

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И
ИНФОРМАТИКИ**

Кафедра РОС

СОГЛАСОВАНО:

Руководитель ЦЗОПБ _____ Е.А.
Шиховец

Практикум

по дисциплине

ЦИФРОВАЯ СХЕМОТЕХНИКА

для направления 11.03.02 Центра заочного обучения по программам
бакалавриата

**Разработка и моделирование цифровых блоков электронной
аппаратуры.**

Составитель:

И.Б. Андреев _____

Рассмотрено и одобрено

на заседании кафедры РОС

Протокол № _____

Дата _____

Зав. кафедрой _____ А.В.

Пестряков

Москва 2023

ПРАКТИКУМ

по дисциплине

ЦИФРОВАЯ СХЕМОТЕХНИКА

Разработка и моделирование цифровых блоков электронной аппаратуры.

Составитель И.Б. Андреев, к.т.н., доцент

Рекомендовано для проведения практических занятий и выполнения контрольных работ студентам ЦЗОПБ по направлениям подготовки 11.03.02 Инфокоммуникационные технологии и системы связи по дисциплине Б1.О.15 Цифровая Схемотехника.

Издание утверждено

Рецензенты:

А.А. Кубицкий, к.т.н., доцент

О.Е. Данилин, к.т.н., доцент

ВВЕДЕНИЕ

Цель данного практикума состоит в том, чтобы активизировать работу студентов над материалом курса, способствовать закреплению теоретических положений, привить навыки самостоятельной работы со схемами. Практикум ознакомит студентов с элементами проектирования простых цифровых схем.

Наиболее подробно рассматриваются базовые вопросы курса: реализация логических функций, схемы и модели на основе логических элементов, дешифраторов и мультиплексоров, методы моделирования цифровых схем в статическом и динамическом режимах.

В составе практикума имеются краткие теоретические сведения, вопросы и задачи, предназначенные для проведения контрольных работ.

1. Теоретическая часть

Комбинационными цифровыми схемами называют схемы, выходные сигналы которых зависят только от сигналов, поданных на их входы в данный момент, и не зависят от состояния схемы в предыдущий момент времени. К комбинационным цифровым схемам обычно относят такие микросхемы, как дешифраторы, шифраторы, мультиплексоры и демультиплексоры.

Создание цифровых комбинационных устройств в настоящий момент в значительной степени формализовано, при этом для создания их принципиальных схем могут оказаться полезны некоторые законы алгебры логики, которые получены из таблиц истинности простейших логических элементов.

Комбинационные схемы строятся из элементарных логических элементов И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и др.. Соединяют эти элементы так, как это следует из логической формулы, т.е. вход одного элемента, в котором часть аргументов обработана как указано в формуле, подключается ко входу другого, где выполняется дальнейшая обработка логической функции. В схеме не должно быть обратных связей, т.е. соединения выходов последующих схем с входами предыдущих.

Таблица 1.

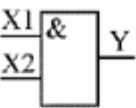
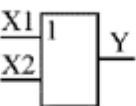
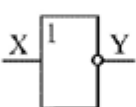
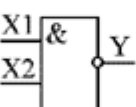
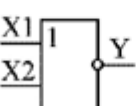
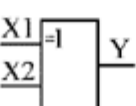
Название элемента	Условное обозначение элемента	Таблица истинности			Условное обозначение логической операции
		X2	X1	Y	
2И		0	0	0	$X1 * X2$ $X1 \wedge X2$
		0	1	0	
		1	0	0	
		1	1	1	
2ИЛИ		0	0	0	$X1 + X2$ $X1 \vee X2$
		0	1	1	
		1	0	1	
		1	1	1	
НЕ			0	1	\bar{X} $1X$
			1	0	
2И-НЕ		0	0	1	$\overline{X1 * X2}$ $\neg(X1 \wedge X2)$
		0	1	1	
		1	0	1	
		1	1	0	
2ИЛИ-НЕ		0	0	1	$\overline{X1 + X2}$ $\neg(X1 \vee X2)$
		0	1	0	
		1	0	0	
		1	1	0	
Исключающее ИЛИ		0	0	0	$X1 \oplus X2$
		0	1	1	
		1	0	1	
		1	1	0	

Таблица 1. Условные графические обозначения и выполняемые логические функции элементов

Двигаясь от начала схемы, на входе каждого элемента записывается буквенное выражение входного сигнала. Входы подаются на условное обозначение логических элементов, на выходе которых записывается логическая формула начала обработки входных сигналов, а затем их выходы соединяются так, как указано в заданной логической функции, и на выходе всей комбинационной схемы записывается выражение выполненной логической функции.

Любая цифровая комбинационная схема (логическая схема без памяти) полностью описывается таблицей истинности. При этом не обязательно чтобы все комбинации входных цифровых сигналов были

полезными. Возможна ситуация, когда только часть комбинаций входных логических сигналов является полезной. В этом случае выходные сигналы цифрового устройства для оставшихся комбинаций входных логических сигналов могут быть доопределены произвольно. Обычно при этом стараются выбирать цифровые значения выходных сигналов таким образом, чтобы схема цифрового устройства получилась простейшей.

Для реализации цифровых логических схем с произвольной таблицей истинности используется сочетание простейших логических элементов "И", "ИЛИ", "НЕ". Существует два способа синтеза цифровых схем, реализующих произвольную таблицу истинности. Это СКНФ (логическое произведение суммы входных сигналов) и СДНФ (сумма логических произведений входных сигналов).

При синтезе цифровой схемы, реализующей произвольную таблицу истинности, каждый выход анализируется (и строится схема) отдельно и независимо.

Совершенная дизъюнктивная нормальная форма (СДНФ)

Дизъюнктивная нормальная форма (ДНФ) представляет собой «сумму произведений», причём в качестве операции «умножения» выступает операция И (конъюнкция), а в качестве операции «сложения» — операция ИЛИ (дизъюнкция). Сомножителями являются различные переменные, причём они могут входить в произведение как в прямом, так и в инверсном виде.

Примером такой функции может служить следующее выражение:

$$y = \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + x_1 \cdot x_2 + x_2 \cdot x_3.$$

Совершенная дизъюнктивная нормальная форма (СДНФ) — одна из форм представления булевой функции в виде логического выражения. Представляет собой частный случай ДНФ, удовлетворяющий следующим трём условиям:

- в ней нет одинаковых слагаемых (элементарных конъюнкций);
- в каждом слагаемом нет повторяющихся переменных;
- каждое слагаемое содержит все переменные, от которых зависит булева функция (каждая переменная может входить в слагаемое либо в прямой, либо в инверсной форме).

Любая булева функция, не являющаяся ложной, может быть приведена к СДНФ, причём единственным образом, то есть для любой выполнимой функции алгебры логики существует своя СДНФ, причём единственная.

Для реализации таблицы истинности при помощи логических элементов "И" (СДНФ) достаточно рассмотреть только те строки таблицы истинности, которые содержат логические "1" в выходном сигнале. Строки, содержащие в выходном сигнале логический 0 в построении цифровой схемы не участвуют.

Каждая строка, содержащая в выходном сигнале логическую "1", реализуется схемой логического элемента "И" с количеством входов, совпадающим с количеством входных сигналов в таблице истинности.

Входные сигналы, описанные в таблице истинности логической единицей, подаются на вход этого логического элемента непосредственно, а входные сигналы, описанные в таблице истинности логическим нулем, подаются на вход этого же логического элемента "И" через инверторы. Объединение сигналов с выходов логических элементов "И", реализующих отдельные строки таблицы истинности, производится при помощи логического элемента "ИЛИ". Количество входов в логическом элементе "ИЛИ" определяется количеством строк в таблице истинности, в которых в выходном сигнале присутствует логическая единица.

Рассмотрим конкретный пример. Пусть необходимо реализовать таблицу истинности, приведенную в таблице 2:

Входы				Выходы	
X4	X3	X2	X1	Y1	Y2
0	0	0	0	0	0
<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	1
<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	0	1	1	0	0
<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	1

Таблица 2. Произвольная таблица истинности.

Для синтеза цифровой схемы, реализующей сигнал Out0, достаточно рассмотреть строки, с подчеркнутыми цифрами. В таблице истинности три строки, содержащие единицу в выходном сигнале Out0, поэтому в формуле СДНФ будет содержаться три произведения входных сигналов:

$$Y1 = \overline{X4} * \overline{X3} * \overline{X2} * X1 + \overline{X4} * X3 * X2 * \overline{X1} + X4 * X3 * \overline{X2} * \overline{X1}$$

Так как количество переменных в каждом слагаемом данного логического выражения равно, то такое логическое выражение называется совершенным. (Совершенная Дизъюнктивная Нормальная Форма — СДНФ)

Полученное логическое выражение реализуется в схеме, приведенной на рисунке 2. Переменные X обозначены, как входы схемы (In), переменные

Y, как выходы (Out). Как и в формуле, каждая строка реализуется своим логическим элементом "И", затем выходы этих логических элементов объединяются при помощи логического элемента "ИЛИ". Количество входов логического элемента "И" (дизъюнкция) в СДНФ однозначно определяется количеством входных сигналов в таблице истинности. Количество этих элементов, а значит и количество входов в логическом элементе "ИЛИ" определяется количеством строк с единичным сигналом на реализуемом выходе цифровой схемы.

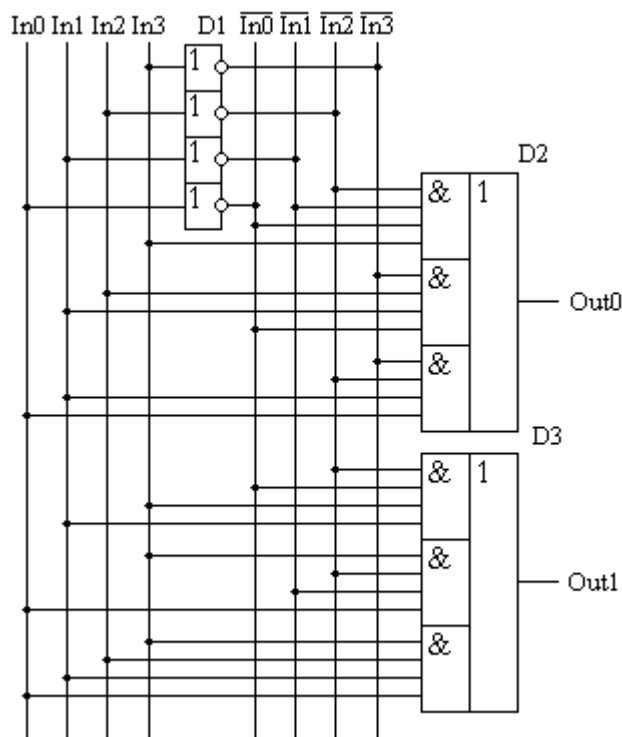


Рис.1. Принципиальная схема, реализующая таблицу истинности, приведенную в таблице 2.

