МИНИСТЕРСТВО ОБРАЗОВАНИЕ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

«МИСиС»

НОВОТРОИЦКИЙ ФИЛИАЛ

Кафедра Электроэнергетики электротехники

С.Н. Басков, К.В. Лицин

Алгебра логики и основы дискретной техники

Лабораторный практикум

Новотроицк, 2016 г

УДК 621.31

ББК 31.26

Рецензенты:

Зам. директора по учебной работе Института «Энергетики и автоматизированных систем», доцент кафедры «Электроники и микроэлектроники» ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.Н. Носова»,
к.т.н., доцент Д.Ю. Усатый

Доцент кафедры «Электроэнергетики и электротехники» ФГАОУ ВО «Национальный исследовательский технологический университет «МИСиС»,
к.т.н., доцент М.Н. Давыдкин

Басков С.Н., Лицин К.В. Алгебра логики и основы дискретной техники: лабораторный практикум. – Новотроицк: НФ НИТУ «МИСиС», 2016.- 61с.

Учебное пособие предназначено для студентов направления 13.03.02 «Электроэнергетика и электротехника», изучающих дисциплины: «Алгебра логики и основы дискретной техники», «Дискретная математика», «Цифровая и аналоговая электроника». В данной части практикума рассмотрено применение программы MicroCap для исследования микросхем и выражений, реализованных с помощью логических элементов. Приведены виртуальные лабораторные работы по темам: «Основные логические элементы», «Исследование дешифраторов», «Исследование двоичных сумматоров, «Цифровые компараторы», «Мультиплексоры и демультиплексоры», «Синтез и исследование триггеров».

Рекомендовано Методическим советом НФ НИТУ «МИСиС»

Содержание

ВВЕДЕНИЕ

Алгебра логики – наука, изучающая процессы, протекающие с использованием цифровых сигналов.

Цель настоящего лабораторного практикума - углубление и закрепление теоретических знаний по проектированию и применению наиболее распространенных цифровых элементов, узлов и устройств, а также приобретение навыков работы с цифровыми интегральными схемами и устройствами, построенными на их основе.

Все лабораторные работы выполняются с использованием программного обеспечения MicroCap.

Вариант лабораторных работ выбирается в соответствии с номером студента в списке группы.

Защита лабораторных работ проводится устно, для этого студент должен иметь отчет о проделанной работе. Отчет должен содержать:

- название работы;
- цель работы;
- теоретические сведения;
- таблицы истинности;
- структурные схемы исследования;
- временные диаграммы результатов исследования;
- выводы по работе;
- ответы на контрольные вопросы.

ЛАБОРАТОРНАЯ РАБОТА №1
ОСНОВНЫЕ ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

Цель работы:Исследование функционирования основных логических элементов и построение на их основе логических схем.

1 Теоретическое введение

Логической переменной называется величина, которая может принимать одно из двух возможных состояний (значений), одно из которых обозначается символом «0», другое – «1» (для обозначения состояний возможно применение и других символов, например, «Да» и «Нет» и др.). Сами двоичные переменные чаще обозначают символами x_1, x_2,… В силу определения логические переменные можно называть также двоичными переменными.

Логической (булевой) функцией называется функция двоичных переменных (аргументов), которая также может принимать одно из двух возможных состояний (значений): «0» или «1». Значение некоторой логической функции n переменных определяется или задается для каждого набора (сочетания) двоичных переменных. Количество возможных различных наборов, которые могут быть составлены из n аргументов, очевидно, равно 2^n. Логическая функция обычно обозначается y. Логические переменные и функции взаимосвязаны между собой с помощью таблицы истинности.

Рассмотрим основные логические функции.

Функция «отрицание» – это функция одного аргумента (другие названия функции: инверсия, логическая связь НЕ, inverter). Аналитическая форма задания этой функции: $y = \bar{x}$.

Таблица истинности представлена в таблице 1.1.

Таблица 1.1. – Таблица истинности логического элемента «НЕ»

x_1	y
0	1
1	0

Обозначение и временная диаграмма представлены на рисунке 1.1.

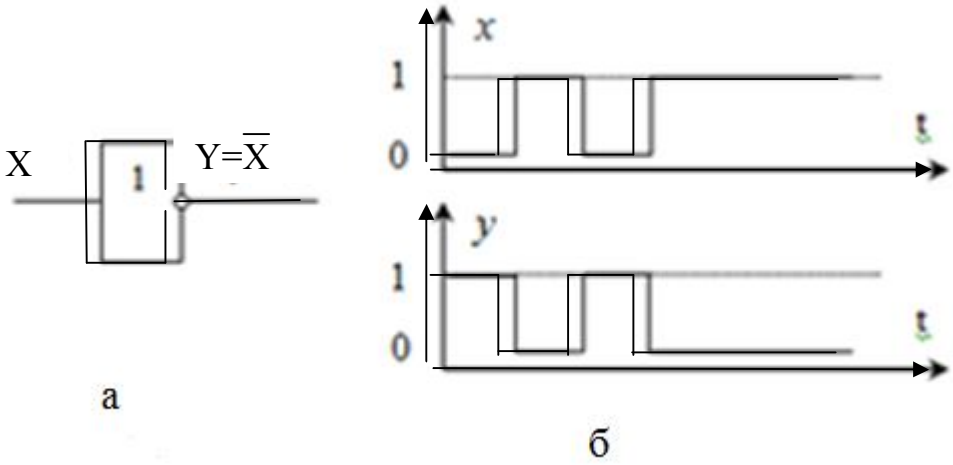


Рисунок 1.1 - Обозначение (а) и временная диаграмма (б) элемента «НЕ»

Функция «Конъюнкция» представляет собой логическое умножение, как минимум двух переменных x_1 и x_2 равна «1» в том случае, когда и x_1 и x_2 равны «1».

Таблица истинности представлена в таблице 1.2.

Таблица 1.2. – Таблица истинности логического элемента «И»

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

Конъюнкция фактически представляет собой обычное умножение (нулей и единиц). Иногда эту функцию обозначают $x_1 \& x_2$ или $x_1 \wedge x_2$.Иначе эту функцию ещё называют логическая «И», «AND».

Обозначение и временная диаграмма представлены на рисунке 1.2.

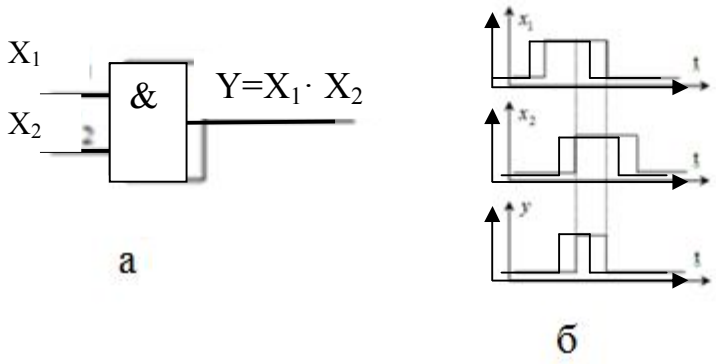


Рисунок 1.2 - Обозначение (а) и временная диаграмма (б) элемента «И»

Дизъюнкция – это логическое сложение, как минимум, двух переменных x_1 и x_2. Принимает значение «1», если или x_1, или x_2 равна «1» (отсюда название операции логическое ИЛИ). В тех случаях, когда число переменных больше двух, их конъюнкция равна «1» при равенстве «1» всех переменных; дизъюнкция равняется «1», если хотя бы одна из переменных имеет значение «1».

Таблица истинности представлена в таблице 1.3.

Таблица 1.3. – Таблица истинности логического элемента «ИЛИ»

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

Обозначение и временная диаграмма элемента «ИЛИ» представлены на рисунке 1.3.

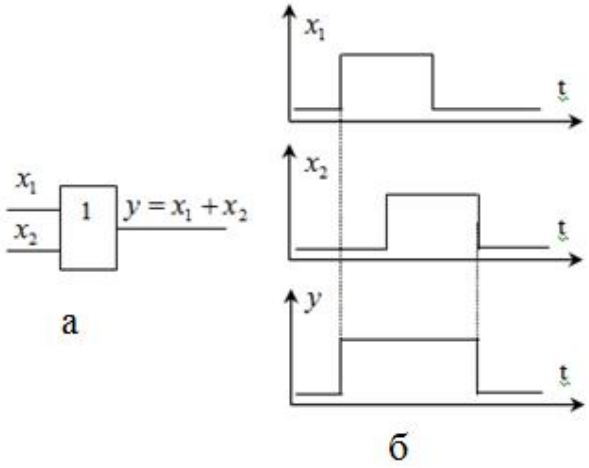


Рисунок 1.3 - Обозначение (а) и временная диаграмма (б) элемента «ИЛИ»

Сложение по модулю 2 (логическая неравнозначность, исключающее «ИЛИ», строгая дизъюнкция, XOR, поразрядное дополнение, побитовый комплемент, жегалкинское сложение) - булева функция, а также логическая и битовая операция. В случае двух переменных результат выполнения операции является истинным тогда и только тогда, когда лишь один из аргументов является истинным. Для функции трёх и более переменных, результат выполнения операции будет истинным только тогда, когда количество аргументов равных 1, составляющих текущий набор - нечетное. Такая операция естественным образом возникает в кольце вычетов по модулю 2, откуда и происходит название операции.

Таблица истинности представлена в таблице 1.4.

Таблица 1.4. – Таблица истинности логического элемента «исключающее ИЛИ»

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

Обозначение элемента исключающее «ИЛИ» представлено на рисунке 1.4.

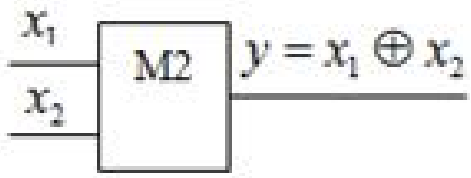


Рисунок 1.4 - Обозначение элемента исключающее «ИЛИ»

При описании функций алгебры логики алгебраическим выражением используются две стандартные формы ее представления – так называемые дизъюнктивная и конъюнктивная нормальные формы.

Дизъюнктивной нормальной формой (ДНФ) называется логическая сумма элементарных логических произведений, в каждое из которых аргумент или его инверсия входит один раз.

Конъюнктивной нормальной формой (КНФ) называется логическое произведение элементарных логических сумм, в каждую из которых аргумент или его инверсия входят один раз.

Рассмотрим на примере данных, представленных в таблице 1.5, составление ДНФ и КНФ, а так же построения логической схемы на основе ДНФ.

Таблица 1.5 – Исходные данные для получения ДНФ и КНФ

x_0	x_1	x_2	y
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

ДНФ представляет собой вид:

$$y = x_0 \cdot x_1 \cdot \overline{x}_2 + x_0 \cdot \overline{x}_1 \cdot x_2 + \overline{x}_0 \cdot x_1 \cdot x_2 + x_0 \cdot x_1 \cdot x_2 ,$$

КНФ описана в виде:

$$y = (x_0 + x_1 + x_2) \cdot (\overline{x}_0 + x_1 + x_2) \cdot (x_0 + \overline{x}_1 + x_2) \cdot (x_0 + x_1 + \overline{x}_2)$$

На основе ДНФ в программе MicroCap построена логическая схема, представленная на рисунке 1.5.

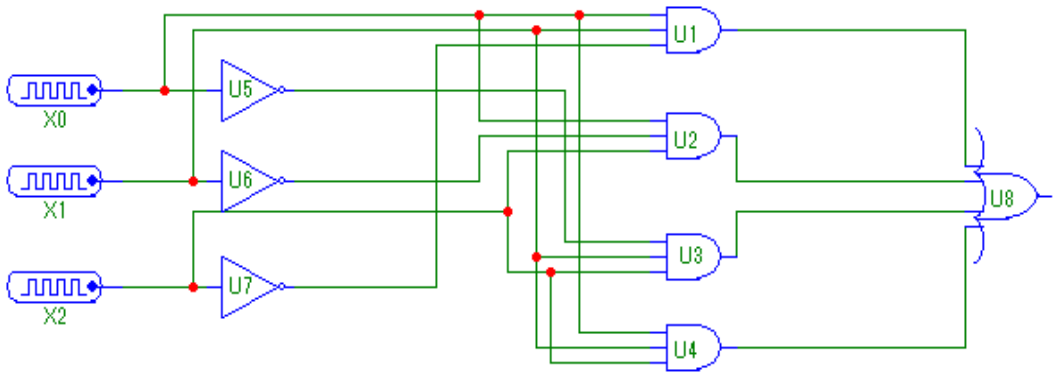


Рисунок 1.5 - Логическая схема в программе MicroCap

Отметим, что при составлении ДНФ выбираются те строчки, где выходная величина (в нашем случае «y») принимает значения равные «1». Инверсия ставится над теми входными сигналами, которые равны «0». При составлении выражения для КНФ выбираются те строчки, где выходная величина «y» равна «0». Соответственно, инверсия ставится над теми входными сигналами, которые равны «1».

2 Выполнение работы

2.1 С помощью программы MicroCap исследовать логические элементы, согласно своему варианту. Вид логических элементов (ЛЭ) представлен в

таблице 2.1. Цифра перед названием ЛЭ означает число входов.

Таблица 2.1 – Таблица исследуемых логических элементов

№ варианта	Исследуемые логические элементы								
	НЕ	2И	2И-НЕ	2ИЛИ	3И	3И-НЕ	3ИЛИ	4И	4ИЛИ
1	+			+	+	+		+	+
2	+	+	+				+	+	+
3	+	+		+		+		+	+
4	+		+		+		+	+	+
5	+	+		+		+	+	+	
6	+			+	+	+		+	+
7	+	+	+				+	+	+
8	+	+		+		+		+	+
9	+		+		+	+	+		+
10	+	+		+		+		+	+
11	+	+	+	+		+		+	
12	+		+		+		+	+	+
13	+	+		+		+	+	+	
14	+			+	+	+		+	+
15	+	+	+				+	+	+
16	+	+			+	+		+	+
17	+		+		+	+	+		+
18	+	+	+			+		+	+
19	+	+	+	+		+		+	
20	+			+	+	+		+	+

2.2 Схема исследования логического элемента содержит генератор сигнала «DClock», который находится по пути: Digital Primitives–Stimulus Generators–DClock.

