **индикатора F1-F4**;

– для заданий по разработке счетчиков источником исходных данных является **индикатор CNT**.

4. Спроектировать цифровую схему, соответствующую выполняемому заданию, и реализовать ее на свободном пространстве (рис.63, маркер 5).

5. Подключить входы схемы к необходимым источникам сигналов в зависимости от задания (это либо логические переменные A,B,C,D, либо синхросигнал — рис.63, маркеры 3 и 4).

6. Подключить выходы разработанной схемы к одному из входных портов УУАП (в зависимости от задания, см. рис. 64, маркер 4).

7. Указать частоту работы УУАП, для чего зайти в меню Logisim «Моделировать» -> «Тактовая частота» и установить, например, 16 герц (это достаточно быстро, чтобы не ждать долго, но еще видно, как распространяются сигналы по проводам вашей схемы).

8. Запустить процесс автоматической проверки, нажав **комбинацию Ctrl+K**. Таким образом Вы включите тактовый генератор, который находится внутри УУАП, вследствие чего приведете в действие все схемы внутри УУАП, а также схему, разработанную вами. УУАП будет генерировать необходимые входные сигналы для Вашей схемы и оценивать результаты ее работы. При этом провода Вашей схемы будут мигать оттенками зеленого в соответствии с движением нулей и единиц (ярко-зелёный соответствует единице, темно-зеленый — нулю).

9. Наблюдайте за работой схемы, ожидая окончания процесса проверки. Признаком окончания проверки является прекращение мигания проводов и загорание индикатора правильности работы схемы либо индикаторов ошибки.

10. Остановите работу тактового генератора УУАП, **еще раз нажав Ctrl+K**.

11. Ознакомьтесь с полученными результатами проверки. **Обратите внимание:** правильность работы схемы **еще не означает**, что практическая работа

выполнена правильно. **Возможно, что Вы реализовали схему не тем способом, каким требовалось в задании.** Если схема работает правильно, то продемонстрируйте ее работу преподавателю (для чего ее потребуется перезапустить в присутствии преподавателя), чтобы получить разрешение на оформление отчета.

Методика исправления ошибок в обобщенном виде описана в приложении 3. Если приведенных там рекомендаций оказалось недостаточно, то обратитесь к преподавателю.

12. Исправьте ошибки, если они есть, внося необходимые изменения в Вашу схему.

13. Перезапустите работу схемы, для чего нажмите комбинацию **Ctrl+R** (сброс предыдущих результатов работы схемы), **примените личный код** нажатием соответствующей кнопки, **и переходите к пункту 8** данного алгоритма. При сбросе установленная ранее тактовая частота не изменяется.

ПРИЛОЖЕНИЕ 3.

РЕКОМЕНДАЦИИ ПО ОТЛАДКЕ СХЕМ В СРЕДЕ LOGISIM

При выполнении практических работ по логическим основам вычислительной техники полученные студентом схемы поначалу могут либо вовсе не работать, либо работать неправильно.

Причиной этого могут являться ошибки проектирования и ошибки сборки схемы.

Признаком неправильной работы комбинационных схем является включение индикаторов ошибки. Индикаторы показывают, какие входные наборы обрабатываются схемой неправильно.

Признаком неправильной работы счетчика служит отсутствие зеленого светового сигнала, названного «Признак правильности работы схемы».

Начинать отладку схемы рекомендуется с поиска ошибок сборки. Это подразумевает выполнение следующих действий.

- Проследить взглядом за соединением проводов. Возможно, где-то отсутствует контакт. Иногда для того, чтобы это заметить, полезно осторожно пошевелить отдельные элементы схемы.

- Убедиться в отсутствии короткого замыкания. Если оно есть, то в процессе работы схемы провода, ведущие к месту короткого замыкания, будут окрашиваться в красный цвет.

- Проверить соответствие реализованной схемы ее формуле. Иначе говоря, подключение элементов или сами выбранные элементы могут не соответствовать логической формуле или непосредственно таблице истинности (например, для мультиплексора 16-1, для преобразователя кодов).