Для построения схемы, реализующей сигнал Out1 (Y1), достаточно рассмотреть строки, выделенные курсивом. Эти строки реализуются микросхемой D3. Принцип построения этой схемы не отличается от примера, рассмотренного выше. В таблице истинности присутствуют всего три строки, содержащие единицу в выходном сигнале Out1, поэтому в формуле СДНФ для Y1 будут присутствовать три произведения входных сигналов:

$$Y2 = \overline{X4} * \overline{X3} * \overline{X2} * X1 + \overline{X4} * \overline{X3} * X2 * X1 + \overline{X4} * X3 * X2 * X1$$

Совершенная конъюнктивная нормальная форма (СКНФ)

Еще одним способом реализации цифровых комбинационных схем является запись логического выражения в совершенной конъюнктивной нормальной форме (СКНФ). Применение СКНФ оправдано при большом количестве логических единиц в выходном сигнале проектируемой цифровой схемы, как это показано в качестве примера в таблице истинности 2 (Таблица 3), или в ЭСЛ-микросхемах (где базовым логическим элементом является «Или-Не», в отличие от базового «И-Не» в ТТЛ-микросхемах).

Входы				Выходы	
X4	X3	X2	X1	Y1	Y2
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Таблица 3. Пример таблицы истинности 2

Для реализации цифрового комбинационного устройства по таблице истинности при помощи логических элементов "ИЛИ" (СКНФ) достаточно рассмотреть только те строки таблицы истинности, которые содержат логические "0" в выходном сигнале. Строки, содержащие в выходном сигнале логическую "1" в построении логического выражения, а, следовательно, и принципиальной схемы цифрового устройства не участвуют. Каждая строка, содержащая в выходном сигнале логический "0", реализуется схемой логического элемента "ИЛИ" с количеством входов, совпадающим с количеством входных сигналов в таблице истинности.

Для построения схемы, реализующей сигнал Y1 (Out0 на схеме рис.2), достаточно рассмотреть строки, выделенные курсивом. В рассматриваемой таблице истинности имеются всего две строки, содержащие логический ноль в выходном сигнале Out0, поэтому в формуле СКНФ будет содержаться две суммы входных сигналов:

$$Y1 = (\overline{X4} + \overline{X3} + \overline{X2} + X1) * (\overline{X4} + X3 + \overline{X2} + \overline{X1})$$

Входные сигналы, описанные в таблице истинности логическим нулём, подаются на вход этой схемы непосредственно, а входные сигналы, описанные в таблице истинности логической единицей, подаются на логического элемента "ИЛИ" через инверторы. Объединение сигналов с выходов схем "ИЛИ", реализующих отдельные строки таблицы истинности, производится при помощи схемы логического элемента "И". Количество входов в схеме "И" определяется количеством строк в таблице истинности, в которых в выходном сигнале присутствует логическая единица.

Аналогично реализуется и функция Y2 (выход Out1 на рис. 2):

$$Y2 = (\overline{X4} + X3 + \overline{X2} + X1) * (\overline{X4} + X3 + X2 + \overline{X1})$$

Полученная формула реализуется в схеме на рисунке 2.

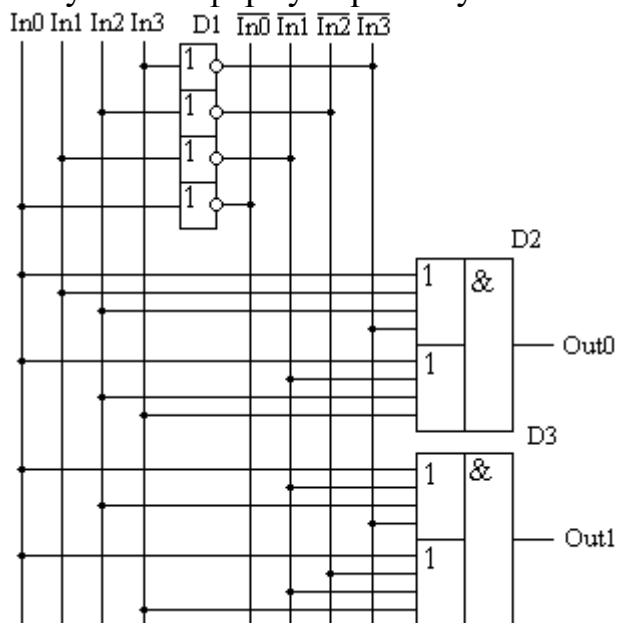


Рис.2. Принципиальная схема, реализующая таблицу истинности, приведенную в Таблице 2.

Минимизация логических функций

Полученное логическое выражение в ряде случаев можно значительно упростить, или минимизировать. Для этого используется несколько методов.

1. Метод непосредственных преобразований.

При этом методе используются законы алгебры логики.

Напомним некоторые определения и соотношения, используемые в алгебре логики.

Все возможные логические функции от одной переменной представлены в таблице 4.

Таблица 4.

Функция	x		Наименование функции	Обозначение функции
	x=0	x=1		
f_0	0	0	Константа "ноль"	$f(x)=0$
f_1	0	1	Тождественная функция	$f(x)=x$
f_2	1	0	Отрицание	$f(x)=\overline{x}$ $f(x)=\overline{x}$
f_3	1	1	Константа "единица"	$f(x)=1$

Все возможные логические функции от двух переменных представлены в таблице 5.

Таблица 5.

№ функ- ции	Значение функции на наборах логичес- ких переменных				Наименование функции	Обозначение функции
	x=0 y=0	x=1 y=0	x=0 y=1	x=1 y=1		
f_0	0	0	0	0	Константа "ноль"	$f(x,y)=0$
f_1	0	0	0	1	Конъюнкция	$f(x,y)=x \& y$ $f(x,y)=x \wedge y$ $f(x,y)=x \bullet y$ $f(x,y)=xy$
f_2	0	0	1	0	Запрет по y	$x \Delta y$
f_3	0	0	1	1	x	$f(x,y)=x$
f_4	0	1	0	0	Запрет по x	$y \Delta x$
f_5	0	1	0	1	y	$f(x,y)=y$
f_6	0	1	1	0	Сумма по mod2 (неравнозначность)	$f(x,y)=x \oplus y$
f_7	0	1	1	1	Дизъюнкция	$f(x,y)=x \vee y$ $f(x,y)=x + y$
f_8	1	0	0	0	Стрелка Пирса (Вебба)	$f(x,y)=x \downarrow y$ $f(x,y)=x \circ y$
f_9	1	0	0	1	Равнозначность	$f(x,y)=x \equiv y$ $f(x,y)=x \oslash y$
f_{10}	1	0	1	0	Инверсия y	$f(x,y)=\neg y$ $f(x,y)=\bar{y}$
f_{11}	1	0	1	1	Импликация от y к x	$f(x,y)=y \rightarrow x$
f_{12}	1	1	0	0	Инверсия x	$f(x,y)=\neg x$ $f(x,y)=\bar{x}$
f_{13}	1	1	0	1	Импликация от x к y	$f(x,y)=x \rightarrow y$
f_{14}	1	1	1	0	Штрих Шеффера	$f(x,y)=x / y$
f_{15}	1	1	1	1	Константа "единица"	$f(x,y)=1$

Ниже представлены основные соотношения для некоторых наиболее часто используемых логических функций.

Таблица 6.

Отрицание	Конъюнкция		Дизъюнкция	
$\bar{0} = 1$	$0 * 0 = 0$	$x * \bar{x} = 0$	$0 + 0 = 0$	$x + \bar{x} = 1$
$\bar{1} = 0$	$0 * 1 = 0$	$x * x = x$	$0 + 1 = 1$ $1 + 1 = 1$	$x + x = x$
$\bar{\bar{x}} = x$	$1 * 1 = 1$	$x * x * \dots * x = x$	$0 + x = x$	$x + x + \dots + x = x$

	$0 * x = 0$	$x * y = y * x$	$1 + x = 1$	$x + y = y + x$
	$1 * x = x$	$x * y * z = (x * y) * z = x * (y * z)$		$x + y + z = (x + y) + z = x + (y + z)$
Штрих Шеффера			Стрелка Пирса	
$0/0 = 1$	$1/x = \bar{x}$	$x/(y/z) \neq (x/y)/z$	$0 \downarrow 0 = 1$	$1 \downarrow x = 0$
			$0 \downarrow 1 = 0$	$x \downarrow x = \bar{x}$
$0/1 = 1$	$x/x = \bar{x}$	$x/y = y/x$	$1 \downarrow 1 = 0$	$x \downarrow \bar{x} = 0$
			$0 \downarrow x = \bar{x}$	$x \downarrow y = y \downarrow x$
$1/1 = 0$	$x / x = 1$	$0/x = 1$	$x \downarrow (y \downarrow z) \neq (x \downarrow y) \downarrow z$	

Правила де Моргана:

$$\overline{x_1 * x_2 * \dots * x_n} = \bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n$$

$$\overline{x_1 + x_2 + \dots + x_n} = \bar{x}_1 * \bar{x}_2 * \dots * \bar{x}_n$$

Используя метод непосредственных преобразований, можно получить различные варианты схем, в соответствии с теми критериями, исходя из которых, разработчик определяет оптимальность схемы.

Пример.

$$y = \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + x_1 \cdot x_2 + x_2 \cdot x_3.$$

Реализация указанной логической функции на основе метода непосредственных преобразований дают нам три варианта (при желании, можно придумать и больше!) решения задачи.