После того, как элемент «DClock» расположен в рабочей области необходимо поставить соответствующие числа в полях ZEROWIDTH («длительность нуля») и ONEWIDTH («длительность единицы»). Расположение данных полей представлено на рисунке 2.1. Для первого задающего сигнала должны стоять значения 50N, для второго 100N, для третьего 200N, т.е. каждый следующий больше предыдущего в 2 раза.

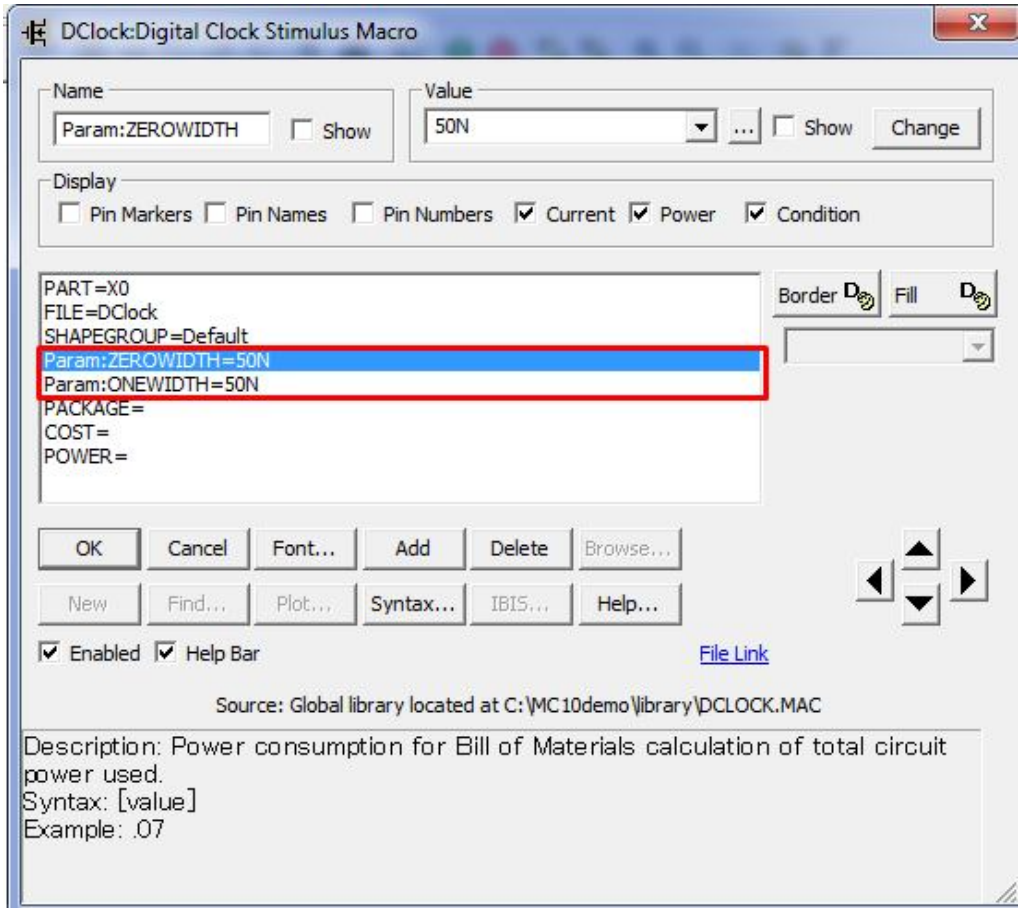


Рисунок 2.1 - Задание длительности «1» и «0» входного воздействия

2.3 Все основные логические элементы находятся: Digital Primitives–Standard Gates. При расположении каждого из них в рабочем поле выбрать «D0_GATE».

Пример схемы исследования логического элемента «НЕ» представлен на рисунке 2.2.

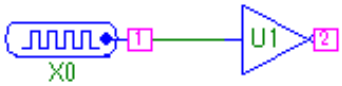
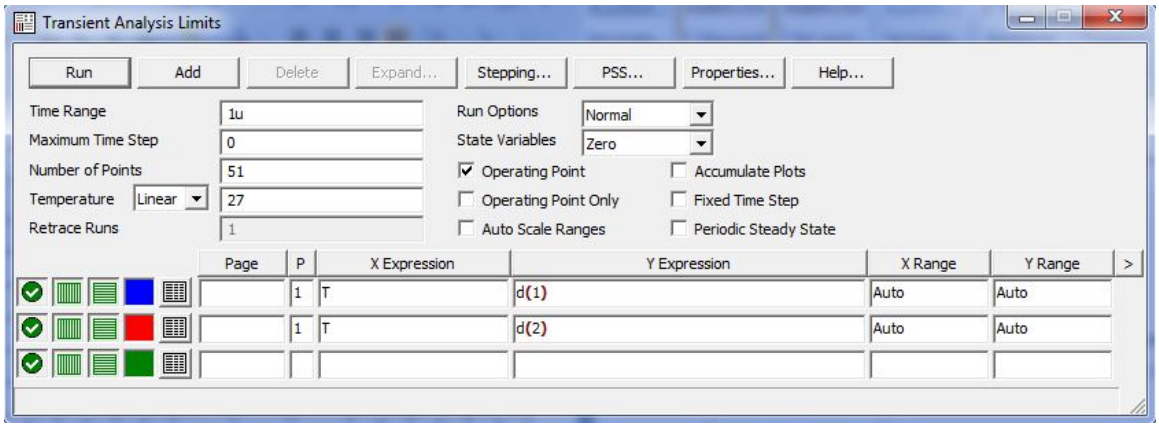
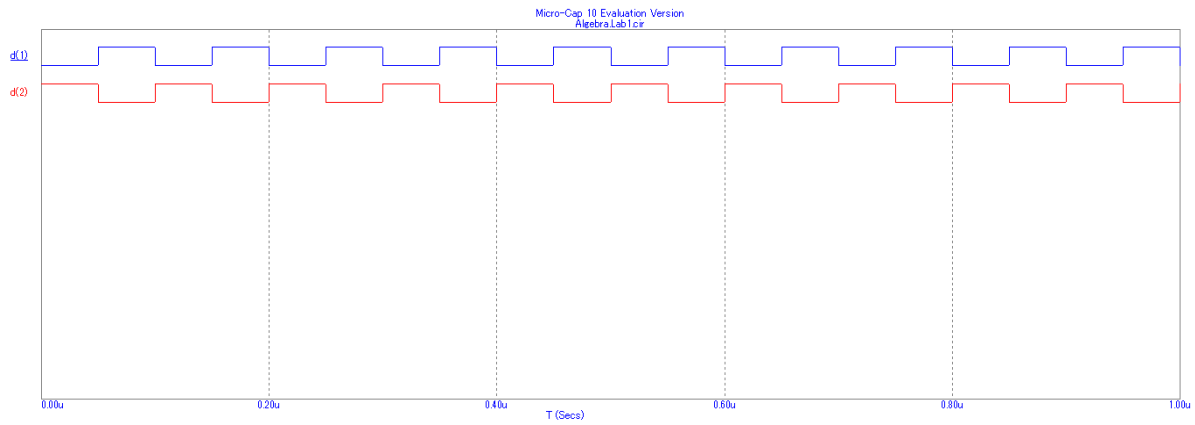


Рисунок 2.2 - Схема исследования логического элемента «НЕ»

2.4 Получить результат исследования в виде графиков, нажав «Analysis» - «Transient». В полученном окне (Рисунок 2.3, а) настроить необходимое количество входов и выходов. В случае логического элемента «НЕ» вход обозначен цифрой 1, выход 2 (см. Рисунок 2.2). Нажать на кнопку Run и получить результат (Рисунок 2.3, б)



а)



б)

Рисунок 2.3 - Окно настройки и временная диаграмма логического элемента «НЕ» в программе MicroCap

2.4 Собранную схему и временную диаграмму сохранить себе в отчёт. Аналогичные действия проделать с остальными логическими элементами, отмеченными знаками «+» в своём варианте.

2.5 Реализовать логические функции, соответствующие вашему варианту по таблице 2.2. Сохранить схему и временную диаграмму себе в отчет. На основе последней построить таблицу истинности.

Таблица 2.2 –Реализация логической схемы на основе функции

№ варианта	Функция, подлежащая реализации
1	а) $y=\overline{x_1\cdot x_2\cdot x_3}+(x_1+x_2)$ б) $y=\bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3+\bar{x}_1\cdot x_2+\bar{x}_2\cdot x_3$ в) $y=\overline{(x_1+x_2+\bar{x}_3)}\cdot(\bar{x}_1+\bar{x}_2+x_3)$
2	а) $y=\overline{x_1\cdot x_2}+(x_1+x_2+x_3)$ б) $y=\bar{x}_1\cdot x_2\cdot\bar{x}_3+x_1\cdot\bar{x}_2+x_2\cdot x_3$ в) $y=\overline{(x_1+x_2+\bar{x}_3)}\cdot(\bar{x}_1+x_2\cdot\bar{x}_3)$
3	а) $y=\overline{x_1\cdot x_2\cdot x_3}+\overline{(x_1+x_2)}$ б) $y=\bar{x}_1\cdot x_2\cdot x_3+x_1\cdot\bar{x}_2+\overline{x_2\cdot x_3}$ в) $y=\overline{(x_1+\bar{x}_2+\bar{x}_3)}\cdot(x_1+\bar{x}_2\cdot x_3)$
4	а) $y=x_1\cdot x_2\cdot x_3+\overline{(x_1+x_2+x_3)}$ б) $y=\bar{x}_1\cdot x_2\cdot x_3+\overline{x_1\cdot\bar{x}_2}+x_2\cdot x_3$ в) $y=\overline{(\bar{x}_1+x_2+\bar{x}_3)}+(x_1+\bar{x}_2\cdot x_3)$
5	а) $y=(x_1+x_2+x_3)+\overline{x_1\cdot x_2\cdot x_3}$ б) $y=\bar{x}_1\cdot\bar{x}_2\cdot x_3+\overline{x_1\cdot\bar{x}_2}+\bar{x}_2\cdot x_3$ в) $y=\overline{(\bar{x}_1+x_2+\bar{x}_3)}\cdot\overline{(x_1+\bar{x}_2\cdot x_3)}$

Продолжение таблицы 2.2

6	а) $y = \overline{x_1 \cdot x_2 \cdot x_3} + (x_1 + x_2)$ б) $y = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 + \bar{x}_2 \cdot x_3$ в) $y = \overline{(x_1 + x_2 + \bar{x}_3)} \cdot (\bar{x}_1 + \bar{x}_2 + x_3)$
7	а) $y = \overline{x_1 \cdot x_2} + (x_1 + x_2 + x_3)$ б) $y = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 + x_2 \cdot x_3$ в) $y = \overline{(x_1 + x_2 + \bar{x}_3)} \cdot (\bar{x}_1 + x_2 \cdot \bar{x}_3)$
8	а) $y = \overline{x_1 \cdot x_2 \cdot x_3} + \overline{(x_1 + x_2)}$ б) $y = \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_2 + \overline{x_2 \cdot x_3}$ в) $y = \overline{(x_1 + \bar{x}_2 + \bar{x}_3)} \cdot (x_1 + \bar{x}_2 \cdot x_3)$
9	а) $y = x_1 \cdot x_2 \cdot x_3 + \overline{(x_1 + x_2 + x_3)}$ б) $y = \bar{x}_1 \cdot x_2 \cdot x_3 + \overline{x_1 \cdot \bar{x}_2} + x_2 \cdot x_3$ в) $y = \overline{(\bar{x}_1 + x_2 + \bar{x}_3)} + \overline{(x_1 + \bar{x}_2 \cdot x_3)}$
10	а) $y = (x_1 + x_2 + x_3) + \overline{x_1 \cdot x_2 \cdot x_3}$ б) $y = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + \overline{x_1 \cdot \bar{x}_2} + \bar{x}_2 \cdot x_3$ в) $y = \overline{(\bar{x}_1 + x_2 + \bar{x}_3)} \cdot \overline{(x_1 + \bar{x}_2 \cdot x_3)}$
11	а) $y = \overline{x_1 \cdot x_2 \cdot x_3} + (x_1 + x_2)$ б) $y = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 + x_2 \cdot x_3$ в) $y = \overline{(x_1 + x_2 + \bar{x}_3)} \cdot (\bar{x}_1 + \bar{x}_2 + x_3)$
12	а) $y = (\bar{x}_1 + x_2 + \bar{x}_3) + x_2 \cdot x_3 \cdot \overline{x_1 \cdot x_2}$ б) $y = \bar{x}_1 \cdot x_2 + (\bar{x}_3 + x_1) \cdot \bar{x}_2 + x_2 \cdot x_3$ в) $y = \overline{x_1 \cdot \bar{x}_3} + (\bar{x}_1 + x_2 \cdot \bar{x}_3)$