Если поиски ошибок сборки не дали результатов, то мы имеем дело с ошибкой проектирования.

Ошибка проектирования при моделировании схем СДНФ и СКНФ может заключаться в:

- неправильной записи формул СДНФ и СКНФ по таблице истинности;
- неправильном построении самой таблицы истинности по персональным исходным данным.

Ошибки проектирования при моделировании схем, реализующих минимизированные функции в заданном логическом базисе, могут проявляться на разных этапах. Рекомендуется соблюдать следующий порядок поиска.

- Проверить правильность перевода минимальной нормальной формы в

заданный логический базис.

- Проверить правильность записи минимальной формы по выделенным на карте Карно интервалам.
- Проверить правильность выделения интервалов на карте Карно.
- Проверить правильность разметки карты Карно.
- Проверить правильность заполнения карты Карно по исходной таблице истинности (или таблице переходов для счетчика).
- Проверить правильность построения таблицы истинности по исходным данным.
- Убедиться в правильности понимания условия задачи, а также в том, что исходные данные соответствуют персональному варианту.

В рамках данных практических работ наиболее сложная схема с точки зрения отладки это схема счетчика с минимизированной логикой управления. Начинать отладку счетчика необходимо с поиска ошибок сборки, как было описано выше. Если ошибки сборки не найдены, то необходимо искать ошибку проектирования. Она будет связана с одной или несколькими комбинационными схемами, управляющими триггерами. Чтобы определить эти схемы, сначала нужно найти ошибочное состояние счетчика, которое складывается из состояний отдельных триггеров.

Присмотритесь к отдельным триггерам. В их центре можно увидеть цифры 0 или 1 в зависимости от их текущего состояния. Тогда, если предположить, что старшинство триггеров на схеме возрастает слева направо, мы сможем прочесть двоичное число, соответствующее текущему состоянию счетчика. Для этого нужно рассматривать триггеры справа налево. Теперь мы сможем найти неверно работающие схемы управления триггеров, если запустим работу счетчика в пошаговом режиме. Для этого нужно выключить встроенный тактовый генератор, если он включен (присмотритесь к миганию сигнала синхронизации), нажав CTRL-K, перевести схему в исходное состояние (CTRL-R), и подавать синхроимпульсы вручную, по одному, нажимая CTRL-T.

При поступлении очередного синхроимпульса, счетчик переключится в очередное состояние. Это состояние необходимо сверить с таблицей переходов счетчика, которая была восстановлена по исходным данным. Продолжаем переходить в очередное состояние счетчика до обнаружения первого рассогласования с таблицей переходов. Допустим, счетчик перешел в состояние 1001, а должен был перейти в состояние 1011. Это значит, что неправильно работает схема управления триггера Q2. Может быть обнаружено и больше различий,

следовательно, неправильно работают уже несколько управляющих схем.

После того, как были найдены неправильно работающие схемы управления, для каждой из них необходимо выполнить последовательность проверочных действий как было описано для минимизированных функций.

Если перечисленные рекомендации не помогли Вам найти ошибку, то обратитесь за помощью к преподавателю.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Программа-тренажер «Логика». Разработчик К.Ю. Поляков.
<http://kpolyakov.spb.ru/prog/logic.htm>.
2. Программа построения и моделирования логических схем Logisim.
<http://www.cburch.com/logisim/>
3. Справочная система программы Logisim. Устанавливается вместе с программой [2]. Также доступно: <http://www.cburch.com/logisim/ru/docs.html>.
4. Описание библиотеки элементов Logisim. Устанавливается вместе с программой [2]. Также доступно: <http://www.cburch.com/logisim/ru/docs.html>.
5. Учебная среда «Исполнители». Разработчик К.Ю. Поляков.
<http://kpolyakov.spb.ru/school/robots/robots.htm>

Учебное издание

ИНФОРМАТИКА

Методические указания по выполнению практических работ

Печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Физ. печ. л. _____. Тираж _____ экз. Изд. № _____. Заказ № _____.

МИРЭА — Российский технологический университет (МИРЭА)
119454, Москва, пр. Вернадского, д. 78