При выборе конкретного варианта необходимо исходить из того либо иного критерия, по которому данный вариант решения является более предпочтительным перед другими

При разработке схем на основе конкретной элементной базы, количество оборудования обычно измеряется **количеством корпусов интегральных микросхем, используемых в схеме.**

В теоретических разработках, ориентируются на произвольную элементную базу, а поэтому для оценки затрат оборудования используется оценка сложности схем по Квайну.

Сложность (оценка, цена) по Квайну определяется суммарным количеством входов логических элементов в составе схемы.

При данном способе оценки, единицей сложности является один вход логического элемента.

Сложность инверсного входа обычно принимается равной двум (если на вход схемы сигналы поступают только в прямой форме).

Указанный подход к оценке сложности схем целесообразен по следующим причинам:

- 1) сложность схемы легко вычисляется по логическим функциям, на основе которых строится схема, например, для ДНФ сложность схемы равна сумме количества букв (при том, что букве со знаком отрицания соответствует сложность два) и количества знаков дизъюнкции (увеличивается на единицу для каждого дизъюнктивного выражения);
- 2) все классические методы минимизации логических функций обеспечивают минимальность схемы именно в смысле оценки по Квайну;
- 3) практика показывает, что схема с минимальной оценкой по Квайну обычно реализуется наименьшим количеством конструктивных элементов (корпусов интегральных микросхем).

В качестве критерия можно использовать и быстродействие схемы.

Быстродействие комбинационной схемы оценивается максимальной задержкой сигнала при прохождении сигнала от входа схемы к ее выходу.

То есть быстродействие комбинационной схемы определяется промежутком времени от момента поступления входных сигналов до момента установления соответствующих значений выходных сигналов.

Задержка сигнала кратна числу элементов, через которые проходит сигнал от входа к выходу схемы. Поэтому быстродействие схемы характеризуется значением $r \cdot t$, где: t является задержкой сигнала на одном элементе; значение r определяется количеством уровней комбинационной схемы, которое рассчитывается приведенным ниже образом.

Входам комбинационной схемы приписывается **нулевой** уровень.

Логические элементы, связанные только с входами схемы, относятся к **первому** уровню.

Элемент относится к уровню **k**, если он связан по входам с элементами уровней **k-1**, **k-2**, и т.д.

Максимальный уровень элементов **r** определяет количество уровней комбинационной схемы, называемое **рангом схемы**.

Также в качестве критериев можно применять: стоимость схемы, сложность ее изготовления или обслуживания (настройки, ремонта), надежность, а также другие соображения.

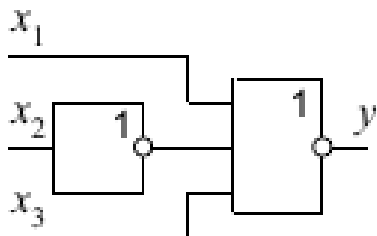


Рис. 3. Реализация логической функции.

Данный вариант позволяет реализовать функцию, используя всего 2 логических элемента, инвертор и «3 ИЛИ-НЕ». Этот вариант можно считать оптимальным по двум «классическим учебным» критериям. Цена по Квайну

– минимальна (равна 6), задержка – то же (2 задержки элемента). Но вот с точки зрения практической реализации нам понадобятся две разные интегральные схемы. Причем, если инвертор – вещь простая и распространенная (да еще и 5 «запасных» инверторов в корпусе останется!), то «3 ИЛИ-НЕ», особенно для ТТЛ микросхем – достаточно «экзотична».

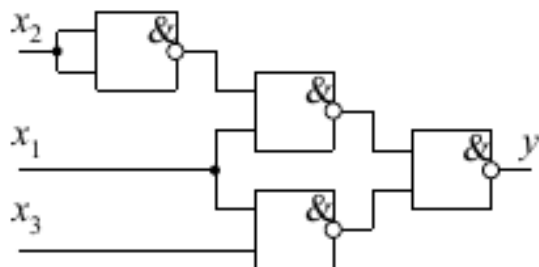


Рис. 4. Реализация логической функции в базисе «И-НЕ».

Второй вариант использует только элементы «2 И-НЕ». Их уже 4, а не 2, как в прошлом варианте. Задержка тоже выше – равна трем «элементарным». Цена по Квайну – 8.

Но, с учетом того, что элемент «2 И-НЕ» является базовым для ТТЛ логики, мы получим два преимущества:

1. Микросхема типа 155 (555, 1555...) ЛА3, пожалуй, наиболее распространена в ТТЛ логике, следовательно, ее проще найти и она обойдется дешевле. Для примера, на момент написания 155 ЛА 3 на «чипе дипе» стоит 34 рубля, 155 ЛЕ4 («3 ИЛИ-НЕ») там просто нет, а на другом сайте она стоит уже 60 рублей...
2. В одном корпусе 555 ЛА 3 – 4 элемента «2 И-НЕ», т.е. в данном случае нам понадобится не 2 корпуса ИС, а один, что опять скажется на цене (и габаритах!) нашего устройства.

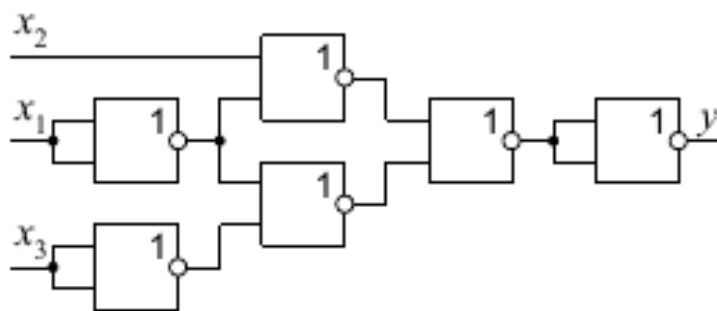


Рис. 5. Реализация логической функции в базисе «ИЛИ-НЕ».

Третий вариант – только на элементах «2 ИЛИ-НЕ».

Цена по Квайну – 12. Задержка - 4. Количество корпусов ИС – 2. Все плохо, плохо...

Преимущество: именно элемент «2 ИЛИ-НЕ» является базовым для технологии КМОП ИС (серия 561, 564 и т.д.).

Метод непосредственных преобразований применяется в том случае, если логическая функция задана небольшим количеством аргументов, обычно 2-3 аргумента. Если число аргументов больше, то этот становится трудоемким. Поэтому применяют такие методы минимизации, как метод Квайна, Квайна – Мак-Класки, Петрика, карт Карно и другие.

В качестве примера напомним о применении карт Карно.

Пример.

Пусть задана логическая функция четырех переменных

$$P=f(X_4, X_3, X_2, X_1)$$

Функция записывается в виде

$$P \{ 0, 1, 4, 6, 7, 8, 9, 10, 14, 15 \}.$$

Данная запись обозначает, что функция P равна единице при указанных значениях десятичных эквивалентов логических аргументов. В остальных случаях функция равна нулю. Таблица истинности данной функции приведена в табл.7.

Таблица 7.

ДЗ	X ₄	X ₃	X ₂	X ₁	P
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

После подстановки единичных значений функции в карту можно выделить четыре контура, изображенных разными цветами и цифрами, (рис.11).

		X ₂ X ₁			
X ₄ X ₃		X ₂ X ₁			
		00	01	11	10
	00	1	1		1
	01	1			1
	11		1	1	
	10		1	1	1

Рис. 11. Карта Карно для таблицы истинности 6.

Контуры I и II содержат по четыре единицы, а контуры III и IV – по две. Запишем конъюнкцию для первого контура:

$$k_1 = X_4 * X_3 * X_2 * X_1$$

Аргументы X_2 и X_3 в этом контуре принимают значения 0 и 1, поэтому их можно исключить из выражения. Аргументы X_4 и X_1 принимают значение 1, и их записывают без инверсии. Если аргумент в контуре равен нулю, то его записывают с инверсией. В итоге конъюнкция для первого контура запишется в виде

$$k_1 = X_4 * X_1$$

Аналогично записываем конъюнкции для остальных контуров

$$k_2 = \overline{X_4} * \overline{X_1};$$

$$k_3 = \overline{X_4} * \overline{X_3} * \overline{X_2};$$

$$k_4 = \overline{X_3} * X_2 * \overline{X_1}$$

Минимальную функцию получаем дизъюнкцией полученных значений конъюнкций для каждого контура

$$P_{min} = X_4 * X_1 + \overline{X_4} * \overline{X_1} + \overline{X_4} * \overline{X_3} * \overline{X_2} + \overline{X_3} * X_2 * \overline{X_1}$$

Функциональная схема устройства, соответствующая минимальной функции приведена на рис.12.

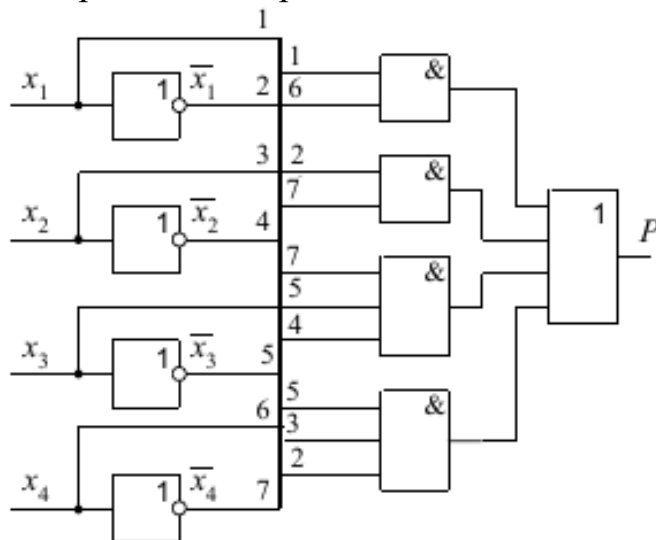


Рис. 12. Реализация таблицы истинности 6.

Следующим этапом разрабатывается принципиальная схема с минимальным числом элементов или на указанной элементной базе, например на элементах «И-НЕ». При этом выполняются преобразования логического уравнения с использованием формул де Моргана.