Продолжение таблицы 2.2

13	а) $y=\overline{x_1\cdot x_2\cdot x_3}+\overline{(x_1+x_2)}$ б) $y=\bar{x}_1+x_2\cdot x_3+x_1\cdot\bar{x}_2+\overline{x_2+x_3}$ в) $y=(\overline{x_1+\bar{x}_2\cdot\bar{x}_3})\cdot(x_1+\bar{x}_2\cdot x_3)+\bar{x}_2\cdot x_1$
14	а) $y=x_1\cdot x_2\cdot x_3+\overline{(x_1+x_2+x_3)}$ б) $y=\bar{x}_1\cdot x_2\cdot x_3+\overline{\bar{x}_1\cdot\bar{x}_2}+x_2+x_3$ в) $y=(\overline{\bar{x}_1+x_2+\bar{x}_3})\cdot(\overline{x_1+\bar{x}_2\cdot x_3})$
15	а) $y=\overline{x_1\cdot x_2\cdot x_3}+\overline{(x_1+x_2)}$ б) $y=\bar{x}_1\cdot x_2\cdot\bar{x}_3+x_1\cdot\bar{x}_2+\overline{x_2\cdot x_3}$ в) $y=(\overline{x_1+\bar{x}_2+\bar{x}_3})\cdot(x_1+\bar{x}_2\cdot x_3)$
16	а) $y=x_1\cdot x_2\cdot x_3+\overline{(x_1+x_2+x_3)}$ б) $y=\bar{x}_1\cdot x_2\cdot x_3+\overline{x_1\cdot\bar{x}_2}+x_2\cdot x_3$ в) $y=(\overline{\bar{x}_1+x_2+\bar{x}_3})+(x_1+\bar{x}_2\cdot x_3)$
17	а) $y=\overline{x_1\cdot x_2}+(x_1+x_2+x_3)$ б) $y=\bar{x}_1\cdot x_2\cdot\bar{x}_3+x_1\cdot\bar{x}_2+x_2\cdot x_3$ в) $y=(\overline{\bar{x}_1\cdot\bar{x}_2+x_3})\cdot(\bar{x}_1+x_2+\bar{x}_3)$
18	а) $y=x_1\cdot\bar{x}_2\cdot x_3+\overline{x_1\cdot x_2}\cdot x_2+x_3$ б) $y=\bar{x}_1+(x_2\cdot\bar{x}_3)+x_1+\bar{x}_2\cdot\overline{x_2+x_3}$ в) $y=(\overline{x_1+x_2+\bar{x}_3})\cdot(\bar{x}_1+x_2\cdot\bar{x}_3)$
19	а) $y=\overline{x_1\cdot x_2\cdot x_3}+\overline{(x_1+x_2)}$ б) $y=\bar{x}_1\cdot x_2\cdot x_3+x_1\cdot\bar{x}_2+\overline{x_2\cdot x_3}$ в) $y=(\overline{x_1+\bar{x}_2+\bar{x}_3})\cdot(x_1+\bar{x}_2\cdot x_3)$

Окончание таблицы 2.2

20	а) $y = \overline{x_1 \cdot x_2 \cdot x_3} + \overline{(x_1 + x_2)}$ б) $y = \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_2 + \overline{x_2 \cdot x_3}$ в) $y = \overline{(x_1 + x_2 + \bar{x}_3)} \cdot (\bar{x}_1 + \bar{x}_2 + x_3)$

2.6 Сделать выводы по работе.

3 Контрольные вопросы

3.1 Объясните, как на логической схеме можно проверить исправность соединительных проводников (отсутствие обрывов)?

3.2 Что такое таблица истинности ЛЭ или устройства, осуществляющего некоторое логическое преобразование?

3.3 Укажите размерность таблицы истинности (число строк и число столбцов) следующих логических элементов: 4И и 2 ИЛИ.

3.4 Объясните, почему неиспользуемые входы ЛЭ “ИЛИ”, “ИЛИ-НЕ” соединяют с корпусом (уровнем логического “0”), а на неиспользуемые входы ЛЭ “И”, “И-НЕ” подается напряжение уровня логической “1”?

3.5 Используя ЛЭ наборного поля получите три различных варианта схем, реализующих логическую функцию “5И-НЕ”. Который из них является наиболее оптимальным (рациональным)?

3.6 Какую логическую функцию реализует цепочка из К последовательно соединенных инверторов, если К – нечетное число, К – четное число? Чему эквивалентны такие цепочки?

3.7 Изобразите временные диаграммы, характеризующие функционирование ЛЭ: НЕ, 3И, 3ИЛИ, 3И-НЕ, 3М2.

ЛАБОРАТОРНАЯ РАБОТА №2
ИССЛЕДОВАНИЕ ДЕШИФРАТОРОВ

Цель работы: Изучить принципы проектирования дешифраторов в заданном базисе логических элементов, а также исследовать функционирование спроектированных дешифраторов и интегральных схем дешифраторов.

1 Теоретическое введение

Дешифраторы (декодеры) позволяют преобразовывать одни виды бинарных кодов в другие. Например, преобразовывать позиционный двоичный код в линейный восьмеричный или шестнадцатеричный. Преобразование производится по правилам, описанным в таблицах истинности, поэтому построение дешифраторов не представляет трудностей. Для построения дешифратора можно воспользоваться правилами построения схемы для произвольной таблицы истинности.

Дешифраторы используются в микропроцессорных системах для адресации блоков памяти и периферийных устройств. Кроме того, специализированные дешифраторы применяются для подключения различных индикаторов.

Дешифраторы выпускаются в виде отдельных микросхем или используются в составе других микросхем. В настоящее время десятичные или восьмеричные дешифраторы используются в основном как составная часть таких микросхем, как мультиплексоры, демультиплексоры, постоянные и оперативные запоминающие устройства (ПЗУ и ОЗУ).

Дешифраторы входят в состав практически всех серий цифровых интегральных схем и отличаются:

- числом выходов (полные и неполные дешифраторы);
- видом преобразования - в прямой (прямые выходы) или обратный

(инверсные выходы) унитарный код;

- наличием или отсутствием стробирующего (управляющего) входа. Сигнал на этом входе разрешает или запрещает выполнение микросхемой операции дешифрирования;

- быстродействием, которое характеризуется средним временем задержки распространения сигнала от входа до выхода $t_{зд.р.ср}$;

- энергопотреблением; т.е. мощностью, потребляемой от источника питания.

Схема дешифратора имеет n входов, на которые поступают соответствующие разряды двоичного кода x_n, x_{n-1}, …, x_2, x_1 и m выходов, на которых формируются разряды унитарного кода y_{m-1}, ..., y_1, y_0. При этом дешифратор реализует m функций вида:

$$y_j = f_j(x_n, x_{n-1}, ..., x_2, x_1) = \begin{cases} 1, если \sum_{i=1}^{n} x_i \cdot 2^{i-1} = j \\ 0, если \sum_{i=1}^{n} x_i \cdot 2^{i-1} \neq j \end{cases} \quad (1.1)$$

Функции (1.1) соответствуют преобразованию двоичного кода в прямой унитарный код и могут быть записаны в виде:

$$\begin{aligned} y_0 &= \bar{x}_n \bar{x}_{n-1} ... \bar{x}_2 \bar{x}_1, \\ y_1 &= \bar{x}_n \bar{x}_{n-1} ... \bar{x}_2 x_1, \\ y_2 &= \bar{x}_n \bar{x}_{n-1} ... x_2 \bar{x}_1, \\ y_{m-1} &= x_n x_{n-1} ... x_2 \bar{x}_1, \\ y_m &= x_n x_{n-1} ... x_2 x_1. \end{aligned} \quad (1.2)$$

Изложенное выше соответствует полному дешифратору, т.е. дешифратору, для которого $m=2^n$. На практике часто встречаются неполные дешифраторы, для которых $m<2^n$, следовательно, реализующие лишь некоторые из функций

(1.2). Например, число входов n=3, то число выходов $m=2^3=8$. Таблица истинности такого дешифратора представлена ниже.

Таблица 1.1 – Таблица истинности дешифратора

Входы			**Выходы**							
X1	**X2**	**X3**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
0	0	0	**1**	0	0	0	0	0	0	0
1	0	0	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
1	1	0	0	0	0	**1**	0	0	0	0
0	0	1	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
0	1	1	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**

Схема дешифратора, реализующая эту таблицу истинности, и его условно-графическое обозначение приведены на рисунке 1.1.

Аналогичным образом по предварительно составленной таблице истинности можно получить принципиальную схему и для любого другого дешифратора (декодера).

Наиболее распространены схемы восьмеричных и шестнадцатеричных дешифраторов.

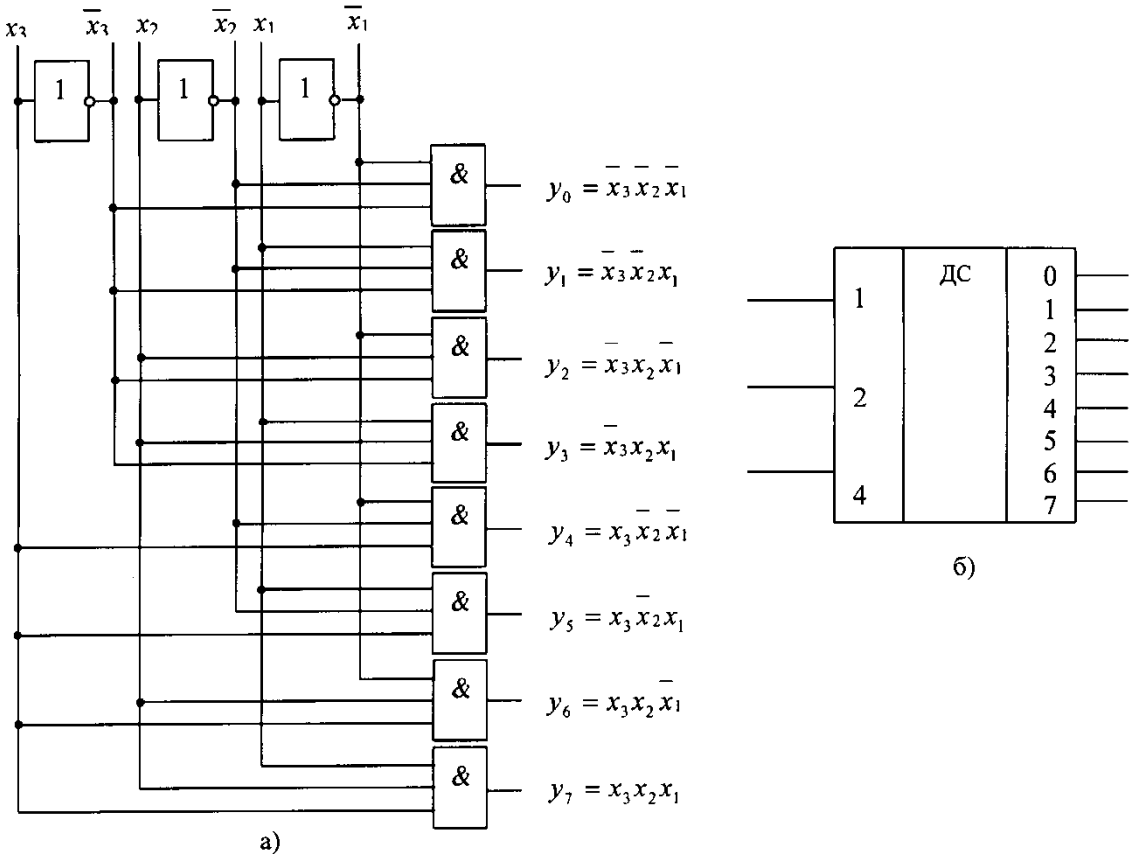


Рисунок 1.1 - Логическая схема (а) и условно-графическое обозначение (б) линейного дешифратора 3 в 8

В то же время для логических элементов, используемых в схемах линейных дешифраторов, характерно значительное число требуемых входов (коэффициент объединения по входу $К_{об}$) логического элемента, равное разрядности дешифрируемого числа - n. В составе интегральных схем (ИС), выпускаемых промышленностью, обычно отсутствуют логические элементы с коэффициентом объединения более восьми и этим значением ограничена разрядность входных чисел линейного дешифратора, если не применяются дополнительные расширители по входу.

При построении схем линейных дешифраторов существенным ограничением, кроме того, является высокая требуемая нагрузочная способность (коэффициент разветвления по выходу $К_{раз.}$) ЛЭ входного регистра, с которых значения разрядов числа подаются на входы дешифратора. Для любого линейного дешифратора требуемая нагрузочная способность ЛЭ

входного регистра равна половине общего числа логических элементов дешифратора: $К_{раз}=0,5\cdot 2^n$. Так как коэффициент разветвления базовых ЛЭ не превышает $К_{раз}=10\div 20$, то для линейных дешифраторов без принятия специальных мер максимальная разрядность дешифруемых чисел n = 4÷5.

Пирамидальный дешифратор.

Усовершенствование структуры дешифраторов позволяет исключить отмеченные ограничения и сводится оно к формированию частичных конъюнкций, используемых в дальнейшем для получения требуемых выходных функций. Пирамидальная структура - один из видов структур дешифратора, реализующих такой принцип построения. Последний основан на том, что добавление одного разряда входной переменной увеличивает число конъюнкций вдвое за счет умножения исходной конъюнкции на дополнительную переменную в прямой и инверсной форме. Поясним сказанное следующим примером. Пусть имеется конъюнкция двух переменных $x_2 \cdot x_1$. При введении добавочного разряда x_3 эта конъюнкция образует две новых: $x_3 x_2 x_1$ и $\bar{x}_3 x_2 x_1$, для получения которых потребуется два двухвходовых ЛЭ «И». Последовательно наращивая структуру, можно построить пирамидальный дешифратор на произвольное число входов.

Пирамидальные дешифраторы отличаются от линейных использованием только двухвходовых конъюнкторов вне зависимости от разрядности дешифрируемого числа, а коэффициент разветвления ЛЭ входного регистра и всех логических элементов дешифратора также равен двум. Таким образом, пирамидальные дешифраторы свободны от ограничений, свойственных линейным дешифраторам, но в них используется большее количество ЛЭ, определяемое как $N=4\cdot(2^{n-1}-1)$. При проектировании цифровых устройств на ИС первостепенную роль играет не количество ЛЭ в устройстве, а количество требуемых корпусов ИС. В то же время количество ЛЭ, располагаемых в одном корпусе ИС, определяется главным образом требуемым количеством выводов. Следовательно, в одном корпусе ИС можно расположить большее число

двухвходовых конъюнкторов, чем трехвходовых, и пирамидальная структура дешифратора, оцениваемая по требуемому числу корпусов ИС, может оказаться эквивалентной или более предпочтительной, чем линейная.

На рисунке 1.2 приведена схема пирамидального дешифратора трехразрядного числа. Пирамидальный дешифратор четырехразрядного числа можно получить добавлением в схему (рисунок 1.2) третьего каскада, содержащего 2^4=16 конъюнкторов и образующего четырехбуквенные конъюнкции.

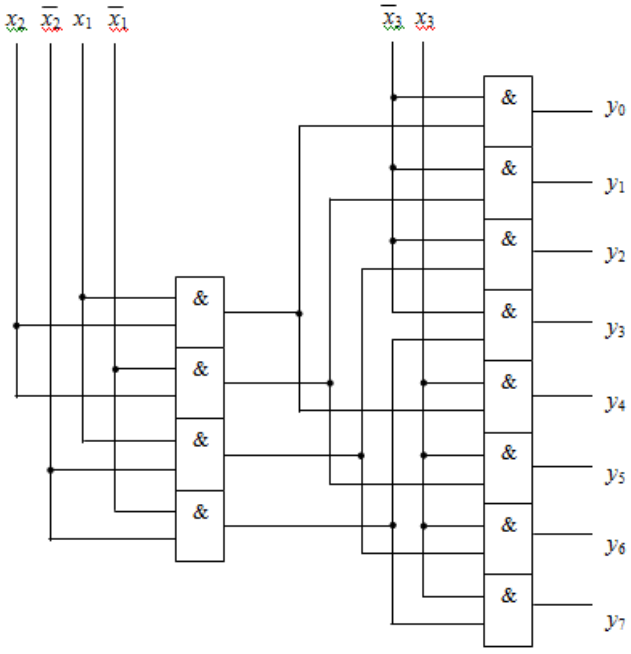


Рисунок 1.2 - Логическая схема пирамидального дешифратора 3 в 8.

Пирамидальные дешифраторы также как прямоугольные, относятся к разряду многоступенчатых дешифраторов, особенностью которых является применение во всех ступенях дешифрации двух входных вентилей с обязательным подключением выходов элемента *i*-ой ступени ко входам только двух элементов (*i*+1)-ой ступени. Число ступеней (*k*) в пирамидальном дешифраторе на единицу меньше разрядности дешифрируемого числа $k = M–1$, число вентилей в каждой из ступеней определяется из выражения:

$$B_i = 2^{i+1} \qquad (1.3)$$

где i –номер ступени пирамидального дешифратора.

Общее количество вентилей на дешифратор определяется из выражения:

$$\sum B = \sum_{i=1}^{k} 2^{i+1},\ i = 1,2,3... \qquad (1.4)$$

Учитывая, что первая ступень дешифратора всегда содержит 2^2вентилей, а в последующих ступенях число вентилей всегда удваивается, можно записать выражение (1.2) как функцию от разрядности числа n:

$$\sum B = 2^2 \cdot (2^{n-1} - 1) \quad (1.5)$$

При реализации пирамидального дешифратора на элементах «И», его быстродействие определяется как:

$$\tau_{Д} = \tau_{CP} \cdot (n-1),$$

где τ_{CP} - время работы одного вентиля.

Недостатком пирамидальных дешифраторов следует считать большое число ступеней (n-1), снижающих быстродействие дешифратора.

2 Выполнение работы

2.1 Изучить теоретический материал про дешифраторы.

2.2 На его основании составить таблицы истинности и логические выражения для схем линейного дешифратора:

– 2 в 4;

– 3 в 8;

– 4 в 16;

и пирамидального дешифратора 3 в 8.

2.3 Реализовать данные логические выражения в программе MicroCap.

3 Контрольные вопросы

3.1. Дайте определение дешифратора.

3.2. Что понимают под унитарным кодом?

3.3. Чем отличается полный дешифратор от неполного?

3.4. Спроектируйте дешифратор «4-16» по

– линейной схеме;

– пирамидальной схеме.

3.5. Какая схемная реализация является более оптимальной с точки зрения:

– аппаратурных затрат;

– быстродействия?

3.6. Оцените потребное количество и типы ЛЭ и ИС, необходимых для построения дешифраторов а)«6-64», б)«8-256» по линейной и пирамидальной схемам.

ЛАБОРАТОРНАЯ РАБОТА №3
ИССЛЕДОВАНИЕ ДВОИЧНЫХ СУММАТОРОВ

Цель работы: Изучить правила выполнения арифметических действий над двоичными числами и исследовать принципы построения двоичных сумматоров.

1 Теоретическое введение

Арифметические действия (операции) относятся к числу наиболее распространенных операций, выполняемых цифровыми устройствами (ЦУ).

Сумматор комбинационного типа (Summator) – это узел цифровой системы, выполняющий арифметическое суммирование кодов слагаемых. В зависимости от системы счисления различают:

–двоичные сумматоры;

–двоично-десятичные сумматоры (в общем случае двоично-кодированные);

–десятичные сумматоры;

–прочие (например, амплитудные).

В компьютере под знак числа отводится бит в старшем разряде слова: если этот разряд установлен в "1", то число отрицательно, "0" –число положительно. Максимальное число со знаком, которое можно представить с помощью n битов равно $2^{n-1}–1$. Отрицательные числа в компьютере представляются двумя типами кода: обратным и дополнительным. Обратным кодом числа называют код, в котором все разряды слова инвертированы. Дополнительным кодом называют код, формируемый из двоичного путем инвертирования всех разрядов слова и сложением инверсного кода с единицей. Указанные коды имеют другое название–коды с неполным и полным дополнением.

Правила сложения: сложение двоичных чисел производится поразрядно от младшего разряда к старшему; в младшем разряде вычисляется сумма младших разрядов слагаемых A и B. Эта сумма может быть записана либо в

виде одноразрядного числа *S*, либо двухразрядного числа *SC*, где *S* – сумма; *C* – перенос; во всех последующих разрядах сумма вычисляется путем сложения разрядов слагаемых *A* и *B* и переноса *C*. Сумма может записана либо в виде одноразрядного числа *S* или двухразрядного числа *SC*.

Пример двоичного сложения:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 0
1 + 1 = 0 перенос 1 в старший разряд

$$\begin{array}{r} 0101 \\ +\ 0110 \\ \hline 1011 \\ \text{перенос} \end{array} \qquad \begin{array}{r} 5 \\ +\ 6 \\ \hline 11 \end{array}$$

Пример двоичного вычитания:

0 - 0 = 0
0 - 1 = 1 заем 1 из старшего раряда
1 - 0 = 1
1 - 1 = 0

$$\begin{array}{r} \text{заем} \\ 0110 \\ -\ 0101 \\ \hline 0001 \end{array} \qquad \begin{array}{r} \\ 6 \\ -\ 5 \\ \hline 1 \end{array}$$

Замена вычитания сложением:

– при записи отрицательных чисел обратным кодом можно инвертировать вычитаемое и прибавить его к уменьшаемому. Если при сложении отрицательных чисел в знаковом разряде возникает перенос, то бит переноса необходимо прибавить к младшему разряду результата сложения;

– при записи отрицательных чисел в дополнительном коде необходимо постоянно прибавлять 1 к младшему разряду суммы.

Простейшим двоичным суммирующим элементом является четверть сумматор. Такое название этот элемент получил из-за того, что он имеет в два раза меньше выходов и в два раза меньше строк в таблице истинности по сравнению с полным двоичным одноразрядным сумматором. Это устройство нам известно как элемент "сложение по модулю 2", "исключающее ИЛИ", "неэквивалентность", "XOR". Схема (Рисунок 1.1) имеет два входа *A*и *B* для двух слагаемых и один выход *S* для суммы.

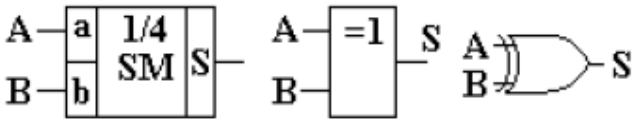


Рисунок 1.1 - Условные графические обозначения четвертьсумматора

Его работу отражает таблица истинности (таблица 1.1).

Таблица 1.1 - Таблица истинности четвертьсумматора

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

Соответствующее уравнение имеет вид:

$$S = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B \qquad (1.1)$$

Четвертьсумматор в базисах И-НЕ, ИЛИ-НЕ и с использованием только одного инвертора может быть представлен выражениями [1]:

$$S = \overline{\overline{\overline{A}\,\overline{AB}} \cdot \overline{B\,\overline{AB}}}$$

$$S = \overline{B + \overline{A + B}} \qquad (1.2)$$

$$S = (A + B)\overline{AB}$$

Полусумматорами называются устройства с двумя входами и двумя выходами, на которых вырабатываются сигналы суммы двух одноразрядных двоичных чисел и переноса.

Для синтеза полусумматора воспользуемся таблицей сложения двоичных чисел, на основании которой построим таблицу истинности (табл.1.2).

Таблица 1.2 – Таблица истинности полусумматора

$\mathbf{A_i}$	$\mathbf{B_i}$	$\mathbf{S_i}$	$\mathbf{C_{i+1}}$
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

На основании таблицы истинности, выписав сумму минтермов, построим переключательные функции в СДНФ для результата сложения S и переноса C_{i+1} и минимизируем их.

$$\begin{aligned} S_i &= \overline{A}_i \cdot B_i + A_i \cdot \overline{B}_i = A_i \oplus B_i \\ C_{i+1} &= A_i \cdot B_i \end{aligned} \qquad (1.3)$$

Полусумматор реализует лишь часть задачи суммирования, так как не учитывает еще одной входной величины – переноса из соседнего младшего разряда данных. По этой причине он осуществляет сложение только в разряде единиц многоразрядного двоичного слова. Одноразрядный двоичный сумматор (полный сумматор), состоит из двух комбинационных схем: одна формирует результат сложения S_i, вторая – бит переноса C_{i+1}. Таблица истинности полного сумматора представлена ниже.

Таблица 1.3 – Таблица истинности полного сумматора

C_{i-1}	A_i	B_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Для полного сумматора минимизированные переключательные функции для S_iи C_{i+1} будут иметь вид:

$$\begin{aligned} S_i &= C_{i-1} \oplus (A_i \oplus B_i) \\ C_{i+1} &= C_{i-1} \cdot (A_i \oplus B_i) + B_i \cdot A_i \end{aligned} \qquad (1.4)$$

Сумматоры с последовательным переносом выполняют сложение кодов поразрядно начиная с младшего бита. Это достигается за счет комбинационного сумматора на три входа. Образующийся в данном разряде перенос C_{j-1} задерживается на время $t_{зд}$ и поступает на вход C_j сумматора в момент поступления следующего разряда слагаемых. Таким образом, последовательно разряд за разрядом производится сложение кодов чисел. Достоинством последовательного сумматора является простота аппаратурной реализации, а недостатком – большое время суммирования.

Сумматор с параллельным (ускоренным) переносомпозволяет достичьболее высокого быстродействия. Параллельная схема каскадирования использует параллельный групповой или ускоренный перенос,причем схема сумматора значительно усложняется по сравнению с сумматором с последовательнымпереносом.

Суммируемые коды поступают на входы сумматора одновременно по всем разрядам. Значениеокончательного переноса формируется специальной схемой, называемой схемой ускоренного переноса.

С целью повышения быстродействия сумматоры в интегральном исполнении изготавливают с малойразрядностью обрабатываемых слов, чаще всего, четырехразрядными.

В программе MicroCap в качестве сумматора с параллельным переносом используется готовая микросхема 7483A (рисунок 1.2).

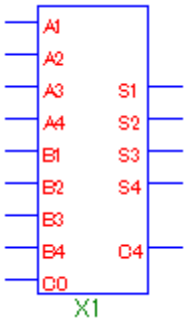


Рисунок 1.2 - Микросхема 7483A, реализующая сумматор с параллельным переносом.

При необходимости величина разрядов числа может быть увеличена. Схема параллельного сумматора для двух восьмиразрядных чисел A и B показана на рисунке 1.3.

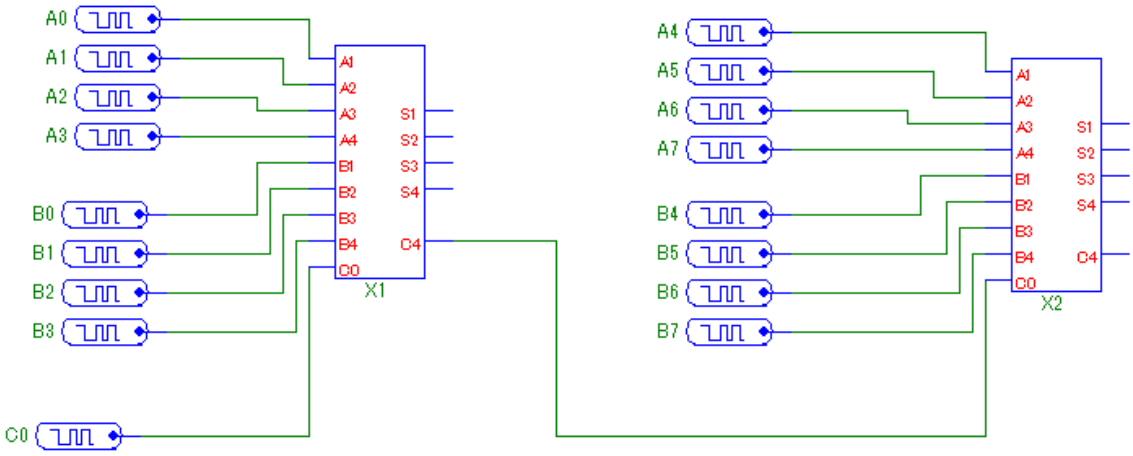


Рисунок 1.3 - Схема параллельного восьмиразрядного сумматора

2 Выполнение работы

2.1 Реализовать четверть сумматор в программе MicroCap на элементе XOR. Привести его таблицу истинности.