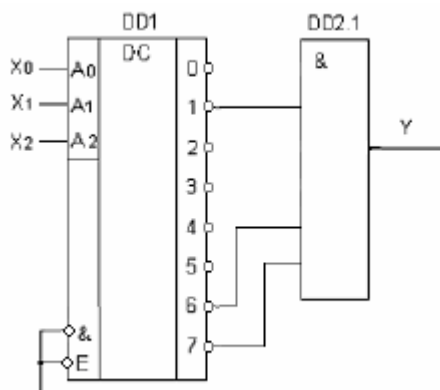
Логические элементы отнюдь не единственные цифровые схемы, при помощи которых могут быть реализованы те или иные комбинационные схемы. Оставим за рамками данного пособия различные типы программируемых логических схем (ПЛИС). Эти устройства не относятся к простейшим цифровым узлам, да и для реализации «чисто комбинационных» устройств их применяют нечасто.

А вот более сложные интегральные схемы, такие, как мультиплексоры, дешифраторы, вполне могут быть использованы для построения достаточно

Дешифраторы

Дешифратором называется цифровая комбинационная схема с несколькими входами и выходами, преобразующая код, подаваемый на входы в сигнал на одном из выходов. На выходе дешифратора всегда присутствует только один сигнал, причем номер этого сигнала однозначно определяется входным кодом. Если дешифратор, имеющий n информационных входов, имеет 2^n выходов, то такой дешифратор называется полным. Если выходов меньше – дешифратор неполный. Условное графическое изображение (УГО) дешифратора на принципиальных схемах сопровождается буквами DC (от английского Decoder).

Связано это с количеством контактов ИС. Если для дешифраторов 2-4 и 3-8 вполне подходят обычные 14-ти и 16-контактные корпуса, то для реализации дешифратора 4-16 нужно уже 24 «ноги» в корпусе. А это уже – приличные габариты. Дальнейшее наращивание разрядности приведет к крупногабаритным микросхемам с единственной реализуемой функцией. Что не есть хорошо! Проще уже переходить к микроконтроллерам...



Все достаточно очевидно, входные величины подаются на адресные входы дешифратора. Выходы дешифратора, соответствующие единице на выходе заданной логической функции (согласно ее таблице истинности) объединяются элементом И. Возможны самые разные варианты схемотехники – для прямых и инверсных выходов дешифратора, для объединения «нулей» или «единиц» функции, с элементами И или ИЛИ на выходе...

Мультиплексоры

16

на УГО микросхемы мультиплексора – MUX (достаточно просто догадаться – почему). Впрочем, иногда встречается и обозначение MS.

Приведем два примера мультиплексоров.

Микросхема 561 (564) КП 2. 8-канальный мультиплексор /демультиплексор, предназначенный для переключения цифровых и аналоговых сигналов. Состоит (функционально) из дешифратора и 8 ключей. Управление осуществляется 3-разрядным двоичным кодом. Причем микросхема – универсальна! Двоичный код просто определяет, какой из выводов X подключен ключом к выводу Y. Направление информационного сигнала может быть любым! Разработчик аппаратуры получает широчайшие возможности, используя эту микросхему.

Ну, а в качестве «классического» мультиплексора, приведем микросхему К555КП15, мультиплексор цифровой, 8-входовой, технология ТТЛШ.

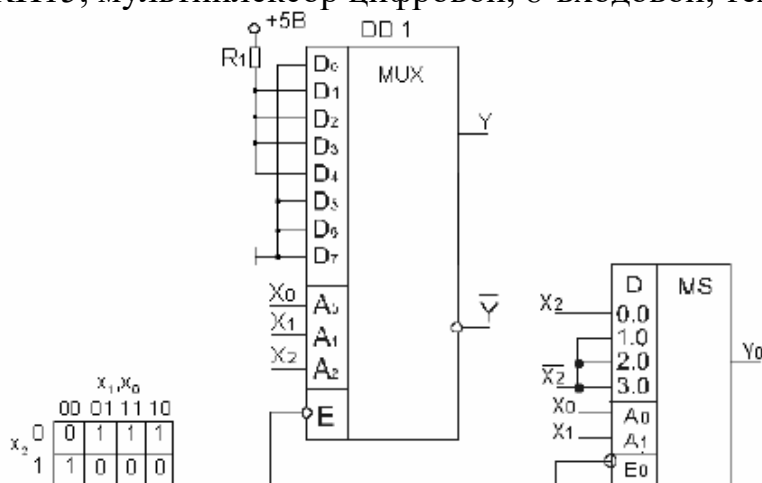


Рис. 14. Реализация логической функции на мультиплексоре

В качестве примеров практической реализации цифровых комбинационных устройств, рассмотрим синтез булевой функции на логических элементах, а также на дешифраторах и мультиплексорах.

Оба задания выполняются обучающимся по индивидуальному варианту входных переменных, выбираемых в соответствии с правилом, описанным ниже.

Необходимо синтезировать булеву функцию четырех входных переменных (X_1, X_2, X_3, X_4), причем обязательно в составе выходной функции Y должны быть не менее 4 единичных и 4 нулевых значений. Остальные значения выходной функции – произвольны, т.е. могут быть нулевыми, либо единичными, по усмотрению разработчика.

Для выбора единичных значений возьмем фамилию обучающегося, выполняющего задания, а для нулевых значений – его имя. Для примера АНДРЕЕВ ИГОРЬ.

За единичные значения примем двоичные коды, соответствующие порядковым номерам (в русском алфавите) первых букв фамилии. Следует учесть, что двоичная кодировка начинается с нуля, т.е. двоичный код для буквы «А» будет равен 0000. Также учтем, что в русском языке в именах и

фамилиях практически не применяются буквы «Ё» и «Ъ». В крайнем случае, их можно заменить на «Е» и «Ь» без смысловых потерь.

Старший разряд кода считаем входной переменной X_4 , младший – X_1 .

Исходя из приведенных соображений получим $Y=1$ для $X_4X_3X_2X_1$:

А – 0 – 0000

Н – 13 – 1101

Д – 4 – 0100

Р – 16 – 10000 – игнорируем, т.к. $X_4X_3X_2X_1$ совпадает с кодом буквы «А»

Е – 5 – 0101

По аналогии, составим коды для четырех обязательных нулевых значений:

И – 8 – 1000

Г – 3 – 0011

О – 14 – 1110

Р – 16 – игнорируем, т.к. коды для «нулей» и «единиц» совпадать не могут.

Ь – 26 – 11010, т.е. 1010.

В случае коротких имен и фамилий, либо большого количества повторяющихся букв, можно взять любое другое «кодовое слово» - место рождения, кличку домашнего любимца... Запрещено лишь брать 4 кода подряд, облегчая себе задачу...

2. СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ НА ОСНОВЕ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

2.1. Цель работы

1.1. Синтез комбинационной схемы (КС): построение схемы для заданной булевой функции с помощью стандартных логических элементов в соответствии с вариантом индивидуального задания.

1.2. Изучение работы программы схемотехнического моделирования МС-11. (или другие версии).

2.2. Задание на теоретическую часть

Построить комбинационные схемы, реализующие определенную булеву функцию, которая принимает значение 0 или 1 при заданном условии.

Необходимо выполнить следующие этапы:

1. Выбрать вариант задания и составить булеву функцию в соответствии с ним.

Например: Для пробного варианта, приведенного выше, функция будет выглядеть следующим образом (СДНФ):

$$Y = \overline{X_4} * \overline{X_3} * \overline{X_2} * \overline{X_1} + X_4 * X_3 * \overline{X_2} * X_1 + \overline{X_4} * X_3 * \overline{X_2} * \overline{X_1} + \overline{X_4} * X_3 * \overline{X_2} * X_1$$

2. Составить таблицу истинности функционирования КС по заданной булевой функции.

3. Произвести построение схемы на логических элементах в соответствии с заданным вариантом (по заданной логической функции составить комбинационную схему, применив СКНФ или СДНФ). Обучающийся имеет право выбрать форму ДНФ или КНФ, исходя из удобства реализации варианта задания.

4. Произвести оптимизацию КС, минимизировать функцию. При составлении карты Карно, обучающийся имеет право присвоить выходной функции Y , не определенной в соответствии с вариантом задания, значение, равное 0 или 1 по своему усмотрению. Выбор конкретных значений должен определяться из соображений максимального упрощения схемы, полученной в результате оптимизации исходной булевой функции.

5. Пример минимизации Булевой функции для приведенного выше пробного варианта показан на рис. 15. Полужирным курсивом выделены «нули» и «единицы» функции, обязательные в соответствии с заданием. При этом добавление всего двух единичных значений (ячейки выделены фоновым цветом) позволяет значительно упростить вид функции, по сравнению с исходной СДНФ:

$$Y = X3 * \overline{X2} + \overline{X4} * \overline{X2}$$

		X2X1		X1	X2X1	X2
X4X3		00	01	11	10	
	00	1	1	0		
X3	01	1	1			
X4X3	11	1	1		0	
X4	10	0			0	

Рис. 15. Оптимизированная Булева функция

6. Синтезировать соответствующую предыдущему пункту схему.

7. Реализовать логическую функцию, используя двухвходовые элементы базиса «И-НЕ» или «ИЛИ-НЕ», согласно заданному преподавателем варианту.

2.3. Задание на моделирующую часть.

1. Построить схему на определенных логических элементах по заданной логической функции, установив нужные параметры.

2. Провести исследование всех синтезированных схем в режиме Transient или в режиме Dynamic DC (по выбору обучающихся), составить таблицы истинности и сравнить их с исходными.

2.4. Методические указания по моделирующей части работы

1. Загрузить программу схемотехнического моделирования МС 11 (либо другой версии ПО) двойным щелчком по ее пиктограмме; при этом на

экране появляется рабочее окно главного меню (рис. 16).