2.2 Реализовать четверть сумматор в программе MicroCap в базисах И-НЕ, ИЛЕ-НЕ на основе выражений (1.2). Привести вывод выражений (1.2), основанных на уравнении (1.1).

2.3 Реализовать полусумматор в программе MicroCap на основе выражений (1.3).

2.4 Реализовать схему полного сумматора в программе MicroCap на основе выражений (1.4).

2.5 На основании микросхемы 7483A реализовать сложение двух четырёхразрядных чисел. В отчёт сохранить ScreenShot схемы и временной диаграммы.

2.6 Согласно своего варианта на основании таблицы 2.1 осуществить перевод числа из десятичной системы в двоичную, и выполнить сложение двух чисел на основе схемы параллельного восьмиразрядного сумматора.

Таблица 2.1 – Варианты заданий

Вариант	**Задание**		
1	1) 24+56	2) 156+32	3) 90+121
2	1) 84+12	2) 126+43	3) 78+151
3	1) 67+77	2) 46+22	3) 190+20
4	1) 98+58	2) 100+57	3) 58+121
5	1) 67+12	2) 6+43	3) 22+192
6	1) 111+78	2) 86+77	3) 24+101
7	1) 37+92	2) 26+63	3) 56+102
8	1) 67+12	2) 26+89	3) 22+192
9	1) 132+42	2) 201+3	3) 54+78
10	1) 52+92	2) 221+2	3) 79+98
11	1) 77+88	2) 245+10	3) 98+88
12	1) 45+72	2) 11+34	3) 111+98
13	1) 85+79	2) 41+84	3) 200+12
14	1) 85+55	2) 81+78	3) 110+92
15	1) 105+105	2) 88+113	3) 210+2
16	1) 115+75	2) 48+123	3) 210+22
17	1) 75+128	2) 46+112	3) 200+55
18	1) 89+49	2) 49+84	3) 220+12
19	1) 35+89	2) 31+122	3) 100+102
20	1) 68+29	2) 66+84	3) 20+190
21	1) 102+73	2) 120+54	3) 106+46
22	1) 182+13	2) 1+78	3) 186+46
23	1) 95+72	2) 11+44	3) 191+88
24	1) 105+72	2) 24+74	3) 101+98
25	1) 109+42	2) 24+94	3) 101+98

3. Контрольные вопросы

3.1 Дайте определение одноразрядного сумматора и четвертьсумматора. Приведите для них логические выражения.

3.2 Укажите достоинства и недостатки двоичных сумматоров с последовательным переносом.

3.3 Укажите достоинства и недостатки двоичных сумматоров с параллельным переносом.

3.4 На базе 7483А спроектируйте схему 8-разрядного сумматора - вычитателя.

3.5 Представьте таблицу истинности для 4-разрядного сумматора с параллельным переносом.

ЛАБОРАТОРНАЯ РАБОТА 4

ЦИФРОВЫЕ КОМПАРАТОРЫ

Цель работы: Изучить правила выполнения операции сравнения двоичных чисел и исследовать принципов построения цифровых компараторов.

1 Теоретическое введение

Компаратором (устройством сравнения) называют функциональный узел, обеспечивающий сравнение двух чисел A и B. Если A и B – n-разрядные двоичные числа, то компаратор именуют цифровым.

Простейшие компараторы формируют на выходе однобитовый сигнал равенства, или неравенства сравниваемых чисел A и B. Эти отношения используются как логические условия в микропрограммах, в устройствах контроля и диагностики ЭВМ, в устройствах автоматики компараторы используются для сигнализации о выходе величин за установленные пределы и т.д.

Компараторы строятся на основе поразрядных операций над одноименными разрядами обоих слов. Слова равны, если попарно равны все одноименные их разряды. Признак (условие) равенства i-х разрядов сравниваемых слов A и B:

$$r_{ip} = a_i b_i + \overline{a}_i \overline{b}_i = \overline{a_i \overline{b}_i + \overline{a}_i b_i} = \overline{a_i \oplus b_i} = 1 \qquad (1.1)$$

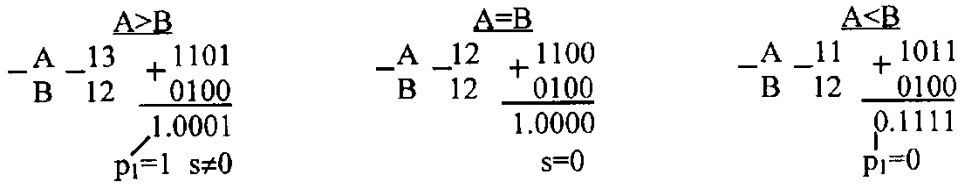
Условие неравенства i-х разрядов:

$$r_{in} = a_i \overline{b}_i + \overline{a}_i b_i = a_i \oplus b_i = 1 \qquad (1.2)$$

Более сложные компараторы выявляют не только факт равенства двух n-разрядных чисел, но и сравнивают числа по значению. Такие компараторы имеют три выхода: “A>B”, “A=B”, “A<B”, и в зависимости от соотношения

величин A и B активный уровень (- уровень логической 1) появляется на одном из этих выходов.

Построить такой компаратор можно на базе двоичного сумматора, выполнив на нем операцию вычитания A-B и проанализировав полученный результат. Для этого на сумматор нужно число B подать в дополнительном коде. Тогда выходной перенос сумматора (p_1) будет равен 0 лишь в том случае, когда A строго меньше B. Равенство разности 0 является признаком того, что A=B. Единица переноса при нулевой сумме указывает на то, что A строго больше B. Сказанное иллюстрируют следующие примеры:



(1.1)

Цифровая схема сравнения это цифровой аналог компаратора, являющегося одним из важнейших устройств импульсной техники. Цифровые схемы сравнения формируют на выходе сигнал, равный единице при равенстве подаваемых на вход двух двоичных чисел. Таблица истинности цифровой схемы сравнения двух двухразрядных чисел A (a и b) и B (c и d) представлена в таблице 1.1.

Таблица 1.1 - Таблица истинности цифровой схемы сравнения двух двухразрядных чисел

A		**B**		**A>B**	**A<B**	**A=B**
a	**b**	**c**	**d**			
0	0	0	0	0	0	1
1	0	0	0	1	0	0
0	1	0	0	1	0	0
1	1	0	0	1	0	0
0	0	1	0	0	1	0
1	0	1	0	0	0	1
0	1	1	0	0	1	0
1	1	1	0	1	0	0
0	0	0	1	0	1	0
1	0	0	1	1	0	0

Продолжение таблицы 1.1

A		B		A>B	A<B	A=B
a	b	c	d			
0	1	0	1	0	0	1
1	1	0	1	1	0	0
0	0	1	1	0	1	0
1	0	1	1	0	1	0
0	1	1	1	0	1	0
1	1	1	1	0	0	1

В программе MicroCap используется компаратор, построенный на базе микросхемы 7485. Он состоит из полного многоразрядного вычитателя и логики сравнения (Рисунок 1.1).

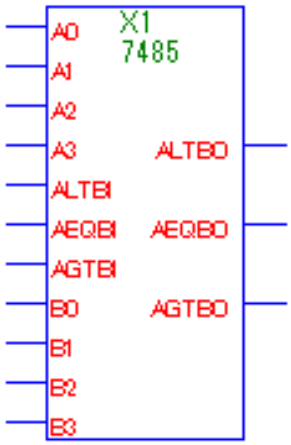


Рисунок 1.1 - Компаратор на базе микросхемы 7485 в программе MicroCap

Компаратор содержит входы:

– A0 – A3 –число A;

– B0 – B3 – числоB;

– ALTBI – команда меньше;

– AEQBI – команда равно;

–AGTBI – команда больше.

Для активации команд необходимо подать на нужный вход ALTBI, AEQBI и (или) AGTBI логическую единицу.

2 Выполнение работы

2.1 На основании таблицы 1.1 составить цифровую схему сравнения. Для этого составить выражения в форме СДНФ, упростить их с помощью карты Карно или на основе законов алгебры логики. По полученным уравнениям составить схему.

2.2 Используя микросхемы 7485 реализовать восьмиразрядный цифровой компаратор, схема которого представлена на рисунке 2.1

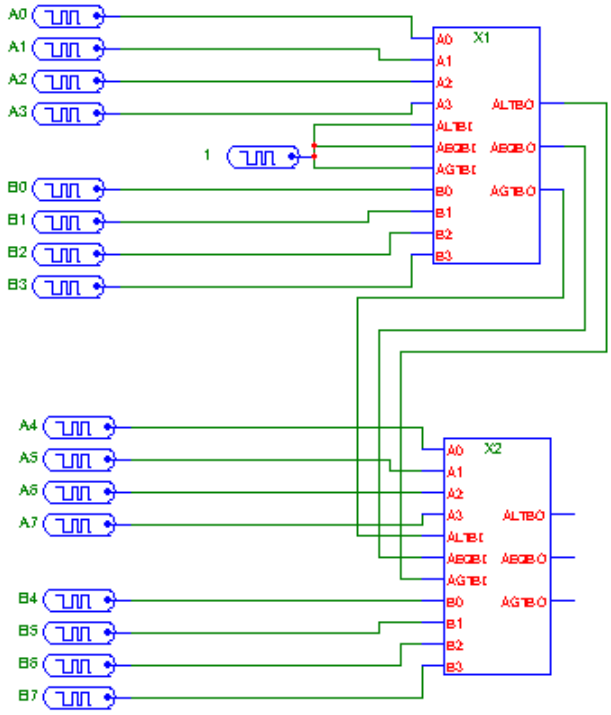


Рисунок 2.1 - Реализация восьмиразрядного компаратора на микросхемах 7485

Проверить правильно работы собранной схемы, выполнив согласно своему варианту сравнение двух чисел. Перед процедурой сравнений необходимо их предварительно перевести в двоичный код.

Таблица 2.1 – Варианты заданий

Вариант	**Задание**		
1	1) 56 и 56	2) 156 и 32	3) 90 и 121
2	1) 84 и 84	2) 126 и 43	3) 78 и 151
3	1) 67 и 77	2) 46 и 46	3) 190 и 20
4	1) 98 и 98	2) 100 и 57	3) 58 и 121
5	1) 67 и 12	2) 43 и 43	3) 22 и 192
6	1) 111и 78	2) 86 и 86	3) 24 и 101
7	1) 37 и 92	2) 63 и 63	3) 56 и 102
8	1) 67 и 12	2) 89 и 89	3) 22 и 192
9	1) 132 и 42	2) 201и 201	3) 54 и 78
10	1) 92 и 92	2) 221 и 2	3) 79 и 98
11	1) 77 и 88	2) 245 и 10	3) 98 и 98
12	1) 45 и 72	2) 11и 11	3) 111 и 98
13	1) 85 и 85	2) 41 и 84	3) 200 и 12
14	1) 85 и 55	2) 81 и 81	3) 110 и 92
15	1) 105 и 67	2) 88 и 113	3) 210 и 210
16	1) 115 и 75	2) 123 и 123	3) 210 и 22
17	1) 75 и 128	2) 112 и112	3) 200 и 55
18	1) 89 и 49	2) 47 и 88	3) 220 и 220
19	1) 35 и 89	2) 122 и 122	3) 140 и 102
20	1) 68 и 29	2) 66 и 66	3) 20 и 190
21	1) 102 и 73	2) 120 и 120	3) 106 и 46
22	1) 182 и 13	2) 78 и 78	3) 186 и 196
23	1) 95 и 95	2) 11 и 44	3) 191 и 88
24	1) 105 и 72	2) 74 и74	3) 101 и 98
25	1) 109 и 42	2) 94 и 94	3) 101 и 98

3 Контрольные вопросы

3.1 Приведите определение цифрового компаратора и перечислите его возможные применения.

3.2 Запишите условия равенства (неравенства) одноименных разрядов сравниваемых чисел A и B.

3.3 Чему равно значение выхода схемы (Рисунок 1.1) при а) A=B, б)A<B и в)A>B?

3.4 Используя 7485 спроектируйте схему 16-ти разрядного цифрового компаратора.

ЛАБОРАТОРНАЯ РАБОТА 5
МУЛЬТИПЛЕКСОРЫ И ДЕМУЛЬТИПЛЕКСОРЫ

Цель работы: Изучить принципы построения мультиплексоров и демультиплексоров и провести их экспериментальное исследование в программе MicroCap.

1 Теоретическое введение

Мультиплексором называется комбинационное логическое устройство, предназначенное для управляемой передачи данных от нескольких источников информации в один выходной сигнал.

В цифровых схемах требуется управлять ключами при помощи логических уровней. Иначе задачей мультиплексора является выполнение функции электронного ключа с электронным управлением посредством цифрового сигнала.

Входы мультиплексора подразделяются на информационные $Д_0$, $Д_1$, …, $Д_{n-1}$ и адресные (управляющие)A_0, A_1, …, A_{k-1}. Принято считать, что $2^k = n$, где k и n – число адресных и информационных входов соответственно. Двоичный код, поступающий на адресные входы, определяет (выбирает) один из информационных входов, значение переменной с которого передается на выход y, т.е. мультиплексор реализует функцию:

$$y = Д_i, если \sum_{i=0}^{k-1} A_i \cdot 2^i \qquad (1.1)$$

Кроме того, микросхема мультиплексора имеет вход разрешения работы E.