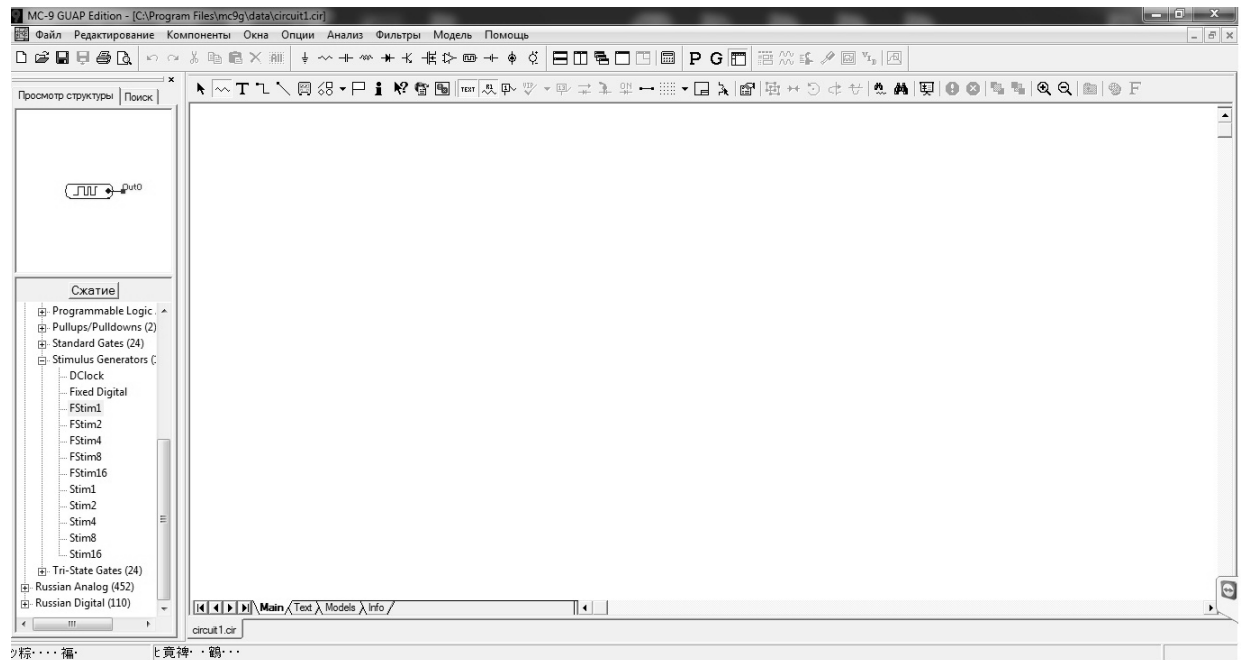


Рис. 16. Рабочее окно программы.

2. В нем открываем новый схемный файл (SCHEMATIC) с расширением .cir, где будем создавать чертеж исследуемой схемы.

3. Исследование схемы можно производить, как в статическом, так и в динамическом режиме.

4. При исследовании в статическом режиме, на каждый вход схемы подаются сигналы логического 0 или 1 с помощью анимированного логического ключа (Component – Animation – Animated Digital Switch).

5. Состояние выходного сигнала фиксируется пробником-светодиодом (Component – Animation – Animated Digital LED). Такие же светодиоды могут устанавливаться в любом месте схемы, если необходимо определение логического уровня сигнала в данной точке.

6. Пример схемы с использованием упомянутых выше элементов показан на рис. 17. Режим исследования – *Dynamic DC*.

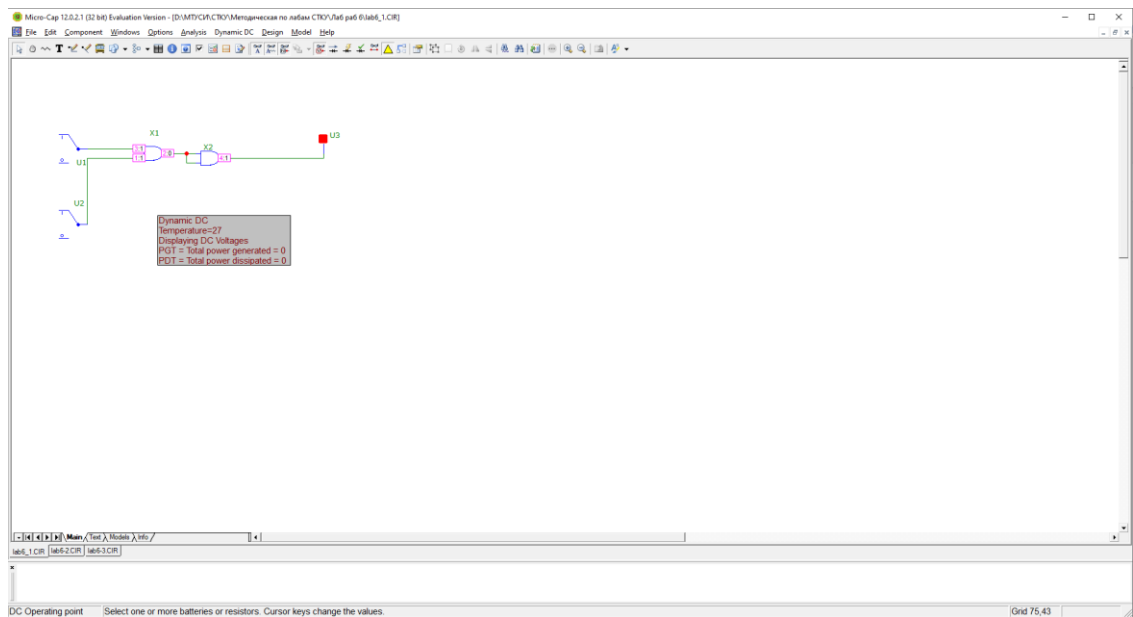


Рис. 17. Логическая схема в режиме *Dynamic DC*.

7. Меняя кликом состояние ключей, необходимо перебрать все варианты, возможные для комбинации входных сигналов, и составить таблицу истинности исследуемой схемы.

8. Исследование схемы в динамическом режиме производится при включении режима *Transient*. В этом случае, для анализа схемы необходимо использовать генераторы сигналов, которые автоматически генерируют последовательности импульсов в соответствии с установленными свойствами и программой. Они расположены в меню *Component - Digital Primitives - Stimulus Generators*.

9. Пример схемы, реализующей минимизированную Булеву функцию приведенного выше пробного варианта, показан на рис. 18. Режим исследования – *Transient*.

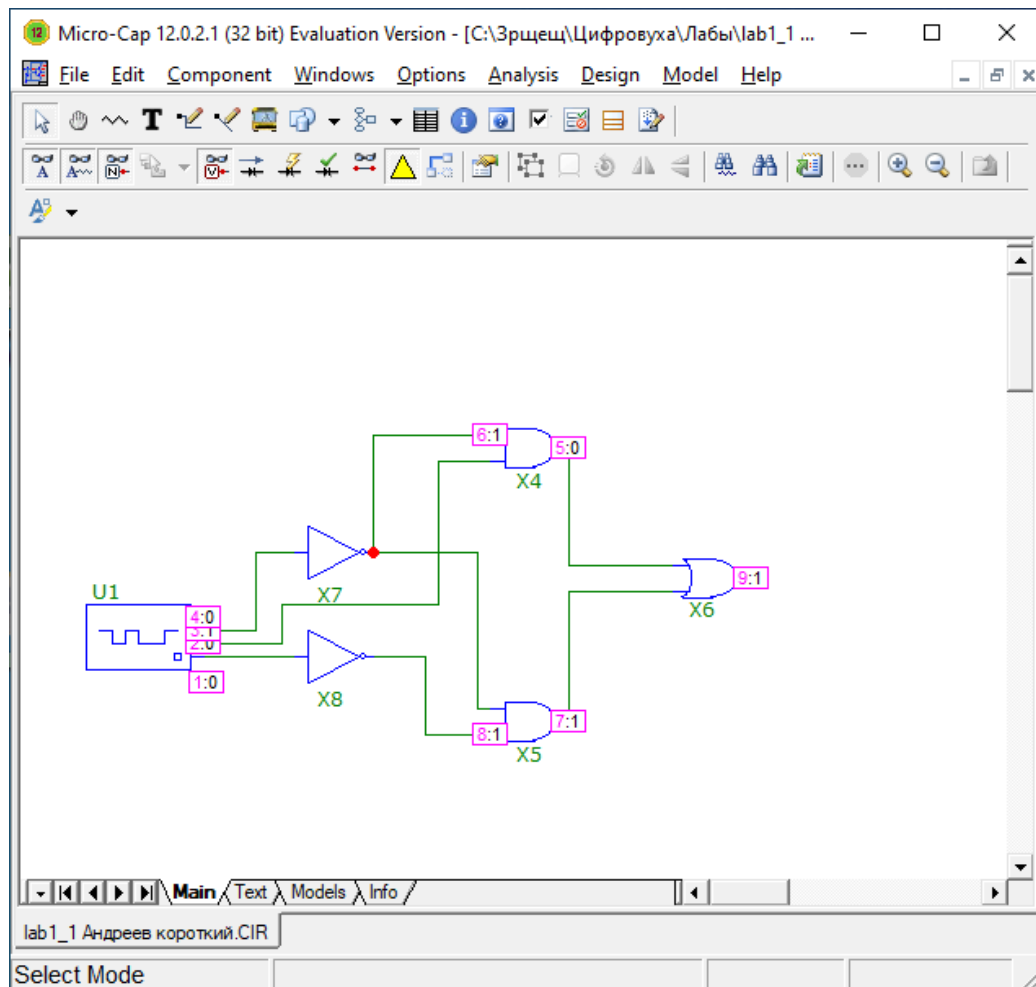


Рис. 18. Исследование схемы с использованием генератора импульсов.

Для генерации различных последовательностей сигналов (например, чтобы автоматически перебирать все возможные комбинации входов $x1$, $x2$, $x3$, $x4$ исследуемой схемы) используются генераторы дискретных последовательностей *Stim1*, *Stim2*, *Stim4*, *Stim8*, *Stim16*. Они имеют разное количество выходов, сигналы на которых можно запрограммировать в соответствии с требованиями. Программирование проводится в окне свойств генераторов (рис. 19).