Таблица истинности мультиплексора с двумя адресными и четырьмя информационными входами представлена ниже.

Таблица 1.1 – Таблица истинности мультиплексора

E	***A_0***	***A_1***	***Q***
1	x	x	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3

На основании данной таблицы истинности можно составить логическую схему мультиплексору (Рисунок 1.1)

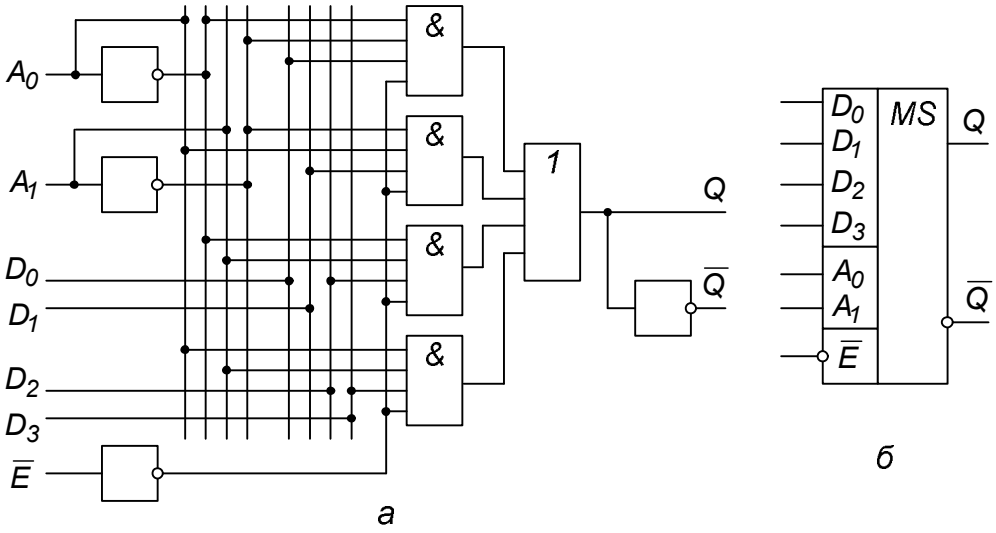


Рисунок 1.1 - Логическая схема (*а*), условно-графическое обозначение (*б*) мультиплексора

На схеме обозначено: A_0, A_1 – адресные входы, определяющие вход, с которого сигнал будет передан на выход Q; D_0, D_1, D_2, D_3– информационные входы; E – вход разрешения работы.

Учитывая, что вход E дает разрешающий сигнал, то работа мультиплексора описывается соотношением:

$$y = Д_0 \overline{A_1}\,\overline{A_0} + Д_1 \overline{A_1} A_0 + Д_2 A_1 \overline{A_0} + Д_3 A_1 A_0. \qquad (1.2)$$

Из (1.2) следует, что при любом значении адресного кода все слагаемые, кроме одного равны нулю. Ненулевое слагаемое равно $Д_i$, где i – значение текущего адресного кода.

В [2] предлагается вариант построения мультиплексора на основе дешифратора (Рисунок 1.2).

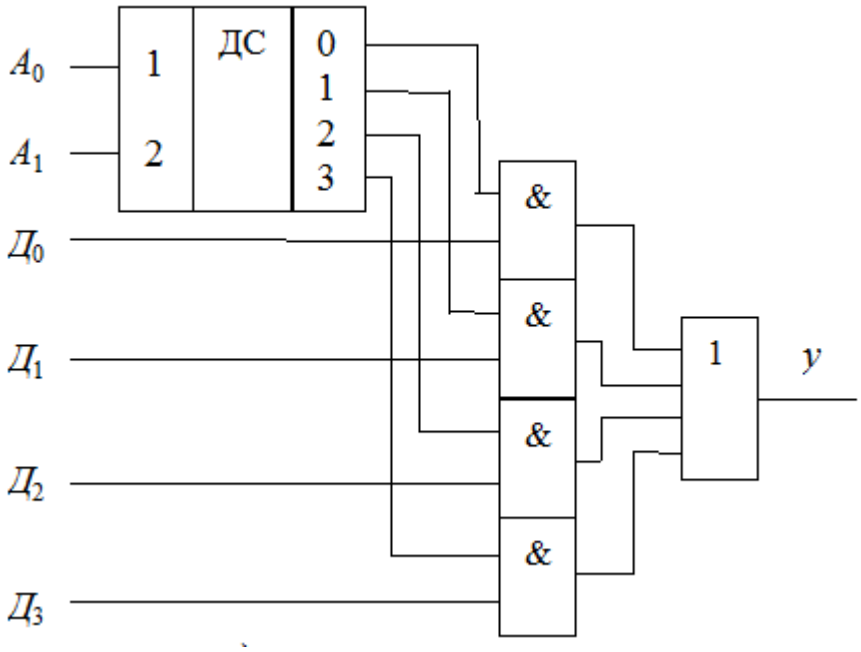


Рисунок 1.2 - Мультиплексор 4-1 на основе дешифратора

В данном случае мультиплексор построен как совокупность двухвходовых конъюкторов данных (их число равно числу информационных входов), управляемых выходными сигналами дешифратора, дешифрирующего двоичный адресный код. Выходы конъюкторов объединены схемой ИЛИ.

Мультиплексоры 4-1, 8-1, 16-1 выпускаются в составе многих серий цифровых интегральных схем и имеют буквенный код КП. Например, К555КП1 – мультиплексор 2-1 (в данном корпусе размещаются четыре мультиплексора), К555КП12 – мультиплексор 4-1 (в одном корпусе размещаются два мультиплексора) и т.д.

В тех случаях, когда функциональные возможности интегральных схем мультиплексоров не удовлетворяют разработчиков по числу информационных

входов, прибегают к их каскадированию с целью наращивания числа входов до требуемого значения. Наиболее универсальный способ наращивания размерности мультиплексора состоит в построении пирамидальной структуры, состоящей из нескольких мультиплексоров. При этом первый ярус схемы представляет собой столбец, содержащий столько мультиплексоров, сколько необходимо для получения нужного числа информационных входов. Все мультиплексоры этого столбца коммутируются одним и тем же адресным кодом, составленным из соответствующего числа младших разрядов общего адресного кода. Старшие разряды адресного кода используются во втором ярусе, мультиплексор которого обеспечивает поочередную работу мультиплексоров первого яруса на общий выход.

Демультиплексором называется комбинационное логическое устройство, предназначенное для управляемой передачи данных от одного источника информации в несколько выходных каналов.

Таблица функционирования демультиплексора, имеющего n = 4 информационных выходов (y_0, y_1, y_2, y_3) и k = 2 адресных входов (A_0, A_1), представлена в табл. 1.2.

Таблица 1.2 – Таблица истинности демультиплексора

E	A_0	A_1	Q_0	Q_1	Q_2	Q_3
1	x	x	0	0	0	0
0	0	0	D	0	0	0
0	0	1	0	D	0	0
0	1	0	0	0	D	0
0	1	1	0	0	0	D

Логическая схема демультиплексора и его условно-графическое обозначение приведены на рисунке 1.3.

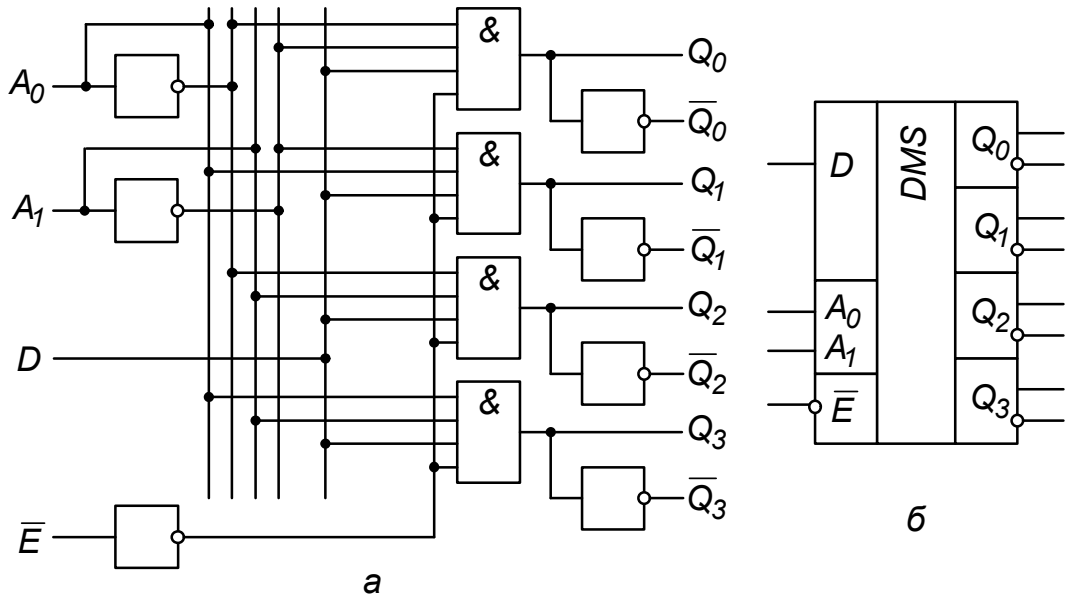


Рисунок 1.3 - Логическая схема (*а*), условно-графическое обозначение (*б*) демультиплексора

Уравнения, описывающие работу демультиплексора, представлены в виде функции (1.4):

$$y_0 = Д \cdot \overline{A_1} \cdot \overline{A_0};\ y_1 = Д \cdot \overline{A_1} \cdot A_0;\ y_2 = Д \cdot A_1 \cdot \overline{A_0};\ y_3 = Д \cdot A_1 \cdot A_0. \qquad (1.4)$$

Использование мультиплексора и демультиплексора.

Термином “мультиплексирование” называют процесс передачи данных от нескольких источников по общему каналу, а устройство, осуществляющее на передающей стороне операцию сведения данных в один канал, принято называть мультиплексором. Подобное устройство способно осуществлять временное разделение сигналов, поступающих от нескольких источников, и передавать их в канал (линию) связи друг за другом в соответствии со сменой кодов на своих адресных входах.

На приемной стороне обычно требуется выполнить обратную операцию – демультиплексирование, т.е. распределение порций данных, поступивших по каналу связи в последовательные моменты времени, по своим приемникам. Эту операцию выполняет демультиплексор.

Мультиплексор можно использовать в качестве универсального логического элемента для реализации любой логической функции от числа аргументов, равного числу адресных входов мультиплексора.

В программе MicroCap микросхема мультиплексора имеет название 74ALS151 (Рисунок 1.4).

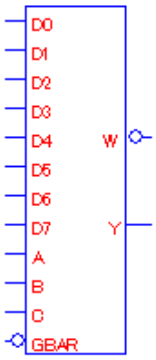


Рисунок 1.4 - Микросхема мультиплексора 74ALS151

На микросхеме 74ALS151:

– D0–D7 информационные входы;

– A,B,C – адресные входы;

– GBAR – управляющий вход;

– Y– прямой выход;

– W–инверсный выход.

В программе MicroCap микросхема демультиплексора имеет название 74HC138 (Рисунок 1.5).

Рисунок 1.5 - Микросхема демультиплексора 74HC138.

На микросхеме 74HC138:

– A,B,C – адресные входы;

– G2ABAR,G2BBAR – управляющий вход;

– Y0 – Y7 – выходы

2 Выполнение работы

2.1 Реализовать в программе MicroCap мультиплексор «4-1» на основании таблицы 1.1. и схемы, изображённой на Рисунок 1.1.

2.2 Реализовать в программе MicroCap мультиплексор «4-1» на основе дешифратора, согласно рисунку 1.2.

2.3 Реализовать в программе MicroCap мультиплексор «8-1». Данный мультиплексор должен иметь три адресных и восемь информационных входов. Записать на основе полученных данных таблицу истинности.

2.4 Реализовать в программе MicroCap демультиплексор «1-4»

2.5 Исследовать в программе MicroCap микросхему мультиплексора 74ALS151 и демультиплескора 74HC138.

2.6 Сделать выводы по работе.

3. Контрольные вопросы

3.1 Дайте определение мультиплексора и демультиплексора.

3.2 Перечислите применения мультиплексоров и демультиплексоров.

3.3 В чем суть каскадирования мультиплексоров? Объясните как на основе ИС мультиплексоров “8-1” спроектировать мультиплексор на 16, 32, и т.д. входов.

3.4 Объясните, как с помощью демультиплексора можно осуществить преобразование последовательного кода в параллельный.

3.5 Объясните, как с помощью мультиплексора можно осуществить преобразование параллельного кода в последовательный.

ЛАБОРАТОРНАЯ РАБОТА 6
СИНТЕЗ И ИССЛЕДОВАНИЕ ТРИГГЕРОВ

Цель работы: изучение функционирования триггеров различных типов, принципов их синтеза и взаимопреобразования.

1 Теоретическое введение

Все цифровые устройства принято разбивать на два вида:

– комбинационные цифровые устройства (КЦУ);

– последовательностные цифровые устройства (ПЦУ).

Отличительные особенности этих классов ЦУ состоят в следующем. Для КЦУ значения выходных переменных в некоторый момент времени определяются только значениями входных переменных в тот же момент времени. Для ПЦУ значения выходных переменных определяются не только входными переменными в данный момент, но и их значениями в предшествующие моменты времени. Примером, поясняющим принцип работы ПЦУ, является телефон. Чтобы соединиться с определенным абонентом, следует набрать последовательность цифр, соответствующую его номеру. Произойдет ли подключение к нужному абоненту, когда набирается последняя цифра, зависит как от этой цифры, так и от ранее набранной комбинации цифр.