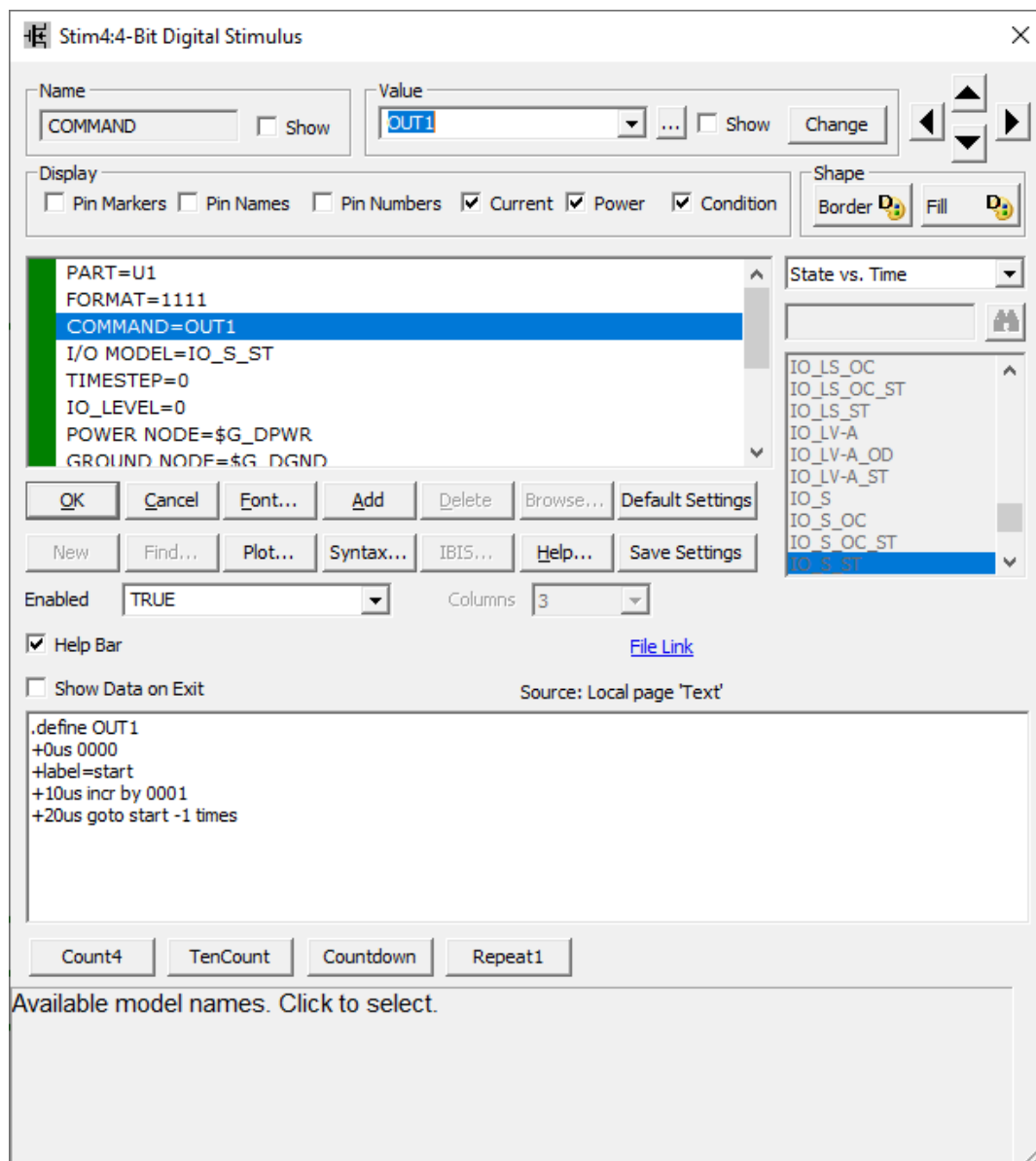


Рис. 19. Настройка генератора импульсных последовательностей.

Свойство *COMMAND* задает последовательности выходных сигналов генератора, а свойство *FORMAT* – формат представления чисел в этих командах. Значение *FORMAT* содержит цифры 1, 3 или 4 и их комбинации, где 1 означает двоичный формат числового представления числа, 3 – восьмеричный, а 4 – шестнадцатеричный. Сумма цифр в значении *FORMAT* должна быть равна числу выходов генератора. Например, 1111 – четыре выхода генератора задаются в двоичном формате, 44 – восемь выходов генератора задаются в шестнадцатеричном формате.

Для задания последовательности выходных сигналов выбираем свойство *COMMAND* и задаем идентификатор последовательности в поле Value, например *OUT1*. В текстовом окне пишется сама последовательность после фразы «.define *OUT1*». Команды последовательности начинаются с символа «+», после которого пишется время относительно начала последовательности. Самый простой вариант команды – непосредственное задание выходного значения сигналов генератора в формате, определяемом свойством *FORMAT*.

Для выполнения данной лабораторной работы рекомендуется использовать задание цикла работы генератора с 4 выводами (хотя можно использовать любой, удобный разработчику способ, возможности управления режимами генераторов весьма широки!). Для этого используются метки, формат объявления метки:

+*label=name*, где *name* – имя метки

Команда возврата на метку – *goto*, ее формат:

+*t goto name x times*, где *t* – время, а *x* – количество повторов. Если *x* = -1, зацикливание повторяется бесконечное число раз, например:

.define OUT1

+0ms 0000

+label=start

+10ms incr by 0001

+20ms goto start -1 timesB

В этом случае выходные сигналы генератора будут выглядеть в соответствии с рис. 20.

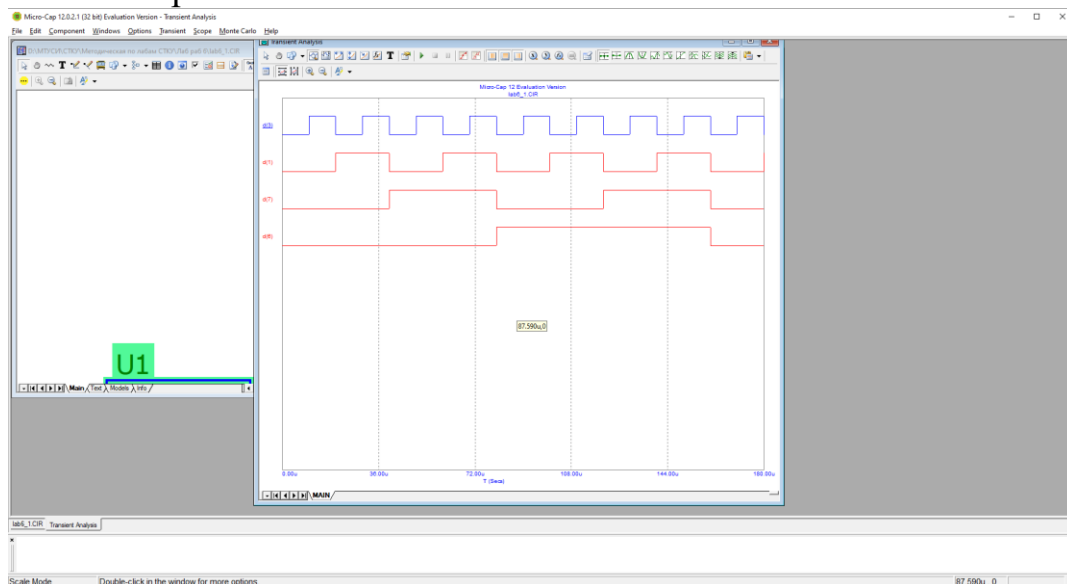


Рис. 20. Выходные сигналы генератора.

Таким образом, одновременно (точнее – последовательно во времени) можно наблюдать все возможные состояния схемы.

10. Для наблюдения выходного сигнала, а также (в случае необходимости) сигнала в любой точке схемы, достаточно включить нужный узел в состав наблюдаемых сигналов. Следует помнить, что цифровые сигналы обозначаются не как привычные для аналоговой техники «U(i), а как «d(i)», с указанием корректного обозначения узла...

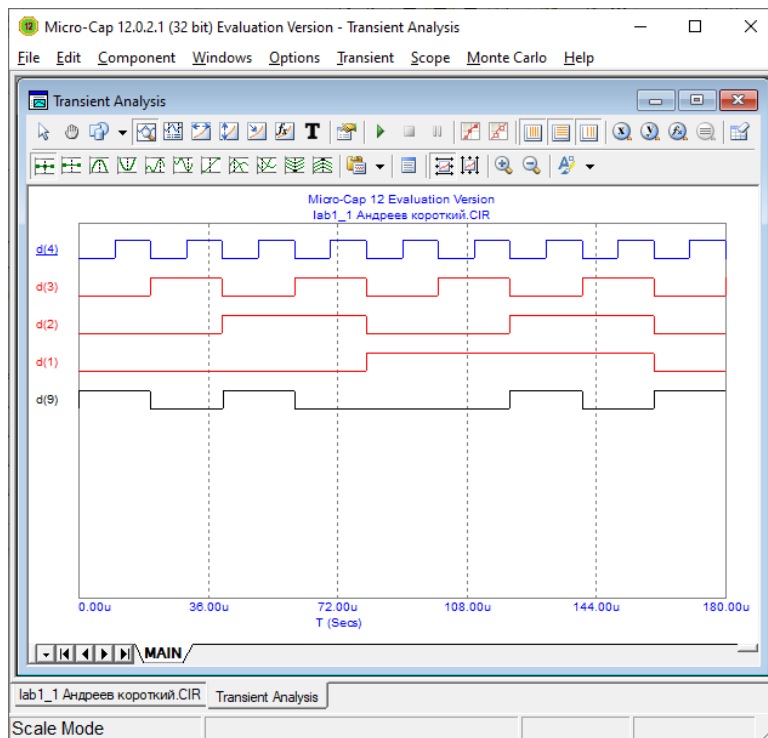


Рис. 21. Реализация оптимизированной функции в динамическом режиме.

11. На рис. 21 показан результат моделирования в динамическом режиме для оптимизированной Булевой функции пробного варианта. По выходному сигналу можно определить, что все обязательные «единицы» и «нули» варианта реализованы. Т.е. задание выполнено.

12. По временной диаграмме входных и выходных сигналов также составляется таблица истинности, для ее сверки с исходной. Совпадение таблиц истинности (для обязательных исходных данных) доказывает правильность синтеза схемы.

2.5. Контрольные вопросы.

1. На основе чего строятся комбинационные цифровые схемы?
2. Какие две постановки задачи синтеза комбинационных схем?
3. Каковы показатели качества синтезируемой схемы?

2.6. Содержание отчета.