Изменения значений входных переменных цифровых устройств происходят дискретно во времени. При этом временные интервалы, в течение которых эти значения сохраняются неизменными, называют тактами работы цифрового устройства. Если пронумеровать такты в порядке их возрастания, то для некоторого k-го такта работы ПЦУ зависимость выходных переменных от входных в общем виде может быть задана соотношением (1):

$$Y^k = F(X^k; X^{k-1}; X^{k-2}; ...; X^{k-r}), \qquad (1.1)$$

где $Y^k = (y_1^k, y_2^k, ..., y_m^k)$ -вектор выходных переменных, соответствующий k-ому такту работы;

m – число выходов ПЦУ;

$X^{k-j} = (x_1^{k-j}, x_2^{k-j}, ..., x_n^{k-j})$ - вектор входных переменных соответственно k-го, k-1, …, k-r тактов работы, j=0,1,…,r;

n – число входов ПЦУ;

F – оператор преобразования ПЦУ.

Для реализации зависимости (1.1) ПЦУ должно характеризоваться свойством запоминания входных переменных, т.е. устройство должно обладать памятью. Память ПЦУ может охватывать произвольное, но обязательно конечное число (r) тактов работы. Поэтому за ПЦУ закрепились также следующие наименования: цифровое устройство с памятью, многотактное цифровое устройство, конечный автомат.

Свойство запоминания информации обеспечивается наличием у ПЦУ r различных устойчивых внутренних состояний Q_1, Q_2, …, Q_r, каждое из которых характеризуется определенной комбинацией сигналов во внутренних цепях ПЦУ. По аналогии со входными и выходными переменными внутренние переменные (состояния) кодируются двоичными L-разрядными числами. Значение L определяется из соотношения $L=[\log_2 r]+1$.

Из вышеизложенного следует: ПЦУ – это цифровой преобразователь информации, способный принимать различные состояния, хранить (сохранять) их, переходить под воздействием входных сигналов из одного состояния в другое и формировать выходные сигналы. Следовательно, задание оператора, реализуемого ПЦУ предполагает:

– установление связи выходных переменных со входными и внутренними переменными для одного и того же такта работы ПЦУ, т.е. связи вида

$$Y^k = \Phi(X^k; Q^k) \tag{1.2}$$

– установление связи внутренних переменных для (k+1)-го такта со значениями входных и внутренних переменных k-го такта, т.е. связи вида

$$Q^{k+1} = \Psi(X^k; Q^k) \tag{1.3}$$

Приведенные соотношения именуют функциями (уравнениями) выходов (1.2) и переходов (1.3).

Простейшими ПЦУ являются триггеры. Триггер – простейшее последовательностное устройство, которое может длительно находиться в одном из нескольких возможных устойчивых состояний и переходить из одного в другое под воздействием входных сигналов. Триггер предназначен для хранения значения одной логической переменной (или значения однозначного двоичного числа; при хранении многоразрядных двоичных чисел для запоминания значения каждого разряда числа используется отдельный триггер). В соответствии с этим триггер имеет два состояния: одно из них обозначается как состояние 0, другое – как состояние 1. Воздействуя на входы триггера, его устанавливают в нужное состояние. Классификация тригеров представлена на рисунке 1.1.

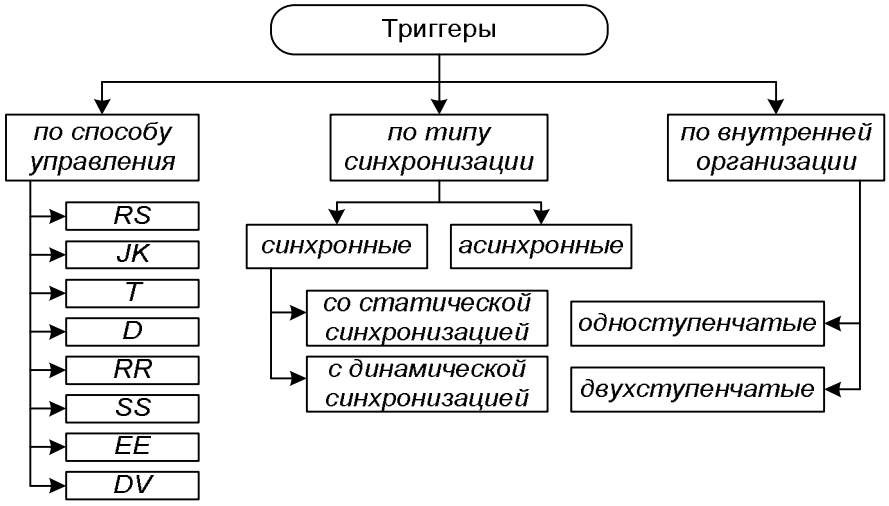


Рисунок 1.1 - Классификация триггеров

По способу работы с сигналами различают синхронные и асинхронные триггерные схемы. Асинхронный триггер изменяет свое состояние непосредственно в момент появления соответствующего информационного сигнала. Синхронные триггеры реагируют на информационные сигналы только при наличии соответствующего сигнала на так называемом входе синхронизации "С". Этот вход также иногда обозначают терминами строб, такт.

Синхронные триггеры, в свою очередь, подразделяют на триггеры со статическим (статические) и динамическим (динамические) управлением по входу синхронизации "С". Статические триггеры воспринимают информационные сигналы при подаче на вход "С" логической 1 (прямой вход) или логического 0 (инверсный вход). Динамические триггеры воспринимают информационные сигналы при изменении (перепаде) сигнала на входе С от 0 к 1 (прямой динамический С-вход) или от 1 к 0 (инверсный динамический С-вход). Наиболее распространенные условно-графические обозначения входов синхронизации приведены на рисунке 1.2

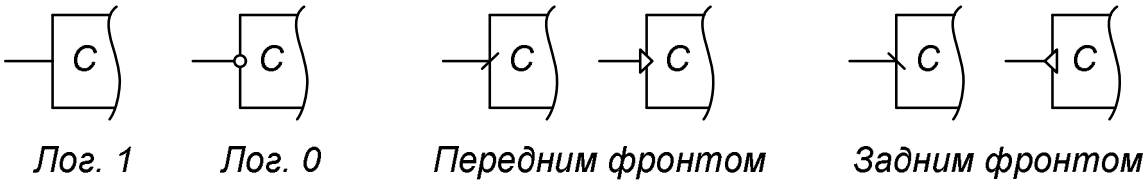


Рисунок 1.2 - Условно-графические обозначения входов синхронизации

По внутренней организации триггеры подразделяют на одноступенчатые (однотактные) и двухступенчатые (двухтактные). В одноступенчатом триггере имеется одна ступень запоминания информации, а в двухступенчатом – две такие ступени. Вначале информация записывается в первую ступень, а затем переписывается во вторую и появляется на выходе. Двухступенчатый триггер обозначают ТТ.

По способу управления различают *RS*, *D*, *JK*, *T*, *RR*, *SS*, *EE* и *DV* триггеры. Каждый тип триггера имеет собственную таблицу истинности. Выходное состояние триггера обычно обозначают буквой Q_t или Q_{t+1}.

Индекс возле буквы означает состояние до подачи сигнала (t) или после подачи сигнала ($t+1$).

Если триггер синхронный, то существует также дополнительный вход синхронизации. Для того чтобы такой триггер учел информацию на

синхронных входах, на входе синхронизации необходимо сформировать активный сигнал.

Как правило, входы триггеров обозначают следующим образом:

– S (от англ. Set, установить) – вход в RS-триггере;

– R (от англ. Reset, сброс) – вход в RS-триггере;

– J (от англ. Jump, прыжок) – вход в JK-триггере;

– K (от англ. Kill, убить) – вход в JK-триггере;

– T (от англ. Toggles, переключить) – счетный вход в T-триггере;

– C (от англ. Clock, время) вход синхронизирующего сигнала (при тактировании по фронту он часто обозначается стрелкой: стрелка внутрь – тактирование по переднему фронту, наружу – по заднему);

– D (от англ. Delay, задержка) – вход в D-триггере;

– E или EN (от англ. Enable, разрешить) – дополнительный асинхронный управляющий вход для разрешения приема информации (иногда используют букву V).

Входы J, K, T, D всегда синхронные, т.е. тактируются по синхронизирующему сигналу на входе C. Разумеется, в каждом конкретном триггере имеются лишь некоторые из перечисленных входных линий. Входы S и R зачастую присутствуют не только в RS триггерах, но и в других типах триггеров, где предназначены, в основном, для асинхронного сброса устройства в 0 или установки в 1.

Рассмотрим принцип работы основных видов триггеров.

RS-триггер- триггер, который сохраняет свое предыдущее состояние при нулевых входах и меняет свое выходное состояние при подаче на один из его входов единицы (или наоборот – сохраняет состояние при единичных входах и меняет состояние при нулевых). При подаче 1 на вход *S* выходное состояние становится равным логической 1, а при подаче 1 на вход *R* выходное состояние становится равным логическому 0. Если RS-триггер синхронный, то состояние его входов учитывается только в момент тактирования, например по переднему фронту импульса.

Таблица истинности RS-триггера имеет вид, представлены ниже.

Таблица 1.1 – Таблица истинности RS-триггера

S	***R***	Q_{t+1}	***Функция***
0	0	Q_t	Хранение
1	0	1	Установка в 1
0	1	0	Установка в 0
1	1	x	Запрещенная комбинация

Простейшая логическая схема RS-триггера, реализующая эту таблицу, приведена на рисунке 1.3а. Состояние, при котором на оба входа *R* и *S* одновременно поданы логические 1, является запрещенным.

Так, согласно схеме RS-триггера (Рисунок 1.3а) при подаче на оба инверсных входа лог. 1 триггер перейдет в состояние, когда на обоих выходах будут лог. 0, что не соответствует логике его функционирования, поскольку инверсный выход будет равен неинверсному, т.е. 0 = 1.

RS-триггер также можно реализовать на логических элементах 2И-НЕ. Простейшая схема такого RS-триггера приведена на рисунке 1.3б.

Отличие в работе этой схемы заключаться в том, что входы являются инверсными, поэтому сброс и установка триггера осуществляется нулевыми логическими уровнями, а запрещенной комбинацией является подача на оба входа логического 0.

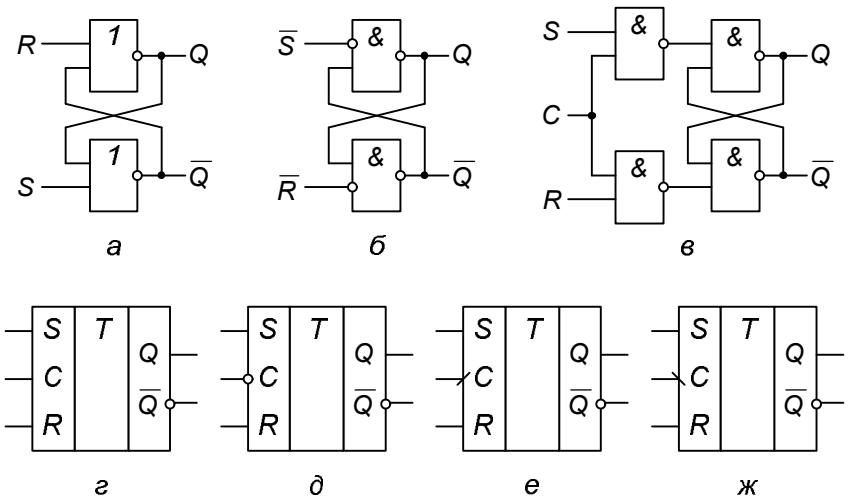


Рисунок 1.3 - Логическая схема RS-триггера (а, б – упрощенные,в – серийно выпускаемого синхронного одноступенчатого)и его условно-графические обозначения (г, д – со статическойи е, ж – динамической синхронизацией)

Серийно выпускаемые синхронные одноступенчатые RS-триггеры снабжены дополнительным входом*C*, на который подается синхросигнал, разрешающий обработку входных сигналов *S* и *R* (Рисунок 1.3в). Неактивное значение синхросигнала обеспечивает на входах запоминающей ячейки состояние управляющих сигналов $S = R = 1$, что соответствует для нее режиму хранения.

На рисунке 1.3 е-ж приведены условно-графические обозначения RS-триггера с различной синхронизацией. На рисунках 1.3г и 1.3д изображены RS-триггеры со статической синхронизацией (прямой и инверсной), Рисунок 1.3е и ж – с динамической синхронизацией соответственно передним и задним фронтом синхросигнала.

D-триггер.

В RS-триггерах для записи логического 0 и логической 1 требуются разные входы, что не всегда удобно. При записи и хранении данных один бит может принимать значение как нуля, так и единицы. Для его передачи достаточно одного провода. Как мы уже видели ранее, сигналы установки и сброса триггера не могут появляться одновременно, поэтому можно

объединить эти входы при помощи инвертора, как показано на рисунке 1.4. Такой триггер получил название D-триггер.

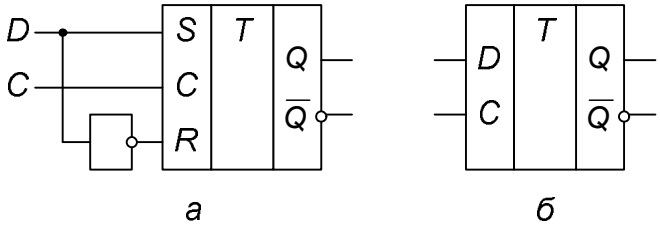


Рисунок 1.4 - Логическая схема D-триггера (*а*) и его условно-графическое обозначение (*б*)

Таблица истинности D-триггера представлена ниже.

Таблица 1.2 - Таблица истинности D-триггера

C	***D***	Q_t	Q_{t+1}	***Функция***
0	x	0	0	Режим хранения информации
0	x	1	1	
1	0	x	0	Режим записи информации
1	1	x	1	

Как видно из этой таблицы, D-триггер способен запоминать по синхросигналу и хранить один бит информации.