1. Титульный лист;
2. Цель работы;
3. Вариант задания на работу;
4. Таблица истинности заданной булевой функции;
5. 3 схемы на определенных логических элементах (СКНФ или СДНФ, оптимизированная схема, и оптимизированная схема, реализованная на базовых элементах 2 «И-НЕ» или 2 «ИЛИ-НЕ»;
6. Таблицы истинности для каждой схемы;
7. Временные диаграммы сигналов (при наличии, при исследовании схемы в динамике). При исследовании схем в статическом режиме необходимы копии схем при всех «рабочих» заданных значениях входных

величин (т.е., при использовании СДНФ необходимо показать реализацию всех 4 значений «выходной единицы»;

8. Выводы.

3. СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ НА ОСНОВЕ ДЕШИФРАТОРОВ И МУЛЬТИПЛЕКСОРОВ

3.1. Цель работы

1. Синтез комбинационной схемы (КС): построение схемы для заданной булевой функции с помощью дешифраторов и мультиплексоров соответствии с вариантом индивидуального задания.

2. Изучение работы программы схемотехнического моделирования МС-11. (или другие версии).

3.2. Задание на теоретическую

Построить комбинационные схемы, реализующие определенную булеву функцию, которая принимает значение 0 или 1 при заданном условии.

Необходимо выполнить следующие этапы:

1. В качестве исходных данных принять вариант предыдущего задания.

2. Составить таблицу истинности функционирования КС по заданной булевой функции.

3. Синтезировать схему на основе дешифратора и мультиплексора в соответствии с заданным вариантом (по заданной логической функции составить комбинационную схему).

4. Синтезировать схему преобразователя кодов на основе дешифратора, соответствующую варианту задания.

3.3. Задание на экспериментальную часть.

1. Открыть либо синтезировать модель комбинационной схемы на основе дешифратора и мультиплексора.

2. Отладить схему таким образом, чтобы реализовать заданную логическую функцию.

3. Открыть модель преобразователя кодов на дешифраторе.

4. Отредактировать схему, реализовав полученное задание.

3.4. Методические указания по моделирующей части работы

1. Загрузить программу схемотехнического моделирования МС 9 (11) двойным щелчком по ее пиктограмме; при этом на экране появляется рабочее окно главного меню (рис. 22).

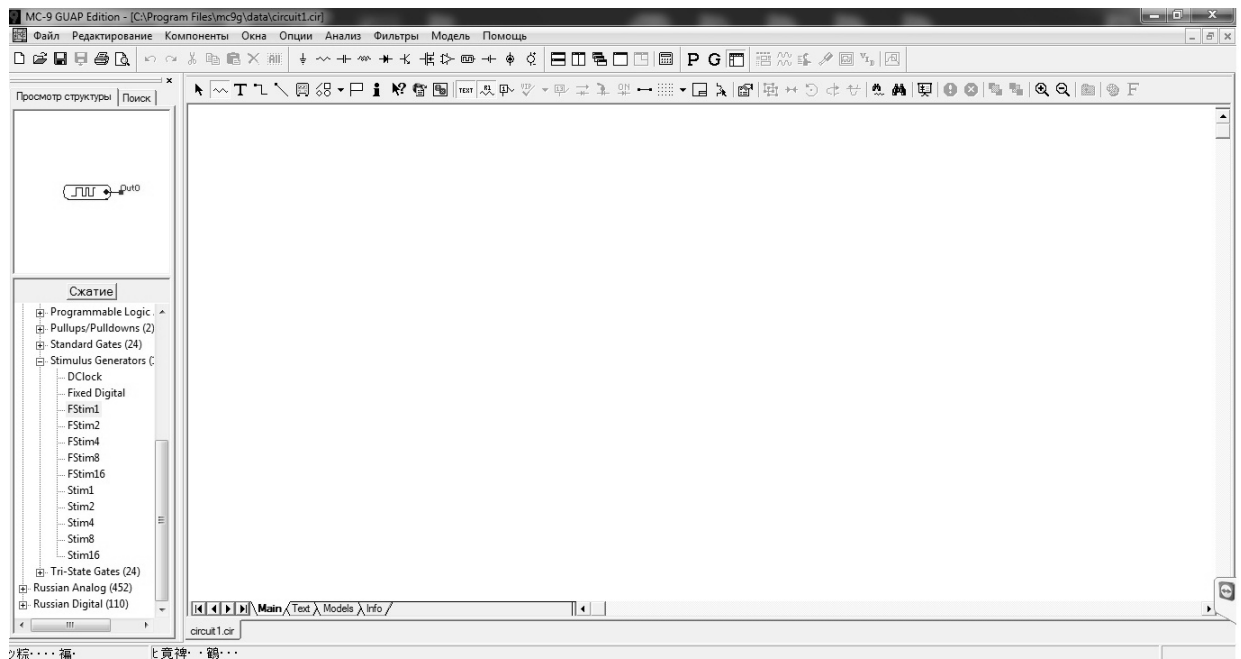


Рис. 22. Рабочее окно программы.

2. В меню *FILE* открываем новый схемный файл lab2.1 с расширением .cir, где содержится чертеж исследуемой схемы.

3. Исследование схемы проводится, в статическом режиме.

4. При исследовании в статическом режиме, на каждый вход схемы подаются сигналы логического 0 или 1 с помощью анимированного логического ключа (*Component – Animation – Animated Digital Switch*).

5. Состояние выходного сигнала фиксируется пробником-светодиодом (*Component – Animation – Animated Digital LED*). Такие же светодиоды могут устанавливаться в любом месте схемы, если необходимо определение логического уровня сигнала в данной точке.

6. Пример схемы с использованием упомянутых выше элементов показан на рис. 23. Режим исследование – *Dynamic DC*.

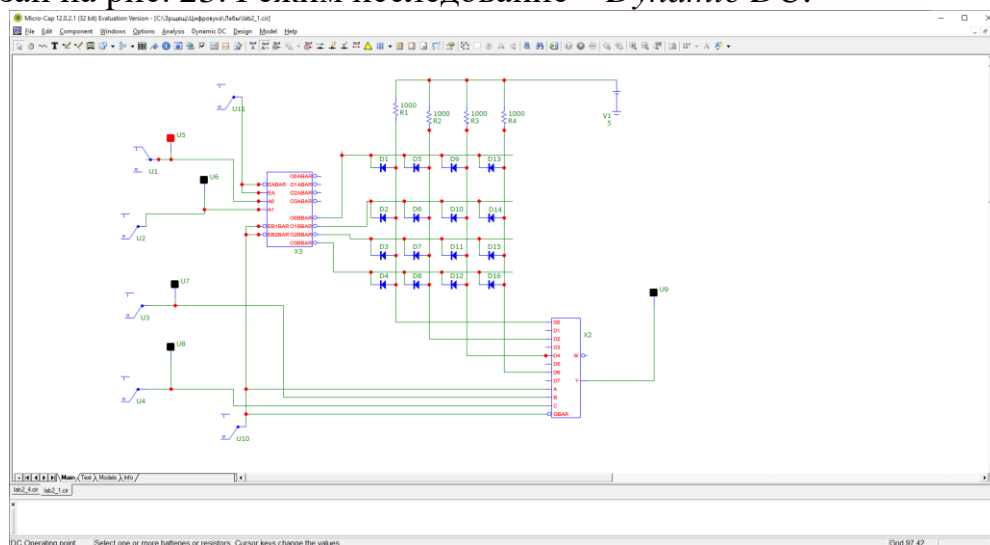


Рис. 23. Логическая схема в режиме *Dynamic DC*.

а. Редактирование (отладка) схемы заключается в изменении структуры матрицы связей выходов дешифратора со входами

мультиплексора таким образом, чтобы схема реализовала логическую функцию в соответствии с заданным вариантом.

б. При сохранении связи в узле матрицы (диод соединяет строку и столбец матрицы), значение выходной функции для набора входных величин, соответствующих адресу строки и столбца, будет равно логическому нулю.

9. При устранении связи строки и столбца (обрыв диода или его удаление), выходная функция принимает значение логической единицы.

10. Меняя кликом состояние ключей, необходимо перебрать все варианты, возможные для комбинации входных сигналов, и составить таблицу истинности исследуемой схемы.

11. Совпадение таблиц истинности (полученной при отладке схемы и заданной) доказывает правильность синтеза схемы.

12. В меню *FILE* открываем новый схемный файл lab2.4probe с расширением .cir, где содержится чертеж исследуемой схемы. Указанная модель реализует пробный вариант задания, приведенный ранее.

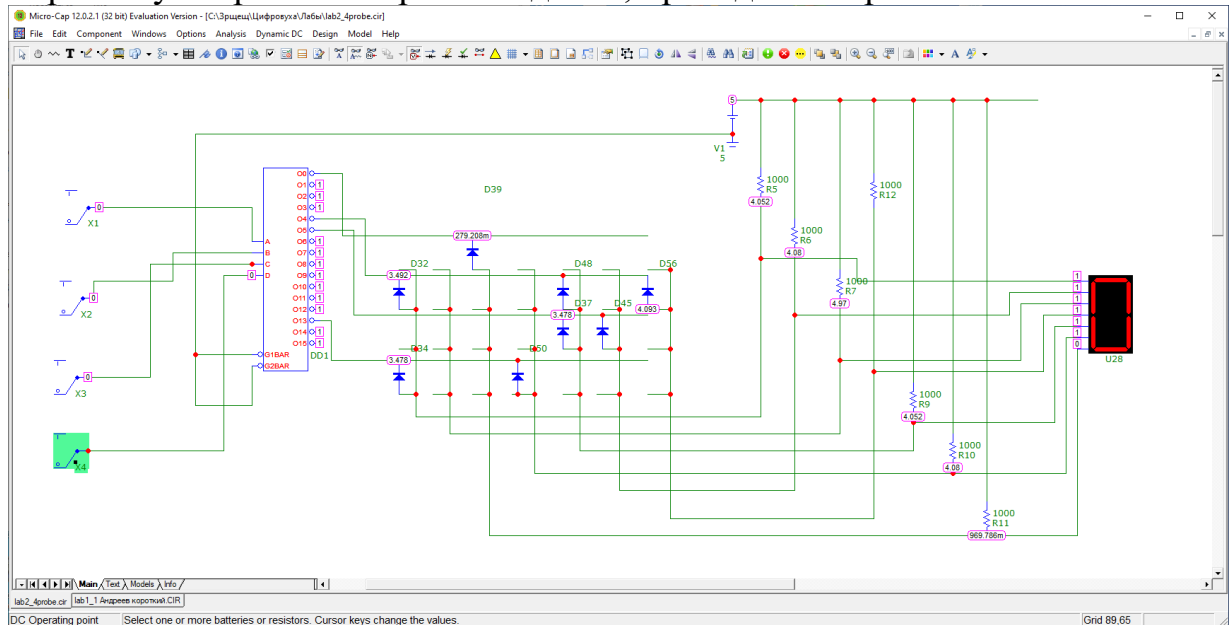


Рис. 24. Схема преобразователя кодов в режиме *Dynamic DC*.

13. Схема, показанная на рис. 24, является фрагментом преобразователя входного 4-разрядного двоичного кода в код управления семисегментным светодиодным индикатором.

14. Упрощение схемы (и ее неполная функциональность, доступно отображение только 4 комбинаций входного кода) связано с ограничениями студенческой версии программы MicroCap (не более 50 элементов) и чисто учебными задачами модели. В профессиональной версии ПО может быть реализовано преобразование входного кода В ЛЮБОЙ выходной код.

15. Данная схема во многом близка как к структурам постоянных запоминающих устройств (ПЗУ, ППЗУ, РППЗУ), так и к структурам простейших ПЛИС, РЕАЛЬНО используемых для реализации как комбинационных, так и более сложных схем.

16. Исследование схемы производится также в статическом режиме, как это было описано для предыдущей схемы.

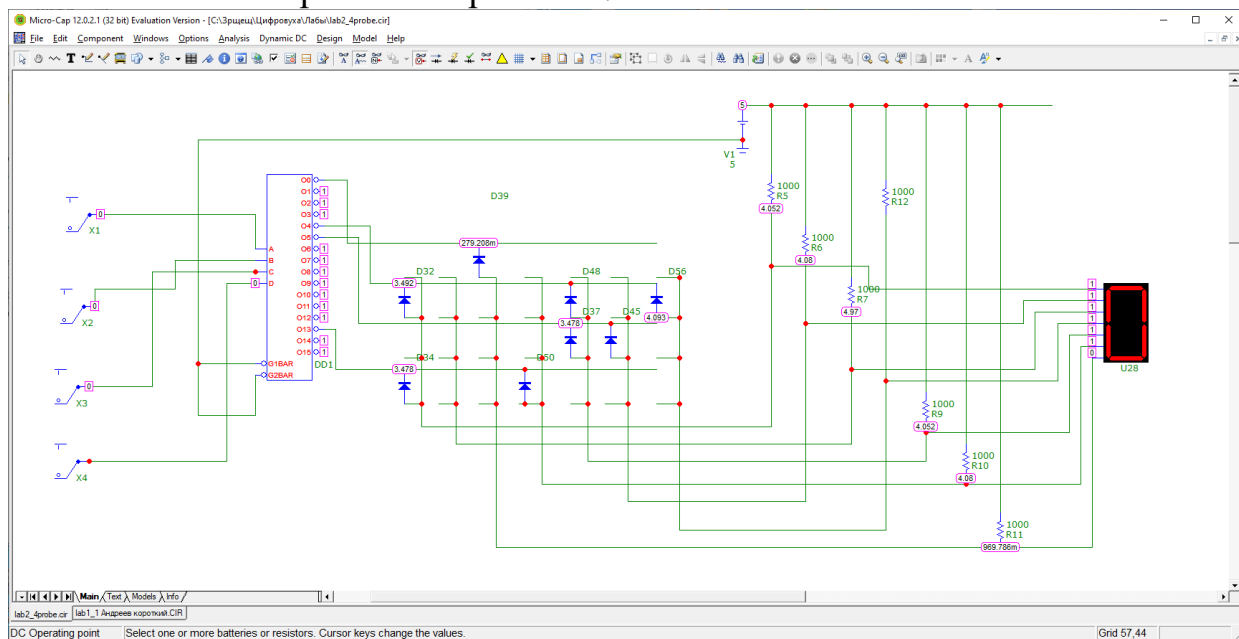
17. В качестве первого шага моделирования обучающийся должен выбрать те выходы дешифратора DD1, которые будут активны при актуальном варианте задания. В пробном варианте – это коды «0», «4», «5» и «13». Семисегментный индикатор при этом показывает символы «0», «4», «5» и «d» («13» в шестнадцатичном коде).

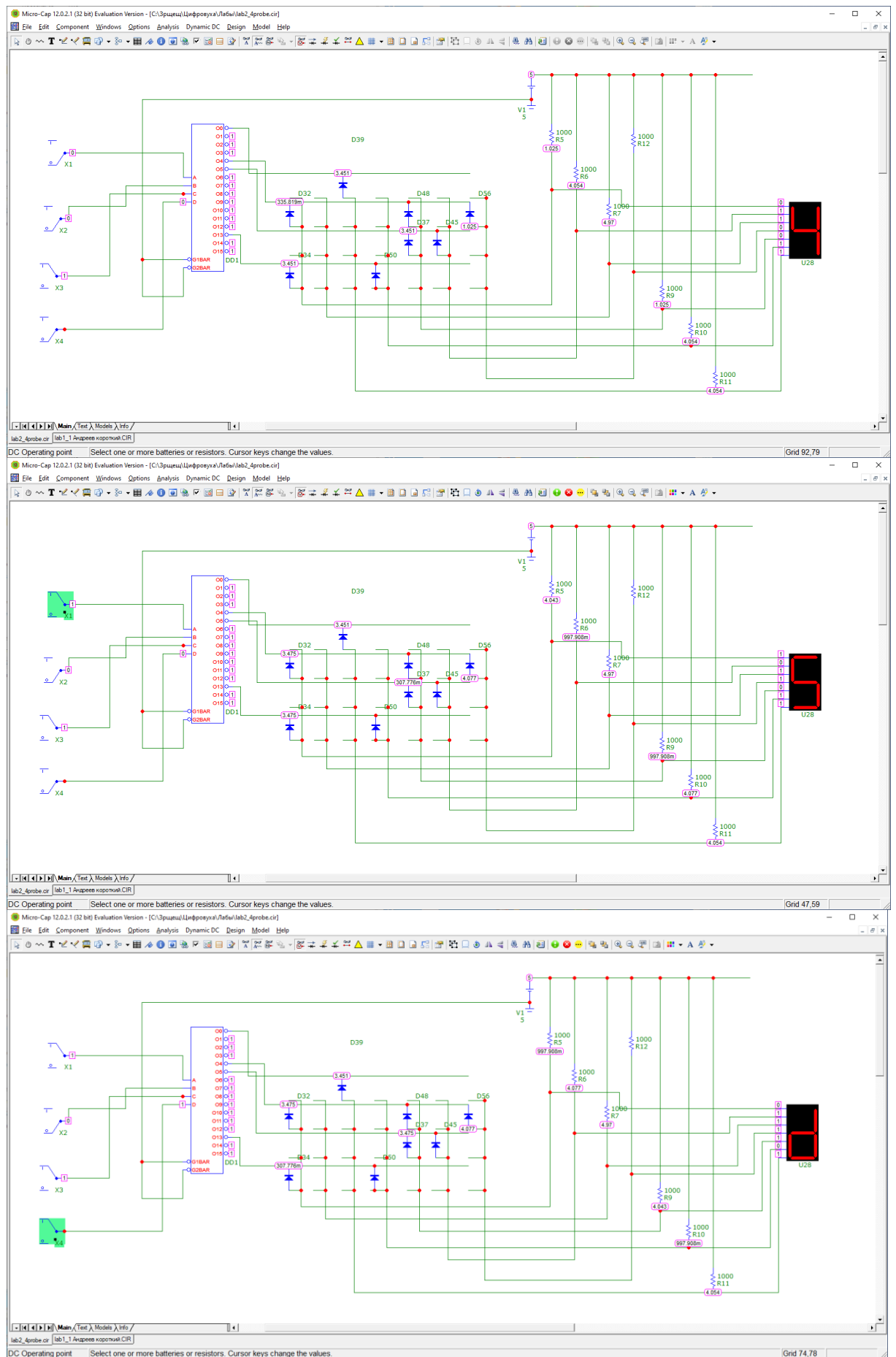
18. Меняя соединения матрицы диодов, необходимо реализовать заданную для каждого варианта систему выходных логических функций, т.е. преобразовать набор кодовых символов входного кода в набор символов выходного кода.

19. В ряде вариантов целесообразно заменить семисегментный индикатор на набор светодиодов. Необходимо лишь помнить, что при любом изменении схемы, число элементов не должно превышать 50, иначе программа немедленно выдает сообщение об ошибке.

20. Выходные коды фиксируются и выносятся в отчет не только в виде таблицы истинности, но и в виде скриншотов, демонстрирующих заданные варианты входных и выходных кодов.

21. На последующих 4 рисунках показаны варианты реализации отображения на семисегментном индикаторе состояний входного двоичного кода в соответствии с пробным вариантом.





22. При реализации актуальных вариантов заданий, коды от «10» до

«15» должны отображаться символами «A», «b», «C», «d», «E» и «F».

3.5. Содержание отчета.

1. Титульный лист;
2. Цель работы;
3. Вариант задания на работу;
4. Таблица истинности заданной булевой функции;
5. Схема, реализующая заданную функцию;
6. Таблица истинности для отлаженной схемы;
7. Схема отлаженного преобразователя кодов;
8. Таблица истинности (исходная и итоговая);
9. Скриншоты схемы с заданными кодовыми состояниями;
10. Выводы.

ПРАКТИКУМ

по дисциплине

ЦИФРОВАЯ СХЕМОТЕХНИКА

РАЗРАБОТКА И МОДЕЛИРОВАНИЕ ЦИФРОВЫХ БЛОКОВ ЭЛЕКТРОННОЙ АППАРАТУРЫ

Для студентов ЦЗОПБ

Направление подготовки: 11.03.02

Подписано в печать . .2023г. Формат 60х90 1/16.

Объём усл.п.л. Изд. № .