Так как информация на выходе остается неизменной до прихода очередного импульса синхронизации, D-триггер называют также триггером с запоминанием информации или триггером-защелкой.

JK-триггер

JK-триггер работает так же, как RS-триггер, с одним лишь исключением: при подаче логической 1 оба входа *J* и *K* состояние выхода триггера изменяется на противоположное. Таблица истинности JK-триггера представлена ниже.

Таблица 1.3 - Таблица истинности JK-триггера

J	***K***	Q_{t+1}	***Функция***
0	0	Q_t	Хранение
0	1	0	Установка в 0
1	0	1	Установка в 1
1	1	$\overline{Q_t}$	Инвертирование предыдущего состояния

Вход *J* (от англ. *Jump* – прыжок) аналогичен входу *S* у RS-триггера, вход *K* (от англ. *Kill* – ликвидировать) аналогичен входу *R*. При подаче 1 на вход *J* и 0 на вход *K* выходное состояние триггера становится равным логической 1. А при подаче 1 на вход *K* и 0 на вход *J* выходное состояние триггера становится равным логическому 0. JK-триггер в отличие от RS-триггера не имеет запрещенных состояний на основных входах, однако это никак не помогает при нарушении правил разработки логических схем. На практике применяются только синхронные JK-триггеры, то есть состояния основных входов *J* и *K* учитываются только в момент тактирования, например по положительному фронту импульса на входе синхронизации.

В качестве примера на рисунке 1.5а приведен один из вариантов логических схем JK-триггера.

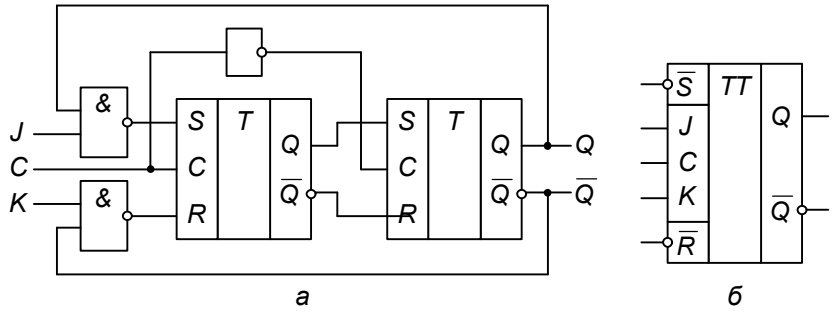


Рисунок 1.5 - Логическая схема JK -триггера (*а*)и его условное обозначение (*б*)

В промышленно выпускающихся микросхемах триггеров обычно реализуются как входы *J* и *K*, так и входы *R* и *S*, что позволяет устанавливать триггер в заранее определенное исходное состояние.

Т-триггер – это триггер, логическое состояние выходного сигнала которого изменяется на противоположное по каждому такту синхроимпульса при условии наличия логической 1 на входе Т. Т-триггер часто называют счетным триггером. Он может строиться как на JK, так и на D-триггерах (Рисунок 1.6). Как можно видеть в таблице истинности JK-триггера (см. п.4.3.1.4), он переходит в инверсное состояние каждый раз при одновременной подаче на входы *J* и *K* логической 1. Это свойство позволяет создать на базе JK-триггера Т-триггер, объединяя входы *J* и *K*. Наличие в D-триггере динамического *C* входа также позволяет получить на его основе Т-триггер. При этом вход *D* соединяется с инверсным выходом, а на вход*C* подаются счетные импульсы. В результате триггер при каждом счетном импульсе запоминает значение, т.е. будет переключаться в противоположное состояние.

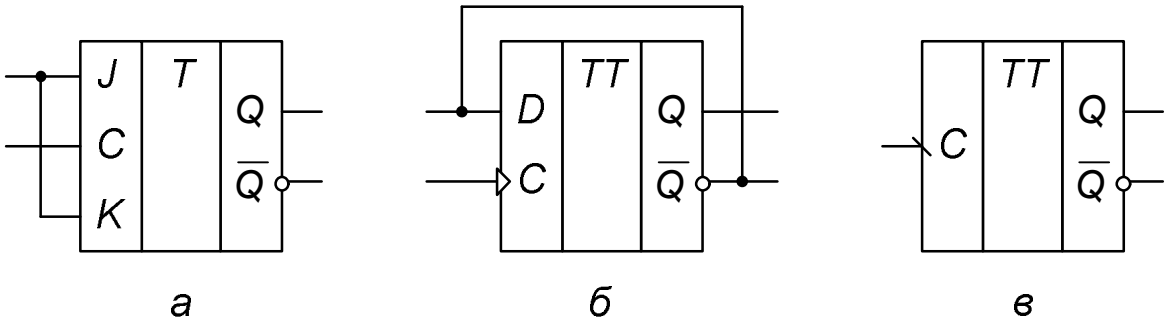


Рисунок 1.6 - Логическая схема Т-триггера, построенная на основе JK-триггера (*а*) и D-триггера (*б*), и его условно-графическое обозначение (*в*)

Временная диаграмма Т-триггера, работающего по заднему фронту синхронизирующего сигнала, приведена на рисунке 1.7. Переключение выходного состояния триггера на противоположное происходит по заднему фронту каждого импульса на входе синхронизации.

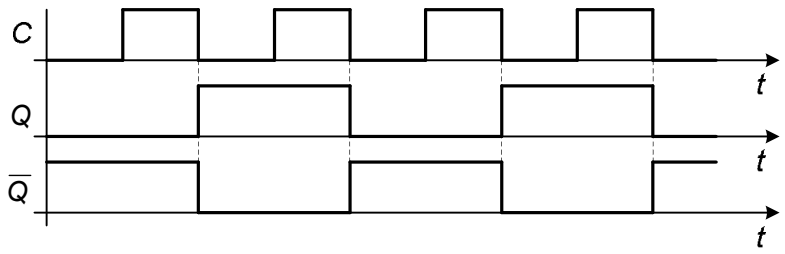


Рисунок 1.7 - Идеализированная временная диаграмма работы Т-триггера с переключением по заднему фронту синхросигнала

В цифровых электронных устройствах Т-триггер часто применяют для понижения частоты в 2 раза, при этом наT вход подают единицу, а на C – сигнал с частотой, которая будет поделена. Кроме того, Т-триггеры используются при построении схем различных счетчиков. Поэтому Т-триггеры выпускаются промышленностью как в виде отдельных микросхем, так и включаются в состав больших интегральных схем (БИС) различного назначения.

2 Выполнение работы

2.1 Изучить принцип работы и реализовать в программе MicroCapсхемы RS-триггеров, изображенных на рисунке 1.3а – 1.3в. Сохранить временную диаграмму из программы MicroCap и на основании неё составить таблицу истинности для трёх случаев, аналогичную таблице 1.1.

2.2 Собрать схему, изображённую на рисунке 2.1. RS-триггернаходитсяпопути: DigitalPrimitives – Gated Flip-Flops/Latches – SRFF.

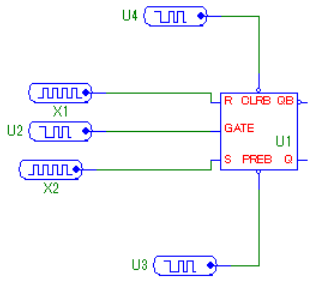


Рисунок 2.1 - Схема исследования RS-триггера

В блок X1(Reset, R)задать продолжительность логических «1» и «0» равную 100N. В блок X2(Set, S)задать продолжительность логических «1» и «0» равную 50N. В блокU2 (Clk, C)задать продолжительность логической «1» от 0nsдо 350ns. Затем должен идти логический «0». В блоки U3 и U4 задать постоянную логическую «1». Снять временную диаграмму всех входов, а также прямого и инверсного выхода. Составить таблицу истинности.

2.3 Собрать схему, изображённую на рисунке 2.2.Входные сигналы задать аналогично предыдущему заданию.

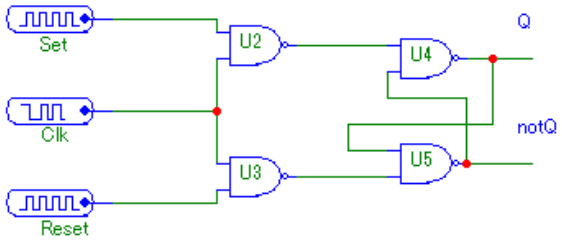


Рисунок 2.2 - Функциональная схема RS-триггера

Снять временную диаграмму всех входов, а также прямого и инверсного выхода. Составить таблицу истинности.

2.4 Собрать схему, изображённую на рисунке 2.3.Входные сигналы задать аналогично предыдущему заданию.

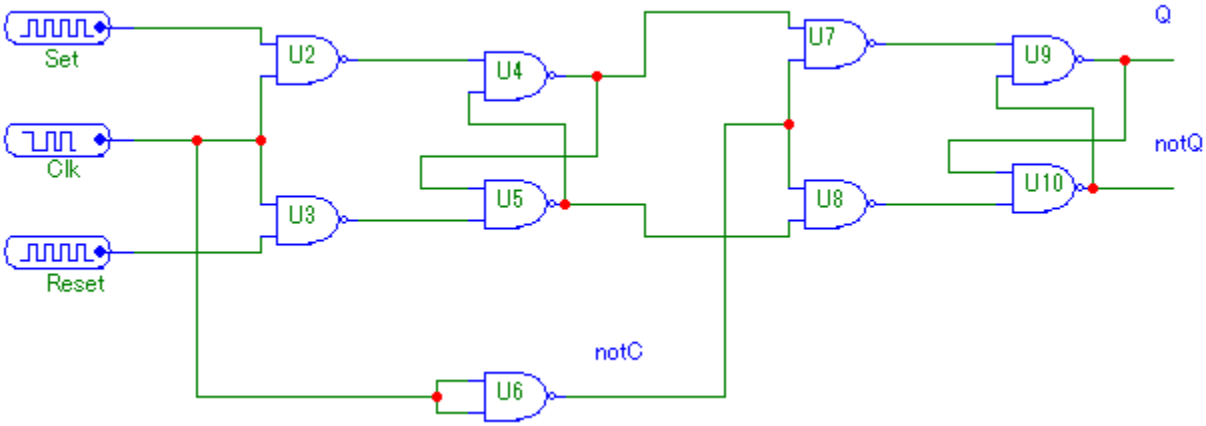


Рисунок 2.3 - Схема исследования двухступенчатого RS-триггера

Снять временную диаграмму всех входов, а также всех выходов, отмеченных на рисунке. Составить таблицу истинности.

2.5 Собрать схему D-триггера, изображённую на рисунке 2.4.

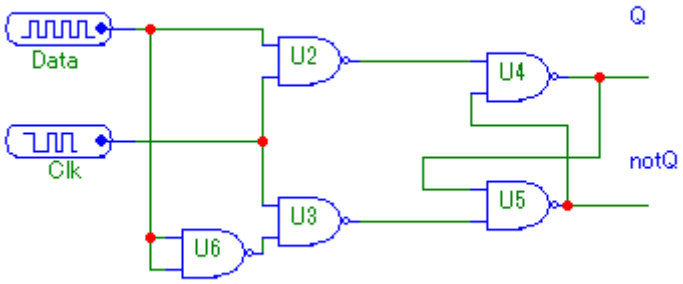


Рисунок 2.4 - Схема исследования D-триггера

Вход Dataимеет продолжительность логических «1» и «0» равную 50N. Вход Clkимеет продолжительность логической «1» от 0nsдо 350ns. Затем должен идти логический «0». Снять временную диаграмму всех входов, а также прямого и инверсного выхода. Составить таблицу истинности. Проверить работоспособность данного триггера на примере модели из MicroCap. Онанаходитсяпопути: DigitalPrimitives – GatedFlip-Flops/Latches –LATCH.

3. Контрольные вопросы

3.1 Приведите определение ПЦУ.

3.2 Приведите определение триггера, перечислите его отличительные особенности.

3.3 Что такое таблица переходов триггера? Изобразите таблицы переходов известных вам типов триггеров.

3.4 Изобразите временные диаграммы известных вам типов триггеров.

3.5 В чем отличие синхронных триггеров, управляемых уровнем, от триггеров с динамическим управлением?

3.6 Докажите возможность преобразования синхронного RS-триггера в D-триггер; JK-триггера в D- и T-триггеры, D-триггера в T-триггер.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Касьянов А.Н. Micro-Cap в схемотехнике: Учебное пособие. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2004. 112с.

2 Тлостанов Ю.К. Лабораторный практикум по дисциплине "Основы цифровой техники". – Каб.-Балк. ун-т, 2002. – 110 с.

3 Малахов О.С., Радионов А.А. Схемотехника цифровых электронных устройств: учеб.пособие. – Магнитогорск: ФГБОУ ВПО «МГТУ», 2012. – 152 с.

БАСКОВ СЕРГЕЙ НИКОЛАЕВИЧ
ЛИЦИН КОНСТАНТИН ВЛАДИМИРОВИЧ

АЛГЕБРА ЛОГИКИ И ОСНОВЫ ДИСКРЕТНОЙ ТЕХНИКИ

Лабораторный практикум
Для студентов направления подготовки
13.03.02 «Электроэнергетика и электротехника»
всех форм обучения

Подписано в печать 16.11.2016 г.		
Формат 60x90 1/16 Рег.№ 90	Печать цифровая Тираж 50 экз.	Уч.-изд.л. 3,8125

ФГАОУ ВО
Национальный исследовательский технологический университет «МИСиС»
Новотроицкий филиал
462359, Оренбургская обл., г. Новотроицк, ул. Фрунзе, 8.
E-mail: nfmisis@yandex.ru
Контактный тел. 8 (3537) 679729.